Воронежский государственный университет Факультет
прикладной математики, информатики и механики

Математика, информационные технологии, приложения

*Межвузовская научная конференция
молодых ученых и студентов*

Воронеж,
24-25 апреля 2024 г.

Воронеж
Издательско-полиграфический центр
«Научная книга»
2024

УДК 531(063)+51-7(063)
ББК 22.2я5+22.1я5
М34

Математика, информационные технологии, приложения : сборник трудов Межвузовской научной конференции молодых ученых и студентов, Воронеж, 24-25 апреля 2024 г. –
М34
Воронеж : Научная книга, 2024.–1058 с.
ISBN 978-5-4446-****-*

Традиционно конференция является междисциплинарной. Важнейшая ее цель – ознакомление молодых специалистов с новыми веяниями в современной науке, а также обмен знаниями и достижениями в предложенных научных направлениях.

УДК 531(063)+51-7(063)
ББК 22.2я5+22.1я5

ISBN 978-5-4446-****-*

УДК : 004.056.53

ЗАЩИТА В ЦИФРОВОМ МИРЕ: КЛЮЧЕВЫЕ АСПЕКТЫ БЕЗОПАСНОСТИ ИНТЕРНЕТ-СЕТЕЙ И УСТРОЙСТВ

Г.С. Абдулаев

Воронежский государственный университет

Введение

В нашем современном мире, где цифровые технологии проникают во все сферы нашей жизни, безопасность интернет-сетей и устройств становится критически важной. Каждый день миллионы людей и организаций сталкиваются с угрозами в сети, начиная от вирусов и хакерских атак, заканчивая фишингом и мошенничеством. В этой статье мы рассмотрим ключевые аспекты безопасности интернет-сетей и устройств, а также предложим практические советы по защите вашей онлайн-жизни.

1.Основные угрозы

Первый шаг к обеспечению безопасности вашей интернет-сети и устройств - это понимание основных угроз. Вирусы, вредоносные программы, хакерские атаки, фишинг - это только некоторые из них. Вирусы и вредоносные программы могут заразить ваше устройство и украсть ваши личные данные, а хакерские атаки могут привести к утечке конфиденциальной информации или даже к полной блокировке системы.

1.1. Вирусы и вредоносные программы

Это одна из наиболее распространенных угроз в сфере кибербезопасности. Вирусы и вредоносные программы могут заразить ваше устройство через вредоносные веб-сайты, электронную почту, загрузки файлов и другие источники. Они могут привести к утрате данных, краже личной информации, вымогательству или даже полной блокировке работы устройства.

1.2. Хакерские атаки

Хакерские атаки могут включать в себя различные методы, такие как атаки на слабые пароли, взлом аккаунтов, атаки отказом в обслуживании (DDoS) и многие другие. Хакеры могут использовать уязвимости в сетевых протоколах или программном обеспечении, чтобы получить несанкционированный доступ к вашим данным или сети.

1.3. Фишинг и социальная инженерия

Фишинг - это метод мошенничества, при котором злоумышленники выдают себя за доверенные организации или лица с целью обмана пользователей и получения их конфиденциальной информации, такой как пароли или данные банковских карт. Социальная инженерия включает в себя манипуляции и обман людей с целью получения доступа к системам или информации.

1.4. Утечка данных

Утечка данных может произойти из-за взлома системы, несанкционированного доступа к базам данных или утраты устройства с хранящейся на нем чувствительной информацией. Это

может привести к серьезным последствиям, таким как утечка личных данных пользователей, финансовые потери и повреждение репутации организации.

1.5. Кибершпионаж и кибершантаж

Кибершпионаж представляет собой проникновение в сети и устройства с целью получения конфиденциальной информации, такой как коммерческие секреты, планы разработок или государственные секреты. Кибершантаж включает в себя угрозы или давление на жертву для получения денежных средств или других выгод.

1.6. Уязвимости интернет-протоколов и программного обеспечения

Уязвимости в интернет-протоколах и программном обеспечении могут быть использованы злоумышленниками для проведения различных видов атак, включая внедрение в систему, исполнение кода или кражу данных. Регулярное обновление программного обеспечения и применение патчей безопасности помогают уменьшить риски, связанные с этими уязвимостями. Вот несколько типов уязвимостей, которые могут быть обнаружены в интернет-протоколах и программном обеспечении:

Атака на буфер переполнения: злоумышленники могут эксплуатировать уязвимости буфера переполнения в программном обеспечении компании, чтобы выполнить вредоносный код на сервере или рабочей станции. Например, в 2003 году атака червями Blaster и Sasser использовала уязвимость буфера переполнения в операционной системе Windows для распространения вредоносного кода и инфицирования множества компьютеров.

SQL-инъекции: это тип атаки, при котором злоумышленники внедряют SQL-запросы в веб-приложения, которые взаимодействуют с базой данных. Если веб-приложение не надежно обрабатывает ввод данных пользователя, злоумышленники могут получить несанкционированный доступ к базе данных и красть или изменять конфиденциальную информацию. Например, в 2017 году атака WannaCry использовала уязвимость, связанную с SQL-инъекцией, для распространения вредоносного кода и шифрования файлов на компьютерах во всем мире.

Атака на службу DNS: злоумышленники могут использовать уязвимости в протоколе DNS (Domain Name System) для перенаправления пользователей на фальшивые веб-сайты или для атаки на сетевую инфраструктуру компании. Например, атака DDoS на службу DNS Dyn в 2016 году привела к временному отключению многих крупных веб-сайтов, таких как Twitter, Netflix и Amazon, из-за перегрузки службы DNS.

Уязвимости веб-приложений: злоумышленники могут использовать уязвимости в веб-приложениях компании для взлома системы, кражи данных или распространения вредоносного кода.Например, в 2013 году атака на Target Corporation использовала уязвимость в веб-приложении для получения доступа к данным о кредитных картах более 40 миллионов клиентов.Это одна из самых крупных утечек данных в истории розничной торговли. Злоумышленники получили доступ к системе Target через учетные данные поставщика, которые были скомпрометированы и использованы для входа в систему. После этого они установили вредоносное программное обеспечение на серверы Target, с помощью которого собирали и копировали данные кредитных карт при каждой транзакции, совершаемой в магазинах Target.

Подделка IP-адреса (IP Spoofing): этот метод заключается в том, чтобы изменить исходный IP-адрес отправителя, чтобы сделать трафик похожим на легитимный. Например, злоумышленник может подделать IP-адрес в пакетах данных таким образом, чтобы они казались отправленными изнутри доверенной сети, и таким образом обойти правила, установленные на межсетевом экране.Например атака на сервер игрового хостинга Quake III Arena.В 2012 году хакеры осуществили атаку на сервер Quake III Arena, используя метод

подделки IP-адреса. Они подделали IP-адреса исходящего трафика, чтобы казалось, что он исходит от доверенных игровых клиентов. Затем они отправили большое количество фальшивых запросов на сервер, перегрузив его и приведя к отказу в обслуживании (DoS).

Атака с использованием фрагментации пакетов (Packet Fragmentation): при фрагментации пакетов данные разбиваются на более мелкие фрагменты для передачи по сети. Злоумышленник может использовать этот метод для создания пакетов с нежелательными заголовками, которые обманывают межсетевой экран и позволяют проникнуть в защищенную сеть.Например атака Teardrop. Атака Teardrop была одной из первых известных атак с использованием фрагментации пакетов. Она была направлена на операционные системы Windows 95 и Windows NT, которые не могли правильно обрабатывать фрагментированные IP-пакеты. Злоумышленники отправляли специально сформированные фрагментированные пакеты на уязвимую систему, вызывая ее аварийное завершение работы.

Служебные протоколы (Tunneling): например использование протокола ICMP для скрытия туннелей. Злоумышленники могут использовать ICMP-туннелирование для передачи данных через сеть, обходя правила межсетевого экрана. Например, они могут создать туннель через ICMP Echo Request/Reply (ping) и использовать его для передачи вредоносных данных или обхода ограничений, наложенных на другие протоколы.

2. Методы защиты интернет-сетей

Существует множество методов защиты интернет-сетей, которые могут помочь вам обезопасить вашу сеть от угроз. Один из самых распространенных методов - использование межсетевых экранов (firewalls), которые могут блокировать нежелательный трафик. Кроме того, важно использовать VPN-сервисы для шифрования вашего интернет-трафика и механизмы аутентификации, чтобы предотвратить несанкционированный доступ к вашей сети.

2.1 Использование межсетевых экранов (firewalls)

Межсетевые экраны (firewalls) являются первой линией защиты сети от несанкционированного доступа извне. Они могут фильтровать трафик, блокировать подозрительные соединения и предотвращать атаки, такие как DDoS (атаки отказом в обслуживании).

2.2 VPN-сервисы (Virtual Private Networks)

VPN-сервисы обеспечивают шифрование вашего интернет-трафика и создают защищенное соединение между вашим устройством и удаленной сетью. Это особенно важно при использовании общедоступных Wi-Fi сетей, где ваш трафик может быть подвержен перехвату.

2.3 Механизмы аутентификации и авторизации

Использование сильных методов аутентификации, таких как многофакторная аутентификация, помогает предотвратить несанкционированный доступ к сети и ресурсам. Только авторизованные пользователи должны иметь доступ к конфиденциальной информации и критическим системам.

2.4 Системы обнаружения и предотвращения вторжений (IDS/IPS)

Системы обнаружения и предотвращения вторжений (IDS/IPS) являются важным компонентом безопасности сетей. Они работают на уровне сетевого трафика, анализируя его на наличие подозрительной активности и реагируя на угрозы в реальном времени.

Системы обнаружения вторжений (IDS): IDS предназначены для мониторинга сетевого трафика и обнаружения аномальных или вредоносных действий. Они используют различные

методы обнаружения, такие как сигнатурное обнаружение (сравнение с предопределенными шаблонами) и анализ поведения (выявление аномалий в обычном сетевом трафике). Когда IDS обнаруживает потенциальную угрозу, он генерирует предупреждение или уведомление, которое может быть передано администратору сети для принятия дальнейших мер.

Системы предотвращения вторжений (IPS): IPS расширяют функциональность IDS, позволяя не только обнаруживать угрозы, но и автоматически блокировать или предотвращать их. IPS может действовать на уровне сетевых устройств, маршрутизаторов или файрволов, блокируя подозрительный трафик или атакующие пакеты данных до того, как они достигнут целевых систем.

2.5 Обновление программного обеспечения

Регулярное обновление операционных систем, прикладного программного обеспечения и антивирусных баз данных помогает устранять уязвимости и обеспечивать защиту от последних угроз.

2.6 Сегментация сети

Сегментация сети представляет собой разделение сети на отдельные сегменты или зоны с целью уменьшения атаков и ограничения распространения угроз в случае компрометации. Вот несколько ключевых аспектов сегментации сети:

Физическая сегментация: это разделение сети на физически отдельные сегменты с помощью различных сетевых устройств, таких как маршрутизаторы или коммутаторы. Каждый сегмент может иметь свою собственную локальную сеть (LAN) и физически разделенные сетевые устройства.

Логическая сегментация: это разделение сети на логические сегменты с помощью виртуальных локальных сетей (VLAN) или виртуальных частных сетей (VPN). Логическая сегментация позволяет организовывать группы устройств по функциональности или уровню безопасности, не зависимо от их физического расположения.

Сегментация по уровням доступа: это определение различных уровней доступа к сети и ресурсам в зависимости от роли или полномочий пользователей или устройств. Например, сегментация по уровням доступа позволяет разделить сеть на зоны для административного доступа, рабочих групп, гостевого доступа и т. д.

Сегментация сети позволяет ограничить возможность распространения атак на всю сеть и минимизировать воздействие компрометации на отдельные сегменты. Это важный элемент стратегии защиты сети, который помогает улучшить ее общую безопасность.

2.7 Шифрование данных

Использование шифрования данных в покое и в движении обеспечивает дополнительный уровень защиты для конфиденциальной информации, передаваемой по сети.

2.8 Резервное копирование данных

Регулярное создание резервных копий данных позволяет быстро восстановить работоспособность системы в случае успешной атаки или сбоя оборудования.

2.9 Обучение персонала

Обучение сотрудников основам кибербезопасности, таким как создание надежных паролей, распознавание фишинговых атак и правила использования общего сетевого ресурса, также является важным компонентом защиты интернет-сетей.

Заключение

В заключение, важно отметить, что безопасность сетей и защита от кибератак являются непрерывным процессом, который требует внимания и усилий со стороны организаций и индивидуальных пользователей. Атаки на сети постоянно эволюционируют, а злоумышленники постоянно ищут новые методы обхода защиты. В этом контексте важно применять все доступные средства и технологии для обеспечения безопасности сети.

Поддержание обновленного программного обеспечения, регулярное обучение персонала по вопросам кибербезопасности, использование механизмов защиты, таких как межсетевые экраны, системы обнаружения и предотвращения вторжений (IDS/IPS), а также внедрение строгих политик безопасности - все это является ключевыми элементами для обеспечения безопасности сети.

Кроме того, важно осознавать, что безопасность сетей - это коллективное дело. Сотрудничество между различными организациями, обмен информацией об угрозах и совместное использование лучших практик могут значительно повысить эффективность защиты сетей и обеспечить более надежную защиту от кибератак.

Наконец, нельзя забывать, что сохранение безопасности сетей требует постоянного внимания и адаптации к изменяющимся угрозам. Постоянное обновление знаний, мониторинг сетевой активности и реагирование на инциденты - это основные составляющие успешной стратегии защиты сети в современном цифровом мире.

Литература

1. Асылбеков У.Б. КИБЕРБЕЗОПАСНОСТЬ: защита в цифровом мире: учеб. пособие / У.Б Асылбеков, А. А. Исмаилова. – Алмааты : Изд-во «Бастау», 2021. – 340 с.
2. ГОСТ Р ИСО/МЭК 27002-2021 Информационные технологии. Методы и средства обеспечения безопасности. Свод норм и правил применения мер обеспечения информационной безопасности. – Введ. 30.11.2021 – Москва
3. Информационная безопасность, защита данных. – Режим доступа: https://habr.com/ru/hubs/infosecurity

УДК *004.62*

ПЕРСПЕКТИВЫ ИНТЕГРАЦИИ ОТЕЧЕСТВЕННЫХ КРИПТОАЛГОРИТМОВ В ПРОТОКОЛ TRANSPORT LAYER SECURITY (TLS)

В.В. Авраменко

Воронежский государственный университет

Введение

Тема данной статьи - перспективы интеграции отечественных криптоалгоритмов в протокол Transport Layer Security (TLS). Особое внимание уделим изучению возможности реализации протокола TLS с отечественными криптоалгоритмами и рассмотрению процесса интеграции отечественных криптоалгоритмов в протокол TLS, включая Магму и Кузнечик, а также обсудим ключевые аспекты этого процесса.

Протокол Transport Layer Security (TLS) является основным стандартом для защищенной передачи данных в интернете. Он обеспечивает шифрование, аутентификацию и целостность данных, обеспечивая тем самым конфиденциальность и безопасность коммуникаций между клиентом и сервером. Однако в последнее время возникает интерес к интеграции отечественных криптоалгоритмов в TLS для обеспечения национальной безопасности и независимости в области криптографии. В рамках данной темы мы рассмотрим этот процесс, включая результаты и уже внедренные изменения в TLS 1.2.

1. TLS

1.1. Что такое и для чего нужен

TLS (Transport Layer Security — протокол защиты транспортного уровня), как и его предшественник SSL (Secure Sockets Layer — слой защищённых сокетов), — криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет. TLS и SSL используют асимметричное шифрование для аутентификации, симметричное шифрование для конфиденциальности и коды аутентичности сообщений для сохранения целостности сообщений.

Данный протокол широко используется в приложениях, работающих с сетью Интернет, таких как веб-браузеры, работа с электронной почтой, обмен мгновенными сообщениями и IP-телефония (VoIP).

Протокол TLS предназначен для предоставления трёх услуг всем приложениям, работающим над ним, а именно: шифрование, аутентификацию и целостность. Технически, не все три могут использоваться, однако на практике, для обеспечения безопасности, как правило используются все три.

Для того чтобы установить криптографически безопасный канал данных, узлы соединения должны согласовать используемые методы шифрования и ключи. Протокол TLS однозначно определяет данную процедуру. Следует отметить, что TLS использует криптографию с открытым ключом, которая позволяет узлам установить общий секретный ключ шифрования без каких-либо предварительных знаний друг о друге.

2. Криптоалгоритмы

2.1. Кузнечик и Магма

Кузнечик и Магма — это два различных российских криптографических алгоритма, используемых для шифрования данных. Вот основные отличия между ними:

1. Тип алгоритма.

Кузнечик: является симметричным блочным шифром. Он использует ключ длиной 256 бит и работает с блоками данных размером 128 бит.

Магма (GOST 28147-89): также является симметричным блочным шифром. Он работает с блоками данных размером 64 бит и ключом длиной 256 бит.

2. Стойкость к атакам.

Кузнечик: обеспечивает более высокий уровень стойкости к криптоаналитическим атакам благодаря большей длине блока данных и ключа.

Магма: имеет более низкий уровень стойкости в сравнении с Кузнечиком из-за меньшей длины блока данных и ключа.

3. Стандартизация.

Кузнечик: стандартизирован в России и рекомендуется для использования в различных сферах, включая шифрование данных и защиту информации.

Магма: также стандартизирован в России и использовался в различных государственных стандартах, но может быть менее распространенным на международном уровне.

2.2. Реализация и модификация

Первым шагом в интеграции отечественных криптоалгоритмов в TLS является их реализация в коде протокола, т.е. необходимо разработать код, который реализует российские криптоалгоритмы, такие как ГОСТ-алгоритмы, Кузнечик и другие, в соответствии с спецификациями TLS. Это включает разработку функций для шифрования, хэширования и аутентификации, а также интеграцию этих функций в процесс обмена данными в рамках TLS.

Для поддержки новых криптоалгоритмов необходимо модифицировать протокол TLS. Важно изменить протокол TLS таким образом, чтобы он мог поддерживать новые криптоалгоритмы. Это включает добавление соответствующих идентификаторов алгоритмов в список поддерживаемых алгоритмов в TLS Handshake Protocol, а также обновление других частей протокола для совместимости с новыми алгоритмами.

2.3. Применение

Важным этапом в интеграции отечественных криптоалгоритмов в протокол TLS было успешное внедрение алгоритмов Кузнечик и Магма. Это произошло благодаря работе специалистов из отечественных криптографических компаний и стандартизационных органов.

Кузнечик был интегрирован в качестве симметричного шифра, а Магма (GOST 28147-89) был включен в список криптографических алгоритмов, поддерживаемых в TLS 1.2.

Это значимое событие означает, что теперь TLS 1.2 поддерживает использование российских криптографических алгоритмов, что в свою очередь способствует увеличению уровня защиты в сетевых коммуникациях, особенно в контексте национальной безопасности.

В дальнейшем планируется продолжение работ по интеграции отечественных криптоалгоритмов в более новые версии протокола TLS, такие как TLS 1.3, а также расширение списка поддерживаемых алгоритмов для обеспечения большей гибкости и совместимости.

Заключение

Интеграция отечественных криптоалгоритмов в протокол TLS является важным шагом для обеспечения безопасности сетевых коммуникаций. Успешное внедрение алгоритмов Кузнечик и Магма в TLS 1.2 открывает новые возможности для использования российских криптографических технологий на мировой арене и способствует национальной безопасности.

Литература

1. Что такое TLS. – Режим доступа: https://habr.com/ru/articles/258285/
2. Стандартизирован TLS 1.2 с «Кузнечиком» и «Магмой». – Режим доступа: https://www.cryptopro.ru/news/2018/08/standartizirovan-tls-12-s-kuznechikom-i-magmoi

УДК 004.451

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ РЕБЕНКА НАВЫКАМ ПО МЕТОДИКЕ VB-MAPP

С. В. Агеева, И. В. Замятин

Воронежский государственный университет

Введение

В наше время все чаще у детей обнаруживается ряд отклонений в развитии, объединяемых под термином расстройства аутистического спектра (далее – РАС).

Данное расстройство характеризуется дефицитом в социальных взаимодействиях и коммуникациях. Детям, страдающим РАС, свойственны тревожность, страхи, погруженность в себя. Эти состояния могут сочетаться с агрессией, самоагрессией, с негативной реакцией на любые изменения привычного образа жизни. Таким образом, данное расстройство не позволяет детям с РАС развиваться в том же темпе, что и их здоровые сверстники.

На сегодняшний день разработано множество методов терапии, направленных на улучшение качества жизни детей и обеспечение им возможности социализации, обучения и освоения навыков самостоятельной жизни. Для этого важно, чтобы у педагогов была возможность проводить занятия на основании различных методик. Однако постоянный поиск материалов и формирование заданий для обучения требует большого количества времени и сил, которые можно было бы направить на непосредственное взаимодействие с ребенком. Одним из вариантов решения данной проблемы является создание мобильного приложения, которое поможет специалистам обучать ребенка необходимым знаниям и навыкам.

Целью данной работы является разработка прототипа мобильного приложения под Android для обучения ребенка навыкам, которые проверяется при проведении методики VB-MAPP.

1. Технические требования

На первом этапе работы необходимо было собрать требования от заказчика и изучить программу VB-MAPP MAPP (Verbal Behavior Milestones Assessment and Placement Program).

VB-MAPP представляет собой программу оценки навыков речи и социального взаимодействия для детей с аутизмом и другими нарушениями. Существует 5 компонентов VB-MAPP [1]:

1. Оценка вех развития;
2. Оценка преград для обучения;
3. Оценка переходов;
4. Анализ заданий и мониторинг приобретения навыков;
5. Рекомендации по построению индивидуальной программы обучения.

В данной работе рассматривается первый компонент «Оценка вех развития» и его подкомпонент «Поведение слушателя».

Данный компонент разделен на 3 условных уровня возрастного развития. При разработке мобильного приложения используется ряд заданий (критериев), которые описаны в данной методике:

- 0-18 мес.
    - Правильно выбирает предмет в наборе из четырех, для 20 различных предметов или картинок (например, «Покажи кота»).
- 18-30 месяцев.
    - Правильно выбирает предмет в наборе из 6, для 40 различных предметов или картинок (например, «Найди банан»).
    - Обобщает навыки слушателя, выбирая предмет из набора, содержащего 8 расположенных в случайном порядке предметов. Ребенок может различать 50 предметов, представленных в трех вариантах (например, может найти поезда трех различных видов).
    - Выполняет 50 двухкомпонентных инструкций (формата существительное-глагол и/или глагол-существительное) (например, покажи мне, где мальчик ест).
    - Правильно выбирает названный предмет в книжке, на сюжетной картинке или в естественной среде для 250 предметов.
- 30-48 месяцев.
    - Выбирает предметы по цвету и форме в наборе из 6 подобных стимулов, для 4 цветов и 4 форм (например, «Покажи красную машину»).
    - Выбирает предметы из набора подобных стимулов на основании 4 пар качественных прилагательных (например, большой-маленький) и выполняет действия по инструкции, содержащей 4 пары наречий, образованных от качественных прилагательных (например, тихо-громко).

Далее в рамках работы были сформулированы миссия приложения, окружение системы, описаны роли, структура базы данных, диаграммы использования и прецеденты. Рассмотрим некоторые из них.

Например, миссия приложения сформулирована следующим образом: «Облегчить и оптимизировать процесс работы педагогов, связанный с поиском, печатью и организацией картинок, предоставляя им удобный инструмент для хранения и поиска нужных изображений, а также возможность создания заданий на основе базы данных картинок».

Описание функциональности приложения: «Приложение должно обеспечивать проведение оценки навыков детей по методике VB-MAPP. Оно должно быть интуитивно понятным и простым в использовании, чтобы специалисты могли легко пользоваться им».

Далее представлены несколько требований к окружению системы:

- Приложение должно быть совместимо с операционной системой Android.
- Приложение должно быть совместимо с размером планшета 1280 x 800 пикселей (DEXP C38 Kid’s).

Было выделено 3 роли: Администратор (педагог, авторизованный в приложении), Пользователь (педагог без авторизации в приложении), Ребенок.

Для администратора были описаны следующие требования:

- Авторизация: Педагог может войти в систему с использованием уникального имени пользователя и пароля.
- Управление базой данных (авторизованный доступ): Педагог имеет доступ к функционалу обновления, добавления и удаления данных в базе приложения.
- Формирование заданий: Педагог может генерировать новые задания.

Диаграмма вариантов использования выглядит следующим образом (рис. 1):

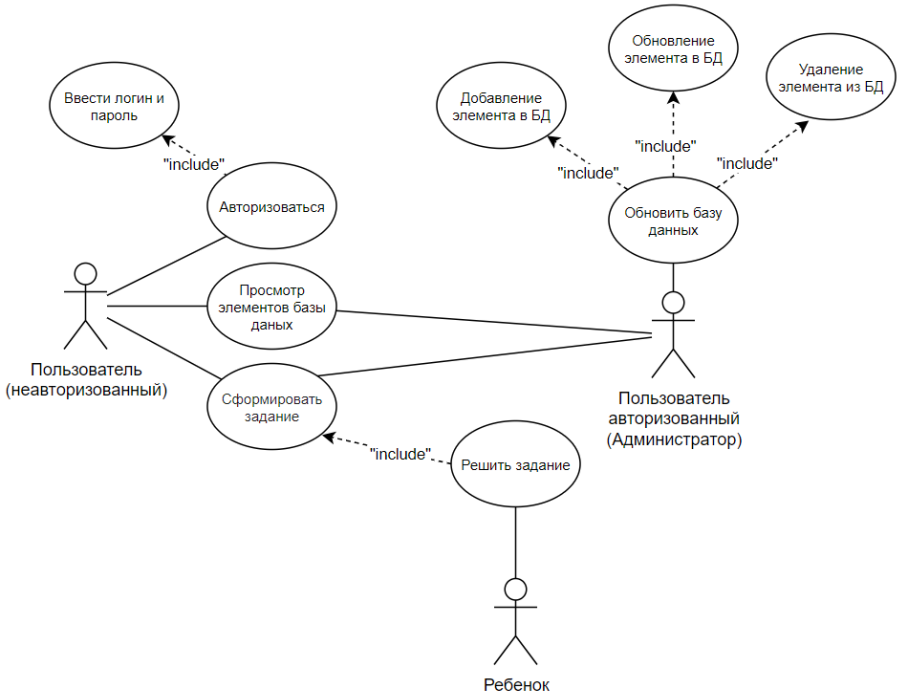


Рис. 1. Диаграмма вариантов использования

Прецедент – Формирование одного задания

Действующее лицо: Педагог (Пользователь или администратор).

Цель: Сформировать задание.

Предусловия: База данных с картинками существует, доступна и заполнена элементом, который требуется для формирования задания.

Главная последовательность:

1. Педагог заходит в систему и открывает БД.
2. Педагог нажимает "Выбрать задание".
3. Система отображает страницу с выпадающем списком доступных категорий и характеристик картинок.
4. Педагог выбирает необходимую категорию и / или характеристику картинки.
5. Система формирует задание и отображает его на экране.
6. Педагог показывает ребенку задание.
7. Ребенок выбирает правильный ответ и озвучивает педагогу.
8. Педагог нажимает «Выход».

Альтернативная последовательность (база данных недоступна):

1. Педагог заходит в систему и открывает БД.
2. Педагог нажимает "Выбрать задание".
3. Система отображает страницу с выпадающем списком доступных категорий и характеристик картинок.
4. Педагог выбирает необходимую категорию и / или характеристику картинки.
5. Система выдает ошибку во всплывающем окне.
6. Педагог закрывает всплывающее окно и пробует заново.

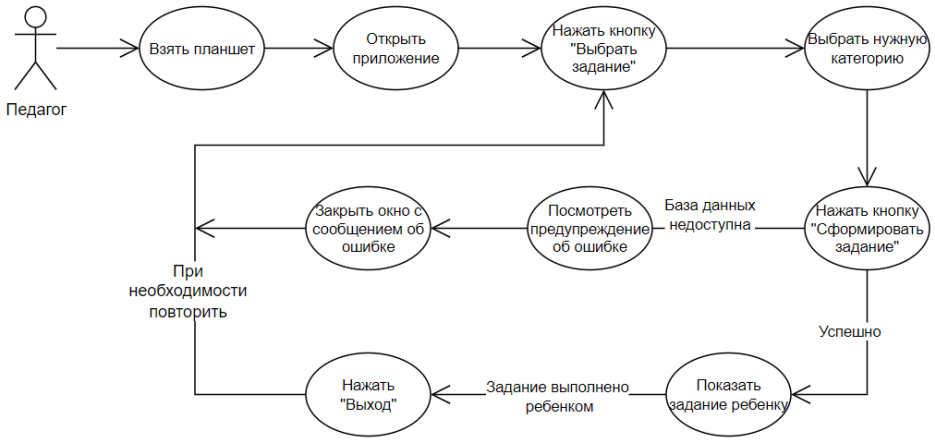


Рис. 2. Формирование задания

База данных картинок содержит следующие поля:

1. picture_id – Уникальный номер картинки
2. picture – Картинка, закодированная в формат base64.
3. category – Категория
4. subCategory – Подкатегория
5. pictureName – Название картинки
6. color – Цвет
7. size – Размер предмета
8. shape – Форма
9. actionOn – Действия с предметом (играть, есть)
10. action – Что делает предмет (катается, пищит)
11. qualitativeAdjective – Качество (теплый, мягкий)

2. Пользовательский интерфейс

На главном экране представлены две кнопки: «База данных» и «Выбор игры». На рисунке 3 представлен интерфейс базы данных.

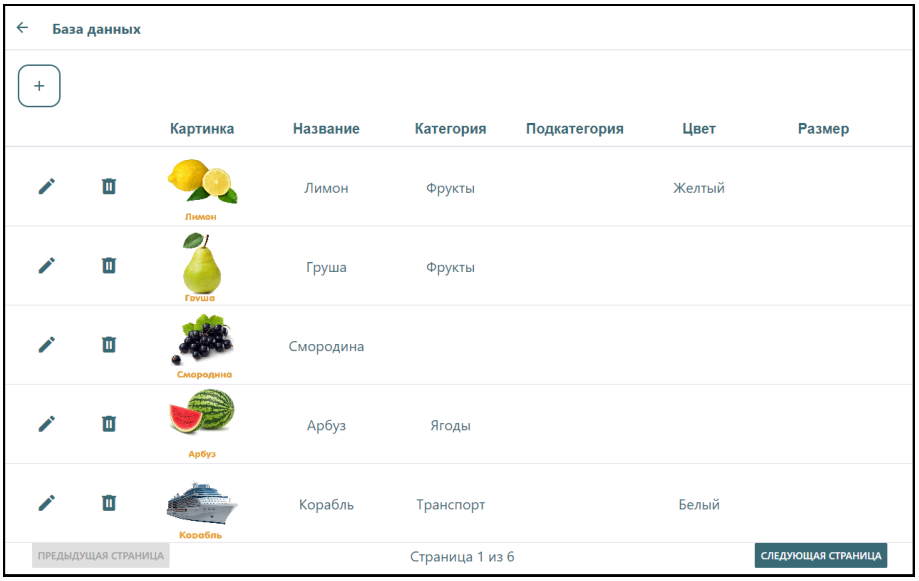


Рис. 3. Пример страницы «База данных»

На данной странице пользователь может посмотреть существующую базу данных. Так как каждая картинка описана 11 характеристиками, то в логику приложения была добавлена возможность прокручивания таблицы влево-вправо. Также была добавлена пагинация для удобного просмотра всех элементов базы данных. В левой части экрана расположены кнопки добавления нового объекта, а также обновления и удаления уже существующего объекта.

На рисунке 4 представлен пример страницы добавления нового элемента базы данных. Пользователь может заполнить поля с описанием объекта, а также добавить необходимую картинку с устройства.

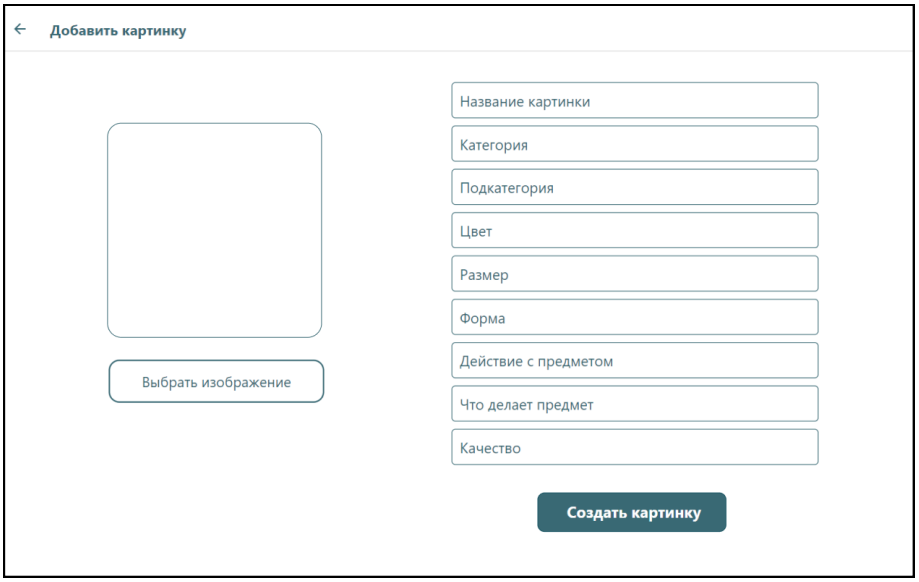


Рис. 4. Пример страницы «Добавить картинку»

Если требуется частично обновить существующий объект, то нажав на «карандашик» на странице с базой данных, откроется следующее окно (рис. 5).

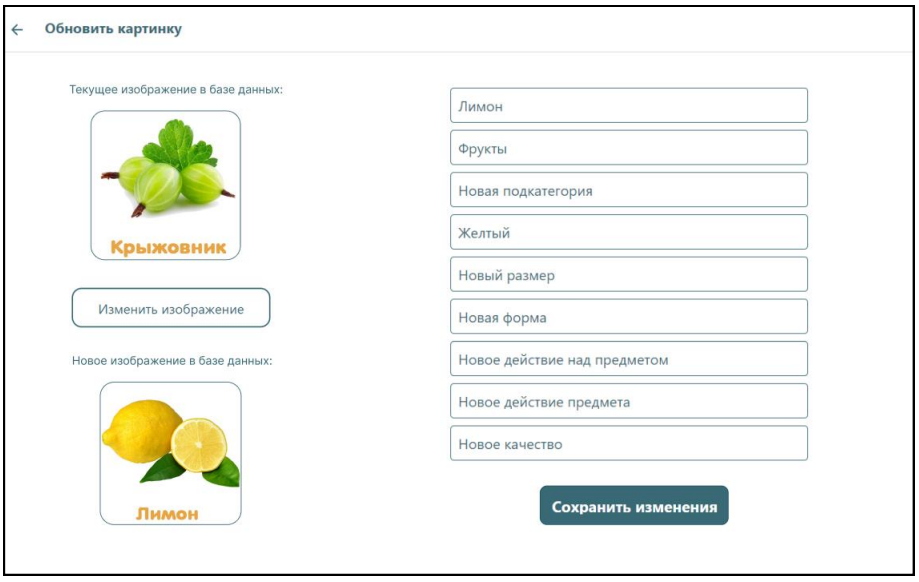


Рис. 5. Пример страницы «Обновить картинку»

Для того, чтобы удалить картинку, пользователю необходимо нажать на значок «корзинка» рядом с выбранной картинкой.

Таким образом, педагог может просматривать базу данных, добавлять в нее новые элементы, а также обновлять и удалять их при необходимости.

Вторая часть функциональности приложения заключается в генерировании специальных заданий. На данный момент разработано две игры: «Покажи…» и «Найди лишнее».

Интерфейс страницы с выбором игры выглядит следующим образом (рис. 6).

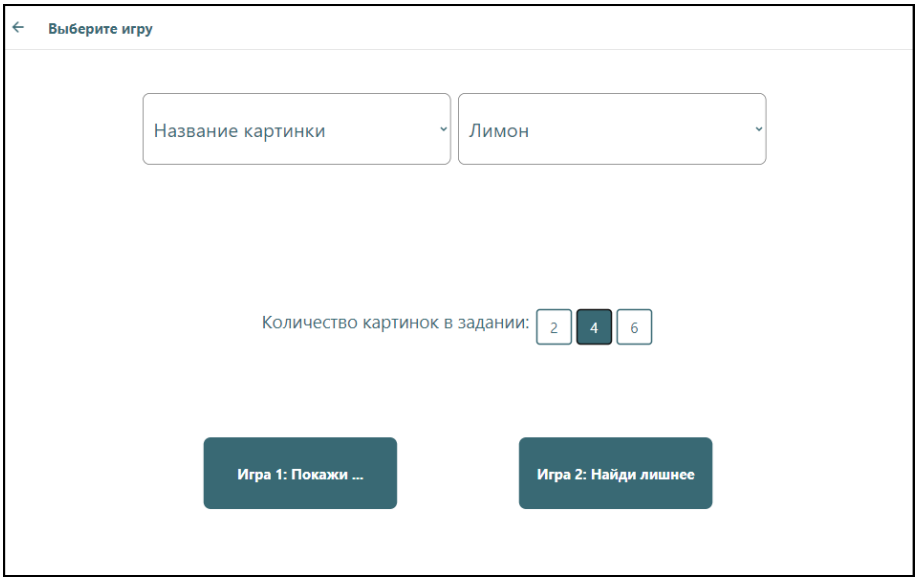


Рис.6. Пример страницы «Выберите игру»

На странице с выбором игры пользователь может выбрать по какой характеристике (названию, цвету, форме) он хочет сформировать задание и из выпадающего списка выбрать конкретное значение. Также педагог может усложнять и упрощать задания в зависимости от того, сколько будет картинок на экране: две, четыре или шесть.

Как только педагог выбрал тему и количество картинок в задании, он выбирает игру и переходит на новую страницу (рис.7).

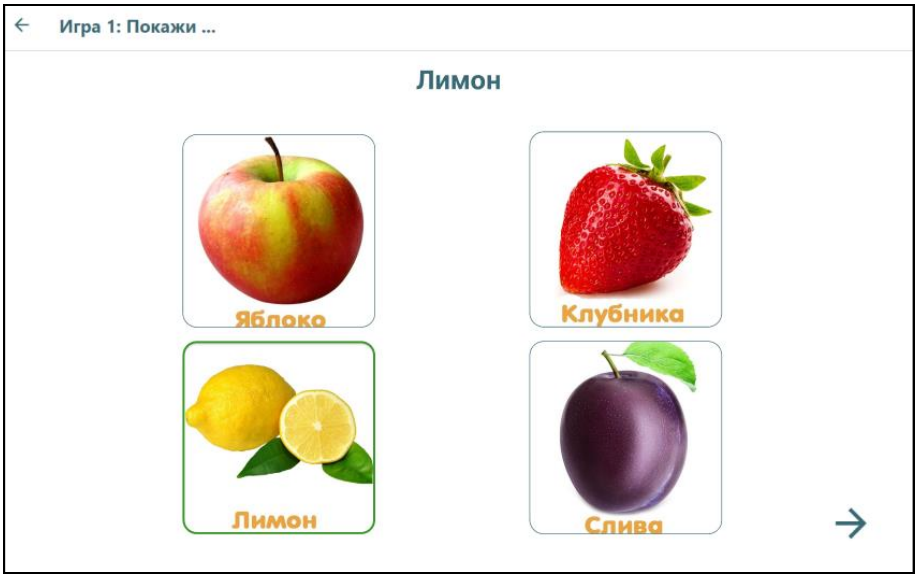


Рис. 7. Пример страницы «Покажи…»

В данном задании ребенку предлагается из четырех картинок выбрать ту, на которой изображен лимон. Если ребенок указывает на правильный ответ, то картинка подсвечивается зеленым цветом. При необходимости педагог может продолжать генерировать задания до тех пор, пока ребенок не научится отличать нужный объект среди других.

Заключение

Таким образом, разработанное мобильное приложение предлагает педагогам сформировать собственную базу данных картинок с их характеристиками и на основании загруженного материала создавать каждый раз новые индивидуальные задания.

Разработанное приложение является прототипом и в дальнейшем будет дополняться новым функционалом, таким как авторизация пользователей, создание личного кабинета для каждого ребенка, элементы поощрения ребенка за правильно выполненные задания, новые задания и т.д.

Литература

1. VB-MAPP: Программа оценки навыков речи и социального взаимодействия для детей с аутизмом и другими нарушениями развития. – Режим доступа: https://autism-frc.ru/early-help/metody/1420. – (Дата обращения: 15.04.2024).

УДК 004.6, 004.9

АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОЦЕССА ФОРМИРОВАНИЯ ОТЧЕТНОСТИ ПО ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ В ВУЗАХ

Л.В. Апалькова

Воронежский государственный университет

Введение

Согласно федеральному закону N 273-ФЗ "Об образовании в Российской Федерации" от 29 декабря 2012 г. организация и осуществление образовательной деятельности по образовательным программам высшего образования (бакалавриата, специалитета и магистратуры) регламентируется федеральным государственным образовательным стандартом (ФГОС). ФГОС задает требования к качеству реализации образовательной программы и, в частности, практической подготовки. Качество реализации практической подготовки с позиции надзорных органов оценивается на основании таких документов как приказы и распоряжения на практику, договоры с организациями, отчеты руководителей практик, ведомости успеваемости студентов. Пакет документов в большинстве вузов принято вести в электронном (цифровом) виде, однако его формирование осуществляется вручную. Имеющаяся автоматизация затрагивает процессы создания учебных планов с практической подготовкой, рабочих программ и ведомостей. Отчетные документы, такие как приказы, распоряжения и все виды отчетов создаются в электронном виде, но вручную. Огромный объем такого документооборота, необходимость его формирования своевременно и в строгом соответствии с регламентами создает большую нагрузку на ППС и административные отделы вуза. В связи с этим, решение задачи по автоматизации данного процесса является актуальным. В данной работе рассматривает разработка алгоритмического обеспечения процесса формирования отчетности по практической подготовке в вузах.

1. Обзор существующих решений и постановка задачи

Существует несколько подходов к решению описанных ранее проблем. Среди них отказ от ведения отчетности, унификация документов и автоматизация работ по формированию пакета документов. Отказ от ведения отчетности является нарушением ФЗ "Об образовании в Российской Федерации" и осуществления образовательной деятельности, поэтому далее не рассматривается. Унификация заключается в установлении единообразия состава и форм документов, фиксирующих осуществление однотипных функций и задач. Использование для автоматизации ПО достаточно сложно для вуза, так как зачастую ПО либо имеют высокую стоимость приобретения и обслуживания и обладает избыточными функциональными возможностями, либо, напротив, обладает слишком малым набором функций, не покрывающим потребности вуза. При этом не все ПО совместимо с ПО, используемым в поддержке образовательного процесса. Более того, ключевая проблема заключается в отсутствии подходящей технологии формирования пакета необходимых документов по требованиям стандартов и с учетом макетов, разрабатываемых вузом самостоятельно, так как организация работы кафедр университета достаточно специфична и не похожа на аналогичный процесс у предприятий, на которые и рассчитано большинство систем электронного оборота документов. Одним из решений данной проблемы является создание программного

обеспечения (ПО) автоматизирующего процессы документооборота вуза для сопровождения программ практической подготовки. Ставится задача: разработать алгоритмическое обеспечение для возможности автоматизации подготовки пакета документов по программам практической подготовки вуза.

2. Пакет документов и источники данных

Пакет документов, формируемый для сопровождения программ практической подготовки и оценки результатов её проведения, реализуемых в Воронежском государственном университете, включает:

1) документы о направлении на практическую подготовку (приказы/распоряжения);
2) договоры с организациями, предоставляющие места для прохождения практик;
3) ведомости успеваемости студентов;
4) отчеты руководителей по практической подготовке (по каждой учебной группе или по руководителю);
5) отчет факультета по всем видам практик об итогах проведения практической подготовки для предоставления в учебно-методическое управление вуза.

Для формирования такого пакета документов входными источниками данных являются: реестр ОПОП, учебные планы, выгрузки из информационной системы вуза о контингенте и распределения нагрузки факультета по ППС, база МТО для прохождения практики.

Приказы и распоряжения готовятся в начале отчетного периода. Договоры с предприятиями заключаются как долгосрочно, так и на период практики (до ее начала). Ведомости успеваемости студентов формируются деканатом и заполняются руководителем в последний день прохождения или день защиты практики. Отчеты руководителей формируются по окончании практики. Отчет факультета формируется в конце отчетного года (периода).

3. Алгоритм формирования пакета документов

Для автоматизации процесса формирования пакета документов необходимо разработать программное обеспечение (ПО), которое позволит автоматически /полуавтоматически такие документы как приказы/распоряжения на практику, отчеты руководителей и факультета. Концептуальная модель ПО представлена на рис. 1.

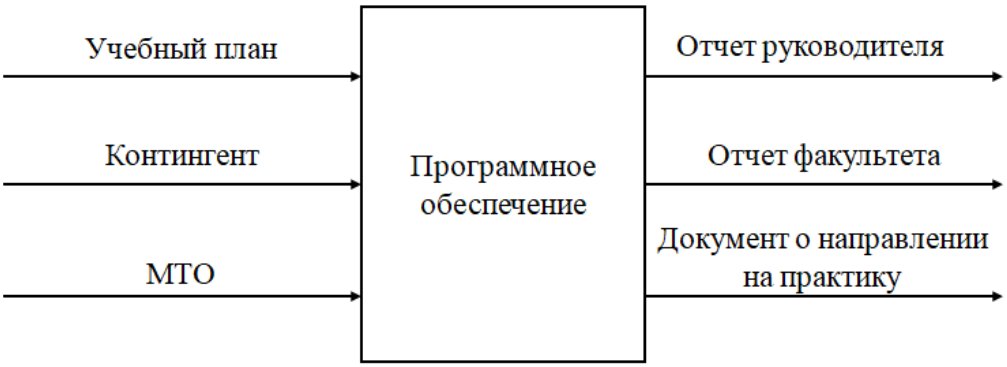


Рис. 1. Концептуальная модель ПО

Алгоритм формирования пакета документов для такого ПО представим следующими шагами.

Шаг 1. Задание периода отчетности. Загрузка реестра реализуемых ОПОП. Формирование списка УП, реализуемых в отчетном периоде.

Шаг 2. Загрузка УП. Формирование перечня практик по направлению (вид, название, курс, семестр) и графика их прохождения (дата начала, дата окончания).

Шаг 3. Запрос информации по назначенным руководителям и ответственным кафедрам.

Шаг 4. Загрузка списка студентов групп (подгрупп) на практику.

Шаг 5. Загрузка мест практики. Формирование перечня мест практик и распределения студентов.

Шаг 6. Формирование свода данных на практическую подготовку в отчетный период.

Шаг 7. Формирование и выгрузка приказа/ распоряжения на практику.

Шаг 8. Загрузка ведомостей. Расчет основных статистик по успеваемости (количество прошедших, получивших оценку «отлично» и т.д.). Занесение информации в свод данных на практическую подготовку в отчетный период.

Шаг 9. Формирование и выгрузка отчетов руководителя. Занесение информации в свод данных на практическую подготовку в отчетный период.

Шаг 10. Формирование и выгрузка отчета по факультету. Занесение информации в свод данных на практическую подготовку в отчетный период.

Модель информационных потоков процесса документооборота вуза по практической подготовке представлена на рис. 2.

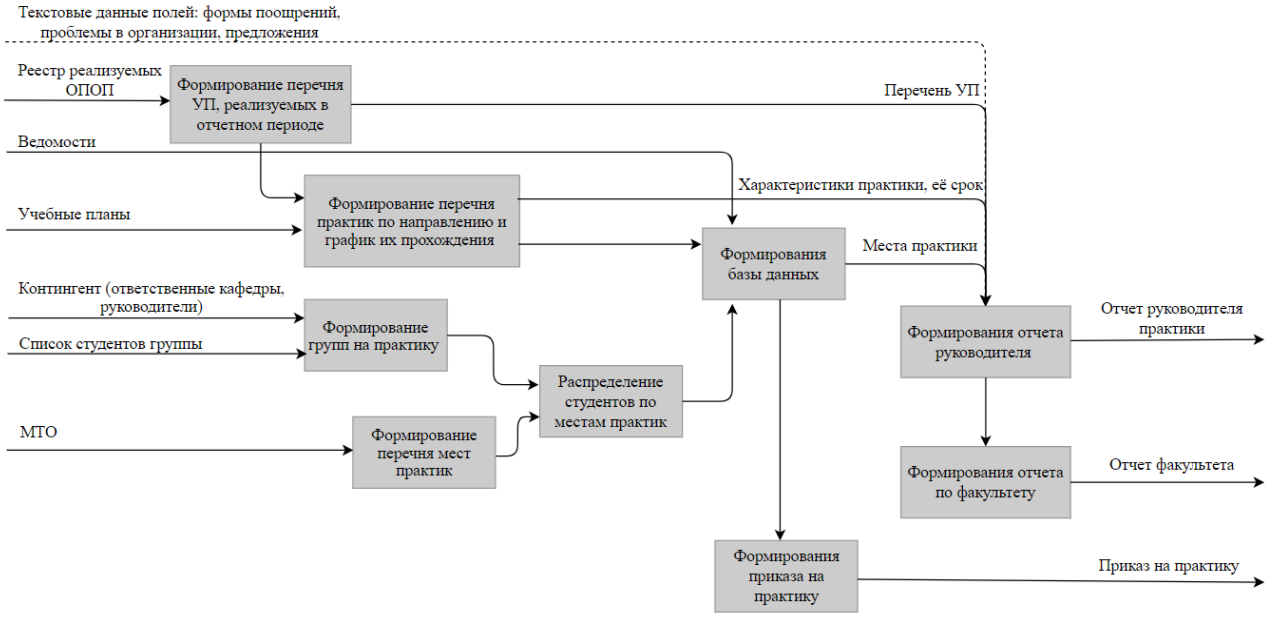


Рис. 2. Модель информационных потоков

При разработке ПО следует учесть необходимость создания внутренней базы данных, которая будет содержать структурированные данные, описывающие взаимосвязи между практиками со всеми их характеристиками, руководителями практической подготовки и данными о распределении студентов в ту или иную организацию.

Заключение

Описанное алгоритмическое обеспечение соответствует требованиям Воронежского

государственного университета в области формирования документов сопровождающих реализацию программ практической подготовки. Среди сложностей его реализации следует отметить отсутствие совместимости ПО вуза, что требует создания дополнительной внутренней базы и не самую оптимальную предварительную обработку информации, получаемую со сторонних систем.

Научный руководитель канд. физ.-мат. наук, доц. кафедры математических методов исследования операций Воронежского государственного университета, Ухлова Вера Владимировна.

Литература

1. Об образовании в Российской Федерации : Федеральный закон Российской Федерации от 29 декабря 2012 г. № 273-ФЗ // Собрание законодательства Российской Федерации. – 2012. – № 53. – Ст. 7598.

2. О практической подготовке обучающихся : Приказ Министерства науки и высшего образования РФ и Министерства просвещения РФ от 5 августа 2020 г. № 885/390 // Министерство науки и высшего образования РФ, Министерство просвещения РФ. – 2020. – № 59778.

3. О порядке организации практической подготовки обучающихся по основным образовательным программам : Инструкция Воронежского государственного университета от 27.11.2020 №10 // ФГБОУ ВО «Воронежский государственный университет». – 2020. – Ст. 2.1.12.

4. Соколов Е. А. Электронный документооборот вуза / С.Н. Середа, Е.А. Соколов // Перспективы развития информационных технологий. – 2012. – № 9. – С. 39-44

5. Чичиль В. О. Разработка автоматизированной системы для организации документооборота невыпускающей кафедры вуза / В. О. Чичиль, И. Ю. Королева. // Молодой ученый. — 2015. — № 23 (103). — С. 74-78.

6. Бесшовная интеграция Microsoft Excel и Word с помощью Python/ - Режим доступа – URL: https://habr.com/ru/companies/skillfactory/articles/553224/

7. Сайт лаборатории ММИС. ПО «Планы» - Режим доступа – URL: https://www.mmis.ru/programs/plany

8. Сайт программного обеспечения «1С». Интеллект Инфо: Образовательные программы. Расширение для 1С:Университет ПРОФ Интеллект Инфо Режим доступа – URL: https://solutions.1c.ru/catalog/intellektinfo_opop?ysclid=liszp1ow5g477654500

УДК 004.056.5

Автоматизация нахождения уязвимостей в компонентах программного обеспечения

К. И. Апасов
Воронежский государственный университет

Введение

В условиях нарастания угроз безопасности информации важным аспектом является вопрос исследования уязвимостей программного обеспечения. Программное обеспечение проникло во все сферы нашей жизни - от финансовой до бытовой, включая большинство устройств в доме. Процессы разработки программного обеспечения предполагают выявление уязвимостей и недекларированных возможностей в исходных кодах программ [1]. Такие уязвимости и недекларированные возможности могут привести к нарушению конфиденциальности и целостности данных, отказу в обслуживании и нарушению функционирования систем в целом.

Причины возникновения уязвимостей многочисленны: это и недочеты в процессе разработки, и ошибки в архитектурных решениях, и использование устаревших или небезопасных библиотек и компонентов [2]. Даже строгая и детальная разработка, соблюдение всех стандартов и применение лучших практик не гарантируют полной защиты от уязвимостей, так как сложность и динамичность современных систем делают их подверженными новым и неожиданным видам угроз. Даже использование сертифицированного П.О. и соблюдение стандартов безопасной разработки (Dev SecOps) не гарантирует безопасность, так как уязвимости существуют и на уровне операционной системы.

Традиционные методы выявления уязвимостей, включая ручной анализ кода, сканирование систем и тестирование на проникновение, долгие годы служили основой безопасности. Однако, с расширением функциональности и сложности программных продуктов, их компонентов и инфраструктурных решений, эти методы становятся все более ресурсоемкими и недостаточно масштабируемыми.

В этом контексте актуальность автоматизации процесса выявления уязвимостей неоспорима. Автоматизированные системы и инструменты способны обеспечивать более широкий и глубокий анализ кода и систем, применять сложные алгоритмы и методы машинного обучения для обнаружения скрытых и сложных уязвимостей, работать с большими объемами данных и быстро адаптироваться к новым угрозам. Эффективная автоматизация позволяет не только сократить время и ресурсы, затрачиваемые на выявление и устранение уязвимостей, но и повысить общий уровень безопасности, обеспечивая надежную защиту для пользователей, корпоративных систем и критической инфраструктуры.

1. Основы безопасности программного обеспечения

1.1. Типы уязвимостей и их классификация

Уязвимости по области происхождения разделяются на следующие классы:

-Уязвимости кода

-Уязвимости конфигурации

-Уязвимости архитектуры

-Организационные уязвимости

-Многофакторные уязвимости

Уязвимости по типам недостатков ИС подразделяются на следующие классы:

- недостатки, связанные с неправильной настройкой параметров ПО.
- недостатки, связанные с возможностью прослеживания пути доступа к каталогам.
- недостатки, связанные с возможностью перехода по ссылкам.
- недостатки, связанные с возможностью внедрения команд ОС.
- недостатки, связанные с межсайтовым скриптингом (выполнением сценариев).
-недостатки, связанные с внедрением интерпретируемых операторов языков программирования или разметки.
- недостатки, связанные с внедрением произвольного кода.
- недостатки, связанные с переполнением буфера памяти.
(Переполнение буфера возникает в случае, когда ПО осуществляет запись данных за пределами выделенного в памяти буфера. Переполнение буфера обычно возникает из-за неправильной работы с данными, полученными извне, и памятью, при отсутствии защиты со стороны среды программирования и ОС. В результате переполнения буфера могут быть испорчены данные, расположенные следом за буфером ипи перед ним. Переполнение буфера может вызывать аварийное завершение или зависание ПО. Отдельные виды переполнений буфера (например, переполнение в стековом кадре) позволяют нарушителю выполнить произвольный код от имени ПО и с правами учетной записи, от которой она выполняется)
- недостатки, связанные с неконтролируемой форматной строкой
- недостатки, связанные с вычислениями.
- недостатки, приводящие к утечке/раскрытию информации ограниченного доступа.
(Утечка информации - преднамеренное или неумышленное разглашение информации ограниченного доступа (например, существуют утечки информации при генерировании ПО сообщения об ошибке, которое содержит сведения ограниченного доступа). Недостатки, приводящие к утечке/раскрытию информации ограниченного доступа, могут возникать вследствие наличия иных ошибок (например, ошибок, связанных с использованием скриптов)
- недостатки, связанные с управлением полномочиями (учетными данными).
- недостатки, связанные с управлением разрешениями, привилегиями и доступом.
- недостатки, связанные с аутентификацией.
- недостатки, связанные с криптографическими преобразованиями (недостатки шифрования).
(К недостаткам, связанным с криптографическими преобразованиями, относятся ошибки хранения информации в незашифрованном виде, ошибки при управлении ключами, использование несертифицированных средств криптографической защиты информации)
- недостатки, приводящие к «состоянию гонки».
(«Состояние гонки» - ошибка проектирования многопоточной системы или приложения, при которой функционирование системы или приложения зависит от порядка выполнения части кода. «Состояние гонки» является специфической ошибкой, проявляющейся в случайные

моменты времени)
-- недостатки, связанные с управлением ресурсами. [2]

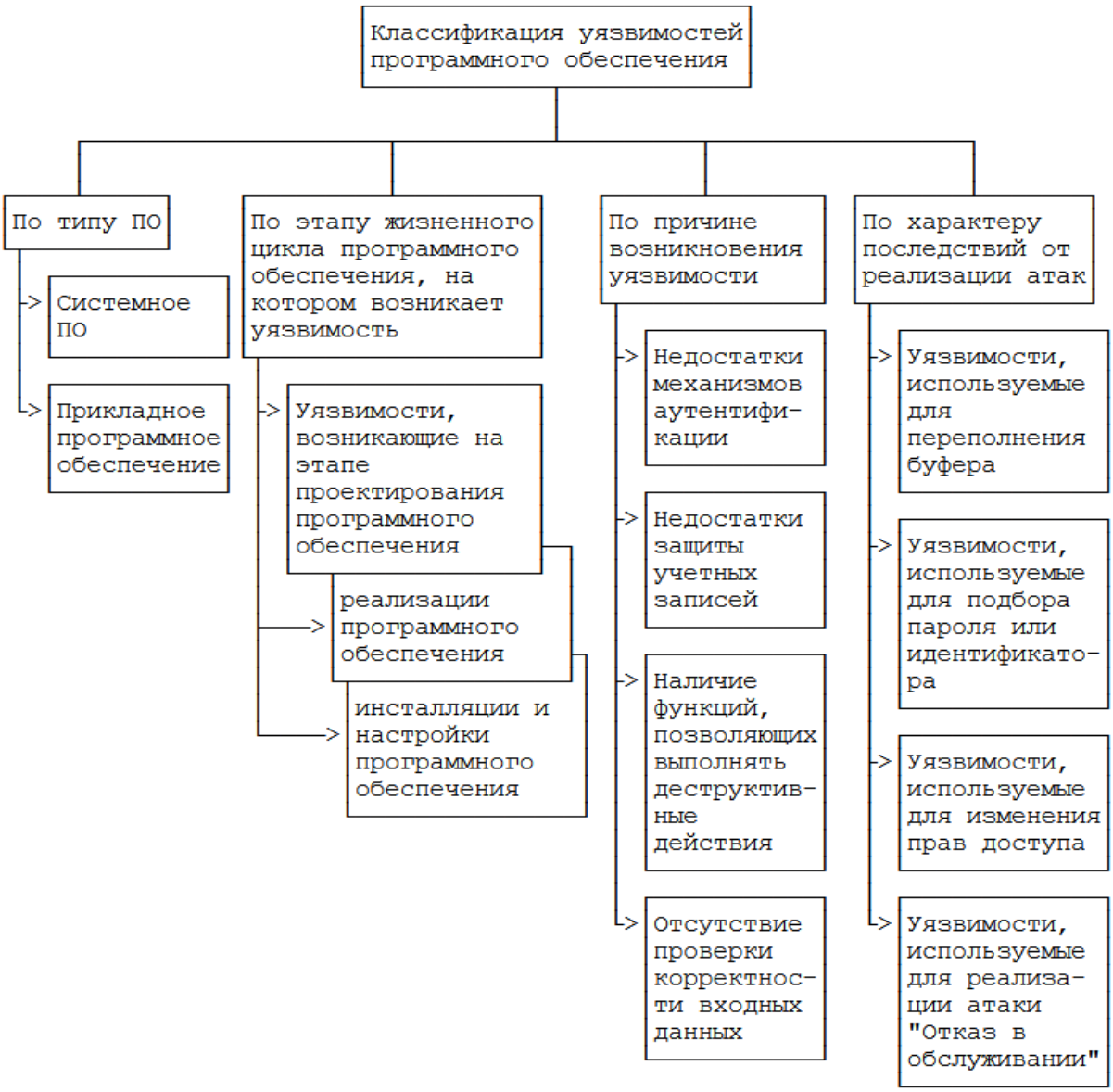


рис.1.Классификация уязвимостей

1.2. Негативные последствия от эксплуатации уязвимостей

- Нарушение тайны переписки, телефонных переговоров и других сообщений
- Финансовый, иной материальный ущерб физического лица
- Нарушение конфиденциальности (утечка) персональных данных
- Нарушение деловой репутации
- Простой информационной системы или сети

- Утечка информации ограниченного доступа [3]

2. Традиционные методы выявления уязвимостей

2.1. Ручной анализ кода

- Статический анализ: Исследование и оценка кода без его выполнения, выявление потенциальных уязвимостей и ошибок на основе анализа структуры и логики программы.
- Динамический анализ: Тестирование программы в реальном времени с использованием различных входных данных для выявления уязвимостей и аномалий в поведении программы.
- Преимущества: Высокая точность при правильной настройке, возможность выявления сложных и специфических уязвимостей.
- Недостатки: Трудоемкость, зависимость от квалификации аналитика, невозможность обнаружения всех видов уязвимостей.

2.2. Сканирование систем

- Сканирование уязвимостей: Процесс автоматического поиска уязвимостей в сетевых службах, операционных системах и приложениях с использованием специализированных инструментов.
- Сканирование веб-приложений: Анализ веб-приложений на наличие уязвимостей, таких как XSS, SQL-инъекции, CSRF и другие.
- Преимущества: Быстрота сканирования, возможность автоматического обновления базы данных уязвимостей.
- Недостатки: Частые ложные срабатывания, неспособность обнаруживать специфические и сложные уязвимости. [5]

2.3. Тестирование на проникновение

- Методологии тестирования: Основные подходы и методики, такие как PTES (Penetration Testing Execution Standard), включая фазы сбора информации, анализа, эксплуатации и отчетности.
- Типы тестирования: Внешние тесты, внутренние тесты, тесты с ограниченным доступом (gray box testing).
- Преимущества: Выявление реальных уязвимостей, оценка реакции системы на атаки, получение рекомендаций по устранению уязвимостей.
- Недостатки: Возможность нарушения работы системы, высокая стоимость и временные затраты, необходимость специализированных навыков.

2.4. Анализ традиционных методов

- Сравнительный анализ: Сравнение эффективности, скорости, стоимости и сложности каждого метода.
- Интеграция и совместимость: Возможность комбинирования различных методов для достижения наилучших результатов.
- Тенденции и перспективы: Развитие традиционных методов, их адаптация к изменяющимся условиям и новым угрозам.

3. Автоматизированные инструменты

3.1. Типы автоматизированных инструментов

- Сканеры уязвимостей: Инструменты для автоматического обнаружения уязвимостей в коде, приложениях и системах.
- Системы статического и динамического анализа: Инструменты, использующие алгоритмы и методы для выявления уязвимостей в коде и при выполнении программы.

3.2. Принципы работы и основные функции

- Алгоритмы сканирования: Основные методы и подходы, используемые для выявления уязвимостей, включая сравнение с базами данных уязвимостей и анализ кода.
- Интеллектуальные технологии: Применение машинного обучения и искусственного интеллекта для улучшения эффективности и точности выявления уязвимостей.
- Отчетность и аналитика: Возможности анализа результатов, создания отчетов и интеграции с другими системами безопасности.

3.3. Преимущества автоматизации

- Скорость и эффективность: Быстрый анализ больших объемов кода и систем, выявление уязвимостей на ранних этапах разработки.
- Масштабируемость: Возможность обработки большого числа проектов и систем, адаптация к изменяющимся условиям и требованиям.
- Снижение ошибок: Уменьшение вероятности пропуска уязвимостей и снижение числа ложных срабатываний. [4]

3.4. Вызовы и ограничения автоматизации

- Ложные срабатывания: Возможность выявления несуществующих уязвимостей из-за ограниченных алгоритмов или специфических особенностей кода.
- Сложность анализа: Трудности в выявлении сложных и специфических уязвимостей, требующих глубокого понимания контекста и логики программы.
- Необходимость обновлений: Постоянное обновление инструментов и баз данных уязвимостей для эффективного обнаружения новых и актуальных угроз.

4. Практический пример успешной автоматизации

- Задача: Обнаружение и устранение уязвимостей в банковских системах для защиты финансовых транзакций и конфиденциальных данных клиентов.

- Решение: 1. Выбор сканера уязвимостей:

-Исследование и анализ доступных на рынке сканеров уязвимостей с учетом требований ФСТЭК России.

-Выбор сканера, поддерживающего российские стандарты безопасности и

имеющего рекомендации от ФСТЭК России.

2. Настройка сканера:

-Настройка сканера, определив параметры сканирования, включая типы уязвимостей, глубину сканирования, частоту запуска сканирований и другие параметры в соответствии с потребностями и требованиями банка.

3. Интеграция с системами мониторинга и управления инцидентами:

- Для оперативного реагирования на обнаруженные уязвимости и сбора отчетности о результатах сканирования сканер был интегрирован с системой мониторинга безопасности и системой управления инцидентами

4. Запуск регулярных сканирований:

-Установлены регулярные запуски сканирований банковских систем согласно заданному расписанию, чтобы обеспечить непрерывное обнаружение и устранение уязвимостей.

5. Автоматическое оповещение о найденных уязвимостях:

После завершения сканирования сканер автоматически отправляет уведомления о найденных уязвимостях ответственным специалистам для принятия мер по их устранению.

6. Проведение регулярных аудитов и анализа результатов:

Команда информационной безопасности регулярно проводит аудиты результатов сканирований, анализирует отчеты о найденных уязвимостях и принимает меры по их устранению

-Результат:

- Снижение числа уязвимостей: Внедрение сканера уязвимостей привело к существенному сокращению числа уязвимостей в банковских системах, повышая уровень безопасности финансовых операций и конфиденциальности данных клиентов.
- Улучшение реакции на угрозы: Благодаря регулярным сканированиям и автоматическому оповещению о найденных уязвимостях, команда информационной безопасности быстро реагирует на потенциальные угрозы и принимает меры по их устранению.
- Укрепление доверия клиентов: Эффективные меры по обеспечению безопасности финансовых данных и операций увеличивают доверие клиентов к банку и способствуют укреплению позиций на рынке

Заключение

В современном информационном обществе обеспечение безопасности программного обеспечения стоит на первом месте, особенно в критически важных секторах, таких как

финансовый. Кейс использования сканера уязвимостей в банковском секторе подтверждает важность автоматизированных инструментов в обнаружении и устранении уязвимостей.

Внедрение автоматизированных сканеров позволяет не только эффективно выявлять уязвимости, но и сокращать время реакции на угрозы, увеличивая уровень безопасности информационных систем. Регулярные сканирования и автоматическое оповещение об уязвимостях обеспечивают оперативное принятие мер по их устранению, минимизируя риски для финансовых операций и конфиденциальности данных клиентов.

Таким образом, автоматизация процесса выявления уязвимостей становится неотъемлемой частью стратегии информационной безопасности, позволяя организациям эффективно защищать свои активы и обеспечивать надежную защиту от современных киберугроз.

Научный руководитель: без уч. звания, кандидат технических наук, доцент кафедры кибербезопасности информационных систем ВГУ, Текунов Василий Васильевич

Литература

1. ГОСТ Р 56939-2016 Разработка безопасного программного обеспечения. Утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 1 июня 2016 г. N 458-ст

2. ГОСТ Р 56546-2015 Защита информации. Уязвимости информационных систем. Классификация уязвимостей информационных систем. Утвержден и введен в действие
Приказом Федерального агентства по техническому регулированию и метрологии от 19 августа 2015 г. №1181-ст

3. Банк данных угроз безопасности информации. Перечень негативных последствий.
– Режим доступа: https://bdu.fstec.ru/threat-section/negatives

4. Приказ ФСТЭК России от 25 декабря 2017 года Об утверждении требований по обеспечению безопасности значимых объектов критической информационной инфраструктуры Российской Федерации.

5. Руководящий документ Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации Классификация по уровню контроля отсутствия недекларированных возможностей
Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации
Классификация по уровню контроля отсутствия недекларированных возможностей
Утверждено решением председателя Государственной технической комиссии при Президенте Российской Федерации от 4 июня 1999 г. N 114

УДК 004.8

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧЕ КЛАССИФИКАЦИИ КОЛЛЕКТИВНЫХ ИДЕЙ

А. В. Астахова

Воронежский государственный университет

Аннотация. В работе представлены результаты классификации идей сотрудников по категориям, которые могут быть выбраны на основе определенных корпоративных процессов в любой компании, при помощи моделей машинного обучения.
Ключевые слова: идея, коллективные идеи, классификация, модели машинного обучения, CatBoost, FastText, градиентный бустинг, категоризация идей.

Введение

Сложно недооценить важность новых идей, которые могут быть предложены сотрудниками компаний. Для этого компании стремятся собирать и формировать идеи сотрудников на внутренних корпоративных ресурсах.

Задача создания и формирования коллективных идей была ранее рассмотрена на примере реализации приложения, которое позволяло организовывать инновационные технологии и инструменты на основе предложений сотрудников внутри компании [4]. Также, была проанализирована возможность генерации и кластеризации новых идей на основе ключевых слов [5].

Данная работа актуальна для задач обработки и классификации идей. Предполагается, что идеи, сформированные на любой информационной платформе внутри компании, могут быть выгружены в определенный формат данных и классифицированы, используя метки класса. Категории класса можно выбрать, основываясь на организации любых внутренних процессов компании. Решение будет рассмотрено на двух моделях машинного обучения.

1. Создание датасета и категоризация данных

Для рассмотрения возможности обработки и классификации идей по меткам класса необходимо сформировать тестовый набор данных (датасет). Предполагается заполнение датасета идеями, которые сотрудники внутри своей компании могли бы предложить для улучшения корпоративных рабочих процессов.

Идеи могут быть созданы на основе известных внутренних процессов компании: оформление, выплаты, карьера, рабочее место, проектная деятельность, технологии, здоровье, социальные сети. Эти процессы будут взяты за основу формирования категории для последующей классификации.

После составления текста идей требуется на основе их семантического содержания присвоить им метку класса. Так, для идеи «Создание комфортной зоны отдыха с мягкими креслами и коврами в офисе на Тверской» можно сопоставить категорию «рабочее место». После присвоения меток на этап обработки текста будут переданы категориальные данные для обучения модели с учителем.

2. Предварительная обработка данных

При создании датасета каждой строке был присвоен текст одной идеи, которой соответствует уникальный идентификатор и ее категория. Сформированный набор данных представлен в формате csv, строки датасета были случайным образом перемешаны (рис. 1).

	idea	category
0	Внедрение системы автоматического логирования ...	ПО
1	Внедрение возможности 'Зарплата в обмен на вре...	выплаты
2	Организация регулярных сессий обучения по ...	ПО
3	Внедрение системы контроля качества кода и ста...	ПО
4	Организация еженедельных занятий йогой или пил...	здоровье
...	...	...
309	Организация еженедельных занятий по медитации ...	здоровье
310	Внедрение системы контроля версий для управлен...	ПО
311	Создание системы непрерывного совершенствовани...	проектная деятельность
312	Организация внутренней академии по управлению ...	проектная деятельность
313	Введение гибкого графика работы	рабочее место

Рис. 1. Сформированный набор данных

Результат проверки балансировки количества идей по выбранным категориям (рис. 2). Количество идей по некоторым категориям меньше, чем по остальным, но такой перевес не является критичным.

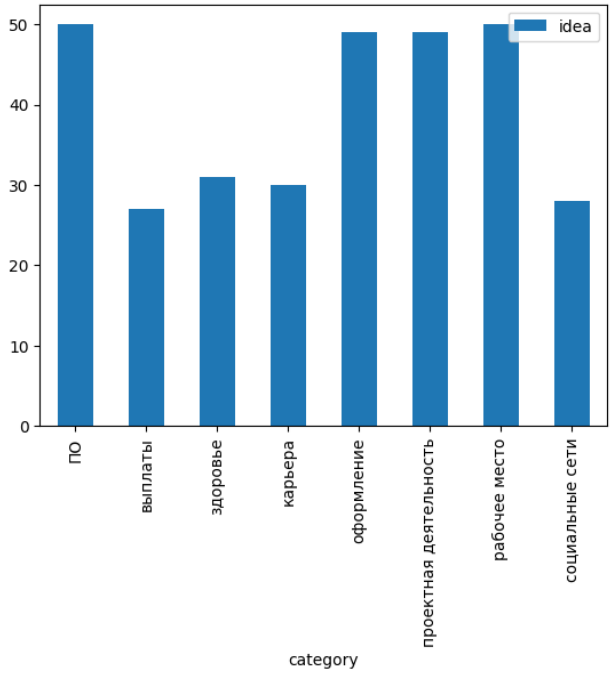


Рис. 2. Распределение количества идей по категориям

2.1. Кодирование нечисловых признаков

Требуется закодировать столбец категория, который имеет много уникальных значений, методом *factorize* [1]. Каждое уникальное значение категории преобразуется в целое число. Таким образом, в сформированном наборе данных метка класса идеи представлена числом в диапазоне от 0 до 7.

2.2. Очистка текста

Учитывая человеческий фактор при создании идей важно избавиться от неинформативных слов. Способ применения списка стоп-слов на основе русского языка позволит увеличить точность обучения.

В данном наборе данных можно исключить следующие стоп-слова: предлоги, причастия, междометия, частицы, нецензурную лексику и т. п. Для удаления стоп-слов из дататсета можно использовать библиотека *nltk* [2]. Допустимо удалить из данного набора данных лишние пробелы и смайлы.

3. Используемые модели машинного обучения

Для классификации идей сотрудников были выбраны следующие методы машинного обучения:

- алгоритм CatBoost;
- алгоритм FastText.

3.1. Модель CatBoostClassifier

Библиотека CatBoost представляет собой реализацию алгоритма градиентного бустинга [3]. CatBoostClassifier является усовершенствованной моделью градиентног бустинга, которая особенно эффективна в работе с категориальными данными. Таким образом, при использовании данного алгоритма не требуется предварительная обработка данных, например, факторизация нечисловых признаков. Еще одним преимуществом модели является автоматическая векторизация текста.

При помощи модели CatBoostClassifier на тестовой выборке была получена точность 0.66.

3.2. Модель FastText

FastText – это Python библиотека, разработанная для обработки естественного языка. Она содержит предобученные векторные представления слов и классификатор. Модель FastText использует n-граммы символов и слов.

Для обучения этой модели требуются данные в особом формате (рис. 3).

```
__label__5 создание брошюр или презентаций о ключевых аспектах работы и культуры компании для новых сотрудников.
__label__2 организация еженедельных занятий по медитации и дыхательным практикам, проводимых квалифицированным инструктором
__label__4 создание внутреннего блога или рассылки с интересными материалами по профессиональной тематике.
__label__7 создание внутреннего форума для обсуждения карьерных вопросов.
__label__1 установление ежемесячных премий лучшим сотрудникам.
__label__3 создание корпоративного журнала или газеты с новостями о компании и достижениями сотрудников.
__label__7 создание внутреннего клуба для обмена опытом и идеями.
__label__1 предусмотреть быструю и удобную подачу заявок на материальную помощь или отпускные
__label__7 внедрение программы ассистирования для новых сотрудников.
__label__0 внедрение гибкого графика работы.
```

Рис. 3. Пример форматирования датасета для использования FastText

Модель обучалась на тестовых данных тремя способами:

- без настройки гиперпараметров;
- с ручным подбором гиперпараметров;
- автоматическим перебором гиперапараметров.

В результате на тестовых данных лучше всего показала себя модель с автоматическим подбором гиперпараметров. Ее точность составила 0.71.

Заключение

После обучения двух моделей можно сделать вывод, что модель FastText смогла более точно предсказывать категории идей, чем CatBoost. Это обусловлено тем, что модель FastText изначально предназначена для обработки текста, а также для векторизации слов используются одновременно и skip-gram, и негативное семплирование, и алгоритм непрерывного мешка.

В итоге, несмотря на небольшое количество тренировочных данных, решение задачи классификации идей сотрудников по выбранным категориям может быть выполнено при помощи модели FastText. При наличии большего набора тренировочных данных точность обучения может стать выше.

Литература

1. Теория и практика машинного обучения : учеб. пособие / В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов. – Ульяновск : УлГТУ, 2017. – 290 с.
2. Элбон К. Машинное обучение с использованием Python. Сборник рецептов / К. Элбон. – Санкт-Петербург : БХВ - Петербург, 2020. – 384 с.
3. Градиентный бустинг – просто о сложном : официальный сайт. – URL: https://neurohive. io/ru/osnovy-data-science/gradientyj-busting/ (последняя дата обращения: 17.04.2024).
4. Астахова, А. В. Разработка приложения для экспертизы коллективных идей / А. В. Астахова // Актуальные проблемы прикладной математики, информатики и механики : сб. тр. Междунар. науч.-тех. конф. (Воронеж, 12–14 декабря 2022 г.) : электронный ресурс. – Воронеж, 2022. С. 741–744.
5. Астахова, А. В. Метод генерации идей для приложения экспертизы коллективных идей / А. В. Астахова // Математика, информационные технологии, приложения: сб. тр. Межвузовская научная конференция молодых ученых и студентов (Воронеж, 26 апреля 2023 г.) : электронный ресурс. – Воронеж, 2023. С. 13–17.

УДК 004.932 : 535.317.64 : 001.891.573 : 004.925.8

ОБЗОР ЗАДАЧИ РЕКОНСТРУКЦИИ ТРЁХМЕРНОЙ МОДЕЛИ ПО НАБОРУ ИЗОБРАЖЕНИЙ

А. И. Баклашов

Воронежский государственный университет

Введение

Восстановление трехмерной модели объекта или сцены по набору изображений является фундаментальной задачей в области компьютерного зрения и фотограмметрии [1]. Применение этой технологии простираются от робототехники и виртуальной реальности до создания цифровых двойников, медицины и архитектуры. В отличие от других методов 3D-реконструкции, этот подход отличается доступностью и простотой в плане технического оснащения: для съемки объекта достаточно нескольких камер. Однако за кажущейся легкостью скрывается сложный процесс преобразования двухмерных изображений в трехмерную модель. Этот процесс предъявляет высокие требования как к этапу съёмки, так и к вычислительной мощности компьютера. В данной статье приводится краткий анализ задачи реконструкции 3D-модели по набору изображений, её особенностей и сложности реализации.

1. Подготовка изображений

Перед началом реконструкции 3D-модели необходимо провести трансформацию изображений. Это необходимо по нескольким причинам. Во-первых, для устранения геометрических искажений, вызванных оптической кривизной линзовых систем – дисторсией, которая проявляется в виде неравномерного растяжения всего изображения [2–5]. Во-вторых, для специального выравнивания изображений с целью более точного обнаружения характерных признаков, по которым можно будет провести триангуляцию точек модели и затем реконструкцию самой поверхности [6].

1.1. Исправление искажения от дисторсии на основе модели Брауна — Конради

Дисторсией называется такой тип оптической аберрации, при котором происходит изменение коэффициента линейного увеличения по полю зрения, в результате чего нарушается геометрическое подобие между наблюдаемым объектом и его изображением. Результатом дисторсии служит искривление прямых линий на изображении, кроме линий, лежащих в одной плоскости с оптической осью [2].

Наиболее распространенной моделью дисторсии является модель Брауна — Конради с тремя радиальными и двумя тангенциальными коэффициентами.

$$\begin{cases} x_d = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1\left(r^2 + 2x^2\right) + 2p_2 xy, \\ y_d = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_2\left(r^2 + 2y^2\right) + 2p_1 xy, \end{cases}$$

где $\left(x_d, y_d\right)$ — нормализованные искажённые координаты; $\left(x, y\right)$ — нормализованные

неискажённые координаты; $r = \sqrt{x^2 + y^2}$ — расстояние от оптического центра; k_1, k_2, k_3 — коэффициенты радиальной дисторсии; p_1, p_2 — коэффициенты тангенциальной дисторсии.

Данная модель содержит два вида дисторсии: радиальную и тангенциальную. Радиальная дисторсия обусловлена кривизной сферической поверхностью линз объектива, тогда как тангенциальная — невыровненным положением главной оптической оси, которая может не проходить через центр кадра, либо вообще не быть перпендикулярной плоскости изображения.

В модели предполагается, что смещение оптической оси учитывается при преобразовании в нормализованные координаты изображения с помощью матриц камер. Вычисление искажённых и неискажённых координат изображения происходит в нормализованных координатах, которые вычисляются с помощью так называемых внутренней и внешней матриц камеры, которые имеют следующий вид:

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}.$$

Коэффициенты внутренней матрицы вычисляются после калибровки камеры. В качестве параметров f_x, f_y используется фокусное расстояние, преобразованное в пиксельный формат, тогда как в качестве параметров c_x, c_y — смещение оптической оси, преобразованное в пиксельный формат. В алгоритмах дисторсии внутренняя матрица используется для преобразования координат изображения из пиксельных в нормализованные.

Коэффициенты внешней матрицы камеры вычисляются перед трансформацией изображения. В качестве параметров f_x, f_y используются коэффициенты масштабирования, тогда как в качестве параметров c_x, c_y — смещение изображения. В алгоритмах дисторсии внешняя матрица используется для преобразования координат изображения из однородных в пиксельные.

Алгоритм исправления искажения дисторсии на изображении выглядит следующим образом:

1. Определение внешней матрицы камеры.
2. Составление карты для трансформации изображения с использованием внутренней и внешней матриц камеры, а также коэффициентов дисторсии.
3. Трансформация изображения с помощью вычисленной карты.

1.2. Ректификация изображений

Ректификация изображений – это процесс трансформации набора изображений с целью обеспечения соответствия горизонтальных линий на всех изображениях одной плоскости. Это позволяет устранить искажения, вызванные перспективой и другими факторами, обеспечить более точное выравнивание изображений для последующей обработки.

1.3. Эпиполярная геометрия

Эпиполярная геометрия описывает стереопару — две камеры с известным взаимным расположением, производящие фотосъемку одного и того же трехмерного объекта. При этом точка, видимая на первом снимке, на втором снимке может находиться только на эпиполярной линии. Эту зависимость возможно использовать для нахождения соответствий между точками на снимках. Предположим, что точка X видна одновременно на левой камере с центром O_L и правой камере с центром O_R. На плоскости изображения левой камеры трёхмерной точке X соответствует двумерная точка X_L, на плоскости изображения правой камеры — точка X_R Линия, соединяющая оптические центры камер O_L и O_R (т.н. базовая линия), пересекает плоскости изображений в точках ε_L и ε_R , называемых эпиполями (Рис. 1).

Очевидно, что для каждой трехмерной точки X существует своя эпиполярная плоскость $O_L O_R X$, проходящая через X(а следовательно, и через X_L) и базовую линию. При этом пересечение эпиполярной плоскости с плоскостью изображения правого снимка формирует эпиполярную линию. Соответственно, все изображения точек $x_1 \ldots x_n$, лежащих на луче $O_L X$, включая саму точку X, на правом снимке будут лежать на данной эпиполярной линии.

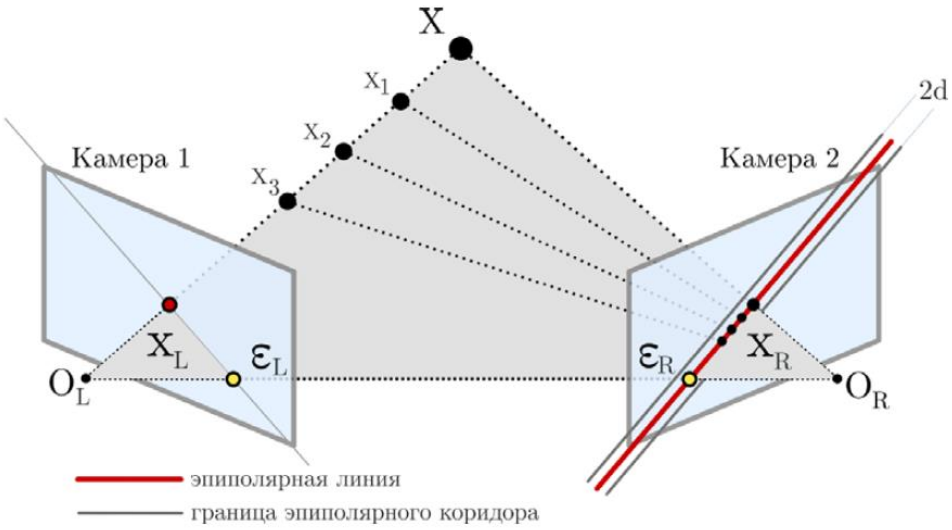


Рис. 1. Схематичное представление эпиполярной геометрии

В случае, если истинные трёхмерные координаты точки X нам неизвестны (например, при обработке фотоснимков, где известны только двухмерные координаты точки X_L), мы не можем однозначно установить положение изображения X_R трёхмерной точки Хна правом снимке из-за потери информации о глубине при проецировании. Однако в таком случае мы можем сузить область поиска соответствующей точки со всей площади изображения до эпиполярной линии.

При известном взаимном расположении камер эпиполярная линия e(вида $ax + by + c = 0$) для точки X_L на правом снимке может быть описана уравнением

$$e = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = F \cdot X_L$$

где F— фундаментальная матрица, вычисляемая через матрицу внутренних параметров камеры Kи существенную матрицу E:

$$F = K^{-1} \cdot E \cdot K = K^{-1} \cdot \begin{bmatrix} 0 & -tx(3) & -tx(2) \\ -tx(3) & 0 & -tx(1) \\ -tx(2) & -tx(1) & 0 \end{bmatrix} \cdot R \cdot K$$

где, в свою очередь, R и tx — матрица поворота и вектор переноса между двумя камерами, формирующими стереопару. При наличии матриц вращения A_L и A_R, а также векторов переноса t_L и t_Lдля левой и правой камер в общей (глобальной) системе координат R и tx можно вычислить следующим образом:

$$R = A_R * A_L^T,$$

$$tx = \frac{t_R^T - R * t_L^T}{\sqrt{\left\| t_R^T - R * t_L^T \right\|}}$$

После вычисления уравнений эпиполярных линий, существует два подхода к дальнейшему применению алгоритмов стереосогласования:

1. Трансформация изображения — преобразование исходных изображений таким образом, чтобы эпиполярные линии стали параллельны сторонам изображения, а соответствующие точки будут располагаться на одних и тех же горизонтальных линиях обоих изображений.

2. Поиск вдоль эпиполярных линий — поиск соответствий для каждого пикселя изображения вдоль эпиполярной линии, вычисленной с помощью соответствующего уравнения.

На практике используется первый вариант, так как он существенно повышает эффективность стереосогласования за счёт упрощения поиска соответствий, которые после преобразований располагаются на одних и тех же горизонтальных эпиполярных линиях.

2. Построение карты диспарантности

Карта диспарантности: это тип карты глубины, использующийся в системах стереокамер. Она вычисляется путем сравнения различий между координатами пикселями одного и того же отличительного признака на двух или более изображениях одной и той же сцены, сделанных с разных точек обзора. Несоответствие является горизонтальным сдвигу пикселей между изображениями, который вызван эффектом параллакса. Чем больше значение несоответствия в пикселе, тем ближе его пространственная точка находится к положению камеры.

Вычисление карты диспарантности обычно проводится в два этапа: на первом этапе поиск происходит вдоль всей эпиполярной линии и носит предположительный характер, ввиду целочисленного значения разницы в пиксельных координатах. Сравнение растровых данных пикселя только по одной координате является некорректным способом сравнения, ввиду того что из-за разного расположения камеры при съемке объекта, отличительные черты с разных ракурсов могут иметь различные растровые данные, как из-за различной освещенности, так и из-за небольшого изменения оттенка. Наиболее эффективным способом является сравнение с данными соседних пикселей, т.е. в окне определенного размера.

Пример такого способа — сравнение пикселей в окне 3×3 с помощью оценки

нормализированной кросс корреляции. Окно размера 3×3 позволяет учитывать информацию о соседях при сравнении, а нормализированная кросс корреляция позволяет отчасти нивелировать различия в освещенности при съемке с разных ракурсов.

$$d_{i,j} = \frac{\sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} \left(x_{ij} - \bar{x}\right)\left(y_{ij} - \bar{y}\right)}{\sqrt{\sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} \left(x_{ij} - \bar{x}\right)} \sqrt{\sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} \left(y_{ij} - \bar{y}\right)}}$$

Где: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ – средние значения пикселей в окне N×N, а x, y — изображения со стереопары.

Рис. 2. Карта диспарантности после первого этапа

На данном этапе при попытке визуализации карты диспарантности можно увидеть «лесенку» из пиксельных областей. (Рис. 2) Получившийся результат не подходит для точной реконструкции, поэтому вторым этапом происходит субпиксельное уточнение значения диспарантности. Для этого обычно аппроксимируют функцию по значениям пикселей. Например это может быть аппроксимации параболы по соседним пикселям по инвертированной оценке нормализированной кросс-корреляции

$$\overline{NCC} = \frac{(1 - NCC)}{2}$$

$$d_{xy} = \begin{cases} d_{xy} - 0.5, & \overline{NCC}_{-1} < \overline{NCC}_{+1}, \overline{NCC}_0 \\ d_{xy} + 0.5\dfrac{\overline{NCC}_{-1} - \overline{NCC}_{+1}}{\overline{NCC}_{-1} + \overline{NCC}_{+1} - 2\overline{NCC}_0}, & \overline{NCC}_0 < \overline{NCC}_{+1}, \overline{NCC}_{-1} \\ d_{xy} + 0.5, & \overline{NCC}_{+1} < \overline{NCC}_{-1}, \overline{NCC}_0 \end{cases}$$

Результатом данных вычислений служит карта диспарантности, вычисленная на субпиксельном уровне (Рис. 3), которую можно применить в алгоритмах расчёта облака точек.

Рис. 3. Карта диспарантности после второго этапа

3. Реконструкция модели

Для каждого пикселя на карте диспарантности мы можем взять по два луча для каждой из камер в стереопаре. Начало лучей соответствует положению камеры, а направление вычисляется с использованием матриц скалиброванных камер. Пересечение этих двух лучей даст нам точку в трехмерном пространстве, соответствующую пикселю на глубинной карте. Однако из-за неточностей в измерениях лучи A и B могут не иметь пересечения. Тогда для определения точки, которая будет является вершиной облака точек мы можем вычислить середину перпендикуляра, между скрещивающимися прямыми образующими лучи A и B. Это позволит получить приблизительное положение точки с учетом погрешности. Проделав данную операцию со всеми пикселями, мы получим облако точек.

Полученное облако точек необходимо триангулировать для получения поверхности, состоящие из треугольников. Это можно сделать с помощью триангуляции Делоне в трёхмерном пространстве или же нахождением выпуклой оболочки для полученного облака точек. В результате получаем полигональную 3D-модель, готовую для дальнейшего использования.

Заключение

В представленной работе проанализирован алгоритм построения трёхмерной полигональной модели по набору изображений.

Работа выполнена под руководством кандидата физ.-мат. наук, доцента кафедры вычислительной математики и прикладных информационных технологий Воронежского государственного университета Королькова Олега Геннадьевича.

Литература

1. Camera Calibration and 3D Reconstruction // OpenCV-Python Tutorials. — URL: https://docs.opencv.org/4.x/d9/db7/tutorial_py_table_of_contents_calib3d.html (дата обращения: 19.04.2024).

2. Distortion (optics) // Wikipedia, the free encyclopedia. — URL: https://en.wikipedia.org/wiki/Distortion_(optics) (дата обращения: 19.04.2024).

3. Villiers J. P. D. Correction of radially asymmetric lens distortion with a closed form solution and inverse function. — URL: https://core.ac.uk/download/pdf/80895194.pdf (дата обращения: 19.04.2024).

4. Leotta M. J. On the Maximum Radius of Polynomial Lens Distortion / M. J. Leotta, D. Russell and A. Matrai // 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2022. — P. 2374-2382. — doi: 10.1109/WACV51458.2022.00243.

5. Баклашов А. И. Исправление геометрических искажений от линз на изображении средствами библиотеки OpenCV: проблемы и пути их решения / А. И. Баклашов, О. Г. Корольков // Актуальные проблемы прикладной математики, информатики и механики: сборник трудов Международной научной конференции. – Воронеж, 2024.

6. Beeler T. High-Quality Single-Shot Capture of Facial Geometry / T. Beeler, B. Bickel, P. A. Beardsley, B. Sumner, M. H. Grossi // 2010 ACM SIGGRAPH 2010 papers.

УДК 004.4, 004.7

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ SPRING FRAMEWORK ПРИ РАЗРАБОТКЕ СЕРВЕРНОЙ ЧАСТИ WEB-ПРИЛОЖЕНИЯ «ОНЛАЙН-КИНОТЕАТР CINEMA-APP»

Д. Ю. Бакулина

Воронежский государственный университет

Введение

Развитие современных веб-приложений неразрывно связано с эффективным использованием средств и технологий, способных обеспечить гибкость, масштабируемость и надежность системы. В этом контексте Spring Framework, один из наиболее популярных и мощных инструментов для разработки Java-приложений, занимает особое место.

Целью данной статьи является рассмотрение особенностей применения Spring Framework при разработке серверной части веб-приложения «Онлайн-кинотеатр Cinema-App». В свете быстро развивающегося рынка онлайн-контента и увеличивающихся требований к пользовательскому опыту, эффективное использование инструментов разработки становится критически важным аспектом проекта.

В данной статье мы рассмотрим не только основные принципы работы с Spring Framework, но и его конкретное применение в контексте создания серверной части веб-приложения «Онлайн-кинотеатр Cinema-App». Мы проанализируем основные компоненты, модули и архитектурные решения, которые делают использование Spring эффективным инструментом для создания современных веб-приложений.

Таким образом, данная статья представляет собой обзор возможностей Spring Framework и их применение в контексте конкретного проекта, демонстрируя преимущества данного подхода и его важность для разработки современных web-приложений.

1. Описание проекта «Онлайн-кинотеатр Cinema-App»

Онлайн-кинотеатр Cinema-App - это инновационное веб-приложение, разработанное для создания современного и удобного пространства для кинолюбителей. Приложение обеспечивает пользователей возможностью удобного просмотра широкого спектра фильмов и сериалов в режиме онлайн.

Проект состоит из двух основных компонентов: клиентской и серверной частей. Клиентская часть представляет собой интуитивно понятный веб-интерфейс, который позволяет пользователям наслаждаться просмотром фильмов и сериалов в высоком качестве, создавать персональные списки просмотра и взаимодействовать с другими функциями приложения.

Серверная часть приложения играет ключевую роль в обеспечении его функциональности и безопасности. В её задачи входит управление базой данных, обработка запросов пользователей, авторизация и аутентификация пользователей, а также синхронизация данных между клиентскими приложениями и базой данных. Кроме того, серверная часть ответственна за обновление и поддержку приложения, включая исправление ошибок и добавление новых функций.

С использованием Spring Framework серверная часть приложения достигает высокой степени надежности, масштабируемости и производительности, что делает Cinema-App надёжным и удобным выбором для всех поклонников кинематографа.

2. Этапы разработки

Разработка web-приложения была разделена на несколько этапов:

1. Анализ требований: на этом этапе происходит изучение потребностей пользователей и определение основных функциональных и нефункциональных требований к приложению. Это включает в себя анализ рынка онлайн-кинотеатров, определение основных функций для пользователей и администраторов, а также выбор технологических стеков для разработки.
2. Проектирование архитектуры: на этом этапе разрабатывается общая архитектура приложения, определяются основные компоненты системы, их взаимосвязи и способы взаимодействия. Разрабатывается структура базы данных для хранения информации о фильмах, пользователях, сеансах и других сущностях приложения.
3. Разработка клиентской части: на данном этапе создается веб-интерфейс для пользователей, который включает в себя функции поиска фильмов, просмотра информации о них, создания персональных списков просмотра и другие функции, необходимые для удобного использования приложения.
4. Разработка серверной части: этот этап включает в себя создание серверной части приложения с использованием Spring Framework. Разрабатываются API для взаимодействия с клиентской частью, обработка запросов пользователей, управление базой данных и обеспечение безопасности приложения.
5. Тестирование: после завершения разработки проводятся тесты, включающие модульное, интеграционное и системное тестирование приложения. Проверяется работоспособность всех функций приложения, его производительность, безопасность и совместимость с различными устройствами и браузерами.
6. Релиз и внедрение: после успешного завершения тестирования приложение готово к выпуску в продакшн. На этом этапе происходит его установка на серверы, настройка среды выполнения и запуск в реальной среде. После выпуска приложения проводится мониторинг его работы и реагирование на обнаруженные проблемы.
7. Поддержка и сопровождение: после выпуска приложения в эксплуатацию начинается его поддержка и сопровождение. Это включает в себя реагирование на обратную связь пользователей, исправление ошибок, обновление функциональности и обеспечение безопасности приложения.

3. Анализ Spring Framework

3.1. Краткое описание

Серверная часть была реализована на языке Java 17 версии, с использованием фреймворка Spring Boot. Он предоставляет набор инструментов и библиотек для упрощения процесса создания и развертывания приложений. При разработке были использованы следующие технологии:

1. Spring Security. Модуль для обеспечения безопасности приложения, который предоставляет механизмы аутентификации и авторизации, что позволяет управлять доступом пользователей к различным частям приложения.
2. Spring Data JPA. Модуль, который предоставляет удобный и простой способ работы с базой данных. Он позволяет сократить количество кода, необходимого

для работы с базой данных, благодаря автоматической генерации SQL-запросов.
3. Spring MVC. Модуль, используемый для обработки REST-запросов и возврата результатов в виде HTML-страницы. Spring MVC обеспечивает эффективное взаимодействие между клиентской и серверной частями приложения.
4. Spring Web: Модуль Spring Framework, предоставляющий инструменты для разработки веб-приложений, включая контроллеры и возможности валидации данных.
5. Mapstruct: Библиотека, которая упрощает преобразование Java объектов в DTO (Data Transfer Objects) и обратно, что улучшает читаемость и поддерживаемость кода.

3.2. Инструментарий

Сервер Apache Tomcat, встроенный в фреймворк, был использован в качестве web-сервера. Для хранения данных была выбрана база данных PostgreSQL. PostgreSQL обладает мощными механизмами безопасности и предоставляет высокую производительность даже для небольших объемов данных Также к данной СУБД уже существует драйвер соотвествующий стандарту JDBC, позволяющий наладить общение между приложением на Spring Framework и используемой базой данных без особых трудностей, а поддержка стандарта JPA позволяет работать со строками из таблицы, будто это Java-объекты

Для работы с данными был выбран ORM Framework Hibernate, который обеспечивает удобную работу с базой данных на уровне объектов и повышает производительность приложения за счет оптимизации запросов к базе данных. Hibernate также обеспечивает безопасность приложения, защищая его от атак типа SQL injection. При помощи него было реализовано сохранение, извлечение и обновление объектов в базе данных с помощью операций CRUD (Create, Read, Update, Delete).

Для управления миграциями базы данных был использован инструмент Liquibase, который обеспечивает контроль версий схемы базы данных и ее обновление в соответствии с изменениями в коде приложения. Так же был использован Maven использование которого, упростило управление зависимостями и сборку проекта, обеспечивая единый способ управления проектом и его зависимостями.

Инструмет Swagger был использован для создания интерактивной документации API, что упростило взаимодействие с API разработанного приложения. Так же был добавлен очень важный инструмент Docker. Применение Docker обеспечило удобную среду выполнения приложения в контейнерах, что улучшило его масштабируемость и портабельность.

4. Реализация проекта

4.1. Проблематика

В результате сравнительного анализа с аналогичными приложениями данной предметной области были обнаружены недостатки, которые требуют внимания и доработки. Одной из главных проблем, выявленных в процессе анализа, является ограниченная функциональность приложения. Несмотря на существующие возможности просмотра фильмов и сериалов, в нем отсутствует ряд важных функций, которые могли бы улучшить пользовательский опыт.

В частности, необходимо обратить внимание на недостаточное количество информации о фильмах и сериалах, что затрудняет выбор пользователей и снижает привлекательность

приложения. Кроме того, в процессе анализа было выявлено, что приложение не поддерживает мультиязычность, что ограничивает доступ к контенту для аудитории, не владеющей английским языком.

Эти недостатки могут существенно сказаться на конкурентоспособности и успешности приложения. Поэтому необходимо провести дальнейшие работы по доработке и улучшению функциональности, а также обеспечить поддержку мультиязычности для привлечения широкой аудитории пользователей.

4.2. Разработка архитектуры приложения

Перед началом реализации проекта была разработана архитектура приложения, которая определяла структуру проекта и его компоненты. Архитектура приложения включала в себя следующие компоненты:

1. Контроллеры (Controllers): они отвечают за обработку HTTP-запросов, получаемых от клиента, и управление процессом обработки запросов. Контроллеры принимают запросы, взаимодействуют с сервисами и возвращают клиенту необходимые данные или ресурсы.
2. Сервисы (Services): сервисы содержат бизнес-логику приложения. Они обеспечивают взаимодействие между контроллерами и репозиториями, выполняя операции над данными, такие как создание, чтение, обновление и удаление. Сервисы также могут содержать логику проверки прав доступа и обработки ошибок.
3. Репозитории (Repositories): репозитории отвечают за взаимодействие с базой данных. Они предоставляют методы для выполнения операций с данными, таких как поиск, добавление, обновление и удаление. Репозитории позволяют абстрагироваться от деталей работы с базой данных и обеспечивают единый интерфейс доступа к данным для других компонентов приложения.
4. Энтити (Entitites): энтити представляют собой объекты данных, которые используются в приложении для хранения информации. Они отражают структуру данных в базе данных и могут содержать аннотации для маппинга на таблицы в реляционной базе данных. Энтити представляют собой сущности приложения, такие как пользователи, фильмы, сеансы и т. д.
5. Мапперы (Mappers): Они отвечают за преобразование объектов между различными форматами, например, между сущностями (entities) и DTO (Data Transfer Objects). Мапперы позволяют управлять трансформацией данных и поддерживать чистоту архитектуры приложения.
6. Исключения (Exceptions): Эти компоненты обрабатывают исключительные ситуации в приложении, такие как ошибки базы данных, ошибки валидации данных или ошибки авторизации. Исключения позволяют более гибко управлять ошибками и обеспечивать информативные сообщения об ошибках для пользователей приложения.
7. DTO (Data Transfer Objects): DTO используются для передачи данных между различными компонентами приложения, такими как контроллеры, сервисы и репозитории. Они позволяют оптимизировать передачу данных, исключая избыточную информацию и обеспечивая необходимую для работы компонентов.
8. Безопасность (Security): Этот компонент отвечает за обеспечение безопасности приложения, включая аутентификацию и авторизацию пользователей, защиту от атак и управление правами доступа к ресурсам приложения.

9. Утилиты (Utils): Утилиты предоставляют набор вспомогательных функций и методов, которые могут быть использованы различными компонентами приложения для выполнения общих задач, таких как работа с датами, форматирование строк, обработка файлов и т. д.

4.3. Разработка базы данных

Была разработана схема базы данных, которая включала в себя следующие таблицы:

1. Фильм (Movie): Хранит данные о фильмах, такие как название, описание, длительность, жанр, рейтинг, постер и т. д.
2. Пользователь (User): содержит информацию о пользователях (логин, пароль и т.д.).
3. Отзыв (Review): Содержит отзывы пользователей о фильмах, включая идентификатор отзыва, идентификатор пользователя, идентификатор фильма, текст отзыва, оценку и т. д.
4. Роль (Role): содержит информацию о роли пользователя (администратор и пользователь).
5. Медиа (MediaFile, MediaList, MediaText, MediaUrl): Содержат информацию о медиафайлах, таких как название файла, путь к файлу, размер файла, формат и т. д.,
6. Жанр (Genre): Хранит информацию о жанрах фильмов, такую как название жанра и описание.
7. Рейтинг (Rating): Содержит информацию о рейтингах, например, идентификатор рейтинга, значение рейтинга, комментарии, дата оценки и т. д.
8. Базовая Энтити (BaseEntity): Это базовая сущность, от которой наследуются другие сущности. Может содержать общие поля, такие как идентификатор, дата создания, дата обновления и т. д.
9. Базовая Медиа (BaseMedia): Аналогично базовой сущности, но специфично для медиафайлов. Может содержать общие поля, такие как название, описание, формат, размер и т. д.
10. Гендер (Gender): Хранит информацию о поле (мужской, женский, другой), например, идентификатор пола, название пола и т. д.
11. Профиль (Profile): Содержит информацию о профилях пользователей, такую как идентификатор профиля, имя, фамилия, дата рождения, пол, адрес и т. д.
12. Картинка Профиля (ProfilePicture): Хранит данные о картинках профиля пользователей, например, идентификатор картинки, путь к файлу, формат, размер, дата добавления и т. д.

Заключение

Результатом исследования стало обоснование выбора Spring Framework в качестве основного инструмента для разработки серверной части проекта. Spring Framework предоставляет мощный и гибкий инструментарий, позволяющий значительно упростить процесс создания и развертывания веб-приложений. Использование Spring Framework при разработке серверной части веб-приложения "Онлайн-кинотеатр Cinema-App" обеспечивает эффективную и надежную реализацию функциональности, удовлетворяя требованиям проекта и обеспечивая высокий уровень безопасности и производительности.

Литература

1. Документация Spring Boot – Режим доступа: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
2. Документация Hibernate ORM – Режим доступа: https://hibernate.org/orm/documentation/6.1/
3. Документация Spring Web – Режим доступа: https://docs.spring.io/spring-framework/reference/
4. Документация Spring Data JPA – Режим доступа: https://docs.spring.io/spring-data/jpa/reference/#reference
5. Документация Spring Security – Режим доступа: https://docs.spring.io/spring-security/reference/
6. Документация Spring Validation – Режим доступа: https://docs.spring.io/spring-framework/reference/core/validation.html
7. Документация PostgreSQL – Режим доступа: https://www.postgresql.org/docs/current/index.html
8. Документация Maven – Режим доступа: https://maven.apache.org/guides/index.html
9. Документация Docker – Режим доступа: https://docker-docs.uclv.cu/
10. Документация Liquibase – Режим доступа: https://docs.liquibase.com/home.html

УДК 004.925

АЛГОРИТМ СОКРАЩЕНИЯ КОЛИЧЕСТВА ПОЛИГОНОВ ТРЁХМЕРНОЙ МОДЕЛИ ЗА СЧЁТ УДАЛЕНИЯ НАБОРА ЗАДАННЫХ ВЕРШИН

С. А. Батуров

Воронежский государственный университет

Введение

Одной из основных задач компьютерной графики является визуализация трёхмерных полигональных моделей. При этом первоочередную роль играет производительность алгоритмов, выполняющих данную задачу. Одним из способов повышения скорости визуализации является замена высокополигональных моделей низкополигональными там, где это уместно (например, для объектов, расположенных на заднем плане и не требующих детальной отрисовки) [1]. В данном контексте интерес представляет задача оптимизации количества полигонов в модели, которая в некоторых случаях может сводиться к двум этапам: удалению некоторого набора вершин и последующей перестройки модели.

Настоящая статья посвящена разработке алгоритма сокращения количества полигонов в трёхмерной модели за счёт удаления некоторого набора заранее помеченных вершин.

1. Общая идея алгоритма

Пусть задана некоторая триангулированная трёхмерная полигональная модель, а также массив вершин, помеченных на удаление. Рассмотрим задачу удаления помеченных вершин из исходной модели и построения новой триангулированной модели и сформулируем алгоритм, необходимый для получения решения.

Для начала стоит отметить, что при удалении вершины происходит также удаление всех смежных с ней рёбер, что приводит к созданию отверстий («дыр») в сетке модели, которые необходимо заполнить. Причём в процессе удаления вершин могут оставаться «висячие» вершины, рёбра и грани модели, которые могут довольно сильно осложнить задачу заполнения «дыр». Данная ситуация возможна только при наличии замкнутых контуров, состоящих только их помеченных вершин. По этой причине удаление стоит производить в несколько этапов, таким образом, чтобы подобные элементы не появлялись.

2. Описание алгоритма удаления вершин

Пусть каждая вершина имеет метку 1, если её необходимо удалить, и 0, если нет. Во время процесса будем помечать удалённые вершины меткой 0. Будем вести цикл пока вершин, помеченных единицей, не останется. Рассмотрим один шаг цикла и опишем этапы алгоритма.

Шаг 1. Производим дополнительную разметку вершин. Будем давать вершине метку −1, если её необходимо удалить на следующих итерациях, а 1 – если её необходимо удалить на данной. Пометки будем делать следующим образом: перебираем каждый полигон, если в нём две или более вершины, поставленные на удаление в данной итерации, то помечаем их −1, кроме одной. После данного этапа получается такой набор вершин, после удаления которых «висячих» элементов сетки не останется.

Шаг 2. Производим удаление всех вершин, помеченных единицей, вместе со смежными гранями. Если в модели присутствуют рёбра, которые не находятся в составе полигонов, то мы их также удаляем.

Шаг 3. Заполняем полученные дыры в сетке модели. При заполнении будут рассматриваться только те вершины, которые составляют контур образовавшегося отверстия. Более подробно данный шаг будет описан в следующем разделе настоящей статьи.

Шаг 4. Если вершин, помеченных −1, не осталось, то алгоритм завершается. Иначе переходим к первому шагу.

3. Описание алгоритма заполнения отверстий

При удалении вершины будем помечать контур отверстия, образованный удалением вершины. Рассмотрим работу алгоритма на одном из контуров.

Шаг 1. Для заполнения, выберем некоторую вершину и смежные с ней две вершины и сформируем из них полигон.

Шаг 2. Для предыдущих двух вершин рассмотрим две следующие вершины. Если они являются смежными друг другу на контуре, то завершаем алгоритм. Если для них смежная вершина контура общая – то, переходим к шагу 4. Иначе, переходим к шагу 3.

Шаг 3. Формируем два новых полигона на основе этих четырёх вершин так, чтобы их общая сторона являлась диагональю четырёхугольника.

Шаг 4. Формируем полигон из данных трёх вершин.

После прохождения алгоритма будет получена модель более низкого разрешения, сохраняющая общую форму изначального объекта. Данный алгоритм работает с любой сеткой, однако для моделей более низкого разрешения многие детали теряются и заметен некоторый уровень деформации.

4. Пересчёт нормалей для новой сетки

Для формирования полноценной трёхмерной модели недостаточно только удалить вершины и сформировать сетку. Необходима ещё и информация о нормалях вершин. Так как во время процесса удаления изменяется сетка и форма модели, необходимо обновлять информацию о нормалях вершин, иначе для полученной модели некорректно будут работать, например, алгоритмы освещения [2].

Пересчёт нормалей будем производить в предположении, что для каждой вершины нормаль равна среднему арифметическому нормалей смежных с ней вершин, что даёт эффективный способ приближённого расчёта [3].

Рассмотрим некоторую вершину, необозначенную на удаление, с нормалью $\overline{n_0}$ и количеством смежных вершин k, и некоторую смежную с ней вершину, обозначенную на удаление, которая имеет нормаль равную вектору $\overline{n_1}$. Тогда в качестве нормали вершины, заменяющей две данные, можно взять

$$\overline{n_0} := \frac{\overline{n_0} \cdot k - \overline{n_1}}{k-1}.$$

Данный пересчёт производится на каждой из итераций цикла и для каждой смежной вершины, отмеченной на удаление. Таким образом будет произведена коррекция нормалей для всех вершин, которые не будут удалены.

5. Результат работы программы

Для начала рассмотрим простую модель сферы, содержащей в себе 482 вершин и 960 граней. Случайным образом отмечено 236 вершин на удаление. В результате применения разработанного алгоритма остаётся 246 вершин и 570 граней в модели. Время работы алгоритма 0.032 секунды. Результат представлен на рис. 1.

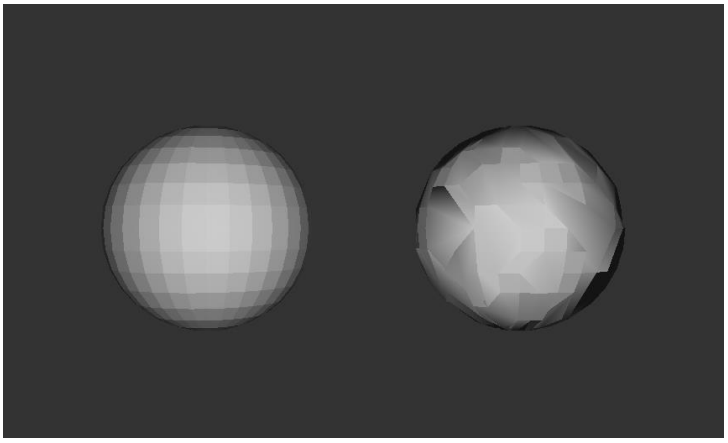


Рис. 1. Визуализация результатов работы
(слева – исходная модель, справа – обработанная)

Заключение

В работе представлен алгоритм сокращения количества полигонов трёхмерной модели, сохраняющий её общую форму. Основным преимуществом разработанного алгоритма является его универсальность: алгоритм не зависит ни от выбора трёхмерной модели, ни от выбора удаляемых вершин. С другой стороны, работа данного алгоритма может занимать достаточно большое время, так как удаление происходит в несколько этапов. Исходя из этого данный алгоритм может быть применим в первую очередь для создания нескольких версий модели различных уровней разрешения, которые могут быть использованы в системах управления уровнем детализации.

Работа выполнена под руководством кандидата физ.-мат. наук, доцента кафедры вычислительной математики и прикладных информационных технологий Воронежского государственного университета Королькова Олега Геннадьевича.

Литература

1. Luebke D. Level of Detail for 3D Graphics / D. Luebke [и др.]. – Сан-Франциско: Morgan Kaufmann Publishers, 2003.

2. Программирование 3d-графики – Режим доступа: https://www.helloworld.ru/texts/comp/games/3d/faq/articles/51.htm (Дата обращения: 19.04.2024).

3. Луна Ф. Д., Введение в программирование трёхмерных игр с DirectX 9.0 / Ф. Д. Луна. – Сетевое издательство NetLib, 2006. – 139 с.

УДК 004

Анализ технологий для создания интернет-магазина

С. Н. Бибанг Ннама

Воронежский государственный университет

Введение

Выбор правильных технологий для создания интернет-магазина играет ключевую роль в успехе бизнеса. От веб-фреймворка до системы управления базами данных, каждое технологическое решение имеет значительное влияние на производительность, безопасность и пользовательский опыт.

В данной статье рассматривается важность выбора правильных инструментов для разработки интернет-магазина, а также проводится анализ современных технологий, включая веб-фреймворки, среды разработки, фреймворки для разработки интерфейса и системы управления базами данных.

Понимание этих технологий поможет разработчикам и бизнес-владельцам принимать обоснованные решения при создании и поддержке онлайн-магазинов. Правильный выбор технологий определяет не только функциональность, но и конкурентоспособность интернет-магазина на современном цифровом рынке.

1. Выбор веб-фреймворка для разработки интернет-магазина

Существует множество веб-фреймворков, предоставляющих разработчикам инструменты для создания веб-приложений.

Один из них — Django, специализированный для Python. Django отличается высокой производительностью и предоставляет широкий спектр инструментов, включая ORM, административный интерфейс, а также обработку URL.

Еще один популярный фреймворк — Spring, который предназначен для создания приложений на языке Java, включая веб-приложения с использованием модуля Spring MVC.

ASP.net позволяет создавать веб-приложения на C# или других языках. Поддерживает две модели: Web Forms для упрощенной разработки и MVC для гибкости. Интегрируется с Visual Studio, обеспечивает безопасность, поддерживает облачные решения Azure.

Рекомендуется выбирать веб-фреймфорк ASP.net, так как он имеет ряд преимуществ перед Django и Spring, а именно: тесная интеграция с другими технологиями Microsoft, такими как Windows Server и SQL Server, что обеспечивает эффективную работу в средах, использующих технологии этой компании; встроенные механизмы безопасности, включая систему идентификации ASP.NET Identity, облегчающие реализацию аутентификации и авторизации в приложениях; повышенная устойчивость приложения за счет сильный типизации, которая способствует предотвращению ошибок на этапе компиляции.

2. Выбор среды разработки

Для выбора среды разработки рассматриваются Visual Studio, Visual Studio Code и SharpDevelop, так это основные среды разработки, поддерживающие ASP.net.

Стоит отдавать предпочтение Visual Studio Code, основываясь на ряде преимуществ.

Visual Studio Code отличается легковесностью, что придает ей большую отзывчивость и скорость по сравнению с полной версией Visual Studio. Это значительно повышает производительность разработчика и делает процесс разработки более эффективным.

Кроме того, Visual Studio Code обладает мощными инструментами для веб-разработки. Среди них — подсветка синтаксиса, автоматическое дополнение кода, поддержка расширений и другие инструменты, которые значительно улучшают опыт разработки веб-приложений. Эти функции помогают разработчикам работать продуктивнее и создавать качественные веб-приложения.

Нельзя не отметить, что Visual Studio Code легко интегрируется с различными облачными сервисами и технологиями. Это делает среду разработки отличным выбором для создания интернет-магазина с использованием ASP.NET, позволяя программистам легко работать с облачными сервисами и развертывать приложения в облаке.

3. Выбор фреймворка для разработки интерфейса

Для разработки интерфейса веб-приложений доступно множество фреймворков, каждый из которых обладает своими особенностями и возможностями.

React — это библиотека JavaScript, предназначенная для создания пользовательских интерфейсов в веб-приложениях. Основой React является компонентная архитектура, что позволяет разбивать интерфейс на множество небольших и переиспользуемых компонентов. Он также использует виртуальный DOM для оптимизации производительности и обновления интерфейса только тех частей, которые действительно изменились.

Angular, в свою очередь, представляет собой полноценный фреймворк для разработки веб-приложений. Он также использует компонентную архитектуру, но предоставляет более широкий спектр инструментов, включая систему двунаправленного связывания данных и маршрутизацию для построения одностраничных приложений.

Vue.js, как прогрессивный JavaScript-фреймворк, обладает легкостью внедрения в проекты и плавным переходом от использования библиотеки к полноценному фреймворку. Он также использует компонентный подход и предоставляет возможности для двунаправленного связывания данных.

Выбор React обусловлен несколькими факторами. React является библиотекой, а не полноценным фреймворком, что обеспечивает большую гибкость в выборе других инструментов и библиотек для дополнительных задач, способствуя модульности кода. Кроме того, использование виртуального DOM позволяет React эффективно обновлять только те части интерфейса, которые действительно изменились, что повышает производительность приложений.

4. Проектирование базы данных

Проектирование базы данных является важной частью создания программного продукта. Для успешной работы, необходимо подобрать подходящую систему управления базами данных.

PostgreSQL — это система управления реляционными базами данных с открытым исходным кодом, обладающая множеством расширенных функций, включая сложные запросы, триггеры, процедуры и обработку геопространственных данных. Она отличается высокой производительностью, надежностью и расширяемостью.

MySQL, также являющаяся системой управления базами данных с открытым исходным кодом, широко применяется в различных типах приложений, включая веб-приложения и системы управления контентом. Её преимуществами являются простота использования,

высокая производительность и быстрая скорость обработки запросов.

Microsoft SQL Server, разработанный компанией Microsoft, предназначен в основном для приложений, разрабатываемых в экосистеме Microsoft. Он предоставляет множество возможностей, таких как хранение данных, поддержка транзакций, процедур хранения и интеграция с другими продуктами Microsoft.

По ряду причин была выбрана СУБД PostgreSQL: она предоставляет мощные механизмы безопасности, включая управление правами доступа и SSL-шифрование, обеспечивающие защиту данных. Также PostgreSQL поддерживает JSONB (бинарный формат JSON) и хранимые процедуры на языке PL/pgSQL. Более того, PostgreSQL предлагает более расширенный набор функций, что делает её подходящим выбором для широкого спектра задач, включая поддержку сложных запросов, полнотекстовый поиск, геопространственные данные и другие возможности.

Заключение

В результате исследования проанализированы и выявлены наилучшие для создания интернет-магазина веб-фреймворк для разработки backend’а веб-приложения, среда разработки, веб-фреймворк для разработки интерфейса и система управления базами данных. Исходя из ряда преимуществ, описанных в статье составлен рекомендательный стек технологий: ASP.net, Visual Studio Code, React, PostgreSQL.

Литература

1. Венц К. Программирование в ASP.NET AJAX. - СПб.: Символ-Плюс, 2017. - 502 с.
2. Чамберс Дж., Пэкетт Д., Тиммс С. ASP.NET Core. Разработка приложений. - СПб.: Питер, 2018. - 464 с.
3. Форсье Дж., Биссекс П., Чан У. ASP.NET Core. Разработка приложений. - СПб.: Символ-плюс, 2009. - 458 с.
4. Walls C. Spring in Action, Sixth Edition. - 6-е изд. - Manning, 2022. – 520 с.
5. Шарп Дж. Microsoft Visual C#. . - 8-е изд. - СПб.: Питер, 2017. - 848 с.
6. Johnson B. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. - 1-е изд. - Wiley, 2019. - 192 с.
7. Wieruch R. The Road to React: The React.js with Hooks in JavaScript Book (2023 Edition). - Kindle изд. - rweruch, 2022. - 394 с.
8. Murray N., Coury F., Lerner A., Taborda C., Raboy N., Holland B. The Ng-book — The Complete Book on Angular . - Fullstack.io, 2018. - 685 с.
9. Garaguso P., Zander O. Vue.js 3 Design Patterns and Best Practices: Develop scalable and robust applications with Vite, Pinia, and Vue Router. - Packt Publishing, 2023. - 296 с.
10. Новиков Б. А. Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. — 2-е изд. — М.: ДМК Пресс, 2020. — 582 с.

УДК 004.8

РЕАЛИЗАЦИЯ АЛГОРИТМА ОБНАРУЖЕНИЯ КЛЮЧЕВЫХ ТОЧЕК НА ИЗОБРАЖЕНИИ

Е. В. Бирюкова, О. С. Лемина

Воронежский государственный университет

Введение

Машинное обучение — это методы искусственного интеллекта, которые позволяют строить обучаемые модели для разных целей. Выделяют три составляющие машинного обучения: данные, признаки и алгоритм. [1, 2]

Обнаружение ключевых точек (keypoint detection) заключается в локализации основных частей объекта. Например, основными частями нашего лица являются кончик носа, брови, уголки глаз и т. д. Эти части помогают представить базовый объект в более детальном виде. Обнаружение ключевых точек используется в различных приложениях, таких как оценка положения (*pose estimation*), обнаружение лиц и т. д. [3]

В статье представлено описание основных шагов реализации алгоритма обнаружения ключевых точек на изображении с помощью машинного обучения.

1. Реализация алгоритма

1.1. Определение гиперпараметров

```
IMG_SIZE = 224
BATCH_SIZE = 64
EPOCHS = 5
NUM_KEYPOINTS = 24 * 2
```

Рис. 1. Определение гиперпараметров

В этой части программы происходит определение *гиперпараметров* модели (рис. 1).

Гиперпараметры модели — это параметры, которые не оптимизируются в процессе обучения модели, а задаются заранее и влияют на процесс обучения и структуру модели. [4]

IMG_SIZE — это размер изображений для обработки.

BATCH_SIZE — определяет количество полных проходов через весь набор данных в процессе обучения модели.

EPOCHS — количество эпох обучения.

NUM_KEYPOINTS задает общее количество ключевых точек, которые модель будет распознавать. Здесь количество ключевых точек равно 24, а умножение на 2 необходимо, так как каждая точка имеет две координаты.

1.2. Загрузка данных

В качестве исходных данных был выбран набор *StanfordExtra*, который содержит 12 000 изображений собак вместе с аннотациями. Он разработан на основе набора данных о собаках от Стэнфордского университета. На рис. 2 приведен фрагмент кода, выполняющий загрузку данных.

```
IMG_DIR = "Images"
JSON = "StanfordExtra_V12/StanfordExtra_v12.json"
#файл метаданных с доп информацией
KEYPOINT_DEF = (
    "https://github.com/benjiebob/StanfordExtra/raw/master/keypoint_definitions.csv"
)

# Загружаем аннотации
with open(JSON) as infile:
    json_data = json.load(infile)
```

Рис. 2. Загрузка данных для обучения

1.3. Подготовка генератора данных

Теперь необходимо реализовать генератор данных, который отвечает за генерацию и предобработку данных, а также применяет аугментацию к пакетам данных с использованием библиотеки *imgaug* для обеспечения разнообразия обучающих данных и улучшения обобщающей способности модели.

Данный класс имеет имя *KeyPointsDataset* и наследует класс *keras.utils.Sequence*. Основными методами класса являются:

- *init*: инициализирует объект класса;
- *len*: возвращает количество шагов (пакетов) в каждой эпохе обучения;
- *on_epoch_end*: вызывается в конце каждой эпохи обучения и перемешивает индексы ключевых изображений, если генератор используется для обучения;
- *getitem*: возвращает пакет данных для заданного индекса — индексы определяются на основе текущего пакета данных, и загружаются соответствующие изображения и их ключевые точки;
- *data_generation*: производит обработку пакета данных, включая загрузку изображений, аннотаций ключевых точек и применение аугментации данных.

1.4 Разбиение набора данных на обучающие и валидационные выборки

```
np.random.shuffle(samples)
train_keys, validation_keys = (
    samples[int(len(samples) * 0.15) :],
    samples[: int(len(samples) * 0.15)],
)
```

Рис. 3. Разбиение набора данных на обучающие и валидационные выборки

На данном этапе необходимо разбить исходный набор данных на *обучающую* и *валидационную* выборки. Разбиение набора данных на выборки представлено на рис. 3.

Обучающая выборка — это набор данных, на котором будет производиться обучение модели; *валидационная выборка* необходима для того, чтобы контролировать процесс обучения. Она поможет предотвратить переобучение и обеспечит более точную настройку входных параметров. [5]

1.5 Построение модели

Набор данных *Stanford Dogs*, на котором основан используемый набор *StanfordExtra*,

был создан путем использования набора данных *ImageNet-1k*. Таким образом, вероятно, что модели, предварительно обученные на наборе данных *ImageNet-1k*, будут полезны для данной задачи. Будем использовать предварительно обученную на этом наборе модель *MobileNetV2* в качестве основы для извлечения значимых особенностей изображений.

Внутри функции построения модели происходит следующее:

1. Загружаются предобученные веса *MobileNetV2* и замораживаются.
2. Создается входной слой, пропускающий данные через *MobileNetV2*.
3. Выполняется слой *Dropout*, который случайным образом удаляет некоторые выходные данные, чтобы предотвратить переобучение модели.
4. Выполняются два слоя *SeparableConv2D* для получения выходных данных. Эти слои используются для обработки изображений и извлечения признаков.
5. Функция возвращает модель *Keras* с входным слоем *inputs* и выходным слоем *outputs*.

1.6 Обучение модели

Теперь необходимо произвести обучение построенной модели. На рис. 4 представлен фрагмент кода, выполняющий эту задачу.

```
model = get_model()
model.compile(loss="mse", optimizer=keras.optimizers.Adam(1e-4))
model.fit(train_dataset, validation_data=validation_dataset, epochs=EPOCHS)
```

Рис. 4. Обучение модели

Этот код использует функцию *get_model* для создания модели, затем компилирует ее с помощью оптимизатора *Adam* и функции потерь *mse*. Затем модель обучается на тренировочном наборе данных с использованием валидационного набора данных в течение заданного числа эпох (EPOCH; в этом примере алгоритма обучение производится в течение пяти эпох). В результате обучения модель будет способна обнаруживать ключевые точки на изображении.

1.7 Выполнение предсказаний и визуализация данных

Здесь используется обученная модель для предсказания координат ключевых точек на изображениях из валидационного набора данных. Сначала из валидационного набора данных выбираются первые четыре изображения и соответствующие им ключевые точки. Затем изображения передаются в модель, и она выполняет предсказания координат ключевых точек на каждом изображении. Фрагмент кода, выполняющий описанные действия, представлен на рис. 5.

```
sample_val_images, sample_val_keypoints = next(iter(validation_dataset))
sample_val_images = sample_val_images[:4]
sample_val_keypoints = sample_val_keypoints[:4].reshape(-1, 24, 2) * IMG_SIZE
predictions = model.predict(sample_val_images).reshape(-1, 24, 2) * IMG_SIZE
```

Рис. 5. Выполнение предсказаний

Далее необходимо визуализировать оригинальные и предсказанные ключевые точки изображения. Для этого используется функция *visualize_keypoints*. Функция принимает два аргумента: *images* и *keypoints*. *images* — это список изображений, а *keypoints* — это

соответствующие им ключевые точки. Функция создает сетку изображений и для каждого изображения показывает его оригинал и изображение с предсказанными ключевыми точками. В конце функция выводит созданную сетку изображений с предсказанными ключевыми точками. Код, выполняющий визуализацию изображений, представлен на рис. 6.

```
# Полученные данные
visualize_keypoints(sample_val_images, sample_val_keypoints)

# Прогнозы
visualize_keypoints(sample_val_images, predictions)
```

Рис. 6. Визуализация изображений

2. Результаты визуализации

Результат визуализации изображения и его оригинальных ключевых точек представлен на рис. 7.

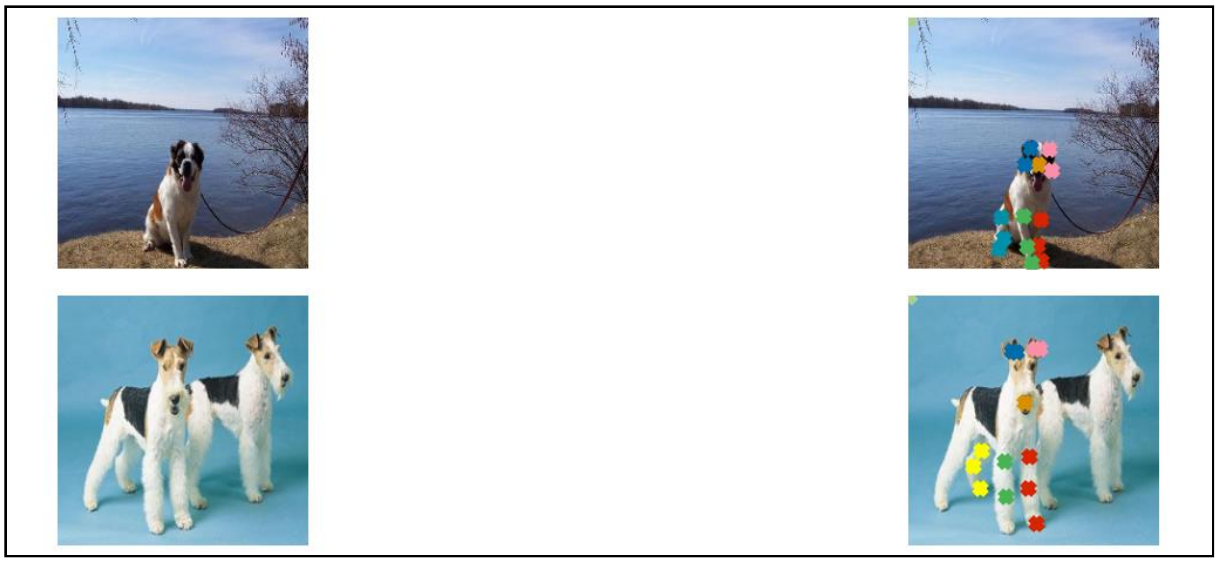


Рис. 7. Результат визуализации изображений с оригинальными ключевыми точками

Результат визуализации изображений и предсказанных ключевых точек представлен на

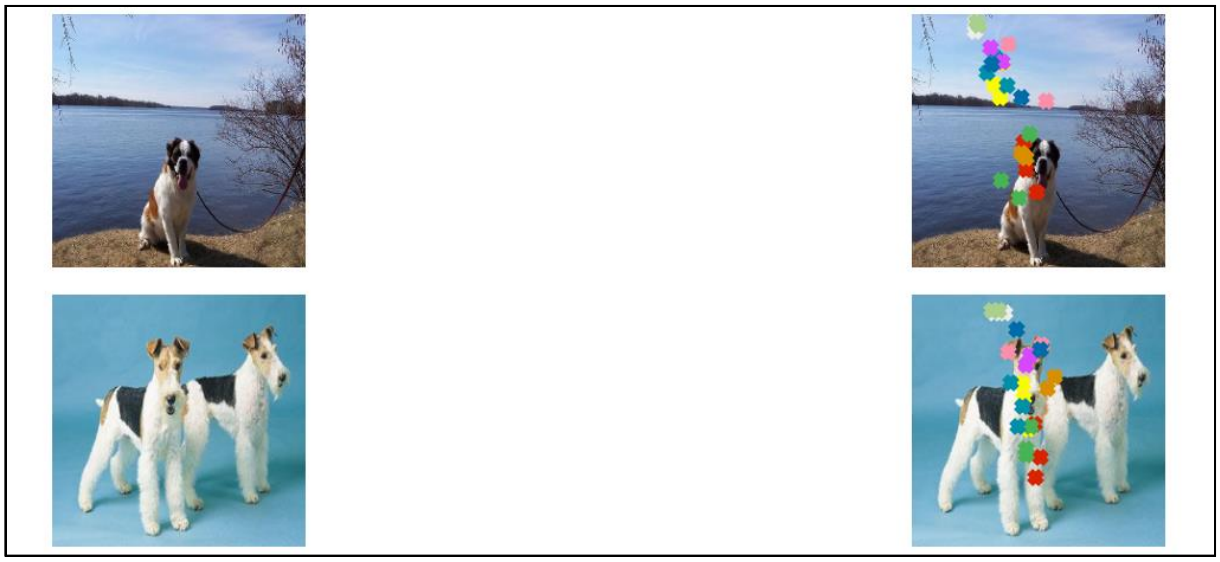


Рис. 8. Результат визуализации изображений с предсказанными ключевыми точками

рис. 8.

Заключение

В ходе реализации алгоритма были выполнены все основные этапы, такие как подготовка данных для обучения, построение модели, обучение модели, визуализация результатов.

Результаты визуализации показывают, что модель научилась распознавать ключевые точки на изображениях. Так как обучение модели проводилось в течение всего пяти эпох, предсказанные ключевые точки совпадают с оригинальными с низкой точностью. С увеличением количества тренировок прогнозы будут улучшаться. Тогда алгоритм будет демонстрировать высокую эффективность и точность распознавания ключевых точек.

Литература

1. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах. – Москва: ДМК Пресс, 2015. – 205 с.
2. Коэльо, Л. П. Построение систем машинного обучения на языке Python / Л. П. Коэльо, В. Ричарт; пер. с англ. Слинкин А. А. – Москва : ДМК Пресс, 2019. – 302 с.
3. Николенко, С. Глубокое обучение. Погружение в мир нейронных сетей / С. Николенко, А. Кадурин, Е. Архангельская. – Санкт-Петербург : Питер, 2018. – С. 81–91.
4. Жерон, О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow / О. Жерон. – Москва : БХВ, 2020. – С. 521–530.
5. Рашка, С. Python и машинное обучение / С. Рашка. – Москва: ДМК Пресс, 2015.– 116 с.

УДК *004.8*

ГЕНЕРАЦИЯ «РЫБНОГО» КОДА С ПОМОЩЬЮ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ

А. Д. Бирюлев

Воронежский государственный университет

Введение

В настоящее время искусственный интеллект позволяет решать множество задач, которые перед ним ставит человечество. Одна из таких задач – генерация исходного кода для написания программных средств, решающих определенные задачи.

Чаще всего, такой код является результатом вывода обученной модели на основе ввода некоторого условия или описания алгоритма, который необходимо построить. Результат должен соответствовать правилам языка, используемого при генерации, а также должен выполнять поставленную задачу. Гораздо реже появляется необходимость в генерации бессмысленного, «рыбного» исходного кода. «Рыбным» он называется по аналогии с «рыбным» текстом (текстом-заполнителем), и используется, например, в веб-дизайне в местах, где обыкновенный «рыбный» текст по тем или иным причинам не подходит. Подобный код-заполнитель может очень пригодиться при прототипировании и разработке онлайн-редакторов кода.

В данной статье рассмотрен подход к генерации такого исходного кода при помощи рекуррентных нейронных сетей.

1. Используемые инструменты

Для генерации «рыбного» исходного кода были выбраны символьные рекуррентные нейронные сети. Они позволяют обрабатывать естественный язык и создавать последовательности текста, прогнозируя каждый следующий символ на основании предыдущей цепочки символов. Реализация такой RNN была взята из библиотеки TensorFlow для языка Python 3, также была использована библиотека NumPy. Работа проводилась в среде Google Colaboratory, которая использовала аппаратный графический ускоритель T4 GPU для обучения модели.

Данные для обучения включали в себя исходных код, написанный на языке JavaScript с использованием библиотек TypeScript и React. Этот код был получен путём автоматической загрузки данных открытых репозиториев с сайта GitHub, и дальнейшей их обработки. Обработка включала в себя удаление всех файлов, кроме имеющих расширения «.js», «.jsx», «.ts» и «.tsx», замену символов переноса строки на токен «<N>» и соединением всего исходного кода в единый файл. Всего файл для обучения включал в себя более 45 млн символов.

2. Модель RNN

2.1. Преобразования и гиперпараметры обучения

Для обучения модели RNN, исходные данные были разбиты на токены и преобразованы

используя слой Keras StringLookup. В дальнейшем, для генерации «рыбного» кода будет применяться обратная функция, восстанавливающая символы из токенов.

Исходный текст был разбит на последовательности из 100 символов, а затем разбит на управляемые последовательности по 64 токена и перемешан, используя TensorFlow Dataset.

Используемая модель Keras включала в себя три слоя:

1. Входной слой Embedding, отображающий все токены символов в вектор с заданными размерами (128).
2. Слой GRU (управляемый рекуррентный блок), с числом нейронов равным 1024;
3. Выходной слой Dense, использующий размер словаря (255) и выводящий логарифмическую вероятность следующего предсказываемого символа в соответствии с текущей моделью.

Описание получившейся модели представлено на рис. 1.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     multiple                  32768

 gru_2 (GRU)                 multiple                  3545088

 dense_2 (Dense)             multiple                  262400

=================================================================
Total params: 3840256 (14.65 MB)
Trainable params: 3840256 (14.65 MB)
Non-trainable params: 0 (0.00 Byte)
```

Рис. 1. Описание используемой модели

В качестве функции потери была выбрана функция SparseCategoricalCrossentropy, которая вычисляет потерю кросс-энтропии между реальными классами и их прогнозами. При обучении использовался оптимизатор Adam (адаптивная оценка момента). Также при обучении были использованы сохраняемые контрольные точки ModelCheckpoint, позволяющие сохранять текущее состояние модели и возобновлять обучение в случае аварийной остановки.

Так как используемый объем исходных данных был достаточно большим, чтобы избежать переобучения, количеством эпох для обучения модели было выбрано 10.

2.2. Результаты обучения

Используя аппаратный графический ускоритель T4 GPU, в среде Google Colaboratory, обучение модели заняло 55 минут. Итоговый loss составил 0.7539.

Для генерации «рыбного» кода, был использован обратный слой StringLookup, восстанавливающий символы из предсказываемых токенов, а также функция генерации, учитывающая текущую последовательность символов и хранящая своё текущее состояние, а также преобразующая специальные токены «<N>» в переносы строк.

Теперь, используя полученную модель, можно генерировать «рыбный» код, начиная с некоторой заготовки. Например, если передать модели на вход текст «`import * as React from 'react';`», можно получить следующую последовательность символов:

```
import * as React from 'react';
import { Icon, ActiOnBase } from '../icons/actions';
```

```
import { MEST } from '..';
import { Cedts } from '../.../src/components/common/Platted';

export default {
    const { canner } = render(<Cannner />);
  }

  const { paylator } = render(<Text />);


import { messageSanger } from 'react-native-size-text-render';
import { LeveContext } from '../../constants';
import React from 'react';
import { createMedia } from '../../../src/medial/utils/useMedia';

export default {
  ename: 'Center',
  cell: {
     enabled: 50px any,
     blace: '#feaker',
};

export default STimeout;

// <reference types="lttw-style-cententer" />
///  <reference types="md" />
/*
// Interface Mattions LICENSE.txt for license information.

export interface IComponent {
  iconstronly = forwardRef<HTMLHTMLF_VRElement,
}) => (
  <IconButton style={{ children, type:
'httpS./styles.child.config.styles.config}'}} />
);

import React from 'react';
import { useFact, StyleSheet, VersionProps, useState } from
'@antdexjs/intl';
import { DefaultChannel } from 'chai';

interface ViewButton {
   /**
   * React as space true;
       */
```

Сгенерированный моделью код не имеет особого смысла, однако, он относительно точно следует используемой структуре и стилизации языка JavaScript с используемыми

библиотеками TypeScript и React, и вполне пригоден для использования в качестве «рыбного» текста.

Заключение

В результате работы был рассмотрен алгоритм генерации «рыбного» исходного кода при помощи рекуррентных нейронных сетей, предложена архитектура сети и гиперпараметров, описана получившаяся модель и приведен пример её работы. Получившаяся модель может быть использована для создания «рыбного» кода в веб-разработке и других областях.

Литература

1. Рекурсивные нейронные сети. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Рекурсивные_нейронные_сети. – Дата обращения 14.04.2024).
2. Генерация текста с помощью RNN. – Режим доступа: https://www.tensorflow.org/text/tutorials/text_generation?hl=ru. – (Дата обращения: 14.04.2024).
3. The Unreasonable Effectiveness of Recurrent Neural Networks. – Режим доступа: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. – (Дата обращения: 14.04.2024).
4. Рекуррентная нейронная сеть (RNN): виды, обучение, примеры. – https://neurohive.io/ru/osnovy-data-science/rekurrentnye-nejronnye-seti/ (Дата обращения: 15.04.2024)

УДК 004

УЯЗВИМОСТИ ВЕБ-ПРИЛОЖЕНИЙ И СПОСОБЫ ИХ ОБНАРУЖЕНИЯ

В. А. Болдорев

Воронежский государственный университет

Введение

Согласно статье Positive Technologies “Уязвимости и угрозы веб-приложений 2020-2021” доля веб-приложений, содержащих уязвимости высокой степени риска, составила 66% в 2020 году, а в 2021 году — 62%, что значительно больше показателя 2019 года. Стоит отметить, что все исследуемые веб-приложения в 2021 году имели уязвимости с низкой степенью риска.

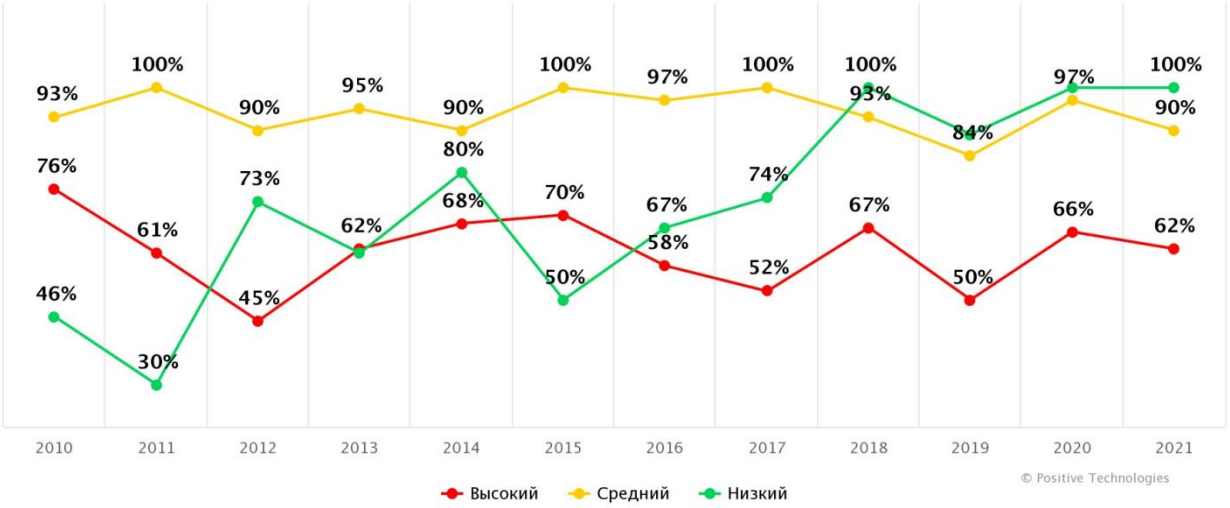


Рис. 2 Доли веб-приложений с уязвимостями различной степени риска

Во всех исследуемых в 2020 - 2021 годах веб-приложениях были выявлены уязвимости, связанные с нарушением контроля доступа. Их наличие приводит к несанкционированному доступу к конфиденциальной информации, модификации или удалению данных.

Исходя из вышесказанного, обнаружение и устранение уязвимостей в веб-приложениях становится приоритетной задачей для разработчиков.

1. Постановка задачи

Цель работы – анализ методов обнаружения уязвимостей и их применение в процессе разработки веб-приложений.

Для достижения поставленной цели в работе предполагается решение комплекса задач:

- изучение распространенных уязвимостей, с которыми можно столкнуться при разработке веб-приложений, с целью получения полного представления об их типах и потенциальных последствиях;
- изучение существующих методов обнаружения уязвимостей и их применение в контексте разработки веб-приложений;

- оценка эффективности и надежности различных методов обнаружения уязвимостей в контексте разработки веб-приложений, а именно: проведение экспериментов, сравнительный анализ результатов и проверка на реальных веб-приложениях.

2. Распространенные уязвимости веб-приложений

Согласно статье OWASP “Top 10 Web Application Security Risks” выделяют следующие распространенные уязвимости:

1. Нарушение контроля доступа (Broken Access Control).
2. Уязвимость к криптографическим атакам (Cryptographic Failures).
3. Инъекции (Injection).
4. Небезопасный дизайн (Insecure Design).
5. Неверная конфигурация безопасности (Security Misconfiguration).
6. Применение в приложениях компонентов с неизвестными уязвимостями (Vulnerable and Outdated Components).
7. Ошибки идентификации и аутентификации (Identification and Authentication Failures).
8. Нарушение целостности данных программного обеспечения (Software and Data Integrity Failures).
9. Ошибки мониторинга и недостаточное ведение журналов (Security Logging and Monitor Failures).
10. Подделка запросов со стороны сервера (Server-Side Request Forgery (SSRF)).

3. Методы обнаружения уязвимостей веб-приложений

Определяют следующие типы используемых в процессе разработки методов тестирования веб-приложения на уязвимости:

1. Static Application Security Testing (SAST) - статический анализ кода.
2. Dynamic Application Security Testing (DAST) – динамический анализ.
3. Interactive Application Security Testing (IAST) - интерактивное тестирование.
4. Software Composition Analysis (SCA) – анализ сторонних компонентов ПО.

3.1. Static Application Security Testing

Static Application Security Testing (SAST) – метод, позволяющий обнаруживать уязвимости без реального запуска исследуемого приложения. Данный подход принято называть тестированием по методу белого ящика.

Данный метод позволяет выявить подобные проблемы: синтаксические ошибки, математические ошибки, явное хранение паролей в коде и т.д.

Преимущества SAST:

- возможность интеграции статического анализа в процесс разработки в IDE;
- работает без запуска приложения;
- покрывает всю кодовую базу приложения;
- указание на точное расположение подозрительного фрагмента кода.

Минусы SAST:

- большое количество ложных срабатываний;
- отсутствует возможность проанализировать библиотеки и фреймворки, если

отсутствует доступ к их исходному коду;
- привязка к определенному языку программирования.

3.2. Dynamic Application Security Testing

Dynamic Application Security Testing (DAST) – метод, анализирующий запущенное приложение путем имитации вредоносных внешних атак с попытками эксплуатации распространенных уязвимостей. Данный подход принято называть тестирование по методу черного ящика.

Преимущества DAST:

- отсутствует привязка к языкам программирования;
- небольшое количество ложных срабатываний;
- не нужен исходный код приложения.

Минусы DAST:

- часто отсутствует возможность максимально имитировать рабочую среду приложения;
- отсутствует возможность полного покрытия кода, так как некоторые модули приложения могут быть недоступны при запуске на моковых данных.

3.3. Interactive Application Security Testing

Interactive Application Security Testing (IAST) – метод, который позволяет анализировать код, но, в отличии от SAST, делает это в режиме реального времени и в интерактивном режиме, аналогичном инструментам DAST.

Преимущества IAST:

- выполнение статического анализа, а также тестирование исполняемого кода.

Минусы IAST:

- инструменты IAST замедляют работу приложения;
- крайне тяжело подготовить сценарии тестирования, позволяющие достичь высокого покрытия кода.

3.4. 4. Software Composition Analysis

Software Composition Analysis (SCA) – метод, который позволяет анализировать сторонние компоненты ПО. Основная цель данного подхода – проверка уязвимостей в сторонних компонентах, используемых в составе приложения. Выделяют несколько подходов к поиску уязвимостей:

- непосредственный анализ сторонних компонентов (в том числе при использовании SAST, если есть доступ к исходному коду того или иного компонента);
- сбор информации из различных баз.

Заключение

На данном этапе исследовательской работы был произведен сбор информации по основным методикам тестирования веб-приложений на уязвимости, выявлены их положительные и отрицательные стороны. Выполнен обзор наиболее распространенных уязвимостей веб-приложений.

Дальнейшая перспектива исследования – применение методов тестирования на существующем веб-приложении, анализ и выбор необходимых инструментов в зависимости от языка программирования, на котором написано веб-приложение, анализ результатов тестирования, исправление найденных уязвимостей.

Литература

1. Воронин В.В. Об анализе методов обнаружения уязвимостей внутри WEB-приложений // Молодежь и XXI век – 2019. – 2019. – Т. 3. – С. 124-126.
2. Акилов М.В.; Ковцур М.М.; Несудимов Е.Ю.; Потемкин П.А. Исследование методик обнаружения уязвимостей WEB-приложений IAST и SAST // Информационная безопасность регионов России (ИБРР – 2021). – 2021. – С. 378-379.
3. Фатхи В.А.; Дьяченко Н.В. Тестирование безопасности приложений // Инженерный вестник Дона. – 2021. – №5.
4. Уязвимости и угрозы веб-приложений в 2020–2021 гг. – URL [электронный ресурс]: https://www.ptsecurity.com/ru-ru/research/analytics/web-vulnerabilities-2020-2021/ (дата обращения 17.12.2023).
5. Виды Application Security Testing. Как не запутаться среди SAST, DAST и IAST. – URL [электронный ресурс]: https://habr.com/ru/companies/pvs-studio/articles/676718/ (дата обращения 26.12.2023).
6. Top 10 Web Application Security Risks. – URL [электронный ресурс]: https://owasp.org/www-project-top-ten/ (дата обращения 01.04.2024).
7. OWASP Top Ten и Software Composition Analysis (SCA). – URL [электронный ресурс]: https://habr.com/ru/companies/pvs-studio/articles/585030/ (дата обращения 02.04.2024).

УДК 004.056.5, 004.9

СОВРЕМЕННЫЕ СПОСОБЫ ГЕНЕРАЦИИ И ОБМЕНА ТОКЕНОВ ДОСТУПА ДЛЯ ВЕБ-ПРИЛОЖЕНИЙ

П. А. Большаков, К. Г. Резников

Воронежский государственный университет

Введение

Большинство современных сайтов и веб-приложений [1] имеют возможность или требуют создать персональный профиль пользователя или учетную запись. Этот процесс называется идентификацией, другими словами это заявление о том, кем именно является человек. В качестве предоставляемых данных могут быть имя, адрес электронной почты, дата рождения и т.д. Авторизация – это проверка наличия прав доступа для конкретного пользователя на определенный ресурс. Процесс входа в свой профиль – это аутентификация, то есть предоставление доказательств того, что человек действительно является тем, кем идентифицировался.

Существует множество способов аутентификации, такие как форма для ввода логина и пароля, указание телефона с подтверждением кода из СМС сообщения, регистрация через сторонние сервисы и т.п. Далее сайты не требуют вводить все повторно после каждого запроса. Процесс автоматической аутентификации может быть реализован разными способами. Например, самый небезопасный из них – сохранение секретных данных в браузере и их отправка на сервер при каждом запросе. Однако для этих целей уже давно разработаны более надежные механизмы и протоколы. Самый распространенный из них – аутентификация при помощи токенов.

Токен – это уникальный ключ, состоящий из набора символов, который может являться результатом шифрования или сгенерированным паролем.

Способов авторизации и аутентификации с помощью токена существует большое количество, о чем далее говорится более подробно. Данные пользователя отправляются на сервер приложения или на сервер аутентификации только один раз при попытке входа, а в ответе на запрос, сервер возвращает клиенту токен, который сохраняется в браузере. При всех последующих запросах будет прикрепляться именно этот токен, в котором содержится обезличенная информация, например, идентификатор или прочие необходимые для приложения данные профиля. Обращение к серверу аутентификации или базе данных (БД) после этого для идентификации не требуется. Каждый токен подписывается криптографически, благодаря чему сервер может ему доверять. При необходимости получения более подробной или конфиденциальной информации о пользователе, приложение воспользуется идентификатором и самостоятельно обратится к базе данных.

JSON Web Tokens (JWT) – это наиболее популярная технология работы с токенами для веб-аутентификации и управления сеансами пользователей в современных веб-приложениях [2]. Есть множество статей о преимуществах JWT, однако очень немногие разработчики программного обеспечения знакомы с потенциальными опасностями и неэффективностью использования таких токенов.

В данной статье рассматривается то, как JWT могут сделать веб-сайты уязвимыми для различных угроз безопасности, если ими не управлять должным образом, методы решения основных проблем и возможные альтернативы. Поскольку JWT широко используются в

методах аутентификации, управления сеансами и контроля доступа, эти недостатки могут поставить под угрозу весь веб-сайт и его пользователей.

1. Устройство JSON Web Token

JSON Web Token является стандартизированный форматом для безопасной передачи данных JSON с криптографической подписью между системами. Потенциально они могут включать данные любого типа, но чаще всего используются для передачи данных о пользователях в рамках процедур аутентификации, управления сеансами и контроля доступа. При этом все данные, необходимые серверу, сохраняются на стороне клиента.

JWT состоит из трех частей:

– Заголовок, содержащий способ шифрования и дополнительную информацию;
– *Claims*, набор полей данных пользователя;
– Подпись для криптографического шифрования.

Пример токена и блоков представлены на рис. 1. Первые два блока представлены в JSON-формате и дополнительно закодированы в формат base64. Это значит, что расшифровать их может любой желающий, если получит доступ к JWT. Заголовок включает в себя информацию о токене, т.е. метаданные, такие как тип токена и используемый алгоритм подписи, например, HMAC SHA256 или RSA. Набор полей содержит произвольные пары имя/значения, притом стандарт JWT определяет несколько зарезервированных имен: *iss*, *aud*, *exp* и другие.

Подпись является последним и самым важным компонентом JWT. Она создается на основе первых двух блоков и секретного ключа, который есть только у сервера приложения и сервера аутентификации. Изменение хотя бы одного байта заголовка, набора полей или ключа приводит к абсолютно другой комбинации, благодаря чему получить секретный ключ из подписи почти невозможно. Она служит для того, чтобы сервер мог проверить надежность полученных данных. Поэтому необходимо ответственно подходить к выбору секретного ключа, он не должен быть слишком коротким и слишком простым, чтобы исключить возможность простого перебора.

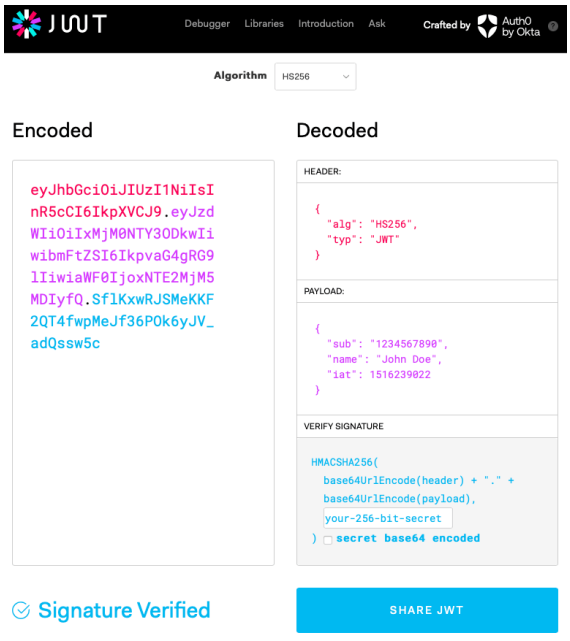


Рис. 1. JWT с расшифровкой

JWT прикрепляется к заголовку Authorization HTTP запроса [3] при отправке на сервер (рис. 2). Приложение сравнивает подпись токена и сгенерированную подпись на основе полученных данных и секретного ключа. Если они совпадают, значит полученным данным можно доверять, и они не были изменены. Также у каждого токена есть срок действия, по истечении которого он становится недействительным. Обычно оно составляет от 5 до 30 минут.

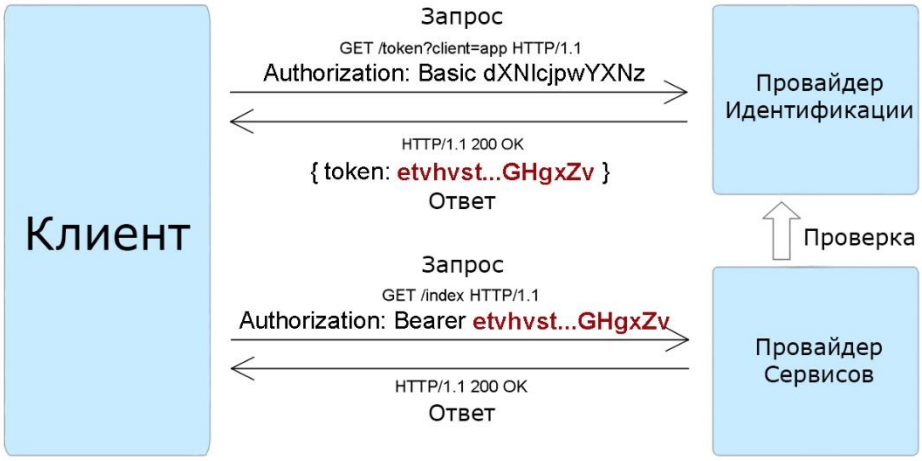


Рис. 2. Схема аутентификации клиента при помощи токена

JWT дает возможность серверу без сохранения данных сеанса удостоверится в том, что запрос сделан конкретным пользователем, проверяя токен в запросе, а не запрашивая пароль каждый раз. При этом вызов базы данных полностью исключается. С другой стороны, JWT являются действительными на всем протяжении срока действия, даже если пользователь уже вышел из системы. Также уже действующие токены невозможно отозвать или признать недействительными, что порождает ряд проблем.

2. Опасность использования JWT

Хотя использование JWT исключает поиск в базе данных, оно привносит в процесс проблемы безопасности и другие сложности [4]. Самая большая проблема заключается в том, что токен будет продолжать работать до тех пор, пока не истечет срок его действия, и у сервера нет простого способа его отозвать. Это может быть чрезвычайно опасно в следующих ситуациях:

JWT может продолжать действовать в течение любого времени, установленного в качестве его срока действия, даже после того, как пользователь вышел из системы. Это означает, что, если кто-то получит доступ к токену в течение этого времени, он сможет продолжать им пользоваться до истечения срока его действия.

Точно также модерация не сможет заблокировать пользователя в системе, поскольку он будет продолжать иметь доступ к серверу до истечения срока действия токена.

Предположим, что пользователь был администратором, уровень которого был понижен до обычного пользователя с более низкими привилегиями. Опять же, это не вступит в силу немедленно, и пользователь останется администратором до истечения срока действия токена.

Кроме того, многие библиотеки, реализующие JWT, имели множество проблем с безопасностью. Помимо этого, многие реальные программы требуют, чтобы серверы сохраняли IP-адрес пользователя и отслеживали API-интерфейсы для регулирования скорости

и внесения в белый список IP-адресов. В результате все равно появляется необходимость обращаться к базе данных. Таким образом, использование JWT не всегда гарантирует отсутствие сохранения состояния.

3. Альтернативные способы аутентификации

Существуют и другие способы аутентификации для веб-приложений, такие как HTTP authentication, сертификаты и различные токены.

1. HTTP authentication – протокол, часто используемый в корпоративной сфере. На запрос пользователя к закрытому ресурсу без авторизации, сервер вернет код 401 с заголовком *WWW-Authenticate* (рис. 3). На это браузер запросит учетные данные пользователя. К каждому последующему запросу будет добавляться заголовок *Authorization*, в котором будут передаваться данные аутентификации в зашифрованном или незашифрованном виде в зависимости от уровня безопасности протокола. Важно отметить, что в HTTP-аутентификации пользователь не имеет возможности выхода из веб-приложения, за исключением перезапуска браузера, а также при проверке всегда требуется подключение к безе данных.

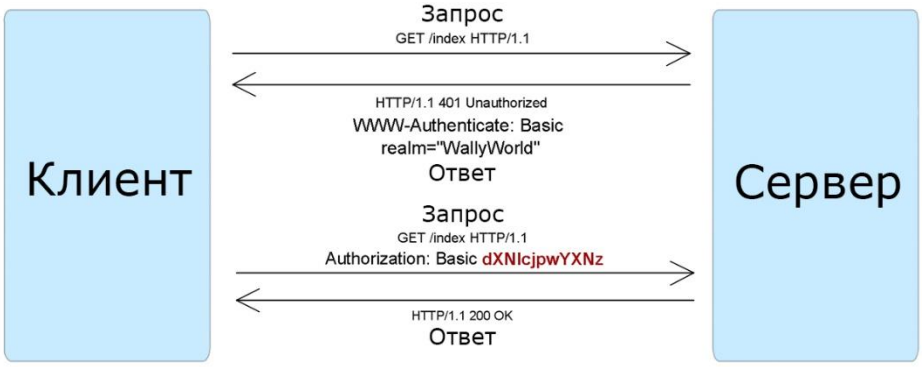


Рис. 3. Пример HTTP аутентификации с использованием Basic схемы

2. Сертификат – набор атрибутов, по которым идентифицируют владельца. Он подписывается системой CA, которая является гарантом его подлинности (рис. 4). Сертификаты криптографически связаны с закрытыми ключами, хранящимися у их владельцев, что позволяет точно подтвердить факт владение. Для аутентификации в веб-приложениях обычно используют сертификаты стандарта X.509, которые являются частью протокола SSL/TLS. Это намного более надежный способ, чем аутентификация при помощи паролей, благодаря созданию цифровой подписи. Однако такой способ малодоступен в широких кругах в связи с трудностями распространения и поддержки сертификатов.

Рис. 4. Использование сертификата для аутентификации

3. Simple Web Token (SWT) – это самый простой из стандартизированных форматов, который представляет собой набор пар имя/значение в кодировании HTML form. Токен подписывается только при помощи симметричного ключа.

4. Security Assertion Markup Language (SAML) – это стандарт, определяющий токены в XML-формате, включающие информацию об эмитенте, о субъекте, необходимые условия для проверки токена и набор дополнительных утверждений о пользователе. Подпись таких токенов осуществляется с помощью ассиметричной криптографии. К тому же, SAML-токены содержат механизм для подтверждения владения токеном, позволяющий при использовании незащищенных соединений предотвратить перехват токенов через MITM атаки. Однако XML не имеет естественного сопоставления документов с объектами языков программирования, а также может занимать больше памяти, чем JSON, из-за чего токены придется отправлять в теле сообщения, которое содержат только такие запросы, как POST, PUT, PATCH и др.

5. WS-Trust – это стандарт, описывающий интерфейс сервиса авторизации Secure Token Service (STS). Данный сервис поддерживает создание, обновление и аннулирование токенов и работает по протоколу SOAP. Кроме того, стандарт поддерживает использование токенов различного формата, но на практике используются в основном SAML.

6. WS-Federation – это механизмы взаимодействия сервисов между организациями, в том числе протоколов обмена токенов. WS-Federation расширяет функции и интерфейс STS.

WS-Trust и WS-Federation были разработаны такими компаниями, как Microsoft, IBM, VeriSign и др. Эти стандарты, наряду с SAML, в основном используются в корпоративных сценариях из-за своей сложности [5].

Сопоставление преимуществ и недостатков всех перечисленных выше способов аутентификации относительно друг друга приведены в табл. 1.

Таблица 1

Таблица сравнения способов аутентификации

Способ аутентификации	**Преимущества**	**Недостатки**
HTTP Authentication	1. Поддерживается всеми браузерами; 2. Прост в реализации.	1. Небезопасен; 2. Требует постоянных вызовов БД; 3. Отсутствует стандартный способ выхода.
Сертификат	1. Высокая надежность; 2. Высокая безопасность.	1. Малодоступен в широких кругах из-за трудностей с распространением и поддержкой.
Simple Web Token (SWT)	1. Формат кодирования HTML form.	1. Нет возможности использовать пару открытого/закрытого ключей.
Security Assertion Markup Language (SAML)	1. Наличие механизма для подтверждения владения токеном.	1. XML не имеет естественного сопоставления документов с объектами языков программирования; 2. Может требовать большого размера памяти, в сравнении с JSON; 3. Токены необходимо отправлять в теле сообщения.
JSON Web Token (JWT)	1. Прост в реализации; 2. JSON имеет естественное сопоставление с объектами языков программирования; 3. Малый размер пакета; 4. Возможность прикрепления к заголовкам запросов.	1. При неправильном использовании может быть причиной уязвимостей.
WS-Trust	1. Поддерживает обновление токенов; 2. Поддерживает аннулирование токенов; 3. Возможность использования токенов различного формата.	1. Сложный в реализации.

WS-Federation	1. Расширяет функции и интерфейс сервиса STS, описанного в стандарте WS-Trust.	1. Сложный в реализации.

4. Компромисс в реализации

Одним из типичных решений проблем JWT является сохранение базы данных отозванных токенов и ее проверка при каждом вызове. Если токен находится в этом отозванном списке, пользователю запрещается доступ к ресурсу. Также из БД периодически должны удаляться токены, срок действия которых уже истек. Но в данном случае необходимо дополнительно делать запросы к БД, чтобы узнать, был ли отозван токен. Такой подход полностью противоречит смыслу JWT.

Ответ заключается в том, чтобы использовать не основную базу данных приложения, а отдельную небольшую БД, вызов которой осуществляется значительно быстрее. Примером такого решения является использование Redis совместно с JWT. Это позволяет устранить большинство угроз безопасности, описанных ранее, при этом сохраняя преимущества JWT.

Заключение

Практически все современные веб-приложения требуют процесса идентификации и аутентификации пользователей. Для обеспечения целостности данных и защиты от взлома, важно грамотно подходить к вопросу в выборе способа реализации защиты.

В статье были представлены несколько способов аутентификации, а также описаны их достоинства и недостатки. Невозможно найти универсальное решение для любой задачи, поэтому всегда нужно подходить к вопросу с учетом особенностей и специфики конкретного проекта.

Литература

1. Резников К. Г. Разработка программного обеспечения для визуализации трехмерных поверхностей в веб-браузере / К. Г. Резников, С. Н. Медведев // Вестник ВГТУ. Том 17, №6. – Воронеж: ВГТУ, 2021 - С. 13-19.

2. Introduction to JSON Web Tokens – Режим доступа: https://jwt.io/introduction – (Дата обращения: 10.03.2024)

3. Петров Р.В. Современные технологии разработки веб-приложений / Р.В. Петров, Д.В. Вагин. – Новосибирск : НГТУ, 2023. – 53 с.

4. Хоффман Э. Безопасность веб-приложений. Разведка, защита, нападение / Э. Хоффман. – СПб : Питер, 2021. – 336 с.

5. Умрихин Е.Д. Разработка веб-приложений с помощью ASP.Net Core MVC / Е.Д. Умрихин. – СПб : БХВ-Петербург, 2023. – 413 с.

УДК 581.19

РАЗРАБОТКА АЛГОРИТМИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ ПРОЕКТА С РЕКОМЕНДАТЕЛЬНЫМИ ЗАВИСИМОСТЯМИ МЕЖДУ РАБОТАМИ

Ю. В. Бондаренко, М. В. Чусов

Воронежский государственный университет

Введение

В настоящее время одной из самых распространенных методологий менеджмента стала проектная деятельность. Проекты могут помочь в достижении целей не только отдельным компаниям и частным лицам, но и целым государствам ([1]-[3]).

Среди большого многообразия типов проектов особое место занимают проекты с рекомендательными зависимостями. В проектах такого типа зависимости отражают предпочтительность завершения одних работ до начала других ([4]-[5]). Проекты с рекомендательными зависимостями между работами, как и любые проекты, нуждаются в компетентном и продуктивном управлении. Использование различных инструментов управления проектами не только помогает достигать поставленных целей в срок и эффективно расходовать ресурсы, но и дает неоспоримое преимущество компаниям, оказавшимся в условиях высокой конкуренции.

Ключевым фактором успеха любой проектной деятельности является наличие четкого заранее определенного плана, минимизации рисков и отклонений от этого плана, а также эффективного управления изменениями в нём. Таким планом выступает календарный план проекта. Именно календарный план является ключевым документом проекта и от качества его разработки в значительной мере зависит успех реализации проекта в целом. Календарное планирование представляет собой процесс составления календарного плана работ в проекте, который включает в себя определение перечня работ проекта, их логические взаимосвязи, исполнителей и продолжительности работ, а также ресурсные, временные и внешние ограничения и на их основе – сроки выполнения работ проекта ([4]-[5]).

Поскольку проекты с рекомендательными зависимостями наиболее востребованы в бурно развивающейся IT-сфере, то разработка моделей и механизмов календарного планирования таких проектов является актуальной задачей.

Целью настоящего исследования является совершенствование процесса календарного планирования проекта с рекомендательными зависимостями между работами посредством разработки и программной реализации алгоритма построения сетевого графика с минимальной продолжительностью проекта, основанного на применении математического инструментария.

Достижение поставленной в исследовании цели потребовало решения ряда задач, среди которых наиболее значимыми и обладающими научной новизной являются:

1. Разработка алгоритма календарного планирования проекта с рекомендательными зависимостями между работами, обеспечивающего формирование сетевого графика с минимальной продолжительностью проекта.
2. Разработка программного продукта, обеспечивающего вычислительную поддержку реализации алгоритма.

1. Алгоритм формирования календарного плана с минимальной

продолжительностью проекта

Рассмотрим некоторый проект, включающий n работ, занумерованных в некотором порядке. Время выполнения каждой работы i по технологии проекта обозначим как τ_i. Полагаем, что для проекта можно построить сетевой график типа «вершина-работа», где непосредственное предшествование работы i работе j отражено в виде дуги (i, j). Начальный сетевой график формируется в соответствии с технологией выполнения работ. Однако, как описано выше, в проекте предполагается возможность отклонения от принятой технологической зависимости, что и означает «мягкие» зависимости или зависимости рекомендательного типа. В случае, если зависимость (i, j) нарушается (то есть, работа j начата до окончания работы i), продолжительность выполнения работы j увеличивается на величину a_{ij}. Требуется составить календарный план выполнения работ, обеспечивающий минимальную продолжительность проекта с рекомендательными зависимостями. Предлагаемый алгоритм основан на теоретическом исследовании, представленном в работе [4], и содержит два основных этапа.

Этап 1. Присваиваем всем работам сетевого графика начальные индексы $\lambda_i = \tau_i$, $i = 1,..., n$.

Этап 2. Для каждой работы $i = 1,..., n$ выполним следующие шаги:

Шаг 1. Обозначим через Q_i – множество работ, которые предшествуют работе i, то есть Q_i – это множество таких работ j, для которых в сетевом графике существует дуга (i, j).

Шаг 2. Обозначим через m_i число дуг, которые заходят в вершину i. Число m_i является мощностью множества Q_i.

Шаг 3. Рассмотрим все подмножества R_i множества Q_i, состоящие из m_i элементов (очевидно их количество равно 2^{m_i}). Для каждого R_i найдем число $t_i(R_i)$ по формуле:

$$t_i(R_i) = \tau_i + \max_{j \in R_i} \lambda_i + \sum_{j \in R} a_{ji}.$$

Затем определим новое значение индекса λ_i рассматриваемой вершины i по формуле:

$$\lambda_i = \min_{R_i} t_i(R_i).$$

Алгоритм заканчивается, когда все индексы установятся. Факт того, что алгоритм не является бесконечным следует из того, что последовательность индексов для каждого i является возрастающей. Также значения λ_i ограничены величиной T, которая вычисляется по формуле:

$$T = \tau_i + \sum_{j \in Q_i} a_{ji}.$$

Таким образом, установившиеся индексы λ_i равны минимальным срокам окончания соответствующих работ. Этот факт опирается на теорему доказанную в [4].

2. Практическая реализация

В качестве инструмента для построения календарных планов было разработано программное обеспечение, которое позволяет оптимизировать календарные планы для

минимальной продолжительности проекта или минимальных дополнительных затрат при рекомендательных зависимостях между работами.

Программный продукт, представленный в данной работе реализован в среде разработки PyCharm (версия: 2022.1.2, сборка: 221.5787.24). Для разработки использовался язык Python 3.8. Методы библиотек NumPy и Pandas использовались для выгрузки данных из файлов и выполнения математических операций. Библиотека PyQt5 использовалась для создания графического интерфейса.

Рассмотрим работу программы на реальных данных. В качестве примера возьмем проект по разработке веб-приложения.

Сетевой график проекта с рекомендательными зависимостями представлен на рис. 1. Число a_{ij} обозначено в днях как первая координата упорядоченной пары над каждой дугой, а вторая координата – стоимость удорожания работ при изменении порядка пары работ. То есть, при невыполнении зависимости (5, 7) увеличение продолжительности работы 7 составит 1 день, а увеличение затрат на выполнение работы 7 составит 60 тыс. рублей.

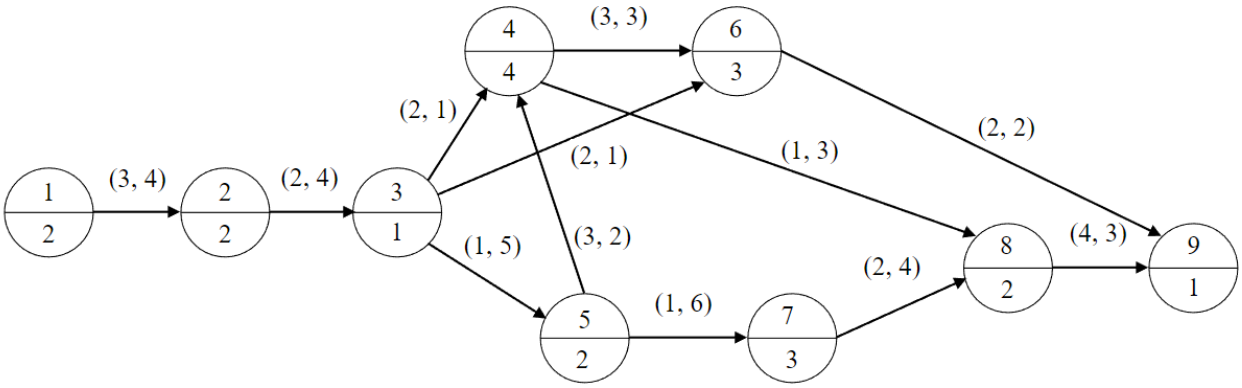


Рис. 1. Сетевой график проекта

Описание работ проекта приведено в табл. 1.

Таблица 1

Описание работ проекта

№	Работа
1	Сбор и анализ требований
2	Написание и согласование технического задания
3	Разработка архитектуры приложения
4	Формирование макета приложения
5	Разработка БД
6	Разработка front-end
7	Разработка back-end
8	Тестирование
9	Оформление документации приложения

Откроем программу и загрузим файл с начальными данными для определения календарного плана с минимальной продолжительностью проекта (рис. 2). После загрузки данных нажмем кнопку «Решение».

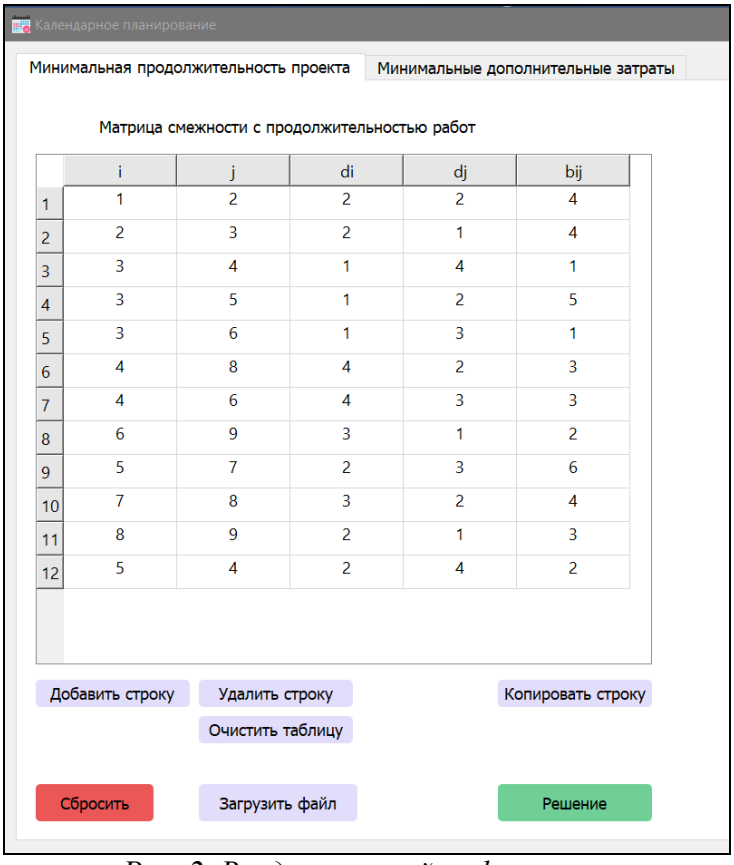
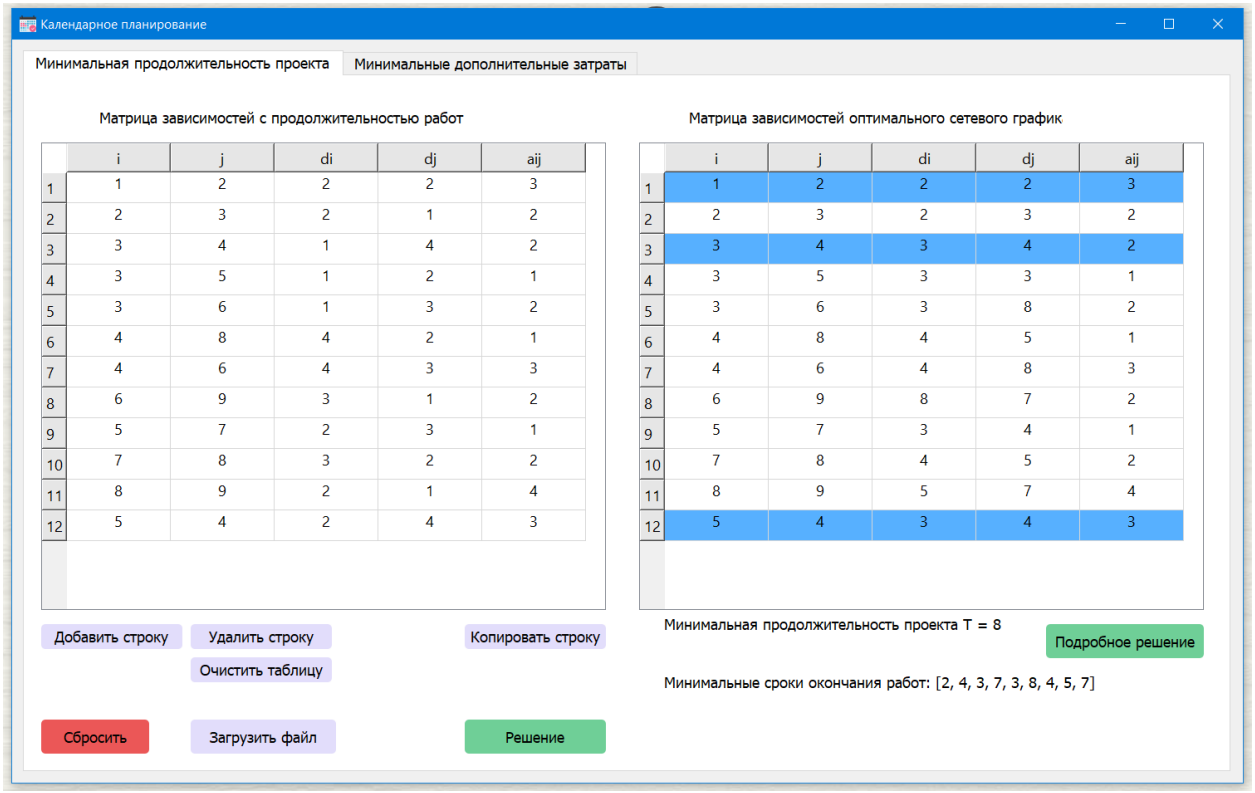


Рис. 2. Ввод начальной информации

После нажатия кнопки «Решение» получаем таблицу, содержащую матрицу зависимостей оптимального сетевого графика (рис. 3).



	i	j	di	dj	aij
1	1	2	2	2	3
2	2	3	2	1	2
3	3	4	1	4	2
4	3	5	1	2	1
5	3	6	1	3	2
6	4	8	4	2	1
7	4	6	4	3	3
8	6	9	3	1	2
9	5	7	2	3	1
10	7	8	3	2	2
11	8	9	2	1	4
12	5	4	2	4	3

	i	j	di	dj	aij
1	1	2	2	2	3
2	2	3	2	3	2
3	3	4	3	4	2
4	3	5	3	3	1
5	3	6	3	8	2
6	4	8	4	5	1
7	4	6	4	8	3
8	6	9	8	7	2
9	5	7	3	4	1
10	7	8	4	5	2
11	8	9	5	7	4
12	5	4	3	4	3

Рис. 3. Результат работы программы

Итак, минимальная продолжительность проекта составит 8 дней. Минимальные сроки свершения работ 1, 2, …, 9 составят соответственно 2, 4, …, 7 дней.

Выделенные цветом зависимости $(1, 2), (2, 3), (3, 4), (5, 4)$ стали жесткими, значит делаем вывод о том, что их следует учесть в календарном плане проекта. Полученный сетевой график с жесткими зависимостями представлен на рис. 4. Диаграмма Ганта представлена на рис. 5.

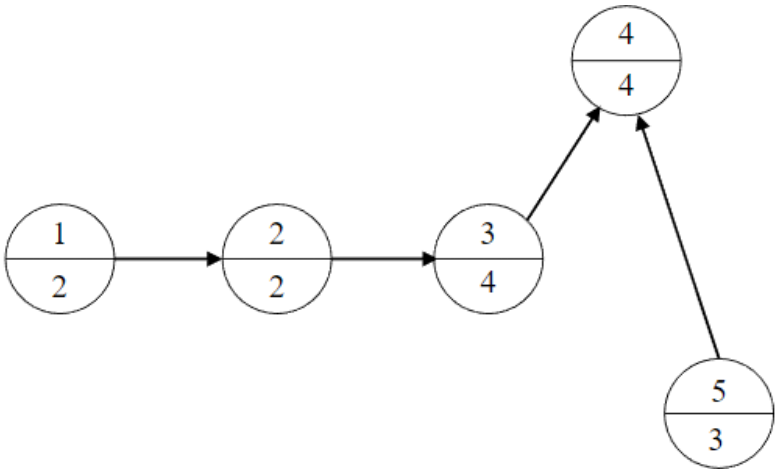


Рис. 4. Сетевой график с жесткими зависимостями

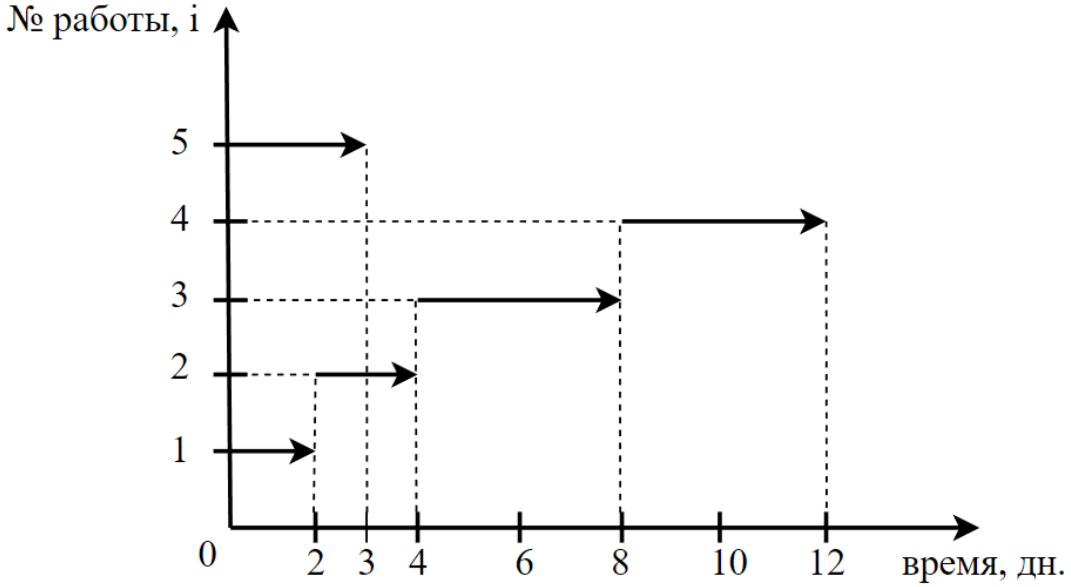


Рис. 5. Диаграмма Ганта

Таким образом делаем вывод о том, что работу 2 предпочтительно начать только после окончания работы 1, работу 3 предпочтительно начать после окончания работы 2, работу 4 предпочтительно начать после окончания работ 3 и 5.

Заключение

Результатом исследования является разработка программно-алгоритмического обеспечения формирования календарного плана проекта с рекомендательными зависимостями. Основу предлагаемого комплекса составляет алгоритм поддержки формирования календарного плана с минимальными дополнительными затратами для проектов с рекомендательными зависимостями. Для реализации данных алгоритмов на практике разработан программный продукт на языке программирования Python.

Литература

1. Полковников А.В. Управление проектами. Полный курс MBA / А.В. Половников, М..Ф. Дубовик. – Москва : Олимп-Бизнес, 2021. – 552 с.
2. Минкевич А. Проджект-менеджмент : Как быть профессионалом // А. Минкевич, С. Дерцап. – Москва : Интеллектуальная литература, 2020. – 232 с.
3. Pasek A. Present project management process / A. Pasek // Science and World. – 2021. – № 12-2 (100). -– С. 53 – 66.
4. Алферов В.И. Прикладные задачи управления строительными проектами / В.И. Алферов, С.А. Баркалов, В.Н. Бурков [и др]. – Воронеж : Воронеж. гос. арх. – строит. ун-т. 2008. – 766 с.
5. Бондаренко Ю.В. Математические методы поддержки сетевого анализа проекта и оценки риска планирования при нечеткой информации о продолжительностях работ / Ю.В. Бондаренко, Е.В. Васильчикова // Вестник Воронежского государственного университета. Серия: системный анализ и информационные технологии. – 2023. – № 2. – С. 100-111.

УДК 81’322.2

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА ФОНЕТИЧЕСКОЙ ЗНАЧИМОСТИ СЛОВ

К. В. Борисенко

Воронежский государственный университет

Аннотация. Работа посвящена описанию программных средств для фоносемантического анализа слов, основанного на изучении звуковых и смысловых ассоциаций, которые вызывают слова у носителей языка. Рассматривается подход к анализу семантики слов на основе их фонетических характеристик.
Ключевые слова: фоносемантика, семантика, фонетическая значимость, разработка.

Введение

В лингвистике исследователи постоянно стремятся расширить знания о структуре и функциональности языка. Ключевую роль играют подходы, способные объединить различные аспекты словообразования и семантики. Одним из таких подходов является теория фоносемантики, которая фокусируется на взаимосвязи между фонетическими характеристиками слов и их семантическим содержанием.

Анализ фонетической значимости слов является вычислительно сложной задачей. Для анализа одного слова требуется значительное количество операций, что делает такую задачу практически неосуществимой без автоматизации расчётов. Для уменьшения трудоемкости работы и обеспечения необходимого качества результатов обработки информации необходимо разрабатывать специальные инструментальные средства автоматизации исследований.

Актуальность работы заключается в том, что автоматизация позволит существенно ускорить и упростить процесс исследований в области фоносемантики. Разработка программного комплекса для фоносемантического анализа слов русского языка представляет собой важный шаг в направлении создания инструментов, специально адаптированных для потребностей исследователей в этой области.

1. Теоретическая база для создания программных средств

Фоносемантика — это теория о взаимосвязи между звуковой формой слова и его семантическим значением. Советский филолог А. П. Журавлев разработал метод, позволяющий оценить воздействие звучания слова на носителей языка [1].

В результате эксперимента была составлена обширная таблица с оценкой каждой звукобуквы (термин, введенный Журавлевым) по 25 шкалам с характеристиками (табл. 1).

Таблица 1

Шкалы для оценки характеристик звукобукв

хороший	плохой
большой	маленький
нежный	грубый
женственный	мужественный

светлый	темный
активный	пассивный
простой	сложный
сильный	слабый
горячий	холодный
быстрый	медленный
красивый	отталкивающий
гладкий	шероховатый
легкий	тяжелый
веселый	грустный
безопасный	страшный
величественный	низменный
яркий	тусклый
округлый	угловатый
радостный	печальный
громкий	тихий
длинный	короткий
храбрый	трусливый
добрый	злой
могучий	хилый
подвижный	медлительный

При этом шкала оценок выглядит следующим образом (рис. 1):

Рис. 1. Шкала оценок звуков (слов)

Если звук получает среднюю оценку от 1 до 2,5, то для характеристики качественного ореола слова выбирается левый признак шкалы (например, «хороший»); если средняя оценка от 3,5 до 5, то в качестве характеристики выбирается правый признак (например, «плохой»). Посередине находится нейтральная, или «серая» зона.

Журавлевым было предложено оценивать слово целиком, а не отдельные звуки, по тем же шкалам, что и для звуков.

После набора статистически значимых оценок для каждого звука и серии дополнительных экспериментов Журавлёв вывел формулы, позволяющие рассчитать коэффициент информативного веса каждого звука и фонетическое значение слова в соответствии со шкалами [2].

$$K_i = \frac{P_{\max}}{P_i},\ i = 1\ldots n,\ \ K_1 = \frac{4 \cdot P_{\max}}{P_1},\ \ K_{уд} = \frac{2 \cdot P_{\max}}{P_{уд}}; \qquad (1)$$

где K_i — коэффициент для учета информативного веса каждого звука в слове (K_1 — коэффициент для начального звука в слове, $K_{уд}$ — для ударного), P_i — частотность звука в речи (P_1 — частотность начального звука в слове, $P_{уд}$ — частотность ударного звука), $P_{\max}$ — максимальная частотность звука в слове, а n — количество звуков в слове.

Когда информативный вес всех звуков слова найден, фонетическая значимость слова вычисляется усреднением взвешенных оценок содержательности составляющих данное слово звуков:

$$F = \frac{\sum F_i K_i}{\sum K_i} \quad i = 1 \ldots n \qquad (2)$$

где F — фонетическая значимость слова, а F_i — фонетическая значимость очередного звука.

2. Пример определения фонетической значимости слов

Вычисление фонетической значимости слова «лик» по шкале «красивый — отталкивающий» (табл. 2). Фонетическая значимость и частотность звукобукв берется из таблиц, составленных Журавлевым в ходе экспериментов.

Таблица 2

Вычисление фонетической значимости слова «лик»

Звукобуквы	Исходные данные		Промежуточные результаты вычислений		
	F_i	P_i	P_{max}/P_i	K_i	F_iK_i
л’	1,9	0,017	1,76	7,04	13,38
и	2,0	0,015	2,00	4,00	8,00
к	3,4	0,030	1,00	1,00	3,40
			Σ	12,04	24,78

$$F = \frac{24{,}78}{12{,}04} = 2{,}1 \qquad (3)$$

Окончательный результат F=2,1, т. е. фонетическая значимость звукового комплекса «л'ик» оценивается признаком «красивый».

Рассмотрим теперь сравнительную оценку слов «лицо — лик — морда — харя — рыло» по шкале «красивый – отталкивающий» (табл. 3).

Таблица 3

Сравнительная оценка слов «лицо — лик — морда — харя — рыло»

Шкала	красивый — отталкивающий				
Слово	лик	лицо	морда	рыло	харя
F	2,1	2,4	2,9	3,3	3,5

3. Программные средства

Приложение для анализа фонетической значимости слов должно реализовывать следующие функции:

- анализ отдельных слов;
- сортировка слов с упорядочением в соответствии с их фонетическими значимостями.

В программном комплексе можно выделить несколько блоков, каждый из которых будет выполнять определенные функции (рис. 2).

Рис. 2. Схема взаимодействия блоков программы

Пользовательский интерфейс обеспечивает взаимодействие пользователя с основными функциями программы и реакцию программы на действия, совершённые пользователем, для выполнения соответствующих операций.

Блок валидации необходим для проверки корректности ввода.

Блок разбиения слов на звукобуквы отвечает за корректное деление слов на звукобуквы для последующих вычислений.

В блоке вычисления фонетической значимости происходят расчеты фонетической значимости слов по формулам.

Файл с оценками звукобукв содержит данные о звуках для вычислений.

Заключение

Программный комплекс для анализа фонетической значимости слов может быть эффективно использован в различных областях. Компании могут использовать программу для анализа звуковых ассоциаций и фонетической значимости названий продуктов, брендов или рекламных слоганов. В академической среде он может стать инструментом для фоносемантических исследований, позволяя исследователям анализировать звуковые и семантические аспекты слов на больших объемах данных. Применение программы в учебных целях также представляет интерес: студенты могут использовать ее для изучения фонетических особенностей языка, проведения собственных исследований и практических упражнений по анализу слов.

Таким образом, программный комплекс открывает новые перспективы в исследовании фонетической значимости слов и может быть полезным инструментом как для специалистов в области лингвистики, так и для обучения студентов.

Литература

1. Журавлев А. П. Фонетическое значение / М-во высш. и сред. спец. образования РСФСР. — Ленинград : Изд-во Ленингр. ун-та, 1974. — 160 с.
2. Журавлев А. П. Звук и смысл : Кн. для внеклас. чтения учащихся ст. классов. — 2-е изд., испр. и доп.—М.: Просвещение, 1991. — 160 с.

УДК 004.4

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА С ПОМОЩЬЮ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

И. С. Борченко

Воронежский государственный университет

Введение

Мобильные приложения представляют собой удобный способ получить доступ к обучающим материалам в любое время и в любом месте. Это позволяет пользователям удобно изучать английский язык, находясь в комфортной для них обстановке. Рекомендательные же системы позволяют адаптировать приложение под индивидуальные потребности пользователей, что способствует более эффективному изучению языка.

Целью данной работы является исследование использования рекомендательных систем в мобильных приложениях, работающих с изучением новых слов, а также разработка такого приложения.

1. Описание проекта

Данное мобильное приложение предназначено для повышения плодотворности от изучения новых слов английского языка путём персонализированного подхода. При первом использовании продукта пользователь должен будет пройти тест на определение уровня знания языка, а также указать сферу своих интересов. Далее, на основе этих данных, приложение составит план подготовки на сегодня.

Приложение состоит из двух частей — клиентской и серверной. Клиентская часть представляет собой мобильный интерфейс, который позволяет пользователю проходить упражнения и взаимодействовать со словарём. Серверная часть выполняет следующие функции:

1. Управление базой данных: серверная часть обрабатывает запросы клиентской части и осуществляет доступ к базе данных, где хранится информация о словах, категориях интересов, к которому оно относится, уровня языка, с которым ассоциируется данное слово (в соответствии с CEFR — общеевропейскими компетенциями владения иностранным языком), а также о частоте использования данного слова.
2. Обработка запросов пользователей: серверная часть обрабатывает запросы, поступающие от клиентской части, и возвращает результаты выполнения операций. Например, при поиске в словаре нового английского слова серверная часть выполняет проверку в базе данных и, в случае наличия введённого термина, возвращает его перевод и перечень категорий интересов, с ним ассоциируемых.
3. Формирование рекомендаций: ежедневно сервер анализирует текущий пользовательский словарь и подбирает новые слова на изучение.
4. Обновление и поддержка приложения: серверная часть обеспечивает обновление и поддержку приложения, включая исправление ошибок, добавление новых функций и техническую поддержку.

2. Технические требования

Приложение включает следующие модули:

1. «Пользователь».
2. «Тест на словарный запас».
3. «Тренажёр».

Модуль «Пользователь» обладает следующими возможностями:

1. Возможность просмотра результатов теста на словарный запас.
2. Возможность просмотра активного словаря на изучении.
3. Возможность ручного добавления нового слова на изучение.
4. Возможность ручного удаления слова с изучения.
5. Возможность редактирования сферы интересов.

Модуль «Тест на словарный запас» обладает следующими свойствами:

1. Поэтапное определение словарного запаса пользователя путём выбора знакомых слов из представленного перечня.

Модуль «Тренажёр» обладает следующими свойствами:

1. Упражнение на изучение новых слов в соответствии с выбранной сферой интересов пользователя.
2. Упражнение на изучение новых слов в соответствии с уровнем словарного запаса пользователя.
3. Упражнение на повтор ранее изученных слов.

3. Этапы разработки

Разработка приложения была разделена на несколько этапов:

1. Анализ рынка: проведение анализа рынка мобильных платформ и обзор существующих программных продуктов, предназначенных для изучения иностранных языков. Также анализ различных подходов к созданию рекомендательных систем.
2. Анализ требований: разработка решения, которое сочетает в себе достоинства существующих подходов и исключает недостатки, связанные со сложностью реализации и интеграции. Определение функциональных и нефункциональных требований к приложению.
3. Проектирование: создание макетов интерфейса приложения (UI/UX), выбор технологий и инструментов, разработка архитектуры приложения.
4. Разработка: непосредственно создание кода приложения, тестирование и отладка функций, исправление ошибок.
5. Тестирование: проверка работоспособности приложения на различных устройствах, в разных условиях, а также поиск и исправление ошибок.
6. Выпуск проекта.

4. Реализация проекта

4.1. Средства разработки

Веб-приложение разрабатывалось в среде Android Studio 2023.1.1 с использованием фреймворка Flutter. В качестве языка программирования был использован тесно ассоциируемый с выбранным фреймворком Dart. Flutter — это фреймворк от компании Google

для создания кроссплатформенных мобильных приложений. Одним из основных преимуществ Flutter является возможность написания одного кода, который будет работать как на Android, так и на iOS. Это упрощает процесс разработки и поддержки приложения. Кроме того, Flutter обладает горячей перезагрузкой и позволяет мгновенно видеть изменения в приложении после внесения кодовых правок, что значительно ускоряет процесс разработки. В целом, Flutter представляет собой мощный инструмент для разработки кроссплатформенных мобильных приложений, который сочетает в себе производительность, удобство и красивый дизайн.

Для хранения данных была использована база данных Microsoft SQL Server. К её основным достоинствам относятся масштабирование системы и малый размер страниц, благодаря чему данные извлекаются быстро, а сложную информацию удобнее хранить. Кроме того, система обрабатывает транзакции в интерактивном режиме, есть динамическая блокировка.

4.2. Разработка базы данных

Была разработана схема базы данных, которая включала в себя следующие таблицы:

1. Слова (Words): содержит информацию о словах, доступных в словаре: их перевод, принадлежность к категориям интересов, ассоциируемый уровень языка, частота использования данного слова, степень знания слова пользователем (по шкале от 0 до 5, где 0 — слово не знакомо, а 5 — слово полностью изучено).
2. Категории интересов (CategoriesOfInterests): содержит перечень категорий, к которым могут относиться слова.
3. Данные пользователя (UserData): содержит информацию о языковом уровне пользователя, список интересующих категорий слов.

4.3. Реализация рекомендательной системы

Рекомендательная часть реализована content-based методом. Content-based метод учитывает опыт и предпочтения только одного пользователя и реализуется с применением *формулы косинусного сходства*:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}.$$

Косинусное сходство (cosine similarity) можно использовать как некую меру дистанции между словами, которая в нашем случае будет зависеть от категории интересов, уровня языка, с которым ассоциируется данное слово, а также от частоты использования данного слова. Чем ближе значение сходства к единице, тем сильнее слова «похожи» друг на друга. В итоге у нас получается матрица дистанций между всеми словами, по которой довольно легко определить ближайшие. Таким образом, берётся изученное или уже известное слово и подбираются ещё незнакомые слова, подходящие для изучения.

Заключение

В результате проделанной работы было исследовано использование рекомендательных систем в мобильных приложениях, работающих с изучением новых слов, а также разработано

приложение, применяющее данную технологию. Полученное приложение рассчитано на широкую аудиторию и предоставляет более эффективный метод для изучения иностранных слов.

Литература

1. Документация Flutter. – Режим доступа: https://docs.flutter.dev/codelabs. – (Дата обращения: 10.04.2024).

2. Документация Dart. – Режим доступа: https://dart.dev/guides. – (Дата обращения: 10.04.2024).

3. Рекомендательные системы: как помочь пользователю найти то, что ему нужно? – Режим доступа: https://habr.com/ru/companies/productstar/articles/523686/. – (Дата обращения: 14.03.2024).

УДК 004.428

Реализация OpenAPI спецификации.

Будиловский А. А

Воронежский государственный университет

1.Введение

В современном мире часто необходимо написать высокопроизводительный бэкенд, особенно в контексте современных приложений, ориентированных на API. Изучив библиотеки на языке программирования c++, можно выделить userver, который обладает наибольшим количеством функций, доступных сразу из него и сочетающихся между собой. Однако при более близком знакомстве можно обнаружить один существенный недостаток, который затрудняет разработку — недружелюбный интерфейс http обработчиков.

Он не предоставляет быстрых и кратких способов парсинга запросов или сериализации ответов. Возникла идея обернуть эту библиотеку, предоставив более удобный интерфейс для пользователей.

Эта статья рассмотрит процесс создания такой “библиотеки”.

2. Постановка задачи.

В данной статье мы ставим перед собой ряд задач, направленных на упрощение разработки и повышение гибкости при создании обёртки для бэкенд библиотеки userver:

1. **Избежание повторяющегося кода**: Целью является создание библиотеки, которая позволит разработчикам определять типы для запросов и ответов без необходимости написания многословного кода для парсинга и сериализации. Это упростит процесс разработки и сделает код более компактным и понятным.
2. **Генерация собственной схемы**: Сервис, использующий библиотеку, должен иметь возможность автоматически генерировать собственную схему, основываясь на типах, используемых в качестве запросов и ответов. Это облегчит процесс документирования API и упростит взаимодействие с ним.
3. **Поддержка сторонних типов**: Важным аспектом нашей библиотеки является возможность использовать уже существующие типы данных из других библиотек. Это позволит разработчикам воспользоваться широким спектром возможностей и интегрировать нашу библиотеку в различные проекты с минимальными изменениями.
4. **Опора на OpenAPI.**

   Обработчики, запросы, ответы и всё, что будет использоваться с данной библиотекой, должны соответствовать OpenAPI спецификации и полностью отображаться в схему.

Цель данной статьи состоит в том, чтобы показать, как эти задачи могут быть решены при создании расширения для библиотеки userver, и какие преимущества это принесет для разработчиков. Мы обсудим методы, которые были использованы для достижения этих целей, а также рассмотрим примеры кода и реальные сценарии использования.

3. Метод решения.

3.1. Принципы архитектуры

Архитектура библиотеки должны быть построена с учетом следующих принципов:

5. Расширяемость: Архитектура библиотеки разработана с учетом возможности легкого расширения функциональности. Это означает, что новые возможности и функциональность могут быть добавлены без изменения существующего кода. Классы и функции организованы таким образом, чтобы их можно было легко расширять или заменять при необходимости.
6. Разделение ответственности: Выделение отдельных абстракций, которые отвечают только за выполнение определенной задачи. Это способствует уменьшению сложности кода, облегчает его понимание и поддержку, а также обеспечивает возможность повторного использования компонентов.
7. Использование вычислений, выполняемых на этапе компиляции. Это позволит компилятору сгенерировать более эффективный и производительный код.
8. Возможность повторного использования: Компоненты и функции библиотеки разработаны с учетом возможности их повторного использования в различных контекстах.

На рисунке 1 приведена общая схема обработки запросов.

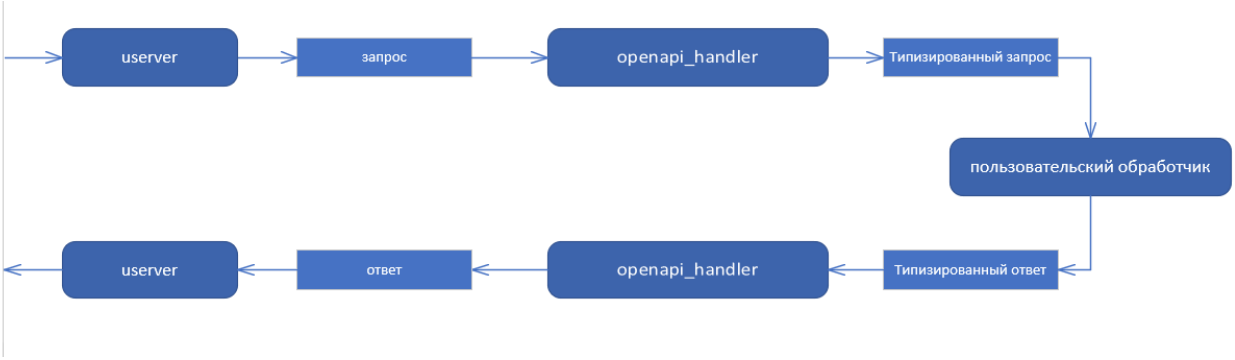


Рис. 1, схема обработки запросов

На рисунке 2 и 3 приведена обобщенная диаграмма классов и задач, которые они выполняют.

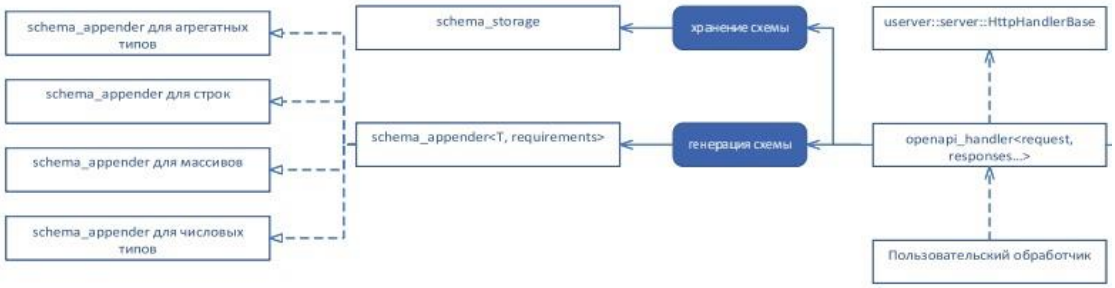


Рис. 2, общая диаграмма классов (часть 1)

Рис. 3, общая диаграмма классов (часть 2)

3.2. Иерархия типов

1. schema_storage - данный класс будет хранить сгенерированную схему для всего сервиса. Является компонентом (компоненты - это синглтоны по имени в userver)
2. schema_appender<T, requirements> - специализации данного класса реализуют логику описания в схеме конкретного типа с наложенными на него ограничениями.
3. request_parser<T,source_type> - специализации реализуют парсинг из http запроса полей
4. response_serailizator<T, source_type> - специализации реализуют сериализацию в http ответ полей
5. validate проверяет соответствия значения требованиям(requirements). Requirements является абстракцией, которая унифицирует накладываемые ограничения на разные типы в openapi во что-то, имеющее единое поведение.
6. openapi_handler - это шаблонный базовый класс, который пользователь библиотеки сможет использовать для того, чтобы писать свои обработчики http запросов

Для интроспекции агрегатных типов (в других языках такие обычно называются data/row types) была использована библиотека boost.pfr. Она позволяет получить типы и имена полей в типе.

4. Результаты

На данный момент реализовано:

- Генерация OpenApi схем
- Шаблонный базовый класс для http обработчика

- Базовые типы из OAS(string, enum, array, object, number)
- Ограничения на базовые типы(пример: minLength)

Пример использования приведён в листинге 1.

```
//объявляем тело запроса

struct Body {

   std::optional<std::vector<std::int64_t>> some_field;

};

//добавляем требования на поле



UOPENAPI_CE_REQUIREMENTS(Body, some_field) = array_requirements{.min_items =
2};

//объявляем запрос

struct Request {

   Body body;

   std::int64_t index_add = 0;

   std::int64_t value_add = 0;

};

//указываем требования на поле запроса

UOPENAPI_CE_REQUIREMENTS(Request, index_add) =
number_requirements<std::int64_t>{

   .minimum = 1, .maximum = 3};

//объявляем ответ

struct Response {

   Body body;

};

using Resp200 = uopenapi::http::response<Response, 200>;

using Base = uopenapi::components::openapi_handler<Request, Resp200>;
```

```
struct Handler : Base {
   static constexpr std::string_view kName = "test-handler";
   Handler(const userver::components::ComponentConfig& cfg,
           const userver::components::ComponentContext& ctx)
       : Base(cfg, ctx) {}
   response handle(Request req) const override {…}
};
int main(int argc, char* argv[]) {
   auto component_list =
       userver::components::MinimalServerComponentList()
           .Append<userver::server::handlers::Ping>()
           .Append<userver::components::TestsuiteSupport>()
           .Append<userver::components::HttpClient>()
           .Append<userver::clients::dns::Component>()
           .Append<userver::server::handlers::TestsControl>()
           .Append<Handler>();
          .Append<uopenapi::components::schema_storage>()
          .Append<uopenapi::components::schema_http_distributor>();
   return userver::utils::DaemonMain(argc, argv, component_list);
}
```

Листинг 1. Пример кода.

В рамках данного примера объявляется запрос и ответ, указываются дополнительные требования на поля запроса (и ответа соответственно), а так же реализуется обработчик с пользовательской логикой.

5. Заключение

В рамках данной исследовательской работы была разработана и представлена архитектура библиотеки, обеспечивающей генерацию OpenAPI схем и обработку HTTP запросов. Реализованы шаблонные базовый класс для HTTP обработчиков, что обеспечивает гибкость и

расширяемость функциональности. Были внедрены базовые типы данных из спецификации OpenAPI, такие как строка, массив, объект и число, а также возможность наложения ограничений на эти типы, например, ограничение на минимальную длину строки.
Полученные результаты представляют собой значительный шаг вперед в разработке инструментов для автоматизации создания и документирования веб-сервисов. Благодаря реализованным функциям разработчики теперь могут создавать API с помощью более удобного и гибкого интерфейса, что способствует повышению производительности и упрощению процесса разработки.
Важно отметить, что результаты данного исследования могут быть использованы не только напрямую, но и опосредованно. Созданная архитектура и реализованные функциональные возможности могут послужить основой для разработки более сложных систем и инструментов, а также стать источником вдохновения для новых исследований в области автоматизации и оптимизации процессов разработки веб-сервисов.

6. Список использованной литературы:

1. Документация библиотеки userver. - Режим доступа: https://userver.tech – (Дата обращения: 20.04.2024)
2. Документация библиотеки boost.pfr – Режим доступа: - (Дата обращения 20.04.2024) https://github.com/boostorg/pfr
3. OpenAPI спецификация – Режим доступа: - (https://github.com/OAI/OpenAPI-Specification/tree/3.0.0)

УДК 004.657

СРАВНЕНИЕ ЭФФЕКТИВНОСТИ РАЗЛИЧНЫХ ТИПОВ ИНДЕКСОВ В POSTGRESQL

А. Э. Бурков, М. В. Матвеева

Воронежский государственный университет

Введение

В современном мире базы данных играют важную роль в обработке и хранении информации. С развитием технологий и увеличением объемов данных, вопрос эффективного управления и доступа к информации становится все более актуальным. В контексте PostgreSQL, одной из наиболее популярных открытых систем управления базами данных, возникает важный вопрос о выборе индексов для оптимизации производительности запросов и экономии пространства. В статье проводится сравнительный анализ эффективности между различными типами индексов (в том числе доступными из расширений) в PostgreSQL, с целью выявить их отличия и преимущества в различных сценариях использования.

1. Обзор рассматриваемых типов индексов

В разделе рассматриваются типы индексов, которые используются для сравнения эффективности.

1.1. *B-tree индекс*

Индекс B-tree в PostgreSQL [1] представляет собой многоуровневую иерархическую структуру, в которой записи упакованы в страницы. Единственная метастраница индекса хранится в фиксированной позиции в начале первого файла сегмента индекса. Все другие страницы разделяются на внутренние и листовые. В листовых страницах индексные записи содержат индексируемые данные и ссылки на строки таблицы. Во внутренних страницах каждая запись ссылается на дочернюю страницу индекса и содержит минимальное значение ключа этой страницы. Схематичный вид данного индекса представлен на рис. 1 [2].

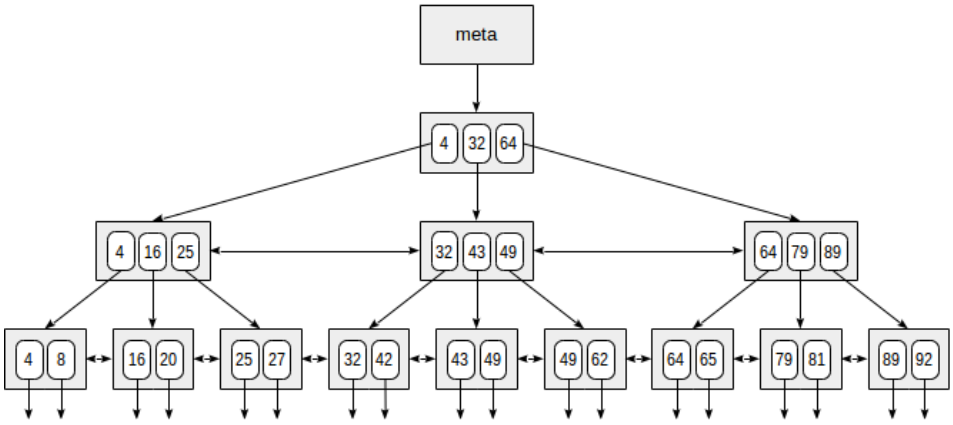


Рис. 1. Устройство B-tree индекса

Данный тип индекса используется для данных, которые можно отсортировать, то есть,

для типа данных должны быть определены операторы сравнения: "=", "<", ">", "<=", ">=".

Он также полезен для запросов, включающих операторы BETWEEN и IN.

Для создания данного индекса используется стандартная команда CREATE INDEX, так как этот тип индекса используется по умолчанию:

```
CREATE INDEX index_name ON table(column1, column2, …);
```

1.2. *BRIN индекс*

BRIN (Block Range Index) — это специальный тип индекса в PostgreSQL, который оптимизирует поиск по диапазонам значений, основываясь на предположении, что данные в таблице упорядочены по некоторому ключу, например, по времени или идентификатору.

Сначала идет страница с метаданными, затем идут страницы блоков с информацией. Каждая строка индекса содержит сводку по какому-то одному диапазону. Между метастраницей и блоками с данными находятся страницы с обратной картой зон, которые фактически являются массивом указателей (ссылок на строки таблицы) на соответствующие индексные строки. Схематичный вид данного индекса представлен на рис. 2.

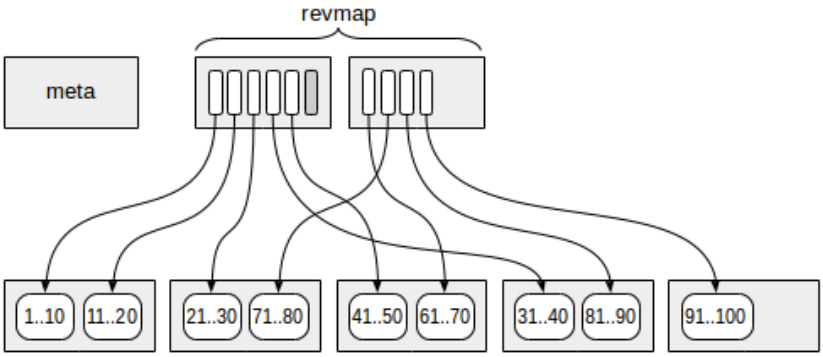


Рис. 2. Устройство BRIN индекса

Данный тип индекса применяется для экономии места на диске, так как он требует значительно меньше пространства, чем другие типы индексов. Он наиболее полезен в ситуациях, когда таблица содержит большое количество данных, упорядоченных по определенному ключу, и когда запросы часто выполняются по диапазону значений.

Общий недостаток заключается в том, что он не гарантирует точности результата. Иногда могут быть ложные срабатывания, особенно в случае, если данные в таблице распределены не равномерно или если есть большие пропуски между значениями.

Ложное срабатывание означает, что поиск по индексу возвращает положительный результат для элемента, который фактически не существует в индексируемых данных. Другими словами, индекс сообщает о наличии элемента, хотя его там на самом деле нет.

Для создания данного индекса используется команда CREATE INDEX с указанием соответствующего метода доступа, используя USING:

```
CREATE INDEX index_name ON table USING brin(column1, column2, …);
```

1.3. *Bloom индекс*

Bloom индекс в PostgreSQL представляет собой тип индекса, использующий фильтры Блума.

Фильтр Блума — это вероятностная и компактная структура данных, которая позволяют быстро определить, присутствует ли элемент в множестве, с определенной вероятностью

ложного срабатывания. Он представляет собой битовый массив (сигнатуру), который изначально заполнен нулями. Выбираются несколько различных хеш-функций, которые отображают любой элемент множества в такое же количество бит сигнатуры. Чтобы добавить элемент в множество, нужно установить в сигнатуре каждый из этих битов в единицу. Следовательно, если все соответствующие элементу биты установлены в единицу — элемент может присутствовать в множестве; если хотя бы один бит равен нулю — элемента точно нет.

Bloom индекс представляет собой плоскую структуру [3]. Сначала идет метастраница, затем обычные страницы с индексными строками. Каждая строка индекса содержит битовый массив и ссылку на табличную строку. Схематичный вид данного индекса представлен на рис. 3.

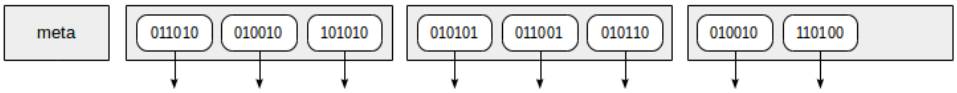


Рис. 3. Устройство bloom индекса

Основное преимущество bloom индексов заключается в их эффективности при поиске элементов в очень больших множествах данных. Однако их использование может быть ограничено, так как они не подходят для всех типов запросов и могут давать ложные срабатывания.

Данный индекс недоступен по умолчанию и его необходимо активировать из дополнительно поставляемого модуля. Для этого используется следующая команда:

```
CREATE EXTENSION bloom;
```

После этого мы можем создать сам индекс:

```
CREATE INDEX index_name ON table USING bloom(column1, column2, …);
```

2. Сравнение возможностей рассмотренных типов индексов

Для облегчения выбора наиболее подходящего типа индекса для достижения наибольшей эффективности необходимо изучить какими возможностями обладают каждый индекс. В табл. 1 представлены основные возможности каждого типа индекса.

Таблица 1

Сравнение возможностей разных типов индексов

Возможность	B-tree индекс	BRIN индекс	Bloom индекс
Поддержка равенства	+	–	+
Поддержка сравнения	+	–	–
Поддержка диапазонов	+	+	–
Поддержка индексации нескольких столбцов	+	+	+

Возможность	B-tree индекс	BRIN индекс	Bloom индекс
Поддержка уникальности индекса	+	–	–
Экономия пространства	Средняя	Высокая	Высокая
Скорость поиска	Высокая	Высокая	Средняя

3. Анализ эффективности индексов

Для демонстрации работы индексов была подготовлена тестовая база данных авиаперевозок с наполнением данными, приближенными к реальным. Размер полученной базы данных составляет 6769 MB [4]. Таблица, для которой создаются индексы, содержит более 30 миллионов записей.

Так как эффективность индексов невозможно сравнить в рамках одинаковых запросов из-за ограничений возможностей каждого, то будем сравнивать их в ситуациях, когда они могут заменять друг друга.

3.1. Поиск по точному значению

Для сравнения воспользуемся двумя типами индексов, для которых определен оператор равенства: B-tree и bloom индексы.

Создадим B-tree индекс и выполним для него запрос поиска всех записей полётов определенного пассажира с определенным номером документа:

```
CREATE INDEX btree_idx ON bookings.flights_bi(passenger_name, passenger_id);
```

Посмотрим сколько дискового пространства занимает созданный индекс (рис. 4):

```
SELECT pg_size_pretty(pg_total_relation_size('bookings.btree_idx'));
```

```
EXPLAIN (costs off, analyze) SELECT * FROM bookings.flights_bi
WHERE passenger_name = 'VITALIY MATVEEV' AND passenger_id = '5651 431832';
```

Результат выполнения представлен на рис. 5.

	pg_size_pretty text
1	333 MB

Рис. 4. Размер B-tree индекса на диске

	QUERY PLAN text
1	Index Scan using btree_idx on flights_bi (actual time=0.042..0.045 rows=4 loops=1)
2	Index Cond: ((passenger_name = 'VITALIY MATVEEV'::text) AND ((passenger_id)::text = '5651 431832'::te...
3	Planning Time: 1.822 ms
4	Execution Time: 0.055 ms

Рис. 5. Результат анализа B-tree индекса по точному значению

При создании bloom индекса необходимо определить длину сигнатуры, так как установленное по умолчанию значение не всегда будет универсальным. Для 30 миллионов строк и вероятности ложного срабатывания в 0,01 возьмем длину, равную 20 бит:

```
CREATE INDEX bloom_idx ON bookings.flights_bi
USING bloom(passenger_name, passenger_id) WITH(length=20, col1=7, col2=7);
```

Посмотрим сколько дискового пространства занимает созданный индекс (рис. 6):

```
SELECT pg_size_pretty(pg_total_relation_size('bookings.bloom_idx'));
EXPLAIN (costs off, analyze) SELECT * FROM bookings.flights_bi
WHERE passenger_name = 'VITALIY MATVEEV' AND passenger_id = '5651 431832';
```

Результат выполнения представлен на рис. 7.

	pg_size_pretty text
1	292 MB

Рис. 6. Размер bloom индекса на диске

	QUERY PLAN text
1	Bitmap Heap Scan on flights_bi (actual time=96.498..96.503 rows=4 loops=1)
2	Recheck Cond: ((passenger_name = 'VITALIY MATVEEV'::text) AND ((passenger_id)::text = '5651 431832'::te...
3	Heap Blocks: exact=4
4	-> Bitmap Index Scan on bloom_idx (actual time=96.486..96.487 rows=4 loops=1)
5	Index Cond: ((passenger_name = 'VITALIY MATVEEV'::text) AND ((passenger_id)::text = '5651 431832'::te...
6	Planning Time: 1.800 ms
7	Execution Time: 96.538 ms

Рис. 7. Результат анализа bloom индекса по точному значению

В табл. 2 представлены результаты для сравнения.

Таблица 2

Результаты выполнения запросов с индексами для поиска по точному значению

Параметр	B-tree индекс	Bloom индекс
Время создания индекса	54.29 с	11.050 с
Занимаемое пространство	333 МБ	292 МБ
Скорость выполнения	0.055 мс	96.538 мс

B-tree индекс показал лучший результат по скорости при поиске точного значения, так как древовидная структура хранит данные в упорядоченном виде, что обеспечивает быстрый поиск, но при этом bloom индекс при таком же количестве индексируемых столбцов занимает немного меньше пространства, что при определенных ситуациях может оказаться необходимым.

3.2. Поиск в диапазоне

В этом случае bloom индекс уже не может быть использован, так как он не поддерживает операторы сравнения. Вместо него будет использоваться индекс BRIN.

Создадим B-tree индекс и выполним для него запрос поиска всех записей полётов, которые вылетали неделю назад:

```
CREATE INDEX btree_idx ON bookings.flights_bi(scheduled_time);
```

Посмотрим сколько дискового пространства занимает созданный индекс (рис. 8):

```
SELECT pg_size_pretty(pg_total_relation_size('bookings. btree_idx));
```

```
EXPLAIN (costs off, analyze) SELECT * from bookings.flights_bi
WHERE scheduled_time >= bookings.now()::date - interval '7 days'
AND scheduled_time < bookings.now()::date - interval '7 days' + interval '1 day';
```
Результат выполнения представлен на рис. 9.

	pg_size_pretty text
1	210 MB

Рис. 8. Размер B-tree индекса на диске

	QUERY PLAN text
1	Index Scan using btree_idx on flights_bi (actual time=0.018..6.936 rows=83954 loops=1)
2	Index Cond: ((scheduled_time >= (('2017-08-15 18:00:00+03'::timestamp with time zone)
3	Planning Time: 1.935 ms
4	Execution Time: 8.706 ms

Рис. 9. Результат анализа B-tree индекса для диапазона

BRIN индекс создается аналогично:
```
CREATE INDEX brin_idx ON bookings.flights_bi(scheduled_time);
```
Посмотрим сколько дискового пространства занимает созданный индекс (рис. 10):
```
SELECT pg_size_pretty(pg_total_relation_size('bookings. brin_idx));
EXPLAIN (costs off, analyze) SELECT * from bookings.flights_bi
WHERE scheduled_time >= bookings.now()::date - interval '7 days'
AND scheduled_time < bookings.now()::date - interval '7 days' + interval '1 day';
```
Результат выполнения представлен на рис. 11.

	pg_size_pretty text
1	184 kB

Рис. 10. Размер BRIN индекса на диске

	QUERY PLAN text
1	Bitmap Heap Scan on flights_bi (actual time=2.221..36.849 rows=83954 loops=1)
2	Recheck Cond: ((scheduled_time >= (('2017-08-15 18:00:00+03'::timestamp with time zone)
3	Rows Removed by Index Recheck: 12285
4	Heap Blocks: lossy=1664
5	-> Bitmap Index Scan on brin_idx (actual time=0.442..0.442 rows=16640 loops=1)
6	Index Cond: ((scheduled_time >= (('2017-08-15 18:00:00+03'::timestamp with time zone)
7	Planning Time: 1.712 ms
8	Execution Time: 38.705 ms

Рис. 11. Результат анализа BRIN индекса для диапазона

В табл. 2 представлены результаты для сравнения.

Таблица 2

Результаты выполнения запросов с индексами для поиска по диапазону

Параметр	**B-tree индекс**	**BRIN индекс**
Время создания индекса	9.191 с	5.021 с
Занимаемое	210 МБ	184 КБ

пространство		
Скорость выполнения	8.706 мс	38.705 мс

B-tree вновь показал быстрый результат, но BRIN оказался намного эффективнее по занимаемому месту на диске.

Заключение

В результате анализа было получено, что B-tree индекс показывает высокую скорость и универсальность в двух видах запросов, но при этом BRIN индекс намного эффективнее при экономии пространства, что делает выбор в его сторону при ограничении дискового пространства и работы с диапазонами данных.

Bloom индекс хоть и не показал высоких результатов по скорости, но оказался эффективнее по занимаемому месту на диске, чем B-tree и тоже может использоваться при определенных условиях запросов и ситуаций с высокой кардинальностью данных при подборе подходящих параметров длины сегмента и количества индексируемых столбцов. Он может быть эффективным для запросов, которые часто выполняются, но допускают небольшой вероятности ложных срабатываний.

Литература

1. Документация PostgreSQL. – Режим доступа: https://postgrespro.ru/docs/postgresql. – (Дата обращения: 08.04.2024).

2. Индексы в PostgreSQL. – Режим доступа: https://habr.com/ru/companies/postgrespro/articles/326096. – (Дата обращения: 08.04.2024).

3. Bloom Indexes in PostgreSQL. – Режим доступа: https://www.percona.com/blog/bloom-indexes-in-postgresql. – (Дата обращения: 09.04.2024).

4. Демонстрационная база данных. – Режим доступа: https://postgrespro.ru/education/demodb. – (Дата обращения: 09.04.2024).

УДК-004.8

Поиск возможных собеседников в социальных сетях на основе анализа текстов постов

А. П. Буркова

Воронежский государственный университет

Введение

Большая часть текстовых данных не структурирована и разбросана. Эти текстовые данные могут дать полезные знания, если они правильно получены и отформатированы. Методы анализа неструктурированного текста включают классификацию текста, анализ настроений, распознавание именованных объектов (NER) и систему рекомендаций, биомедицинский анализ текста, тематическое моделирование и другие.

Настоящая работа посвящена разработке методики поиска возможных собеседников в социальных сетях на основе анализа текстов постов.

Для того что бы проанализировать текстовые посты необходимо получить их векторное представление. Предварительно обученные векторные представления слов — это векторные представления слов фиксированной длины, которые отражают общую семантику фраз и лингвистические шаблоны на естественном языке. Исследователи предложили различные методы получения таких представлений. Было доказано, что встраивание слов полезно во многих приложениях NLP [1].

Наиболее перспективным подходом к анализу естественного языка являются нейросети, использующие механизм attention – способность поиска взаимосвязей между различными частями текста [2], и построенные на архитектуре так называемых трансформеров. Первой такой моделью стала BERT [3]. Одна из ключевых особенностей BERT заключается в том, что она способна генерировать векторное представление, учитывает контекст для всех слов и дольше удерживает информацию о контексте для каждого слова [4]. В настоящее время BERT превосходит нейросетевые модели предыдущего поколения, такие как word2vec, LSTM и др. Архитектура модели BERT включает многослойный двунаправленный кодер transformer. BERT использует маскированное языковое моделирование для оптимизации и объединения позиционного встраивания со статическими встраиваниями слов в качестве входных данных модели. Модель обучается на наблюдениях на учебных данных из нескольких задач во время предварительного обучения. Модель BERT настраивается путем первоначального инициализирования ее с использованием предварительно обученных параметров, а затем точной настройки всех параметров с использованием помеченных данных из последующих заданий [5]. Модели BERT используют встраивания слов-частей. Специальный токен классификации [CLS] всегда является первым токеном в каждой последовательности. Специальный токен [SEP] используется для того, чтобы разделить предложения.

Далее, для того что бы среди всех векторных представлений текстов найти самый близкий по смыслу текст будет использоваться метод на основе kNN. Метод k-ближайших соседей(kNN) — это метрический алгоритм для автоматической классификации объектов или регрессии. В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны. В данной работе семантическая близость текстов и есть “расстояние” между ними. Для этого мы используем kNN-поиск для нахождения наиболее близких и похожих текстов, но это алгоритм основывается на подходе грубой силы, что плохо

скажется на производительности программы. Таким образом, нужно использовать ANN (Approximate Nearest Neighbor) [6].

Approximate kNN-это метод машинного обучения, который используется для быстрого поиска ближайших соседей в больших наборах данных. Он является аппроксимационным, что означает, что он не гарантирует точный результат, но вместо этого стремится к нахождению близких к точным ответам с использованием определенных эвристик и методов.

Одним из основных преимуществ метода Approximate kNN является его способность работать с большими наборами данных и быстро находить ближайших соседей. Основная идея работы Approximate kNN заключается в том, чтобы построить некоторую структуру данных, которая позволяет быстро и эффективно приближаться к ближайшим соседям. После построения структуры данных, при поиске ближайших соседей применяются аппроксимационные методы, которые позволяют быстро находить близкие к точным ответы, снижая вычислительную сложность и увеличивая скорость поиска. Для этой работы был выбран алгортм Annoy.

Annoy (Approximate Nearest Neighbors Oh Yeah) [7] простым языком- это двоичное дерево, где каждый узел является случайным разбиением. Разделяем пространство один раз: Annoy делает это, произвольно выбирая две точки, а затем расщепляя гиперплоскость на равном расстоянии от этих двух точек. Продолжим разбиение каждого подпространства рекурсивно. Мы продолжаем делать это, пока в каждом узле не останется элементов K. В итоге мы получаем двоичное дерево, разбивающее пространство. Обратим внимание, если две точки находятся близко друг к другу в пространстве, маловероятно, что какая-либо гиперплоскость будет их разделять. Для поиска любой точки в этом пространстве мы можем обойти двоичное дерево из корня. Каждый промежуточный узел (малые квадраты в дереве выше) определяет гиперплоскость, поэтому мы можем выяснить, какую сторону гиперплоскости нам нужно выбрать, что и определяет, будем ли мы двигаться к левому или правому дочернему узлу. Поиск точки может быть выполнен за логарифмическое время, так как это высота дерева. Однако текущий процесс имеет несколько недостатков. В данном случае количество соседей оказывается фиксированным. Во-вторых, некоторые из ближайших соседей фактически находятся за пределами этого листового многоугольника. Поэтому необходимо модифицировать алгоритм. К примеру, можно использовать очередь с приоритетом. То есть, если стороны узла достаточно близки, то спускаемся по обе стороны от узла. Вместо того, чтобы просто спуститься по одному пути двоичного дерева, рассмотрим еще несколько. Для этого нужно настроить порог того, насколько мы готовы пойти на «неправильную» сторону раскола. Следующим шагом является вычисление всех расстояний и ранжирование точек. Затем мы сортируем все узлы по расстоянию и возвращаем верхние K ближайших соседей.

Таким образом для эмбединга слов будет использоваться алгоритм BERT, а для последующего поиска ближайшего соседа будет использоваться алгоритм ANNOY.

1. Алгоритм поиска собеседников

Перед началом использования алгоритм необходимо настроить BERT, чтобы он строил векторные представления, основываясь на данных из датасетов. Также, на основе векторных преставлениях настроить дерево поиска. Только после предварительного обучения и настройки можно использовать программу.

Алгоритм:

1) На вход получаем первый пост человека.
2) При помощи BERT получаем векторное представления этого поста.
3) Поиск ближайшего соседа по дереву.
4) Вывод ближайший по смыслу пост.

На рис.1 приведена блок-схема алгоритма. Рассмотрим подробнее, как работает BERT и как происходит построение дерева.

Алгоритм работы BERT:

1. Токенизация. Входной текст разбивается на токены (отдельные слова или подслова).
2. Добавление специальных токенов. К началу и концу последовательности токенов добавляются специальные токены (CLS и SEP).
3. Позиционное кодирование. Каждому токену назначается позиционное кодирование, которое указывает на его положение в последовательности.
4. Эмбеддинг токенов. Каждый токен преобразуется во встроенный вектор с помощью таблицы встроенных векторов.
5. Сложение позиционного кодирования. Векторы встроенных токенов складываются с их позиционным кодированием.
6. Многоуровневые кодировщики Transformer. Последовательность векторов токенов проходит через несколько уровней кодировщиков Transformer. Кодировщики Transformer - это нейронные сети, которые позволяют модели изучать отношения между токенами в последовательности.
7. Выходные представления. Выходными представлениями BERT являются векторы признаков для каждого токена в последовательности.

Алгоритм ANNOY:

1. Выбрать случайные 2 точки данных из набора данных.
2. Разбить набор данных на два подмножества. Подмножество точек данных, которые ближе к первой точке обзора, чем к другой точке данных.
3. Рекурсивно построить поддеревья. Для каждого подмножества рекурсивно построить поддеревья, используя тот же алгоритм.

Рис.1 Блок-схема алгоритма

Тестирование алгоритма

Для обучения модели был взят dataset постов в социальной сети на английском языке, так как поставленная задача требовала большое количество постов на разнообразные темы.

Для проверки работы программы были выбраны повествовательные и вопросительные выражения. Это было сделано для того, чтобы проверить алгоритм на различных ситуациях.

Программа принимает на вход пост человека, далее находится координата этого поста в системе уже существующих постов и в результате получаем пост с минимальным расстоянием. Результаты работы программы приведены в таблице.

Таблица 1. Тестовые результаты

Вводимый пост	Близкий по значению пост
“i like ice cream”	“I ate delicious rice pudding”
“I like playing computer”	“Did well in the videogame I was playing”
“I love cooking”	“i finally learned to cook chicken”
“i love “Dune 2””	“I watched a movie with my wife, and we both enjoyed it.”
“you need to buy yuan”	“I won $20 on a fanduel nba fantasy basketball tournament lineup contest.”
“what is your favorite song of 2018?”	“I listened to a new song that I really liked.“
“i love plaing tennis”	“I played tennis with my friends”
“do you like “the beatles”?”	“I was able to find a rare vinyl record I had been looking for for a long time.”
“i love cats”	“My cat snuggled with me while we slept.”

Заключение

В результате выполнения работы был разработан эффективный алгоритм поиска возможных собеседников в социальных сетях на основе анализа текстов постов. Использование модели эмбеддинга BERT и метода Approximate kNN позволило значительно повысить точность и скорость поиска подходящих кандидатов. Разработанный алгоритм обладает высокой степенью автоматизации и может быть применен в различных сферах, где требуется подбор подходящих людей для общения или сотрудничества.

Список используемых источников

1. Moreo A, Esuli A, Sebastiani F (2021) Word-class embeddings for multiclass text classification. Springer, New York - https://doi.org/10.1007%2Fs10618-020-00735-3 -(Дата обращения: 12.04.2024)
2. Chalkidis I, Fergadiotis M, Malakasiotis P, Aletras N, Androutsopoulos I. LEGAL-BERT: The muppets straight out of law school. 2020. arXiv preprint arXiv:2010.02559.
3. Beltagy I, Lo K, Cohan A. SciBERT: A pretrained language model for scientific text. 2019. arXiv preprint arXiv:1903.10676.
4. Lyashevskaya ON, Shavrina TO, Trofimov IV, Vlasova NA. GramEval 2020 shared task: Russian full morphology and universal dependencies parsing. In Proc. of the International Conference Dialogue. 2020, 553-569.
5. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding.
6. Ozan, E.C.; Kiranyaz, S.; Gabbouj, M. K-subspaces quantization for approximate nearest neighbor search. IEEE Trans. Knowl. Data Eng. 2016, 28, с. 1722–1733.
7. E. Bernhardsson, “Annoy: Approximate Nearest Neighbors in C++/Python optimized for memory usage and loading/saving to disk,” Python package version 1.17.3, Jun. 2023

УДК 004.89

АНАЛИЗ ВРЕМЕННЫХ РЯДОВ НА ПРИМЕРЕ ПРОГНОЗИРОВАНИЯ ПОГОДНЫХ УСЛОВИЙ

С. И. Бурляева

Воронежский государственный университет

Введение

Прогнозирование погодных условий повышает производительность сельского хозяйства и обеспечивает продовольствием и водой для поддержания здоровья граждан. Неравномерное распределение осадков в стране влияет на сельское хозяйство, от которого зависит экономика страны.

Существует проблема недостаточной точности прогнозов, которая может влиять на сельское хозяйство, и частным фермерским хозяйствам для поддержания урожая зачастую необходимо знать более точный прогноз.

1. Постановка задачи

Дан набор данных за период с 1 января 2013 г. по 24 апреля 2017г. в городе Дели, Индия. В нем представлены четыре параметра: температура, влажность, скорость ветра и давление. Необходимо рассмотреть способы анализа временных рядов, а также основные методы прогнозирования упорядоченных данных. Для анализа временных рядов и их прогнозирования будет использован язык Python в связке со сторонними библиотеками (pandas, numpy, tensorflow, matplotlib).

Задача решается с помощью нейросети рекуррентной архитектуры. Подготовка данных должна происходить путем их нормализации и сглаживания шума. Данный способ был выбран благодаря его простоте и точности.

2. Основные определения

Временной ряд — это последовательность точек данных, которые появляются в последовательном порядке за некоторый период времени.

Данные временных рядов — это отмеченные значения или результаты наблюдений, полученные в разные периоды времени. Каждая единица статистического материала называется измерением. При анализе данных временного ряда учитывается взаимосвязь измерений со временем, поэтому важно соблюдать правильную последовательность данных во времени.

Z-преобразование – это метод, который нормализует данные на основе среднего значения (μ) и стандартного отклонения (σ) набора данных

Преобразование Фурье – операция, которая сопоставляет первой функции вторую: вторая функция описывает коэффициенты («амплитуды») при разложении первой функции на элементарные составляющие — гармонические колебания с разными частотами. То есть фактически данное преобразование раскладывает функцию на сумму синусов разной частоты.

3. Решение задачи прогнозирования

3.1. Загрузка наборов данных

Для начала, импортируем необходимые библиотеки для дальнейшей разработки. Также загрузим тестовый и тренировочный наборы данных для обучения нейронной сети и прогнозирования погодных условий [1].

```
# Импорт библиотек
from datetime import datetime

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

# Выгрузка данных из таблицы с тренировочными данными
train_data: np.ndarray = pd.read_csv("DailyDelhiClimateTrain.csv").values

# Выгрузка данных из таблицы с тестовыми данными
test_data: np.ndarray = pd.read_csv("DailyDelhiClimateTest.csv").values
```

Рис. 1. Выгрузка таблиц с тренировочными и тестовыми данными

date	meantemp	humidity	wind_speed	meanpressure
Date of format YYYY-MM-DD	Mean temperature averaged out from multiple 3 hour intervals in a day.	Humidity value for the day (units are grams of water vapor per cubic meter volume of air).	Wind speed measured in kmph.	Pressure reading of weather (measure in atm)
1Jan17 – 24Apr17	11 – 34.5	17.8 – 95.8	1.39 – 19.3	59 – 1.02k
2017-01-01	15.91304347826087	85.8695652173913	2.743478260869565	59.0
2017-01-02	18.5	77.22222222222223	2.8944444444444444	1018.2777777777778
2017-01-03	17.11111111111111	81.88888888888889	4.016666666666667	1018.3333333333334
2017-01-04	18.7	70.05	4.545	1015.7
2017-01-05	18.38888888888889	74.94444444444444	3.3000000000000003	1014.3333333333334
2017-01-06	19.318181818181817	79.31818181818181	8.681818181818182	1011.7727272727273
2017-01-07	14.708333333333334	95.83333333333333	10.041666666666664	1011.375

Рис. 2. Изображение первых строк тестовой таблицы

3.2. Нормализация данных

В дальнейшем, для удобства представления описания алгоритма, будут рассмотрены только данные о температуре.

В соответствии с научным подходом, ни одна модель не способна интерпретировать строковое представление даты «2017-01-01», поэтому предлагается преобразовать значение первого столбца в целые числа, представляющие количество дней с начала отсчета.

```python
def days_since_zero_date(date_str: str) -> int:
  # Преобразование строки в объект datetime
  date_format: str = "%Y-%m-%d"
  date_obj = datetime.strptime(date_str, date_format)

  # Нулевая дата
  zero_date = datetime.strptime("2013-01-01", date_format)

  # Вычисление разницы в днях
  delta = date_obj - zero_date
  days = delta.days

  return days
```

Рис. 3. Функция перевода даты в количество дней от 2013-01-01

Далее переведем все даты в загруженных таблицах в дни и выведем результат.

```python
# Применяем ранее определенную функцию ко всем датам датасета
train_data[:, 0] = np.vectorize(days_since_zero_date)(train_data[:, 0])
test_data[:, 0] = np.vectorize(days_since_zero_date)(test_data[:, 0])

train_data = train_data.astype("float")
test_data = test_data.astype("float")

print(train_data[0])
```

```
[   0.          10.          84.5          0.        1015.66666667]
```

Рис. 4. Результат работы функции, переводящей даты в дни

Для нормализации данных следует использовать метод Z-нормализации по следующей формуле:

$$x_i^{'} = \frac{x_i - \bar{X}}{\sigma_x} \quad (1)$$

Где:

1. $\bar{X}$ – среднее значение выборки;
2. σ_x – стандартное отклонение выборки;
3. x_i – ненормализованное значение;
4. $x_i^{'}$ – нормализованное значение.

В фрагменте кода на рис. 5. был создан класс, который реализует логику нормализации данных. Последние две строки содержат нормализацию обучающего набора данных (за исключением значений дней). Метод DeNormalizeData позволяет преобразовывать данные, полученные с использованием модели, в удобный формат.

```python
class Normalize:
  def __init__(self, data: np.ndarray) -> None:
    self.data: np.ndarray = np.copy(data) # Записываем копию данных

    # Вычисляем среднее по каждому столбцу
    self.__mean: np.ndarray = data.mean(axis=0)
    # Вычисляем стандартное отклонение по каждому столбцу
    self.__std_dev: np.ndarray = data.std(axis=0)

  def normalizeData(self) -> np.ndarray:
    # Возврат нормализованных даннах по формуле
    return (self.data - self.__mean) / self.__std_dev

  def DeNormalizeData(
    self, normalized_data: np.ndarray, axes: list[int] = [0, 1, 2, 3]
  ) -> np.ndarray:
    # Денормализация днанных по указанной оси
    return normalized_data * self.__std_dev[axes] + self.__mean[axes]

# Нормализация температуры
train_normalize_class = Normalize(train_data[:, 1:])
train_data[:, 1:] = train_normalize_class.normalizeData()
```

Рис. 5. Класс реализующий логику нормализации данных

Построим график температуры для наглядности.

```python
# Создаем поле графика
fig, ax = plt.subplots(1, 2)

# Лимиты на осях
ax[0].set_ylim([-10, 40])
ax[1].set_ylim([-10, 40])

# Подписи осей
ax[0].set_ylabel("Темпереτура")
ax[0].set_xlabel("День")
ax[1].set_xlabel("День")

ax[0].set_title("Нормализованная температура")
ax[1].set_title("Реальная температура")

# Сетка на осях
ax[0].grid()
ax[1].grid()

# Отображение нормализованных и ненормализованных данных
ax[0].plot(train_data[:, 1], c="b", linewidth=1)
ax[1].plot(
  train_normalize_class.DeNormalizeData(train_data[:, 1], axes=[0]),
  c="r",
  linewidth=1,
)

plt.show()
```

Рис. 6. Реализация построения графиков температуры

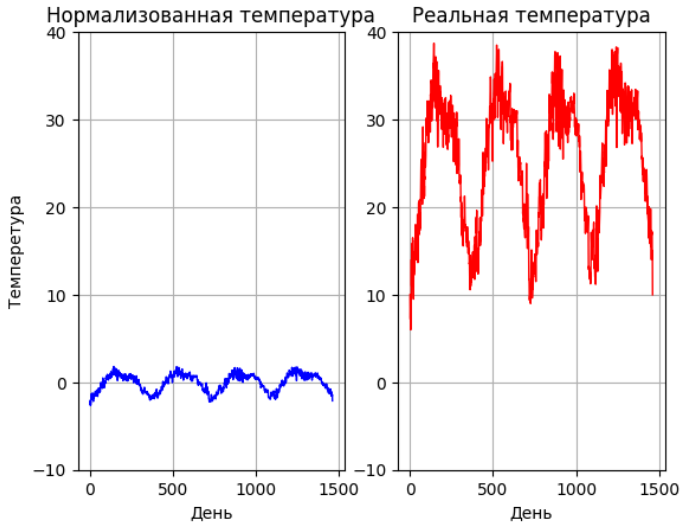


Рис. 7. График нормализованной и реальной температур

На рис. 7. можно заметить, что нормализация данных позволила сохранить их информационную ценность и обеспечить легкое преобразование в исходные данные (второй график). Это делает нормализованные данные более удобными для анализа с использованием моделей машинного обучения.

3.3. Сглаживание шума

Для уменьшения шума используется метод скользящей средней. Суть этого метода заключается в перемещении «окна» определенного размера по всему графику. Во время движения «окна» рассчитывается среднее значение всех элементов, оказавшихся внутри него. В результате происходит сглаживание шумовых колебаний графика, так как они взаимно компенсируют друг друга.

$$SMA_t = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-i} \tag{2}$$

Где:

1. SMA_t – значение простого скользящего среднего в точке t;
2. n – количество значений исходной функции для расчета скользящего среднего (размер «окна»);
3. p_{t-i} – значение исходной функции в точке $t-i$.

Построим график температуры с устраненным шумом.

```
window_size = train_data.shape[0]
denoised_data: np.ndarray = (
  pd.Series(train_data[:, 1])
  .rolling(window=window_size)
  .mean()
  .iloc[window_size - 1 :]
  .values
)
```

Рис. 8. Реализация метода скользящей средней

```
# Создаем поле графика
fig, ax = plt.subplots(2)
fig. tight_layout ()

# Лимиты на осях
ax[0].set_xlim([-10, 1400])
ax[1].set_ylim([-10, 2])

# Подписи осей
ax[0].set_ylabel("Нормированная температура")
ax[0].set_xlabel("Время")

ax[0].set_title("Устранение шума методом скользящей средней")

# Сетка на осях
ax[0].grid()

# Отображение устранения шума
ax[0].plot(
  train_normalize_class.DeNormalizeData(train_data[:, 1], axes=[0]),
  c="b",
  alpha=0.25,
  linewidth=1,
)
ax[0].plot(
  train_normalize_class.DeNormalizeData(denoised_data, axes=[0]),
  c="r",
  linewidth=1,
)

plt.show()
```

Рис. 9. Реализация построения графика устранения шума

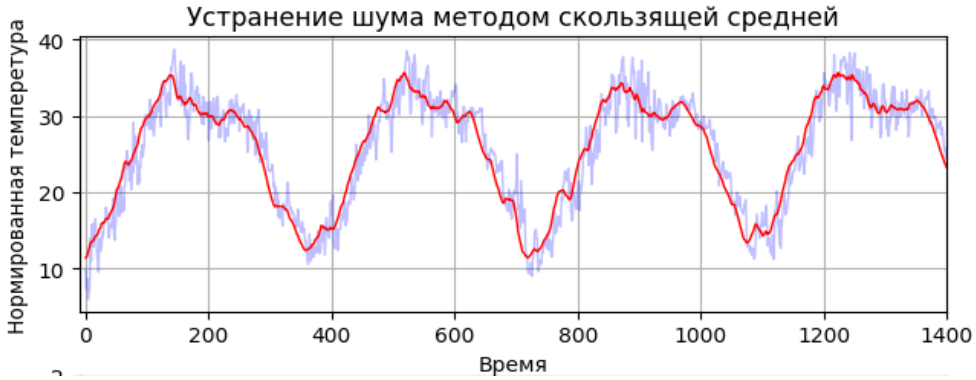


Рис. 10. График устранения шума методом скользящей средней

3.3.Прогнозирование данных

Анализ графика температуры показывает его ярко выраженную периодичность и схожесть с синусоидой. Это указывает на необходимость подбора подходящей синусоиды или их суммы, которая обеспечит наилучшее соответствие температурным данным.

Подберем подходящую кривую с помощью преобразования Фурье и градиентного спуска [138-139].

$$G(t) = \int_{-\infty}^{\infty} g(t)e^{-2\pi ift}dt \quad (3)$$

Где:

1. f – исследуемый диапазон частот;
2. $G(t)$ – амплитуда синусоиды частоты f, которая является слагаемым исходной функции.

В результате, взяв несколько синусоид с максимальным амплитудами (наиболее точно аппроксимирующих имеющиеся данные) и сложив их графики, получим достаточно точную аппроксимацию температуры (в рассматриваемом случае).

К сожалению, преобразование Фурье не предоставит все веса искомой функции, однако для решения данной задачи необходима только информация о частотах, поскольку остальные параметры могут быть получены с использованием метода градиентного спуска. Теперь рассмотрим алгоритм подробнее.

В целях упрощения модели предполагается разложение данных на сумму пяти синусоидальных функций. Ограничение количества слагаемых позволит избежать потери обобщающей способности модели.

```
# Значения на оси X (абсциссе), на основе которых модель будет делать предсказания
x_data:np.ndarray = np.linspace(0, len(denoised_data), len(denoised_data))

# Дискретное Преобразование Фурье, список амплитуд
mfft:np.ndarray = np.fft.fft(denoised_data)

# Получаем индексы частот, которые соответствуют самым значимым (с высокой амплитудой) синусоидам.
imax:np.ndarray = np.argsort(np.absolute(mfft))[::-1]
```

Рис. 11. Реализация разложения данных на сумму пяти синусоид

```
# Кол-во синусов, которые мы будем суммировать
number_of_sinuses: int = 5

# Берем первые number_of_sinuses самых высоких амплитуд (они соответствуют частотам)
imax = imax[:number_of_sinuses]

# Вычисляем частоту каждой синусоиды
frequency:np.ndarray = np.array(imax) / len(denoised_data)
```

Рис. 12. Вычисление частоты каждой синусоиды

Сначала определим функцию, которая будет приближать данные о температуре. Учитывая, что рассматривается сумма пяти синусов, выбор модели представляется очевидным.

$$Y(\omega, x) = b + \sum_{i=0}^{4} \omega_{3i} \sin(\omega_{3i+1}x + \omega_{3i+2}) \quad (4)$$

Необходимо инициализировать параметры таким образом, чтобы обеспечить их корректную оптимизацию (многие модели испытывают трудности при оптимизации параметров тригонометрических функций со случайно выбранными начальными частотами).

1. Параметр ω_{3i+1} инициализируем значением стандартного отклонения всей выборки $(\omega_{3i} = \sigma_x)$.
2. Параметр ω_{3i+1} инициализируем полученной частотой (умноженной на 2π) для каждого синуса соответственно $(\omega_{3i+1} = 2\pi f_i)$. Это нужно сделать, чтобы модель не подбирала частоту волны, в противном случае параметры не будут оптимизированы.
3. Параметр сдвига по абсциссе инициализируем нулем.
4. Параметр b инициализируем математическим ожиданием выборки $(b = \bar{X})$.

```
# Начальные параметры
init_params: np.ndarray = np.array(
    [
    np.array([np.std(denoised_data), frequency[i] * 2 * np.pi, 0.0])
    for i in range(number_of_sinuses)
    ]
  )

bias: float = np.mean(denoised_data)
```

Рис. 13. Инициализация начальных параметров

Оптимизируем параметры с использованием градиентного спуска в сочетании с оптимизатором Adam. В качестве функции ошибки модели используем среднеквадратическую ошибку MSE [400].

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y + \bar{y})^2 \tag{5}$$

Где:

1. y – реальное значение целевой переменной (метка);
2. $\bar{y}$ – предсказание модели.

Перейдем к методу градиентного спуска.

Градиентный спуск заключается в том, что по мере обучения модели вычитаем из каждого веса значение его локальной частной производной от функции ошибки. Данный алгоритм приводит к ее минимизации, то есть к наиболее точным ответам модели.

$$\omega_n = \omega_n - \frac{\delta E}{\delta \omega_n} * \rho \tag{6}$$

Где:

1. E – функция ошибки;
2. ω_n – данный вес;
3. ρ - коэффициент скорости обучения.

3.3 Модель

Создадим модель с помощью библиотеки tensorflow [1]. Для решения данной задачи выберем рекуррентную нейронную сеть, благодаря своей уникальной способности обрабатывать последовательности данных и учитывать предыдущие состояния для прогнозирования последующих значений.

```
# Импорт модулей
import tensorflow as tf
from keras import layers
from keras.optimizers import Adam

tf.random.set_seed(8)

# Определяем слой
class SinLayer(layers.Layer):
  def __init__(self): # Инициализируем методы и атрибуты родительского класса
    super(SinLayer, self).__init__()

  def build(self, _): # Задаем начальные параметры
    self.kernel = self.add_weight(
      "kernel", shape=(number_of_sinuses, 3), trainable=True
    ) # Веса модели

    # Свободный коэффициент
    self.bias = self.add_weight(name="bias", shape=(), trainable=True)

  def call(self, inputs): # Реализация функционала нашей модели (ранее в статье)
    result: float = 0
    for i in range(number_of_sinuses):
      result += self.kernel[i][0] * tf.sin(
        self.kernel[i][1] * inputs + self.kernel[i][2]
      )
    return result + self.bias # Результат работы модели
```

Рис. 14. Определение функциональности модели и ее параметров

Теперь определим саму модель и скомпилируем ее.

```
# Определение модели
model = tf.keras.Sequential(
  [
    layers.Input(shape=(1,)),
    SinLayer(),
  ]
)

# Компилируем модель оптимизатором Adam (с наиболее подходящими параметрами)
# и ошибкой MSE
model.compile(Adam(0.001, 0.8, 0.9), "mean_squared_error")

# Задаем модели ранее определенные веса для правильной оптимизации весов
model.set_weights([init_params, bias])
```

Рис. 15. Определение и компиляция модели

Обучим модель на тренировочных данных и построим график изменения ошибки по мере обучения.

```
# Получаем историю ошибки модели
history = model.fit(x_data, denoised_data, epochs=70)

# Отображаем ее на графике
plt.plot(history.history["loss"])
plt.grid()
plt.xlabel("Эпоха")
plt.ylabel("Значение ошибки MSE на данной эпохе")
plt.show()
```

```
46/46 [==============================] - 0s 3ms/step - loss: 0.1812
Epoch 13/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1660
Epoch 14/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1511
Epoch 15/70
46/46 [==============================] - 0s 2ms/step - loss: 0.1480
Epoch 16/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1227
Epoch 17/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1253
Epoch 18/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1175
Epoch 19/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1194
Epoch 20/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1134
Epoch 21/70
46/46 [==============================] - 0s 3ms/step - loss: 0.1035
Epoch 22/70
```

Рис. 16. Обучение модели

Эпоха — это количество полных прохождений всего набора данных. Чем больше эпох, тем лучше натренирована нейросеть.

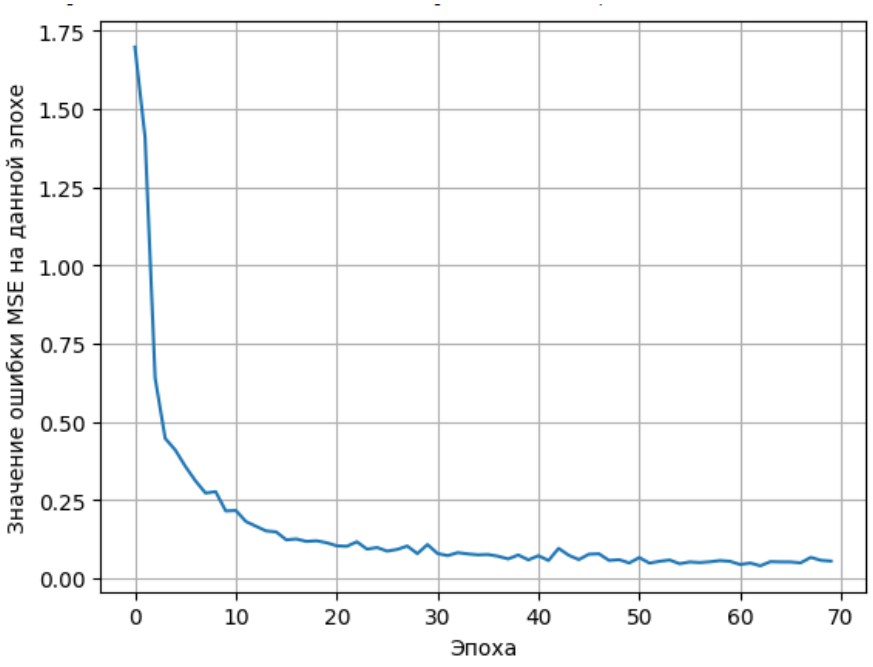


Рис. 17. График изменения ошибки

На рис. 17. можно заметить, что ошибка модели сильно уменьшилась, модель натренировалась.

Теперь посмотрим, как модель предсказывает данные тренировочной выборки. Для этого также построим график.

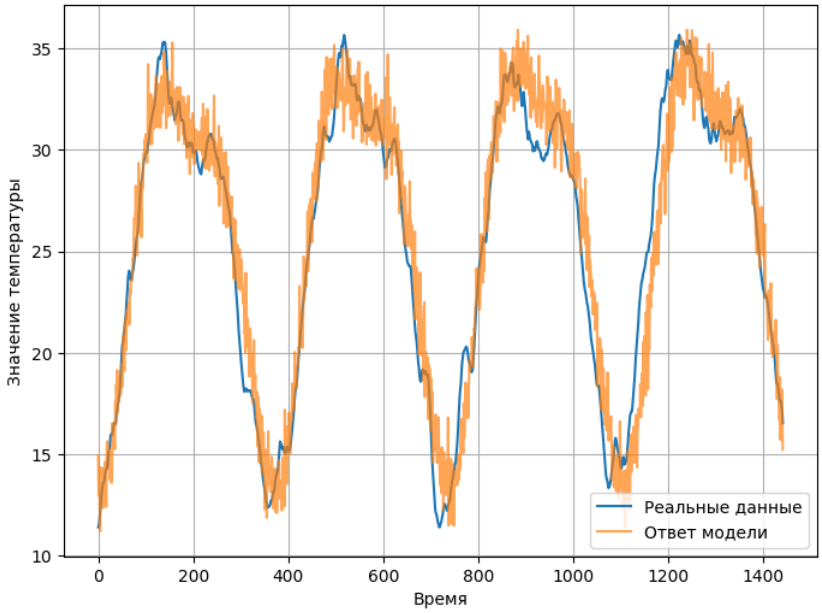


Рис. 18. График изменения ошибки

Из графиков видно, что модель успешно уловила все тренды и периодизацию.

3.4. Вычисление ошибки на тестовой выборке

Для вычисления ошибки модели будем использовать функцию MAE:

$$MAE(y, \bar{y}) = \frac{1}{n}\sum | y - \bar{y} | \qquad (7)$$

Фактически, данная ошибка – среднее отклонение предсказанных данных от действительных. Вычислим её на тестовой выборке (на денормализованных, реальных данных):

```
# Определяем функцию ошибки
def MAE(predictions: np.ndarray, labels: np.ndarray) -> float:
  return np.mean(np.abs(predictions - labels))

# Выводим значение ошибки
print(
  MAE(
    train_normalize_class.DeNormalizeData(
      model(test_data[:, 0]).numpy().T[0], 0
    ) + np.random.normal(size=test_data[:, 0].shape),
    test_data[:, 1],
  )
)
```

```
5.071901681975096
```

Рис. 19. Вычисление ошибки на тестовой выборке

Было совершено преобразование data.numpy().T0, так как модель возвращает тензор TensorFlow формы (114, 1). Это преобразование превращает ответ модели в массив NumPy и выделяет 114 элементов, представляющих собой ответ.

После выполнения кода ошибка модели составила ~5 градусов Цельсия. Таким образом, регрессионная модель прогнозирует температуру на 114 дней вперед с погрешностью чуть больше 5 градусов.

Заключение

В рамках исследования была успешно решена задача анализа временных рядов на примере прогнозирования погодных условий с использованием методов Z-нормализации, градиентного спуска и преобразования Фурье. Для этого была выбрана рекуррентная нейронная сеть.

Литература

1. Dayli Climate time series data. – Режим доступа: https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data. (Дата обращения: 16.04.2024)
2. Руководство по краткосрочным прогнозам погоды : учебник / А. А. Васильев [и др.]; под ред А. А. Васильев. – 3-е изд., стереотип. – Ленинград Гидрометеоиздат, 1986. – С. 138-215.
3. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле. – Санкт-Петербург : Питер, 2018. – 400 с.
4. Прогнозирование временных рядов. – Режим доступа: https://www.tensorflow.org. – (Дата обращения: 17.04.2024)

УДК 004.75

Реализация сервиса ограничителя пропускной способности в распределенной системе

А. О. Варфоломеев, М. В. Матвеева

Воронежский государственный университет

Введение

Как правило в системах со множеством клиентов от разных клиентов исходит разный трафик. Необходимо распределять квоту между клиентами, чтобы не получилось ситуации, когда один клиент потребляет все ресурсы приложения, из-за чего ограничиваются остальные клиенты. Например, такое возможно, когда злоумышленник проводит атаку DDOS [5], посылая большое количество запросов в приложение с целью отказа в обслуживании.

Ограничение пропускной способности в распределенных системах является ключевым аспектом обеспечения их безопасности, надежности и производительности [1,2]. Цель работы — предложить подход к регулированию потока данных и запросов в условиях, где требуется масштабируемость и отказоустойчивость учитывая распределенную среду.

1. Анализ алгоритмов ограничения трафика

Существует два вида ограничения трафика: ***Traffic policing*** и ***Traffic shaping***. Policing ограничивает скорость путём отбрасывания лишнего трафика. Всё, что превышает установленное значение, срезается и отбрасывается. Из-за строгости принимаемых мер это называется ***Hard Policing*** [2]**.**

Shaping ограничивает скорость путём буферизации лишнего трафика. Весь приходящий трафик проходит через буфер. Сервис из этого буфера с постоянной скоростью изымает запросы.

Если скорость поступления пакетов в буфер ниже выходной, они в буфере не задерживаются. В случае если скорость поступления выше выходной, они начинают скапливаться. Выходная скорость при этом всегда одинакова. Таким образом всплески трафика складываются в буфер и будут отправлены, когда до них дойдёт очередь.

Главный недостаток такого подхода — это непредсказуемая задержка, так как при заполнении буфера, пакеты будут находится в нём до тех пор, пока до них не дойдет очередь. Поэтому данный вид ограничения подходит не для всех типов трафика.

Существует несколько основных самых распространенных алгоритмов ограничения трафика [5].

1.1. Алгоритм Leaky Bucket

Leaky Bucket (протекающее ведро) — это алгоритм, который обеспечивает наиболее простой, интуитивно понятный подход к ограничению скорости обработки при помощи очереди, используя механизм FIFO.

Преимущество данного алгоритма состоит в том, что он сглаживает всплески и обрабатывает запросы примерно с одной скоростью, его легко внедрить на одном сервере или балансировщике нагрузки, он эффективен по использованию памяти, так как размер очереди для каждого пользователя ограничен. Однако при резком увеличении трафика очередь

может заполниться старыми запросами и лишить систему возможности обрабатывать более свежие запросы.

Алгоритм является реализацией подхода Traffic Shaping.

1.2. Алгоритм Token Bucket

Token bucket (ведро с токенами) — разновидность алгоритма Leaky Bucket. В такой реализации в буффер с постоянной скоростью добавляются токены, а при обработке запроса токен из буффера удаляется; если же токенов недостаточно, то запрос отбрасывается. Ключевым отличием от реализации Leaky Bucket является то, что токены могут накопиться при простое системы, и дальше может случиться всплеск нагрузки, при этом запросы будут обработаны, так как имеется достаточное количество токенов, в то время как Leaky Bucket гарантированно будет сглаживать нагрузку даже в случае простоя.

Алгоритм является реализацией Traffic policing.

1.3. Алгоритм Fixed Window

В алгоритме *Fixed Window* (фиксированное окно) для отслеживания запросов используется окно, равное n секундам.

Преимущество этого алгоритма состоит в том, что он обеспечивает обработку более свежих запросов, не зависая на обработке старых. Однако одиночный всплеск трафика вблизи границы окна может привести к удвоению количества обработанных запросов, поскольку он разрешает запросы как для текущего, так и для следующего окна в течение короткого промежутка времени, это может привести к резкой повышенной нагрузке на сервис.

1.4. Алгоритм Sliding Log

Алгоритм *Sliding Log (*скользящий журнал*)* предполагает отслеживание временных меток каждого запроса пользователя. Эти записи сохраняются, например, в хеш-множество или таблицу и сортируются по времени. Записи за пределами отслеживаемого интервала отбрасываются.

Преимущество этого алгоритма в том, что он не подвержен недостаткам, которые возникают на границах Fixed Window, то есть ограничение скорости будет строго соблюдаться. Кроме того, поскольку отслеживаются запросы каждого клиента в отдельности, не возникает пикового роста нагрузки в определенные моменты, что является ещё одним недостатком предыдущего алгоритма. Однако хранить информацию о каждом запросе может быть затратно по ресурсам, кроме того, каждый запрос требует вычисления количества предыдущих запросов.

2. Реализация

В ходе анализа вариантов решения, был выбран алгоритм, реализующий Traffic policing — Token Bucket. Этот алгоритм позволяет кратковременные всплески, которые довольно важно заметить в условиях высоконагруженной распределенной системы со множеством клиентов.

В рамках работы рассматривается распределенная система. В качестве глобального хранилища используется NoSQL СУБД ***Redis*** [3].

Алгоритм Token Bucket можно разбить на 3 шага:

1. Получить текущее значение токенов для клиента;

2. Проверить количество токенов, в случае если текущее количество больше, чем требуется для разрешения запроса, разрешить запрос, иначе отправить клиенту 429 код;

3. Обновить токены: по истечению определенного периода времени, количество токенов необходимо обновить.

В Redis хранится 4 значения по ключу client:

1. *quota* — максимальное количество токенов;
2. *last_tokens* — количество токенов на последний момент обращения;
3. *last_updated* — время последнего обращения;
4. *refill* — количество токенов восполняемых в секунду.

2.1. Обращение в Redis

Операции Redis не являются атомарными, это является недостатком в распределенной среде, так как последовательные операции чтения и записи, могут привести к состоянию гонки и непредвиденным результатам. Это означает, что ограничитель скорости может пропускать больше запросов, чем максимально заданное количество.

В качестве решения ситуации состояния гонок, выбран вариант реализации через ***Lua*** [3] скрипт. С помощью Lua скриптов, можно написать хранимые процедуры Redis. Основной их особенностью является то, что они представляют собой атомарные операции, что исключает состояние гонки.

Ниже представлен скрипт, который обращается в Redis, обновляет количество токенов, проверяет достаточно ли токенов чтобы принять запрос и отправляет результат в виде логического значения.

```
-- Входные параметры
-- Key: название клиента
-- Args:
   -- quota: квота на клиента
    -- refill: количество восполняемых токенов в секунду
    -- requested: количество запрашиваемых токенов
    -- now: текущее время
local tokens_key = KEYS[1]..":tokens"          -- Ключ для счетчика токенов
local last_access_key = KEYS[1]..":last_access" -- Ключ для времени последнего доступа
local quota_key = KEYS[1]..":quota"            -- Ключ для квоты
local refill_key = KEYS[1]..":refill"          -- Ключ значения восполняемых токенов
local quota = tonumber(ARGV[1])     -- Квота токенов
local refill = tonumber(ARGV[2])    -- Восполнение токенов в секунду
local requested = tonumber(ARGV[3]) -- Количество запрашиваемых токенов
local now = tonumber(ARGV[4])       -- Текущее время в микросекундах
-- Проверяем, существует ли ключ, если нет, то инициализируем его
local key_exist = tonumber(redis.call("get", quota_key))
if key_exist == nil then
   redis.call("setex", tokens_key, 3600, quota)     -- инициализируем счетчик на максимум
   redis.call("setex", last_access_key, 3600, 0)    -- инициализируем время последнего доступа
   redis.call("setex", refill_key, 3600, refill)    -- инициализируем количество восполняемых токенов в секунду
   redis.call("setex", quota_key, 3600, quota)      -- инициализируем квоту
end
```

```
local last_access = tonumber(redis.call("get", last_access_key))
local last_tokens = tonumber(redis.call("get", tokens_key))
local elapsed = math.max(0, now - last_access)
local add_tokens = math.floor(elapsed * refill / 1000000)
local new_tokens = math.min(quota, last_tokens + add_tokens)
-- Считаем новое время последнего доступа.
-- Не используем текущее время, так как это приведет к ошибке округления.
local new_access_time = last_access + math.ceil(add_tokens * 1000000 / refill)

-- Проверяем достаточно ли токенов
local allowed = new_tokens >= requested
if allowed then
   new_tokens = new_tokens - requested
-- Обновляем состояние
redis.call("setex", tokens_key, 3600, new_tokens)
redis.call("setex", last_access_key, 3600, new_access_time)
redis.call("setex", quota_key, 3600, quota)
redis.call("setex", refill_key, 3600, refill)

return allowed
```

2.2. Встраивание в код программы

Для дальнейшего использования в программе, необходимо реализовать *Middleware [5]* — код, который выполняется для каждого входящего запроса.

Функция createRateLimiterMiddleware создает middleware. В функции происходит извлечение имени клиента из заголовков запроса, затем вызывается метод AllowRequest, который возвращает логическое значение, означающее достаточно ли токенов для пропуска запроса. В случае если токенов не хватает, то клиенту возвращается HTTP код 429 — Too Many Requests, сигнализирующий о том, что сервис принимает слишком много запросов и не может их все обработать.

Функция AllowRequest обращается в Redis используя Lua скрипт, описанный выше.

```
func createRateLimiterMiddleware(redis redisClient.Client, rateLimiterQuota, rateLimiterRefill, rateLimiterRequested int, ) echo.MiddlewareFunc {
    return func(c echo.Context) error {
      client:= getClient(c)
      allow := AllowRequest(
        c.Request().Context(),
        redis,
        token,
        rateLimiterQuota,
        rateLimiterRefill,
        rateLimiterRequested,
      )

      if allow {
        return next(c)
      }
```

```
            c.Response().Header().Set("Retry-After", 60)
            return c.NoContent(http.StatusTooManyRequests)
        }
    }


    func AllowRequest(ctx context.Context, client redisClient.Client, group string, quota, refill, requested int) bool {
        s := redis.NewScript(string(allowRequestScript))
        runScript := client.RunScript(ctx, s, []string{group}, quota, refill, requested, time.Now().UnixMicro())

        return runScript.bool()
    }
```

Заключение

Целью работы было проанализировать различные варианты решения задачи ограничения входящего трафика, а также реализовать сервис, который мог бы работать в условиях распределенной системы.

Был реализован middleware на языке Golang [4], который в дальнейшем можно встроить в любой сервис для ограничения трафика.

Решение базируется на использовании алгоритма Token Bucket, который позволяет гибко управлять распределением ресурсов и обрабатывать входящий трафик с учетом заданных ограничений.

Чтобы поддерживать синхронизацию и согласованность данных, использованы Lua скрипты, представляющие собой атомарные операции в Redis [3].

Результат работы сервиса – ограничение входящих запросов для каждого клиента.

Научный руководитель — старший преподаватель кафедры программного обеспечения и администрирования информационных систем факультета ПММ ВГУ, Матвеева Мария Валерьевна.

Литература

1. Таненбаум Э. Распределенные системы. Принципы и парадигмы / Э. Таненбаум – Санкт-Петербург: Питер, 2003. – 877 с

2. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка/ М. Клеппман – Санкт-Петербург: Питер, 2024. - 640 с

3. Документация Redis. – Режим доступа: https://redis.io/docs/latest/. – (дата обращения: 01.04.24).

4. Documentation Golang. – Режим доступа: https://go.dev//. – (дата обращения: 01.04.24).

5. Таненбаум Э. Компьютерные сети/ Э. Таненбаум, Н. Фимстер, Д. Уэзерол – 6-е изд., перераб. и доп. – Санкт-Петербург: Питер, 2023. – 992 с

УДК 004.93

МОДЕЛИ РАСПОЗНАВАНИЯ СОСТОЯНИЯ УСТАЛОСТИ ВОДИТЕЛЯ АВТОТРАНСПОРТА

К. Р. Ватутин, О. Ю. Лавлинская

Воронежский государственный университет

Введение

Высокий уровень стресса, сонливость и недостаток концентрации внимания являются ключевыми факторами, которые увеличивают риск возникновения опасных ситуаций на дорогах, вплоть до аварий. Эти факторы стимулируют развитие исследований в области предотвращения дорожно-транспортных происшествий. Основная цель этих исследований – разработка устройств и механизмов, которые могли бы эффективно мониторить и оценивать поведение водителей во время управления транспортным средством. Такие технологии способствуют увеличению безопасности дорожного движения, снижая вероятность возникновения аварийных ситуаций.

Вождение автомобиля требует строгого соблюдения правил безопасности, которые напрямую связаны с личной безопасностью водителя и безопасностью дорожного движения. Поэтому контроль за поведением водителя во время вождения является критически важной задачей. Безопасность на дорогах обеспечивается различными методами. Во-первых, автопроизводители используют разнообразные средства для поддержания внимания водителей: специализированная дорожная разметка и покрытия, разработанные для того, чтобы стимулировать бодрость водителей, а также голосовые помощники и бортовые компьютеры в навигационных системах, которые предупреждают о необходимости сделать перерыв в поездке. Во-вторых, современные технологии распознавания позволяют определять усталость водителя по биометрическим данным. На основании полученной информации о состоянии водителя вырабатывается управляющий сигнал или звуковое оповещение для предотвращения опасной или аварийной ситуации на дороге [1].

Актуальной задачей является разработка инструментальных подходов к оценке состояния водителя на основе моделей идентификации лица с учетом положения глаз, рта, положения головы (поворота, наклона). Рассмотрим существующие модели оценки состояния водителя с целью выбора наилучших моделей оценки усталости водителя автотранспорта в условиях городского движения для реализации собственного программного решения для оповещения водителя о необходимости остановки движения в состоянии сонливости.

Разработка методов для оценки уровня бодрости водителя через анализ его лица, включая положение глаз, рта и ориентацию головы (наклоны и повороты), становится важной задачей. Рассмотрим существующие подходы к мониторингу состояния водителя, чтобы выбрать наиболее эффективные модели для определения усталости. Целью является разработка программного продукта, который будет предупреждать водителя о необходимости остановиться, если обнаружены признаки сонливости.

1. Оценка качества идентификации лица в моделях распознавания

1.1 Оценка качества входных данных модели

Подход к оценке качества изображений включает в себя анализ таких факторов, как разрешение, контрастность, освещенность и размытие. Важно учитывать резкость изображения в области глаз, так как именно там происходит фокусировка внимания для анализа параметров взгляда. Использование традиционных методов оценки качества изображений может быть дополнено анализом областей, содержащих глаза и периферийных областей лица, для определения качества изображения более точно. Например, вместо простого анализа всего изображения лица после его обнаружения, можно выделить области глаз и применить к ним более детальную оценку качества изображения.

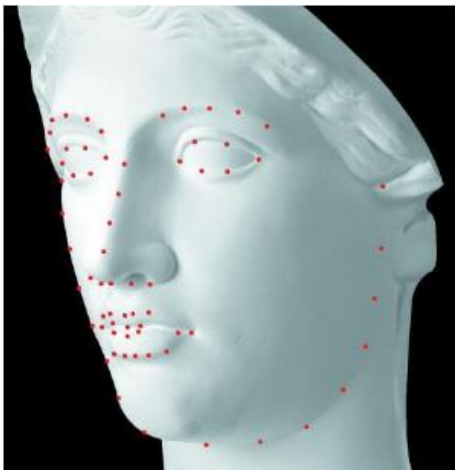


Рис. 1. Схематическая диаграмма ключевых точек лица

Процесс обучения модели, основанной на методе слежения за взглядом, является ключевым элементом для повышения эффективности систем распознавания усталости водителя. Этот процесс включает в себя следующие детализированные этапы:

1) Определение областей интереса, связанных с глазами, на изображениях лиц. Для эффективного слежения за взглядом необходимо точно определить, где именно находятся глаза на лице. Это достигается с помощью CNN алгоритма компьютерного зрения, позволяющего автоматически обнаруживать и выделять ключевые точки лица.
2) Обучение модели на наборе изображений, включающем различные условия освещения.
3) Оценка производительности модели на тестовом наборе данных, включающем изображения лиц с разными уровнями качества.

1.2 Математическая модель обнаружения ключевых точек лица

Рассмотрим алгоритм PFLD, который представляет собой упрощенный алгоритм обнаружения с высокой точностью, небольшой по размерности моделью и высокой скоростью обработки данных, и обеспечивает производительность сверхреального времени на мобильном терминале [3].

PFLD представляет собой эффективный детектор ключевых точек лица, который демонстрирует высокую точность в различных сложных условиях, включая непостоянное освещение, окклюзию, разнообразие выражений лица и различные позы. В его основе лежит алгоритм, который использует специальную ветвь сети для анализа геометрической структуры каждого изображения лица и последующей коррекции расположения ключевых точек. Далее представлено подробное описание функции потерь, основной и вспомогательной подсетей этого алгоритма.

С точки зрения функции потерь обычно используются следующие варианты: MSE, Softmax + потеря перекрестной энтропии, Smooth L1 [3] и т. д. Чтобы сбалансировать обучающие данные в различных ситуациях и учесть экстремальные ситуации, которые могут

возникнуть в реальности (например, обнаружение лиц в случай окклюзии лица) алгоритм может добавлять закрытые лица в данные обучения и обеспечивать согласованность пропорций данных обучения в различных ситуациях. Этот метод может эффективно повысить производительность, контролируя форму выборки данных. И PFLD использует новую функцию потерь для решения проблемы дисбаланса выборки в различных ситуациях. Математическая модель представлена формулой 1:

$$L = \frac{1}{M} \sum_{m=1}^{M} \sum_{n=1}^{N} \left(\sum_{c=1}^{C} w_n^c \sum_{k=1}^{K} \left(1 - \cos \theta_n^k\right) \right) d_{n2}^{m2} \quad (1)$$

где W_n^c — регулируемая весовая функция (разные веса выбираются для разных ситуаций, таких как нормальная ситуация, ситуация с окклюзией, ситуация с темным освещением и т. д.), θ — трехмерный угол Эйлера позы лица (K =3), d — регрессионный ориентир и метрика Ground-True (как правило, также может быть выбрана метрика MSE, L1) [4].

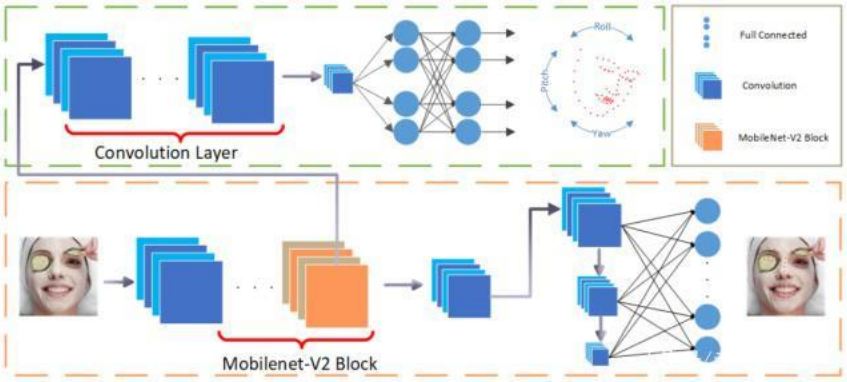


Рис. 2. Общая структура PFLD

Задача функции потерь заключается в том, чтобы присваивать меньший вес данным с более крупным объемом выборки, таким как фронтальные лица (ситуации с небольшими углами Эйлера), в процессе обратного распространения градиента. Архитектура функции потерь модели решает проблему несбалансированных обучающих выборок, рис. 2. С точки зрения магистральной сети, чтобы удовлетворить потребности устройств с низкой вычислительной мощностью, таких как встроенные или мобильные терминалы, при этом структура сети MobileNet v2 [4] урезана, чтобы уменьшить количество избыточных параметров, и для извлечения признаков используется упрощенная модель, выходные признаки модели структурно изменены для повышения выразительности модели.

Наконец, чтобы объективно сравнить производительность различных алгоритмов ключевых точек лица, в нашем подходе, в качестве индекса оценки алгоритма используется средняя ошибка нормализации (NME).

Формула расчета выглядит следующим образом:

$$e = \frac{\sum_{i=1}^{N} x_i - x_i^*}{N * d}, \quad (2)$$

где x_i — предсказанная i координата, x_i* — i-й слева буква Ground-Truth, d — евклидово расстояние между двумя координатными точками, а N — количество ключевых точек.

2. Алгоритм обнаружения усталости при вождении

Алгоритм обнаружения усталости при вождении, использующий технологию айтрекинга (Eye Tracking), предоставляет точный метод анализа различных параметров глазного поведения водителя, чтобы выявить признаки утомления. Основываясь на детальном наблюдении за

направлением взгляда, частотой моргания, изменением размера зрачка и анализе коэффициента EAR (Eye Aspect Ratio), этот подход позволяет эффективно оценивать уровень усталости водителя.
Методы и алгоритмы айтрекинга:

1) Мониторинг направления взгляда: Алгоритмы, используемые для отслеживания направления взгляда, основаны на распознавании положения зрачка и его движении относительно глазной ямки. Используя камеры, системы айтрекинга способны точно фиксировать направление взгляда в реальном времени.

2) Анализ частоты моргания: Частота моргания может указывать на уровень усталости, поскольку с увеличением утомления частота моргания увеличивается. Для этого анализа система регистрируют количество морганий за определенный временной интервал.

3) Определение размера зрачка: Размер зрачка может варьироваться в зависимости от уровня усталости; обычно, при утомлении зрачки становятся менее реактивными и могут расширяться. Методы измерения размера зрачка включают анализ видеоизображений, на которых фиксируется диаметр зрачка в различных световых условиях.

4) Расчет Eye Aspect Ratio (EAR): EAR — это количественный параметр, измеряющий соотношение открытости глаз. Он рассчитывается как отношение вертикального расстояния между веками к горизонтальному расстоянию между угловыми точками глаза. Уменьшение EAR указывает на увеличение усталости, поскольку веки водителя начинают закрываться.

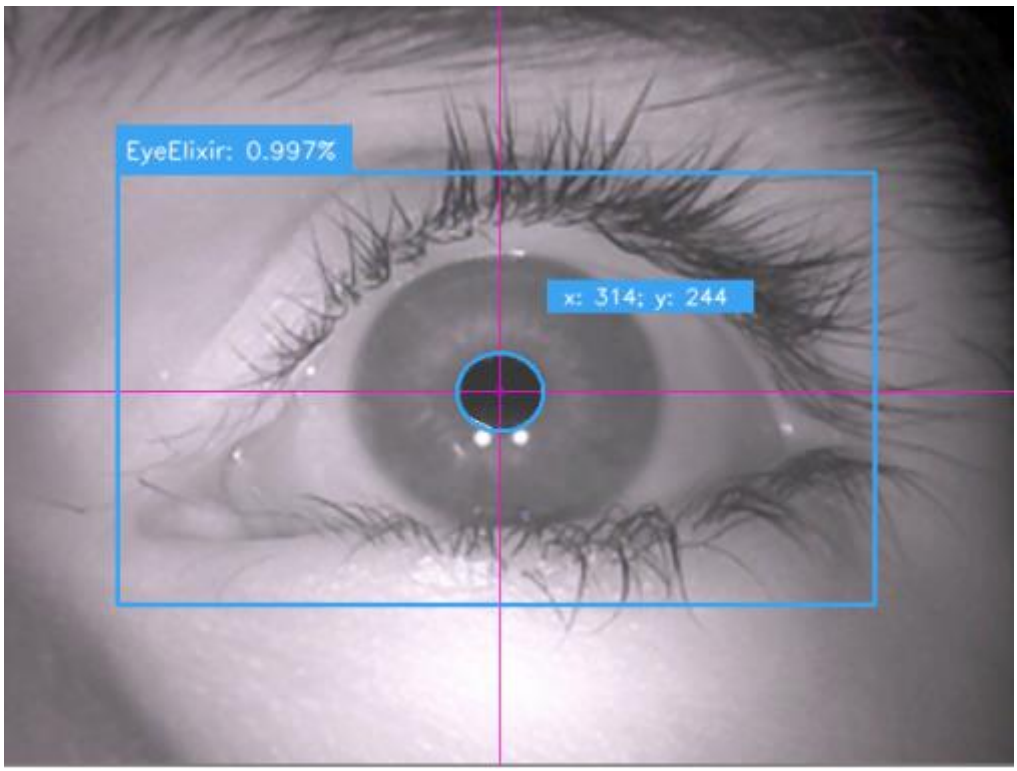


Рис. 3. Обнаруженный глаз и зрачок

Заключение

В заключении отметим, что представленные в статье модели распознавания усталости водителя реализованы в различных инструментальных библиотеках, таких как:

PyTorch - библиотека машинного обучения с открытым исходным кодом, упрощающая создание и обучение моделей глубокого обучения, в том числе для распознавания лиц и построения 3D-моделей. В ней реализован алгоритм PFLD для расставления ключевых точек на лице.

TensorFlow – это библиотека машинного обучения с открытым исходным кодом, разработанная компанией Google. Она позволяет создавать и обучать различные модели глубокого обучения, включая алгоритм CNN (Convolutional Neural Network).

Литература

1. Петросянц Д. Г. Нейросетевая сверточная модель анализа зрачковых реакций для оценки функционального состояния усталости водителей автотранспортных средств.\ Д. Г. Петросянц и др.\\ ФГБОУ ВО «Казанский национальный исследовательский технический университет им. А. Н. Туполева – КАИ», Казань — 2019. — с. 145–152.
2. Кореневский Н. А. Нечеткие модели оценки уровня эргономики технических систем и её влияния на состояние здоровья человека-оператора с учетом функционального резерва организма. \ Н. А. Кореневский, С. Н Родионова. Т. Н Говорухина. М. А Мясоедова \\ Моделирование, оптимизация и информационные технологии. 2019;7(1). Доступно по: https://moit.vivt.ru/wp-content/uploads/2019/01/KorenevskiySoavtori_1_19_1.pdf
3. Хэ Цзяньбао, Оценка и анализ небезопасного поведения водителя [J], Times Auto, 2021(05): 175–176.
4. Го Биньлин. Анализ небезопасного поведения водителей автомобилей [J] Техническое обслуживание автомобилей и вождения (Maintenance Edition), 2018(06): 80–82.
5. Статья [Электронный ресурс] — Режим доступа: http://habrahabr.ru/post/73501/ (дата обращения 17.04.2024)
6. Спирин, И. А. Исследование и применение eye-tracking технологии на человеке / И. А. Спирин. — Текст: непосредственный // Молодой ученый. — 2016. — № 2 (106). — С. 227-230. — URL: https://moluch.ru/archive/106/25349/ (дата обращения: 17.04.2024).

УДК 004.896

СИСТЕМЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ

Е. С. Вернер

Воронежский государственный университет

Введение

Системы технического зрения служат в робототехнических комплексах для получения изображений для распознавания и классификации различных объектов, а также для обработки информации о состоянии окружающей среды. Данная проблема решается в несколько этапов: получение и обработка графической информации, затем сегментация, детекция или нахождение границ объекта, после этого непосредственно распознавание образов и дальнейшая интерпретация полученных результатов.

1. Структура системы технического зрения

1.1. Строение системы технического зрения

Современная робототехника предполагает возможность перемещения роботов в окружающей среде, подверженной изменениям, а также выполнение функций, требующих наличия технического зрения. Чаще всего робототехнические комплексы выполняют какую-либо определённую задачу и адаптированы под определённые условия.

При разработке робототехнических комплексов нужно учитывать, какие задачи поставлены перед ним, чтобы подобрать необходимые сенсорные устройства для их решения. Так, например, для решения задачи перемещения внутри рабочей зоны робот может быть оснащен лазерными сканирующими дальномерами и GPS-устройствами для определения собственного положения, тогда как для решения задачи локальной навигации робот может быть оснащен ультразвуковыми или инфракрасными датчиками по периметру. Однако, все вышеперечисленные средства не могут дать роботу полной картины происходящего вокруг него, так как использование различных датчиков определения расстояния позволяют роботу определять расстояние до объектов и их габариты, но не позволяют определять другие их свойства – форму, цвет, положение в пространстве, что приводит к невозможности классифицировать такие объекты по каким–либо критериям.

Существуют различные решения, основанные на разных принципах определения положения, например, на базе анализа структурированного света, на основе времяпролетных принципов, стереоскопические решения с выделением особых точек изображения. Тем не менее, популярностью по-прежнему пользуются классические решения на основе цифровых видеокамер, так как они позволяют за небольшие вложения получить легко настраиваемый вариант системы определения координат и класс объекта высокой надежностью и простотой установки. Изображения разбивается отдельные пиксели, предварительно происходит их обработка, и система пытается сделать выводы с помощью системы распознавания образов.

1.2. Методы фильтрации контуров

В системе технического зрения информация об изображении, при помощи оптико-электронных преобразователей и видео датчиков, представляется в форме электрических

сигналов. Это, по существу, первичное преобразование. Обычно изображение считывается при помощи оптической камеры, чувствительного элемента, сканирующего устройства, затем сигнал усиливается. Полученная таким образом информация обрабатывается иерархически. Сначала изображение обрабатывается видеопроцессорами. Здесь ключевой параметр — контур изображения, который задается координатами множества составляющих его точек. Существует несколько видов алгоритмов, решающих задачу фильтрации контуров: оператор Лапласа, оператор Робертса, оператор Собеля, оператор Кэнни и другие. Реализация многих фильтров есть в библиотеке OpenCV.

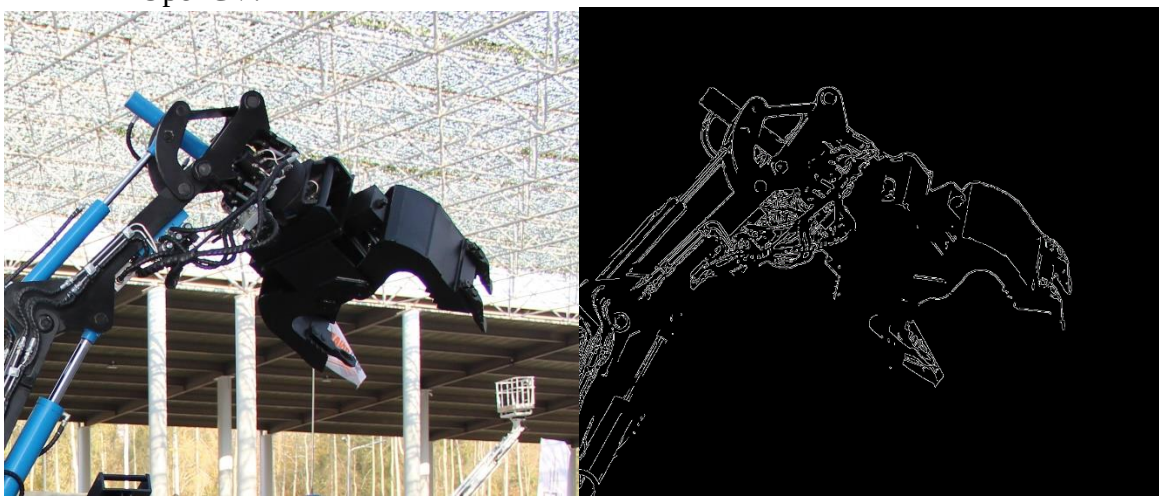


Рис. 1 Навесное оборудование – грейфер с зубьями до и после применения нахождения контура с помощью оператора Кэнни.

Основными этапами работы алгоритма Кэнни являются сглаживание, при котором происходит удаление шумов, поиск градиентов, для которого алгоритм Кэнни использует четыре фильтра нахождения горизонтальных, вертикальных и диагональным рёбер, подавление немаксимумов, когда только локальные максимумы отмечаются как границы, применение двойной пороговой фильтрации для определения потенциальных границ, трассировка области неоднозначности, заключающаяся в подавлении всех краёв, не связанных с определёнными границами.

Фильтр границ Кэнни является одним из самых лучших алгоритмов нахождения контуров изображений, так как хорошо обнаруживает рёбра изображения, их положение и совершает единственный отклик на одну границу.

1.3. Этапы работы системы технического зрения

На первом этапе работы системы технического зрения формируются управляющие сигналы для робота. Видео датчики присоединяются к другим частям системы технического зрения при помощи особых кабелей, например оптоволоконных, через которые информация передается на высокой частоте и с минимальными потерями. Сами видео датчики могут иметь точечные, одномерные или двумерные чувствительные элементы. Точечные чувствительные элементы способны принимать видимое излучение с мелких частей объекта, и для получения полного растрового изображения необходимо произвести сканирование по плоскости. Одномерные чувствительные элементы более сложны, они состоят из линейки точечных элементов, которые в процессе сканирования движутся относительно объекта. Двумерные элементы представляют собой, по сути, матрицу дискретных точечных элементов. Оптическая система проецирует изображение на чувствительный элемент, предварительно определяется размер рабочей зоны, охватываемой датчиком. Оптическая система имеет линзовый объектив с настраиваемой диафрагмой, чтобы производить регулировку количества поступающего света и

фокусировку четкости, когда расстояние от объектива до объекта изменяется. Это низший уровень системы.

Далее идет анализ, описание и распознавание — здесь используются современные компьютеры и сложное программно-алгоритмическое обеспечение — средний уровень. Высший уровень — это уже искусственный интеллект. Широко распространены в промышленности системы технического зрения, обеспечивающие надлежащее качество работы с плоскими изображениями и объектами простых форм. Они служат в распознавании, сортировке и укладке деталей, в проверке размеров деталей, сопоставлении их с чертежом и т. д.

Активно разрабатываемые сегодня интеллектуальные и адаптивные роботы способны работать с трехмерными изображениями и более сложными объектами, производить более точные измерения, более внимательно и быстро распознавать объекты. Главное направления научно-технического поиска сегодня — совершенствование систем технического зрения и программно-алгоритмического обеспечения к ним, создание специальных компьютеров, а также принципиально новых систем технического зрения, ведь применение робототехники все больше востребовано и область ее промышленного внедрения непрерывно расширяется.

Существует несколько подходов к построению системы распознавания образов, одни из них основаны на алгоритмах контурного анализа, так как контур объекта часто является его уникальной характеристикой, другие же основаны на поиске особых точек, остающихся стабильными при смене освещения или угла зрения.

Для построения системы распознавания можно использовать алгоритмы классификации и кластеризации, такие как k-means и AdaBoost, применять нейронные сети, а также создать искусственную иммунную систему.

2. Алгоритм работы искусственной иммунной системы

Алгоритм работы искусственной иммунной системы можно разделить на два этапа. Первый этап – это непосредственно обучение модели, а второй этап – тестирование и работа с моделью. Обучение модели проводится один раз и представляет собой обучение без учителя, то есть не требует вмешательства экспериментатора в процесс, второй же этап можно проводить сколь угодно большое количество раз. В процессе обучения система должна научиться самостоятельно находить закономерности принадлежности к определённому классу объекта.

Обучение искусственной иммунной системы является достаточно зависимым от ресурсов этапом. Требуется достаточно много времени и хорошие вычислительные возможности для обучения на больших датасетах. Скорость обучения так же зависит от размера изображения и количества генерируемых лимфоцитов.

Процесс обучения искусственной иммунной системы можно представить в виде следующей схемы:

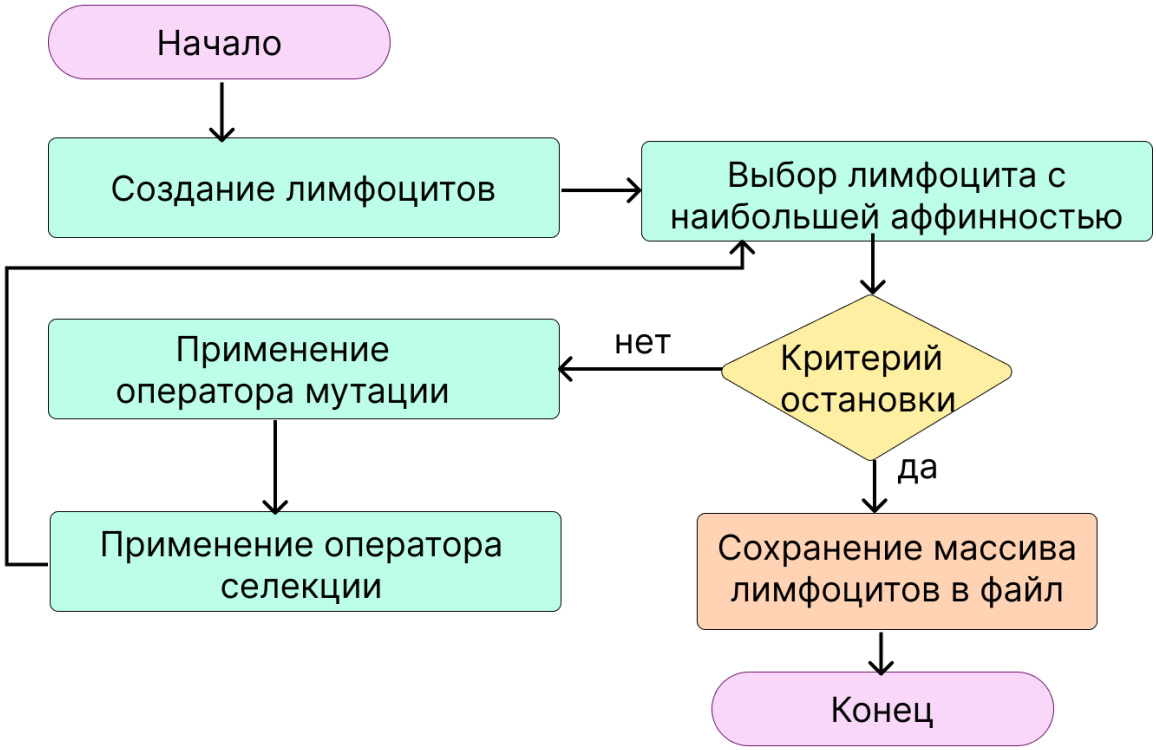


Рис. 2. Обучение модели искусственной иммунной системы

Алгоритм распознавания чаще всего состоит из нескольких частей: предварительной обработки данных, обучения и тестирования. Алгоритм, основанный на использовании искусственной иммунной системы, предполагает обучение без учителя. Системе самостоятельно необходимо обнаружить внутренние зависимости для определения объектов на изображении.

Последовательность действий выглядит следующим образом:

1. На первом этапе происходит предварительная обработка изображений – это один из самых важных этапов работы программы.

1) Сначала происходит перевод изображения в серый цвет для того, чтобы сократить время выполнения работы программы.

2) Затем изображение бинаризуется.

3) Если изображения большого размера, они масштабируются с помощью алгоритма Брезенхема.

2. Затем происходит генерация лимфоцитов случайным образом. Лимфоцит представляется в виде класса, который хранит массив чисел, метку (соответствующую антигену) и дату последней активации.

3. На вход системе подаётся массив картинок – антигены, в данном этапе участвуют только картинки из обучающей выборки. Число всех возможных антигенов небольшое и известно заранее. Изображения представлены в виде массива из 0 и 1, где 0 соответствует белому цвету, а 1 – чёрному.

4. Для каждого лимфоцита вычисляется мера аффинности к текущему антигену, если она больше порогового значения. Мера аффинности вычисляется по формуле расстояния Хэмминга, которое представляет собой количество позиций, в которых соответствующие символы различаются.

5. Далее в полученном массиве находится элемент с наилучшим показателем аффинности.

6. Затем применяется оператор мутации к 20% лимфоцитов, обладающих высоким показателем аффинности, происходит случайная перестановка фрагментов лимфоцита.

7. Оператор селекции оставляет лучшие лимфоциты из текущего набора и множества мутаций. При этом каждый лимфоцит хранит отметку времени о последней активации. Для того, чтобы лимфоциты, когда-либо активированные вообще, не были удалены, необходимо периодически давать системе для распознавания часть обучающей выборки. Те лимфоциты, которые не активируются, подлежат удалению.

8. Если показатель аффинности достиг желаемого порога или если превышено заданное число итераций, алгоритм заканчивает свою работу. Процесс обучения системы заканчивается. В противном случае происходит возвращение на 4й шаг алгоритма.

В качестве меры аффинности чаще всего применяются расстояния Хэмминга или Левенштейна.

$$D(i,j) = \begin{cases} 0, & i = 0,\ j = 0 \\ i, & j = 0,\ i > 0 \\ j, & i = 0,\ j > 0 \\ \min\{ & \\ \quad D(i,j-1) + 1, & \\ \quad D(i-1,j) + 1, & j > 0,\ i > 0 \\ \quad D(i-1,j-1) + \mathrm{m}(S_1[i], S_2[j]) & \\ \} & \end{cases},$$

Рис. 3 Формула нахождения расстояния Левенштейна

Заключение

В данной статье описывается алгоритм работы системы технического зрения, алгоритм обучения искусственной иммунной системы в задаче распознавания образов, а также аппарат для подготовки входных данных.

Был рассмотрен алгоритм нахождения контуров с помощью оператора Кэнни.

Предложенный алгоритм работы искусственной иммунной системы был реализован с помощью языка Python 3.

Литература

1. А. В. Селеменев. Применение искусственных иммунных систем для обнаружения сетевых вторжений / А. В. Селеменев [и др.]; // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. - 2019. - № 2.- С.50-56.

2. И.Ф. Астахова, Model and algorithm of an artificial immune systems for the recognition of single symbols and their comparison with existing methods / И.Ф. Астахова [и др.] // WSEAS Trans. on Information Science and Applications 13 (2016). – p. 37–43.

3. Астахова И.Ф. Модели распознавания образов на основе нечетких нейронных сетей. Практическое применение / И.Ф. Астахова, В.А.Мищенко, А.В. Краснояров. — Berlin: Palmarium Academic Publishing. — 2013. — 104 с

4. Rosset, Zhu and Hastie. Boosting as a Regularized Path to a Maximum Margin Classifier. Journal of Machine Learning Research 5 (2004) 941–973 c.

УДК 004.4

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ BLAZOR WEBASSEMBLY

С. Д. Виноградов, К. Г. Резников

Воронежский государственный университет

Введение

Длительное время написание клиентских веб-приложений было уделом языка программирования JavaScript [1]. Появление в 2015 году WebAssembly и начавшееся постепенное внедрение поддержки этой технологии в наиболее популярные браузеры начало менять данную тенденцию. WebAssembly – это результат компиляции с языка программирования высокого уровня, представляющий собой двоичный формат команд для виртуальной машины [2]. На основе этой технологии была разработана подсистема Blazor под названием Blazor WebAssembly. Blazor представляет собой UI-фреймворк для создания интерактивных приложений на платформе .NET. Вышедшая в мае 2020 года подсистема позволила разработчикам создавать интерактивные приложения клиентской стороны на языке программирования C#. По причине того, что данная технология появилась сравнительно недавно, внимание уходит в пользу других более популярных и проверенных аналогов. В данной статье рассмотрим преимущества и недостатки Blazor WebAssembly, а также особенности использования и интеграции технологии в проекты.

1. Альтернативные решения и особенности Blazor WebAssembly

Среди аналогов Blazor WebAssembly стоит выделить следующие технологии:

1. Angular – TypeScript фреймворк для создания клиентских приложений с поддержкой языков JavaScript и Dart. Разработчиком фреймворка является корпорация Google, которая использует Angular для создания своих внутренних инструментов и общедоступных веб-сайтов. Фреймворк регулярно обновляется, имеет техническую поддержку и исчерпывающую документацию. Главным недостатком Angular является сложность фреймворка, вследствие чего он не подходит для создания небольших приложений, но при этом является одним из лучших решений при создании комплексных систем.
2. React.js – JavaScript библиотека, которая не являетсмя фреймворком, а в следствии чего не навязывает конкретную архитектуру приложения, позволяя разработчику использовать ее вместе с другими инструментами по своему усмотрению. Из преимуществ библиотеки React.js стоит выделить JSX синтаксис, позволяющий описывать пользовательский интерфейс приложений более удобным и наглядным образом. Также важным преимуществом React.js является наличие виртуального DOM [3]. Виртуальный DOM повышает производительность приложения благодаря тому, что все изменения, вызванные взаимодействием пользователя или другими событиями, сначала обрабатываются в нем, а уже потом переносятся на реальный, что приводит к сокращению количества обновлений дерева при тривиальных изменениях. Главным недостатком React.js является то, что технология охватывает только уровень пользовательского интерфейса приложения, вследствие чего возникает потребность в дополнительных инструментах для разработки [4]. Отдельно стоит выделить быстрое развитие React.js, что является как преимуществом, так и недостатком. С

одной стороны частые обновления привносят новые опции для разработки, с другой – приходится подстраиваться под нововведения и вносить правки в код, в то время как надлежащей документации еще не существует.

3. Vue.js – фреймворк для разработки пользовательских интерфейсов и одностраничных приложений на языке программирования JavaScript. Основными преимуществами данной технологии являются ее маленький размер, наличие виртуального DOM, трехэлементная компонентная архитектура и гибкость, которая обеспечивается тем, что Vue.js опирается собственный внутренний программный комплекс и не требует никаких других инструментов для работы [5]. Для написания шаблонов, помимо JavaScript, допускается использование HTML и JSX. Среди недостатков Vue.js стоит выделить сложность разработки крупномасштабных проектов, так как технология ориентирована на средние, малые и гибридные проекты.

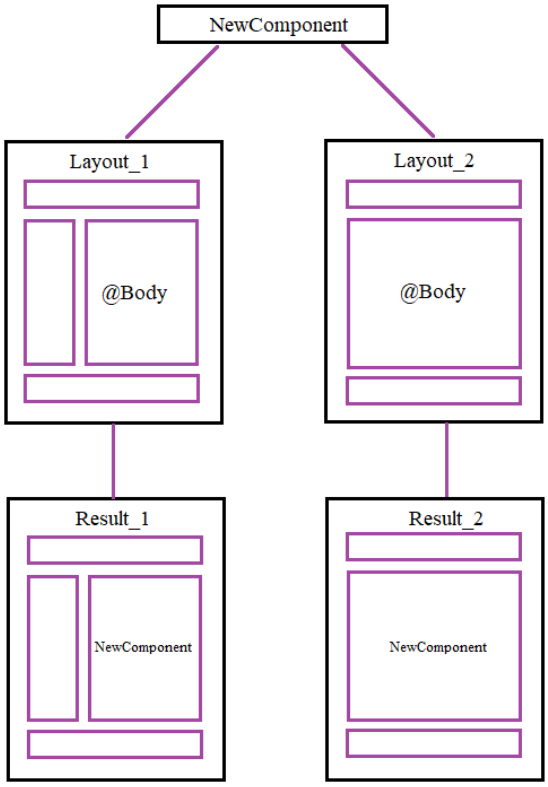


Рис. 1. Схематичное представление механизма слоев

Ключевой особенностью приложений Blazor WebAssembly является то, что подобно одностраничным веб-приложениям, разработанным с помощью других инструментов, они работают на стороне клиента. Во время запуска все файлы, зависимости и среда выполнения .NET загружаются браузером, что и обуславливает основные преимущества данного фреймворка: более быстрое и отзывчивое взаимодействие с пользователем, поскольку обновление интерфейса происходит без обращения к серверу, возможность автономной работы и упрощенное масштабирование, так как при обращении запускается копия приложения. Данная технология значительно снижает нагрузку на сервер, при этом дополнительно существует возможность бессерверного развертывания приложения при передаче всех необходимых файлов браузеру. Для работы с Blazor WebAssembly не требуется знание JavaScript, а в качестве языка программирования применяется C# [6]. Для описания пользовательского интерфейса используются стандартные HTML и CSS. Blazor WebAssembly

использует компонентную структуру с возможностью выбора одного из трех подходов: Mixed (в компоненте находится и верстка, и сам код), Partial class (разделение блоков на отдельные файлы) и Base class (наследование). Каждый компонент имеет расширение *.razor*. Также существует возможность использования слоев – блоков кода, необходимых для более удобного написания приложения (рис. 1).

Рассмотрим процесс интеграции Blazor WebAssembly в проекты.

2. Интеграция Blazor WebAssembly в проекты

Для добавления Blazor WebAssembly в проект, Microsoft Visual Studio (интегрированная среда разработки программного обеспечения) по умолчанию предоставляет несколько шаблонов приложений (рис. 2).

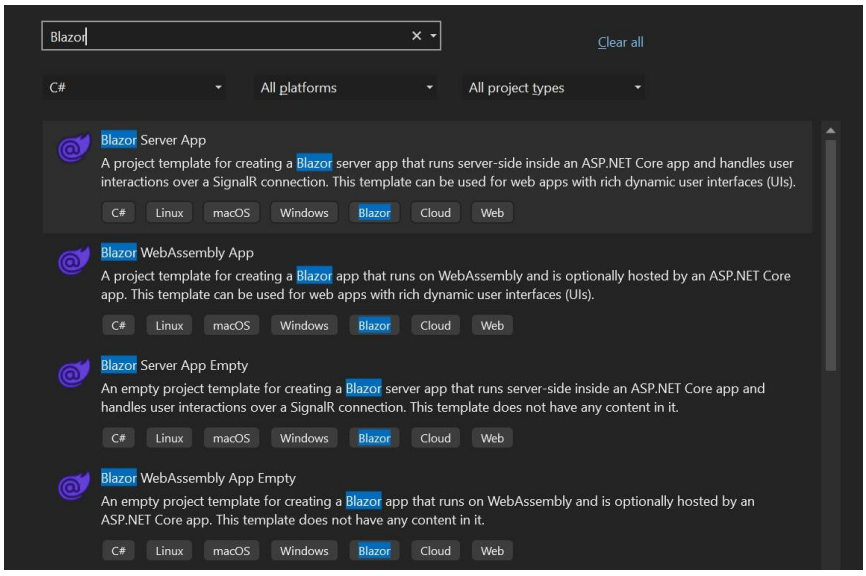


Рис. 2. Шаблоны Blazor на примере Microsoft Visual Studio 2022

Структура приложения Blazor WebAssembly на примере разрабатываемого проекта представлена на рис. 3.

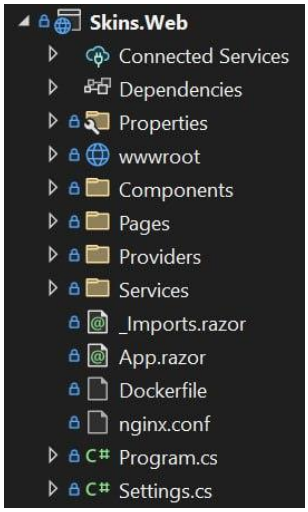


Рис. 3. Структура приложения Blazor WebAssembly

Blazor WebAssembly предоставляет возможность сбора всех необходимых страницам директив using (для пространств имен) в одном файле *_Imports.razor* (рис. 4).

```
@using System.Net.Http
@using System.Net.Http.Json
@using Microsoft.AspNetCore.Components.Routing
@using Microsoft.AspNetCore.Components.Web
@using Microsoft.AspNetCore.Components.WebAssembly.Http
@using Microsoft.JSInterop
@using Skins.Web
@using Skins.Web.Pages.Shared
@using Skins.Web.Components
@using MudBlazor
@using Microsoft.AspNetCore.Authorization
@using System.ComponentModel.DataAnnotations
@using Microsoft.AspNetCore.Components.Forms
```

Рис. 4. Файл _Imports.razor

Благодаря наличию в Microsoft Visual Studio системы NuGet, упрощающей процесс подключения сторонних библиотек и пакетов в проекты .NET, появляется возможность существенного сокращения времени разработки приложения благодаря использованию уже готовых решений от других пользователей. Ярким примером тому является библиотека MudBlazor, которая значительно упрощает верстку, содержа в себе большое количество компонентов на выбор, использование которых сводится к замене на них стандартных HTML компонентов [7].

Для связи с контроллерами API, в создаваемом приложении присутствуют сервисы, к которым обращаются страницы для получения и передачи информации. Все сервисы подключаются в файле *Program.cs* (рис. 5).

```
using Blazored.LocalStorage;
using Skins.Web;
using Microsoft.AspNetCore.Components.Authorization;
using Microsoft.AspNetCore.Components.Web;
using Microsoft.AspNetCore.Components.WebAssembly.Hosting;
using MudBlazor.Services;

var builder = WebAssemblyHostBuilder.CreateDefault(args);

builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");

builder.Services.AddAuthorizationCore();
builder.Services.AddBlazoredLocalStorage();
builder.Services.AddMudServices();

builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress) });

builder.Services.AddScoped<AuthenticationStateProvider, ApiAuthenticationStateProvider>();
builder.Services.AddScoped<IAuthService, AuthService>();
builder.Services.AddScoped<ISkinService, SkinService>();
builder.Services.AddScoped<IConfigurationService, ConfigurationService>();

await builder.Build().RunAsync();
```

Рис. 5. Файл Program.cs

3. Преимущества и недостатки Blazor WebAssembly

Blazor со своими подсистемами является полноценным и удобным для .NET разработчика фреймворком. В распоряжении программиста находятся платформа .NET с множеством уже реализованных Microsoft алгоритмов, а при необходимости и система пакетов NuGet. Так как язык программирования C# является строго типизированным, большинство ошибок выявляются еще на этапе компиляции. Безопасность приложения зависит от используемых библиотек и алгоритмов шифрования. Несмотря на то, что обработка первого

запроса занимает какое-то время, все последующие действия и запросы выполняются без каких-либо существенных задержек.

Помимо преимуществ Blazor WebAssembly, у этой технологии есть и существенные недостатки:

– Возможности приложения ограничены ресурсами браузера, вследствие того, что все файлы загружаются клиенту, увеличивается время выполнения первого запроса и без должного внимания к безопасности возникает риск утечки данных, что требует от разработчика большего внимания и времени к разработке системы;

– Старые браузеры не поддерживают технологию WebAssembly и соответственно не могут запускать приложения Blazor WebAssembly. Для решения этой проблемы существует подсистема Blazor Server.

Заключение

Несмотря на то, что Angular, React.js и Vue.js до сих пор остаются наиболее востребованными решениями при разработке веб-приложений, компания Microsoft, желая изменить этот порядок, создала мощный фреймворк, который уже способен составить серьезную конкуренцию популярным аналогам. Так как данный фреймворк появился сравнительно недавно, многие элементы системы еще требуют доработки, но благодаря регулярным обновлениям и обширному сообществу .NET разработчиков Blazor и его подсистемы быстро развиваются, а сообщество постоянно расширяется, что существенно повышает рейтинг технологии и может стать новым популярным выбором у разработчиков при решении широкого спектра задач.

Литература

1. Резников К. Г. Разработка программного обеспечения для визуализации трехмерных поверхностей в веб-браузере / К. Г. Резников, С. Н. Медведев // Вестник ВГТУ. Том 17, №6. – Воронеж: ВГТУ, 2021 - С. 13-19.

2. Шабан Д. WebAssembly как успешный пример реализации концепции кроссплатформенности / Д. Шабан // Научно-технические инновации и веб-технологии. – 2022. – № 1. – С. 32–38.

3. Чиннатамби К. Изучаем React / К. Чиннатамби. – 2-е изд., перераб. и доп. – Санкт-Петербург : БОМБОРА, 2022. – 368 с.

4. Стефанов С. React. Быстрый старт / С. Стефанов. – Санкт-Петербург : Питер, 2023. – 304 с.

5. Хэнчетт Э. Vue.js в действии / Э. Хэнчетт, Б. Листуон. – Санкт-Петербург : Питер, 2020. – 304 с.

6. What Is Blazor WASM?. – Режим доступа: https://blazorise.com/blog/what-is-blazor-wasm – (Дата обращения 24.02.2024).

7. Blazor University. – Режим доступа: https://blazor-university.com/overview/ – (Дата обращения 09.03.2024).

УДК: 004.93

Использование различных видов нейросетей для распознавания звуков кашля

А. В. Воротынцев

Воронежский государственный университет

Введение

При терапии пациентов с заболеваниями дыхательной системы врачи сталкиваются с трудностями в получении достоверной информации об их самочувствии. Это связано с тем, что используемые методы имеют ограничения по частоте применения (например, рентгенография и компьютерная томография). Разговор с пациентом помогает выявить лишь субъективные данные о его состоянии, включая частоту и характер кашлевых приступов. Важно постоянное наблюдение за состоянием пациента. Это повысит эффективность лечения и уменьшит вероятность развития осложнений. В некоторых случаях необходимо применять дистанционный подход к лечению. Для этого необходимо разработать систему, которая позволяет осуществлять эффективный процесс лечения без необходимости посещать медицинские учреждения и поддерживать связь с врачом на постоянной основе.

Нейросеть может стать одним из элементов такой системы. Она будет решать задачу выделения кашлей из звуковых данных.

В настоящее время проводятся исследования возможностей нейросетей для распознавания звуков кашля. Группа ученых из компании Google разработала нейросеть, которая определяет заболевание человека по кашлю и дыханию. Несмотря на то, что клинические испытания еще не были завершены, разработчики уверены в огромном потенциале использования нейросетей в решении подобных задач [1].

Команда ученых из MIT обнаружила, что искусственный интеллект может уловить разницу между кашлем больных людей и здоровых. Для создания моделей использовалась в том числе рекуррентная нейронная сеть [2].

Группа ученых, которые занимаются вопросом применения робототехники и искусственного интеллекта при пандемии COVID-19 применяла сверточную нейронную сеть для распознавания кашля. Такой сети требовалось создание спектрограммы в качестве предварительной обработки звука [3].

Российские ученые создали проект “Acoustery”. Это система мониторинга здоровья на основе анализа аудиозаписи звуковая кашля. В ней применяется технологии машинного обучения. Принцип работы системы основан на представлении звука в виде спектрограммы [4].

1. Структура системы

Процесс определения звуков кашля включает три этапа. На первом этапе система получает звуковые данные. Эти данные представлены в несжатом виде в формате wav.

На втором этапе выполняется предварительная обработка этих данных, для того чтобы они стали пригодны для распознавания. Выделяются временные характеристики звука. Также на основе звуковых данных может быть построена спектрограмма. Она представляет зависимость спектральной плотности мощности сигнала от времени.

На втором этапе происходит предварительная обработка данных для того, чтобы они стали пригодны для распознавания нейросетью. На основе звуковых данных может быть построена спектрограмма, которая представляет зависимость спектральной плотности мощности сигнала от времени. Также выделяются временные характеристики звука.

На третьем этапе происходит процесс распознавания кашля. На основе подготовленных звуковых данных формируется входной вектор. Он подается на вход нейросетевой модели, которая выполняет задачу классификации. Модель определяет, к какому классу относится текущий звуковой фрагмент: кашель или шум.

2. Реализация системы

2.1. Постановка задачи

Для реализации нейросетей, способных распознавать звуки кашля, используется высокоуровневый прикладной интерфейс Keras [5], предназначенный для работы с платформой Tensorflow [6]. Для написания программы используется язык программирования Python.

Для достижения цели применяются различные архитектуры искусственных нейронных сетей, такие как сверточная и рекуррентная сети.

В качестве функции потерь используется бинарная кросс энтропия, которая также известна как логарифмическая потеря. Она широко используется в машинном обучении для задач бинарной классификации.

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_{true}\log(y_{pred}) + (1 + y_{true})\log(1 - y_{pred})]$$

где N — количество образцов в наборе данных,

y_{true} — это действительная метка для каждого образца. В поставленной задаче бинарной классификации это будет либо 0, либо 1.

y_{pred} — это предсказанная вероятность того, что образец принадлежит положительному классу (классу 1). Это значение, которое выдает модель на выходе.

Для оптимизации модели используется алгоритм Adam (Adaptive Moment Estimation). Коэффициент скорости обучения в нем установлен на значение 0,001.

Обучение сети происходит в течение 10 эпох.

Для оценки работы нейросети используется точность (accuracy).

Так как эти нейронные используют разные типы данных на вход, то необходимы предварительные преобразования.

2.2. Подготовка данных

Для обучения моделей используется выборка из 800 звуковых фрагментов. В нее входят звуки кашля и сторонние шумы. Для оцифровки звуковых сигналов использовался метод PCM (Pulse Code Modulation). Записи сделаны в формате wav с частотой дискретизации 44 кГц.

Средняя продолжительность кашля принята равной половине секунды. Таким образом, из используемой выборки были исключены звуковые фрагменты с меньшей продолжительностью, а фрагменты с большей продолжительностью были разделены на фрагменты меньшего размера.

Для сверточной нейронной сети были подготовлены спектрограммы. Для их создания использовался пакет librosa [6], написанный на языке Python. Спектрограммы представляют собой изображения, размер которых составляет 128 x 384 пикселей.

На рис. 1 представлена спектрограмма звука кашля

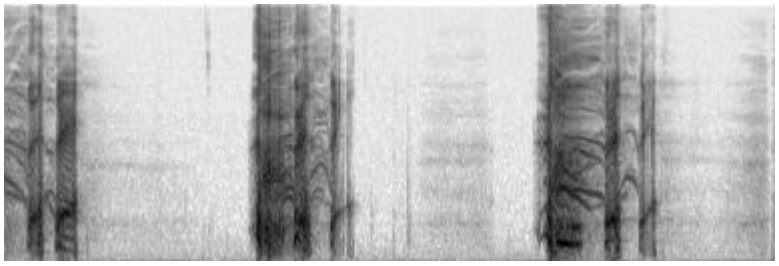


Рис. 1. Спектрограмма звука кашля

На рис 2 представлена спектрограмма стороннего шума.

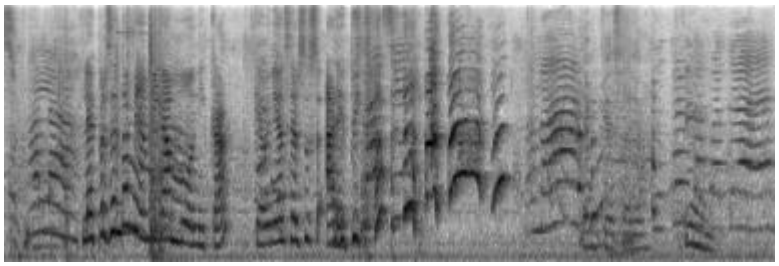


Рис. 2. Спектрограмма стороннего *шума*

Чтобы преобразовать изображения пригодный для распознавания нейросетью использовался вспомогательный класс ImageDataGenerator из пакета preprocessing.image. Этот пакет включен в пакет keras.

Для рекуррентной нейронной сети используется временная характеристика звука - изменение амплитуды во времени. Создаются датафреймы для обучающих и тестовых данных, в которых каждому файлу присваивается метка класса (1 для наличия кашля, 0 для прочих шумов). Для того, чтобы извлечь временные характеристики звука используется пакет librosa.

Затем из фрагментов аудиозаписей формируются массивы, каждый из которых содержит 22000 вещественных чисел.

Данные делятся в пропорции 3 к 4, на две составляющие: тренировочную и проверочную выборки.

Визуализация звуковых данных кашля представлена на рис. 3

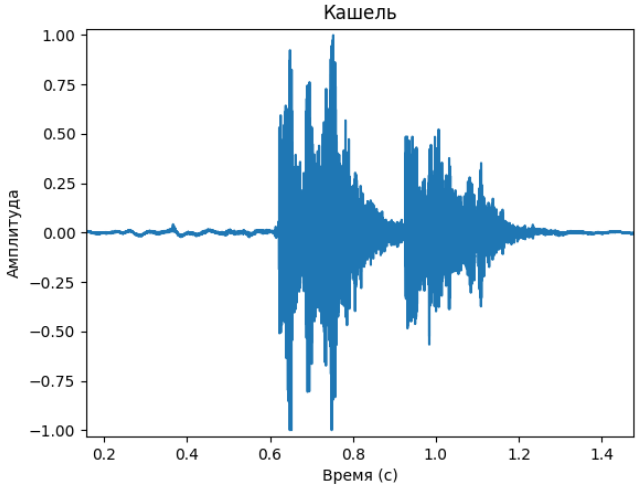


Рис. 3. Визуализация звуковых данных кашля

Визуализация звуковых данных шума представлена на рис. 4

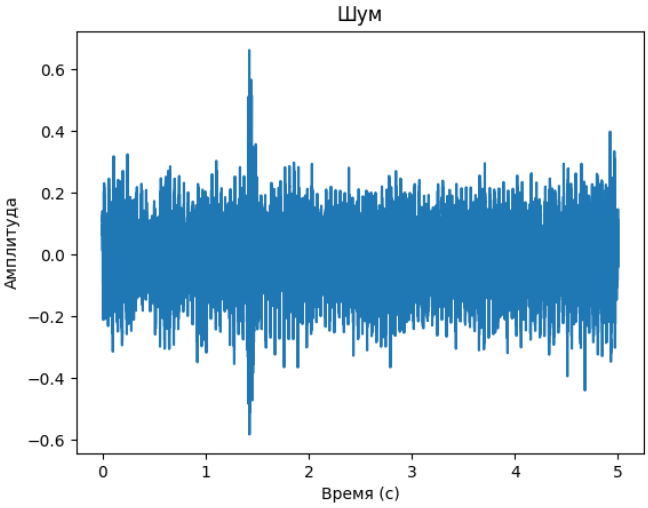


Рис. 4. Визуализация звуковых данных шума

2.3. Сверточная нейронная сеть

Для работы с изображением спектрограммы звука была построена сверточная нейронная сеть (CNN).

Используется модель Sequential, которая представляет собой последовательный стек слоев. В ней каждый слой добавляется один за другим.

В сеть ключены 3 слоя двумерной свертки (Conv2D). Первый из них содержит 32 фильтра, второй 64 фильтра, третий 128 фильтра. Используется размер окна свертки 2x2. В качестве функции активации используется выпрямитель (ReLU).

После каждого сверточного слоя располагается слой пулинга. Он необходим для выполнения операции подвыборки. Его размер пула составляет 2x2. В нем применяется размер

шага, равный 2. Применяется алгоритм максимального пулинга. Эти слои позволяют сократить количество вычислений в модели и снижают риск переобучения.

Для того чтобы преобразовать данные из двумерной формы в одномерную используется слой сглаживания Flatten.

За ним располагается полносвязный слой с 512 нейронами и функцией активации ReLU.

Последний слой содержит один нейрон и использует сигмоидальную функцию активации. Она используется для прогнозирования вероятности принадлежности к классу. Это необходимо для решения задачи бинарной классификации.

Схема нейронной сети представлена на рис. 5.

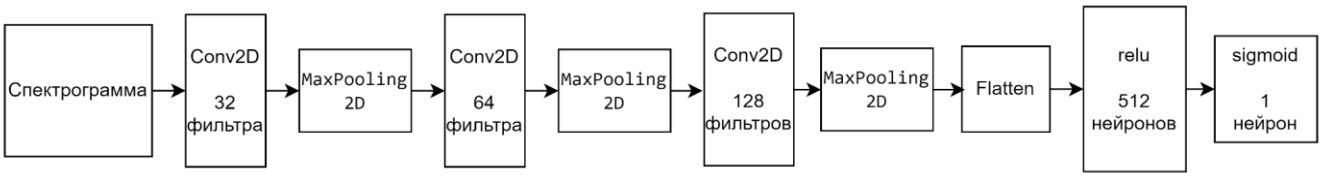


Рис. 5. Сверточная нейронная сеть

2.4. Рекуррентная нейронная сеть

Для работы с временными характеристиками звука была построена рекуррентная нейронная сеть (RNN). В ней связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени, например, изменение амплитуд звука.

Для создания сети также используется последовательная модель Sequential.

Затем используется слой долгой краткосрочной памяти LSTM (Long short-term memory). Размерность слоя составляет 128 единиц. Он способен запоминать информацию из прошлых шагов в долгосрочной перспективе. В качестве функции активации данного слоя выступает сигмоид.

Для решения проблемы переобучения сети был использован слой исключения (Dropout). Он случайным образом обнуляет выходы нейронов во время обучения.

Последним слоем модели является полносвязный слой с одним нейроном и функцией активации sigmoid. Он позволяет получить вероятности принадлежности к каждому из двух классов.

Схема нейронной сети представлена на рис. 6.

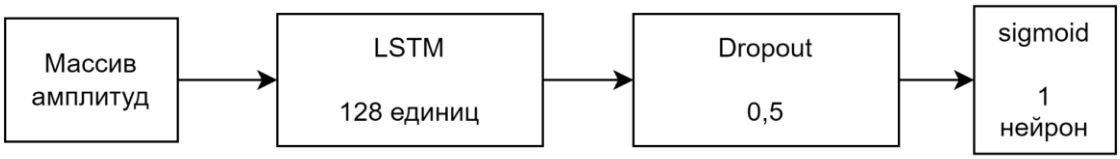


Рис. 6. Рекуррентная нейронная сеть

Заключение

В ходе исследования применимости данных сетей для решения задачи определения звуков кашля были получены следующие результаты (табл.1).

Таблица 1

	CNN	RNN
Точность обучения	0.738	0.6
Точность валидации	0.675	0.51

Таким образом можно сделать вывод о том, что наибольшую эффективность показа сверточная нейронная сеть. Она использует спектрограммы звука. Эта модель, построена на основе архитектуры Sequential с использованием трех слоев свертки (Conv2D) и пулинга (MaxPooling2D). Применение функции активации ReLU в сверточных слоях позволило извлечь релевантные признаки из спектрограмм, а использование сигмоидальной функции активации на выходном слое обеспечило вероятностную интерпретацию результатов. Она показала точность 0.738 для тестовой выборки и 0.675 при валидации.

Рекуррентная нейронная сеть не показала достаточной эффективности. Это можно объяснить тем, что звуковые фрагменты кашля слишком малы для того чтобы сеть выделила полезную информацию и закономерности для распознавания кашля. В данной задаче не играют роль длительные зависимости между элементами.

Следует также отметить, что точность валидации не превышает 80 % и требует повышения качества распознавания за счет варьирования предварительной обработки данных и параметров используемых нейронных сетей.

Литература

1. Google AI could soon use a person’s cough to diagnose disease. Режим доступа: https://www.nature.com/articles/d41586-024-00869-0. – (Дата обращения: 24.03.2024).

2. Artificial Intelligence for Detecting COVID-19 with the Aid of Human Cough, Breathing and Speech Signals: Scoping Review. Режим доступа: https://www.embs.org/ojemb/articles/artificial-intelligence-for-detecting-covid-19-with-the-aid-of-human-cough-breathing-and-speech-signals-scoping-review/ – (Дата обращения: 26.03.2024).

3. Cough Recognition Based on Mel-Spectrogram and Convolutional Neural Network. Режим доступа: https://www.frontiersin.org/articles/10.3389/frobt.2021.580080/full – (Дата обращения: 28.03.2024).

4. Acoustery: официальный сайт - Режим доступа: https://www.acoustery.tech/ (дата обращения: 30.03.2024)

5. Keras : официальный сайт – Режим доступа: https://keras.io/ (дата обращения: 30.10.2023).

6. TensorFlow : официальный сайт – Режим доступа: https://www.tensorflow.org/?hl=ru (дата обращения: 30.10.2023).

7. Librosa : официальный сайт – Режим доступа: https://librosa.org/ (дата обращения: 30.10.2023).

УДК 3054

Исследование индекса пространственной индексации R-дерева и его разновидности

И. М. Гаврилов, О. В. Авсеева

Воронежский государственный университет

Введение

В мире современных информационных технологий, где данные растут экспоненциально, эффективное управление информацией становится первостепенной задачей. Пространственное индексирование данных играет ключевую роль в обеспечении систематизации и доступа к пространственным данным, повышая производительность и удобство использования. Среди различных методов пространственной индексации R-деревья широко известны своей эффективностью при обработке многомерных данных.

R-дерево — это тип сбалансированного дерева, впервые представленном Гутманом в 1984 году как динамическая индексная структура для пространственного поиска, предназначенная для работы с геометрическими данными, такими как точки, отрезки прямых, поверхности, объемы и гиперобъемы в высокоразмерных пространствах. R-деревья были широко адаптированы и оптимизированы посредством различных модификаций, что привело к появлению нескольких вариантов, таких как R+-деревья и R*-деревья. Каждый вариант предлагает различные характеристики и оптимизации, направленные на повышение производительности запросов, минимизацию перекрытия между узлами и улучшение использования пространства.

Целью данной работы является исследование алгоритмов R-дерева для пространственной индексации с упором на различные стратегии и варианты разделения. В частности, в исследовании будут анализироваться и сравниваться алгоритмы линейного и квадратичного разделения R-дерева, а также рассматриваться особенности и эффективность R+ и R* деревьев.

1. R-дерево

R-деревья – это иерархические структуры данных, основанные на B+-деревьях. Они используются для динамической организации множества d-мерных геометрических объектов, представляя их минимальными ограничивающими d-мирными прямоугольниками (для простоты, в дальнейшем MBR). Каждый узел R-дерева соответствует MBR, ограничивающий его потомков и может содержать ограниченное количество записей [1].

MBR, окружающие разные узлы, могут не только перекрывать друг друга, но и быть включены (в геометрическом смысле) в другие узлы, но ассоциироваться только с одним из них. По этой причине пространственный поиск может посетить множество узлов, прежде чем подтвердить существование данного MBR. Кроме того, представление геометрических объектов через их MBR может привести к ложным срабатываниям. Для того чтобы устранить ложные срабатывания, необходимо исследовать объекты-кандидаты. Например, когда два многоугольника не пересекаются друг с другом, но их MBR пересекаются. Поэтому R-дерево играет роль механизма фильтрации, позволяющего сократить дорогостоящее прямое исследование геометрических объектов.

Из определения R-дерева следует, что оно является сбалансированным по высоте

деревом. Оно представляет собой обобщение структуры B+-дерева для многих измерений. R-деревья являются динамическими структурами данных, т. е. для обработки вставок и удалений не требуется глобальная реорганизация структуры.

Вставки в R-дерево обрабатываются так же, как и вставки в B+-дереве. R-дерево обходится с корня, чтобы найти подходящий лист для размещения новой записи. Запись вставляется в найденный лист, а затем все узлы на пути от корня до этого листа обновляются соответствующим образом. Если подходящий лист не может вместить новую запись из-за того, что он переполнен (в нем уже содержится максимальное количество записей), то он разделяется на два узла. Разбиение в R-деревьях отличается от разбиения в B+-деревьях, поскольку учитывает другие критерии. Задача алгоритма разбиения состоит в том, чтобы минимизировать вероятность обращения к новым созданным узлам для одного и того же запроса.

1.1. Алгоритм линейного разделения

Алгоритм линейного разделения — один из самых простых и ранних методов, используемых для разделения узлов в R-деревьях, предложенный Гуттманом. Когда узел превышает свою емкость, этот алгоритм выбирает две записи в качестве первых элементов двух новых узлов. Процесс выбора этих первых записей имеет решающее значение и основан на выборе пары прямоугольников, которые находятся как можно дальше друг от друга, тем самым обеспечивая минимизацию перекрытия между группами, которые формируются вокруг каждого прямоугольника.

После того, как начальные объекты выбраны, оставшиеся записи в случайном порядке добавляются к одной из двух групп в зависимости от того, расширение какой группы потребует наименьшего увеличения для размещения новой записи.

Линейный характер этого алгоритма делает его вычислительно эффективным при низкой сложности O(n), что делает его подходящим для случаев, где необходима быстрая обработка. Однако этот метод может привести к плохо сбалансированным деревьям, поскольку он не равномерно распределяет записи, что может привести к большей глубине дерева и увеличению времени поиска. На рис. 1 показан пример работы R-дерева с линейным алгоритмом разделения узла.

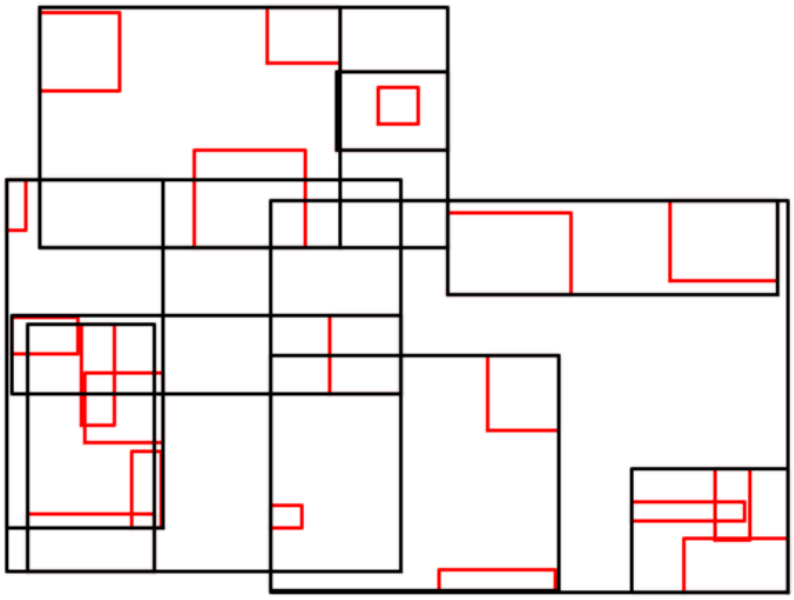


Рис. 1. Линейное разделения узла

1.2. Алгоритм квадратичного разделения

Квадратичное разделения, также предложенное Гуттманом, фокусируется на оптимизации пространства, занимаемого записями, за счет более высоких вычислительных затрат. Как и линейное разделение, оно начинается с выбора двух начальных записей, которые, если их сложить вместе, создадут как можно больше мертвого пространства (мертвое — это пространство, которое остается в MBR, если игнорировать области двух объектов), таким образом пытаясь уменьшить перекрытие пространства и охватить как можно большую площадь [1].

Процесс распределения остальных записей заключается в выборе записи, у которой разница мертвого пространства, если его отнести к каждой из двух групп, максимальна и добавления этой записи в ту группу, которая требует меньшего расширения MBR. Так повторяется до тех пор, пока все записи не будут распределены по группам.

Этот шаг имеет квадратичную сложность $O(n^2)$, поскольку на нем оценивается комбинация каждой записи с каждой группой.

Алгоритм квадратичного разделения обычно создает более сбалансированные деревья и лучшее использование пространства, чем линейное разделение. Это может привести к повышению производительности запросов, особенно в средах, где операции чтения выполняются гораздо чаще, чем операции записи. Однако повышенные вычислительные затраты во время вставки могут оказаться недостатком в случаях с большим количеством вставок записи. Пример работы R-дерева с квадратичным разделением показан на рис. 2.

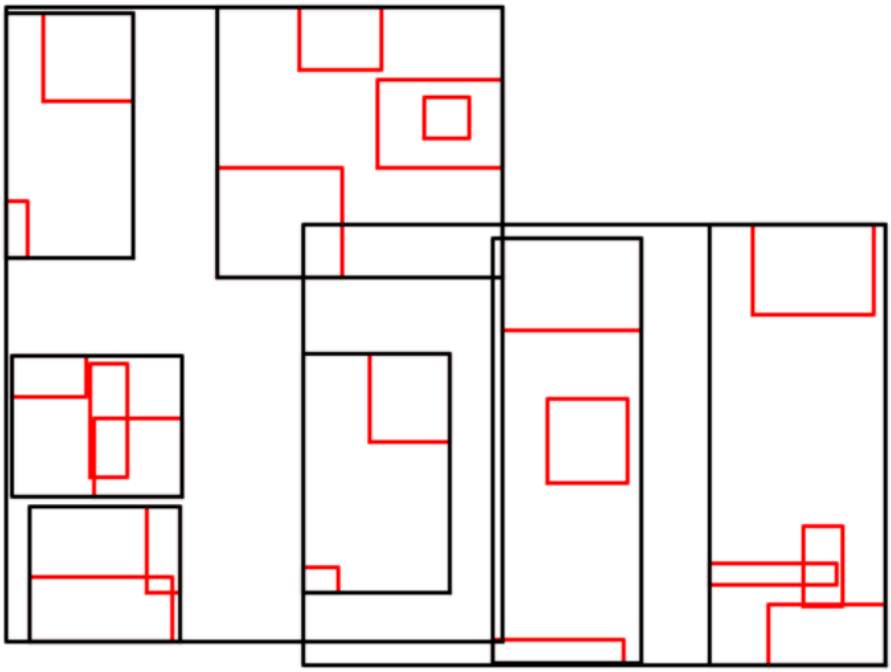


Рис. 2. Квадратичное разделения узла

2. R+-дерево

R+-дерево является расширением, которое позволяет избежать посещения нескольких путей при запросе местоположения точки и, таким образом, улучшить производительность поиска [2].

R+-деревья не допускают перекрытия MBR на одном и том же уровне дерева, что является значительным структурным улучшением по сравнению с традиционными R-деревьями. Для достижения этой цели вставленные объекты, если они охватывают MBR нескольких узлов, должны быть разделены на два или более MBR, что в свою очередь приводит к дублированию записей и избыточному их хранению в нескольких узлах.

Алгоритм обработки запроса поиска аналогичен тому, что используется для R-деревьев, единственное отличие – устранение дубликатов в результате поиска, чтобы не сообщать об объекте более одного раза. Однако алгоритмы вставки, удаления и разбиения узлов отличаются из-за применяемой техники обрезания.

Чтобы вставить новую запись, алгоритм вставки начинается с корня и определяет узлы, которые пересекаются с записью. Затем вставляемая запись обрезается, и процедура рекурсивно применяется к соответствующим поддеревьям.

Во время выполнения алгоритма вставки узел может стать переполненным, и тогда в нем больше нельзя будет хранить записи. Чтобы справиться с этой ситуацией, необходим механизм разбиения узлов, как и в случае R-дерева. Основное отличие алгоритма разбиения R+-дерева от алгоритма R-дерева заключается в том, что в дополнение к восходящему распространению может потребоваться распространение вниз [3]. Пример работы R+-дерева показан на рис. 3.

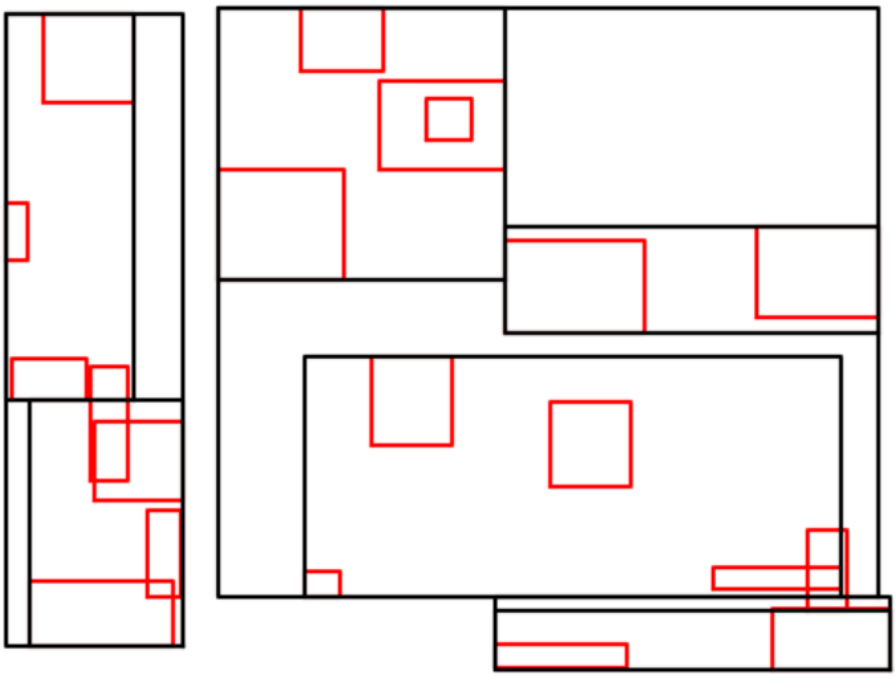


Рис. 3. R+-дерево

3. R*-дерево

R*-деревья построены на основе R-деревьев, но включают в себя несколько оптимизаций, направленных на улучшение баланса дерева и минимизацию площади и перекрытия ограничивающих прямоугольников на всех уровнях дерева. В структуре

используется сложный механизм повторной вставки и улучшенные стратегии разделения, которые имеют решающее значение для балансировки дерева во время обновлений [4, 5].

В R*-деревьях используется двухэтапный подход к обработке вставок. Если выбирается лист, который не может вместить запись (т. е. в нем уже есть максимальное количество записей), то R∗-дерево не прибегает сразу к разбиению узла. Вместо этого оно находит часть записей из переполненного узла и вставляет их обратно.

Алгоритм повторной вставки позволяет добиться своеобразной ребалансировки дерева и значительно повысить производительность при обработке запросов. Однако повторная вставка является дорогостоящей операцией. Поэтому для каждого уровня дерева допускается только одно ее применение. Если переполнение не может быть устранено повторной вставкой, выполняется разбиение узлов.

Алгоритм разбиения включает два этапа. На первом этапе определяется ось разбиения среди всех измерений. Осью разбиения является та, которая имеет наименьший общий периметр. Выполняется сортировка всех записей по координатам их левых границ. Затем рассматривается каждое разбиение отсортированного списка. Второй проход повторяет этот процесс в отношении к правым границам MBR. Наконец, общий периметр по оси равен сумме всех периметров, полученных в результате двух проходов по этой оси. Когда выбрана ось разбиения, алгоритм сортирует записи (в соответствии с их нижней или верхней границей) по выбранному измерению и снова рассматривает все возможные разбиения. Окончательное деление — это то, которое имеет минимальное перекрытие между MBR результирующих узлов.

Пример работы R*-дерева показан на рис. 4.

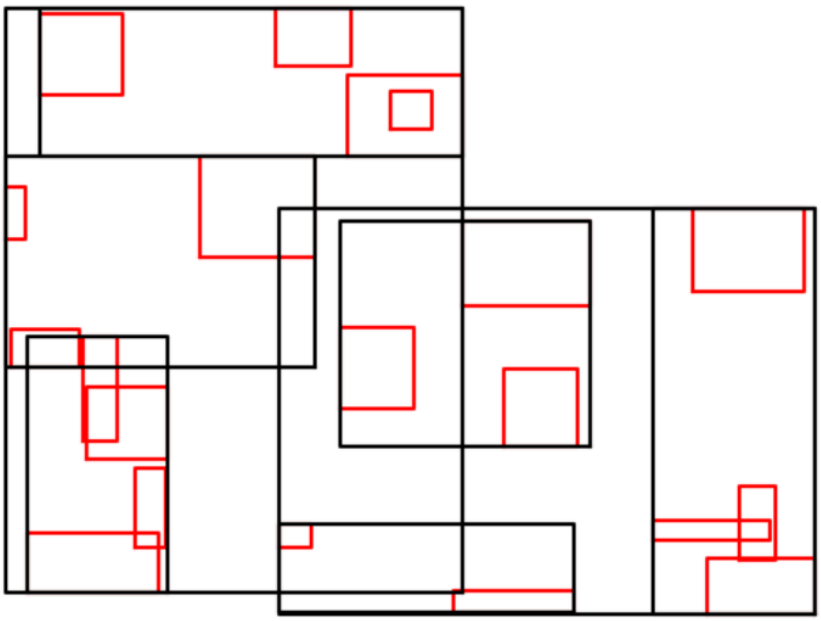


Рис. 4. R-дерево*

4. Сравнение эффективности

Эффективность использования конкретной разновидности R-дерева будем оценивать по двум метрикам: построения индекса и поиск по диапазону в зависимости от количества входных данных.

Из графиков (рис. 5, 6) можно сделать вывод, что рассматриваемые разновидности показывают приблизительно одинаковые характеристики на случайно сгенерированных данных, кроме R*-дерева, которое значительно уступает остальным по построению индекса, но с другой стороны имеет гораздо более быстрый диапазонный поиск.

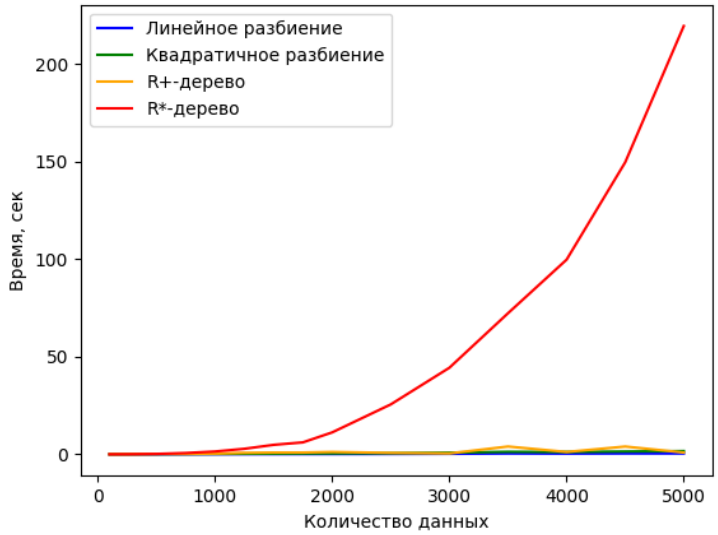


Рис. 5. Построение дерева

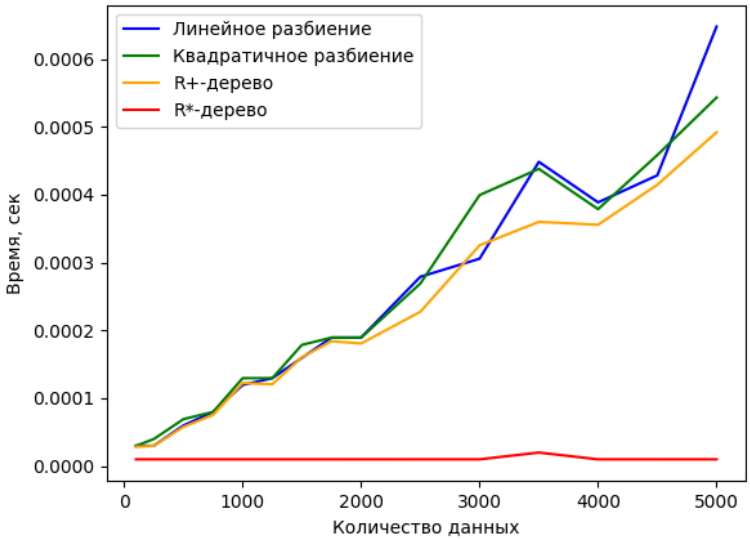


Рис. 6. Поиск диапазона

Заключение

В работе мы рассмотрены различные алгоритмы и структуры, связанные с R-деревьями. Особое внимание уделено механизмам разделения узлов при переполнении и отличительным характеристикам вариантов, а именно R+-деревьев и R*-деревьев. Каждая из этих структур предлагает уникальные преимущества и решает конкретные проблемы, связанные с управлением и запросом пространственных данных.

Литература

1. R-tree. – Режим доступа: https://en.wikipedia.org/wiki/R-tree. – (Дата обращения: 15.04.2024).
2. R+-tree. – Режим доступа: https://en.wikipedia.org/wiki/R%2B_tree. – (Дата обращения: 16.04.2024).
3. Sellis, T. The R + -tree: A Dynamic Index For Multi-dimensional Objects / T. Sellis, N. Roussopoulos, C. Faloutsos // Proceedings of the 13th VLDB Conference. – 1987 – С. 507–517.
4. R*-tree. – Режим доступа: https://en.wikipedia.org/wiki/R*-tree. – (Дата обращения: 17.04.2024).
5. R*-tree в Go. – Режим доступа: https://habr.com/ru/articles/666904/. – (Дата обращения: 17.04.2024).

УДК 519.82

МАШИННОЕ ОБУЧЕНИЕ В ПРОГНОЗИРОВАНИИ

М. В. Глуховский, Л. А. Смирнова

Воронежский государственный университет

Введение

В настоящее время человечеству приходится работать с большим объемом данных, которые становится всё сложнее и затратнее по времени анализировать вручную. На помощь приходит интеллектуальный анализ данных, обладающий огромным количеством средств и методов, способных обработать данные и получить результат за считанные минуты.

Целью данной работы является изучение методов машинного обучения, с помощью которых решается задача прогнозирования стоимости квартиры на основе данных о квартирах России.

1. Основные понятия машинного обучения, прогнозирования и корреляции

Машинное обучение (МО) — это одна из форм искусственного интеллекта [2]. При использовании МО выявляются закономерности в данных с помощью специальных алгоритмов. И далее на основе этих закономерностей создается модель данных для прогнозирования. Со временем и с количеством обработанных результатов прогнозы становятся точнее. Благодаря адаптивному характеру МО, оно отлично подходит для сценариев, в которых данные постоянно изменяются, свойства запросов или задач нестабильны или написать код для решения фактически невозможно.

Рассмотрим процесс МО, который используется при решении задач:

1) сбор и подготовка данных: определяется тип данных и выбирается алгоритм МО, при проверке находятся ошибки и устраняются проблемы целостности данных;
2) обучение модели: подготовленные данные делятся на две группы — набор для обучения и набор для проверки;
3) проверка модели: проводится оценка модели с помощью набора для проверки;
4) интерпретация результатов: результаты изучаются для получения аналитических сведений, формирования выводов и прогноза.

Цель прогнозирования — предсказание будущих событий. Решение задачи прогнозирования требует некоторой обучающей выборки данных: тщательного исследования исходного набора данных и методов, подходящих для их анализа. Задача прогнозирования — одна из наиболее сложных задач Data Mining, основанного на статистическом анализе, МО и математике [3].

Взаимосвязь между разными показателями в статистике называется ***корреляцией***, она используется, чтобы оценить зависимость переменных друг от друга.

Коэффициентами корреляции называют показатели, которые выражают силу корреляции между переменными [7]. Выбор коэффициента корреляции зависит от ситуации, т.к. каждый из них лучше подходит для конкретного случая.

Среди распространенных коэффициентов корреляции выделяют:

– *коэффициент Пирсона* показывает прямолинейную связь между переменными. Данный коэффициент принимает значения в промежутке [-1,1]. Близость значения к 1 означает сильную положительную связь между показателями, а близость к -1 — сильную

отрицательную. Близкое к 0 значение, включая сам 0, говорит, что корреляции нет. Коэффициент Пирсона для переменных x и y вычисляется по формуле:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 (y_i - \bar{y})^2}},$$

где x_i — значения, принимаемые переменной x, y_i — значения, принимаемые переменной y, $\bar{x}$ — среднее значение x, $\bar{y}$ — среднее значение y;

– *коэффициент Кендалла* показывает корреляцию между факторами, которые можно ранжировать по какому-то признаку. Вместо значений показателя используются номера, присвоенные значениям при ранжировании. Значение коэффициента тоже [-1,1], подходит только для оценки линейной связи. Коэффициент Кендалла для переменных x и y вычисляется по формуле:

$$\tau_{xy} = \frac{2S}{n(n-1)},$$
$$S = P - Q,$$

где n — число парных наблюдений, P — суммарное число наблюдений, следующих за текущими наблюдениями с большим значением рангов y, Q — суммарное число наблюдений, следующих за текущими наблюдениями с меньшим значением рангов y;

– *коэффициент Спирмена* так же предназначен для оценки ранжированных показателей. Однако он больше подходит для малых выборок. Коэффициент Спирмена для переменных x и y вычисляется по формуле:

$$\rho_{xy} = \frac{6\sum_{i=1}^{n} d^2}{n(n^2 - 1)},$$

где n — число парных наблюдений, $\sum_{i=1}^{n} d^2$ — сумма квадратов разностей рангов.

Коэффициенты корреляции используются в случае линейной корреляции, когда график одного показателя как бы «повторяет» другой. Еще есть нелинейная корреляция: одна переменная изменяется равномерно, а другая неравномерно, но взаимосвязь при этом есть. Для оценки нелинейной корреляции (при изменении одной величины характер изменения другой величины не линеен, а описывается другими законами) не пользуются коэффициентами, а используют корреляционное отношение, как более общий показатель.

Рассчитать корреляцию для каких-то факторов можно и вручную, но обычно пользуются вспомогательными инструментами:

– онлайн-сервисы;
– Excel;
– языки программирования и др.

2. Основные алгоритмы МО

Алгоритм МО — это процесс, который использует данные и создает модели МО, готовые к производству. Рассмотрим основные алгоритмы МО, сферы их применения, а также достоинства и недостатки (табл. 1) [1]:

Таблица 1

Основные алгоритмы МО

Название алгоритма	Описание / сфера применения	Достоинства	Недостатки
Линейная регрессия	Данный алгоритм предсказывает простые линейные зависимости в данных и значения (рис. 1). Он применяется при работе с переменными, линейно зависящими от одной или нескольких других переменных.	Интуитивно понятная модель машинного обучения, простая в реализации и быстрая в работе.	Хорошо работает только на очень простых зависимостях, но плохо предсказывает сложные зависимости.
Логистическая регрессия	Данный алгоритм позволяет разделять несложные объекты на два класса, чье состояние управляется и описывается небольшим числом параметров (рис. 2). Логистическая регрессия выдает ответ в виде числа в промежутке [0,1]. Если число ниже определенного порога значения, то объект относится к первому классу объектов, а если выше — то ко второму. Калибровка порогового значения для разделения объектов на классы подбирается в ходе калибровки алгоритма. Алгоритм применяется, как правило, там, где требуется несложная классификация небольшого числа объектов на небольшое число классов.	Быстрота и наглядность.	Может классифицировать только относительно простые объекты, но не подходит для разделения объектов на несколько классов, т.к. ответ этого алгоритма — по сути, двоичный сигнал типа «да-нет».
Деревья принятия решений	Алгоритм представляет состояния объекта в виде дерева (рис. 3). Данный алгоритм используется для классификации и предсказания состояний на основе имеющихся данных.	Надежность, простота в применении, интуитивная понятность.	Неэффективность в задачах регресии.
Алгоритм случайного леса	Из простых деревьев строится группа, которая называется *лес*, где каждое дерево немного отличается от остальных (рис. 4). Деревья в лесу «голосуют» за определенные варианты решений, и самое часто встречающееся решение становится ответом системы [5]. Случайный лес используется там же, где и деревья, то есть для предсказаний на основе параметров и для классификации.	Объединение простых алгоритмов в группы часто дает хорошие результаты.	Как и у классических решающих деревьев, сфера применения достаточно ограничена.
Бустинг	Суть этого метода в том, что сильный классификатор создается на основе слабых: каждая новая модель учится на ошибках предыдущей (рис. 5), то есть каждый раз добавляются все новые и новые модели, которые пытаются исправить ошибки своих предшественников. Бустинг подходит практически для любых задач.	Точность результатов.	Модели могут быть весьма большими — в бустинг нужно включать наборы из других моделей, что усложняет построение итоговой системы.

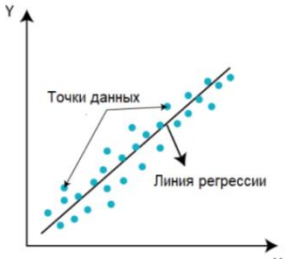


Рис. 1. Линейная регрессия

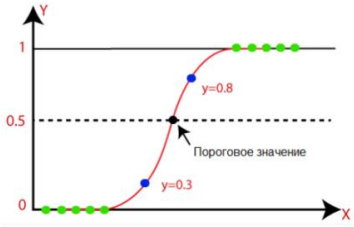


Рис. 2. Логистическая регрессия

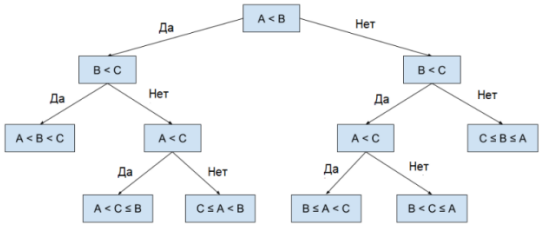


Рис. 3. Дерево принятия решений

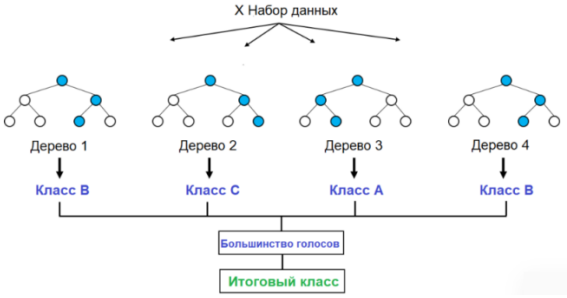


Рис. 4. Случайный лес

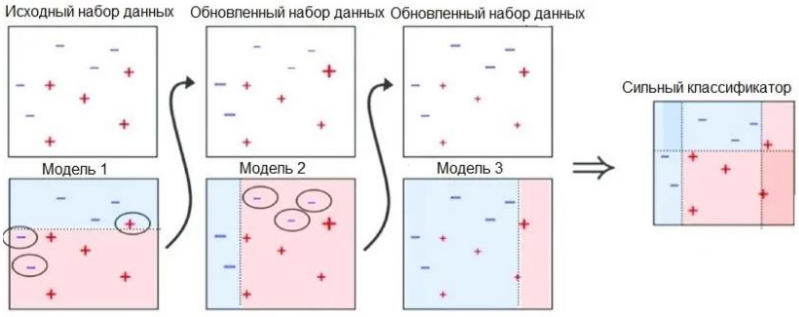


Рис. 5. Бустинг

3. Задача прогнозирования стоимости квартиры

Рассмотрим задачу:

По данным о характеристиках квартир России и их ценах необходимо предсказать стоимость случайной квартиры с известными характеристиками при помощи языка программирования Python.

Решение:

Подготовка данных:

Данные взяты с сайта Kaggle (https://www.kaggle.com/datasets/mrdaniilak/russia-real-estate-2021). Имеется таблица, содержащая информацию о более чем 1 млрд. квартир России. В ней представлены основные характеристики квартир, такие как: общая площадь, местонахождение, количество комнат и, самое главное, стоимость квартиры. В задаче будет рассмотрено 13 характеристик. Часть исходных данных представлена на рис. 6:

price	level	levels	rooms	area	kitchen_area	geo_lat	geo_lon	building_type	object_type	postal_code	street_id	house_id
5800000	8	17	2	81.7	23.0	51.644787	39.168873	3	0	394055.0	NaN	NaN
2500000	5	9	1	32.0	6.0	51.654484	39.121593	0	0	394062.0	NaN	NaN
5500000	5	9	4	80.0	9.0	51.716743	39.178584	2	0	394077.0	125005.0	933491.0
3150000	1	10	3	69.0	9.0	51.592697	39.246193	2	0	394090.0	205259.0	2662104.0
4058176	9	13	2	63.6	14.0	51.659333	39.196923	2	2	394000.0	NaN	980206.0
...	...	...	...	...	...	...	...	...	...	...	...	...
3900000	12	13	2	45.0	8.0	51.693400	39.255200	0	0	394063.0	NaN	NaN
4170000	22	25	2	72.0	13.0	51.706123	39.271228	0	2	394063.0	204576.0	2819354.0
7250000	2	8	2	72.1	15.5	51.726095	39.206203	0	2	394043.0	406607.0	1183847.0
3900000	5	5	2	42.0	8.0	51.657504	39.149040	0	0	394038.0	138251.0	2122730.0
3200000	25	25	1	35.1	6.0	51.712799	39.198546	0	0	394053.0	459265.0	2162946.0

Рис. 6. Исходные данные

Удаляем строки, содержащие пропуски или некорректные данные, например, когда площадь квартиры меньше 0.

Заметим, что сравнивать все регионы вместе не совсем корректно, поскольку в зависимости от города может сильно отличаться цена за 1 квадратный метр. В качестве примера будем рассматривать г. Воронеж. Для этого сначала отберём квартиры в 36 регионе, а затем выберем только те квартиры, которые находятся в пределах Воронежа по координатам (координаты каждой квартиры представлены в данных).

Построим матрицу корреляции (рис. 7):

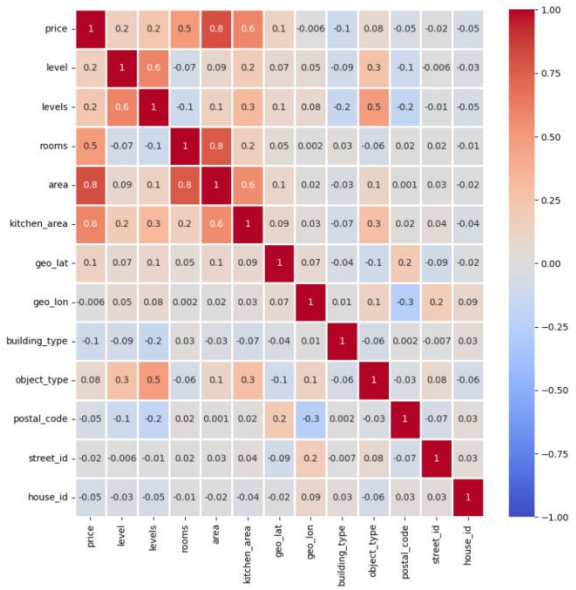


Рис. 7. Матрица корреляции для г. Воронеж

Целевой переменной является столбец price (цена квартиры).

Исключим столбцы, оказывающие слабое влияние на целевую переменную: postal_code (почтовый индекс), street_id (номер улицы), house_id (номер дома), geo_lat и geo _lon (координаты квартиры).

Когда две или более переменных имеют высокую корреляцию, они содержат почти одинаковую информацию, поэтому включаем только одну из них в модель. Высокая корреляция между переменными может привести к проблеме, известной как мультиколлинеарность [8], когда становится трудно определить независимое влияние каждой переменной на целевую переменную. Мультиколлинеарность негативно влияет на модели машинного обучения. Так, например, для линейных моделей она может приводить к неустойчивости коэффициентов.

Обратим внимание, что rooms (количество комнат) и area (общая площадь) имеют высокий коэффициент корреляции, поэтому можем исключить из рассмотрения rooms, так как у этого столбца меньшая корреляция с целевой переменной.

Разделим данные на тренировочные и тестовые, выделив под последние 10% от всех данных.

Предполагаем, что между характеристиками дома и его стоимостью линейная зависимость. Обучим модель линейной регрессии на тренировочной выборке и предскажем значения на тестовой, сравнивая с настоящими (рис. 8):

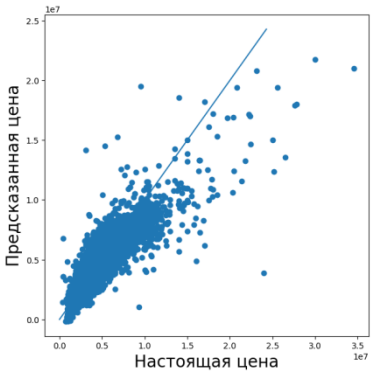


Рис. 8. Сравнение предсказанной цены и реальной для линейной регрессии для г. Воронеж

Оценка точности данной модели линейной регрессии — 0,73. Точность модели машинного обучения измеряется по шкале от 0 до 1, следовательно, получены средние результаты. Однако попробуем его улучшить за счет использования другого метода — бустинга (рис. 9):

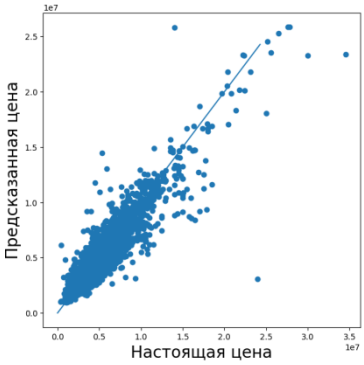


Рис. 9. Сравнение предсказанной цены и реальной для бустинга для г. Воронеж

Оценка точности данной модели — 0,86.

И попробуем также построить один из видов деревьев принятия решений — регрессионное дерево (рис. 10):

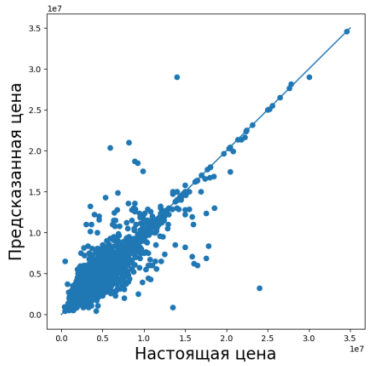


Рис. 10. Сравнение предсказанной цены и реальной для регрессионного дерева для г. Воронеж

Оценка точности данной модели — 0,84.

Самый лучший результат показал бустинг. Однако попробуем и его улучшить за счёт предобработки исходных данных.

В таблице изначально присутствовали координаты квартир, но в необработанном виде они оказывали слабое влияние на целевую переменную. Рассчитаем расстояние от каждой квартиры до центра города. Центром города считается главный почтамт. Под расстоянием подразумевается геодезический путь, то есть кратчайший путь с учетом эллиптической формы земли [6].

Построим модель бустинга с учётом расстояния до центра города (рис. 11):

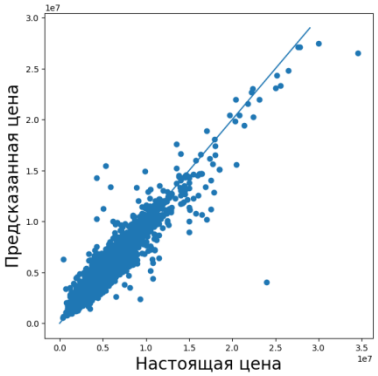


Рис. 11. Сравнение предсказанной цены и реальной для бустинга с учетом расстояния до центра для г. Воронеж

Оценка точности данной модели — 0,91.

Предположим, что размер города влияет на то, насколько сильно расстояние до центра коррелирует с ценой квартиры. Проделаем аналогичные действия для крупного города — Москвы. Без учета расстояния получаем точность модели — 0,83, а с учетом — 0,93. А также для маленького города — Борисоглебск. Без учета расстояния получаем точность модели — 0,7, а с учетом — 0,61.

Заметим, что расстояние до центра для Москвы действительно оказалось более значимым, чем для Воронежа. А для Борисоглебска результаты и вовсе оказались противоположными. Объяснением тому может служить, что из любой точки Борисоглебска дорога до центра не займёт слишком много времени, зато на окраине более современные дома.

Заключение

В ходе работы были рассмотрены основные понятия машинного обучения, прогнозирования и корреляции. Также были рассмотрены основные алгоритмы машинного обучения. Рассмотрена задача прогнозирования стоимости квартиры, которая реализована на языке программирования Python. Точность модели получилась достаточно высокой. Можно сделать вывод о том, что важен не только выбор подходящей модели машинного обучения, но и качественная предобработка данных.

Литература

1. Смирнова Л.А. Машинное обучение в прогнозировании / Л.А. Смирнова, Е.М. Аристова // Актуальные проблемы прикладной математики, информатики и механики : сб. тр. Междунар. науч.-техн. конф. (Воронеж, 4-6 декабря 2023 г.) : электронный ресурс. – Воронеж, 2023. – С. 658–665.
2. Сапрыкин О.Н. Интеллектуальный анализ данных: учебное пособие / О.Н. Сапрыкин. – Самара : Издательство Самарского университета, 2020. – 80 с.
3. Барсегян А.А. Метода и модели анализа данных: OLAP и Data Mining / А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод. – Санкт-Петербург : БХВ-Петербург, 2004. – 336 с.
4. Щеголева С.А. Корреляционный анализ в статическом контроле качества: методические указания / С.А. Щеголева; Инженерная школа ДВФУ. – Владивосток : Дальневосточный федеральный университет, 2014. – 37 с.
5. Лимановская О.В. Основы машинного обучения: учебное пособие / О.В. Ливановская, Т.И. Алферьева; Министерство науки и высшего образования РФ. – Екатеринбург : Издательство Уральского университета, 2020. – 88 с.
6. Чугреев И.Г. Основы геодезии: учебно-методическое пособие. – Москва : МИИГАиК, 2017. – 146 с.
7. Коэффициенты корреляции. Студопедия. – URL : https://studopedia.ru/17_110094_koeffitsienti-korrelyatsii.html. – (Дата обращения: 25.10.2023).
8. Что такое мультиколлинеарность? Вот все, что вам нужно знать. Skine. – URL : https://skine.ru/articles/260134/. – (Дата обращения: 26.10.2023).

УДК 004

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ «ИНТЕРАКТИВНАЯ ГАЗЕТА ВОРОНЕЖСКОЙ ОБЛАСТИ» НА ПЛАТФОРМЕ DJANGO

А. П. Голованова

Воронежский государственный университет

Введение

Широкое распространение получили веб приложения в виде социальных сетей, включая интерактивные газеты, однако в настоящее время доступ ко многим из них закрыт на территории Российской Федерации.

По этой причине возникла необходимость создания веб-приложения, которое даёт возможность любому пользователю свободно написать свою или прокомментировать чью-то статью, при необходимости отредактировать или удалить её. Данное приложение должно соответствовать современным требованиям к процессам аутентификации пользователя.

1. Анализ функциональности

У каждого пользователя есть возможность просмотра статей, однако написать новую или прокомментировать уже существующую можно только после успешного входа в систему. Если появляется необходимость смены пароля, то ссылка на восстановление приходит на почту, указанную при регистрации. Может быть использован также фильтр статей по дате и автору.

При взаимодействии с сайтом пользователю предоставляется ряд возможностей (рис. 1).

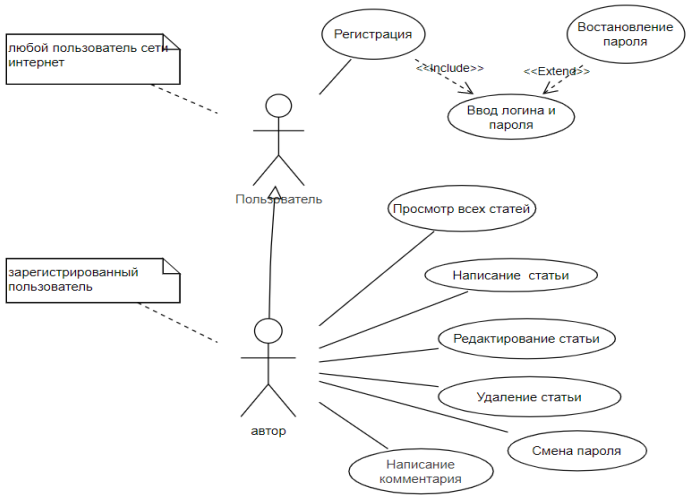


Рис. 1. Диаграмма вариантов использования приложения

Администратору доступно ручное управление параметрами учётных записей, администрирование базы данных, проверка журналов и логов приложения.

2. Технические требования

Приложение должно открываться во всех основных браузерах(Safari macOS, MicrosoftEdge, GoogleChrome, Яндекс.Браузер, MozillaFirefox), быть доступным к просмотру с разных устройств(ПК, ноутбук, смартфон), а также поддерживать понятную и удобную навигацию при переходах по внутренним страницам. Скорость открытия ресурса не должна превышать 3 секунд.

Разработка приложения предусматривает использование фреймворков и библиотек со встроенными функциями безопасности, которые помогут свести к минимуму появление уязвимостей в процессе реализации.

Для защиты от межсайтового выполнения сценариев (XSS-атаки) используется экранирование (добавление определённых комбинаций символов перед символами или строками для недопущения их некорректной интерпретации).

Архитектура приложения должна обладать гибкостью для обеспечения легкости добавления новых функций.

3. Средства реализации

Программной основой служит фреймворк Django. Он использует архитектуру model-view-template (MVT), шаблон, основанный на наборе лучших практик для организации кода.

В качестве СУБД была выбрана SQLite. Она является самой простой на сегодняшний день, потому что работает с одним файлом и не требует сложной установки. Django использует SQLite по умолчанию.

Для упрощения работы с элементами HTML был использован Bootstrap - свободный набор инструментов для создания сайтов и веб-приложений. Для сброса пароля был использован сторонний сервис «Яндекс Почта».

В базе данных хранятся сведения о пользователе, статье и комментарии (рис.2).

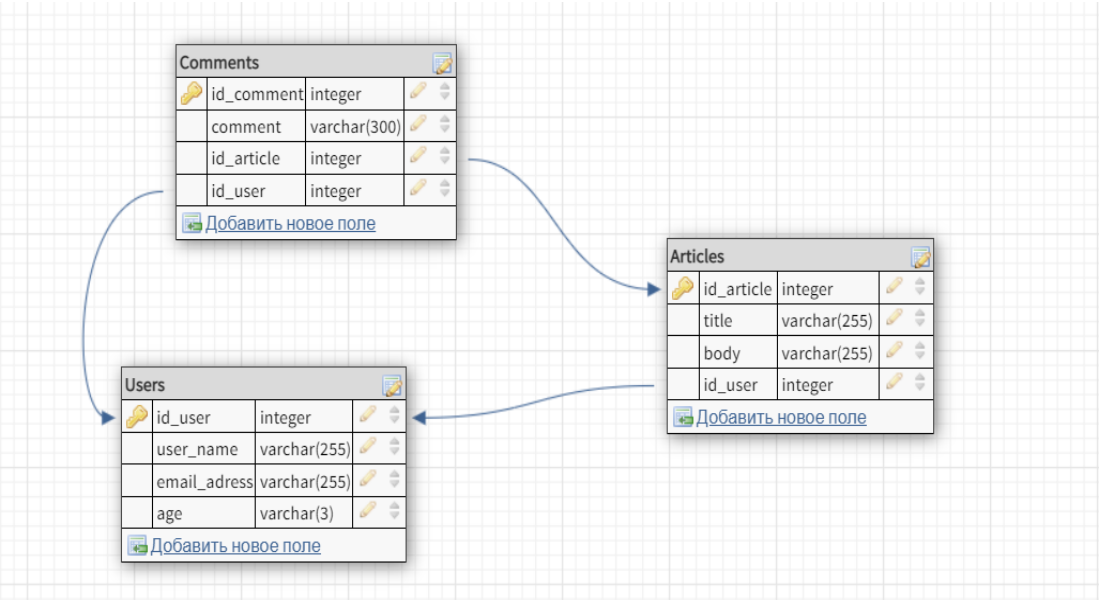


Рис. 2. Схема базы данных приложения

Заключение

В результате работы создано web-приложение, отвечающее всем требованиям постановки задачи. Оно предоставляет пользователю ряд функциональных возможностей:

написания статей, удаление и редактирование, создание комментариев к определённой статье, возможность входа/выхода из системы, сброс и изменение пароля. Разработан рабочий интерфейс, реализован механизм ввода и сохранения введенной информации. Во время работы были проанализированы необходимые возможности приложения, выбрана модель для его реализации, спроектирована структура сайта, разработана база данных, выбран сервис для отправления ссылки на восстановление пароля. Информационная система рассчитана на малую и среднюю нагрузку в зависимости от используемых серверных мощностей.

Литература

1. Форсье Дж Django: Разработка веб-приложений на Python / Дж Форсье – Москва: Символ-Плюс, 2018. – 456 с.
2. Персиваль Г. Архитектурные шаблоны на Python: разработка на основе тестов, проектирование на основе предметной области и микросервисы, управляемые событиями/ Г. Персиваль. – Москва : Символ-плюс, 2020. – 343 с.
3. Головатый А. Django. Подробное руководство. / А. Головатый – Москва : Символ-Плюс, 2018. – 549 с.
4. Vincent W. S. Django for Beginners: Build Websites with Python and Django (Welcome to Django) / W. S. Vincent. – California : WelcomeToCode, 2020. – 317 с.

УДК: 004

АНАЛИЗ МЕТОДА МАТРИЧНОЙ ФАКТОРИЗАЦИИ FUNCK SVD В РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМАХ

Д. О. Головатюк

Воронежский государственный университет

Введение

Рекомендательные системы имеют большую актуальность в настоящее время из-за постоянно растущего объема информации [1,2]. Они помогают пользователям находить необходимую информацию, товары или услуги, улучшая их опыт пользователей и способствуя увеличению объемов продаж для компаний. Благодаря развитию технологий машинного обучения и искусственного интеллекта, рекомендательные системы могут анализировать множество данных о пользователях и их предпочтениях, предлагая им персонализированные рекомендации. Это увеличивает вероятность того, что пользователи совершат покупку или найдут нужную информацию, что в свою очередь улучшает конверсию и общую прибыль компаний. Таким образом, актуальность рекомендательных систем неизменно растет, и они играют важную роль в повышении эффективности бизнеса и улучшении пользовательского опыта.

1. Постановка задачи

Существует множество алгоритмов для построения рекомендательных сервисов, которые делятся по способу формирования данных. Выделяют четыре вида рекомендательных систем: контентная фильтрация; коллаборативная (совместная) фильтрация; фильтрация, основанная на знаниях; гибридные рекомендательные системы.

Настоящая работа посвящена реализации и анализу алгоритма матричной факторизации в применении к задаче формирования рекомендаций книг.

Рассматривается матрица оценок, в которой отображаются данные о рейтингах книг для каждого пользователя. Каждая строчка матрицы соответствует рассматриваемому пользователю, а столбец - конкретной книге. Используется номинальная шкала оценок от 1 до 5, в которой оценка 1 соответствует понятию «совсем не понравилось», а оценка 5 – понятию «очень понравилось». Таким образом, на пересечении строк и столбцов матрицы будет располагаться значение от 1 до 5, соответствующее оценке, поставленной пользователем просмотренной книге. Задача заключается в том, чтобы построить такую модель, которая сможет порекомендовать пользователю ранее не им прочтенные книги, которые с большой вероятностью могли бы его заинтересовать.

2. Матричная факторизация

Матричная факторизация — это операция разложения матрицы взаимодействия пользователя и элемента на произведение нескольких матриц меньшей размерности. Также результаты прогнозирования можно улучшить, присвоив скрытым факторам различные веса регуляризации в зависимости от популярности элементов и активности пользователей [3,5].

Одним из наиболее часто используемых методов матричной факторизации является

метод SVD (сингулярное разложение), где большое значение возложено на матрицу оценок, которая довольно разрежена. Более приемлемый метод для расчетов матриц с большим количеством пропусков – метод Funck SVD [6]. Вместо того, чтобы применять всю матрицу, он использует только то, что нужно знать.

2.1. Построение факторизации с помощью Funck SVD

Необходимо создать две матрицы **Q** (матрица факторов пользователей) и **P** (матрица факторов элементов), при перемножении которых мы окажемся как можно ближе к исходной матрице:

$$[\mathbf{M}] = \mathbf{Q} \cdot \mathbf{P} \quad (1)$$

где каждая оценка представима в виде:

$$r_{u,i} = q_i \cdot p_i \quad (2)$$

Цель состоит в том, чтобы взять все оценки и создать две матрицы таким образом, чтобы i строка матрицы факторов элементов умножалась на u столбец матрицы факторов пользователей, в результате чего должна получится матрица, содержащая похожие оценки. Необходимо найти такие матрицы **Q** и **P**, которые минимизируют уравнение (3) с использованием алгоритма стохастического градиентного спуска:

$$\min_{p,q} = \sum_{(u,i)\in K}^{n} (r_{u,i} - p_i q_i)^2 \quad (3)$$

Чтобы исключить переобучение модели, вводится коэффициент регуляризации. Тогда минимизируется следующая функция, которая рассчитывает, как сильно предсказанная оценка отличается от реальной:

$$\min_{p,q} \sum_{(u,i)\in known}^{n} (r_{u,i} - p_i q_i) + \lambda(\square q \square^2 + \square p \square^2) \quad (4)$$

Взяв частные производные по каждой из оптимизируемых переменных, получаются правила для стохастического градиентного спуска, эффективного с точки зрения производительности, где единовременно рассматривается одна оценка. Используя рассчитанные ошибки, умножая их на скорость обучения (γ), и вычитая собственный вектор, умноженный на коэффициент регуляризации, получаются обновленные вектора p и q:

$$e_{ui} = r_{ui} - q_i p_u \quad (5)$$

$$q_i = q_i + \gamma(e_{ui} p_u - \lambda q_i) \quad (6)$$

$$p_u = p_u + \gamma(e_{ui} q_i - \lambda p_u) \quad (7)$$

2.2. Факторизация матриц с учетом отклонения пользователя и элемента

Для получения более точной оценки вводятся базовые предикторы, которые складываются из базовых предикторов отдельных пользователей b_u и базовых предикторов отдельных продуктов b_i, а также общего среднего рейтинга по базе μ, тогда оценка будет задаваться следующим образом:

$$r_{u,i} = q_i \cdot p_i + b_u(t) + b_i(t) \quad (8)$$

Тогда, с учетом этих поправок, все вышеупомянутые формулы изменятся соответственно:

Функция минимизации:

$$\min_{p,q} \sum_{(u,i)\in known}^{n} (r_{u,i} - p_i q_i) + \lambda(\sum {b_i}^2 + \sum {b_u}^2 + \square q \square^2 + \square p \square^2) \quad (9)$$

Правила градиентного спуска:

$$e_{ui} = r_{ui} - \mu - b_i - b_u - q_i p_u \quad (10)$$

$$b_i = b_i + \gamma(e_{ui} - \lambda b_i) \quad (11)$$

$$b_u = b_u + \gamma(e_{ui} - \lambda b_u) \quad (12)$$

$$q_i = q_i + \gamma(e_{ui} p_u - \lambda q_i) \quad (13)$$

$$p_u = p_u + \gamma(e_{ui} q_i - \lambda p_u) \quad (14)$$

Матрица предсказанных оценок в обоих случаях рассчитывается как матрица вида:

$$\mathbf{M} = \begin{pmatrix} r_{00} & \cdots & r_{0m} \\ \vdots & \ddots & \vdots \\ r_{n0} & \cdots & r_{nm} \end{pmatrix} \quad (15)$$

3. Реализация метода и результаты расчетов

В рамках реализации на языке Python были создан класс FunckSVDRecommender, который имеет общие функции для двух задач (fit()-обучение модели, score()-оценка качества модели и предсказание книги для коктретного пользователя – recommender()), и изменяемые функции, абстрактно (predict()-предсказание неизвестных оценок). Абстрактные функции реализованы уже в каждом из дочерних классов, а именно FunckSVDBasic – классическая метод факторизации и FunckSVDPredictors – метод факторизации с учетом отклонений. UML-диаграмма классов представлена на рис 1.

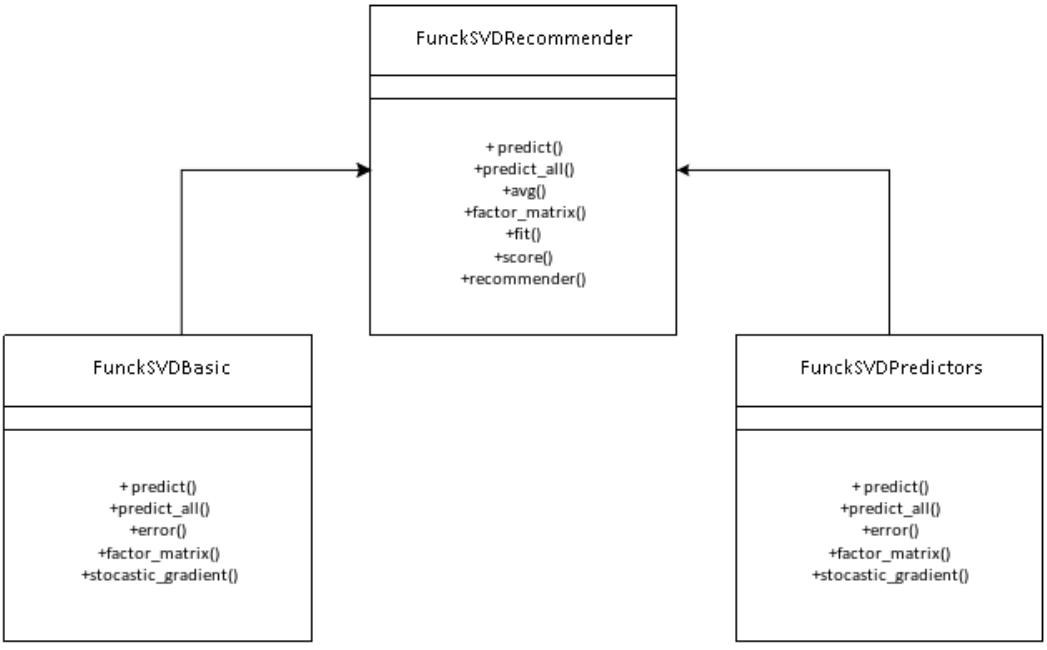


рис 1. UML-диаграмма классов

Для обучения модели были взяты следующие данные по пользователям и книгам:

Таблица 1

Матрица рейтингов книг по каждому пользователю

	Name	Book1	Book2	…	Book9	Book10
0	Sadie Hogan	0	1	…	1	1
1	Carlos Woods	4	1	…	1	4
2	Randy Cannon	5	4	…	0	2
3	Nina Jordan	2	2	…	3	5
⋮	⋮	⋮	⋮	…	⋮	⋮
97	Theodore Harmon	4	1	…	5	3
98	Howard Barber	5	0	…	3	2
99	Mina Gregory	4	3	…	2	3

[100 rows x 11 columns]

Зафиксировав подобранные параметры, при которых модели имеет наивысшую оценку качества, были обучены модели FunckSVDBasic и FunckSVDPredictors со следующими параметрами (инициализация признаков – 40, скорость обучения – 0.002, регуляризация – 0.02, сколько итераций – 8000). В результате были получены следующие матрицы с предсказанными значениями:

Таблица 2

Матрица рейтингов книг по каждому пользователю с предсказанными значениями для модели FunckSVDBasic

	Name	Book1	Book2	…	Book9	Book10
0	Sadie Hogan	2.981095	1.027285	…	1.031167	1.033681
1	Carlos Woods	3.953378	1.014867	…	1.019634	3.964364
2	Randy Cannon	4.972355	3.982476	…	3.157550	2.018092
3	Nina Jordan	1.999474	1.996955	…	2.981191	2.987688
⋮	⋮	⋮	⋮	…	⋮	⋮
97	Theodore Harmon	1.039924	2.981232	…	4.955329	2.986131
98	Howard Barber	2.482403	3.975390	…	2.996620	2.010612
99	Mina Gregory	2.981814	2.010889	…	2.007855	2.999038

[100 rows x 11 columns]

Оценка модели по r2-метрике: 0.9896351098154793

Таблица 3

Матрица рейтингов книг по каждому пользователю с предсказанными значениями для модели FunckSVDPredictors

	Name	Book1	Book2	…	Book9	Book10
0	Sadie Hogan	3.258762	1.042401	…	1.042717	1.043194
1	Carlos Woods	3.973937	1.034807	…	3.973937	1.034807
2	Randy Cannon	4.975120	3.985766	…	3.164921	2.016362
3	Nina Jordan	2.013451	2.012639	…	2.995147	3.004741
⋮	⋮	⋮	⋮	…	⋮	⋮

97	Theodore Harmon	3.985813	1.050345	…	4.960945	2.996196
98	Howard Barber	4.964578	2.945957	…	3.002672	2.013947
99	Mina Gregory	3.982250	2.992259	…	2.016043	3.003268

[100 rows x 11 columns]

Оценка модели по r2-метрике: 0.9996344520800895

Для оценки качества построенных моделей была использована R2-метрика. Суть ее работы заключается в измерении количества отклонений в прогнозах, объясненных набором данных. R2-метрика вычисляется по формуле [4]:

$$R^2 = 1 - \frac{\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{n}(y_k - \langle y \rangle)^2} \quad (16)$$

где:

$y_k, \hat{y}_k$ – фактические и расчетные значения рассматриваемой переменной.

$\langle y \rangle$ – среднее арифметическое всех фактических переменных

В случае модели, построенной с помощью классической факторизации, оценка равна 0.9896351098154793. В случае модели, построенной с помощью факторизации с учетом отклонений, оценка равна 0.9996344520800895. По данным оценкам видно, что метод, основанный на учете отклонений, показывает лучший результат, а также можно сделать вывод о хорошем качестве моделей поскольку оценка стремится к 1.

Выводы

Таким образом, был реализован метод FunckSVD для генерации рекомендаций книг пользователю, который с высокой точностью может предсказать, какая книга могла бы заинтересовать того или иного пользователя.

Литература

1. Фальк, К. Рекомендательные системы на практике: практическое руководство / К. Фальк; пер. с англ. Д. М. Павлова. – Москва: ДМК Пресс, 2020. - 448 с.

2. А. Р. Кутянин Рекомендательные системы: обзор основных постановок и результатов, Интеллектуальные системы. Теория и приложения / А. Р. Кутянин. – 2017. – Т. 21, № 4. – С.18–30

3. Сравнение матричной факторизации с трансформерами на наборе данных MovieLens с применением библиотеки pytorch-acceleratd. – Режим доступа: https://habr.com/ru/companies/wunderfund/articles/645921/ – (Дата обращения: 11.03.2023).

4. Оценка R2 в машинном обучении. – Режим доступа: https://biconsult.ru/products/ocenka-r2-v-mashinnom-obuchenii – (Дата обращения: 17.03.2023).

5. Анализ алгоритмов для системы рекомендаций. – Режим доступа: https://www.kubsu.ru/sites/default/files/users/18392/portfolio/chenib_020303_kursovaya_2sem.pdf – (Дата обращения: 20.03.2023).

6. Simon Funk's Netflix Recommendation System in Tensorflow. – Режим доступа: https://temugeb.github.io/tensorflow/python/svd/2021/02/04/Funk-SVD-recommender-p1.html – (Дата обращения: 20.03.2023).

УДК *004*

МЕТОД МАТРИЧНОЙ ФАКТОРИЗАЦИИ SVD В РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМА

Е. О. Головатюк

Воронежский государственный университет

Введение

Рекомендательные системы имеют большую актуальность в современном мире из-за большого объема информации, доступной пользователю. Они помогают упорядочить и фильтровать информацию, предлагая пользователям наиболее подходящий контент на основе их интересов. Также они способствуют улучшению пользовательского опыта, обеспечивая персонализированный контент, что увеличивает вероятность удовлетворения потребностей пользователя и удержания его на платформе.

В данной работе рассматривается возможность применения метода матричной факторизации SVD для получения рекомендаций кулинарных рецептов [1].

1. Постановка задачи

Будем рассматривать матрицу, в которой приведены данные о рейтингах рецептов для каждого пользователя. То есть каждая строка будет соответствовать конкретному пользователю, а каждый столбец конкретному рецепту. На пересечении строки и столбца будет стоять оценка, в диапазоне от 1 до 5, которую пользователь поставил определенному рецепту. Задача будет состоять в том, чтобы порекомендовать пользователю рецепт, который находится в матрице оценок, но который он еще не просматривал.

2. Матричная факторизация методом SVD

Существует множество способов формирования рекомендаций на основе совместной фильтрации. Один из них — это матричная факторизация. Она является распространенным и эффективным способом внедрения системы рекомендаций. Алгоритмы матричных факторизаций работают путем разложения матрицы взаимодействия пользователя и элемента на произведение нескольких матриц меньшей размерности. Идея матричных факторизаций заключается в представлении пользователей и элементов в скрытом пространстве меньшей размерности. Одним из наиболее часто используемых методов матричной факторизации является алгоритм SVD. Он используется при решении самых различных задач: в сфере электронной торговли: оперативная классификация продукции и анализ поведения пользователей; решение проблемы рекомендации мультимедийного контента; решение проблемы распознавания изображений.

Сингулярное разложение матриц — это представление прямоугольной матрицы в виде произведения нескольких матриц особого вида. Пусть $\mathbf{A}$ – матрица оценок. Благодаря SVD-разложению можно представить матрицу $\mathbf{A}$ в виде:

$$\mathbf{A} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V}^T \quad (1)$$

где $\mathbf{U}$ — унитарная матрица признаков пользователей, $\mathbf{V}$ — унитарная матрица признаков элементов., $\mathbf{\Sigma}$ — матрица, на главной диагонали которой лежат неотрицательные числа, которые называются сингулярными числами матрицы. В свою очередь эти числа располагаются в порядке убывания, начиная с первого максимального сингулярного значения [1–3].

2.1. Основной алгоритм реализации метода SVD

Шаг 1: Вычисление матрицы симметричной факторизации $\mathbf{A}^T \times \mathbf{A}$. Для вычисления полного SVD матрицы $\mathbf{A}$ сначала нужно вычислить собственные значения и собственные векторы некоторой симметричной вещественной или унитарной матрицы, полученной как произведение матриц $\mathbf{A}^T$ и $\mathbf{A}$. Произведение этих двух матриц дает симметричную матрицу, для которой легко вычисляются собственные значения δ.

Шаг 2: Вычисление сингулярных значений матрицы $\mathbf{A}^T \times \mathbf{A}$. Найдем диагональную матрицу сингулярных значений $\mathbf{S}$ путем вычисления квадратного корня из каждого собственного значения и разместить полученные числа вдоль диагонали матрицы $\mathbf{S}$:

$$\mathbf{S} = \begin{pmatrix} \sqrt{\delta_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sqrt{\delta_n} \end{pmatrix} \quad (2)$$

Чтобы найти обратную матрицу $\mathbf{S}^{-1}$ нам нужного разделить 1 на каждое из сингулярных значений, которые стоят на диагонали матрицы $\mathbf{S}$:

$$\mathbf{S}^{-1} = \begin{pmatrix} \frac{1}{s_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{s_n} \end{pmatrix} \quad (3)$$

Шаг 3: Вычисление правых сингулярных векторов матрицы $\mathbf{A}^T \times \mathbf{A}$. Для составления матрицы правых сингулярных векторов матрицы $\mathbf{A}^T \times \mathbf{A}$ потребуются собственные значения матрицы $\mathbf{A}^T \times \mathbf{A}$. Алгоритм их нахождения будет рассмотрен позднее. Пусть $\mathbf{V} = (\mathbf{V_1}, \mathbf{V_2}, \dots, \mathbf{V_n})$ – собственные векторы матрицы $\mathbf{A}^T \times \mathbf{A}$, которые соответствуют собственным значениям $\delta_1, \delta_2, \dots, \delta_n$ матрицы $\mathbf{A}^T \times \mathbf{A}$. Найдем абсолютную скалярную длину L каждого из этих векторов:

Пусть $\mathbf{V}_i = (v_{i1}, v_{i2}, \dots, v_{in})$ — собственный вектор матрицы $\mathbf{A}^T \times \mathbf{A}$, тогда

$$L_i = \sqrt{{v_{i1}}^2 + {v_{i2}}^2 + \dots + {v_{in}}^2} \quad (4)$$

Теперь каждый из собственных векторов матрицы $\mathbf{A}^T \times \mathbf{A}$ нужно поделить на

соответствующую абсолютную скалярную длину L_i. Запишем полученные векторы вдоль строк в матрице $\mathbf{V}$:

$$\mathbf{V} = \begin{pmatrix} \frac{v_{01}}{L_0} & \cdots & \frac{v_{0n}}{L_0} \\ \vdots & \ddots & \vdots \\ \frac{v_{n1}}{L_n} & \cdots & \frac{v_{nn}}{L_n} \end{pmatrix} \quad (5)$$

Транспонирования матрицам $\mathbf{V}^T$ — ортогональная матрица правых сингулярных векторов.

Шаг 4: Вычисление левых сингулярных векторов матрицы $\mathbf{A}^T \times \mathbf{A}$. Получим матрицу левых сингулярных векторов $\mathbf{U}$ как произведение данной матрицы $\mathbf{A}$ на матрицу правых сингулярных значений $\mathbf{V}$ и обратную диагональную матрицу $\mathbf{S}^{-1}$ [3].

2.2. Основной алгоритм реализации метода SVD

Шаг 1. Вычисление максимального собственного значения матрицы $\mathbf{A}^* = \mathbf{A}^T \times \mathbf{A}$. Для этого используется метод простых итераций:

1. Начальные данные для алгоритма: $\mathbf{A}^* = \mathbf{A}^T \times \mathbf{A}$ – матрица факторизации, $\mathbf{R} = (1,\ldots,1)$ — исходный единичный вектор, e — постоянное значение погрешности точности алгоритма, $\mathbf{V}^*$ — вектор, содержащий результаты вычислений на каждой итерации, $\mathbf{M}$ — результирующая матрица, столбцами которой являются значения вектора $\mathbf{V}^*$
2. δ, δ_{old} — собственные значения для текущей и предыдущей итерации. Изначально $\delta = 0$, а $\delta_{old} = \delta$.
3. Вектор $\mathbf{V}$ равен произведению матрицы $\mathbf{A}^*$ на вектор $\mathbf{R}$
4. Разместим вектор $\mathbf{V}$ вдоль столбца результирующей матрицы $\mathbf{M}$.
5. $\mathbf{R} = \mathbf{V}$.
6. Если это не первая итерация алгоритма ($i > 0$), то нужно вычислить значение δ:

$$\delta = \frac{\mathbf{M}(1,i)}{\mathbf{M}(1,i-1)} \quad (6)$$

7. Если это не первая итерация ($i > 0$), то нужно найти разницу $\Delta = \delta - \delta_{old}$
8. Если значение Δ больше, чем значение ошибки точности e, то нужно вернуться к шагу 2, в противном случае нужно завершить вычисления и перейти к шагу 9.
9. После проделанных вычислений будет найдено максимальное собственное значение для данной матрицы $\mathbf{A}^*$: $\delta \to \delta_{\max}$

Шаг 2: Вычисление собственного вектора матрицы $\mathbf{A}^*$. Найдем собственный вектор, соответствующий $\delta_{\max}$. Для этого следует вычесть это собственное значение из каждого

диагонального элемента матрицы $\mathbf{A}^*$, а затем нужно найти нетривиальное решение для полученной системы линейных уравнений с помощью преобразования Жордана — Гаусса.

Вычисление последующих собственных значений и собственных векторов происходит с помощью матрицы $\mathbf{B}$ уменьшенного размера, которая является эквивалентной матрице $\mathbf{A}^*$. Для матрицы $\mathbf{B}$ делаются те же самые вычисления, что и на шагах 1 и 2.

$$\mathbf{B}^* = \mathbf{H} \times \mathbf{A}^* \times \mathbf{H}^{-1} = \begin{pmatrix} b^*_{11} \dots & b^*_{1n} \\ \vdots & \mathbf{B} \\ b^*_{m1} & \end{pmatrix} \quad (7),$$

где $\mathbf{H}$ — эрмитова сопряжение матрицы $\mathbf{A}^*$, $\mathbf{H}^{-1}$ – эрмитова сопряженная обратная матрицы $\mathbf{A}^*$, $\mathbf{B}$ — эквивалент матрицы $\mathbf{A}^*$

$$\mathbf{H} = \begin{pmatrix} \frac{1}{x_1} & 0 & \dots & 0 \\ \frac{x_2}{x_1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_m}{x_1} & 0 & \dots & 1 \end{pmatrix} \quad (8)$$

$$\mathbf{H}^{-1} = \begin{pmatrix} x_1 & 0 & \dots & 0 \\ -x_2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -x_m & 0 & \dots & 1 \end{pmatrix} \quad (9)$$

где x_i — составляющее собственного вектора $\mathbf{X}$.

Каждый собственный вектор, вычисленный на шаге 2, на самом деле не является собственным вектором матрицы факторизации $\mathbf{A}^* = \mathbf{A}^T \times \mathbf{A}$. Эти векторы намеренно вычисляются для того, чтобы была возможность найти эквивалентную матрицу $\mathbf{B}$, для которой будет находится максимальное собственное значение $\delta_{\max}$, которое является следующим собственным значением матрицы факторизации $\mathbf{A}^* = \mathbf{A}^T \times \mathbf{A}$. Все собственные векторы матрицы факторизации $\mathbf{A}^* = \mathbf{A}^T \times \mathbf{A}$ вычисляются отдельно с помощью метода Жордана — Гаусса [6].

2.2. Основной алгоритм реализации метода SVD

Базовые предикторы могут быть использованы для заполнения пустых ячеек матрицы

оценок для реализации SVD.

Базовый предиктор:

$$b_{ui} = \mu + b_u + b_i \quad (10),$$

где b_{ui} — базовый прогноз элемента i для пользователя u, b_u — отклонение пользователя, b_i — отклонение элемента, μ — среднее арифметическое всех оценок [3].

Сумма разниц между оценками пользователей и средним, деленная на число оценок:

$$b_u = \frac{1}{|I_u|}\sum_{i \in I_u}(r_{(u,i)} - \mu) \quad (11)$$

$$b_i = \frac{1}{|U_i|}\sum_{u \in U_i}(r_{(u,i)} - b_u - \mu) \quad (12)$$

3. Реализация метода и результаты расчетов

На языке Python были реализованы классы SVDmodel и DataPrepare. Первый класс отвечает за создание модели и имеет стандартные функции обучения fit(), предсказания неизвестных оценок predict() и предсказания для конкретного пользователя recommendation(). Второй класс отвечает за предварительную подготовку данных и имеет функцию BasePredictor(), которая позволяет заполнять неизвестные оценки в матрице базовыми предикторами.

В качестве тестируемых выборок были рассмотрены матрицы различного объема. Например, была взята следующая выборка:

Таблица 1

Матрица рейтингов блюд по каждому пользователю

		Recipe1	Recipe2	…	Recipe9	Recipe10
0	User1	0	1	…	1	1
1	User2	4	1	…	1	4
2	User3	5	4	…	0	2
3	User4	2	2	…	3	5
⋮	⋮	⋮	⋮	…	⋮	⋮
197	User198	4	1	…	5	3
198	User199	5	0	…	3	2
199	User200	4	3	…	2	3

После применения метода SVD к данной выборке получилась следующая выборка с предсказанными значениями:

Таблица 2

Матрица рейтингов блюд по каждому пользователю с предсказанными значениями

		Recipe1	Recipe2	…	Recipe9	Recipe10
0	User1	2.981095	1.027285	…	1.031167	1.033681
1	User2	3.953378	1.014867	…	1.019634	3.964364
2	User3	4.972355	3.737537	…	3.157550	2.018092
3	User4	1.999474	1.821051	…	2.981191	4.987688
⋮	⋮	⋮	⋮	…	⋮	⋮
197	User198	3.979593	1.039924	…	4.955329	2.986131
198	User199	4.982403	2.482403	…	2.996620	2.010612
199	User200	3.97346	2.981814	…	2.007855	2.999038

Для того, чтобы оценить качество построенной модели, было решено сравнить результат работы реализованного метода SVD с встроенным:

Таблица 3

Сравнение предсказанных оценок с помощью реализованного и встроенного метода SVD

Реализованный метод SVD	Встроенный метод SVD
Оценка Recipe1 для User1	
2.981095	3.258762
Оценка Recipe10 для User7	
3.984933	3.990930
Оценка Recipe9 для User13	
1.937257	1.937555
Оценка Recipe2 для User22	
1.988615	2.674512

Можно заметить, что полученные значения примерно совпадают со значениями, полученными с помощью встроенного метода.

Также качество модели было оценено с помощью R^2-метрики. Оценка R^2 – это показатель, который используется для оценки производительности модели машинного обучения. Принцип его работы заключается в измерении количества отклонений в прогнозах, объясненных набором данных. Иначе говоря, это разница между выборками в наборе данных и прогнозами, сделанными моделью [4].

R^2-метрика определяется по формуле:

$$R^2 = 1 - \frac{\sum_{k=1}^{n} (y_k - \hat{y}_k)^2}{\sum_{k=1}^{n} (y_k - \langle y \rangle)^2} \quad (13)$$

где:

$y_k, \hat{y}_k$ – фактические и расчетные значения рассматриваемой переменной.

$\langle y \rangle$ – среднее арифметическое всех фактических переменных

Оценка модели, основанная на выборке из Таблицы 1, по R^2-метрике получилась равной 0.97665333. Полученное значение близко к 1, что говорит о хорошем качестве модели.

Таким образом, был реализован метод SVD для рекомендации рецептов пользователям, который с достаточно хорошей точностью предсказывает предпочтения конкретных пользователей.

Литература

1. Фальк, К. Рекомендательные системы на практике : практическое руководство / К. Фальк ; пер. с англ. Д. М. Павлова. – Москва : ДМК Пресс, 2020. - 448 с.
2. А. Р. Кутянин Рекомендательные системы: обзор основных постановок и результатов, Интеллектуальные системы. Теория и приложения / А. Р. Кутянин. – 2017. – Т. 21, № 4. – С.18–30
3. Сравнение матричной факторизации с трансформерами на наборе данных MovieLens с применением библиотеки pytorch-acceleratd. – Режим доступа: https://habr.com/ru/companies/wunderfund/articles/645921/ – (Дата обращения: 11.03.2023).
4. Оценка R2 в машинном обучении. – Режим доступа: https://biconsult.ru/products/ocenka-r2-v-mashinnom-obuchenii – (Дата обращения: 17.03.2023).
5. Анализ алгоритмов для системы рекомендаций. – Режим доступа: https://www.kubsu.ru/sites/default/files/users/18392/portfolio/chenib_020303_kursovaya_2sem.pdf – (Дата обращения: 20.03.2023).
6. Гриншпон, И. Э. Линейная алгебра. Векторная алгебра. Аналитическая геометрия: Курс лекций [Электронный ресурс] / И. Э. Гриншпон. – Томск: ТУСУР, 2019. – 128 с. – Режим доступа: https://edu.tusur.ru/publications/8974 – (Дата обращения: 17.03.2023).

УДК *004.8*

РАСПОЗНАВАНИЕ И КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ БЛЮД С ПОМОЩЬЮ МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ

А. А. Гончарова

Воронежский государственный университет

Введение

В современном мире растущий интерес к методам глубокого обучения и их применению в области компьютерного зрения способствовал значительному улучшению возможностей распознавания и классификации объектов на изображениях. Одной из задач в этой области является распознавание и классификация изображений блюд. Это имеет большое значение для различных областей, таких как ресторанный бизнес, здравоохранение, диетология и фитнес. Актуальность данного проекта в том, что распознавание и классификация изображений блюд могут значительно улучшить процессы автоматизации кулинарного производства, помочь в создании персонализированных диет и улучшить качество обслуживания в ресторанном бизнесе. Также такая классификация может пригодиться при разработке систем мониторинга рациона, подсчета калорийности, при разработке систем самообслуживания на линиях раздачи блюд в столовых и заведениях быстрого питания.

1. Постановка задачи и этапы ее решения

Основная цель данной работы заключается в том, что на основе изображений блюд различной кухни следует обучить трансферные нейронные сети так, чтобы они могли распознавать и правильно классифицировать блюда.

Процесс разработки модели по предсказанию класса изображения блюда состоит из следующих этапов:

1) сбор и формирование исходных данных для обучающей и тестовой выборки;
2) предварительная обработка данных;
3) реализация нескольких моделей, использующих разные архитектуры с помощью трансферного обучения.
4) исследование моделей с лучшей точностью путем использования разных параметров в ходе обучения;
5) анализ полученных результатов обучения и выбор наилучшей модели и ее параметров.

2. Сбор, анализ и подготовка исходных данных

Исходные данные были взяты с конкурса «iFood 2019 Challenge». В изначальном датасете было 251 блюдо и примерно 80 тысяч изображений. Мною были дополнительно добавлены блюда и размерность датасета увеличилась до 288 блюд и более 100 тысяч изображений.

Максимальный размер изображения 400 на 400 пикселей, формат – JPEG. Датасет представлен тренировочной и валидационной выборками, а также текстовым файлом со списком всех блюд.

Предварительная модель используется для обработки входных данных. С помощью нее

мы нормализуем входные тензоры и обогатим данные.

Обогащение (аугментация) данных достигается комбинированием поворота, сдвига и масштабирования изображения. Аугментация данных может пригодиться не только для расширения датасета, но также для обучения более надежных моделей. Например, не на всех изображениях блюдо находится в центре или под идеальным углом в 0 градусов.

Предварительная обработка производилась с использованием библиотеки Keras языка Python. Данная библиотека предоставляет функцию ImageDataGenerator, которая проводит аугментацию данных в ходе их загрузки из каталога с помощью случайных преобразований изображений (поворот, приближение, отзеркаливание). Для нормализации (то есть для приведения значений в диапазон от 0 до 1) используется параметр rescale, который делит значение каждого канала в пикселе на 255.

Проверочный набор, в отличие от обучающего, мы не будем расширять с помощью аугментации. Причина в том, что при динамической аугментации проверочный набор будет меняться в каждой итерации, в результате чего оценка точности будет непоследовательной, из-за чего возникнут сложности при сравнении результатов разных итераций.

3. Построение нейросети и ее обучение

Мною была выбрана кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains на основе IntelliJ IDEA PyCharm для разработки нейронной сети, в качестве GPU использовался GPU видеокарты ноутбука.

Обучение будет производиться с помощью технологии «transfer learning». Так называется методика передачи модели обучения. Основная идея переноса обучения основывается на том, что обучив нейронную сеть на большом наборе данных, мы можем применить полученную модель и к набору данных, который раньше эта модель ещё не встречала.

Чтобы перенести знания из одной модели в другую, нужно повторно использовать больше универсальных слоев (расположенных ближе к входу) и меньше специализированных для конкретных задач (расположенных ближе к выходу). После начала обучения универсальные слои останутся замороженными (то есть их нельзя будет изменить), а вновь добавленные слои для конкретных задач будут доступны для изменения. Благодаря такому подходу прием переноса обучения помогает быстро обучать новые модели [1]. На рис. 1 показан процесс создания модели для решения задачи Y на основе модели, предварительно обученной для задачи X.

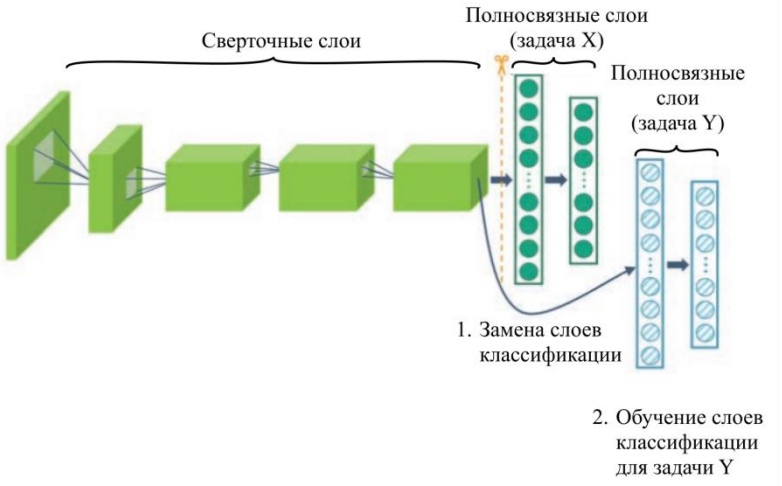


Рис.1. Общая схема переноса обучения

TensorFlow Hub содержит некоторые предобученные модели, в которых последний слой классификации был исключён из архитектуры нейронной сети для последующего переиспользования. Достаточно указать URL вектора признаков нужной обучающей модели и затем встроить модель в наш классификатор с указанием последнего слоя с нужным количеством выходных классов. Именно последний слой и будет подвергнут обучению и изменению значений параметров.

Сначала происходит создание модели.

```
model = tf.keras.Sequential([hub.KerasLayer(ef, input_shape=(IMG_HEIGHT,
IMG_WIDTH, 3),trainable = True, load_options = load_locally),
              tf.keras.layers.Dense(1024, activation='relu'),
              tf.keras.layers.Dropout(0.2),
              tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')])
```

Создание модели будет производиться с помощью класса Sequential библиотеки Tensorflow. Sequential – это модель для построения архитектуры нейронной сети в виде последовательных слоев. Также Sequential позволяет расширять существующую модель, изменяя ее параметры и добавляя новые слои. Воспользуемся данным свойством Sequential и передадим в эту модель предобученные модели из TensorFlow Hub через URL вектора признаков.

В Keras существуют вспомогательные слои, один из них определяется классом Input. Этот слой служит для описания формы входных данных. Если модель не имеет слоя Input, то размерность входного вектора устанавливается по входному тензору при первом вызове. Но если явно указать размерность через класс Input, то модель сети строится сразу с начальным набором весов. Так как мы используем предобученные модели, необходимо явно указывать размерность входного вектора через класс Input при создании модели.

При создании модели, использующей архитектуру предобученной модели, есть параметр trainable, отвечающий за то, будет ли слой замороженным или нет. Слой можно разморозить, если данные, на которых обучалась трансферная модель, сильно отличаются от наших данных, из-за чего может быть низкая точность. Разморозка слоя может повысить точность.

Также при создании модели мы можем добавить новые слои, такие как Dropout и Dense. Рассмотрим их подробнее.

Dropout – слой, позволяющий избежать переобучения. Проходящие через него дробные значения признаков заменяются нулевыми.

Dense – полносвязный слой, слой, выходные нейроны которого связаны со всеми входными нейронами. Слой выполняет линейное преобразование [2], результатом которого является сумма произведений значений входных нейронов на матрицу весов с примененной к ней функцией активации.

$$z_i = \sigma(w_{i0} + w_{i1} * x_1 + ... + w_{im} * x_m),$$

где z_i – выходное -е значение, $x = (x_1,...,x_m)$ – входной вектор, σ – функция активации, $W = w_{ij}, i = 1,...,m, j = 1,...,m$ – матрица весов

Затем происходит компиляция модели.

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0003),
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=False),
    metrics=['accuracy', ],
    steps_per_execution=32,)
```

Компиляция модели производится с помощью метода compile. Параметры этого метода следующие:

– optimizer – оптимизатор (алгоритм, помогающий минимизировать функцию потерь). Будем используем один из самых быстрых алгоритмов: Adam. Это метод

стохастического градиентного спуска, основанный на адаптивной оценке моментов первого и второго порядка [3].

– loss – функция потерь (определяет величину штрафа за ошибочные прогнозы в процессе обучения). Наша цель — минимизировать значение этой функции. В качестве функции потерь используем CategoricalCrossentropy, которая вычисляется по формуле:

$$CCE = -\sum_{i=1}^{n}(x_i * log\,(y_i)),$$

где x_i – истинное значение, y_i – прогнозируемое значение, n – размер вектора $x_i\,(y_i)$.

– metrics – метрика, оценивающая точность модели. Будем использовать метрику Accuracy, рассчитывающуюся как отношение количества правильных прогнозов к их общему количеству

– steps_per_execution – количество пакетов данных, запускаемых во время каждого отдельного вызова скомпилированной функции [3].

Для трансферного обучения я решила выбрать следующие модели: mobilenet-v1, mobilenet-v3, inception-v3, efficientnet-v2. Результаты 20 эпох обучения данных моделей представлены в табл. 1.

Таблица 1

Результаты трансферного обучения четырех моделей

Название модели	Точность на обучающей выборке	Точность на тестовой выборке
mobilenet-v1	58%	43%
mobilenet-v3	75%	50%
inception-v3	54%	39%
efficientnet-v2	66%	47%

4. Исследование и анализ работы модели с разными параметрами

Согласно табл.1 двумя лучшими моделями оказались mobilenet-v3 и efficientnet-v2. Произведем корректировку параметров этих моделей в целях улучшения точности на обучающей и тестовой выборке.

Будем дорабатывать модель комбинированием изменения параметра trainable и добавления слоев Dropout и Dense. Результаты обучения моделей представлены в табл. 2

Таблица 2

Результаты трансферного обучения mobilenet-v3 и efficientnet-v2 с разными параметрами

Название модели	Слой Dropout	Слой Dense	Заморозка слоя	Точность на обучающей выборке	Точность на тестовой выборке
mobilenet-v3	Dropout (0.2)	-	Да	85%	58%
efficientnet-v2	Dropout (0.2)	-	Да	71%	49%
mobilenet-v3	Dropout (0.2)	Dense(512)	Да	75%	53%
efficientnet-v2	Dropout (0.2)	Dense(512)	Да	70%	48%
mobilenet-v3	-	-	Нет	82%	64%
efficientnet-v2	-	-	Нет	85%	68%
mobilenet-v3	Dropout (0.2)	-	Нет	91%	82%
efficientnet-v2	Dropout (0.2)	-	Нет	92%	84%

Исходя из табл. 2, можно сделать вывод, что при замороженном слое, вне зависимости от других параметров, точность на тестовой выборке достаточно низкая, а при разморозке слоя

она значительно возросла. Лучшая точность достигается при комбинировании размороженного слоя и слоя Dropout.

Протестируем лучшую полученную модель (efficientnet-v2 с Dropout (0.2) и размороженным слоем). Результаты предсказания модели представлены на рис. 2.

Рис.2. Результаты предсказания модели

Заключение

В ходе выполнения данной работы была построена и обучена модель глубокого обучения, распознающая и классифицирующая изображения блюд. Обучение осуществлялось методом переноса обучения с использованием четырех различных предобученных моделей. Лучшую точность показали архитектуры моделей mobilenet-v3 и efficientnet-v2. При дальнейшем обучении моделей с изменением параметров обучения и добавления новых слоев было выяснено, что точность выше, если слой будет размороженным, а лучшую точность показала модель efficientnet-v2 со слоем Dropout(0.2) и размороженным слоем. Данную модель можно использовать в дальнейшем в мобильном приложении, чтобы при распознавании изображения выводить пользователю информацию об ингредиентах блюда и о его рецепте.

Литература

1. Анирад Коул Искусственный интеллект и компьютерное зрение. Реальные проекты на Python, Keras и TensorFlow / Анирад Коул, Сиддха Ганджу, Мехер Казам – Санкт-Петербург : Питер, 2023. – 624 с.

2. Келлер А. Изучаем OpenCV 3. / А. Келлер, Г. Брэдски – Москва : ДМК Пресс, 2017. –826 с.

3. Документация библиотеки Keras. – Режим доступа: https://keras.io/. – (Дата обращения: 15.04.2024).

УДК 004.42

TCP ПРОТИВ UDP: ОБЗОР ТРАНСПОРТНЫХ ПРОТОКОЛОВ

П. Р. Губанов

Воронежский государственный университет

Введение

В конце 20-го века в эпоху зарождения сетевых технологий протоколы TCP (Transmission Control Protocol) и UDP (User Datagram Protocol) стали основными строительными блоками Интернета. Разработанные в рамках ARPANET в 1970-х годах, эти протоколы играют важнейшую роль в передаче данных между узлами сети, обеспечивая надёжность и эффективность взаимодействия.

Цель данной статьи – провести глубокий анализ и сравнение этих двух протоколов, рассматривая их особенности, преимущества и ограничения, а также их роль в современной сетевой инфраструктуре. В статье будут разобраны различные сценарии сетевой коммуникации и предложены рекомендации по выбору протокола для оптимальной передачи данных.

1. Описание протоколов

Протоколы TCP и UDP являются ключевыми компонентами сетевой коммуникации, играющими важную роль в передаче данных в современных компьютерных сетях. Далее будут представлены основные характеристики и принципы работы обоих протоколов для визуализации их роли и значения в сетевой инфраструктуре.

1.1. Протокол TCP

Протокол TCP является одним из основных протоколов транспортного уровня в семействе протоколов TCP/IP. Его основное предназначение - обеспечение надежной и упорядоченной доставки данных между узлами сети. Далее будут рассмотрены его основные характеристики:

- **Механизм установления соединения:** TCP использует механизм установления (представлен на рис. 1) и завершения соединения для обеспечения стабильной связи между отправителем и получателем. Этот процесс включает в себя обмен специальными пакетами данных, так называемый "TCP handshake", в результате чего обе стороны соглашаются на параметры соединения [3].

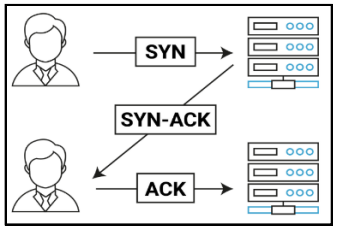


Рис. 1. Схема механизма установления соединения

- **Управление потоком:** TCP обеспечивает контроль потока данных, регулируя скорость передачи данных, чтобы избежать перегрузок на сетевом уровне. Это достигается путем использования механизмов, таких как "окна приема"

и "подтверждение приема", которые позволяют получателю контролировать скорость получения данных от отправителя.

- **Контроль ошибок:** TCP включает в себя механизмы обнаружения и исправления ошибок для обеспечения целостности данных. Это достигается путем использования контрольных сумм для проверки целостности пакетов данных и механизмов повторной передачи в случае обнаружения ошибок.

1.2. Протокол UDP

Протокол UDP является другим протоколом транспортного уровня, используемым в сетях TCP/IP. В отличие от TCP, UDP предлагает более простую и менее надежную модель доставки данных. Рассмотрим его основные характеристики:

- **Отсутствие установления соединения:** в отличие от TCP, UDP не имеет механизмов установления и поддержания соединения между отправителем и получателем. Каждый пакет данных рассматривается как отдельное сообщение, что делает UDP более легковесным и быстрым протоколом. Механизм обмена сообщениями по протоколу UDP представлен на рис. 2.

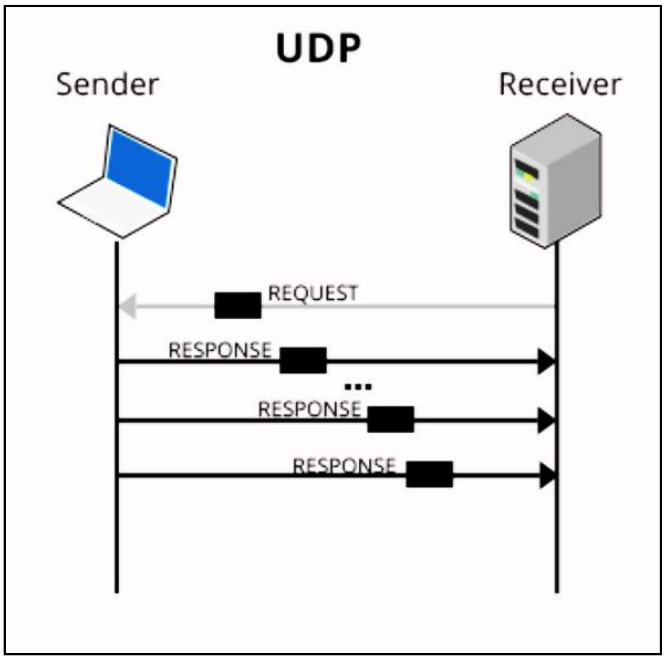


Рис. 2. Протокол UDP: механизм обмена сообщениями

- **Отсутствие контроля потока и ошибок:** UDP не предоставляет механизмов управления потоком данных и контроля ошибок. Это означает, что UDP не гарантирует порядок доставки пакетов и не обеспечивает надежность передачи данных [1].
- **Простота и эффективность:** благодаря отсутствию сложных механизмов TCP, UDP является более простым и эффективным протоколом, особенно в сценариях, где скорость и минимальная нагрузка на сеть более важны, чем надежность передачи данных.

2. Сравнение особенностей протоколов

Рассматривая различные аспекты сетевой архитектуры, был проведён сравнительный анализ особенностей протоколов TCP и UDP и выявлены их различия в надежности передачи данных, скорости и использовании ресурсов сети. Результаты анализа распределены по ключевым характеристикам и представлены далее.

2.1. Надёжность передачи данных

Одним из основных различий между TCP и UDP является подход к надежности передачи данных:

- **TCP:** TCP обеспечивает надежную доставку данных путем использования механизмов управления потоком и контроля ошибок. Это гарантирует, что данные будут доставлены в правильном порядке и без потерь. Однако такая надежность может приводить к увеличению задержек в сети из-за необходимости дополнительной обработки данных на уровне протокола.
- **UDP:** в отличие от TCP, UDP не обеспечивает гарантированной доставки данных и не включает в себя механизмы контроля ошибок. Это означает, что пакеты данных могут быть потеряны или доставлены в неправильном порядке. Однако такой подход позволяет сократить задержки и нагрузку на сеть, что делает UDP более подходящим выбором для приложений, где скорость более важна, чем надежность.

2.2. Скорость передачи данных

Другим важным аспектом сравнения TCP и UDP является скорость передачи данных:

- **TCP:** из-за механизмов управления потоком и контроля ошибок TCP может создавать дополнительную нагрузку на сеть и приводить к увеличению задержек. Это особенно заметно в условиях высоких нагрузок на сеть или при передаче больших объемов данных. Однако TCP подходит для приложений, где надежность имеет большее значение, чем скорость, таких как передача файлов или веб-серверы.
- **UDP:** благодаря отсутствию сложных механизмов управления потоком и контроля ошибок, UDP обеспечивает более высокую скорость передачи данных. Это делает UDP идеальным выбором для приложений, где скорость имеет большее значение, чем надежность, таких как потоковое видео или онлайн-игры [2].

2.3. Использование ресурсов сети

Рассмотрим использование ресурсов сети при использовании TCP и UDP:

- **TCP:** из-за механизмов управления потоком и контроля ошибок TCP может создавать дополнительную нагрузку на сеть, особенно в условиях высоких нагрузок. Это может привести к затратам большего количества ресурсов сети для обеспечения надежной передачи данных.
- **UDP:** благодаря отсутствию сложных механизмов управления потоком и контроля ошибок, UDP создает минимальную нагрузку на сеть. Это делает UDP более эффективным выбором в сценариях, где важна минимальная задержка и максимальная пропускная способность.

2.4. Использование нестабильной сети

В современных условиях многие пользователи сталкиваются с нестабильными сетевыми соединениями, такими как Wi-Fi или мобильные сети 4G. В таких сетях частые изменения маршрутизаторов и потеря пакетов могут серьезно повлиять на производительность сетевых приложений. Далее будет представлено сравнение, как протоколы TCP и UDP справляются с такими условиями:

- **TCP:** протокол ориентирован на соединение и может столкнуться с проблемами в нестабильных сетях из-за необходимости поддерживать установленное соединение и контролировать его состояние. Когда маршрутизаторы или точки доступа часто меняются, TCP может столкнуться с задержками и потерями пакетов из-за необходимости переустановки соединения.
- **UDP:** протокол не имеет установленного соединения, поэтому может лучше справляться с нестабильными сетевыми условиями. Поскольку UDP не требует поддержания состояния соединения, он более гибок и может быстрее адаптироваться к изменениям в сети. Однако, из-за отсутствия механизмов контроля ошибок и управления потоком, UDP может быть менее надежным в условиях высокой потери пакетов или перегрузки сети.

3. Применение TCP и UDP в различных сценариях

Каждый из этих протоколов имеет свои особенности, которые делают их подходящими для определенных типов сетевых приложений. Далее будут рассмотрены различные сценарии использования TCP и UDP, проанализированы их особенности и преимущества, а также приведем конкретные примеры приложений, где каждый из этих протоколов находит свое наиболее эффективное применение.

3.1. Сценарии использования TCP

Протокол TCP часто находит свое применение в сценариях, где надежность передачи данных играет решающую роль. Рассмотрим некоторые из них:

- **Передача файлов:** передача файлов с использованием TCP обеспечивает надежную и упорядоченную доставку данных. Это особенно важно, когда важно сохранить целостность и последовательность переданных данных, как это часто бывает при передаче файлов [4].
- **Веб-серверы:** они обычно используют TCP для обработки HTTP-запросов и передачи веб-страниц. Такой подход обеспечивает надежную доставку данных и минимизирует риск потери или повреждения информации, что особенно важно для передачи конфиденциальных данных пользователей. Схема применения TCP в клиент-серверной архитектуре представлена на рис. 3.

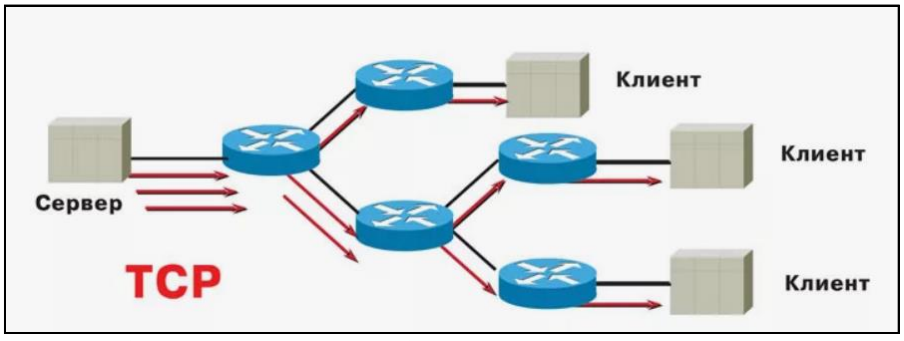


Рис. 3. Схема применения TCP в клиент-серверной архитектуре

- **Электронная почта:** почтовые серверы используют TCP для передачи электронных сообщений между почтовыми клиентами и серверами. Это гарантирует, что сообщения будут доставлены в целости и сохранности, что критически важно для обмена важной корпоративной информацией.

3.2. Сценарии использования UDP

Протокол UDP находит свое применение в сценариях, где скорость и минимальная нагрузка на сеть имеют большее значение, чем надежность передачи данных:

- **Потоковое видео:** потоковая передача видео является одним из наиболее распространенных сценариев использования UDP. Благодаря отсутствию механизмов управления потоком и контроля ошибок UDP обеспечивает более высокую скорость передачи данных и меньшую задержку, что делает его идеальным для потоковой передачи видео.

- **Онлайн-игры:** многопользовательские онлайн-игры часто используют UDP для обмена игровыми данными между сервером и клиентами. Благодаря минимальной задержке UDP обеспечивает более быструю реакцию на действия игроков, что является решающим фактором в соревновательных онлайн-играх.
- **Трансляции мультимедийного контента:** UDP также широко используется для трансляций мультимедийного контента, таких как интернет-радио или потоковое аудио/видео. В таких сценариях скорость передачи данных и минимальная задержка играют более важную роль, чем надежность передачи данных.

3.3. Примеры использования TCP и UDP

Пример 1: Передача файла через FTP (File Transfer Protocol).

FTP является протоколом, используемым для передачи файлов между клиентом и сервером. При передаче файлов через FTP, TCP обеспечивает надежную доставку данных, гарантируя, что файл будет передан и получен без потерь. Это особенно важно в случае передачи критически важной информации, такой как документы или программные файлы [5].

Пример 2: Онлайн-игра с множеством игроков.

В онлайн-играх с множеством игроков, таких как шутеры от первого лица или многопользовательские ролевые игры, UDP может использоваться для передачи игровых данных между сервером и клиентами. Это позволяет минимизировать задержку и обеспечить более плавный геймплей, что особенно важно в соревновательных играх, где реакция игроков играет ключевую роль.

Заключение

В результате проведения анализа и сравнения по различных аспектам сетевой инфраструктуры, были рассмотрены основные протоколы транспортного уровня - TCP и UDP, и их различное применение в современных сетевых приложениях. Каждый из этих протоколов имеет свои особенности, преимущества и недостатки, которые следует учитывать при выборе подходящего протокола для конкретного сценария использования – например, TCP обеспечивает надежную передачу данных, подходит для передачи файлов и веб-серверов, но UDP предлагает более высокую скорость и используется в потоковом видео, онлайн-играх.

Список литературы

1. User Datagram Protocol. Wikipedia. – URL: https://en.wikipedia.org/wiki/User_Datagram_Protocol (дата обращения: 03.02.2024).
2. TCP and UDP - what's the difference? – URL: https://wiki.merionet.ru/articles/tcp-i-udp-v-chem-raznica (дата обращения: 15.01.2024).
3. Transmission Control Protocol (TCP). – URL: https://www.techtarget.com/searchnetworking/definition/TCP (дата обращения: 20.01.2024).
4. TCP Protocol Structure. – URL: https://iteducate.github.io/networks/tcp_prot/ (дата обращения: 04.02.2024).
5. TCP and UDP, or Two Pillars of the Internet. – URL: https://habr.com/ru/articles/711578/ (дата обращения: 27.01.2024).

УДК 004.896

ИССЛЕДОВАНИЕ ЗАВИСИМОСТИ РЕЗУЛЬТАТОВ ОБУЧЕНИЕ НЕЙРОСЕТЕЙ ОТ СЛОЖНОСТИ ОБУЧАЮЩИХ ДАННЫХ НА ПРИМЕРЕ BERT И SBERQUAD

В.М. Гудков

Воронежский государственный университет

Введение

Процесс сбора и подготовки обучающих данных для моделей машинного обучения является сложным и трудоемким процессом, требующим непосредственного участия человека. От качества обучающих данных во многом зависит качество результирующей модели.

Целью данной работы является исследование зависимости результатов процесса обучения нейросетей применяемых в экстрактивных вопрос-ответных системах от состава обучающих данных с применением поиска частых подпоследовательностей для оценки повторяемости и сложности вопросов.

1. Используемые данные, архитектуры, алгоритмы

1.1. Обучающие данные

Для обучения нейросетей использовался датасет SberQuAD[1], который состоит из троек: контекст, вопрос, часть контекста, содержащая ответ на вопрос. Пример данных представлен на рисунке 1.

P6418 Термин Computer science (Компьютерная наука) появился в 1959 году в научном журнале Communications of the ACM, в котором Луи Фейн (Louis Fein) ратовал за создание Graduate School in Computer Sciences (Высшей школы в области информатики) . . . Усилия Луи Фейна, численного аналитика Джорджа Форсайта и других увенчались успехом: университеты пошли на создание программ, связанных с информатикой, начиная с Университета Пердью в 1962.

P6418 The term "computer science" appears in a 1959 article in Communications of the ACM, in which Louis Fein argues for the creation of a Graduate School in Computer Science . . . Louis Fein's efforts, and those of others such as numerical analyst George Forsythe, were rewarded: universities went on to create such departments, starting with Purdue in 1962.

Q11870 Когда впервые был применен термин Computer science (Компьютерная наука)?

Q11870 When did the term "computer science" appear?

Q28900 Кто впервые использовал этот термин?

Q28900 Who was the first to use this term?

Q30330 Начиная с каого учебного заведения стали применяться учебные программы, связанные с информатикой?

Q30330 Starting with wich university were computer science programs created?

Рис. 1. Пример данных SberQuAD

Датасет объемом в 50 тысяч экземпляров был разделен:

- 90% обучающие данные;
- 10% тестовые данные.

Дополнительно набор данных содержит 25 тысяч неразмеченных экземпляров, состоящих только из вопроса и контекста.

1.2. Архитектура нейросети

Для проведения эксперимента использовалась нейронная сеть с BERT-архитектурой[2], предварительно обученная на корпусе русскоязычного текста ruBERT-base[3].

Определение токена начала и конца части контекста, отвечающей на предоставленный вопрос, происходит по средствам двух линейных слоев размерности, равной длине входной последовательности. В результате применения SoftMax (формула (1)) к каждому выходному вектору имеем два вектора: вектор вероятностей начала и вектор вероятностей конца ответа.

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_k}} \quad (1)$$

Токеном начала или конца считается токен с наибольшей вероятностью в соответствующем векторе вероятностей (формулы (2) и (3)). Оценка результата – произведение вероятностей токенов начала и конца (формула (4)).

$$start = \arg\max(output_{start}) \quad (2)$$

$$start = \arg\max(output_{start}) \quad (3)$$

$$answerScore = output_{start}[start] \cdot output_{end}[end] \quad (4)$$

Так как архитектура BERT не подразумевает разделение вопроса и контекста, обе части объединяются через специальный токен. В случае отсутствия ответа в переданном контексте, начало и конец вопроса считаются равными и ссылаются на добавленный разделитель. Пример входных данных и возможных выводов нейронной сети представлен на рисунке 2 [4].

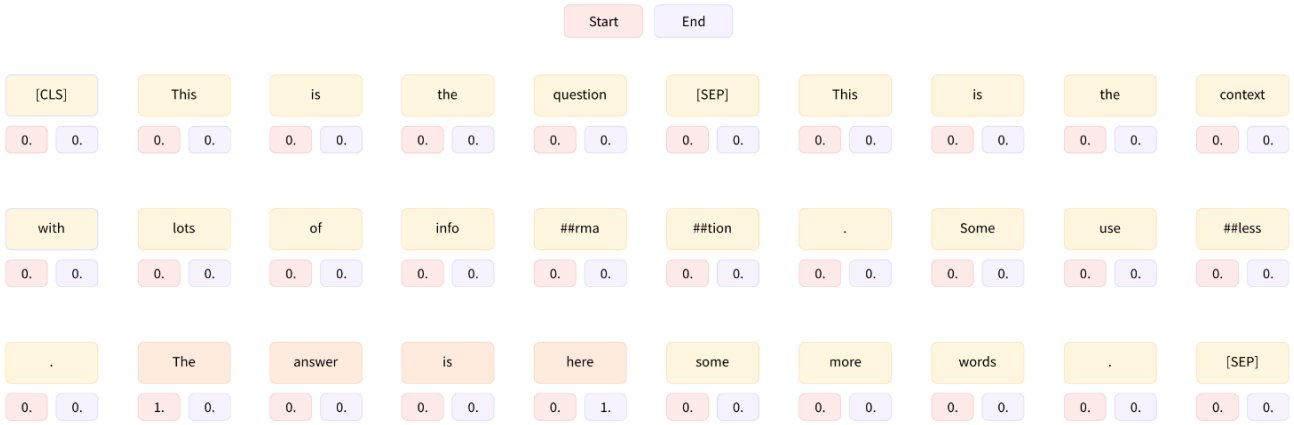


Рис. 2. Пример входных данных и выходов нейросети

Функция потерь - средняя кросс энтропия векторов начала и конца (формулы (5) – кросс энтропия и (6) – полная функция потерь).

$$crossEntropyLoss(y, x) = -\sum_{i=1}^{n} \log(y_i) x_i \quad (5)$$

$$loss = \frac{crossEntropyLoss(output_{start}, y_{start}) + crossEntropyLoss(output_{end}, y_{end})}{2} \quad (6)$$

В качестве метрик [5] используется полное совпадение (exact match, em) и f1 (формула (7)).

$$f1 = \frac{2\,precision \cdot recall}{precision + recall} \quad (7)$$

Здесь precision - число токенов верно попавших в ответ относительно всех токенов предсказанного ответа, recall – число токенов верно попавших в ответ относительно тех токенов которые действительно должны были попасть в ответ. Можно интерпретировать данную метрику как среднее пересечение правильных и предсказанных ответов.

1.3. Извлечение шаблонов

Сложность и используемость шаблонов вопросов оценивалась при помощи анализа частых последовательностей.

Так как текст вопроса является последовательностью событий, состоящих в появлении слов, из него можно извлечь наиболее часто встречающиеся подпоследовательности. Число последовательностей r из D, для которых некоторая последовательность s является подпоследовательностью $(s \subseteq r)$ называется поддержкой (support). Для удобства использования получившееся число стоит разделить на общее число последовательностей для получения значения поддержки в виде доли от общего числа вопросов (формула (8)).

В контексте текстов подпоследовательность является шаблоном и паттерном текста.

$$support(s, D) = \frac{\left|\{r : r \in D, s \subseteq r\}\right|}{|D|} \quad (8)$$

Расчет количества шаблонов вопросов длинной больше 2, советующих значениям поддержки от 0.001 до 0.02, представлен на рисунке 3.

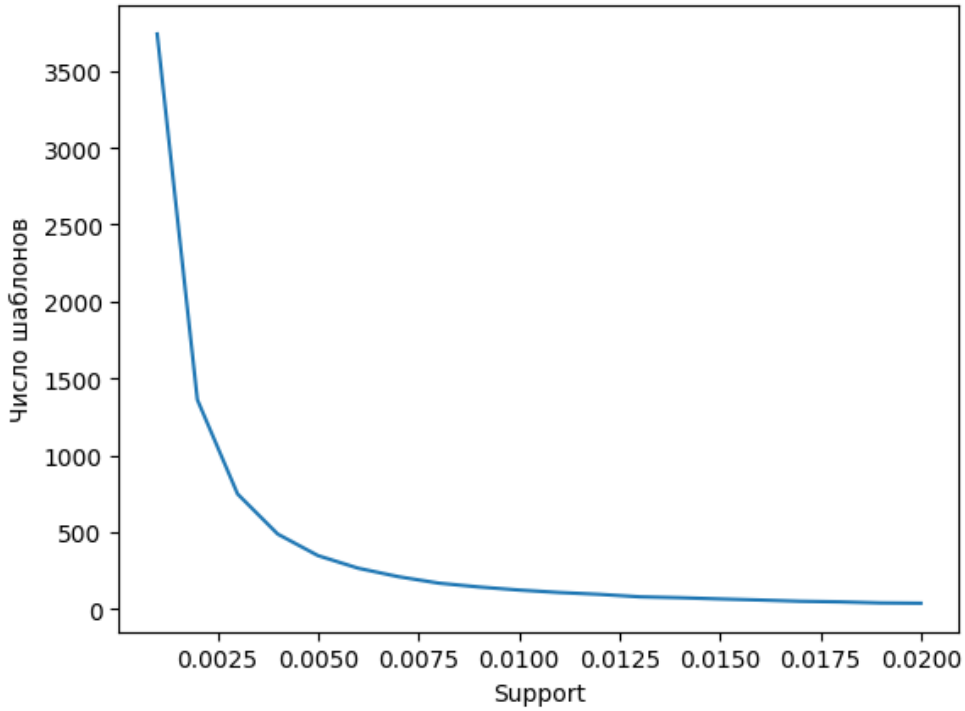


Рис. 3. Зависимость числа шаблонов от значения поддержки

В таблице 1 представлены отношения числа вопросов, соответствующих шаблонам с заданной поддержкой и длинной, к общему числу вопросов.

Таблица 1

Соотношения параметров шаблонов с числом совпадений

Поддержка	Минимальная длинная	% от общего числа вопросов
0.001	2	91
	3	54
	4	21
0.005	2	76
	3	37
	4	6
0.01	2	66
	3	20
	4	6
0.02	2	48
	3	17
	4	0

2. Вычислительный эксперимент

2.1. Описание и результат

Для проведения экспериментов данные были разделены на соответствующие выделенным шаблонам и несоответствующие. Всего было получено 22 различных набора, использованных для обучения нейросети.

Качество полученной модели оценивалось по метрикам em и f1 на тестовой выборке (не задействованной в процессе выделение шаблонов) и на части данных оригинального набора

данных, не попавшей в результате фильтрации по шаблону. Дополнительная проверка на неиспользованных данных контролирует сохранение предсказательной способности на исключенных вопросах.

Дополнительно рассчитывается отношение f1 к объему использованных данных (9).

$$score_i = \frac{f1_i - \min\limits_{k\in\{1..n\}}(f1_k)}{size_i} \qquad (9)$$

Таким образом, наборы данных при использовании которых модель показывает лучшие результаты при меньшем объеме самого набора будут иметь б*о*льший score. Стоит заметить, что шкала оценок score является порядковой.

Результаты тестирования нейросетей, обученных на данных, советующих шаблонам, представлены в таблице 2.

Таблица 2

Результаты обучения на данных, соответствующих шаблонам

support	len	positive					
		size	validation		remaining		score
			em	f1	em	f1	
0,001	2	0,91	**60,7**	**80,9**	60,5	78,7	0,36
	3	0,54	**60,4**	**80,6**	58,4	78,1	0,6
	4	0,21	**53,7**	**74,7**	51,4	71,9	***1,28***
0,005	2	0,76	**60,4**	**80,9**	59,2	78,7	0,43
	3	0,37	**58**	**78,9**	56,5	76,9	***0,84***
	4	0,06	**27,7**	**49,6**	25,8	46,5	0,3
0,01	2	0,66	**59,3**	**79,8**	57,7	77,6	0,48
	3	0,2	**52,7**	**74,1**	50,9	71,6	***1,31***
	4	0,06	**26,5**	**47,8**	25,1	45,2	0
0,02	2	0,48	**59**	**79,2**	57,6	77,2	0,65
	3	0,18	**45,4**	**67,1**	42,9	63	***1,07***
	4	0					

Результаты тестирования нейросетей, обученных на данных, не советующих шаблонам, представлены в таблице 3.

Таблица 3

Результаты обучения на данных, не соответствующих шаблонам

support	len	negative					
		size	validation		remaining		score
			em	f1	em	f1	
0,001	2	0,09	**43,4**	**64,2**	41,45	64,2	0
	3	0,46	**59,6**	**79,7**	57,9	79	***1,73***
	4	0,79	**59,9**	**80,1**	55,7	79	1,01
0,005	2	0,24	**57,6**	**77,9**	55,6	76,7	***3,24***
	3	0,63	**60**	**80,2**	58,54	79,7	1,27
	4	0,94	**61,2**	**81**	62,4	84	0,86
0,01	2	0,34	**59,2**	**79,2**	57,6	78,4	***2,32***

	3	0,8	**60,2**	**80,1**	56,3	79,3	1,001
	4	0,94	**61,2**	**81**	62,4	84	0,86
0,02	2	0,52	**60**	**79,8**	57,5	78,8	***1,53***
	3	0,82	**60,1**	**80,4**	55,5	79,2	0,98
	4	1					

2.2. Интерпретация результатов

Как можно заметить, все результаты, попавшие в топ 4, лучше на данных, исключающих частые шаблоны. Дополнительно, для варианта с исключением лучше избегать коротких шаблонов, а для вариантов с включением использовать длинные шаблоны.

При этом не обнаружено улучшение качества от уменьшения количества данных, т.е. одновременное использование частых и редких или простых и сложных шаблонов не препятствует обучению.

Лучшие показатели получены на полном наборе данных.

Заключение

В результате исследования можно сделать вывод о том, что наибольший вклад в успешное обучение вносят неповторяющиеся вопросы нетривиального строения.

Как следствие, при сборе данных стоит обращать внимание на структуру вопросов, и в случае возникновения необходимости ручного создания обучающих данных, фокусироваться на разнообразии, а не на объеме.

Литература

1. SberQuAD – Russian Reading Comprehension Dataset: Description and Analysis / Efimov Pavel [и др.] // Experimental IR Meets Multilinguality, Multimodality, and Interaction – Springer International Publishing, 2020 – С. 3-15.

2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Электронный ресурс] / Jacob Devlin [и др.] // arXiv. – 2019. – Режим доступа: https://arxiv.org/abs/1810.04805 - (Дата обращения: 06.04.2024).

3. A Family of Pretrained Transformer Language Models for Russian [Электронный ресурс] / Dmitry Zmitrovich [и др.] // arXiv. – 2023. – Режим доступа: https://arxiv.org/abs/2309.10931 - (Дата обращения: 09.04.2024).

4. Question answering - Режим доступа: https://arxiv.org/abs/2309.10931 - (Дата обращения: 09.04.2024).

5. SQuAD: 100,000+ Questions for Machine Comprehension of Text [Электронный ресурс] / Pranav Rajpurkar [и др.] // arXiv. – 2016. – Режим доступа: https://arxiv.org/abs/1606.05250 - (Дата обращения: 19.04.2024).

УДК 004.942

ИССЛЕДОВАНИЕ МЕХАНИЗМОВ ИСПОЛЬЗОВАНИЯ БИБЛИОТЕКИ PYO В ПРОЦЕССЕ ПОСЛЕДОВАТЕЛЬНОГО НАЛОЖЕНИЯ ЦИФРОВЫХ ЭФФЕКТОВ НА ЗВУКОВОЙ ФАЙЛ В ФОРМАТЕ WAV

Д. И. Дроздова, М. К. Чернышов

Воронежский государственный университет

Введение

В современном мире музыки огромную роль играют программы для изменения голоса, добавления различных эффектов в аудиозапись. Все аудиозаписи артистов корректируются и изменяются с использованием различных эффектов: эхо, реверс, эффекта многоголосия, фильтров и корректировки аудио частот, добавления хрипа. Множество микшерных пультов содержит в себе программы, позволяющие накладывать один эффект звука на другой, тем самым смешивая звуковые потоки в реальном времени и воспроизводя измененный звук.

В данной работе рассматривается один из подходов к реализации механизмов наложения звуковых эффектов на оцифрованный сигнал, представленный в виде файла формата WAV. В качестве инструментария выбрана библиотека PYO, представляющая собой модуль Python, написанный на языке программирования C для создания сценария цифровой обработки сигналов. Данный модуль Python содержит классы для обработки разнообразных типов аудиосигналов. Библиотека PYO содержит в себе различные примитивы, включающие в себя математические операции, базовую обработку сигналов (задержки, генераторы синтеза, фильтрации) и многое другое

1. Основные понятия и используемая терминология

На рис. 1 представлена структура библиотеки PYO.
Описанный в работе механизм последовательного наложения звуковых эффектов относится к области «audio processing», изображенной на схеме.

Чтобы начать работу с библиотекой PYO, нужно создать и запустить **сервер**. Сервер в программе – это экземпляр класса Server, который запускает аудио- и MIDI-интерфейсы для работы со звуком. При создании сервера можно указать размер буфера, больший чем тот, который установлен по умолчанию, для того чтобы избежать задержек при работе со звуковыми потоками. Также можно указать частоту дискретизации звука для улучшения качества обрабатываемых аудио данных.

В библиотеке PYO звуковые объекты, моделирующие оцифрованные звуковые сигналы, называются **источниками**. Аудио объекты, изменяющие звук, называются **процессами:** они способны сами возвращать новые аудио объекты. Процессы могут изменять как источники, так и процессы. Аудио объекты отправляются на **приемник** для вывода, которым может быть физическое аудиоустройство или один из каналов динамика.

Большинство классов в библиотеке является наследниками класса pyoObject. Это удобно при реализации различных функций, которые могут принимать в качестве параметров объекты, унаследованные от pyoObject, выполняя при этом над ними одни и те же операции. Данная возможность активно применяется в работе.

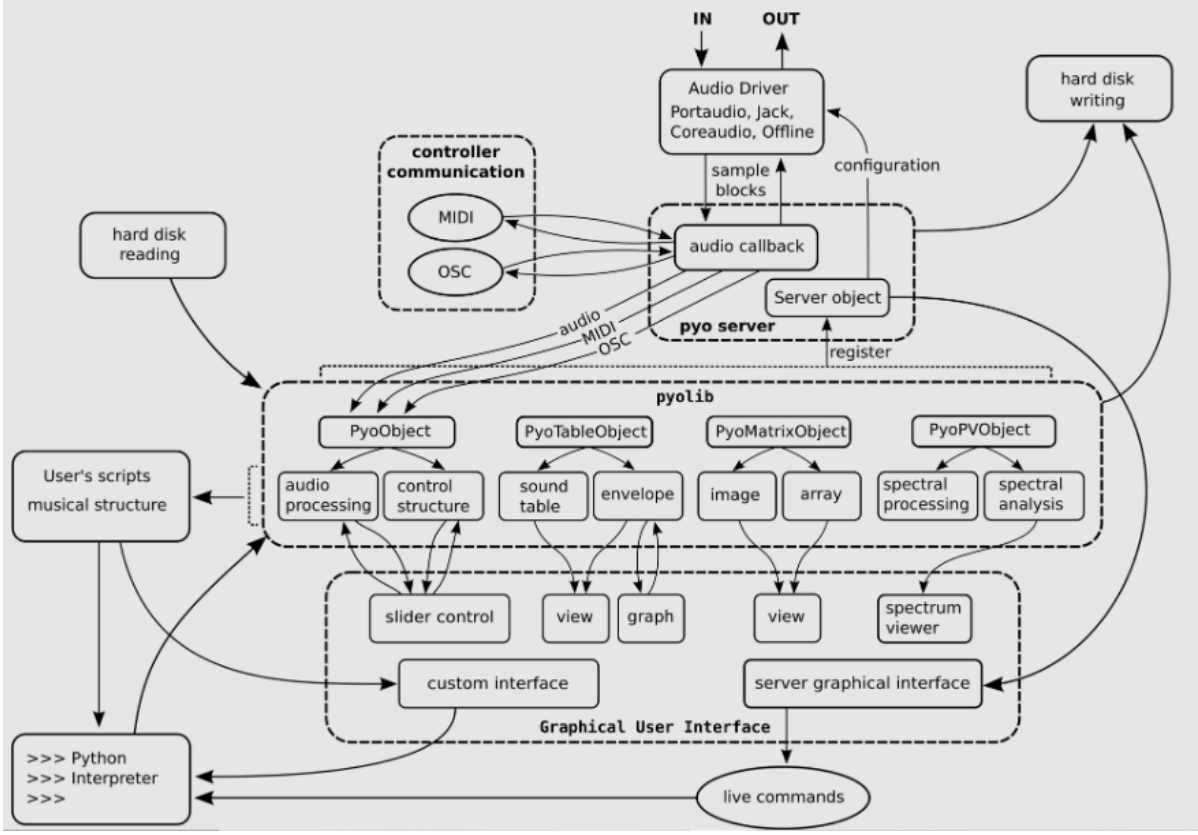


Рис. 1. Структура библиотеки PYO

2. Механизм последовательного применения нескольких эффектов к аудио данным

На рис.2 изображена разработанная на основе документации схема прохождения звука через устройства, реализующие различные эффекты. Механизм последовательного прохождения звука через каждое устройство позволяет применять эффекты одновременно.

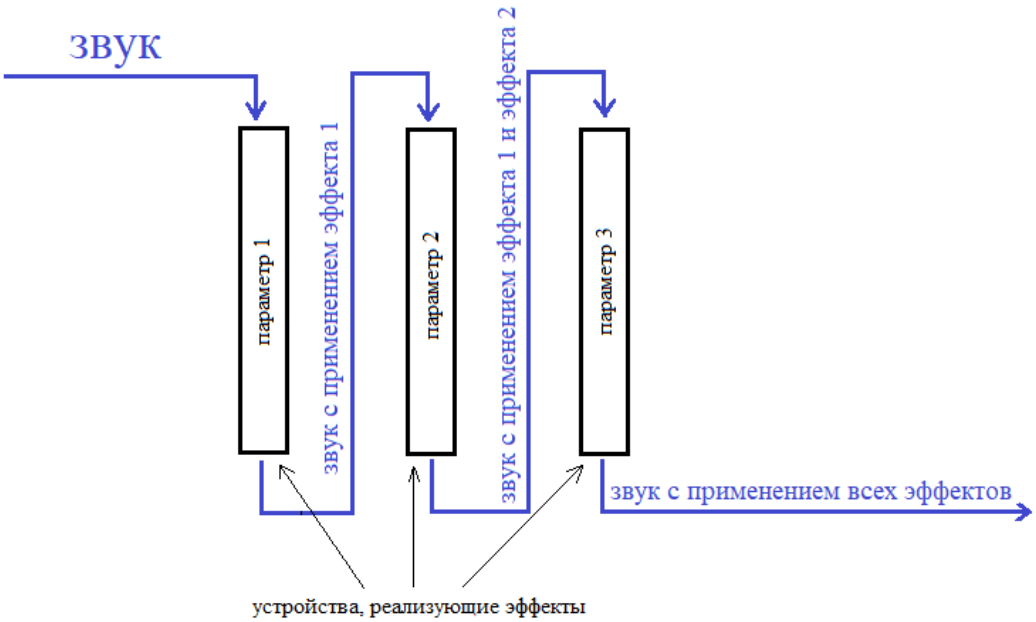


Рис. 2. Схема применения к звуку нескольких эффектов одновременно

В каждом устройстве есть один или несколько параметров, которые отвечают за степень изменения звука. Например, параметр 1 отвечает за степень изменения звука для эффекта 1. У каждого параметра существует такое значение, при котором звук не изменится, поэтому если требуется добавить к звуку, например, только эффект 3, то нужно для параметров 1 и 2 установить значения, при которых применение эффекта 1 и эффекта 2 к исходному сигналу на звук никак не повлияет. В итоге на выходе будет получен звук с применением только эффекта 3 несмотря на то, что сигнал был пропущен через все три устройства. Такой механизм позволяет сделать применение каждого из эффектов независящим от применения остальных.

В процессе реализации, описанным ниже в данной статье, устройства, изображенные на рис. 2, будут создаваться с помощью объектов классов Harmonizer, Delay, Freeverb, Chorus, FreqShift библиотеки PYO.

3. Реализация процедуры наложения аудио эффектов

На рис. 3 продемонстрирован фрагмента исходного кода, описывающего процесс реализации наложения друг на друга различных аудио эффектов.

SfPlayer – класс библиотеки PYO. Он считывает аудио данные из файла формата WAV, путь к которому передается в параметр path. Класс заботится о преобразовании частоты дискретизации аудиофайла в соответствии с настройкой частоты дискретизации сервера. В итоге в self.a хранятся исходные аудио данные. После этого создается объект hr класса Harmonizer, конструктор которого принимает на вход аудио данные. Затем создается объект d класса Delay, и конструктор этого класса принимает уже не self.a (исходные аудио данные), а self.hr (аудио данные, которые были пропущены через гармонизатор). Таким образом, каждый следующий объект создается на основе предыдущего объекта, то есть на основе исходных аудио данных, которые были пропущены через все «фильтры», расположенные перед создаваемым объектом. В частности, последний объект self.sh класса FreqShift создается на основе предыдущего объекта self.chor класса Chorus.

```
def playing_file(self):
    self.a = pyo.SfPlayer(path=self.path_playing_file, speed=[1, 1])
    self.hr = Harmonizer(self.a)
    self.hr.setTranspo(0)
    self.d = Delay(self.hr, feedback=0)
    self.d.setFeedback(0)
    self.rev = Freeverb(self.d, size=0, bal=0.5)
    self.chor = Chorus(self.rev, depth=[1.5, 1.6], feedback=0.5, bal=0.5)
    self.sh = FreqShift(self.chor).out()
    self.sh.setShift(0)
```

Рис.3. Реализация наложения аудио эффектов

В итоге описанный механизм последовательного создания объектов устроен таким образом, что каждый объект в действительности создается на основе предыдущего, что позволяет реализовать механизм одновременного добавления нескольких эффектов к аудиозаписи.

Метод out(), который посылает на выход аудио данные, применяется только к самому последнему объекту, потому что последний объект уже «содержит» в себе все предыдущие. Отметим, что если бы метод out() применялся ко всем объектам, получилось бы так, что на выход подавалось бы параллельно сразу несколько экземпляров аудио данных. Это, в свою очередь, приводит к появлению всевозможных аудио помех, что недопустимо.

Заключение

Итогом данной работы является созданное на основе библиотеки PYO приложение, позволяющее реализовать механизм одновременного наложения в режиме реального времени различных цифровых эффектов на аудио данные, представленные в виде файла формата WAV.

Литература

1. Документация к библиотеке PYO. – Режим доступа: https://belangeo.github.io/pyo/
2. Загуменнов, А. П. Компьютерная обработка звука. Полное руководство / А. П. Загуменнов – Москва: ДМК Пресс, 2000. – 384 с.
3. Кинтцель, Т. Руководство программиста по работе со звуком / Т. Кинтцель – Москва: ДМК Пресс, 2000. – 432 с.

УДК: 004.942

МОДЕЛЬ SID РАСПРОСТРАНЕНИЯ ИНФЕКЦИЙ С ДИНАМИЧЕСКИМ РЕГУЛИРОВАНИЕМ ЧИСЛЕННОСТИ ПОПУЛЯЦИИ

И.А. Дудкин

Воронежский государственный университет

Введение

Компартментальные модели широко используются в математическом моделировании инфекционных заболеваний. В этих моделях население разделяется на группы, или компартменты, с определенными ярлыками, такими как, например, S, I, R, обозначающими уязвимые (Susceptible), инфицированные (Infected) и восстановившиеся (Recovered) люди соответственно. Люди могут переходить из одного компартмента в другой, что отражает прогресс заболевания.

Компартментальные модели широко используются в различных областях науки, включая эпидемиологию, экологию, фармакокинетику, океанологию и другие. Актуальность этих моделей обусловлена их способностью выявлять закономерности и механизмы формирования неоднородности.

В эпидемиологии компартментальные модели используются для прогнозирования развития эпидемических процессов и оценки эффективности мер по сдерживанию эпидемии. Они могут быть использованы для моделирования распространения вирусной инфекции, такой как COVID-19, и для получения начальных данных и параметров модели, которые могут помочь в принятии решений о предотвращении или выходе из критической ситуации.

Компартментальные модели были разработаны в начале 20 века и использовались для изучения распространения малярии и других инфекционных заболеваний. Одними из первых работ в этой области были работы Росса в 1916 году, Росса и Хадсона в 1917 году, Кермака и Маккендрика в 1927 году и Кендалла в 1956 году. Модель Рида – Фроста также была значимой, но часто пренебрегаемой, предшественницей современных эпидемиологических моделей.

Существует два основных типа компартментальных моделей: детерминистические и стохастические. Детерминистические модели используют обычные дифференциальные уравнения для предсказания распространения заболевания, в то время как стохастические модели используют случайные величины для более реалистичного, но более сложного в анализе представления.

1. Модель SID

1.1. Система дифференциальных уравнений модели SID

На основе простейшей модели SIR была разработана модель SID. Причиной для разработки этой модели стало желание рассмотреть более реалистичную ситуацию. Модель SIR не подразумевает, что люди, заразившиеся инфекцией, могут умереть, что, как показал опыт с COVID-19, не отражает реальное положение дел в некоторых ситуациях.

Именно поэтому было принято решение создать модель SID, компартменты которой расшифровываются как:

S: Количество людей, подверженных заражению. Когда встречаются восприимчивое и инфицированное лица, восприимчивое лицо заражается и переходит в компартмент инфицированных.

I: Количество инфицированных. Это люди, которые уже заразились и могут заразить других.

D: Количество людей, которые умерли после заражения инфекцией.

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta SI}{N} + \gamma I \\ \frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I - \sigma I \\ \frac{dD}{dt} = \sigma I \end{cases}$$

В данной системе β – шанс заражения, γ – шанс выздоровления, и σ – шанс смерти, а $N = S + I + D$ – есть сумма всех компартментов или же общее количество людей в модели.

Рассмотрим подробнее каждое уравнение системы. В первом уравнении наблюдается изменение компартмента S, люди могут заболевать с шансом β, уменьшая количество восприимчивых, и люди могут выздоравливать с шансом γ, увеличивая количество восприимчивых.

Второе уравнение системы говорит, что люди могут заболевать, увеличивая количество инфицированных I с шансом β, как уже было сказано ранее. Люди могут выздоравливать с шансом γ, уменьшая количество инфицированных и люди могут умирать с шансом σ.

Число мертвых зависит от числа инфицированных и шанса смерти σ.

1.2. Выводы о модели

Несмотря на то, что модель является достаточно простой, уже на этом этапе она позволяет оценить масштабы эпидемии при наличии статистики болезни и спрогнозировать потери.

2. Решение системы уравнений

Для решения системы уравнений использовалась собственная реализация численного метода Рунге-Кутта 4-го порядка точности.

Этот метод используется для решения задачи Коши для системы обыкновенных дифференциальных уравнений первого порядка. Он основан на приближенном вычислении значений функции в последующих точках с помощью итерационной формулы, которая вычисляется в четырех стадиях. В каждой стадии вычисляется значение функции в промежуточной точке с помощью разделенных разностей второго, третьего и четвертого порядков.

Данный метод был выбран не случайно, выбор был сделан на основе следующих критериев:

1. Высокая точность: Метод Рунге-Кутта 4-го порядка обеспечивает более точные результаты на каждом шаге интегрирования по сравнению с методами низших порядков точности.
2. Эффективность: Этот метод является одним из наиболее эффективных и широко используемых при численном решении дифференциальных уравнений. Его простота и

эффективность делают его популярным среди специалистов в области численного моделирования.

3. Стабильность: Метод Рунге-Кутта 4-го порядка обладает хорошей устойчивостью при численном интегрировании, что позволяет избежать основных проблем, связанных с неустойчивостью в процессе решения дифференциальных уравнений.

Особенностью реализации стала возможность решать задачу Коши для любого количества дифференциальных уравнений при соответствующем количестве начальных условий.

3. Разработка приложения с графическим интерфейсом

Следующей задачей в текущей работе стала реализация приложения с графическим интерфейсом пользователя.
Графический интерфейс пользователя (GUI) - способ взаимодействия пользователя с компьютером и другими устройствами, который использует графические элементы, такие как кнопки, меню, окна и иконки. Он позволяет пользователю управлять системой с помощью простых и интуитивно понятных операций.
Графический интерфейс пользователя может быть реализован в виде окна с различными элементами меню, которыми можно управлять с помощью мыши, стилуса или касаний.
Для разработки GUI приложения был выбран фреймворк Qt, а конкретно его языковая привязка PyQt. Выбор в пользу Qt был сделан, исходя из его следующих преимуществ

1. Кроссплатформенность: Qt позволяет создавать приложения, которые могут работать на различных операционных системах без изменений в исходном коде. Это делает разработку универсальных приложений более эффективной и экономичной по ресурсам и времени.
2. Обширная функциональность: Qt предоставляет разработчикам широкий набор инструментов и библиотек для создания разнообразных приложений, включая графические интерфейсы, работу с сетью, базами данных и многое другое. Это позволяет ускорить процесс разработки и повысить качество готового продукта.
3. Простота использования: Qt обладает интуитивно понятным API и хорошей документацией, что делает процесс разработки более простым и понятным для разработчиков.
4. Активное сообщество: Qt имеет большое и активное сообщество разработчиков, которые поддерживают и развивают этот фреймворк. Это обеспечивает доступ к обширным ресурсам, помощь в решении проблем и обмен опытом.
5. Гибкость и расширяемость: Qt предоставляет возможность создавать собственные компоненты и расширять функциональность фреймворка под конкретные потребности проекта. Это позволяет разработчикам создавать уникальные и инновационные приложения, адаптированные под конкретные требования и задачи.

4. Интерфейс приложения

Рассмотрим интерфейс приложения (рис. 1, 2).

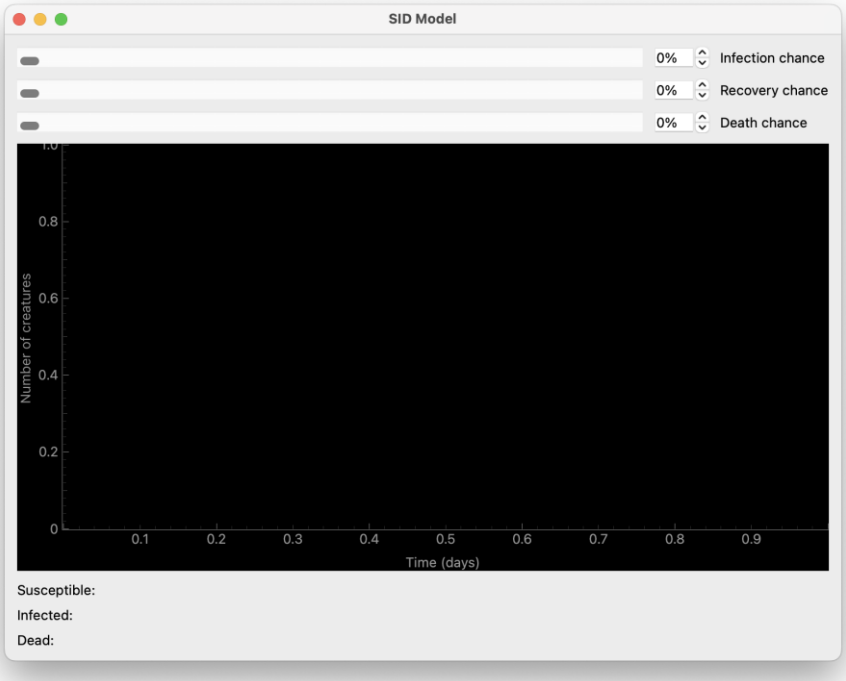


Рис. 1. Неактивное окно приложения

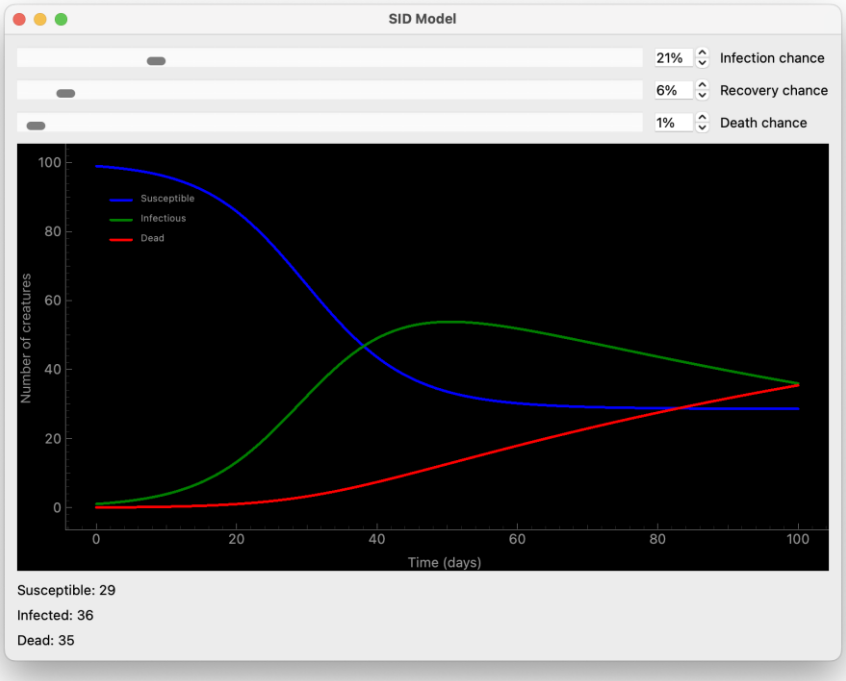


Рис. 2. Активное окно приложения

В верхней части приложения располагаются слайдеры и счетчики для регулирования вероятностей инфицирования, восстановления и смерти. В нижней части располагаются текстовые поля, в которые выводятся финальные результаты моделирования.

В центре окна приложения располагается виджет для построения графиков, в котором рисуются три цветных линии, каждая из которых наглядно иллюстрирует работу модели и

количество людей в разных точках времени. Также на графике отображается легенда, которая информирует пользователя о том, какому компартменту соответствует та или иная линия.

Заключение

Разработка приложения для модели SID требует тщательного планирования и реализации, а также глубокого понимания математических и вычислительных принципов. Это позволяет создать мощный инструмент для изучения и анализа эпидемиологических процессов.

Автор выражает благодарность научному руководителю Половинкину И.П. за постановку задачи и внимание к работе.

Список использованных источников

1. A stochastic SIRS epidemic model with infectious force under intervention strategies. – Режим доступа: https://www.sciencedirect.com/science/article/pii/S0022039615004271 – (Дата обращения: 12.02.2024).

2. An SIRS model with nonmonotone incidence and saturated treatment in a changing environment – Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9392446/ – (Дата обращения: 13.02.2024).

3. Layered SIRS model of information spread in complex networks – Режим доступа: https://www.sciencedirect.com/science/article/pii/S0096300321006135 – (Дата обращения: 13.02.2024)

4. An SIRS Epidemic Model Supervised by a Control System for Vaccination and Treatment Actions Which Involve First-Order Dynamics and Vaccination of Newborns – Режим доступа: https://www.mdpi.com/2227-7390/10/1/36 – (Дата обращения: 13.02.2024)

УДК *004.4*

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ЗООМАГАЗИНА

Е. А. Евтеева

Воронежский государственный университет

Введение

Популярность мобильных устройств возрастает с каждым днём и из-за этого в современном мире большинству организаций для поддержания успешной конкуренции могут потребоваться мобильные приложения, которыми можно привлечь дополнительных клиентов. Существующие решения не всегда универсальны и некоторые элементы приложения не всегда соответствуют уникальным требованиям конкретных организаций или нуждаются в существенных доработках, что порой не выгодно. Кроме того, возникает проблема интеграции новых функций в существующие системы. Соответственно лучшим выходом является собственный уникальный продукт, разработанный для конкретных целей и задач. Из чего следует, что в условиях конкурентного рынка зооиндустрии и ветеринарных услуг критически важно иметь эффективные инструменты для привлечения и удержания клиентов. Мобильное приложение, разработанное для зоомагазинов, может стать хорошим решением для удовлетворения этой потребности.

Целью данной работы является разработка мобильного приложения для предоставления информации о типе животных, породах, типах и схемах питания, подходящих товарах и предлагаемых услугах, с целью выбора наиболее подходящих и эффективных подходов к клиенту. Для достижения цели были изучены и проанализированы различные методы разработки клиент-серверных приложений.

1. Анализ задачи

1.1. Функциональные требования

На этапе проектирования была выделена роль для авторизованного пользователя. У каждого пользователя имеется доступ к экрану со списком товаров, а также краткое описание животных, пород и советов по дрессировке. Эти данные отображаются на главной странице приложения, что обеспечивает удобство и быстроту доступа к ним.

Приложение предоставляет пользователям следующие функции: просмотр информации о товарах и услугах зоомагазина; тест для подбора породы питомца на основе предпочтений пользователя; краткая информация о различных породах домашних животных; советы по дрессировке и воспитанию собак; контакты ветеринарных клиник, аптек и зоомагазинов.

На рис.1 представлена диаграмма активности, которая отображает все возможности пользователя:

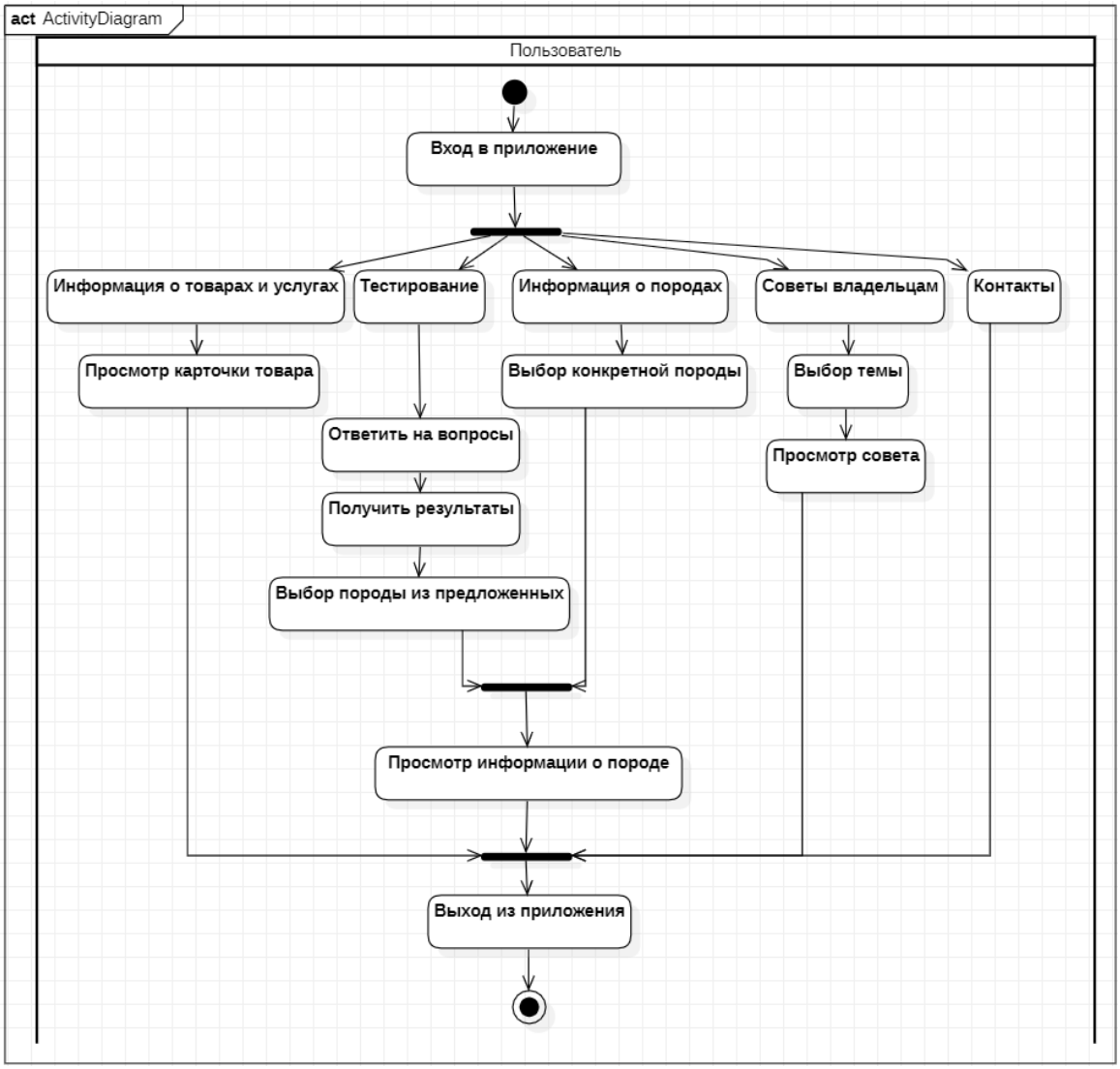


Рис. 1. Диаграмма активности

1.2. Используемые технологии

Приложение разработано в среде Xcode. В качестве языка программирования выбран Swift. Также используется облачная платформа Firebase для хранения информации.

Xcode - это мощная и удобная среду разработки для создания мобильных приложений для iOS.

Swift - это современный и высокопроизводительный язык программирования, который позволяет разработчикам создавать надежные и эффективные приложения.

Firebase - это платформа для разработки мобильных приложений от компании Google, в которой есть самые современные функции для разработки, перекомпоновки и улучшения приложений. Firebase предоставляет набор облачных инструментов, которые упрощают разработку и развертывание мобильных и веб-приложений, готовую базу данных, функции аутентификации и другие важные сервисы [1].

Комбинация этих технологий позволит создать мобильное приложение для зоомагазинов, которое будет надежным, удобным в использовании и масштабируемым для удовлетворения потребностей растущего бизнеса.

Далее описаны преимущества использования Firebase:

- Firebase — эффективный инструмент для разработки приложений, позволяющий сократить время разработки и ускорить выход приложений на рынок. Он предоставляет доступ к множеству готовых сервисов, что помогает разработчикам избегать написания повторяющегося кода и создания бэкенда с нуля;

- Firebase предлагает архитектуру без серверов. В отличие от традиционного сервера, который должен работать непрерывно, Firebase активирует серверы только при необходимости, оптимизируя использование ресурсов. Таким образом, клиенты платят только за фактическое использование серверов, что снижает проблемы масштабирования и повышает эффективность среды [2];

- приложения защищены от потенциальной потери данных благодаря возможности автоматического резервного копирования, доступной на платформе Firebase.

2. Реализация

Разработанное приложение является клиент-серверным, из этого следует, что оно состоит из двух основных компонентов: клиента и сервера.

Принцип работы следующий:

Запрос: клиент отправляет запрос на сервер. Это может быть запрос на получение данных, обновление информации или выполнение какой-либо операции.

Обработка: сервер получает запрос, обрабатывает его — например, извлекает данные из базы данных или выполняет какую-либо логику.

Ответ: после обработки запроса сервер отправляет ответ обратно клиенту. Клиент интерпретирует этот ответ и предоставляет соответствующую информацию пользователю или выполняет соответствующее действие [3].

2.1. Реализация клиентской части

Клиентская часть приложения содержит в себе интерфейс для взаимодействия с пользователем, включая экраны входа и регистрации, формы ввода данных, кнопки и элементы навигации, а также отображение данных в реальном времени. Оно также обрабатывает локальные данные, такие как кэш и настройки пользователя. Для реализации этих деталей используются инструменты Firebase. Например, Firebase Authentication позволяет пользователям создавать учётные записи, входить в систему и выходить из неё, хранит информацию о профилях пользователей; Firebase Realtime Database синхронизирует данные между клиентами в реальном времени, позволяя считывать, записывать и прослушивать изменения данных на сервере; Firebase Storage хранит изображения, видео и другие файлы на серверах Google Cloud Storage, позволяя клиентам загружать, скачивать и удалять файлы из

хранилища [4].

2.2. Реализация серверной части

Серверная часть приложения отвечает за хранение и обработку данных, бизнес-логику и взаимодействие с клиентом. Основные компоненты серверной части включают:

- Google Cloud Functions — бессерверные функции, использующиеся для обработки логики на стороне сервера, например, создания новых пользователей, отправки уведомлений и обработки платежей;

- Google Cloud Storage — служба долгосрочного хранения файлов, которая обеспечивает безопасное и масштабируемое хранилище для пользовательских данных, генерируемых приложением;

- Google Cloud Pub/Sub — сервис очередей сообщений для отправки и получения сообщений между различными компонентами системы;

- база данных NoSQL — долговременное хранилище для пользовательских данных, метаданных и другой информации, необходимой для приложения.

Серверная часть разрабатывается с акцентом на масштабируемость и надёжность. Возможности Google Cloud позволяют автоматически адаптироваться к нагрузке, а сервисы Google Cloud Storage и Pub/Sub гарантируют высокий уровень доступности и обеспечивают корректность передачи сообщений.

Использование вышеперечисленных служб Firebase повышает эффективность взаимодействия клиентской и серверной частей приложения в режиме реального времени. Например, серверная часть будет эффективно обрабатывать запросы клиентов, управлять данными и обеспечивать бесперебойную работу мобильного приложения. В свою очередь клиентская часть может отправлять запросы на сервер, получать данные и обрабатывать события в реальном времени.

Для реализации мобильного приложения зоомагазина в качестве исходных данных могут быть использованы:

- данные о товарах: цена, тип животного, тип товара, категория цены;

- сведения о животном: тип, название, краткая характеристика;

- информация и текст советов по дрессировке, питанию и оказанию первой помощи;

- вопросы к тестированию.

Исходя из этого, можно составить диаграмму классов, которая отражает структуру данных приложения (рис. 2).

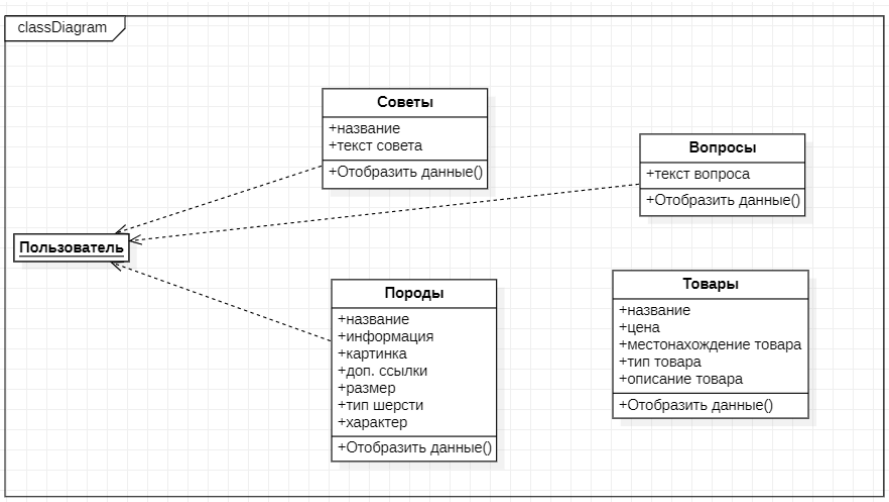


Рис. 2. Диаграмма классов

Заключение

В заключении можно сказать, что разработка представленного мобильного приложения для зоомагазинов позволяет автоматизировать работу зоомагазина, повысить уровень удовлетворённости клиентов и привлечь новых покупателей. Использование представленных инструментов Firebase и сервисов Google Cloud обеспечивает эффективное взаимодействие клиентской и серверной частей приложения, а также гарантирует надёжность и масштабируемость системы.

Литература

1. Что такое Firebase? Раскрываем все тайны. Режим доступа: https://blog.back4app.com/ru/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-firebase/#nbspFirebase-2. –(Дата обращения: 17.04.2024).

2. Мархакшинов А. Л. Разработка бессерверных мобильных приложений / А. Л. Мархакшинов, А. А. Тонхоноева, Е. Р. Урмакшинова // Вестник НГУ. -2019. -Т.17, №4. -С. 66-73.

3. Мобильная разработка. Интернет и клиент-серверные приложения. Режим доступа: https://dev.to/mainarthur/mobilnaia-razrabotka-kak-rabotaiet-intierniet-i-skhiema-raboty-kliient-sierviernykh-prilozhienii-37k1. –(Дата обращения: 19.04.2024).

4. Хуссейн, К. Mastering Firebase / К. Хуссейн. – Иркутск: ITSumma Press, 2023. - 496 с.

УДК 656.212

ОПТИМИЗАЦИЯ РАБОТЫ ГРУЗОВЫХ ТЕРМИНАЛОВ

В. А. Еремеева

Воронежский государственный университет

Введение

Грузовой терминал – это совокупность зданий, оборудования, персонала и технологических систем, скоординированных для проведения различных логистических операций. В его задачи входит прием, загрузка-выгрузка, временное хранение, сортировка и обработка грузов, а также предоставление коммерческих и информационных услуг грузополучателям, перевозчикам и другим участникам логистического процесса в различных видах транспорта, включая одиночные, многомодальные, интермодальные и прочие перевозки [1].

1. Особенность работы грузовых терминалов

Современные терминалы не ограничиваются простыми складскими помещениями для небольших грузов; они также выступают в роли ключевых центров по распределению товаров и баз для обеспечения поставок. В логистических цепях производителей они приобретают все большее значение и становятся неотъемлемыми звеньями.

Универсальные терминалы в основном занимаются обработкой небольших грузовых отправлений. Основные операции на таких терминалах включают:

1. Проведение маркетинговых исследований в области транспортно-логистических услуг.
2. Заключение договоров с клиентами, прием заявок и их обработка.
3. Сбор и распределение грузов.
4. Временное хранение.
5. Осуществление операций по консолидации, разукрупнению, сортировке, комплектации и другим видам обработки грузов.
6. Межтерминальные перевозки и доставка грузов конечным потребителям.
7. Предоставление информационной и компьютерной поддержки для сервисных услуг терминала.
8. Осуществление расчетов за транспортно-логистические услуги.

Таким образом, универсальные грузовые терминалы функционируют как комплексы, предоставляющие широкий спектр логистических операций, связанных с обработкой и доставкой грузов, а также предлагающие различные сервисы в сфере транспортно-логистической деятельности [2].

2. Моделирование процессов работы грузовых терминалов как систем массового обслуживания

Для пояснения процессов обработки транспорта на грузовых контейнерных терминалах применяются вероятностные модели. При этом системы обработки грузов могут переходить из одного состояния в другое в случайные моменты времени. Анализируя возможные сценарии и

состояния системы, а также учитывая случайные факторы, возможно оптимизировать процессы и повысить эффективность работы терминала.

Будем считать, что система из нескольких грузовых терминалов может находиться в следующих состояниях:

E_0- ни один терминал не занят;

E_1- занят 1 терминал;

E_i - занято ровно i терминалов;

E_s - заняты все S терминалы;

E_{s+1} - заняты все S терминалы. В очереди находится 1 транспортное средство;

E_{s+d}- заняты все S терминалы. В очереди находится d транспортных средств.

Традиционная теория обслуживания изучает системы с множеством каналов, где количество доступных оборудованных площадок S соответствует количеству терминалов. Каждый терминал может обслуживаться одной оборудованной площадкой независимо от других терминалов (это известно как система массового обслуживания без взаимной помощи). Кроме того, терминалы могут использовать все доступные площадки или только часть из них (система массового обслуживания с полной или частичной взаимной помощью).

Если интенсивность обработки груза в каждом терминале равна μ_0, а вероятность перехода системы из состояния E_n в состояние E_{n+1} аналогична классической вероятности обработки отдельной заявки в теории массового обслуживания и зависит от количества активных обслуживающих терминалов, то общая интенсивность обслуживания в определенном состоянии n может быть вычислена с использованием принципа линейной суперпозиции. Это означает, что она равна сумме интенсивностей всех работающих приборов обслуживания и кратна интенсивности обслуживания одного отдельного прибора μ_0 .Следовательно, результирующая интенсивность обслуживания не может превышать S_{μ_0}, где S_{μ_0} представляет собой максимальную интенсивность обслуживания при максимальном количестве активных терминалов. Интенсивность обслуживания одним прибором остается постоянной и не зависит от состояния системы, что означает, что вероятность обработки отдельной заявки не меняется в зависимости от текущего состояния СМО [3].

Предположим, что процесс обработки грузов является марковским случайным эргодическим процессом. Это означает, что с течением времени вероятности нахождения системы в различных состояниях приближаются к стационарным значениям. По сути, при достаточно длительном времени $t \to \infty$ вероятности состояний практически не зависят от начального состояния в момент времени $t = 0$ и не зависят от продолжительности времени. Это предположение возможно, потому что все события, приводящие к изменению состояния системы, являются независимыми и простыми.

Для определения значений вероятностей отдельных состояний в стационарных режимах необходимо рассмотреть систему уравнений, где λ – интенсивность потока заявок:

$$\begin{aligned} r_1\mu_0 P_1 &= \lambda P_0; \\ r_2\mu_0 P_2 &= (\lambda + r_1\mu_0)P_1 - \lambda P_0; \\ r_3\mu_0 P_3 &= (\lambda + r_2\mu_0)P_2 - \lambda P_1; \\ &\vdots \\ r_n\mu_0 P_n &= (\lambda + r_{n-1}\mu_0)P_{n-1} - \lambda P_{n-1}. \end{aligned} \quad (1)$$

Введем обозначение $\psi = \frac{\lambda}{\mu_0}$ и назовем это отношение приведенной плотностью потока поступления заявок. Затем, решив систему уравнений (1), мы получим следующие значения стационарных вероятностей для каждого состояния.

$$P_1 = \frac{1}{r_1}\psi P_0,$$

$$P_2 = \frac{1}{r_2}\psi P_1 = \frac{1}{r_1 r_2}\psi P_0,$$

$$P_3 = \frac{1}{r_3}\psi P_2 = \frac{1}{r_2 r_3}\psi^3 P_0,$$

$$P_n = \frac{1}{r_n}\psi P_{n-1} = \frac{1}{\prod_{i=1}^{n} r_i}\psi^n P_0, \quad (2)$$

а также нормировочное условие:

$$P_0\left[\sum_{n=0}^{D}\frac{1}{\prod_{i=0}^{n} r_i}\psi^n\right] = 1. \quad (3)$$

Среднее общее число заявок можно вычислить через преобразование:

$$n = d + S - \sum_{n=0}^{S}(S-n)\frac{\psi^n}{\prod_{n=1}^{S} r_n}p_0. \quad (4)$$

Вне зависимости от значений ψ и S среднее число судов в терминале равно сумме среднего числа обработанных заявок ($n_{\text{обр}}$) и среднего числа заявок, находящихся в очереди (d) т.е.

$$n = n_{\text{обр}} + d.$$

Соответственно, среднее число заявок, находящихся в обработке:

$$n_{\text{обр}} = S - \sum_{n=0}^{S}(S-n)\frac{\psi^n}{\prod_{n=1}^{S} r_n}P_0. \quad (5)$$

Среднее время ожидания транспорта в очереди на разгрузку и среднее общее время пребывания транспорта в терминале определяются с помощью формул Литтла:

$$T_{\text{ож}} = \frac{d}{\lambda} = \frac{P_0}{\lambda}\frac{\psi^{S+1}}{\prod_{i=1}^{S-1} r_i (r_{max} - \psi)^2}. \quad (6)$$

Среднее общее время пребывания транспортной единицы в терминале:

$$T = \frac{n}{\lambda} = \frac{d}{\lambda} + \frac{S}{\lambda} + \frac{1}{\lambda}\left[S - \sum_{n=0}^{S}(S-n)P_n\right] = T_{\text{ож}} + \frac{1}{\lambda}\left[S - \sum_{n=0}^{S}(S-n)\right] \quad (7)$$

Соответственно, среднее время обработки одной транспортной единицы:

$$T_{\text{обр}} = \frac{1}{\lambda}\left[S - \sum_{n=0}^{S}(S-n)P_n\right]. \quad (8)$$

Рассмотрим задачу оптимизации планирования работы терминала при известном числе заявок. Наша цель - выбрать интенсивность потока так, чтобы прибыль в единицу времени была максимальной. Для этой оптимизации мы используем выражение (9) в качестве основы.

$$Э_n = C_0^{'}\varphi S - C_1\varphi\tau_{\text{ож}} - C_2^{'}\varphi S\left(1 - K_{\text{пр}}\right) - C_2^{'}SK_{\text{пр}} - C_3 S = C_0^{''}\varphi s - C_1\varphi S\tau_{\text{ож}} - C_2^{''}S. \quad (9)$$

Зная значения коэффициентов C_0, C_1, C_2, а также интенсивности обработки грузов μ и коэффициент простоя $K_{\text{ф}}$, вычисляется соотношение $C_0^{'}/C_1$. На основе этого соотношения и известного числа терминалов S определяется оптимальное значение коэффициента загрузки каналов φ. Далее легко определяется оптимальная приведенная плотность входного потока и

его оптимальная интенсивность. На рис.1 приведены расчетные параметры работы комплекса грузовых терминалов.

```
Введите количество терминалов:5
Введите количество клиентов:30
Введите интенсивность потока обработки:6
====================
Основные показатели
Результирующая интенсивность обслуживания: 180.0
Интенсивность потока поступления: 1.254
Результирующая интенсивность обработки грузов: 0.29
Плотность потока прихода клиентов: 0.209
====================
========================================
Коэффициент интенсивности обработки в %:
0.01 0.029 0.045 0.026 0.048
========================================
========================
Стационарные вероятности
0.0 0.0 0.0 0.0 0.0
========================
==================
Средние показатели
Число клиентов, находящихся в очереди: 25
Среднее число клиентов,находящихся в очереди: 0
Среднее количество клиентов, находящихся в терминале: 55
Среднее количество клиентов, находящихся в обработке: 30
Среднее время ожидания в очереди: 19.94 минут(ы)
Среднее общее время пребывания в терминале: 3.99 минут(ы)
Среднее время обработки одного клиента: 23.92 минут(ы)
Среднее приведенное время ожидания: 119.64 минут(ы)
Среднее приведенное время пребывания в терминале: 19.09 минут(ы)
```

Рис.1. Анализ производительности грузовых терминалов

Заключение

В данной работе была выделена значимость и актуальность оптимизации деятельности грузовых терминалов, а также предложен метод моделирования их процессов на основе концепции систем массового обслуживания (СМО). Улучшение работы грузовых терминалов имеет ключевое значение для повышения эффективности логистических цепей. Применение моделирования на основе СМО способствует более глубокому пониманию процессов, происходящих на терминалах, и формированию оптимальных стратегий управления ресурсами и организации их работы.

Литература

1. Волгин, В. В. Логистика приемки и отгрузки товаров: практическое пособие / В. В. Волгин. - Москва: Дашков и К, 2009. - 457 с
2. Миротина Л.Б. Логистика. Управление в грузовых транспортно-логистических системах // Л.Б. Миротина учебное пособие М.: Юристъ, 2002. — 414 с.
3. Русинов И. А. Оптимизация работы контейнерных терминалов / И. А. Русинов // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2010. – № 22(198). – С. 31-36.

УДК *531.3, 533.2*

МЕТОДИКА РАСЧЕТА ЗАПУСКА И ОСТАНОВА ДВИГАТЕЛЯ В SIMINTECH

И.В. Еремин

Воронежский государственный университет

Введение

Развитие космической промышленности на сегодняшний день является одним из главных направлений научно-технического комплекса. Все более перспективным направлением становится создание многоразовых ракет-носителей, так как использование такого вида ракет имеет ряд преимуществ, таких как дешевизна эксплуатации, более высокая экологичность, повышенная частота и скорость запусков.

В качестве топлива для многоразовых ракет-носителей часто рассматривают пару кислород и метан, так как они недороги в производстве, легки в хранении, нетоксичны, а также после отработки двигателя на метане не придется очищать его от сажи.

1. Постановка задачи

Задача состоит в математическом моделировании параметров для запуска и остановки двигателя, работающего на компонентах метан и кислород, по схеме, соответствующей определенной программе работы. В качестве газа для продувки и вращения роторов турбонасосного агрегата используется азот.

В рамках физической формулировки задачи анализируются процессы потока жидкости и газа, сгорания топливных компонентов и динамики твердых тел. Эти процессы описываются с помощью уравнений теоретической механики, гидромеханики, термодинамики и газодинамики, которые обычно применяются для анализа работы жидкостных ракетных двигателей. Основой для формирования уравнений служат законы сохранения массы, импульса и энергии.

Для имитации работы отдельных узлов применяются обобщенные параметры, полученные в ходе проектирования и уточняемые в процессе тестирования. Аналитические соотношения, использованные в исследовании, указаны в источниках [1-7].

Математическая формулировка задачи включает в себя создание полной системы обыкновенных дифференциальных и алгебраических уравнений, разработку алгоритма для достижения сходимости и стабильности решения, а также программирование на ЭВМ с использованием динамической среды разработки SimInTech. Модель двигателя в программе представлена в виде блочной структуры, где каждая часть двигателя имеет свою подмодель, которая может быть применена для моделирования двигателей с различными расчетными схемами и характеристиками.

1.1. Расчетная схема

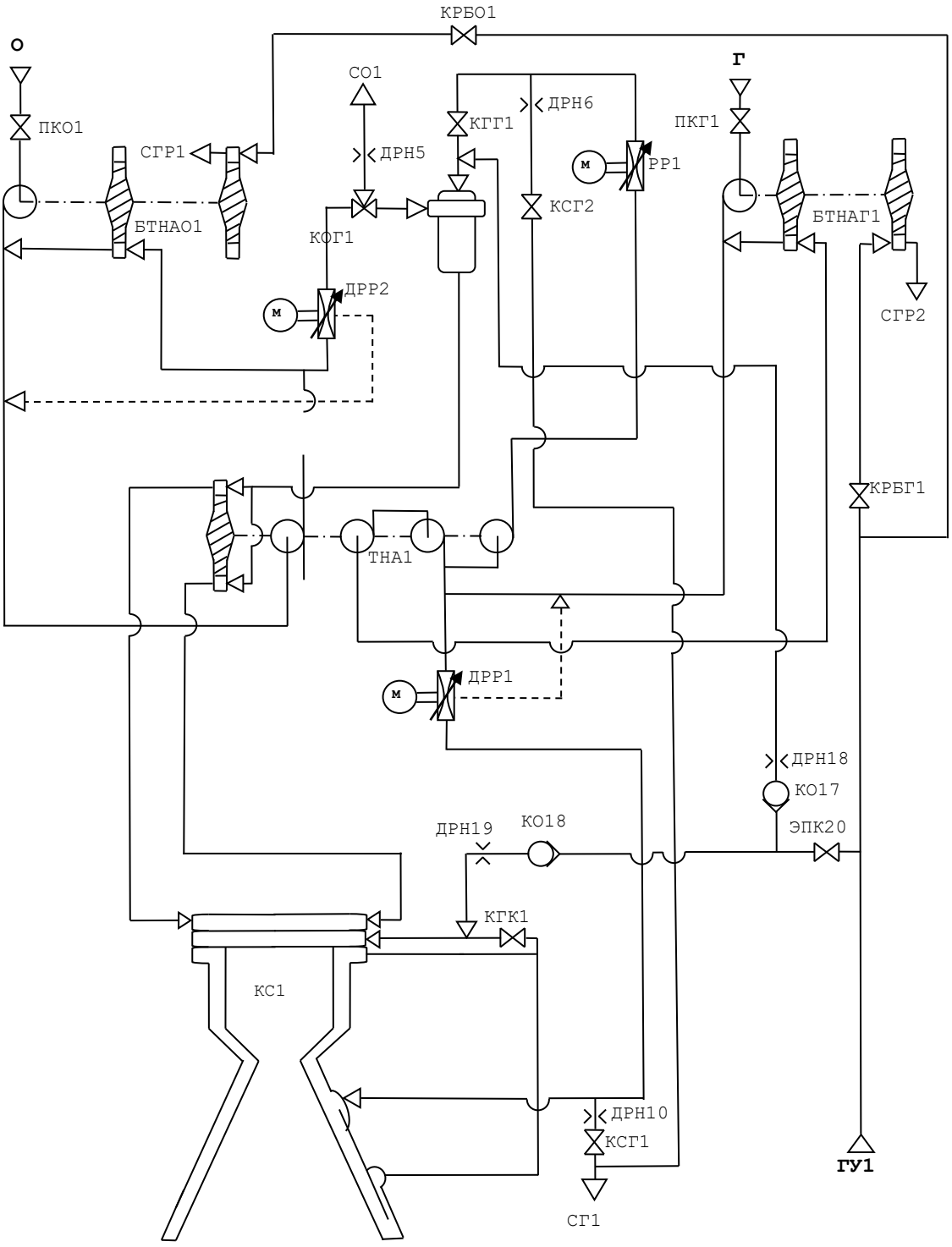


Рис. 1. Расчетная схема двигателя

- БТНАГ1 – бустерный турбонасосный агрегат горючего;
- БТНАО1 – бустерный турбонасосный агрегат окислителя;
- Г – подача горючего в двигатель;

- ГГ1 – газогенератор;
- ГУ1 – подача газа управления и продувки в двигатель;
- ДРН5, ДРН6, ДРН10, ДРН18, ДРН19 – шайбы дроссельные;
- ДРР1 – дроссель на линии горючего камеры;
- ДРР2 – дроссель на линии окислителя газогенератора;
- КГГ1 – клапан горючего газогенератора;
- КГК1 – клапан горючего камеры;
- КО17, КО18 – клапаны обратные;
- КОГ1 – клапан окислителя газогенератора;
- КРБО1, КРБГ1 – клапаны подачи газа раскрутки роторов
БТНАО1 и БТНАГ1;
- КС1 – камера;
- КСГ1 – клапан слива горючего перед трактом охлаждения камеры;
- КСГ2 – клапан слива горючего после регулятора;
- О – подача окислителя в двигатель;
- ПКГ1 – пусковой клапан горючего;
- ПКО1 – пусковой клапан окислителя;
- РР1 – регулятор;
- СГ1 – слив горючего;
- СГР1, СГР2 – сброс газа раскрутки роторов БТНАО1 и БТНАГ1;
- СО1 – слив окислителя;
- ТНА1 – турбонасосный агрегат;
- ЭПК20 – электропневмоклапан подачи азота продувки
форсуночной головки окислителя ГГ1,
форсуночных головок горючего ГГ1 и КС1.

На расчетной схеме не показаны запальные устройства ГГ1 и КС1, работа которых обеспечивает воспламенение компонентов топлива в этих агрегатах.

1.2. Описание программы

Методика реализована в среде динамического моделирования технических систем SimInTech. Программа выполнена в виде блочной структуры, где блок представляет из себя субмодель с входными и выходными портами, по которым при помощи связей и сигналов передаются на каждом шаге вычисления данные из других блоков. В свойствах субмодели указываются начальные данные, характеристики агрегата, константы и таблицы. Уравнения и условия записываются внутри субмоделей в блоке «язык программирования» на собственном языке SimInTech.

К проекту подключена база данных, в которой записана исходная информация по физическим свойствам компонентов топлива, продуктов сгорания и газа продувки, в виде констант и таблиц, табличные характеристики агрегатов, информация по циклограмме запуска и остановы двигателя.

Расход в магистралях окислителя и горючего рассчитывается по отдельным участкам, для каждого такого участка используется универсальный блок, всего 15 блоков, в которых записаны физические свойства компонента и свойства участка магистрали. Для учета узлов используется отдельные блоки, в которые в зависимости от схемы входят расходы из разных участков магистралей, всего в модели присутствует 6 узлов.

Для учета продувки и раскрутки используется блок ГУ, в котором рассчитывается расход газа на раскрутку турбин БТНАО и БТНАГ, а также на продувку головки горючего газогенератора и камеры сгорания.

Модели ТНА, БТНАО и БТНАГ состоят из отдельных субмоделей для насоса, турбины и ротора, для которых задаются свойства и характеристики при помощи таблиц из базы данных.

Для моделирования газогенератора, газовода и камеры сгорания используется один и тот же типовой блок-субмодель с различными свойствами и связями. Всего присутствует 3 таких блока.

Форсуночные головки горючего газогенератора и камеры сгорания описаны отдельными блоками. Тракт охлаждения и полость перед КГК также представлены отдельными блоками.

2. Результаты расчета

Расчет производился с шагом $\Delta t = 10^{-5}$ с методом Эйлера. На рисунках 2 – 8 представлены графики давлений в точках за насосами, за трактом охлаждения, в газогенераторе, газоводе и в камере сгорания, расходы через линии магистрали окислителя и горючего, обороты роторов БТНАО, БТНАГ и ТНА. На рисунке 2, в начале расчета, с -5 до 0 секунды давление практически не растет, идёт раскрутка турбин, что видно по оборотам роторов БТНАО и БТНАГ, ротор ТНА при этом практически не прибавляет в оборотах до нулевой секунды. Примерно с 0.6 секунды давление начинает расти соответственно работе двигателя по циклограмме. Двигатель выходит на предварительный режим при помощи регулятора и дросселей. С 6 по 10 секунду двигатель работает в стационарном режиме, что видно по графику давлений, после чего работа дросселей и регулятора приводит к снижению давлений во всем двигателе. На рисунках 3 и 4 расход соответствует работе регулятора, выходя на общий режим с 6 секунды, после чего, с 10 секунды регулятор настраивается на останов, что также видно по графику.

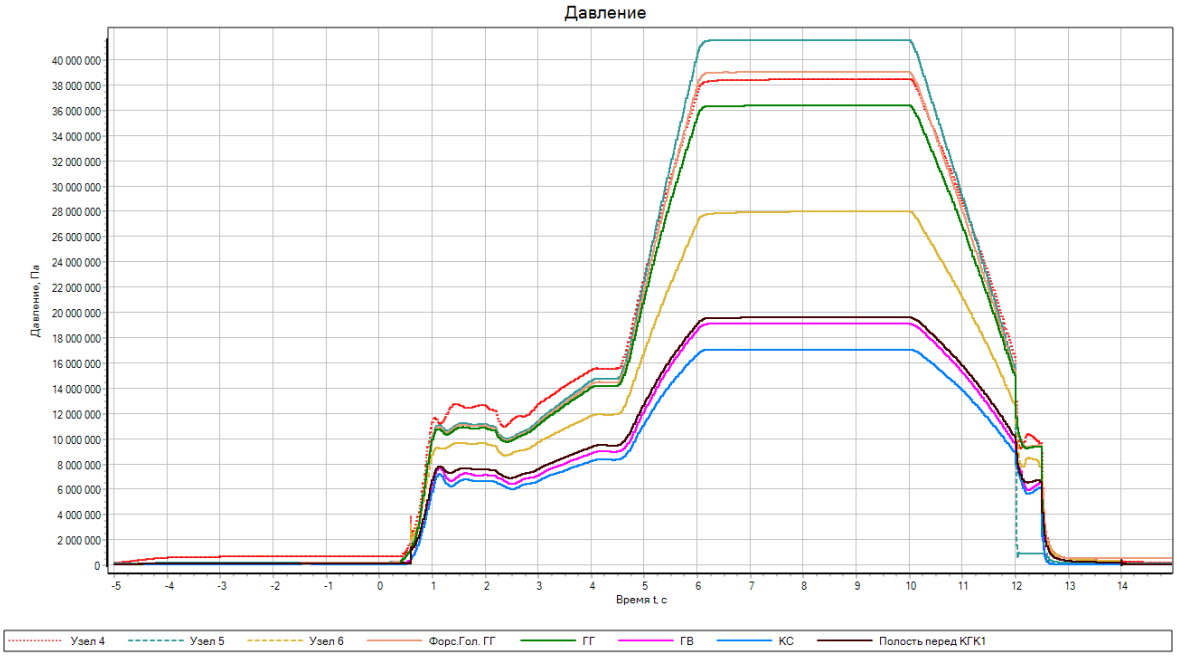


Рис. 2. График давлений

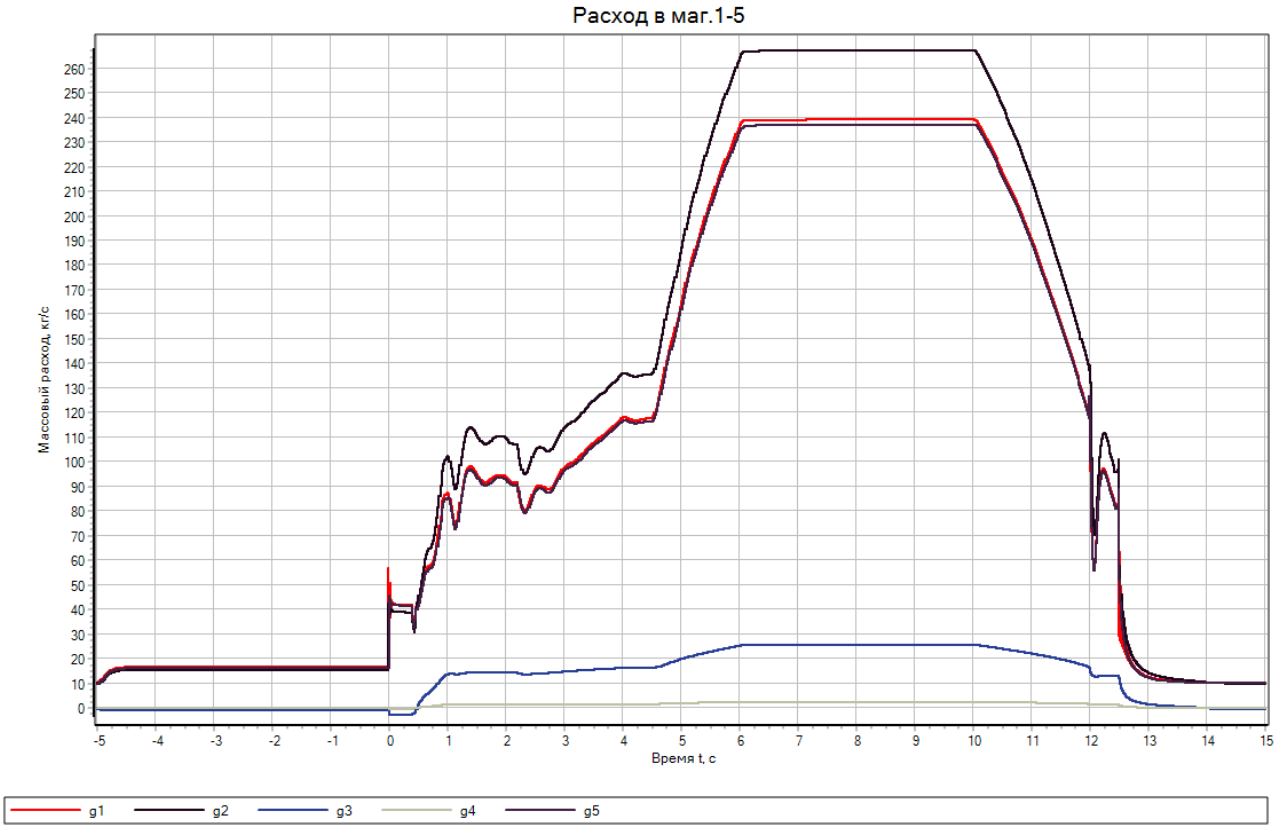


Рис. 3. Расход в магистрали окислителя

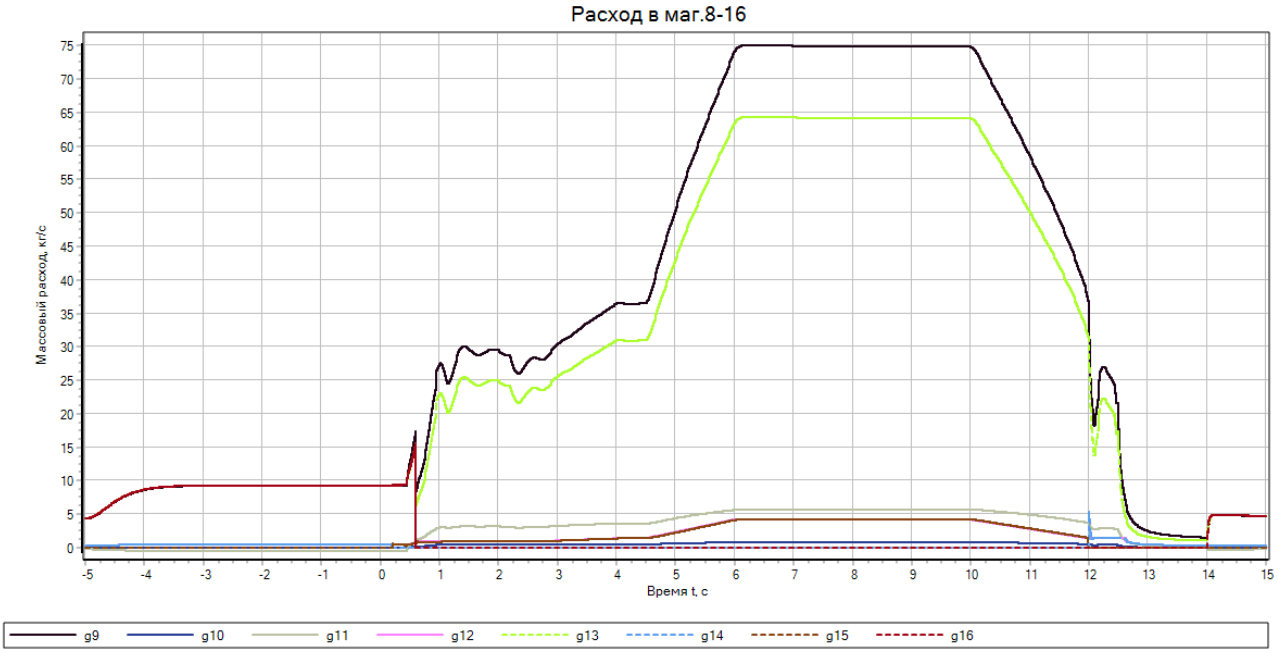


Рис. 4. Расход в магистрали горючего

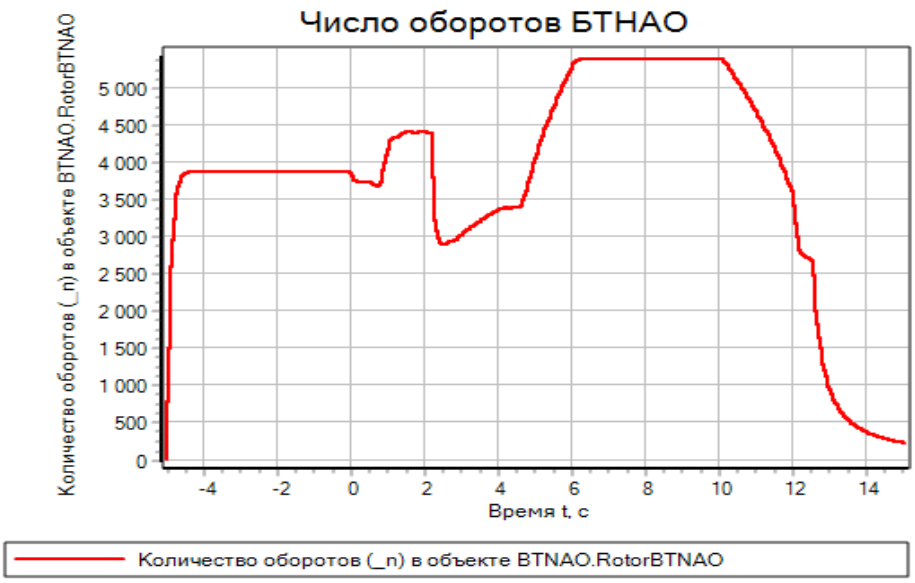


Рис. 5. Обороты ротора БТНАО

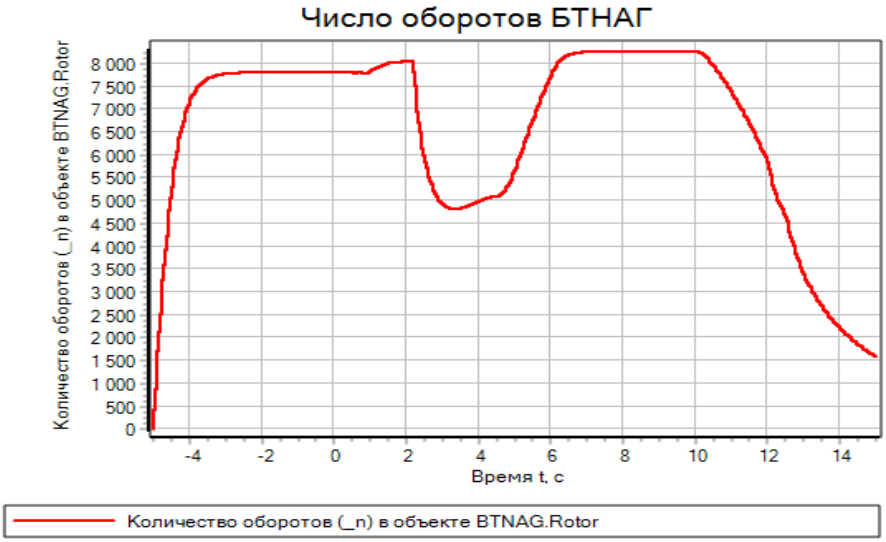


Рис. 6. Обороты ротора БТНАГ

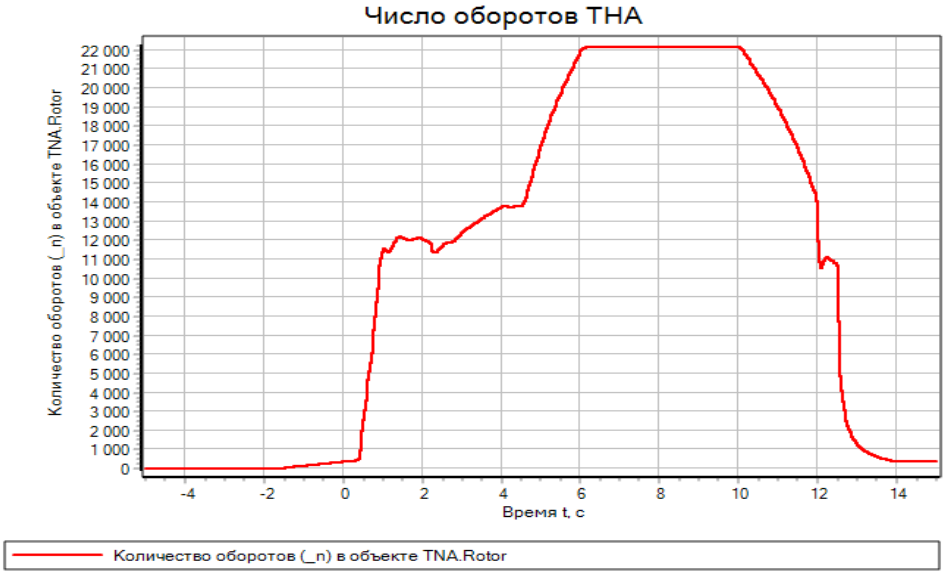


Рис. 7. Обороты ротора ТНА

Заключение

Реализована и протестирована методика расчета запуска и останова двигателя на компонентах кислород и метан. Построены графики, на основе которых можно провести анализ работоспособности системы. Данная методика, разработанная в ПО SimInTech может быть применена в производстве для расчета реальных двигателей.

Литература

1. Махин В.А., Пресняков В.Ф., Белик Н.П. Динамика жидкостных ракетных двигателей. – М., Машиностроение, 1969.

2. Добровольский М.В. Жидкостные ракетные двигатели. – М., Машиностроение, 1971.

3. Беляев Е.Н., Чванов В.К., Черваков В.В. Математическое моделирование рабочего процесса жидкостных ракетных двигателей. – М., издательство МАИ, 1999, – 228 с.

4. Лойцянский Л.Г. Механика жидкости и газа. – М., Наука, 1973, – 902 с.

5. Абрамович Г.Н. Прикладная газовая динамика. – М., Наука, 1969, – 824 с.

6. Овсянников Б.В., Боровский Б.И. Теория расчета агрегатов питания ЖРД. – М., Машиностроение, 1971.

7. Вукалович М.П., Новиков И.И. Термодинамика. – М., Машиностроение, 1972, – 670 с.

УДК 004.6

АНАЛИЗ ПРИМЕНЕНИЯ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРЕОБРАЗОВАНИЯ ВЕКТОРНЫХ ПРЕДСТАВЛЕНИЙ ТЕКСТА

Д. О. Ершов

Воронежский государственный университет

Введение

При анализе больших объёмов текстовых данных нередко возникает необходимость изучения распределения образцов по смысловому содержанию. Современные методы кластерного анализа позволяют существенно сократить временные затраты на исследовании данных, автоматически разбивая выборку на группы текстов, объединенных общим семантическим контекстом.

На качество кластеризации влияют многие факторы: способ получения векторных представлений текста, выбор алгоритма кластеризации, метрика близости текста и свойства исследуемых данных. Однако ключевым аспектом при выделении кластеров является метод вычисления близости между векторными представлениями текста. Для получения релевантных результатов как кластеризации, так и алгоритмов поиска, близкие по тематике тексты должны иметь меньшее расстояние в векторном пространстве, чем далёкие по смыслу. Достижение данного свойства возможно не только путём оптимизации метрики близости, но и за счёт преобразования самих векторных представлений текста — эмбеддингов.

Современные способы получения текстовых эмбедингов, основанные на использовании больших языковых моделей, предполагают обучение на массивах разнородных данных. Это позволяет сформировать словарь объемом несколько десятков тысяч слов, чтобы показать хорошие результаты на текстах, посвященных широкому спектру тем. Однако в случае, когда исследуемые данные уже объединены общей тематикой, возникает необходимость корректировки текстовых эмбеддингов для повышения точности кластеризации.

Таким образом, адаптация текстовых эмбеддингов к специфике исследуемых данных является перспективным направлением для улучшения качества кластеризации тематически однородных текстов. Данный подход предполагает модификацию векторных представлений текста с учётом особенностей конкретной предметной области, что позволяет более точно отразить семантические связи между текстовыми образцами и повысить эффективность выделения кластеров.

1. Исследование

1.1. Постановка задачи

Целью данного исследования является изучение влияния преобразования текстовых эмбеддингов на эффективность решения задачи кластеризации. Для этого потребуется собрать выборку текстовых данных, объединённых общими тематическими категориями, и произвести их векторизацию с применением большой языковой модели. Далее необходимо составить обучающий датасет, разметив текстовую выборку на определённое количество кластеров. Трансформация эмбеддингов будет осуществляться посредством их умножения на матрицу, вычисленную с использованием алгоритма стохастического градиентного спуска,

оптимизирующего расстояния между векторными представлениями текстов, принадлежащими к одному кластеру.

С целью объективной оценки влияния применённого преобразования на качество кластеризации необходимо рассчитать метрики для внешней и внутренней валидации кластеризующего алгоритма как до, так и после применения преобразования текстовых эмбеддингов.

Сравнительный анализ полученных результатов позволит сделать выводы об эффективности предложенного подхода к повышению качества кластеризации текстовых данных.

1.2. Сбор и обработка данных

Для проведения исследования был собран датасет, состоящий из 213 вопросов, опубликованных участниками публичного форума в социальной сети. Тексты вопросов написаны на русском языке и характеризуются схожей структурой: приветствие, краткое описание ситуации и непосредственно сам вопрос.

Предварительная обработка текста, например стеминг и лематизация, не проводилась, так как токенизаторы больших языковых моделей обучались именно на неочищенных сырых данных.

Анализ распределения длин вопросов после проведения токенизации (рис. 1) показал, что более 98% наблюдений имеют длину, не превышающую 512 токенов, что означает возможность использования языковой модели архитектуры RoBERTa [1], контекстное окно которой как раз равно 512 токенам, для получения текстовых эмбеддингов.

На первом этапе кластеризации с использованием алгоритма k-means было выделено 7 кластеров. После детального изучения результатов, данные были размечены на 15 кластеров, которые более точно разбивают вопросы по тематическим группам.

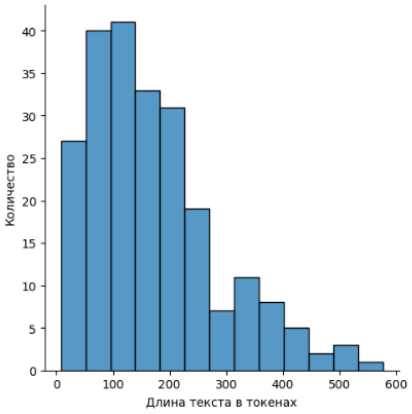


Рис 1. Распределение вопросов по длине

Для более качественной кластеризации текстов по смыслу потребуется сократить расстояние между образцами, принадлежащими к одному кластеру и увеличить его для текстов, которые принадлежат разным. Составим обучающую выборку, в каждой строке которой расположим пары образцов которые находятся как в одном кластере, так и в разных. Если в строке содержится пара образцов, принадлежащих одному кластеру, установим косинусное расстояние равное 1. Для негативных пар установим значение -1. Таким образом для части исходной выборки был сформирован обучающий датасет (рис 2) содержащий идентификационные номера для пар текстов, принадлежащих ко всем 15 кластерам.

	id1	id2	label
0	18361	16199	1
1	18361	18598	1
2	18361	19661	1
3	18361	18649	1
4	18361	18065	1
...	...	...	...
2550	19040	18779	-1
2551	19040	18499	-1
2552	19040	16972	-1
2553	19040	18449	-1
2554	19040	16640	-1

4686 rows × 3 columns

Рис 2. Обучающая выборка.
Колонки id1 и id2 соответствуют векторным представлениям текстов

1.3. Векторизация текста

Для получения текстовых эмбеддингов была использована предобученная модель RuRoBERTa, объём обучающего корпуса которой составил 250 гигабайт русскоязычного текста. Данная языковая модель использует преимущества архитектуры *Transformer*, основанной на механизме самовнимания (*self-attention*) [2], что позволяет эффективно моделировать долгосрочные зависимости в последовательностях. Главное отличие от классической архитектуры заключается в том, что RuRoBERTa содержит только кодировщик, вместо кодировщика и декодера. Это позволяет использовать большие объёмы неразмеченных данных для обучения.

Полученные с помощью RuRoBERTa текстовые эмбеддинги могут быть использованы для решения таких задач, как классификация и кластеризация текстов, восстановление пропущенных слов, ответы на вопросы и другие. Высокое качество эмбеддингов, генерируемых моделью, обусловлено её способностью уловить глубокие семантические и синтаксические связи в текстах, что является результатом предобучения на огромном объёме данных и использования современной архитектуры Transformer.

Существует несколько способов для получения векторизированного текста, путём извлечения состояния разных слоёв большой языковой модели (см. рис. 3). В данной работе был применён метод конкатенации векторов, соответствующих последним четырём скрытым слоям. В результате для каждого вопроса из исследуемого датасета был получен вектор размерности 3072, представляющий собой комбинированное описание текстового содержимого вопроса, учитывающее информацию из различных уровней абстракции языковой модели

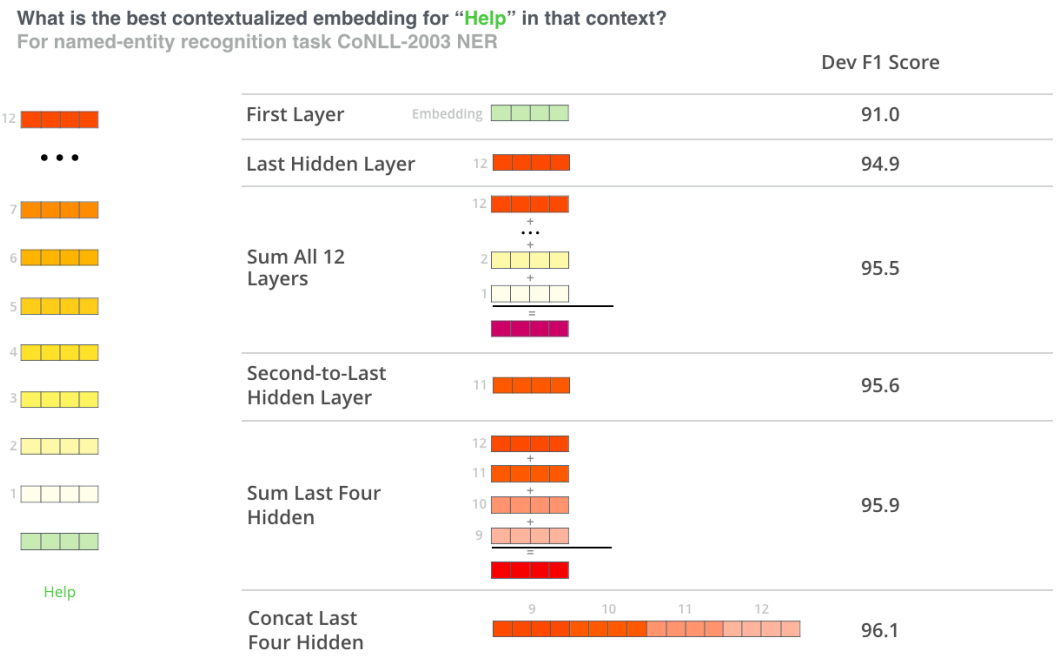


Рис. 3. Сравнение способов извлечения текстовых эмбеддингов модели архитектуры BERT

1.4. Преобразование текстовых эмбеддингов

Для каждой пары вопросов из обучающего набора данных было вычислено косинусное расстояние, и проведен анализ распределения пар (рис. 4), принадлежащих одному или различным кластерам, в зависимости от значения расстояния между ними. Полученные результаты явно свидетельствуют о невозможности разделения вопросов из исследуемого датасета на требуемые кластеры, поскольку косинусное расстояние близко к единице вне зависимости от принадлежности образцов к одному кластеру. Данный вывод также подкрепляется относительно низкой точностью определения принадлежности вопросов к одному или разным кластерам на тестовом наборе данных, составляющей 56.7%. Таким образом, исходное векторное представление вопросов не обеспечивает достаточной дискриминативной способности для эффективного решения задачи кластеризации.

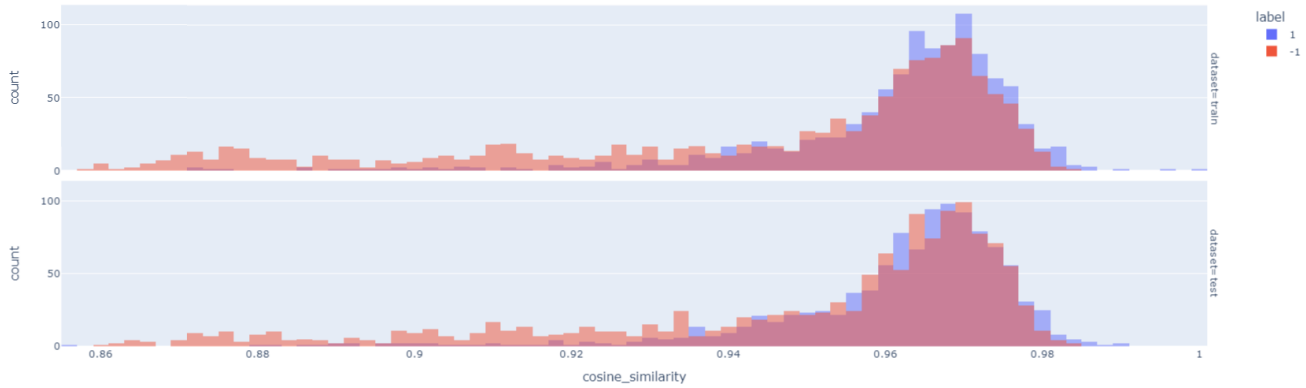


Рис. 4. .Распределение позитивных и негативных пар

Решим данную проблему корректировкой эмбеддингов, умножив каждый вектор на матрицу, веса которой получим используя алгоритм стохастического градиентного спуска.

Пусть $\mathbf{x}$ — векторное представление текста размера n, $\mathbf{M}$ — матрица размером $n \times m$.

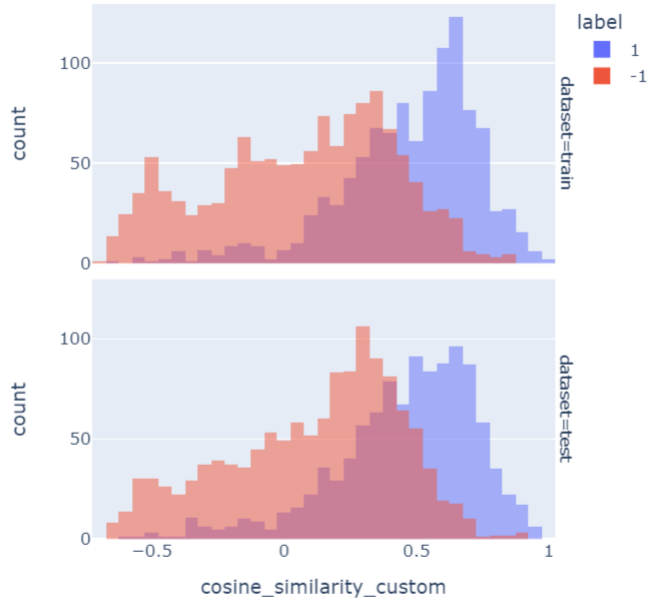
Для парыв векторов **x1** и **x2** определим distance = cosine_distance($\mathbf{x1} * \mathbf{M}, \mathbf{x2} * \mathbf{M}$)

Тогда функция потерь *loss* = mean_squared_error(distance, actual_distance), где actual_distance желаемое расстояние между векторами из тренировочной выборки.

Для каждой итерации алгоритма будем обновлять веса матрицы $\mathbf{M}$ по правилу $\mathbf{M} = \mathbf{M} - \nabla\mathbf{M} \bullet learning\_rate$.

Для оценки качества обучения рассчитаем метрику $accuracy = \frac{correct}{total}$ для каждой эпохи градиентного спуска. В результате удалось увеличить точность с 56.7% до 74.2%.

Оценим распределение пар по косинусному расстоянию после преобразования эмбеддингов (рис. 5). Заметно, что у пар текстов сильно повысилась разделимость: значительно уменьшилась площадь пересечения для позитивных и негативных примеров.




```
Test accuracy after customization: 73.7% ± 1.8%
```

Рис. 5. Распределение позитивных и негативных пар после обучения

1.3. *Метрики оценки качества кластеризации*

Для оценки качества кластеризации определим метрики для внутренней (1–3) и внешней (4) валидации:

$$Davies - Bouldin\ index = (\frac{1}{n}\sum \max(R_{ij}) \quad (1)$$

где n – количество кластеров, а R_{ij} – определяется как средний показатель сходства каждого кластера с наиболее похожим кластером, где сходство представляет собой отношение расстояний внутри кластера к расстояниям между кластерами. Таким образом, кластеры, которые находятся дальше друг от друга и менее разбросаны, получат более высокий балл.

$$Calinski-Harabasz\ index = \left[\frac{\sum n_i \|c_i - c\|^2}{k-1}\right] / \left[\frac{\sum_{i=1}^{k}\sum_{x\in C_i}\|x - c_i\|^2}{n-k}\right] \quad (2)$$

где n – количество наблюдений для кластера k, c_k – центроида кластера.

$$Silhoutte\ score = \frac{b_i - a_i}{\max\{a_i, b_i\}} \quad (3)$$

где a_i – среднее расстояние между объектами, b_i — минимальное расстояние до ближайшего кластера.

$$Rand\ index = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

где TP – количество пар, которые были правильно определены в один кластер, TN – количество пар, которые были правильно определены принадлежащими к разным кластерам, FP – количество пар, которые были неправильно определены в один кластер, FN – количество пар, которые были неправильно разделены в разные кластеры.

Davies–Bouldin index (1) показывает наилучшее значение равное 0, когда кластеры разделены наилучшим образом. Calinski–Harabasz index (2) обозначает соотношение отношение суммы дисперсии между кластерами и дисперсии внутри кластера. Коэффициент силуэта (3) характеризует уровень перекрытия кластеров, где 1 — отсутствие перекрытия классов (лучшее значение), а -1 — отнесение образцов к неправильным кластерам (худшее значение).

В табл. 1 сравним метрики качества кластеризации для исходных эмбеддингов и преобразованых: заметно значительное увеличение Rand Index, Calinski–Harabasz index и Silhoutte Score [4–6], что означает улучшение результата разбиения образцов на кластеры.

Таблица 1

Сравнение метрик кластеризации

Метрика	До преобразования	После преобразования
Davies-Bouldin index	1.79	2.07
Calinski-Harabasz index	8.22	13.89
Silhoutte Score:	0.07	0.16
Rand Index	0.61	0.71

Заключение

В результате применения алгоритма машинного обучения была получена матрица преобразования для трансформации векторных представлений текстовых данных (эмбеддингов). Использование данной матрицы позволило значительно улучшить показатели качества кластеризации текстовых документов, оцениваемые с помощью соответствующих метрик. Кроме того, применение упомянутого преобразования обеспечило снижение размерности векторов, кодирующих текстовую информацию, с 3072 до 2048 компонент, что соответствует уменьшению в 1.5 раза. Подобная редукция размерности способствует ускорению вычисления метрик близости между векторными представлениями, что оказывает критическое влияние на производительность алгоритмов кластеризации и поиска, оперирующих с данными векторами.

Литература

1. Liu Y. RoBERTa: A Robustly Optimized BERT Pretraining Approach / Y. Liu, M. Ott, N. Goyal, J. Du et al. – Электрон. препринт. – Режим доступа: https://arxiv.org/pdf/1907.11692.pdf

2. Vaswani A., Shazeer N., Parmar N. Attention Is All You Need. // – arXiv.org. 2017. Дата обновления: 12.06.2017. URL: https://arxiv.org/abs/1706.03762/ (дата обращения: 12.04.2023).

3. Lin Wu. Research on Multilingual News Clustering Based on Cross-Language Word Embeddings / Lin Wu, Rui Li, Wong-Hing Lam. – Электрон. препринт. – Режим доступа: https://arxiv.org/ftp/arxiv/papers/2305/2305.18880.pdf

4. Davies, D. L. A Cluster Separation Measure / D. L. Davies, D. W. Bouldin // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1979. – PAMI-1, № 2. – С. 224–227.

5. Caliński, T. A dendrite method for cluster analysis / T. Caliński, J. Harabasz // Communications in Statistics. – 1974. – Vol. 3, № 1. – P. 1–27.

6. Devlin J. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Devlin J., Ming-Wei C., K. Lee, K. Toutanova – Электрон. препринт. – Режим доступа: https://arxiv.org/pdf/1810.04805.pdf

7. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) – Режим доступа: https://jalammar.github.io/illustrated-bert – (Дата обращения: 04.04.2014).

8. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем / Орельен Жерон – Санкт-Петербург.: ООО «Альфа-книга»: 2018. – 688 с.

9. Программирование на Python, том II, 4-е издание. / Лутц М. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

10. Документация библиотеки scikit-learn – Режим доступа: https://scikit-learn.org/stable/index.html – (Дата обращения: 05.04.2014).

УДК 004.42

РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ ДЛЯ ТЕСТИРОВАНИЯ ОПТИМИЗАТОРОВ SQL-ЗАПРОСОВ

Е. Р. Ершова

Воронежский государственный университет

Введение

В настоящее время очень важно писать оптимизированные запросы, чтобы они выполнялись максимально быстро. Однако один запрос при разных наполнениях базы данных может не всегда давать одинаковый и оптимальный план выполнения.

Цель работы – реализовать приложение, которое будет инструментом для дальнейшего исследования и тестирования оптимизаторов SQL-запросов в различных системах управления базами данных.

1. Постановка задачи

Необходимо спроектировать структуру входных файлов, в которых в формате xml [1] будет описана информация по заданным таблицам, а также создать приложение, которое будет:

- выполнять разбор и структурирование информации из файлов создания и заполнения базы данных;
- реализовывать алгоритмы генерации данных;
- заполнять полученными значениями базу данных;
- поддерживать возможность указывать количество строк;
- поддерживать выбор системы управления базами данных.

2. Проектирование структуры файлов

2.1. Структура файла для создания таблиц

Структура xml-файла для создания таблиц содержит следующие теги:

- *database* – корневой элемент, содержащий в себе все таблицы;
- *createTable* – содержит информацию об одной таблице, название которой отображено в атрибуте tableName;
- *column* – содержит информацию об одном столбце, название которого в атрибуте name, а тип данных в атрибуте type;
- *constraints* – тег, используемый, если столбец является первичным ключом. Содержит атрибут primaryKey и nullable;
- *addForeignKeyConstraint* – позволяет добавить информацию о столбце, который является внешним ключом. Атрибутами являются: constraintName – название внешнего ключа, baseTableName и referencedTableName – соединяющиеся таблицы, baseColumnNames и referencedColumnNames – столбцы, по которым идёт соединение таблиц.

2.2. Структура файла для заполнения таблиц

Структура xml-файла для заполнения таблиц содержит следующие теги:

- *root* – корневой элемент, содержащий в себе все таблицы. Содержит атрибут name – наименование корневого элемента;
- *tables* – список всех таблиц;
- *table* – таблица с атрибутом name – наименование таблицы;
- *columns* – список всех столбцов таблицы;
- *column* – столбец с атрибутом name – наименование столбца;
- *generateStrategy* – стратегия генерации. Содержит атрибут type – тип стратегии (primaryKey, foreignKey, range, enumeration, pattern);
- *parameters* – список параметров, которые используются в стратегии;
- *parameter* – параметр стратегии с атрибутом name – наименование параметра;
- *multiValueParameters* – список параметров, относящиеся к перечислениям;
- *multiValueParameter* – параметр стратегии для генерации из множества значений с атрибутом name;
- *values* – список значений для генерации;
- *value* – значение для генерации.

Пример xml-файла заполнения столбца с фамилиями авторов для таблицы с авторами произведений приведён на рис. 1.

```
...
<column name="author_surname" type="varchar(255)">
    <generateStrategy type="enumeration">
        <multiValueParameters>
            <multiValueParameter name="allowedValues">
                <values>
                    <value>Твен</value>
                    <value>Толстой</value>
                    <value>Чехов</value>
                    <value>Достоевский</value>
                    <value>Пушкин</value>
                    <value>Салтыков-Щедрин</value>
                    <value>Солженицын</value>
                    <value>Пастернак</value>
                    <value>Блок</value>
                    <value>Есенин</value>
                    <value>Набоков</value>
                </values>
            </multiValueParameter>
        </multiValueParameters>
    </generateStrategy>
</column>
...
```

Рис. 1 Пример фрагмента xml-файла с описанием одного столбца

3. Реализация приложения

Консольное приложение написано на языке Java [2]. Архитектура представлена на рис.2.

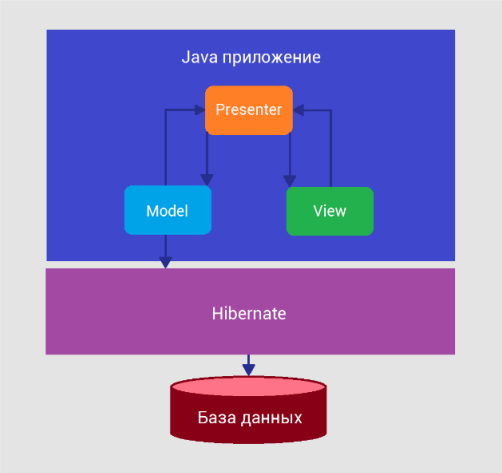


Рис. 2 Архитектура приложения

Шаблон MVP используется для упрощения разработки и тестирования приложения [3]. Рассмотрим подробнее его компоненты:

- model – логика приложения. Набор классов представляет данные, с которыми работает приложение. Классы повторяют наименования тегов в файле для заполнения базы данных;
- view – пользователю в консоли доступно меню: загрузить файл для создания таблиц, загрузить файл для заполнения таблиц, а затем возможность задать количество необходимых строк.
- presenter – компонент, отвечающий за взаимодействие между моделью (model) и представлением (view).

После разбора файла для заполнения таблиц необходимо создать динамические модели, сгенерировать значения и передать их в нужную таблицу. Для этого нужен Hibernate – фреймворк, который используется для работы с базами данных в Java-приложениях [4]. Его настраивают на работу с моделью путём определения конфигураций. Hibernate автоматически преобразует данные между моделью и базой данных.

Данные в таблицах имеют разный вид: строковые выражения, целые значения, вещественные параметры и т.д. Для того, чтобы сгенерировать необходимое количество строк таких данных необходимы различные алгоритмы – стратегии генерации.

Поэтому для каждой стратегии генерации в приложении создан отдельный класс:

- PrimaryKeyCommand
  - класс для стратегии генерирования первичного ключа;
  - применяется для типов number, string;
  - генерируется уникальное значение;
- ForeignKeyCommand
  - класс для стратегии генерирования внешнего ключа;
  - применяется для типа number, string;
  - параметры: refTable – таблица, с которой производится соединение, а также refColumn – уникальный идентификатор столбца, с которым производится соединение;
  - выбирается произвольное значение из исходного столбца;
- RangeCommand
  - класс для стратегии генерирования случайных значений;

  - применяется для типа number, date;
  - параметры: rangeFrom – значение, начиная с которого необходимо генерировать данные, а также rangeTo – значение, до которого необходимо генерировать данные;
  - генерируется значение в указанном диапазоне;
- EnumerationCommand
  - класс для стратегии генерации из перечислений – множества строковых значений;
  - применяется для типов number, string, date;
  - параметр: allowedValues – список допустимых значений;
  - равновероятно выбирается значение из списка допустимых значений;
- PatternCommand
  - класс для стратегии генерации строки по регулярному выражению;
  - используется для типа string;
  - создание осуществляется с помощью библиотеки Generex для генерации строк, соответствующих заданному регулярному выражению.

Все классы стратегий наследуются от абстрактного класса GeneratingCommand и реализуют каждый свой алгоритм генерации.

Таким образом, приложение позволяет заполнять базу данных разным количеством строк для того, чтобы в дальнейшем анализировать планы выполнения запросов при разной наполненности строк и определять оптимальный план среди нескольких систем управления базами данных [5].

Заключение

В результате работы спроектирована структура входных файлов, в которых в формате xml описана информация по заданным таблицам, а также создано приложение, в котором:

- выполняется разбор и структурирование информации из файлов создания и заполнения базы данных;
- реализованы алгоритмы генерации данных;
- полученные значения заполняют базу данных;
- поддерживается возможность указывать количество строк;
- поддерживается выбор системы управления базами данных.

Список литературы

1. A Guide to XML in Java. – Режим доступа: https://www.baeldung.com/java-xml. – (дата обращения: 03.02.2024).
2. Java Documentation. – Режим доступа: https://docs.oracle.com/en/java. – (дата обращения: 15.01.2024).
3. Building MVP apps. – Режим доступа: https://www.gwtproject.org/articles/mvp-architecture.html. – (дата обращения: 20.01.2024).
4. Hibernate. Everything data. – Режим доступа: https://hibernate.org. – (дата обращения: 04.02.2024).
5. Explain command in PostgreSQL. План выполнения оператора. – Режим доступа: https://www.postgresql.org/docs/current/sql-explain.html. – (дата обращения: 27.01.2024).

УДК 004.01

РЕАЛИЗАЦИЯ ОБУЧАЮЩЕГО ПРИМЕРА МУЛЬТИПЛАТФОРМЕННОЙ ШУТЕР-ИГРЫ С ИСПОЛЬЗОВАНИЕМ UNITY

Н. В. Желтиков

Воронежский государственный университет

Введение

Unity — кроссплатформенная среда разработки, позволяющая реализовать проект, включающий в себя основные функции движка, необходимые для понимания основ программирования игр.

Достоинства Unity:

Unity предоставляет такие возможности, как: моделирование физических сред, преграждение света в пространстве, карты нормалей, динамическую механику отрисовки теней и т.д. Кроме того, Unity предлагает визуальный рабочий процесс и межплатформенную поддержку.

Альтернативные инструменты разработки, как правило, представляют собой набор отдельных фрагментов, требующих контроля, а иногда — библиотеки, для работы с которыми необходима собственная настроенная интегрированная среда разработки. В Unity рабочий процесс привязан к визуальному редактору — в нём будут скомпонованы сцены будущей игры.

При использовании последовательного улучшения редактор очень удобен, т.к. имеется возможность модифицировать и двигать объекты после запуска игры.

Независимость от платформы является итогом изначального предназначения Unity исключительно для системы Mac OS. Первая версия вышла в 2005 году, а к настоящему времени выпущено уже пять основных версий.

Дополнением к ключевым достоинствам идёт особенность конструирования игровых объектов, реализованная в Unity в виде модульной системы компонентов.

Недостатки Unity:

В объёмных сценах могут быть потеряны присоединённые компоненты, т.к. визуальный редактор сопровождается сложным кодом.

Второй недостаток — в Unity нет поддержки внешних библиотек кода, их необходимо копировать в проект вручную.

1. Основы работы с Unity

На вкладке *Scene*, расположенной по центру, можно увидеть, как выглядит игра и способы перемещения объектов по сцене (рис. 1.). Объекты в сцене являются сеточными. Сеточным называют объект, создаваемый из набора соединённых друг с другом линий, которые формируют сетку. Меню *Layouts* позволяет осуществить переход от одного способа компоновки к другому.

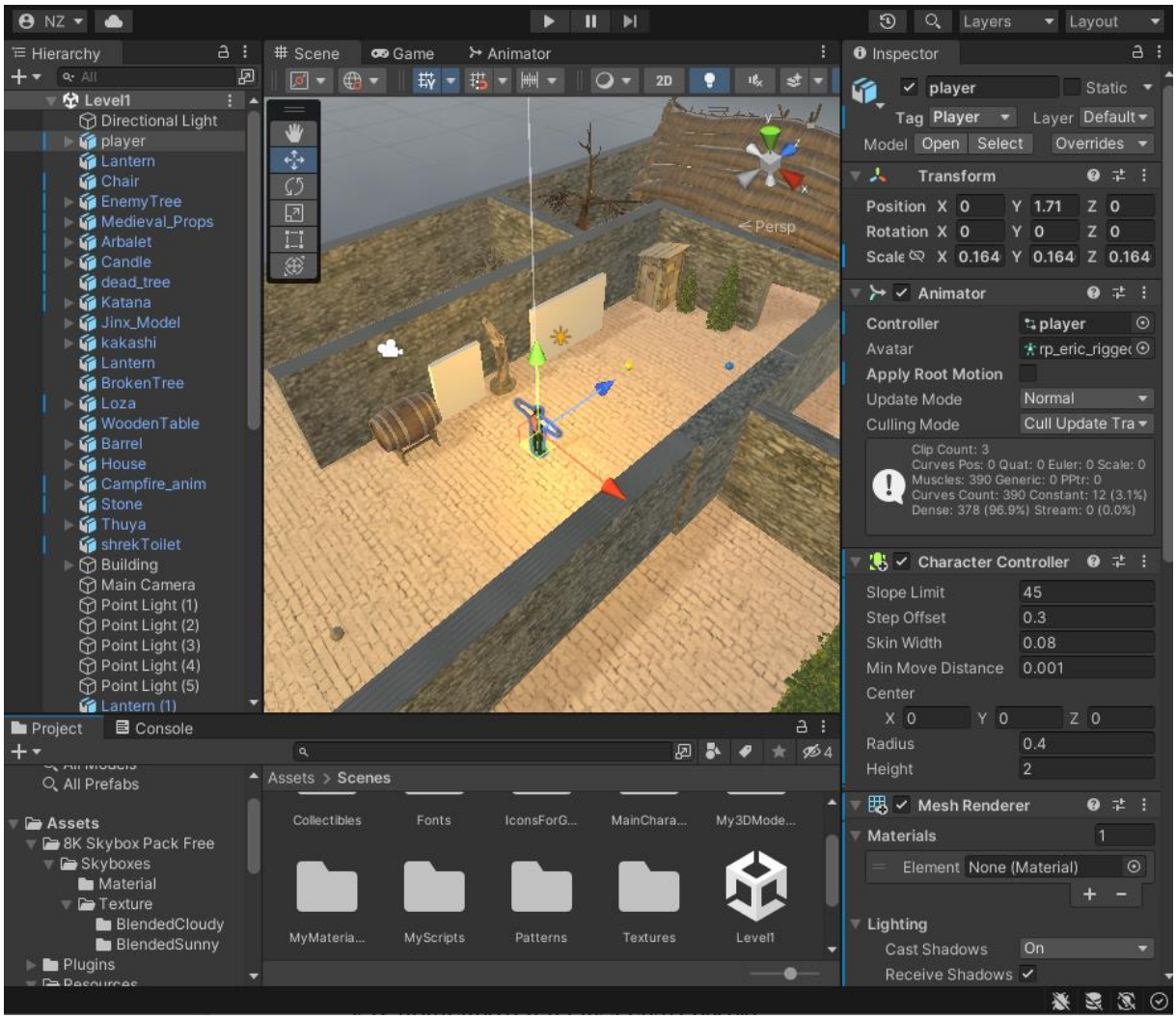


Рис. 1. Начальное окно Unity

Навигация в сценах совершается с помощью мыши или набора клавиш. Основные операции — перемещение, облёт и масштабирование. Выделенный в сцене объект можно перемещать, поворачивать и менять его размер. Перемещение соответствует переносу, облёт — повороту, а масштабирование — изменению размеров.

В левой стороне экрана располагается вкладка *Hierarchy*, в правой — *Inspector*. Список объектов на вкладке представлен в виде древовидной структуры. На вкладке *Inspector* отображается информация о выделенном объекте.

На нижней части экрана расположены вкладки *Project* и *Console*. На вкладке *Project* отображены все ресурсы проекта. В левой части этой панели находится список папок, включённых в проект, при выделении папки отображаются находящиеся в ней файлы. Во вкладку *Console* выводятся связанные с кодом сообщения.

2. Сценарий, свет и камера

Файлы с кодом в терминологии Unity принято называть сценариями. Для запуска кода необходима связь сценария и какого-либо объекта сцены. После создания игрового объекта в каждый набор компонентов может входить исполняемый сценарий. После компиляции код не превращается в самостоятельный исполняемый файл, а выполняется внутри игрового движка. Сценарии, присоединённые к объектам сцены являются экземплярами объектов.

В группу компонентов попадают сценарии, наследующиеся от класса *MonoBehaviour*. Это наследование предоставляет пару стандартных методов, которые следует переопределять: методы Start() и Update(). Start() вызывается при активации объекта, Update() — в каждом кадре. Кадром, или фреймом, является один прогон зацикленного кода.

При создании сценария, Unity в первую очередь копирует стандартный шаблон сценария и класс переименовывается в соответствии с именем файла. Для создания сценария необходимо открыть меню *Assets*, затем *Create* и выбрать команду *C# Script*. Также, можно перетащить файл сценария мышью на произвольный объект сцены для их связывания. Двойной щелчок на значке сценария автоматически откроет его в нужной IDE.

Для создания света необходимо перейти во вкладку *GameObject*, выбрать пункт *Light* и в нём выбрать необходимый осветитель. Существует три различных типа осветителей, проецирующих лучи освещения разными способами — точечный источник, прожектор и направленный источник.

Лучи точечного источника начинаются в одной точке и распространяются во всех направлениях вокруг объекта. Яркость света изменяется при приближении или отдалении от источника за счёт увеличения концентрации лучей. Лучи прожектора исходят из одной точки, но распространяются в пределах ограниченного конуса, в отличие от точечного источника. Лучи направленного источника света распространяются равномерно на протяжении всего пути и параллельно друг другу, одинаково освещая всю сцену.

Для создания камеры необходимо перейти во вкладку *GameObject* и выбрать пункт *Camera*. При работе с игрой от первого лица модель персонажа не играет роли, игрок видеть её не будет. Для того, чтобы объект капсулы вёл себя, как подлинный персонаж, на вкладке *Inspector* необходимо выбрать *Add Component*, перейти в раздел *Physics* и добавить персонажу свойство *Character Controller*.

3. Описание игры

Реализованы осмотр сцены, передвижение, возможность стрельбы персонажа, стрельба и перемещение противников, присвоение текстур и создание 3D-моделей, интерфейса и анимации, присоединение объектам звуков и музыки, и развёртывание игры на платформах Windows, Linux и Mac.

При игре от первого лица предполагаются виды поворота по горизонтали, по вертикали и по диагонали, т.е. комбинированный. Реализация перемещения персонажа аналогична реализации вращения, но отклик идёт не на движение мыши, а на нажатие клавиш WASD. В игре от третьего лица предполагается только горизонтальный тип поворота. Передвижение реализовано при помощи клавиш WASD. В игре с видом сверху движение камеры только горизонтальное, производится с помощью клавиш A/D, а передвижение реализовано с помощью щелчка кнопкой мыши.

Стрельба персонажа реализована с помощью метода бросания лучей. В качестве противника выступит примитив. Для перемещения противников использован метод сферического бросания лучей. В отличие от обычного метода, у сферического радиус распознавания больше, в сравнении с бесконечно тонким лучом, выбрасываемым в первом случае. Стрельба при помощи выпускания лучей мгновенна, т.е. попадание регистрируется в момент щелчка по левой кнопке мыши. Противники же бросают огненные шары, летящие по воздуху и достигающие цели не сразу, в отличие от главного персонажа. Попадания в данном случае фиксируются не методом испускания лучей, а методом распознавания столкновений.

Графическим ресурсом называется единица визуальной информации, которая используется игрой. К графическим ресурсам относятся трёхмерные модели, файлы изображений. Текстурой называется двумерное изображение, накладываемое на какой-либо объект. Трёхмерной моделью является любой объект, присутствующий в сцене. Для текстур используется формат PNG, для 3D-моделей — формат FBX.

Холст — разновидность объектов в Unity, которые визуализируются как игровой интерфейс пользователя. Для работы с пользовательским интерфейсом создано всплывающее окно, на котором расположены все элементы *UI*.

Скелетной анимацией называется процедура связывания набора костей с моделью, осуществляющей движение. После изменения положения кости, перемещается и поверхность модели, связанная с ней. Кости перемещаются как недеформируемые объекты, но окружающая их поверхность может менять свою форму и сгибаться. Сначала определяются анимационные клипы, затем настраивается контроллер, использующийся для воспроизведения этих клипов. Анимация персонажа воспроизводится в соответствии с написанными сценариями движения.

Существует несколько компонентов для работы со звуком: *AudioClip*, *AudioSource* и *AudioListener*. Данное разделение возникло в связи с поддержкой трёхмерного звука, т.к. разные компоненты сообщают информацию о местоположении, использующуюся для управления трёхмерным звуком. Звуки в играх делятся на двумерные и трёхмерные. 3D-звук характерен для трёхмерного моделирования, т.к. он исходит из определённых мест сцены.

Приложение Unity позволяет разворачивать приложения на следующих платформах: Windows, Linux, macOS, IOS, Android, WebGL, Oculus Rift, Steam VR, XBox One, PlayStation, PS Vita, Switch.

3.1. Движение и стрельба

Метод *transform.Rotate(float rotX, float rotY, float rotZ)* поворачивает объект в зависимости от заданных значений в полях координат. Внутри метода *Rotate()* можно реализовать работу и с локальными координатами, и с глобальными. В локальной координатной системе, которая перемещается вместе с объектом, у каждого объекта есть собственная точка начала координат и направления осей. В случае глобальной системы трёхмерная сцена обладает собственными направлениями осей и собственной точкой начала координат.

Для передвижения вместо ответа *MouseX* или *MouseY* возвращается значение *Horizontal* или *Vertical*. Клавиши A и D соответствуют имени *Horizontal*, клавиши W и S соответствуют имени *Vertical*. Значения перемещения меняются по координатам X и Z.

Независимость от кадровой частоты обеспечивается сменой скорости в соответствии с частотой кадров. Решение — умножение скорости персонажа на переменную *deltaTime*. В классе *Time*, который отвечает за урегулирование времени, содержится эта переменная, означающая величину изменения во времени, а именно время между кадрами. При умножении скорости персонажа на эту переменную будет произведено масштабирование, что приведёт к равенству значений скорости на разных устройствах.

Стрельба реализована при помощи бросания лучей. Лучом в сцене называется невидимая линия, начинающаяся в некоторой точке и распространяющаяся в определённом направлении. Формируется луч и затем определяется точка пересечения с объектом. Луч выпускается из точки, в которой находится оружие, после чего летит до момента столкновения

с каким-либо препятствием. В данном случае, луч — путь пули, бросание луча — выстрел из оружия и наблюдение за точкой попадания пули.

3.2. Графика, интерфейс и анимация

Таблица 1.

Игровые графические ресурсы

Тип	Определение типа
Двумерное изображение	Стандартное плоское изображение
Трёхмерная модель	Виртуальные трёхмерные объекты
Материал	Пакет информации, использующийся для определения свойства поверхности какого-любого объекта с присоединённым материалом
Анимация	Пакет информации, который определяет движения и их отображение для связанного объекта
Система частиц	Механизм создания объёмного количества небольших движущихся объектов, использующийся для управления ими

У всех текстур имеется четыре канала. По трём из них поступают видимые цвета: по красному, по зелёному и синему. Четвёртый, альфа-канал, — дополнительный невидимый канал, использующийся для контролирования прозрачности изображения.

Таблица 2.

Форматы двумерных изображений

Формат	Достоинства и недостатки
PNG	Сжатие без потерь, имеется альфа-канала
JPG	Сжатие с потерями, альфа-канал отсутствует
GIF	Сжатие с потерями, альфа-канал отсутствует
BMP	Используется без сжатия, альфа-канал отсутствует
TGA	Используется без сжатия, но предусмотрено и сжатие без потерь, имеется альфа-канал
TIFF	Используется без сжатия, но предусмотрено и сжатие без потерь, альфа-канал отсутствует
PICT	Сжатие с потерями, альфа-канал отсутствует
PSD	Формат *Photoshop*. Используется без сжатия, имеется альфа-канал

Таблица 3.

Форматы файлов трёхмерных моделей

Формат	Достоинства и недостатки
FBX	Поддерживаются сетки и анимация
Collada (DAE)	Поддерживаются сетки и анимация
OBJ	Поддерживаются только сетки, имеют текстовый формат
3DS	Поддерживаются только сетки
DXF	Поддерживаются только сетки
Maya	Работа производится через FBX, требуется установка приложения
3ds Max	Работа производится через FBX, требуется установка приложения
Blender	Работа производится через FBX, требуется установка приложения

Для реализации интерфейса в меню *GameObject* в категории *UI* необходимо выбрать вариант *Canvas*, после чего в сцене появится холст, растянутый на весь экран. Холст представляет собой область, отображающуюся как пользовательский интерфейс. В меню *GameObject* в пункте *UI* имеются команды, позволяющие создать изображение, текст или кнопку.

Для создания анимации персонажа необходимо использование отдельных клипов. Чтобы клип воспроизводился в цикле, необходимо установить флаги Loop Time для зацикливания времени и Loop Pose для повторения анимации.

Контроллер-аниматор — дерево связанных узлов, управление и просмотр которого осуществляются на вкладке Animator. Для создания необходимо в меню Window выбрать команду Animator. На открывшейся панели показываются узлы, принадлежащие выделенному контроллеру. Необходимо соединить узлы друг с другом, чтобы обеспечить переходы между состояниями анимации. Такие узлы контролируют переходы из одного состояния в другое во время игры.

3.3. Звуки и музыка

Таблица 4.

Форматы аудиофайлов

Тип файла	Достоинства и недостатки
WAV	Звуковой файл без сжатия, формат по умолчанию в Windows
AIF	Звуковой файл без сжатия, формат по умолчанию в Mac
MP3	Звуковой файл со сжатием
OGG	Звуковой файл со сжатием
MOD	Формат, используемый музыкальными трекерами
XM	Формат, используемый музыкальными трекерами

Основные отличия форматов заключаются в степени сжатия. Оно уменьшает размер файла за счёт удаления из него части наименее важной информации, из-за этого качество звука остаётся нормальным.

При использовании длительных музыкальных треков расходуется большое количество компьютерной памяти. Необходимо отслеживать два критерия: загрузка музыки в память до момента использования, и потребление большого объёма памяти при загрузке. При помощи метода *Resources.Load()* выполняется оптимизация в процессе загрузки. Загрузка по требованию (*lazy loading*) означает, что загрузка файла в проект запускается только в момент, когда этот файл понадобится.

3.4. Управление в игре от третьего лица

Персонаж в игре от третьего лица поворачивается независимо от камеры. Проверяется, нажаты ли клавиши WASD, если проверка вернула значение true — применяется клавиатурный тип ввода, в обратном случае проверяется указатель мыши. Благодаря независимой проверке вариантов ввода может задаваться собственная скорость вращения в каждой из вариаций.

Положение камеры задано относительно позиции целевого объекта и угла поворота. Умножение вектора position на кватернион даёт смещённое в соответствии с углом поворота положение. Затем новый вектор положения прибавляется к смещению от положения персонажа. Метод *LookAt()* предназначен для нацеливания одного объекта на другой. При применении данного метода к камере, она будет направлена на целевой объект. Для её размещения под нужным углом используется рассчитанное значение поворота.

3.5. Управление в игре с видом сверху

Для реализации перемещения в игре с видом сверху клавиатурный ввод не рассматривается, точка перемещения задаётся щелчком мыши. При этом нет реакции на движение мыши, управление осуществляется только при помощи щелчков. Был использован метод испускания луча, позволяющий определить, какая точка сцены является пунктом назначения. Место попадания луча используется как целевая точка. Также гарантировано, что персонаж будет повёрнут к целевой точке лицом. Поворот блокируется при замедлении персонажа. Далее происходит преобразование задающих направление движения персонажа координат от локальных к глобальным.

В конце проверяется расстояние между персонажем и целью: если персонаж близок к достижению нужной точки, его скорость постепенно будет уменьшаться и при скорости, равной нулю, происходит удаление целевой точки.

3.6. Развёртывание игры

Щелчок по кнопке *Build* открывает окно выбора файла, где необходимо указать путь для генерации пакета приложения. После указания местоположения файла начнётся процесс построения, далее Unity создаст исполняемый файл для платформы, активной в данный момент.

По умолчанию, код может быть запущен на всех платформах одним способом. Однако предоставляется и ряд директив компилятора, запускающих код только на определённой платформе. Директивы существуют для любой платформы, поддерживаемой Unity, следовательно, на каждой из них может быть запущена собственная версия кода.

Заключение

Был реализован обучающий пример мультиплатформенной шутер-игры в кроссплатформенной среде разработки Unity с целью понимания основ разработки игр и было рассмотрено управление персонажем, добавление противников, добавление текстур и 3D-моделей, создание интерфейса, анимации, добавление звуков и фоновой музыки, развёртывание игры на ПК.

Литература

1. Основы Unity. – URL: https://docs.unity3d.com/ru/530/Manual/UnityBasics.html.
2. Unity tutorials point. – URL: https://www.tutorialspoint.com/unity/unity_tutorial.pdf.
3. Unity Best Practices. – URL: https://habr.com/ru/companies/mygames/articles/649687/.
4. Костер Р. Разработка игр и теория развлечений / Р. Костер : пер. с англ. О. В. Готлиб – М. : ДМК Пресс, 2018. – 288 с.
5. Хокинг Дж. Unity в действии. Мультиплатформенная разработка на C# / Дж. Хокинг ; пер. с англ. И. Рузмайкиной. – Санкт-Петербург : Питер, 2018. – 336 с.

УДК 005

Применение детерминационного анализа для анализа несоответствий в системе менеджмента качества

М. С. Жихарев

Воронежский государственный технический университет

Аннотация. Рассматривается возможность применения детерминационного анализа в качестве инструмента выявления несоответствий в системе менеджмента качества. Метод основан на определении корреляций и детерминации между переменными и может быть использован для описания процесса изменения одной переменной под влиянием другой. Преимущества этого метода заключаются в простоте использования, быстром получении визуальных результатов и возможности оперативного управления процессами. Практическое применение детерминационного анализа продемонстрировано на примере изучения социологических данных, что подтверждает универсальность и эффективность данного метода.

Ключевые слова: детерминационный анализ, анализ несоответствий, интенсивность детерминации, корреляция, детерминация, система менеджмента качества.

Введение

Анализ несоответствий — это метод, который компании используют для сравнения своих текущих бизнес-процессов и возможностей с желаемым будущим состоянием. Процедура определения причин несоответствий системы качества установленным требованиям является одним из ключевых разделов Международного стандарта ISO 9001:2015 [1] и связана с рядом обстоятельств, которые влияют на точность получения результата, то есть имеет погрешности, вызванные внутренними и внешними факторами.

Анализ несоответствий имеет большое значение для компаний, так как помогает преодолевать препятствия, связанные с быстрым темпом жизни потребителей, удовлетворением их потребностей, использованием новых технологий и повышением эффективности работы сотрудников. Этот инструмент позволяет компаниям выявить и устранить проблемы в своей деятельности, улучшить производительность и достичь поставленных целей.

Современный менеджмент качества основан на принципе, что деятельность по обеспечению качества не может быть результативной даже когда производственный процесс завершен. Важен не столько контроль качества продукции и услуг, а производственный менеджмент на рабочих местах [1]. Чтобы предотвратить выпуск бракованной продукции, необходимо провести ряд мероприятий по улучшению качества ее изготовления. Один из эффективных инструментов повышения качества изделий является метод анализа видов и последствий потенциальных несоответствий. Анализ позволяет выявить именно те риски, которые обуславливают наибольшие потери потребителя, определить их потенциальные причины и выработать корректирующие мероприятия по их исправлению.

При определении области применения и метода применения анализа в конкретной системе необходимо учитывать: для какой конкретной цели он должен быть использован, и на каком этапе жизненного цикла. Необходимо учитывать уровень знаний о рисках, проблемах и их последствиях. Исходя из этих соображений, можно определить глубину анализа для определенного уровня системы (системы, подсистемы, части, элемента).

Анализ несоответствий заключается в определении потенциальных проблем в конкретной системе в выбранный период времени, чтобы можно было разработать корректирующие или предупреждающие мероприятия. Целями применения могут быть: увеличение безопасность функций и надежности продукции (выявления узких мест); уменьшение гарантийных и сервисных затрат; сокращение процессов разработки; привлечение новых потребителей; сокращение производственных затрат и так далее.

Выявление причин несоответствий искажается появлением погрешностей. Определяемые причины несоответствий и корректно формулируемые на основании этого корректирующие мероприятия способствуют повышению качества продукции (услуги), снижению их себестоимости за счет сокращения издержек, экономии трудозатрат персонала, минимизации повторения несоответствия, предупреждению повторения данных ситуаций в дальнейшем. Использование методов менеджмента качества нацелено на постоянное улучшению конкурентоспособности продукции, повышение лояльности потребителей и других заинтересованных сторон.

Существует большое число методов определения причинно-следственных связей событий, а также вспомогательных методик анализа процессов [2], наиболее известными из которых являются инструменты качества «5 почему» и диаграмма Исикавы «рыбий скелет». Метод «5 почему», как и другие инструменты качества, получивший распространение в крупнейших мировых компаниях, является наиболее простым для понимания, поскольку не требует ни специального программного обеспечения, ни оборудования, ни особых знаний в области менеджмента качества. Для получения достоверного результата считается целесообразным задать вопрос «почему так произошло» порядка пяти раз.

Применение данного метода на российских предприятиях показывает, что не все инструменты качества дают высокий результат, и этому способствует ряд факторов, которые не учитываются в компаниях. При этом в ходе обучения персонала и оказания помощи со стороны отделов качества наблюдается повышение результативности от использования методик.

1. Процесс анализа несоответствий

1.1. Проведение анализа несоответствий

Рассмотрим пошаговое руководство по проведению анализа несоответствий.

1. Определение целей и критериев.

Процесс начинается с четкого определения целей организации и критериев успеха, включает в себя понимание того, чего стремится достичь бизнес, и установление измеримых контрольных показателей.

2. Оценка текущего состояния.

Тщательное изучение текущего состояния организации, включая существующие процессы, технологии и функциональные возможности позволит выяснить, на каком уровне находится организация прямо сейчас. Этот шаг включает подробный анализ повседневных операций.

3. Определение будущего состояния.

Четкое представление о желаемом будущем состоянии, включает в себя определение стратегических целей, контрольных показателей эффективности и ключевых показателей эффективности (КПЭ), которых стремится достичь организация.

4. Определение пробелов.

Сравнение текущего состояния с будущими целями, чтобы выявить пробелы, включает в себя тщательное изучение процессов, технологий и функциональных возможностей, чтобы точно определить области, в которых отсутствует соответствие.

5. Разработка плана действий.

После выявления пробелов разработка подробного плана действий по устранению этих пробелов. Включает в себя определение конкретных стратегий, сроков и ответственности за реализацию.

6. Внедрение изменений.

Выполнение плана действий, внеся необходимые изменения для приведения организации в соответствие с ее целями, включая улучшения процессов, модернизацию технологий или другие стратегические корректировки.

7. Мониторинг и оценка

Постоянное отслеживание и оценка вносимых изменений. Этот шаг гарантирует, что внедренные решения практичны и организация приближается к своим целям [3].

1.2. Преимущества проведения анализа несоответствий

Проведение анализа несоответствий имеет ряд преимуществ.

1. Дорожная карта по улучшению.

Благодаря тщательному выявлению несоответствий анализ несоответствий предлагает точную дорожную карту для улучшения организации. Он выявляет конкретные области, требующие внимания, и позволяет разрабатывать целенаправленные стратегии для повышения общей эффективности.

2. Оптимизированное распределение ресурсов.

Этот анализ облегчает распределение ресурсов, направляя внимание на критические области. Предприятия могут расставлять приоритеты, обеспечивая эффективное использование ресурсов для устранения недостатков. Это способствует повышению экономической эффективности и стратегическому управлению ресурсами.

3. Снижение рисков управления изменениями.

Анализ несоответствий минимизирует риски, связанные с управлением изменениями. Организации могут разрабатывать надежные стратегии управления изменениями на основе всестороннего понимания пробелов и проблем. Это обеспечит более плавный переход и повысит общую адаптивность.

4. Принятие обоснованных решений.

Обеспечивает целостное представление о текущих и будущих целях, помогая руководителям принимать стратегические решения на основе понимания целей и задач организации.

5. Повышение эффективности организации

Устраняя пробелы и приводя процессы в соответствие с целями, анализ несоответствий повышает эффективность организации. Он способствует активному подходу к решению проблем, способствует формированию непрерывного совершенствования в бизнесе.

1.3. Метод анализа несоответствий. Пример использования

Предлагаемый метод основан на применении детерминационного анализа. Метод был использован для исследования вопросов о том, как изменяется отношение студентов к прохождению производственной практики.

В начале календарного 2023 года на кафедре был подготовлен и запущен онлайн-опрос студентов об уровне их удовлетворенности обучением. В опросе принимали участие студенты 2, 3 и 4 курса бакалавриата и 1 и 2 курса магистратуры. Данные для социологического исследования приходят в виде результатов таблицы. Результаты приведены в табл. 1, 2.

Таблица 1

Хотели бы Вы проходить производственную практику уже на 2 курсе?

Да	Нет	Я считаю, что оптимально начинать проведение производственных практик с 3 курса	Да, но, чтобы это было не просто сшивание документов	По желанию студента	Если она давала бы какие-то необходимые навыки, то да
54	30	1	1	1	1
61,4%	34,1%	1,0%	1,0%	1,0%	1,0%
Кол-во студентов - участников опроса					
88 чел.					

Таблица 2

Вы довольны содержанием и условиями прохождения производственных практик?

Да (получили реальные, полезные для будущей работы навыки и умения)	Не совсем (ознакомились с работой, но ничего интересного)	Нет (потеряли время на ненужный, неинтересный труд)	Не проходил (-а)	Не проходил, но расстраивает перспектива	Производственные практики на деле	Ещё не было производственной практики
13	18	13	41	1	1	1
14,8%	20,5%	14,8%	46,6%	1,1%	1,1%	1,1%
Кол-во студентов - участников опроса						
88 чел.						

Оценка условной вероятности или условная эмпирическая частота является наиболее приемлемой мерой установления связи между конкретными признаками и вычисляется следующим образом:

$$P(y/x) = \frac{N_{x_i y_j}}{N_{x_i}} \quad (1)$$

здесь x_i, y_j x_i, y_j– весовые коэффициенты.

У каждой детерминации существуют две условные частоты – характеристики этих детерминаций – интенсивность и емкость [4].

Интенсивность детерминации $a \to b$, где $a = x_i, b = y_j$ вычисляется по формуле 2:

$$I(x_i \to y_j) = P(y/x) = \frac{N_{x_i y_j}}{N_{x_i}} \quad (2)$$

Интенсивность детерминации отражает ее точность/истинность. Она показывает, что

среди респондентов, обладающих признаком *a*, какая-либо доля демонстрируют поведение *b* [4].

Шаг 1. Произведем расчеты интенсивности, вопрос *«Хотели бы Вы проходить производственную практику уже на 2 курсе?».*

Для переменной *«да»:*

$$I(x_i \to y_j) = \frac{54}{88} = 0{,}61$$

Для переменной *«нет»:*

$$I(x_i \to y_j) = \frac{30}{88} = 0{,}34$$

Для переменной *«я считаю, что оптимально начинать проведение производственных практик с 3 курса»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Для переменной *«да, но, чтобы это было не просто сшивание документов»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Для переменной *«по желанию студента»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Для переменной *«если она давала бы какие-то необходимые навыки, то да»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Шаг 2. Произведем расчеты интенсивности, вопрос *«Вы довольны содержанием и условиями прохождения производственных практик?».*

Для переменной *«да (получили реальные, полезные для будущей работы навыки и умения)»:*

$$I(x_i \to y_j) = \frac{13}{88} = 0{,}15$$

Для переменной *«не совсем (ознакомились с работой, но ничего интересного)»:*

$$I(x_i \to y_j) = \frac{18}{88} = 0{,}20$$

Для переменной *«нет (потеряли время на ненужный, неинтересный труд)»:*

$$I(x_i \to y_j) = \frac{13}{88} = 0{,}15$$

Для переменной *«не проходил (-а)»:*

$$I(x_i \to y_j) = \frac{41}{88} = 0{,}47$$

Для переменной *«не проходил, но расстраивает перспектива»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Для переменной *«производственные практики на деле»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Для переменной *«ещё не было производственной практики»:*

$$I(x_i \to y_j) = \frac{1}{88} = 0{,}01$$

Шаг 3. Сравним полученные расчеты и определим интересующие нас переменные. Результаты расчетов интенсивности детерминации приведены в табл. 3, 4.

Таблица 3

Хотели бы Вы проходить производственную практику уже на 2 курсе?

Да	Нет	Я считаю, что оптимально начинать проведение производственных практик с 3 курса	Да, но, чтобы это было не просто сшивание документов	По желанию студента	Если она давала бы какие-то необходимые навыки, то да
54	30	1	1	1	1
61,4%	34,1%	1,0%	1,0%	1,0%	1,0%
Интенсивность детерминации					
0,61	0,34	0,01	0,01	0,01	0,01

Таблица 4

Вы довольны содержанием и условиями прохождения производственных практик?

Да (получили реальные, полезные для будущей работы навыки и умения)	Не совсем (ознакомились с работой, но ничего интересного)	Нет (потеряли время на ненужный, неинтересный труд)	Не проходил (-а)	Не проходил, но расстраивает перспектива	Производственные практики на деле	Ещё не было производственной практики
13	18	13	41	1	1	1
14,8%	20,5%	14,8%	46,6%	1,1%	1,1%	1,1%
Интенсивность детерминации						
0,15	0,20	0,15	0,47	0,01	0,01	0,01

Шаг 4. Определим зависимости (корреляции) между отдельными значениями. Результаты приведены в табл. 5.

Таблица 5

Сравнение значений при разных уровнях интенсивности детерминации

Хотели бы Вы проходить производственную практику уже на 2 курсе?		Вы довольны содержанием и условиями прохождения производственных практик?	
Да	Нет	Да (получили реальные, полезные для будущей работы навыки и умения)	Не совсем (ознакомились с работой, но ничего интересного)
0,61	0,34	0,15	0,20

По результатам исследования можно сделать вывод, что большинство студентов (0,61) хотят проходить производственную практику уже на втором курсе, однако только (0,15) довольны полученным опытом и навыками, в то время как (0,20) считают практику ознакомительной и неинтересной.

Сравнение значений при разных уровнях интенсивности детерминации позволяет определить возможные зависимости или корреляции между отдельными значениями. В данном случае сравнение показателей отношения к прохождению практики на 2 курсе с уровнем удовлетворенности практикой позволяет установить, что более высокая доля студентов, желающих пройти практику на 2 курсе, соответствует более низкой интенсивности детерминации у тех, кто доволен содержанием и условиями практики.

Заключение

Применение детерминационного анализа позволило исследовать изменение отношения студентов к производственной практике и выявить основные проблемы и тенденции. Выводы, полученные из данного исследования, говорят о том, что существует разнообразие мнений и предпочтений среди студентов относительно прохождения производственной практики. Большинство студентов хотят проходить практику уже на втором курсе, что может свидетельствовать о высоком интересе и значимости данного опыта для будущей профессиональной деятельности, но только (0,15) довольны полученным опытом и навыками, а (0,20) считают практику ознакомительной и неинтересной. Эти результаты указывают на необходимость улучшения содержания условий практик и повышения качества образовательных программ, чтобы повысить удовлетворённость студентов и улучшить их подготовку к будущей профессиональной деятельности.

При проведении анализа качественных данных важно учитывать специфику данных и применять подходящие методы их обработки. Детерминационный анализ позволяет получить более точные результаты и обоснованные выводы исследования.

Это позволяет более глубоко и точно оценивать взаимосвязи и закономерности в данных, что, в свою очередь, способствует более точной интерпретации результатов социологических исследований.

Литература

1. ГОСТ Р ИСО 9001:2015. Система менеджмента качества. Общие требования:

дата введения 2015–11–01 / Приказом Федерального агентства по техническому регулированию и метрологии // Техэксперт : Электронный фонд правовой и нормативно-технической документации. – URL: https://clck.ru/3A5Umn.

2. Давыдов, В. М. Методы определения причин несоответствий процесса установленным требованиям / В. М. Давыдов, В. И. Шкробова // Информационные технологии XXI века : сборник научных трудов. – Хабаровск : Тихоокеанский государственный университет, 2023. – С. 281-288.

3. Изучение значимости анализа несоответствий (примеры + шаблоны). – Режим доступа: https://slidemodel.com/fit-gap-analysis/ – (Дата обращения: 19.04.2024).

4. Чесноков С. В. Детерминационный анализ / С. В. Чесноков ; под ред. Е. С. Райской. – Москва : Наука. Главная редакция физико-математической литературы, 1982. – 168 с.

УДК 004.94

АНАЛИЗ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ РЕАЛИЗАЦИИ ДОСТУПНЫХ ИНТЕРФЕЙСОВ ВЕБ-САЙТОВ

Д. А. Жуков

Воронежский государственный университет

Введение

Рассматривается проблема создания веб-приложений, адаптированных для использования людьми с ограниченными возможностями здоровья, которые пользуются различными ассистивными технологиями, в том числе проблема создания доступного интерфейса. Для продукта, ориентированного на международный рынок, доступность — часть законодательства, касающегося цифровых сервисов.

1. Основные принципы разработки доступных интерфейсов

Консорциум Всемирной паутины декларирует следующие основные принципы разработки доступных интерфейсов:

- воспринимаемость;
- управляемость;
- понятность;
- надежность.

2. Доступный дизайн

Визуальное обозначение элементов сайта, с которыми пользователь может взаимодействовать — неотъемлемая часть любого дизайна. Для определенного класса пользователей резкая смена цвета графического элемента страницы, например, может навредить здоровью. Чтобы среагировать на смену цвета элемента веб-сайта, при наведении на него курсора, в среднем человеку нужно 200–300 миллисекунд. Язык стилей CSS позволяет регулировать скорость срабатывания анимации. Некоторые пользователи ослабевают динамику элементов или повышают контрастность текста средствами операционной системы. В 2021 году появилась возможность через CSS определять пользователей, у которых включены эти режимы через медиазапросы.

3. Дерево доступности и ARIA-инструменты

Помимо DOM-дерева, браузер строит дерево доступности, оно помогает пользователям с ограниченными возможностями лучше понять структуру и содержимое веб-страницы, а также позволяет поисковым роботам и специальным программам более точно анализировать и индексировать информацию на странице. Это полезный инструмент контроля при разработке доступных веб-приложений. Открыть дерево доступности можно в инструментах разработчика в браузере.

Каждый HTML-элемент на странице описан в дереве доступности через набор параметров (рис. 1):

- роль элемента — трактовка элемента со стороны браузера. Например, у заголовков роль «heading». При написании семантически правильной разметки в дереве доступности элементам будут прописываться правильные роли. Семантически правильной разметка считается при использовании подходящих контейнеров из семейства контейнеров стандарта HTML5, неправильным считается для всех блоков использовать тег «div». Примеры самых распространенных семантических контейнеров: «header», «main», «footer», «aside», «nav»;
- имя элемента — оно есть не всегда. Например, у заголовков или кнопок имя элемента автоматически берётся из текста внутри них, но у секций или абзацев — нет;
- описание — для лучшего понимания значения элемента его можно снабжать дополнительным описанием (это можно сделать с помощью ARIA-инструментов, об этом ниже).

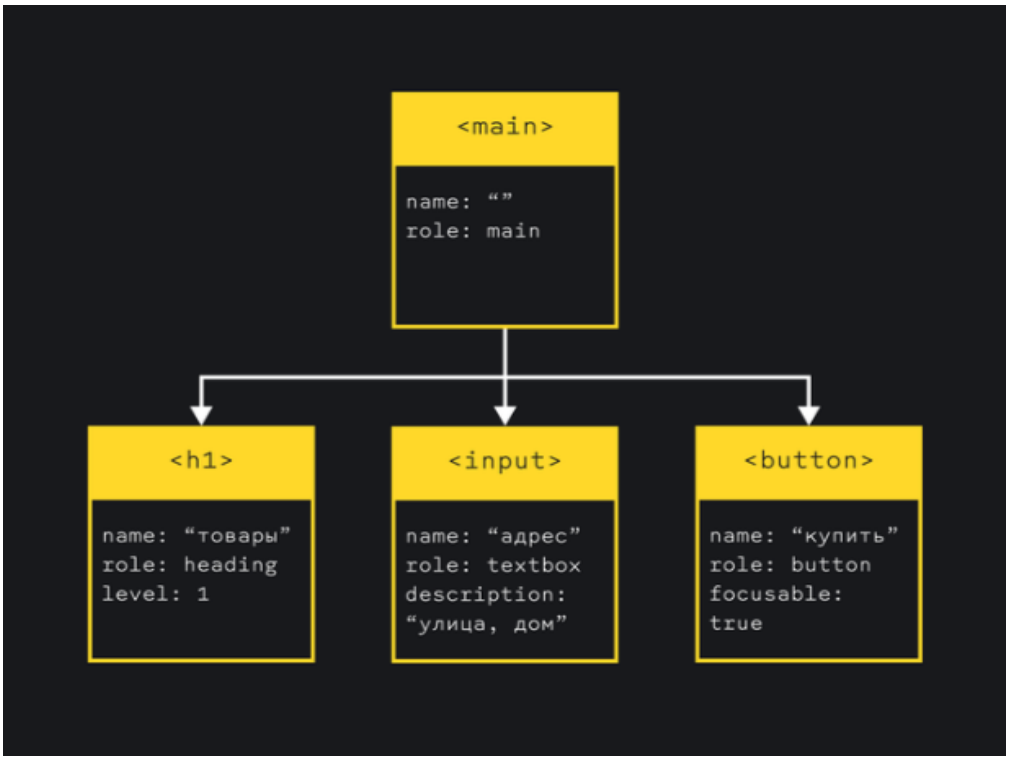


Рис. 1. Дерево доступности

ARIA (Accessible Rich Internet Applications) — дополнительные инструменты, которые позволяют сделать сайт более доступным. ARIA-инструменты разделяются на два класса: ARIA-роли, ARIA-атрибуты.

ARIA-роли позволяют самостоятельно прописать роль HTML-элементу.

ARIA-атрибутов большое количество, некоторые из них:

- атрибут «aria-hidden» со значением «true» позволяет скрывать элементы из дерева доступности, но оставлять их видимыми на странице, например, скрытие от пользователей скринридеров декоративных элементов, которые не содержат никакой важной информации;
- атрибут «aria-label» добавляет элементу имя;

- атрибут «aria-description» позволяет добавлять дополнительное описание элементу, которое озвучит скринридер.

4. Требования к скрытому меню

Скрытое меню представляется в виде значка, при щелчке на котором, появляется список возможных действий. В рамках разработки скрытого меню, которое будет доступно для людей с нарушениями зрения, необходимо выполнить несколько требований:

- посетитель веб-сайта, пользующийся скринридером, должен понимать, что значок отвечает за открытие меню;
- все остальные элементы сайта вне меню должны становиться недоступными для взаимодействия;
- фиксация статуса меню с помощью ARIA-инструментов (закрыто меню или открыто);
- возможность закрыть меню по клику на кнопке, по клику мыши вне его области и по нажатию клавиши «Esc».

5. Реализация скрытого меню

Чтобы сделать недоступными все элементы сайта, которые находятся вне открытого меню, используется тег «dialog». Тег «dialog» был внедрен в HTML для реализации всплывающих окон и для данной задачи он идеально подойдет, так как будет отображаться на самом верхнем слое, не давая пользователю возможность взаимодействовать с остальными элементами сайта. Из-за того, что содержимое кнопки – значок, имени у кнопки в дереве доступности нет, поэтому с помощью ARIA-атрибута «aria-label» устанавливается имя кнопки «открыть меню». Атрибутом «aria-controls» указывается ID элемента «dialog», которым управляет кнопка. Статус меню (свернуто оно или нет) фиксируется с помощью атрибута «aria-expanded» на кнопке с начальным значением «false» и меняться на «true» при открытии меню. Меню также присваивается соответствующая роль — «menu». Возможность закрывать меню тремя способами реализована с помощью языка программирования JavaScript (рис. 2, рис. 3).

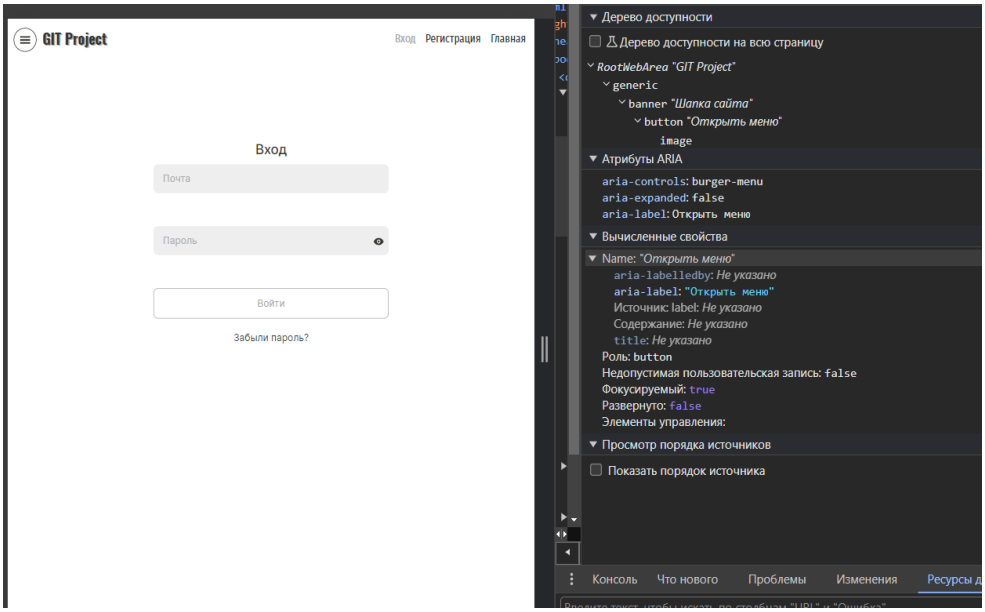


Рис. 2. Закрытое меню и описание кнопки его открытия в дереве доступности

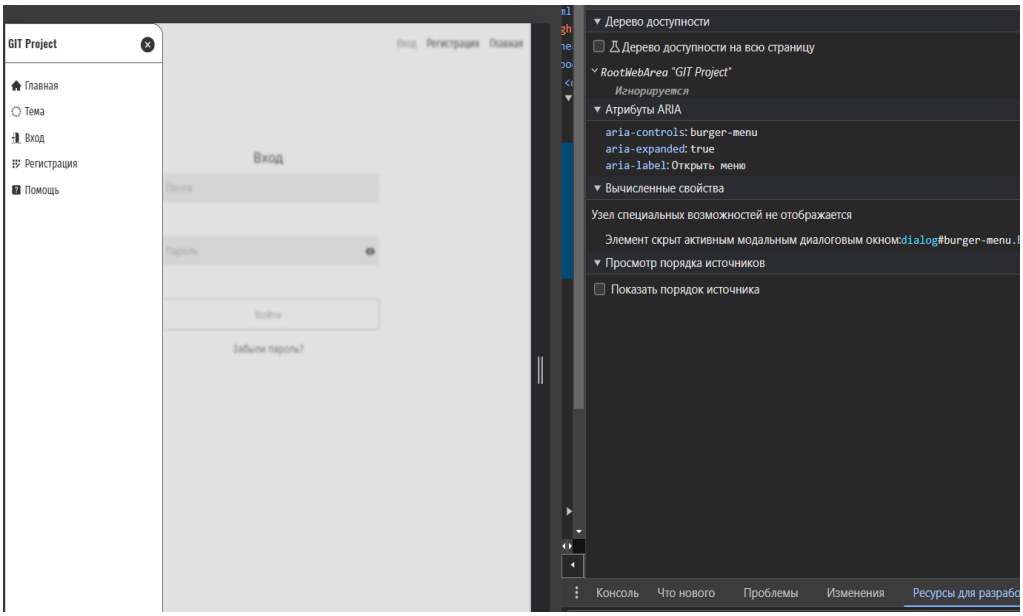


Рис. 3. Меню открыто, кнопка открытия меню пропала из дерева доступности, так как сайт вне меню стал недоступен

Заключение

Рассмотрены инструменты и принципы, которые позволяют адаптировать веб-сайты для людей с ограниченными возможностями здоровья. С учетом поставленных требований разработано скрытое меню, которым могут пользоваться люди с нарушениями зрения.

Литература

1. Руководство по обеспечению доступности веб-контента (WCAG) 2.0. — Режим доступа: https://www.w3.org/Translations/WCAG20-ru/. — (Дата обращения: 10.03.2024).
2. Фрэйн Б. Отзывчивый дизайн на HTML5 и CSS3 для любых устройств / Б. Фрэйн — 3. — Санкт-Петербург: Питер, 2022 — 336 с.
3. Дока. — Режим доступа: https://doka.guide/. — (Дата обращения: 09.03.2024).

УДК 004.031.42

АНАЛИЗ МОДЕЛЕЙ ДЛЯ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

А. В. Загоровский

Воронежский государственный университет

Введение

Рекомендательные модели – это компьютерные системы и алгоритмы, которые анализируют данные о предпочтениях, поведении или характеристиках пользователей, в зависимости от предметной области [1]. Например, это может быть фильм, товар, книга, новость или реклама. На основе этих данных они предоставляют персонализированные рекомендации.

В современном информационном обществе объем информации растет в геометрической прогрессии, и поэтому нам нужны эффективные методы ее фильтрации. Именно в этом контексте рекомендательные системы играют ключевую роль. Они стремятся наиболее точно предсказать предпочтения пользователей и предложить наиболее подходящие товары или услуги. Сегодня такие системы используются практически на каждой крупной онлайн-платформе, такой как интернет-магазин, онлайн-кинотеатр или социальная сеть.

Однако важно отметить, что существует множество различных моделей рекомендательных систем, каждая из которых имеет свои преимущества и недостатки. Необходимо провести анализ различных моделей рекомендательных систем, учитывая их специфические характеристики, такие как точность предсказаний, масштабируемость, вычислительные затраты и адаптивность к изменяющимся потребностям пользователей и рынка. Такой анализ поможет определить наиболее подходящие модели для конкретных ситуаций и задач, а также выявить возможности для улучшения и оптимизации существующих систем.

1. Постановка задачи

Определение обозначений: введение обозначений для структурирования информации и упрощения дальнейшего анализа данных [2]. Необходимо определить следующие обозначения:

- U – множество субъектов (users/пользователей/клиентов);
- I – множество объектов (items/предметов/товаров/ресурсов);
- Y – пространство описаний транзакций (например, пользователя оценил какой-либо объект);
- $D = (u_t, i_t, y_t)_{t=1}^{m} \in U \times I \times Y$ – транзакционные данные.

Агрегирование данных: представление данных в виде матрицы кросс-табуляции $R = \|r_{ui}\|$ размера $|U| \times |I|$, где каждый элемент $r_{ui} = aggr\{(u_t, i_t, y_t) \in D | u_t = u, i_t = i\}$ соответствует оценке, данной пользователем u к объекту i;

Задачей является формирование списка рекомендаций для u и для i.

Использование функций близости: Рассмотрение различных метрик близости, таких как корреляция Пирсона и косинусная мера, для определения сходства между пользователями или объектами. Эти метрики используются в корреляционных методах, таких как коллаборативная фильтрация.

- корреляция Пирсона:

$$sim(u,v) = \frac{\sum_{i \in I(u,v)} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I(u,v)} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I(u,v)} (r_{vi} - \bar{r}_v)^2}};$$

- косинусная мера близости:

$$sim(u,v) = \frac{\sum_{i \in I(u,v)} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I(u,v)} r_{ui}^2 \sum_{i \in I(u,v)} r_{vi}^2}};$$

2. Модели рекомендательных систем

Рекомендательные системы отличаются тем, как они фильтруют данные для прогноза. Это означает, что рекомендация основывается на данных, которые соответствуют определенным критериям. Существует четыре модели такой фильтрации, и в зависимости от них определяются типы рекомендательных систем [3].

- коллаборативная фильтрация (collaborative filtering);
- контентная фильтрация (content-based filtering);
- фильтрация на основе о знаниях (knowledge-based filtering);
- гибридная система (hybrid filtering).

2.1. Коллаборативная фильтрация

Рекомендательные системы с коллаборативной фильтрацией анализируют данные о поведении пользователей с похожими интересами, чтобы предоставить персонализированные рекомендации. В нашем случае, система анализирует не сами товары, а данные о пользователях, которые просматривают эти товары. Если другие пользователи с похожими интересами просматривали и приобретали другие модели с похожими характеристиками, то система склонна предложить эти модели и текущему пользователю.

Этот подход позволяет системе учитывать предпочтения и поведение пользователей, основываясь на сходстве их интересов. Анализируя большой объем данных о поведении пользователей, система может предложить персонализированные рекомендации, принимая во внимание не только характеристики конкретного товара, но и предпочтения пользователей с похожими интересами.

Есть два основных подхода в коллаборативной фильтрации[1]:

1. Корреляционные модели (Memory-Based Collaborative Filtering).
   Явное хранение, как правило, огромной разреженной матрицы исходных данных и работа со строчками, столбцами этой матрицы
2. Латентные модели (Latent Models for Collaborative Filtering)
   Скрытое признаковое описание как пользователей, так и объектов, относительно небольшое векторное представление и далее идёт сравнивание пользователей и объектов – сходство их векторов.

Корреляционные модели, в свою очередь, делятся на:

- основанные на пользователях (user-based);

- основанные на предмете рекомендации (item-based).

Рассмотрим коллаборативную систему, основанную **на пользователях** (user-based). В данной системе алгоритм находит близких по предпочтениям пользователей и рекомендует одному из них то, что уже пробовал другой. Работает по принципу «пользователи, похожие на u_0, также интересовались данным объектом»

1. $U(u_0) := \{u \in U \mid sim(u_0, u) > \alpha\}$ – коллаборация;

   $sim(u_0, u)$ – одна из возможных мер близости u к u_0;
2. $I(u_0) := \{i \in I \mid B(i) = \dfrac{|U(u_0) \cap U(i)|}{|U(u_0) \cup U(i)|} > 0\}$;

   где $U(i) := \{u \in U \mid r_{ui} \neq \varnothing\}$;
3. Отсортировать $i \in I(u_0)$ по убыванию $B(i)$ и взять $top\ N$;

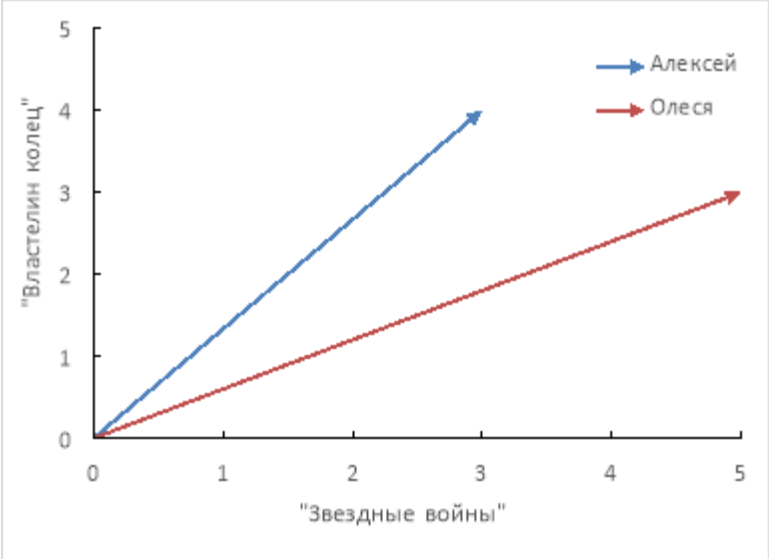


Рис. 1. Векторное представление данных (user-based)

Основной недостаток – отсутствие рекомендаций для нетипичных или новых пользователей, а также проблема «холодного старта» и необходимость хранить всю матрицу R.

Системы основанные **на предмете рекомендации** (item-based), сравнивают непосредственно близость объектов. Сходство определяется на основе предпочтений всех пользователей, которые оставили свои оценки. Работает по принципу «вместе с объектами которые покупал u_0, часто покупают $I(u_0)$»

1. $I(u_0) := \{i \in I \mid \exists i_0 : r_{u_0 i_0} \neq \varnothing\ u\ B(i) = sim(i, i_0) > \alpha\ \}$;

   где $sim(i, i_0)$ – одна из возможных мер сходства i и i_0;
2. Сортировка $i \in I(u_0)$ по убыванию $B(i)$, взять $top\ N$;

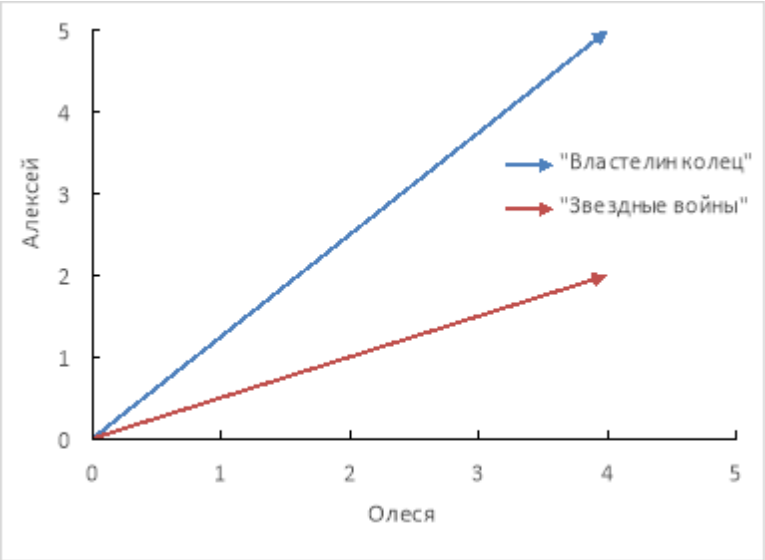


Рис. 2. Векторное представление данных (item-based)

Данная система помогает избавиться от проблемы отсутствия рекомендаций для новых пользователей, но сохраняется проблема «холодного старта» и также необходимость хранить всю матрицу R. Помимо этого появляется недостаток того, что рекомендации часто тривиальны (нет коллаборативности).

2.2. Контентная фильтрация

Рекомендательные системы, основанные на контентной фильтрации, являются наиболее простыми. При подборе рекомендаций они учитывают сходство объектов, услуг, товаров или контента. Если рекомендательная система онлайн-кинотеатра использует контентную фильтрацию, то при просмотре страницы определенного фильма мы увидим похожие фильмы. Например, это могут быть фильмы одного жанра, с одинаковыми актерами в главных ролях или снятые одним режиссером.

Контентная фильтрация в рекомендательных системах обычно основана на векторном представлении контента и профилей пользователей. Для каждого элемента контента, допустим фильма, и для каждого пользователя строятся векторы, где каждая компонента вектора соответствует какому-либо признаку или характеристике контента. Для примера, вектор для фильма может содержать информацию о его жанре, актерах, режиссёре, годе выпуска и т.д. Признаки могут быть бинарными (наличие или отсутствие определённого жанра), числовыми (рейтинг фильма) или категориальными (список актеров, режиссёров).

Таким образом, система, используя ключевые характеристики объектов и профиль пользователя, составляет рекомендации. Данные заполняются на основе взаимодействия пользователя с системой. Вследствие этого возникает проблема «холодного старта», так как о новых пользователях нет никакой информации. Поэтому при первой регистрации на каком-либо сайте пользователям часто предлагается пройти опрос о своих интересах. Основным недостатком такой системы является то, что пользователи не будут пробовать новый контент, так как будут получать рекомендации только в определенных категориях, которые их интересуют.

Преимуществами данной системы будут несложная реализация и отсутствие необходимости в хранении большого количества данных.

2.3. Фильтрация на основе о знаниях

Если система использует фильтрацию на основе знаний, она будет подбирать рекомендации на основе данных, которые объединены какой-то общей темой или интересами. Например, если определенный фильм часто просматривается вместе с другими фильмами из того же жанра, с теми же актерами или режиссером, система может предложить пользователю аналогичные фильмы.

Разработка данной рекомендательной системы является более трудозатратой. Это связано с необходимостью анализа различных характеристик фильмов и их связей между собой, что требует более сложных алгоритмов и обработки данных.

Такие рекомендательные системы встречаются реже, поскольку их разработка требует больших усилий и ресурсов. Однако, когда они реализованы правильно, они могут предлагать пользователю более точные и персонализированные рекомендации, основанные на глубоком понимании характеристик и взаимосвязи между контентом.

2.4. Гибридная система

У всех ранее рассмотренных моделей есть определенные недостатки. Для решения большинства из них были придуманы комбинированные системы, которые используют сразу две модели фильтрации, перечисленные выше. Эти системы могут работать по-разному:

- по отдельности с последующим объединением результата;
- с опорой на одну и какими-то правилами другой фильтрации;
- сочетание двух типов фильтраций.

Например, в онлайн-кинотеатре в рекомендациях пользователю показываются фильмы, похожие на те, которые он уже смотрел и хорошо оценил. Также система учитывает предпочтения пользователей, которые смотрели похожие фильмы. Таким образом, система использует механизмы отбора на основе контента и коллаборативной фильтрации одновременно.

Крупные онлайн-платформы часто используют гибридные системы, потому что они выдают наиболее подходящие рекомендации, которые нравятся пользователям. Однако недостатком таких систем является сложность их разработки.

Заключение

В ходе анализа были рассмотрены основные модели рекомендательных систем: коллаборативная фильтрация, контентная фильтрация, фильтрация на основе знаний и гибридные системы. Каждая из этих моделей обладает своими уникальными особенностями, которые могут быть использованы в различных контекстах.

Результаты анализа позволяют выделить сильные стороны каждой модели и определить области, где их применение может быть наиболее эффективным [4]. Однако, важно учитывать, что выбор подходящей модели зависит не только от самой модели и ее характеристик, но и от специфики задачи, целевой аудитории и доступных ресурсов для ее реализации.

Таким образом, гибкость и адаптивность в выборе модели являются ключевыми аспектами для обеспечения успешной реализации рекомендательных систем. Результаты анализа моделей из данной статьи будут использованы в работе над дипломом по теме «Разработка веб-приложения трекера фильмов с персональными рекоменациями».

Литература

1. Мюллер А. Введение в машинное обучение с помощью Python / Мюллер А., Гвидо С. – Москва: 2017 г. – 393 с.
2. Машинное обучение. Рекомендательные системы. К.В. Воронцов, Школа анализа данных, Яндекс. – Режим доступа: https://youtu.be/J-QueLndVI8?si=dGVevNSBry3DSfCf – (Дата обращения: 17.04.24)
3. Фальк К. Рекомендательные системы на практике / пер. с англ. Д. М. Павлова. – М.: ДМК Пресс, 2020. – 448 с.
4. Оценка точности рекомендательных систем: исследование, проблемы и перспективы. – Режим доступа: https://na-journal.ru/5-2023-informacionnye-tekhnologii/5088-ocenka-tochnosti-rekomendatelnyh-sistem-issledovanie-problemy-perspektivy – (Дата обращения: 18.04.24)

УДК 004.056.5

Обнаружение скрытых вредоносных программ в дампах памяти: подход машинного обучения для повышения уровня кибербезопасности.

А. А. Захарова

Воронежский государственный университет

Аннотация: В сфере кибербезопасности обнаружение и анализ скрытых вредоносных программ остаются серьезной проблемой, особенно в контексте дампов памяти. В этом статье описана платформа на основе машинного обучения, предназначенная для расширения возможностей обнаружения и анализа таких неуловимых угроз для двоичных и многотипных вредоносных программ. Подход использует комплексный набор данных, включающий безопасные и вредоносные дампы памяти, охватывающий широкий спектр скрытых типов вредоносного ПО, включая шпионское ПО, вирусы-вымогатели и троянские кони с их подкатегориями. Используются строгие методы предварительной обработки данных, включая нормализацию дампов памяти и кодирование категориальных данных. Для решения проблемы классового дисбаланса используется метод синтетической передискретизации, обеспечивающий сбалансированное представление различных типов вредоносного ПО. Суть всей структуры заключается в использовании классификатора на основе ансамбля, выбранного из-за его надежности и эффективности при обработке сложных структур данных.

Ключевые слова: Обнаружение запутанных вредоносных программ, Анализ дампа памяти, Расширенная аналитика вредоносных программ, Поведенческие модели вредоносных программ, Расширенная аналитика вредоносных программ, Машинное обучение в области кибербезопасности.

Введение

Скрытое вредоносное ПО - это сложная киберугроза, которая использует методы обхода, чтобы скрыть свое присутствие, что делает ее особенно сложной для обнаружения обычными методами защиты. Этот тип вредоносных программ, умеющий маскироваться под обычные вычислительные операции, представляет серьезную угрозу для цифровых систем. Исследование сосредоточено на разработке передовых методологий для эффективного выявления и анализа этих скрытых угроз, особенно в дампах памяти, где, как известно, они умело маскируют свою деятельность [1-3].

Решение проблемы обнаружения скрытых вредоносных программ крайне важно в современной цифровой экосистеме, где зависимость от технологий достигла небывало высокого уровня [4]. В эпоху, когда данные стали новой валютой, а цифровые взаимодействия лежат в основе большинства наших повседневных действий, потенциальное воздействие вредоносных программ огромно и многогранно. Угрозы, исходящие от необнаруженных вредоносных программ, - от компрометации персональных данных до нарушения работы критически важной инфраструктуры - могут привести к значительным финансовым последствиям, а также к нарушению конфиденциальности и безопасности. Поскольку

цифровые технологии продолжают развиваться и все глубже интегрируются в различные аспекты жизни, обеспечение надежных механизмов защиты от таких скрытых киберугроз становится не просто технической необходимостью, а краеугольным камнем для поддержания доверия и целостности в цифровом мире [5,6].

Решение проблемы обнаружения скрытого вредоносного ПО представляет собой сложную задачу из-за постоянно меняющейся природы методов вредоносного ПО. Эти сложные угрозы предназначены для динамического изменения своего кода или внешнего вида, что позволяет эффективно обходить традиционные системы обнаружения на основе сигнатур. Кроме того, из-за огромного объема и разнообразия вредоносных программ, а также стремительного технологического прогресса становится все труднее поддерживать современные и эффективные методы обнаружения. Эта сложность еще больше возрастает при анализе памяти, где различие между доброкачественными и вредоносными действиями требует тонкого понимания и расширенных аналитических возможностей, поскольку вредоносное ПО часто работает, имитируя законные процессы [7-9]. Следовательно, чтобы оставаться впереди в этой гонке вооружений в области кибербезопасности, требуются постоянные инновации и адаптация стратегий обнаружения.

В исследованиях применяется многогранный подход к решению проблемы обнаружения скрытых вредоносных программ. Используются передовые алгоритмы машинного обучения, в частности, классификаторы, повышающие градиент, для анализа и интерпретации дампов памяти, в которых такие вредоносные программы часто скрыто находятся. Методология включает всестороннюю предварительную обработку данных, балансировку классов с использованием метода синтетической избыточной выборки меньшинства (SMOTE) и тщательный отбор признаков с помощью статистических тестов и показателей получения информации. Эта стратегия позволяет эффективно выявлять тонкие закономерности и аномалии, указывающие на скрытое вредоносное ПО, обеспечивая надежную основу для его идентификации и анализа, выходящего за рамки возможностей традиционных систем обнаружения.

Подход, описывающийся в данной статье, отличается, прежде всего, непревзойденной точностью в обнаружении скрытого вредоносного ПО, обеспечивая замечательную 100%-ную точность по всем показателям оценки как для бинарных, так и для многоклассовых классификаций. Этот уровень точности беспрецедентен в данной области и представляет собой значительное преимущество по сравнению с существующими методами. Интеграция классификаторов с ускорением градиента в сочетании со сложными методами предварительной обработки данных и выбора функций позволяет этой системе выявлять даже самые искусно замаскированные вредоносные программы. Такая высокая точность обеспечивает надежную защиту в различных цифровых средах, значительно снижая риск необнаруженного проникновения вредоносных программ. Кроме того, адаптивность этой модели как к бинарным, так и к многоклассовым типам вредоносных программ демонстрирует ее универсальность и эффективность в борьбе с широким спектром угроз в кибербезопасности, что делает ее ценным инструментом в меняющемся ландшафте цифровой безопасности. В (табл. 1) представлен краткий список сокращений, используемых в моей статье, для большей ясности и понимания.

Таблица 1

Сокращения, используемые в данной статье

Краткая форма	**Аббревиатуры**
CA	Pearson correlation analysis (Корреляционный анализ Пирсона)

DCNN	Dilated CNN (Расширенный канал CNN)
ACC	Accuracy (Аккуратность)
PR	Precision (Точность)
RE	Recall (Отзыв)
FS	F1-score (Формула – результат)
FPR	False positive rate (Частота ложных срабатываний)
ER	Error rate (Частота ошибок)
CK	Cohen’s kappa
AUC	Area under the ROC curve (Площадь под кривой ROC)
RF	Random forest ensemble (Случайный forest ensemble)
BG	Bagging ensemble (Объединение Ансамблей)
VT	Voting ensemble
ADB	AdaBoost ensemble
GB	Gradient boosting ensemble (Повышение градиента)
SMOTE	Synthetic minority over-sampling technique (Метод передискретизации синтетического меньшинства)
OCC	One-class classifier (Классификатор для одного класса)

Структура этой статьи такова: начинается с обзора литературы, изучая предыдущие исследования и теории, имеющие отношение к данной статье. За этим разделом следует разработка предлагаемой методологии, где описывается развитие и тонкости модели обнаружения. Далее представляются результаты анализа, представленные в виде подробных таблиц и рисунков, которые наглядно и статистически демонстрируют все выводы. Статья завершается заключительным разделом, в котором кратко излагаются основные открытия и инсайты, за которыми следует полный список литературы, подтверждающий исследования.

1.1. Литературный обзор

Ситуация с кибербезопасностью постоянно осложняется из-за растущей сложности замаскированных вредоносных программ, которые представляют собой непреодолимое препятствие для поддержания цифровой безопасности. В этом обзоре литературы рассматриваются как основополагающие, так и современные исследования в области обнаружения вредоносных программ, при этом особое внимание уделяется анализу памяти и растущей роли машинных технологий обучения. Исследуя эту развивающуюся область, я критически изучаю существующие модели, оцениваю их успешность и выявляю присущие ей ограничения. Это исследование не только освещает текущее состояние средств защиты от кибербезопасности, но и подчеркивает необходимость в передовых методах обнаружения, способных противостоять все более неуловимым киберугрозам.

1.2. Предыстория замаскированного вредоносного ПО в дампах памяти

В постоянно меняющемся мире кибербезопасности термин “скрытое вредоносное ПО” включает в себя огромную категорию цифровых угроз, которые выходят за рамки обычных механизмов обнаружения. Эти изощренные противники используют методы уклонения с непревзойденным мастерством, стремясь скрыть свою истинную личность и деятельность в рамках обширного пространства цифровых операций. Замаскированное вредоносное ПО достигает этого, используя множество тактических приемов, включая шифрование кода, полиморфное поведение и другие методы маскировки, которые бросают вызов традиционным системам обнаружения на основе сигнатур [4,10].

В области анализа кибербезопасности дампы памяти становятся важнейшим полем

битвы при выявлении скрытой активности вредоносных программ с маскировкой. Дамп памяти - это, по сути, моментальный снимок сложного содержимого оперативной памяти компьютера в определенный момент времени. В контексте анализа вредоносных программ это временное хранилище становится настоящей сокровищницей, предоставляя разрозненную информацию о поведении программ и процессов во время выполнения [11]. Замаскированное вредоносное ПО, осведомленное о нестабильности оперативной памяти, стратегически скрывается в памяти, используя динамичный характер цифровой среды. Анализ дампов памяти становится тонким искусством, поскольку эти коварные объекты часто маскируются под законные процессы, делая границу между доброкачественными и вредоносными действиями нечеткой. Задача заключается не только в выявлении этих злоумышленников, но и в понимании того, какие глубокие изменения они совершают в пределах изменчивой памяти.

Разбираясь в тонкостях дампов памяти, исследователи открывают потенциал для расшифровки поведенческих паттернов скрытого вредоносного ПО, преодолевая ограничения традиционных методов обнаружения. По мере того как цифровые угрозы продолжают становиться все более изощренными, важность анализа дампов памяти становится все более очевидной, что требует инновационных и адаптивных подходов для укрепления арсенала кибербезопасности.

1.3. Обзор подходов к обнаружению скрытого вредоносного ПО в дампах памяти

Исторически сложилось так, что методы обнаружения на основе сигнатур служили краеугольным камнем защиты кибербезопасности. Эти методы, основанные на выявлении вредоносных программ с помощью предопределенных шаблонов и известных сигнатур, когда-то были очень эффективны в борьбе с традиционными угрозами. Однако их эффективность значительно снизилась из-за наличия скрытых вредоносных программ. Основное ограничение систем, основанных на сигнатурах, заключается в их внутренней зависимости от известных баз данных угроз. Эта характеристика делает их особенно неэффективными при обнаружении новых или сильно запутанных вариантов вредоносного ПО, которые отличаются от известных шаблонов [12, 13]. Кроме того, основанные на эвристике подходы, которые были направлены на устранение некоторых недостатков методов, основанных на сигнатурах, путем включения определенной степени поведенческого анализа, также показали свою уязвимость. Хотя эти подходы значительно продвинулись вперед в анализе поведения и атрибутов программ, их эффективность часто подрывается сложными методами максировки. Современные вредоносные программы, обладающие способностью имитировать безобидное поведение или эффективно скрывать свои вредоносные действия, часто не поддаются эвристическому анализу. Это ограничение в первую очередь связано с тем, что эвристические методы опираются на заранее определенные поведенческие правила и паттерны, которые могут не охватывать инновационную тактику, используемую новыми штаммами вредоносных программ [14].

Недавние достижения в области машинного обучения позволили использовать различные подходы для обнаружения скрытых вредоносных программ. Эти методы показали многообещающие результаты, особенно в сценариях бинарной классификации, обеспечивая точность обнаружения приблизительно в пределах 93-99%. Такие высокие показатели точности подчеркивают потенциал машинного обучения в распознавании доброкачественных и вредоносных объектов в бинарном контексте. Однако применение этих классификаторов машинного обучения в сценариях с несколькими классификациями обнаруживает существенное ограничение. Хотя они эффективны в задачах бинарной классификации, где цель

состоит в том, чтобы провести различие между двумя классами (вредоносными или доброкачественными), их производительность снижается при выявлении нескольких семейств вредоносных программ. В сценариях, требующих классификации различных типов вредоносных программ, таких как различение программ-шпионов, вредоносного ПО, троянов, а также их подкатегорий, эти подходы демонстрируют значительно меньшую точность. Это несоответствие в производительности может быть объяснено возросшей сложностью и тонкими различиями между несколькими семействами вредоносных программ, которые создают более сложные условия для алгоритмов классификации, изначально оптимизированных для принятия бинарных решений. Сложные поведенческие паттерны и едва заметные различия в атрибутах, отличающие одно семейство вредоносных программ от другого, требуют более сложного аналитического подхода, позволяющего ориентироваться в сложных и часто перекрывающихся характеристиках различных типов вредоносных программ.

Исходя из данных [15] эффективность расширенного канала CNN (DCNN) в выявлении скрытого вредоносного ПО была оценена с помощью анализа памяти. В то время как DCNN продемонстрировал впечатляющую точность в бинарной классификации (99,92%), его производительность в мультиклассовых сценариях, особенно при различении четырех основных семейств вредоносных программ, была заметно ниже (83,53%). Это говорит об ограниченной способности модели распознавать конкретные типы вредоносных программ в сложных задачах классификации. Кроме того, значительные вычислительные требования DCNN, обусловленные ее сложной архитектурой с несколькими сверточными уровнями и большим количеством нейронов, создают проблемы для реализации на устройствах с ограниченными ресурсами. Эти результаты подчеркивают критический баланс между точностью, специфичностью и вычислительной эффективностью, который необходимо соблюдать в передовых моделях обнаружения вредоносных программ.

Авторы книги [16] представили MalHyStack, модель, предназначенную для обнаружения скрытого вредоносного ПО в сетевых средах. Этот инновационный подход сочетает в себе комплексную систему обучения на начальном уровне и уровень глубокого обучения в качестве последующего этапа. Перед развертыванием этой классификационной модели с помощью CA определяется оптимальное подмножество функций. Несмотря на сложную архитектуру, показатели производительности модели указывают на определенные ограничения. В частности, при классификации четырех типов атак MalHyStack достиг точности 85,04% и скорости отзыва 85,17%. В более сложном сценарии, включающем 16 категорий вредоносных программ, уровень обнаружения снизился до 66,96% при точности 66,94%. Эти цифры, особенно в контексте мультиклассификации, свидетельствуют об ограниченной эффективности модели в быстро меняющейся среде цифровой безопасности, где более высокая точность имеет решающее значение для эффективного обнаружения и предотвращения вредоносных программ.

Внеся заметный вклад в область обнаружения вредоносных программ, авторы книги [17] представили подход RobustCBL, направленный на классификацию различных вредоносных программ. Хотя эта модель представляет собой инновационный шаг в обнаружении многоклассовых вредоносных программ, ее эффективность в точном выявлении различных категорий вредоносных программ имеет определенные ограничения. Эффективность RobustCBL при применении к конкретным семействам вредоносных программ демонстрирует умеренный уровень успешности: он обнаруживает программы-вымогатели в 67% случаев, шпионские программы - в 69%, а трояны - в 71%. Эти цифры, хотя и свидетельствуют о потенциале модели, также подчеркивают ее трудности с последовательной и точной

идентификацией этих распространенных типов вредоносных программ. Относительно низкие показатели обнаружения в этих категориях указывают на необходимость дальнейшего совершенствования способности модели распознавать нюансы характеристик и поведения, которые определяют эти конкретные семейства вредоносных программ. Более того, когда область применения RobustCBL была расширена, чтобы охватить все 16 отдельных классов вредоносных программ в наборе данных, общая точность модели достигла 72,6%. Хотя это свидетельствует о достаточном уровне владения мультиклассификацией, уровень точности не так высок, как хотелось бы для надежных приложений кибербезопасности. Кроме того, существенной проблемой, связанной с RobustCBL, является высокая частота ложных срабатываний (FPR). Высокий показатель FPR означает, что модель часто ошибочно классифицирует безобидное программное обеспечение или процессы как вредоносные, что может привести к ненужным или разрушительным действиям и подорвать доверие к надежности системы.

Подводя итог последним публикациям в области обнаружения вредоносных программ, можно сказать, что в разработке и оценке моделей, позволяющих выявлять сложные, запутанные вредоносные программы в ограниченных системных средах, наблюдается заметный пробел, особенно для обнаружения категорий вредоносных программ с высоким коэффициентом полезного действия. Этот пробел особенно заметен в контексте моделей, которые должны обеспечивать надежные возможности обнаружения при минимальном FPR. Данное исследование удовлетворяет эту критическую потребность, внедряя инновационную структуру. Этот процесс разработан не только для того, чтобы преуспеть в задачах бинарной классификации, но и для того, чтобы умело ориентироваться в сложностях классификации и идентификации различных семейств вредоносных программ. Благодаря тщательной оценке данная модель демонстрирует свою эффективность в борьбе с последними версиями киберугроз, тем самым внося значительный вклад в арсенал инструментов для борьбы с меняющимися вызовами цифровой безопасности.

2. Предлагаемая методология

В данном исследовании предложенная методология, подробно представленная на (рис. 1), представляет собой сложный и многогранный подход к улучшению обнаружения и анализа вредоносных программ. Эта методология объединяет передовые методы машинного обучения, охватывающие различные комплексные модели, такие как GB, RF, ADB, VT и BG, каждая из которых предназначена для решения сложных задач классификации скрытого вредоносного ПО. Используя комбинацию стратегий предварительной обработки данных, выбора функций и коллективного обучения, данный подход направлен на значительное повышение точности и надежности обнаружения вредоносных программ. Системный и целостный характер этой методологии, как показано на (рис. 1), не только иллюстрирует передовые исследования в области кибербезопасности, но и устанавливает новые стандарты в этой области, демонстрируя глубокое понимание как технических сложностей, так и практических последствий анализа вредоносных программ.

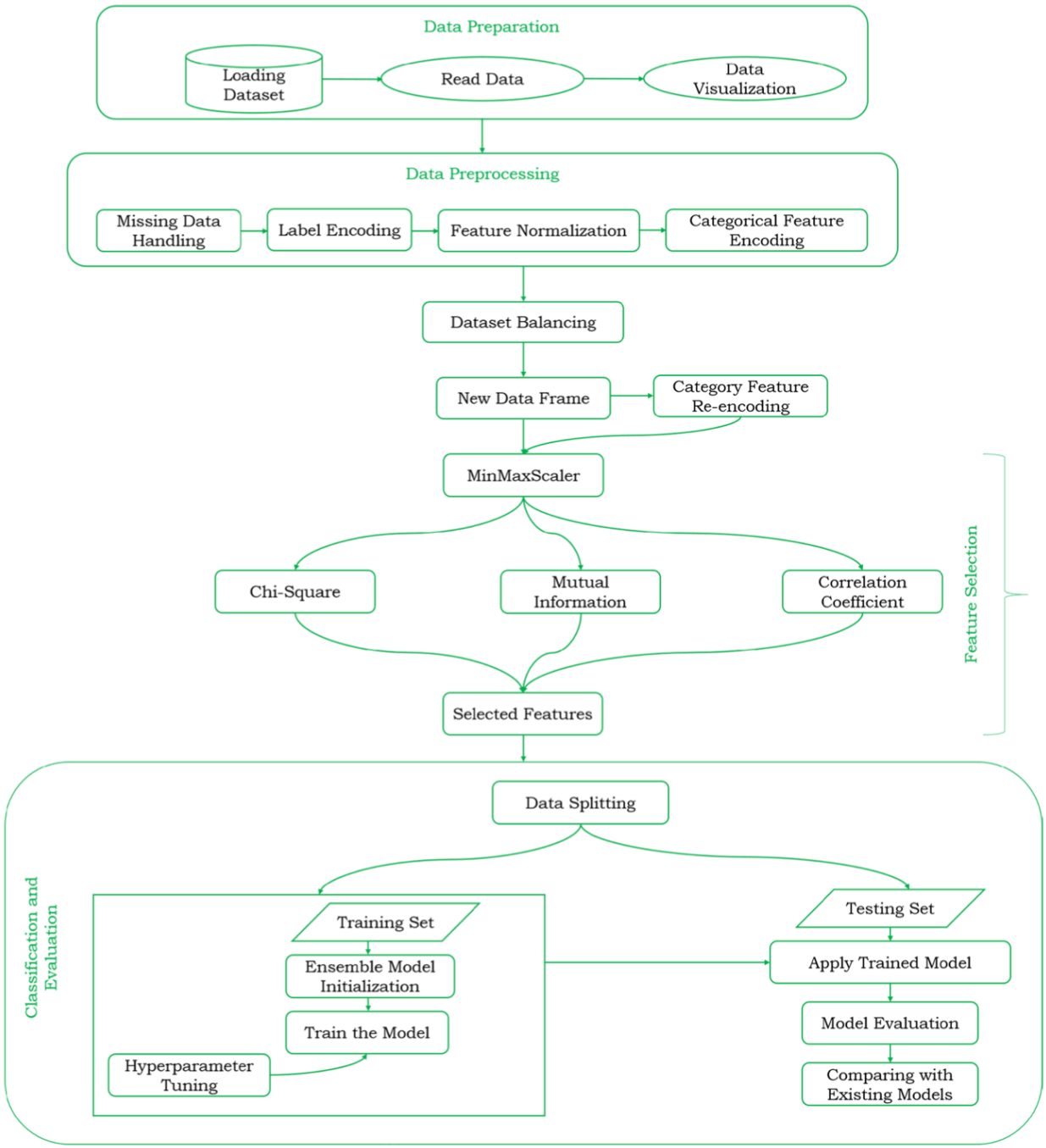


Рисунок 1

2.1. Сбор и предварительная обработка данных

В этом исследовании этап “Сбора и предварительной обработки набора данных” тщательно структурирован, чтобы обеспечить готовность набора данных к углубленному анализу с помощью машинного обучения. Используется набор данных Obfuscated- MalMem2022 [18], который имеет обширную и тщательно отобранную коллекцию дампов памяти, как доброкачественных, так и вредоносных, разработанную для отражения реальных сценариев кибербезопасности. Функции и их подробное описание подробно представлены в разделе

“Приложение”. Этот набор данных играет ключевую роль во всем анализе, предоставляя представление о доброкачественных данных и различных классах вредоносных программ, включая программы-вымогатели, шпионские программы и троянских коней. Подробный перечень точек данных, разделенных на бинарные классы, категории вредоносных программ и конкретные семейства вредоносных программ, систематически представлен в (табл. 2). Этот обширный набор данных не только обогащает исследования разнообразием образцов, но и обеспечивает надежность и достоверность предложенной мной модели обнаружения.

Для обеспечения целостности данных набор данных сначала очищается от пропущенных и бесконечных значений. Это включает замену бесконечных значений на NaN (не число) и последующее удаление этих значений NaN. Математически это выглядит следующим образом: *data = data.replace*({∞,−∞},*NaN*) *и data = data.dropna*(). Этот шаг важен для предотвращения вычислительных ошибок и искажений, которые могут возникнуть из-за неполных или поврежденных данных.

Таблица 2

Распределение данных: подробная разбивка по типам классификации и семействам вредоносных программ

Набор данных	**Двоичный класс с числами**	**Категория вредоносных программ с количеством элементов (4 класса)**	**Семейства вредоносных программ с количеством элементов (16 классов)**
Запутанный-MalMem2022	Доброкачественные (29,298) (50%)	Доброкачественные (29,298) (50%)	Доброкачественные (29,298) (50%)
	Вредоносные программы (29,298) (50%)	Программы-вымогатели (9791) (16,71%)	Shade (2128) (3.63%)
			Ako (2000) (3.41%)
			Conti (1988) (3.39%)
			Maze (1958) (3.35%)
			Pysa (1717) (2.93%)
		Шпионские программы (10,020) (17,10%)	Transponder (2410) (4.11%)
			Gator (2200) (3.75%)
			180Solutions (2000) (3.41%)
			Coolwebsearch (2000) (3.41%)
			TIBS (1410) (2.41%)
		Троянский конь (9487) (16,19%)	Refroso (2000) (3.41%)
			Scar (2000) (3.41%)
			Emotet (1967) (3.36%)
			Zeus (1950) (3.33%)
			Reconyc (1570) (2.68%)

2.2. Классификация и кодирование данных

В исследовании процесс “кодирования и нормализации данных” играет ключевую роль, в первую очередь из-за внутренних характеристик набора данных и требований алгоритмов машинного обучения. Этот процесс служит двум фундаментальным целям: преобразованию категориальных данных в машиночитаемый формат и стандартизации ряда непрерывных числовых характеристик [19, 20]. Категориальный характер некоторых объектов в наборе данных, в частности столбца "Категория", требует кодирования. Этот набор данных включает в себя различные типы вредоносных программ, такие как программы-вымогатели, шпионские программы и трояны, каждая из которых представлена в виде строки.

Алгоритмы машинного обучения по своей сути требуют ввода числовых данных. Для решения этой проблемы мы используем кодировку меток, представленную в виде функции отображения: $f: C \to Z$. Здесь C - это набор категориальных обозначений (например, "Доброкачественное", "Вредоносное ПО", "Шпионское ПО", "Троян"), а Z представляет собой набор целых чисел. Если c_i является категориальным значением в наборе данных, то его кодированное значение z_i задается как $z_i = f(c_i)$, где f - функция кодирования меток, преобразующая каждую уникальную метку в уникальное целое число. Например, у нас есть ярлыки {Доброкачественное ПО, программа-вымогатель, шпионское ПО, троян}, которые могут быть закодированы как {0, 1, 2, 3} соответственно.

Набор данных содержит числовые характеристики, варьирующиеся в различных диапазонах. Без нормализации характеристики с большими диапазонами могут непропорционально повлиять на модель, что приведет к необъективному обучению. Эта проблема особенно актуальна в наборах данных с различными масштабами характеристиками, как в случае с наборами данных по кибербезопасности. Используется StandardScaler, который нормализует объект путем вычитания среднего значения и деления на стандартное отклонение, эффективно преобразуя данные в среднее значение, равное нулю, и стандартное отклонение, равное единице. Для данного признака X с n выборками $X = [x1, x2, \ldots, xn]$, в процессе нормализации значения X корректируются таким образом, чтобы они имели среднее значение, равное нулю, и стандартное отклонение, равное единице. Нормализованное значение $x_{i'}$ для каждой выборки x_i в X рассчитывается с использованием формулы (1).

$$x'_i = \frac{x_i - \mu x}{\sigma x} \quad (1)$$

где μx - среднее значение признака X, вычисленное как $\mu x = \frac{1}{n}\sum_{i=1}^{n} x_i$, а σx - стандартное отклонение X, вычисленное как (2):

$$\sigma x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu x)^2} \quad (2)$$

Кодирование гарантирует, что алгоритмы смогут интерпретировать категориальные данные, в то время как нормализация стандартизирует масштабы объектов, позволяя модели изучать и делать прогнозы без смещения в сторону числового масштаба какого-либо конкретного объекта.

2.3. Преодоление классового дисбаланса с помощью SMOTE

Исходный набор данных, вероятно, имеет несбалансированное распределение по классам, что означает, что некоторые типы вредоносных программ недостаточно представлены. Этот дисбаланс может привести к смещению модели машинного обучения в сторону большинства классов, снижая ее эффективность при точном выявлении менее распространенных типов вредоносных программ. Создавая синтетические выборки для групп меньшинств, SMOTE помогает создать более сбалансированный набор данных, который способствует созданию более надежной и обобщенной модели, способной эффективно обнаруживать различные типы вредоносных программ.

Для каждой выборки x_i в классе меньшинства SMOTE вычисляет ее k - *ближайших* соседей. Пусть $N_k(x_i)$ обозначает множество k – *ближайших* соседей x_i в пространстве признаков. Метрика расстояния, часто евклидова, для двух выборок x_i и x_j задается уравнением (3).

$$d(x_i,x_j) = \sqrt{\sum_{l=1}^{m}(x_{il} - x_{jl})^2} \quad (3)$$

где m - количество признаков.

Синтетический экземпляр x_{new} генерируется путем интерполяции между образцом x_i и одним случайным образом выбранным ближайшим соседом $x_{ni} \in N_k(x_i)$. Формула для создания синтетического образца такова (4):

$$x_{new} = x_i + \lambda * (x_{ni} - x_i) \quad (4)$$

где λ - случайное число от 0 до 1. Это гарантирует, что синтетическая выборка x_{new} будет располагаться на отрезке прямой между x_i и x_{ni} в пространстве признаков. Цель состоит в том, чтобы сбалансировать распределение по классам между большинством и меньшинством. Если размер меньшего класса равен S_{min} и задан желаемый размер после повторной выборки $S_{desired}$, то количество синтетических выборок N_{synth}, которые должны быть сгенерированы для меньшего класса, равно $S_{desired} - S_{min}$. Этот процесс включает в себя многократное применение этапа генерации синтетической выборки, пока не будет создано N_{synth} выборок, тем самым увеличивая класс меньшинства для достижения желаемого баланса классов.

В данном исследовании применение SMOTE является важным шагом для устранения искажений, вызванных несбалансированным распределением классов. Это расширяет возможности модели по изучению обобщенных закономерностей и повышает ее эффективность при точной классификации различных типов вредоносных программ, что важно в контексте кибербезопасности.

2.4. Выбор и масштабирование объектов

В этом исследовании “Выбор и масштабирование признаков” является критическим этапом, имеющим решающее значение для повышения производительности и интерпретируемости модели. Этот этап включает в себя два основных процесса: выбор объектов с помощью SelectKBest с методами chi2 и mutual_info_classif и масштабирование объектов с помощью MinMaxScaler. Критерий chi-квадрат оценивает независимость двух переменных, что делает его пригодным для отбора признаков, целью которого является выявление признаков, которые, скорее всего, не зависят от меток классов [21]. Для данного объекта X и метки класса Y статистика chi-квадрат вычисляется следующим образом (5):

$$X^2(X,Y) = \sum_{i=1}^{n}\frac{(O_i - E_i)^2}{E_i} \quad (5)$$

где Oi - наблюдаемая частота, Ei - ожидаемая частота при нулевой гипотезе независимости, а n

- количество различных значений в X. Чем выше значение chi-квадрат, тем больше вероятность того, что признак зависит от класса и, следовательно, важен для классификации.

Взаимная информация измеряет объем информации, которую можно получить об одной случайной величине, наблюдая за другой [22]. Для объектов X и метки класса Y это определяется как (6):

$$I(X; Y) = \sum_{x \in X, y \in Y} P(x, y) \log \left(\frac{P(x, y)}{P(x)P(y)}\right) \quad (6)$$

где $P(x, y)$ - совместное распределение вероятностей X и Y, а $P(x)$, $P(y)$ - предельные распределения вероятностей X и Y, соответственно. Признаки с более высокими значениями взаимной информации считаются более релевантными для определения принадлежности к классу.

После выбора объекта необходимо выполнить масштабирование, чтобы нормализовать значения объектов в пределах ограниченного диапазона, обычно [0, 1]. MinMaxScaler преобразует каждый объект x по формуле (7):

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (7)$$

где x_{min} и x_{max} - это минимальное и максимальное значения объекта x соответственно. Этот метод масштабирования сохраняет форму распределения набора данных и полезен, когда объекты имеют различные масштабы и диапазоны. В контексте данного исследования комбинация chi-квадрата и взаимной информации для выбора признаков с последующим использованием MinMaxScaler для масштабирования гарантирует, что модель фокусируется на наиболее информативных признаках, которые вносят значительный вклад в задачу классификации. Такой подход не только повышает предсказательную способность модели, но и повышает ее эффективность за счет снижения сложности вычислений, что имеет решающее значение для обработки больших и сложных наборов данных о кибербезопасности.

После завершения этапов предварительной обработки и выбора функций мы применили различные подходы, основанные на ансамбле, включая GB, VT, ADB, RF и BG, каждый из которых был тщательно настроен по гиперпараметрам. В разделе ниже приводится краткое описание каждого из этих обновленных методов, основанных на использовании ансамблей, с описанием их уникальных характеристик и роли в данном исследовании.

3. Модели

3.1. Процесс обучения модели с классификатором повышения градиента

В этом исследовании использование GradientBoostingClassifier (GBC) для классификации нескольких типов вредоносных программ основано на его методологии, основанной на ансамбле, и сложной обработке сложных наборов данных. Эффективность GBC в управлении сложными взаимосвязями данных объясняется его комплексным подходом к обучению и оптимизацией конкретных гиперпараметров. Суть GBC заключается в поэтапном построении аддитивной модели [23]. Формально это выражается следующим образом (8):

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x) + const \quad (8)$$

где $F(x)$ - конечная модель, M - количество этапов, $h_m(x)$ - базовый обучаемый, а γ_m - вес каждого этапа.

Значение числа оценок гиперпараметра, равное 10, определяет количество построенных последовательных этапов, скорость обучения при значении 0,1. Этот параметр масштабирует вклад каждого этапа, влияя на скорость сходимости модели. Он регулирует размер шага в процессе градиентного спуска, максимальная глубина ограничена 3, он управляет максимальной глубиной отдельных этапов, снижая сложность модели и ее переобучение.

GBC оптимизирует дифференцируемую функцию потерь $L(y, F(x))$, где y - фактическое значение, а $F(x)$ - прогноз модели. Потери минимизируются с помощью градиентного спуска, при этом каждый этап строится для моделирования отрицательного градиента функции потерь относительно прогнозов. Модель итеративно обновляет прогнозы на основе уравнения (9):

$$F_{m+1}(x) = F_m(x) + \nu \sum_{i=1}^{n} \gamma_m h_m(x_i) \quad (9)$$

где ν - скорость обучения, а N - количество образцов. Процесс классификации подробно описан в (алг. 1).

Алгоритм 1.

Процесс классификации вредоносных программ по модели с помощью GBC.

1. Начните с исходной модели, обычно с постоянного значения. Часто это логарифмические коэффициенты в случае классификации:

   $$F_0(x) = argmin_\gamma \sum_{i=1}^{N} L(y_i, \gamma);$$

   где L - функция потерь, y_i - истинные метки, а N - количество выборок.

2. Для каждой итерации $m = 1, 2, ..., M$ (где M - количество этапов):
   a. Вычислите псевдовязки для каждого экземпляра в каждом классе k:

   $$r_{ikm} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F_k(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

   b. Сопоставьте этап решений $h_{mk}(x)$ с этими остатками для каждого класса.

3. Определите выходные значения для конечных узлов в каждом этапе:

   $$\gamma_{jkm} = argmin_\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$$

4. Обновите модель для каждого класса k:

$$F_{mk}(x) = F_{m-1,k}(x) + \nu \sum_{j=1}^{J} \gamma_{jkm} I(x \in R_{jm})$$

5. Для нового дампа памяти x_{new} модель выводит набор оценок для каждого класса k. Затем к этим оценкам применяется функция softmax для получения вероятностей:

$$P(y = k \mid x_{new}) = \frac{e^{F}Mk(x_{new})}{\Sigma_{l=1}^{k} e^{F}Ml(x_{new})};$$ где K - общее количество классов.

$$P(y = k \mid x_{new}) = \frac{1}{1 + e^{-F}M(x_{new})}$$

6. Класс с наибольшей вероятностью из выходных данных softmax выбирается в качестве окончательного прогноза для x_{new}.

В (алг. 1), описывающем процесс классификации вредоносных программ модели с помощью классификатора с ускорением градиента (GBC), процедура начинается с инициализации исходной модели, часто это постоянное значение, представленное как $F_0(x)$, обычно логарифмические коэффициенты в классификационных сценариях. Впоследствии, для каждой итерации $m = 1, 2, \ldots, M$ (где M - количество этапов), алгоритм вычисляет псевдовязки для каждого экземпляра в каждом классе k. Эти псевдовязки, обозначаемые как r_{ikm}, отражают разницу между истинными метками y_i и предсказаниями текущей модели $F(x_i)$. Затем этапы решений $h_{mk}(x)$ сопоставляется с этими остатками для каждого класса. Определяются выходные значения для конечных узлов на каждом этапе, y_{jkm}, и модель обновляется для каждого класса k на основе этих значений. Для нового дампа памяти x_{new} модель выводит оценки для каждого класса k. Функция softmax применяется к этим оценкам для получения вероятностей, и класс с наибольшей вероятностью выбирается в качестве окончательного прогноза. Этот итеративный процесс подбора этапов решений и обновления модели расширяет ее прогностические возможности за счет последовательного обучения, что делает ее эффективной при выявлении сложных взаимосвязей в данных.

Включение функции softmax в настройку мультикласса позволяет GBC эффективно выполнять задачи классификации нескольких типов вредоносных программ. Такой подход гарантирует, что каждому дампу памяти присваивается распределение вероятности по всем возможным категориям вредоносных программ, что позволяет провести детальную классификацию, основанную на наибольшей вероятности. Способность функции softmax преобразовывать необработанные оценки в вероятности, которые в сумме равны единице, делает ее идеальным выбором для многоклассовой классификации в контексте обнаружения вредоносных программ. Классификатор Gradient Boosting, основанный на комплексном обучении и тщательной настройке гиперпараметров, является эффективным методом решения многогранной задачи классификации вредоносных программ в области кибербезопасности. Эта модель, способная обрабатывать сложные данные и снижать переобучение, представляет собой сложный подход в области машинного обучения для обнаружения и анализа вредоносных программ.

3.2. Модель тренировочного процесса с BG ensemble

Метод объединения пакетов особенно эффективен для классификации запутанных вредоносных программ из-за присущей ему способности предотвращать переобучение, что является распространенной проблемой в сложных задачах классификации. Благодаря агрегированию прогнозов из нескольких деревьев принятия решений, каждое из которых было подготовлено на основе разных подмножеств данных, пакетирование привносит разнообразие в процесс обучения [24]. Такое разнообразие имеет решающее значение при работе с запутанными вредоносными программами, где незначительные различия в структуре данных могут существенно повлиять на точность классификации. Комплексный подход гарантирует, что модель не будет чрезмерно полагаться на конкретные атрибуты данных, тем самым повышая ее способность обобщать и точно идентифицировать даже сложные, замаскированные вредоносные программы. Процесс классификации подробно описан в (алг. 2).

Алгоритм 2

Процесс классификации вредоносных программ по модели с помощью BG.

1. Определяем набор упаковок: $BaggingClf = \{RF_1, RF_2, ..., RF_n\}$
2. Для каждого RF_1 в $BaggingClf$:

   a. Образец начальной загрузки: $D_i \leftarrow BootstrapSample(D)$

   b. Построить дерево решений DT_{ij} для каждого D_i:

3. Случайный выбор функции: $F_{ij} \leftarrow RandomSubset(F)$
4. Для узла N найдите разбиения , минимизирующие примеси $Gini$:

   $$Gini(N) = 1 - \sum (P_k)^2$$

   $$s^* = argmin_{s \in S} Gini(N)$$

5. Увеличьте дерево до максимальной глубины MaxDepth или до тех пор, пока не будет выполнен критерий.
6. Ансамблевое предсказание для выборки x:

   $$y^{pred}(x) = mode\{RF_1(x), RF_2(x), ..., RF_n(x)\}$$

В (алг. 2), описывающем процесс классификации вредоносных программ в модели с использованием пакетов (BG), набор пакетов, обозначаемый как BaggingClf, определяется как набор отдельных случайных классификаторов, представленных как RF_1, RF_2, вплоть до RF_n. Для каждого RF_i,-пакета CLF выполняется операция начальной выборки, генерирующая набор данных начальной загрузки D_i из исходного набора данных D. Затем для каждого D_i строится дерево решений (DT_{ij}). Построение предполагает случайный выбор объектов, при котором случайным образом выбирается подмножество объектов F_{ij}. Для каждого узла N в дереве оптимальное разделение s^* определяется путем минимизации критерия примеси $Gini$.

Коэффициент примеси $Gini(N)$ измеряет наличие примеси или беспорядка в наборе выборок. Дерево увеличивается либо до максимальной глубины (MaxDepth), либо до тех пор, пока не будет выполнен указанный критерий. Затем вычисляется совокупный прогноз для данной выборки x, как режим прогнозирования из всех классификаторов RF_i. Этот подход использует разнообразие, привносимое выборкой загрузочной ленты и случайным отбором признаков, повышая общую надежность и точность набора пакетов в контексте классификации вредоносных программ. Способность ансамбля агрегировать прогнозы снижает риск переобучения, что делает его особенно эффективным для сложных задач классификации, таких как обнаружение вредоносных программ в дампах памяти.

3.3. Модель тренировочного процесса с VT ensemble

Метод Voting Ensemble также используется для классификации дампов памяти по категориям доброкачественных или различных вредоносных программ [25]. Этот комплексный метод объединяет прогнозы, полученные с помощью нескольких различных классификаторов: Дерева решений, логистической регрессии и классификатора опорных векторов (SVC), каждый из которых дает уникальную информацию. Весь процесс подробно описан в (алг. 3).

Алгоритм 3

Построение классификатора мягкого голосования и процесс прогнозирования.

1. Определение классификаторов для ensemble VT:

   $clf\,1 \leftarrow DecisionTreeClassifier()$

   $clf\,2 \leftarrow LogisticRegression(max\_iter\ =\ 1000)$

   $clf\,3 \leftarrow SVC(probability\ =\ True)$

2. Построение классификатора голосования:

   $eclf \leftarrow VotingClassifier(estimators\ =\ [('dt', clf\,1), ('lr', clf\,2), ('svc', clf\,3)], voting\ = 'soft')$

3. Для каждого классификатора clf_i в ансамбле процесс обучения включает в себя подгонку модели к обучающим данным.
4. Для данной выборки x вероятность принадлежности к классу k, предсказанная классификатором clf_i, равна $P_{ik}(x)$.
5. Таким образом, окончательный прогноз для класса k представляет собой средневзвешенное значение этих вероятностей:

   $$P_{ensemble,k}(x) = \frac{1}{N}\sum_{i=1}^{N} w_i P_{ik}(x);$$

   где N - количество классификаторов, а w_i - веса, присвоенные прогнозу каждого классификатора.

6. В качестве окончательного прогноза выбирается класс $P_{ensemble,k}(x)$ с наибольшей совокупностью вероятностей из всех k классов.
7. Окончательное предсказание: $y^{pred} = argmax^k P_{ensemble,k}(x)$.

В примере классификатора с программным голосованием (алг. 3) описывает процесс построения и прогнозирования для надежного набора классификаторов. Для ансамбля определены отдельные классификаторы, а именно clf_1 (DecisionTreeClassifier), clf_2 (логистическая регрессия с увеличенным значением *max_iter*) и clf_3 (SVC с оценкой вероятности). Затем формируется классификатор голосования ($eclf$), объединяющий эти классификаторы со стратегией "мягкого" голосования. Во время обучения каждый классификатор clf_i проходит процедуру подгонки к данным обучения. Для данной выборки x вероятность принадлежности к классу k, предсказанная clf_i, обозначается как $P_{ik}(x)$. Окончательный прогноз для класса k в ансамбле вычисляется как средневзвешенное значение этих вероятностей, где N представляет количество классификаторов, а w_i обозначает веса, присвоенные прогнозу каждого классификатора. Класс с наибольшей вероятностью из всех классов выбирается в качестве окончательного прогноза (y^{pred}), реализующего операцию *argmax*. Такой подход позволяет принимать коллективные решения с использованием различных классификаторов, повышая адаптивность модели и точность прогнозирования в области обнаружения вредоносных программ.

Этот процесс обучения ансамбля гарантирует, что каждый классификатор вносит свой вклад в понимание данных, а их объединенные прогнозы дают всестороннее представление, тем самым повышая общую эффективность прогнозирования и надежность модели. Этот подход, сочетающий дерево решений, логистическую регрессию и SVC в механизме мягкого голосования, обеспечивает комплексный процесс принятия решений, используя сильные стороны каждого классификатора. Агрегированные прогнозы ensemble обеспечивают более сбалансированную и точную классификацию, необходимую для решения сложной задачи обнаружения вредоносных программ.

3.4. Процесс обучения и классификации модели с помощью ADB ensemble

Метод ADB Ensemble выделяется как надежный подход к классификации дампов памяти по категориям доброкачественных или различных вредоносных программ. ADB, сокращенно от Adaptive Boosting, преуспевает в совершенствовании процесса классификации, итеративно фокусируясь на трудных для классификации примерах [19]. Он объединяет множество слабых обучаемых, обычно простых деревьев принятия решений, для формирования надежного классификатора. Каждый последующий ‘ученик’ адаптируется таким образом, чтобы подчеркнуть те моменты данных, которые предыдущие ученики неправильно классифицировали, тем самым постепенно повышая точность модели. Весь процесс подробно описан в (алг. 4)

Алгоритм 4.

Процесс обучения и прогнозирования модели совместно с ADB.

1. Инициализация: Начните с набора данных D и весов $w_i = \frac{1}{N}$ для каждого экземпляра i, где N - общее количество экземпляров.
2. Для $t = 1 - T$, (где T=10 - количество оценок):

   a. Обучите ‘слабого ученика’ L_t(*DecisionTreewithmax_depth=1*) на наборе данных, используя текущие веса.

   b. Вычислите погрешность ε_t для L_t:

   $\varepsilon_t = \frac{\Sigma_{i=1}^{N} w_i I(y_i \neq L_t(x_i))}{\Sigma_{i=1}^{N} w_i}$, где I - индикаторная функция, y_i - истинная метка, а $L_t(x_i)$ - прогноз.

   c. Вычислите вес учащегося α_t:

   $$\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

   d. Обновите веса для каждого экземпляра:

   $$w_{i,new} = w_i e^{-\alpha_t y_i L_t(x_i)}$$

   e. Нормализуйте веса так, чтобы их сумма равнялась 1.

3. Окончательная модель представляет собой взвешенную комбинацию ‘слабых учащихся’:

   $$AdaBoostModel(x) = sign\left(\sum_{t=1}^{T} \alpha_t L_t(x_i)\right)$$

4. Для нового образца x_{new}окончательная модель AdaBoost предоставляет классификацию:

   $y^{pred} = AdaBoostModel(x^{new})$

(Алг. 4) описывает процесс обучения и прогнозирования модели с помощью AdaBoost (ADB) для обнаружения вредоносных программ. Процесс начинается с инициализации набора данных D и присвоения весов $w_i = \frac{1}{N}$ для каждого экземпляра i, где N - общее количество экземпляров. Для каждой итерации от $t = 1$ до T, (где $T = 10$ – количество оценок) на наборе данных с использованием текущих весов обучается ‘слабый обучаемый’, L_t, в частности дерево решений с max_depth = 1. Вычисляется ошибка ε_t или L_t, отражающая частоту ошибочных классификаций. Тогда вес учащегося α_t вычисляется на основе ε_t. Веса для каждого экземпляра обновляются с использованием α_t, а нормализация гарантирует, что они суммируются до 1.

Итоговая модель представляет собой взвешенную комбинацию ‘слабых учащихся’, а для новой выборки модель AdaBoost предоставляет классификационный критерий. Этот итеративный процесс повышения эффективности фокусируется на экземплярах, которые были неправильно классифицированы на предыдущих итерациях, что повышает способность модели адаптироваться и улучшать свою производительность с течением времени.

Преимущество AdaBoost заключается в том, что он больше фокусируется на экземплярах, которые труднее классифицировать, тем самым повышая общую производительность ансамбля. Этот метод особенно эффективен в сложных задачах классификации, таких как определение различий между различными типами вредоносных программ, благодаря своей адаптивной природе и способности повышать производительность простых моделей.

3.5. Подход Random forest ensemble для классификации вредоносных программ

В этом исследовании ансамблевый метод Random Forest (RF) также используется в качестве ключевого аналитического инструмента для классификации дампов памяти, позволяющего отличать доброкачественные данные от различных типов вредоносных программ [26]. Этот метод ценится за его надежность и точность, особенно при обработке сложных наборов данных с многочисленными функциями. RF сочетает в себе несколько деревьев принятия решений, что снижает риск переобучения при одновременном учете широкого спектра характеристик данных. Каждое дерево дает уникальную перспективу, а их коллективное принятие решений обеспечивает сбалансированную и всеобъемлющую классификацию. Адаптивность и эффективность RF-системы делают ее бесценным компонентом данного исследования, расширяющего возможности в обнаружении и анализе вредоносных программ. Весь процесс подробно описан в (алг. 5).

Алгоритм 5.

Random Forest Ensemble в классификации вредоносных программ.

1. Инициализация: Определите RF с помощью n_estimators=10 и random_state = 42.
2. Для каждого дерева t в RF:

   a. Случайным образом выберите образцы с заменой из X_{train}, чтобы создать загрузочный набор данных D_t.

   b. Увеличьте t на D_t путем рекурсивного разделения узлов на основе подмножеств объектов. В каждом узле:

1. Выберите m объектов случайным образом из общего числа объектов

2. Выберите наилучшее распределение на основе такого критерия примесей, как Gini:

$Gini(S) = 1 - \sum_{i=1}^{c}(P_i)^2$, где P_i - доля образцов, относящихся к классу i

3. После обучения для новой выборки x каждое дерево t в RF предсказывает класс y_t.

4. Окончательный класс предсказания yRF - это режим всех y_t:

   $$yRF(x) = mode\{ y_1(x), y_2(x),..., y_{10}(x)\}$$

5. Предсказать, является ли x_{new} доброкачественным или представляет собой определенный тип вредоносного ПО, используя RF:

   $$y_{new} = yRF(x_{new})$$

На начальном этапе работы алгоритма создается набор Random Forest (RF) для классификации вредоносных программ с ключевыми параметрами, в частности, устанавливается значение *n_estimators* равным 10, а значение *random_state* равным 42. Таким образом, создается набор из 10 деревьев решений с фиксированным случайным начальным числом, обеспечивающий как разнообразие, так и воспроизводимость. Впоследствии для каждого дерева t в пределах RF инициируется процесс начальной загрузки выборки путем случайного выбора выборок с заменой из обучающего набора данных (X_{train}), создавая отличительный набор данных начальной загрузки D_t для каждого дерева. Фаза роста дерева разворачивается по мере того, как каждое дерево решений t строится на D_t, включая рекурсивное разделение узлов на основе случайно выбранных подмножеств признаков. В каждом узле случайным образом выбираются m признаков из общего набора признаков, и оптимальное распределение определяется с использованием критерия примеси, такого как примесь Gini. После этапа обучения каждое дерево t вносит свой вклад в прогнозирование класса y_t для новой выборки x. Окончательный ансамблевый прогноз yRF для новой выборки затем вычисляется как режим для всех y_t.

Метод RF Ensemble особенно эффективен для этого исследования благодаря своей способности обрабатывать многомерные данные и устойчивости к переобучению. Объединяя предсказания нескольких деревьев решений, каждое из которых основано на разных подмножествах данных, RF обеспечивает комплексный подход к классификации сложных и детализированных моделей, типичных для обнаружения вредоносных программ. Этот подход обеспечивает баланс между погрешностями и отклонениями, что приводит к более точным и надежным результатам классификации.

После этапа обучения модель проходит тестирование с использованием определенного набора тестовых данных. В следующем разделе этой статьи будут подробно описаны результаты, полученные с помощью различных оценочных показателей, подчеркивающих эффективность модели. Кроме того, в этом разделе будут представлены сравнительные анализы, демонстрирующие эффективность модели по сравнению с существующими методологиями обнаружения запутанных вредоносных программ.

Результаты и анализ

В этом исследовании по усовершенствованной классификации вредоносных программ экспериментальная установка проводилась на высокопроизводительном устройстве ASUS, оснащенном процессором Intel(R) Core(TM) i7-11700 11-го поколения с базовой частотой 2,50 ГГц и 16,0 ГБ оперативной памяти. Эта реализация, работающая на 64-разрядной системе Windows 11 Pro, использует Anaconda Navigator для управления программными средами, в

основном используя Jupyter Notebook для разработки. Ключевые библиотеки Python, такие как Pandas, Matplotlib, Seaborn, Scikit-learn и Imbalanced-learn, являются неотъемлемой частью исследования, облегчая выполнение задач от предварительной обработки данных до оценки модели машинного обучения. Используется набор методов машинного обучения, включая классификаторы randomForest, Bagging, DecisionTree, LogisticRegression, SVC, голосование, AdaBoost и GradientBoosting, которые оцениваются с использованием таких показателей, как точность, аккуратность, отзывчивость и F1-оценка, что обеспечивает тщательную оценку производительности модели при обнаружении вредоносных программ.

Эта установка обеспечивает необходимую вычислительную мощность и универсальность для решения сложных задач, связанных с исследованиями в области кибербезопасности. Для всесторонней оценки платформы в этом исследовании используется набор данных Obfuscated-MalMem2022, который является ключевым ресурсом для анализа передовых методов обнаружения вредоносных программ. Этот набор данных делится на обучающие и тестовые подмножества с использованием метода “train_ test_split” из библиотеки scikit-learn. Стратегическое распределение 75% расходов на обучение и 25% - на тестирование, обеспечивает надежный процесс обучения и при этом предоставляет значительный набор данных для проверки. Основное внимание уделяется данным тестирования, которые составляют 25% всего набора данных, поэтому все результаты представлены в этом разделе. Для оценки эффективности и точности модели тщательно используется ряд оценочных показателей. Эти показатели, имеющие решающее значение для определения эффективности модели, подробно описаны в (табл. 3), дополненной соответствующими уравнениями. Этот методический подход подчеркивает строгость и точность, присущие процессу оценки предлагаемой системы обнаружения вредоносных программ.

На (рис. 2) показано распределение по классам до применения SMOTE. Изначально в распределении различных категорий вредоносных программ - доброкачественных, программ-вымогателей, программ-шпионов и троянов - наблюдался значительный дисбаланс. Как показано на круговой диаграмме, доброкачественные экземпляры составляют половину набора данных (50%), в то время как категории вредоносных программ (программы-вымогатели, шпионские программы и трояны) представлены в меньшем количестве - от 16,19 до 17,10%. Этот дисбаланс является распространенной проблемой в машинном обучении, особенно в контексте кибербезопасности, поскольку он может привести к появлению предвзятых моделей, которые не справляются с выявлением менее представленных классов.

После применения SMOTE наблюдается впечатляющая трансформация в распределении по классам. Теперь каждая категория содержит одинаковое количество экземпляров, а именно 29,298. Это выравнивание имеет решающее значение для разработки объективной и эффективной модели классификации. SMOTE достигает этого путем передискретизации второстепенных классов, в данном случае программ-вымогателей, шпионских программ и троянских программ, до тех пор, пока они не сравняются по количеству экземпляров с основным классом, который является вредоносным. Сбалансированное распределение после запуска повышает способность модели к обучению на основе одинаково репрезентативного набора данных, гарантируя, что каждому типу вредоносного ПО придается одинаковое значение на этапе обучения. Такой подход снижает риск чрезмерной адаптации к классу majority и улучшает способность модели обнаруживать и классифицировать типы вредоносных программ, которые изначально были недостаточно представлены. Полученное в результате единообразное распределение по всем категориям закладывает прочную основу для построения надежной и эффективной модели классификации вредоносных программ,

необходимой для учета разнообразного и эволюционирующего характера киберугроз.

Таблица 3

Оценочные показатели с надлежащим описанием

Показатели оценки	Описание
Accuracy (ACC)	Аккуратность, определяемая как отношение правильно предсказанных наблюдений к общему количеству наблюдений, рассчитывается как $Accuracy = \frac{True\ Positives(TP) + True\ Negatives(TN)}{Total\ Observations}$, что обеспечивает фундаментальную меру общей корректности прогнозирования моделей
Precision (PR)	Точность, ключевой показатель при оценке модели $\frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$, определяется количественно как доля истинных положительных прогнозов среди всех положительных прогнозов, сделанных с помощью модели
Recall (RE)	Напоминание, важный показатель в моделях классификации, определяется формулой $Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}$, отражающей способность модели правильно идентифицировать все соответствующие экземпляры в наборе данных
F1-score (FS)	Показатель F1, важнейший показатель, который уравновешивает точность и запоминаемость, рассчитывается по формуле $F1 - score = 2 * \frac{Precision\ \times Recall}{recision + Recall}$, тем самым обеспечивая гармоничное среднее значение, которое отражает точность модели при правильной классификации точек данных.
False positive rate (FPR)	FPR, критический показатель для оценки ошибок классификации, равен $FPR\ = \frac{False\ Positives\ (FP)}{False\ Positives\ (FP)\ + True\ Negatives(TN)}$, вычисляемый как количественная оценка доли отрицательных экземпляров, ошибочно классифицированных моделью как положительные
Error rate (ER)	Частота ошибок, показатель, который количественно определяет общую неточность прогнозирования модели, рассчитывается как $Error\ Rate = \frac{False\ Positives\ (FP) + False\ Negatives\ (FN)}{Total\ Observations}$, эффективно измеряя долю всех неверных прогнозов модели
AUC score (AUC)	Показатель площади под кривой (AUC), измеряющий способность модели различать классы, рассчитывается путем сопоставления TPR с FPR при различных пороговых настройках, при этом значение AUC находится в диапазоне от 0 до 1, где более высокое значение указывает на лучшую эффективность классификации
Cohen's Kappa (KP)	Cohen's Kappa, статистический показатель взаимного согласия оценок для категориальных элементов, рассчитывается как $Kappa = \frac{P_o - P_e}{1 - P_e}$, где P_o представляет относительное наблюдаемое согласие между оценщиками, а P_e - гипотетическая вероятность случайного совпадения, что обеспечивает надежную оценку точности

	прогнозирования модели, выходящую за рамки случайной вероятности

Рисунок 2

Распределение вредоносных программ по категориям до балансировки SMOTE

В (табл. 4) представлена комплексная оценка различных классификаторов ensemble для многоклассовой классификации вредоносных программ, сравнивается их эффективность как с применением метода балансировки SMOTE, так и без него. Показатели дают целостное представление о производительности модели для различных классификаторов ensemble. Из таблицы видно, что система Gradient Boosting (GB) обладает превосходными характеристиками по всем параметрам, особенно в сочетании с балансировкой SMOTE, что позволяет добиться отличных результатов в ACC, PR, RE, FS и AUC. Это указывает на то, что GB ensemble, применяемый к сбалансированному набору данных, может эффективно идентифицировать и классифицировать различные типы вредоносных программ с максимальной точностью и достоверностью. Для сравнения, другие групповые методы, такие как Random Forest (RF), Bagging (BG), Voting (VT), and AdaBoost (ADB), демонстрируют различную производительность. В то время как RF, BG и VT превосходно работают и без балансировки, их эффективность повышается при использовании SMOTE, о чем свидетельствуют улучшенные показатели по большинству показателей. Однако ensemble AdaBoost демонстрирует заметное снижение производительности при применении балансировки, что позволяет предположить, что он может быть не столь эффективен при обработке сбалансированных наборов данных в данном конкретном контексте.

В (табл. 4) показана эффективность методов ensemble при классификации вредоносных программ по нескольким классам, в частности, Gradient Boosting, отличающийся непревзойденной производительностью, особенно в сочетании с балансировкой SMOTE. Это понимание подчеркивает важность выбора подходящих методов машинного обучения и сбалансированных стратегий для повышения точности и надежности моделей в приложениях кибербезопасности.

Матрица ‘путаницы’, изображенная на (рис. 3), полученная на основе бинарной классификации "доброкачественных" и "вредоносных программ", убедительно иллюстрирует исключительную эффективность модели в обнаружении вредоносных программ. Диагональные ячейки представляют количество истинно положительных и истинно отрицательных прогнозов, которые удивительно высоки для обоих классов. В частности, модель успешно идентифицировала 7279 экземпляров как "доброкачественные" и 7370 - как "Вредоносные программы" с абсолютной точностью, о чем свидетельствует отсутствие ложных срабатываний и ложноотрицательных результатов. Эта совершенная классификация указывает на то, что модель обладает исключительной способностью отличать доброкачественное программное обеспечение от вредоносного с максимальной чувствительностью и специфичностью. Визуализация матрицы путаницы в виде тепловой карты еще раз подчеркивает точность модели. Четкий контраст между высокими значениями по диагонали (истинные классификации) и нулями вне диагонали (ложные классификации) визуально подтверждает эффективность модели.

Результаты, представленные в матрице ‘путаницы’, являются свидетельством надежности модели в задачах бинарной классификации в области кибербезопасности. Безупречная точность в различении классов "доброкачественных" и "вредоносных программ" демонстрирует потенциал модели как бесценного инструмента для обнаружения вредоносных программ и обеспечения кибербезопасности, способного обеспечить точную и надежную защиту от цифровых угроз.

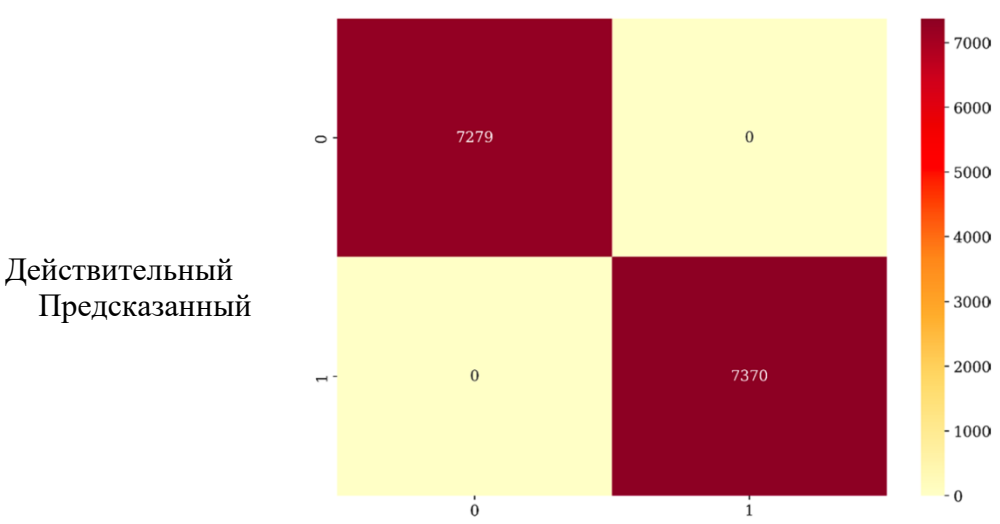
Таблица 4

Показатели производительности (взвешенные) модели с использованием различных методов объединения

Показатели	**Без уравновешивания**					**Уравновешивание с SMOTE**				
	GB	**RF**	**BG**	**VT**	**ADB**	**GB**	**RF**	**BG**	**VT**	**ADB**
ACC	0.99966	0.99932	0.99932	0.99951	0.83378	1.0000	0.99956	0.99997	0.99997	0.74783
PR	0.99966	0.99932	0.99932	0.99951	0.91541	1.0000	0.99956	0.99997	0.99997	0.87412
RE	0.99966	0.99932	0.99932	0.99951	0.83378	1.0000	0.99956	0.99997	0.99997	0.74783
FS	0.99966	0.99932	0.99932	0.99951	0.77772	1.0000	0.99956	0.99997	0.99997	0.66387
FPR	0.00017	0.00021	0.00021	0.00011	0.05020	1.0000	0.00015	0.00001	0.00001	0.08421
ER	0.00034	0.00068	0.00068	0.00049	0.16622	1.0000	0.00044	0.00003	0.00003	0.25217
CK	0.99949	0.99898	0.99898	0.99937	0.75105	1.0000	0.99941	0.99995	0.99995	0.66386
AUC	1.00000	0.99998	1.00000	0.99999	0.91667	1.0000	0.99997	1.00000	1.00000	0.91667

Рисунок 3

Матрица ‘путаницы’ для бинарной классификации



(Табл. 5), демонстрирующая результаты применения модели GB ensemble в контексте бинарной классификации, отражает исключительный уровень производительности с отличными оценками по всем ключевым показателям: точности, аккуратности, отзывчивости и F1-score, каждый из которых достигает максимально возможного значения 1.00000. Это выдающееся достижение подчеркивает беспрецедентную эффективность модели в точном различении классов "доброкачественных" и "вредоносных программ". Такой уровень точности особенно важен в области кибербезопасности, где способность надежно идентифицировать и классифицировать цифровые угрозы имеет первостепенное значение. Безупречная производительность модели по всем этим показателям свидетельствует о сбалансированном и высокоэффективном подходе к классификации, демонстрирующем ее потенциал как надежного инструмента для расширенного обнаружения вредоносных программ, что имеет решающее значение для защиты от развивающихся киберугроз.

На (рис. 4) показана локальная интерпретируемость разработанной модели в контексте обнаружения вредоносных программ. LIME (Local Interpretable Model - независимые объяснения) используется для разъяснения процесса принятия решений в базовой модели с помощью классификатора GB, что способствует прозрачности и пониманию искусственного интеллекта.

В этом сценарии второй экземпляр из тестовых данных относится к классификации на предмет того, представляет ли он вредоносное ПО или нет. На рисунке наглядно представлены ключевые факторы, влияющие на выбор модели для данного экземпляра. Значение перехвата, равное 0.18, соответствует базовой частоте прогнозирования модели, что указывает на вероятность общего прогноза без учета конкретных характеристик. Локальный прогноз 0.82 - это выходные данные модели для рассматриваемого экземпляра, отражающие вероятность того, что он будет классифицирован как вредоносное ПО. Вероятность правильного прогнозирования, равная 0.82, означает уверенность модели в правильной классификации экземпляра как вредоносного ПО.

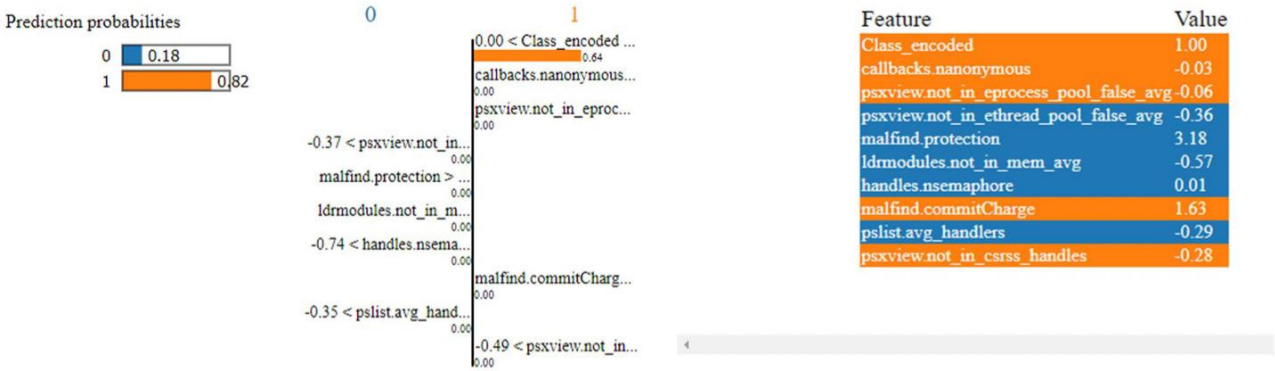
Таблица 5

Результаты работы модели в контексте бинарной классификации

Класс	**ACC**	**PR**	**RE**	**FS**
Доброкачественные (0)	1.0000	1.0000	1.0000	1.0000
Вредоностные(1)	1.0000	1.0000	1.0000	1.0000

Рисунок 4

Подробное объяснение классификации вредоносных программ



Используя “LimeTabularExplainer”, генерируются локальные объяснения для модели. Визуализированное объяснение обеспечивает интуитивное представление важных особенностей локального прогноза. Средняя часть рисунка, сопровождающего рисунок, детализирует вклад каждой детали в решение модели. Этот подход, основанный на LIME, облегчает интерпретацию модели, предоставляя прозрачное описание обоснования решения, что является важнейшим аспектом в обеспечении доверия и надежности в приложениях ИИ, особенно в области кибербезопасности, где точные и интерпретируемые прогнозы имеют первостепенное значение.

Матрица ‘путаницы’, показанная на (рис. 5), отражает выдающуюся производительность модели машинного обучения при классификации различных типов вредоносных программ с помощью метода балансировки SMOTE. Матрица, важнейший инструмент для оценки точности классификационных моделей, демонстрирует исключительную точность, с которой модель позволяет отличать доброкачественное программное обеспечение от вредоносного. Примечательно, что модель точно идентифицировала 7300 экземпляров как доброкачественные, 7364 - как программы-вымогатели, 7388 - как шпионские и 7246 - как вредоносные программы-трояны, без единой ошибки в классификации, что указывает на отсутствие ложных срабатываний и ложноотрицательных результатов во всех этих категориях. Такой уровень точности в различении отличных типов вредоносных программ, особенно в сложной области запутанных вредоносных программ, является замечательным достижением. Этот результат свидетельствует не только о способности модели обнаруживать и анализировать широкий спектр сложных киберугроз, но и подчеркивает ее потенциал как надежного инструмента в арсенале средств защиты от угроз кибербезопасности. Высший балл в матрице путаницы подчеркивает мастерство модели в детектировании нюансов, что жизненно важно как для предотвращения ложных тревог, так и для обеспечения того, чтобы ни одна вредоносная активность не осталась незамеченной.

В (табл. 6) представлены показатели оценки для различных категорий вредоносных программ после внедрения технологии SMOTE и без нее. Также там представлен примерный уровень производительности во всех классах, при этом каждый показатель точности, аккуратности, отзывчивости и F1-score достигает максимального значения 1.0000. Такое единообразие результатов по всем категориям свидетельствует об исключительном уровне эффективности модели, особенно после балансировки набора данных с помощью SMOTE. Достижение идеальных результатов по каждому показателю для каждого класса (обозначенных как 0, 1, 2 и 3) демонстрирует способность модели обнаруживать и классифицировать различные типы вредоносных программ с безупречной точностью. С другой стороны, показатели оценки для тех же категорий вредоносных программ оцениваются без применения SMOTE. Здесь, несмотря на то, что результаты немного ниже, они по-прежнему демонстрируют выдающийся уровень производительности модели. Для класса 0 точность составляет 0,9997, с точностью 0,9993 и показателем F1 0,9997, в то время как отзыв остается идеальным при 1,0000. В классе 1 количество отзывов немного снизилось до 0,9979, но сохраняется на высоком уровне по другим показателям. Классы 2 и 3, как и в сбалансированном случае, сохраняют отличные показатели по всем показателям.

Рисунок 5

Матрица ‘путаницы’, иллюстрирующая эффективность различных категорий вредоносных программ

Матрица ‘путаницы’

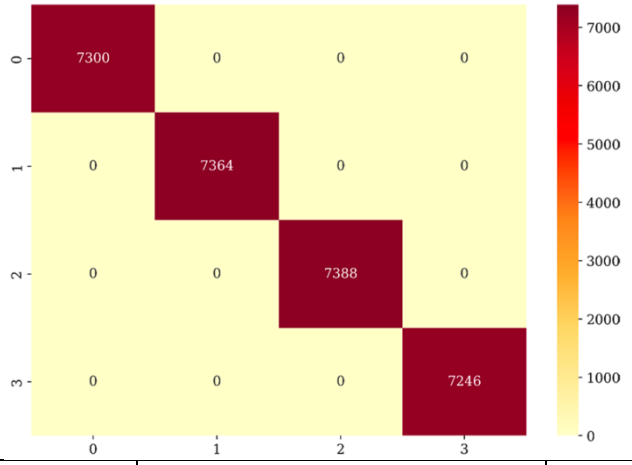


Таблица 6

Показатели оценки для различных категорий вредоносных программ

Категории	**Без уравновешивания**				**Уравновешивание с SMOTE**			
	ACC	**PR**	**RE**	**FS**	**ACC**	**PR**	**RE**	**FS**
Доброкачественные (0)	0.9997	0.9993	1.0000	0.9997	1.0000	1.0000	1.0000	1.0000
Программа-вымогатель (1)	0.9997	1.0000	0.9979	0.9990	1.0000	1.0000	1.0000	1.0000
Шпионские ПО (2)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Троян(3)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

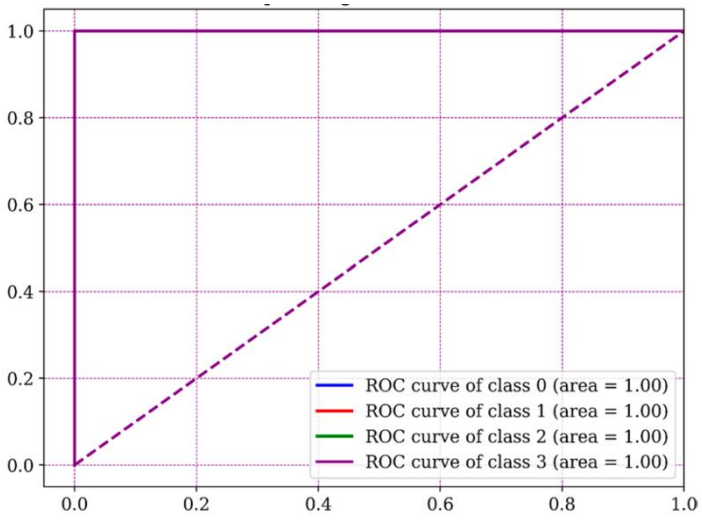
Сравнение этих двух значений из двух разных ситуаций подчеркивает надежность и адаптивность модели в различных сценариях обработки данных. Использование SMOTE явно повысило эффективность модели при устранении дисбаланса классов, о чем свидетельствует улучшение показателей для классов 0 и 1. Это улучшение важно, поскольку оно демонстрирует эффективность модели не только в обнаружении вредоносных программ, но и в поддержании высокой точности и восстановлении сбалансированного набора данных, что часто является проблемой в моделях машинного обучения, работающих с несбалансированными данными.

На (рис. 6) представлены кривые рабочих характеристик приемника (ROC) для каждого класса в модели, что свидетельствует о его исключительной эффективности при классификации вредоносных программ. Кривая ROC - это графическое представление, иллюстрирующее диагностические возможности системы бинарного классификатора, эффективность которой измеряется площадью под кривой (AUC). В данном случае каждый из классов 0, 1, 2 и 3 соответствует различным типам вредоносных программ. Примечательно, что AUC для каждого сорта на (рис. 5) - это 1.00, что является редким и заслуживающим похвалы достижением в моделях машинного обучения, особенно в сложной области кибербезопасности. Этот высший балл указывает на то, что модель обладает исключительной способностью проводить различия между классами с максимальной чувствительностью и специфичностью. Чувствительность (or True Positive Rate) отражает способность модели правильно идентифицировать положительные результаты, в то время как специфичность (or 1—False Positive Rate) указывает на ее способность правильно классифицировать отрицательные результаты. Кривые ROC для всех классов расположены в верхнем левом углу графика, что является идеальным положением, указывающим на незначительную частоту ложноположительных результатов и высокую частоту истинно положительных результатов во всех классах. Это означает, что модель очень эффективна при различении различных типов вредоносных программ с минимальной ошибкой в классификации.

Совершенство этих кривых, особенно при работе с несколькими классами, говорит о том, что лежащие в их основе алгоритмы, методы предварительной обработки и методы выбора характеристик исключительно хорошо отлажены. Достижение показателя AUC в 1.00 для нескольких классов в такой сложной области, как обнаружение вредоносных программ, не является тривиальным и красноречиво свидетельствует о тщательности разработки и реализации модели. Безупречные показатели AUC для всех классов подтверждают статус модели как надежного инструмента в области кибербезопасности, способного обеспечивать высокоточную классификацию. Такой уровень точности имеет решающее значение для эффективных мер кибербезопасности, поскольку стоимость неправильной классификации может быть чрезвычайно высока. Продемонстрированные возможности модели делают ее значительным достижением в борьбе с киберугрозами и открывают многообещающие перспективы для будущих применений в области цифровой безопасности.

Рисунок 6

Кривая ROC модели



(Рис. 7), на котором представлены результаты обучения и перекрестной проверки в зависимости от размера обучающей выборки, дает полное представление о динамике обучения модели и ее эффективности при классификации вредоносных программ. График, на котором по оси x отложены размеры обучающего набора, а по оси y - показатели точности, является важным инструментом для оценки производительности и обобщаемости модели. На этом рисунке оценка за обучение (показана красным цветом) и оценка за перекрестную проверку (показана фиолетовым цветом) демонстрируют тенденцию к сближению по мере увеличения размера тренировочного набора. Такая сходимость является отличительной чертой хорошо работающей модели, указывающей на то, что она не только эффективно учится на основе обучающих данных, но и хорошо обобщает невидимые данные, что отражается в результатах перекрестной проверки.

Примечательно, что показатели точности как при обучении, так и при перекрестной проверке исключительно высоки, что свидетельствует о надежности модели. Высокая точность при обучении позволяет предположить, что модель эффективно отражает основные закономерности в данных. Что еще более важно, высокая точность перекрестной проверки указывает на способность модели поддерживать такую производительность на новых, невидимых данных, что является критически важным аспектом для практического применения.

Результаты, представленные в (табл. 7), подчеркивают замечательную эффективность и согласованность модели с классификатором постепенного повышения эффективности при классификации вредоносных программ как с применением метода синтетической избыточной выборки по незначительной величине (SMOTE), так и без него. В первой части, где подробно описаны результаты перекрестной проверки без балансировки, модель демонстрирует отличные эксплуатационные характеристики во всех аспектах. Показатели ACC, PR, RE и FS стабильно высоки, преимущественно на уровне 0.9997, за исключением показателя Fold 3, который показывает незначительно более низкий, но все же впечатляющий результат – 0.9994. Средние баллы по всем показателям составляют 0.9997 при очень низком стандартном отклонении в 0.0002. Такая согласованность свидетельствует не только о высокой способности модели правильно классифицировать различные типы вредоносных программ, но и о ее надежности в различных наборах данных. Вторая часть, представляющая сбалансированные результаты с помощью SMOTE, демонстрирует еще более исключительную производительность, с идеальными оценками в 1.0000 баллов по всем показателям и кратностям. Это указывает на то, что модель, обученная на сбалансированном наборе данных, может обеспечить безупречную классификацию без каких-либо изменений в производительности при различных этапах перекрестной проверки. Стандартное отклонение в 0.0000 еще больше повышает стабильность и достоверность модели, подчеркивая ее эффективность в решении классовых дисбалансов, что является распространенной проблемой в машинном обучении.

Рисунок 7

Кривая обучения модели

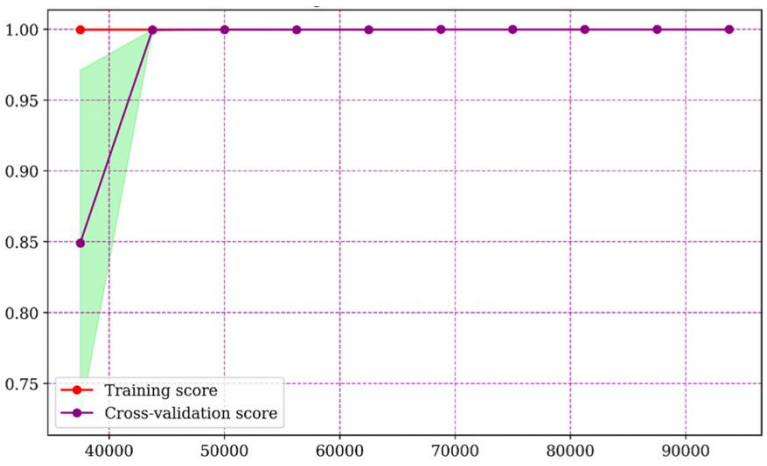


Таблица 7

Обобщенные результаты перекрестной проверки для многоклассовой классификации

Отклонения	**Без уравновешивания**				**Уравновешивание с SMOTE**			
	ACC	**PR**	**RE**	**FS**	**ACC**	**PR**	**RE**	**FS**
Отклонение 1	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Отклонение 2	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Отклонение 3	0.9994	0.9994	0.9994	0.9994	1.0000	1.0000	1.0000	1.0000
Отклонение 4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Отклонение 5	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Значение	0.9997	0.9997	0.9997	0.9997	1.0000	1.0000	1.0000	1.0000
Стандартное отклонение	0.0002	0.0002	0.0002	0.0002	0.0000	0.0000	0.0000	0.0000

Эти результаты демонстрируют исключительную точность и последовательность модели в обнаружении и классификации вредоносных программ, что имеет решающее значение в области кибербезопасности, где неправильная классификация может привести к значительным потерям. Использование SMOTE для балансировки набора данных повышает способность модели к обобщению различных распределений данных, что является ключевым аспектом в обеспечении ее применимости к реальным сценариям, где данные часто могут быть несбалансированными.

В (табл. 8) тщательно сопоставлены показатели производительности предложенной модели с несколькими существующими моделями в области классификации вредоносных программ, как для бинарных, так и для 4-классных элементов. Сравниваемые показатели включают в себя ACC, PR, RE и FS, каждый из которых оценивается для бинарной классификации и классификации по 4 классам. Эта таблица имеет ключевое значение для демонстрации расширенных возможностей предлагаемой модели в точном обнаружении и классификации вредоносных программ. При использовании метода синтетической незначительной избыточной выборки (SMOTE) для балансировки все значения оценочных показателей неизменно достигают идеального балла 1.00 как для бинарных, так и для четырех категорий вредоносных программ. Предлагаемая модель демонстрирует беспрецедентный уровень производительности, достигая экстраординарных 100% по всем показателям для бинарной классификации и почти идеальных результатов в 99.96% для классификации по 4 классам для тестовых данных (20%). Эта исключительная производительность резко контрастирует с другими моделями, перечисленными в таблице. Хотя эти модели демонстрируют похвальную точность в бинарной классификации, они не дотягивают до более сложной классификации по 4 классам, с точностью от 79.16 до 85.04%, и аналогичной тенденцией в других показателях.

Способность предлагаемой модели поддерживать высокую точность и согласованность как в бинарных, так и в многоклассовых сценариях отличает ее от аналогичных моделей. Это указывает не только на точность модели в классификации вредоносных программ, но и на ее надежность при работе с более сложными многоклассовыми элементами. Такая производительность имеет решающее значение в быстро меняющемся мире кибербезопасности, где способность точно различать различные типы атак имеет первостепенное значение.

Таблица 8

Модель, год выпуска с указанием	**Показатели точности (%)**							
	ACC		**PR**		**RE**		**FS**	
	Двоичный	**4 класс**	**Двоичный**	**4 класс**	**Двоичный**	**4 класс**	**Двоичный**	**4 класс**

Показатели эффективности: предлагаемая модель в сравнении с существующими без сбалансированности

MalHyStack, 2023 (Roy et al. 2023) [16]	99.85	85.04	99.97	85.04	99.73	85.17	99.85	84.96
RobustCBL, 2023 (Shafin et al. 2023) [17]	99.96	84.56	100.00	85.00	100.00	85.00	100.00	85.00
CatBoost, 2022 (Dang 2022) [27]	99.97	84.86	99.98	79.69	99.98	88.46	99.97	71.49
DT, 2022 (Mezina and Burget 2022) [15]	99.00	79.16	99.00	69.00	100.00	69.00	99.00	69.00
DCNN, 2022 (Mezina and Burget 2022) [15]	99.92	83.53	99.00	76.00	99.00	75.00	99.00	75.00
Предложенный	100.00	99.96	100.00	99.96	100.00	99.96	100.00	99.96

В (табл. 9) представлен всесторонний обзор показателей оценки модели по 16 различным категориям атак без учета метода SMOTE, включая доброкачественные и различные типы вредоносных программ, программ-шпионов и троянов. Если набор данных сбалансирован, то все значения различных показателей дают значение 1.0000. Модель демонстрирует выдающуюся производительность при обнаружении и классификации различных категорий вредоносных программ. Высокие значения по всем показателям свидетельствуют о том, что модель отличается высокой точностью и эффективностью в распознавании различных категорий атак, что делает ее надежным решением для обнаружения вредоносных программ в широком спектре угроз.

Таблица 9

Показатели оценки для 16 категорий атак

Класс	ACC	PR	RE	FS
Доброкачественные (0)	0.9999	0.9997	1.0000	0.9999
Вирус-вымогатель_Ako (1)	0.9999	1.0000	0.9980	0.9990
Вирус-вымогатель_Conti (2)	0.9999	1.0000	0.9981	0.9990
Вирус-вымогатель_Maze (3)	1.0000	1.0000	1.0000	1.0000
Вирус-вымогатель_Pusa (4)	1.0000	1.0000	1.0000	1.0000
Вирус-вымогатель_Shade (5)	1.0000	1.0000	1.0000	1.0000
Шпионское ПО_180solutions (6)	1.0000	1.0000	1.0000	1.0000
Шпионское ПО_CWS (7)	1.0000	1.0000	1.0000	1.0000
Шпионское ПО_Gator (8)	1.0000	1.0000	1.0000	1.0000
Шпионское ПО_TIBS (9)	1.0000	1.0000	1.0000	1.0000
Шпионское ПО_Transponder (10)	1.0000	1.0000	1.0000	1.0000
Троян_Emotet (11)	1.0000	1.0000	1.0000	1.0000
Троян_Reconyc (12)	1.0000	1.0000	1.0000	1.0000
Троян_Refroso (13)	1.0000	1.0000	1.0000	1.0000
Троян_Scar (14)	1.0000	1.0000	1.0000	1.0000
Троян_Zeus (15)	1.0000	1.0000	1.0000	1.0000

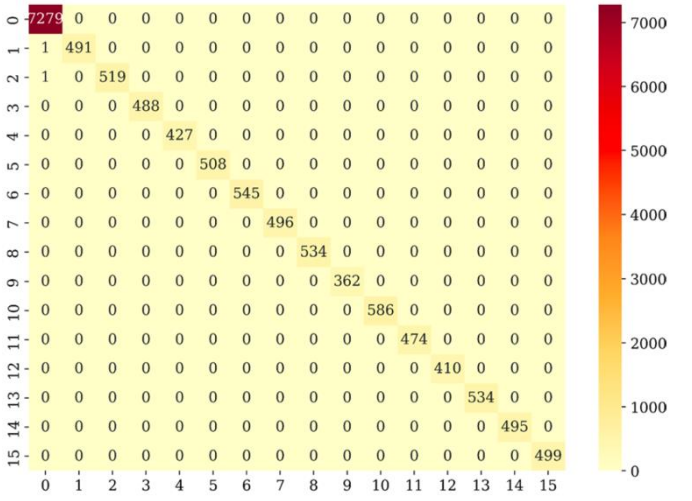
На (рис. 8), демонстрирующем матрицу путаницы для 16 классов вредоносных программ, подверженных атакам, представлена подробная информация о возможностях классификации модели без использования метода SMOTE. Тепловая карта, созданная с четкостью и аккуратностью, иллюстрирует распределение истинных и ложных прогнозов по различным категориям вредоносных программ. Ярко выделенная основная диагональ указывает на большое количество положительных результатов для каждого класса, что иллюстрирует точность модели при правильной идентификации каждого конкретного типа вредоносных программ. Когда набор данных сбалансирован с использованием метода SMOTE, модель достигает замечательной 100%-ной точности во всех классах вредоносных программ,

подверженных субатакам.

Классы, варьирующиеся от "доброкачественных" до различных типов "Программ-вымогателей", "программ-шпионов" и "троянских программ", подразделяются на подклассы с практически нулевыми ложными срабатываниями и ложноотрицательными ошибками. Это проявляется в минимальном количестве недиагональных элементов, что подчеркивает точность модели в различении различных типов вредоносных программ, что является критическим фактором эффективной кибербезопасности. Тщательная идентификация "доброкачественных" случаев среди множества типов вредоносных программ еще раз подчеркивает тонкость понимания модели и возможности ее обнаружения. Особой похвалы заслуживает эффективность модели в точной классификации сложных разновидностей вредоносных программ с высокими показателями истинных положительных результатов и практически нулевым количеством ошибок в классификации. Цветовые градации тепловой карты, варьирующиеся от темно-красных до светло-оранжевых, обеспечивают интуитивное и мгновенное понимание точности классификации модели.

Рисунок 8

Подробная визуализация матрицы ‘путаницы’ для 16 подклассов вредоносных программ



В (табл. 10) представлено убедительное сравнение показателей производительности предлагаемой модели с существующей моделью MalHyStack по 16 категориям атак. Оцениваемые показатели включают ACC, PR, RE и FS, которые являются важнейшими показателями эффективности модели в задачах классификации.

Предлагаемая модель демонстрирует исключительный уровень производительности, при этом каждый показатель достигает почти идеальных значений в 99.98%. Это значительное улучшение по сравнению с существующими моделями. Резкий контраст между этими значениями подчеркивает расширенные возможности предлагаемой модели в точном выявлении и классификации широкого спектра вредоносных атак.

Таблица 10

Сравнение показателей производительности модели для 16 категорий атак без учета балансировки

Модель	ACC	PR	RE	FS
MalHyStack, 2023 (Roy et al. 2023) [16]	66.94	66.94	68.56	66.71
RobustCBL, 2023 (Shafin et al. 2023) [17]	72.60	73.00	73.00	72.00
Предложенный	99.98	99.98	99.98	99.98

Предлагаемый подход с использованием ускоряющих градиент классификаторов, известных своими возможностями коллективного обучения, позволяет предлагаемой модели создавать надежные и сложные границы принятия решений путем объединения нескольких слабых учащихся. Такой комплексный подход повышает способность модели фиксировать сложные взаимосвязи в данных, что обеспечивает превосходную точность прогнозирования. SMOTE используется в сочетании с классификаторами, повышающими градиент, для устранения дисбаланса классов. За счет избыточной выборки малочисленных классов предлагаемая модель обеспечивает более сбалансированное представление различных типов вредоносных программ, тем самым предотвращая предвзятое отношение к доминирующим классам. Такая стратегическая обработка несбалансированных данных повышает чувствительность модели и ее обобщение по различным категориям вредоносных программ. При разработке модели применяются строгие методы отбора признаков, включая статистические тесты и информационно-теоретические подходы. Этот тщательный процесс позволяет выделить ключевые характеристики вредоносного ПО, что позволяет модели сосредоточиться на наиболее показательных характеристиках для точной классификации. Акцент на информативных характеристиках способствует точности и надежности модели. Предложенная модель демонстрирует адаптивность как к бинарным, так и к многоклассовым элементам, демонстрируя свою универсальность в борьбе с широким спектром угроз в кибербезопасности. Такая адаптивность гарантирует, что модель остается эффективной в различных цифровых средах, превосходя ограничения производительности, наблюдаемые в некоторых существующих моделях, которые могут специализироваться на конкретных сценариях. Превосходная производительность предлагаемой модели по сравнению с другими является результатом ее надежного коллективного обучения, эффективной обработки несбалансированных данных с помощью SMOTE, тщательного отбора функций, адаптации к различным сценариям и стремления к непрерывным инновациям. Все эти факторы в совокупности обеспечивают исключительные показатели ACC, PR, RE и FS, позиционируя предлагаемую модель как современное решение в области обнаружения вредоносных программ.

Всесторонний анализ различных рисунков и таблиц в этом исследовании однозначно подтверждает превосходство предложенной модели в классификации вредоносных программ. Его производительность, подтверждаемая почти идеальными показателями по ключевым показателям в нескольких таблицах, свидетельствует об исключительной точности, отзывчивости и F1-показателях как в бинарных, так и в мультиклассовых элементах. Надежность модели еще больше подчеркивается последовательностью полученных результатов, даже при наличии дисбаланса в классе, о чем свидетельствует повышение производительности с помощью технологии SMOTE. Матрицы ‘путаницы’, подробно описанные на рисунках, иллюстрируют беспрецедентную способность модели различать широкий спектр вредоносных программ с минимальными ошибками в классификации. В совокупности эти результаты не только подтверждают передовые аналитические возможности модели в сложной области кибербезопасности, но и демонстрируют ее потенциал как высокоэффективного инструмента обнаружения и анализа скрытых вредоносных программ, внося значительный вклад в эту область и устанавливая новые ориентиры для будущих исследований.

Вывод

В заключение, это исследование представляет собой новаторский подход в области кибербезопасности, направленный на обнаружение и анализ скрытого вредоносного ПО с помощью продвинутой системы, основанной на машинном обучении. Всесторонняя оценка результатов исследования с использованием набора данных Obfuscated-MalMem2022 демонстрирует исключительную способность модели точно классифицировать различные типы вредоносных программ. Применение таких методов, как SMOTE, для устранения классового дисбаланса еще больше повышает производительность модели, обеспечивая почти идеальную точность как в бинарных, так и в многоклассовых элементах (как в 4, так и в 16 классах). Модель успешно достигает точности, превышающей 99%, в трех различных тестах. Подробный анализ, отраженный в рисунках и таблицах, показывает способность модели поддерживать высокие результаты в различных классификациях, что отличает ее от существующих моделей. Надежность предлагаемой модели проявляется в ее способности выполнять сложные задачи классификации по нескольким классам с поразительной точностью, что является важнейшим требованием в современной динамичной среде кибербезопасности. Это исследование не только решает важнейшую задачу обнаружения сложных, запутанных вредоносных программ, но и вносит значительный вклад в область кибербезопасности, предоставляя надежный и эффективный инструмент для практиков и исследователей. Адаптивность и эффективность модели позволяют использовать ее в качестве ориентира для будущих разработок в области решений для обеспечения кибербезопасности, подчеркивая потенциал машинного обучения в борьбе со все более изощренными киберугрозами. Таким образом, это исследование является важной вехой в продолжающихся усилиях по повышению цифровой безопасности и защите от меняющегося ландшафта киберугроз.

Таблица 11

Название объекта и описание набора данных

Название функции	Описание функции
callbacks.ncallbacks	Это количественная оценка количества зарегистрированных функций обратного вызова в системе, необычно высокое количество которых может свидетельствовать о том, что вредоносное ПО пытается перехватить или отслеживать действия системы
pslist.avg_handlers	Это среднее число обработчиков для каждого процесса, позволяющее получить представление о поведении системы; запутанное вредоносное ПО может манипулировать этим, чтобы скрыть свое присутствие или управлять другими процессами
psxview.not_in_eprocess_pool_false_avg	Это отражает среднее число процессов, не перечисленных в пуле электронных процессов, которое может быть повышено в случае запутанного вредоносного ПО, пытающегося скрыться из обычных списков процессов
ldrmodules.not_in_load	Это количественная оценка модулей, которые не находятся в загруженном состоянии, что является возможным сигналом для скрытого вредоносного ПО, которое может выгружать модули, чтобы избежать

	обнаружения
psxview.not_in_csrss_handles_false_avg	Это среднее количество процессов, не обнаруженных в обработчиках CSRSS, что потенциально указывает на запутанное вредоносное ПО, поскольку оно может пытаться избежать привязки к критически важным системным процессам
handles.nevent	Это отслеживает количество дескрипторов событий, которыми может манипулировать запутанное вредоносное ПО в целях синхронизации или для поддержания постоянства
handles.nmutant	Здесь указано количество мутантных дескрипторов, которые запутанная вредоносная программа может использовать для передачи сигналов между различными экземплярами самой себя или для блокировки ресурсов
psxview.not_in_eprocess_pool	Это позволяет подсчитать количество процессов, отсутствующих в пуле EPROCESS, - атрибут, который может быть использован запутанным вредоносным ПО для того, чтобы остаться незамеченным
dlllist.avg_dlls_per_proc	Эта функция отражает среднее количество библиотек DLL, загружаемых за один процесс, при этом запутанная вредоносная программа может загружать необычные библиотеки DLL или манипулировать этим количеством, чтобы скрыть свое присутствие
psxview.not_in_deskthrd_false_avg	Это показывает среднее количество процессов, не обнаруженных в потоке рабочего стола, что является потенциальным показателем скрытого вредоносного ПО, поскольку оно может отключить свои процессы от рабочего стола, чтобы остаться невидимым
handles.nthread	Это количественная оценка количества дескрипторов потоков, которое запутанное вредоносное ПО может увеличить для параллельного выполнения или манипулирования другими процессами
callbacks.nanonymous	Здесь отслеживается количество анонимных обратных вызовов, при этом запутанное вредоносное ПО потенциально регистрирует такие обратные вызовы, чтобы избежать установления авторства
modules.nmodules	Это общее количество загруженных модулей, которое может быть завышено из-за запутанной вредоносной программы, поскольку она загружает дополнительные модули для вредоносных действий
ldrmodules.not_in_mem_avg	Фиксируется среднее количество модулей, отсутствующих в памяти, что может свидетельствовать о скрытой вредоносной программе, которая выгружает модули, чтобы избежать обнаружения на основе памяти
handles.nsemaphore	Это количественная оценка количества дескрипторов семафоров, которыми запутанное вредоносное ПО

	может манипулировать для координации или контроля доступа к ресурсам
svcscan.fs_drivers	Это отражает количество драйверов файловой системы, которые могут быть использованы вредоносным ПО для установки вредоносных драйверов или перехвата файловых операций
svcscan.shared_process_services	Здесь указано количество служб, запущенных в общих процессах, при этом запутанное вредоносное ПО, возможно, внедряется в такие службы для скрытности
ldrmodules.not_in_init_avg	Эта функция отражает среднее количество неинициализированных модулей, что является потенциальным признаком скрытого вредоносного ПО, поскольку оно может попытаться нарушить нормальную инициализацию модуля
svcscan.process_services	Отслеживается количество служб, запущенных в отдельных процессах, - функция, которую запутанное вредоносное ПО может использовать для независимого запуска своих вредоносных служб
handles.nsection	Это количественная оценка количества дескрипторов разделов, которыми может манипулировать обфускированная вредоносная программа для отображения памяти или для сокрытия своего кода в определенных разделах
pslist.nprocs64bit	Представлено количество 64-разрядных процессов, запущенных в системе, на которое может повлиять запутанная вредоносная программа при выборе процессов для внедрения или олицетворения
pslist.nppid	Это отслеживает количество процессов на основе идентификаторов родительских процессов, которыми вредоносная программа может манипулировать, чтобы нарушить связь между родительским и дочерним процессами и скрыть их происхождение
handles.avg_handles_per_proc	Это среднее количество дескрипторов для каждого процесса, и эта цифра может быть завышена из-за запутанного вредоносного ПО, поскольку оно открывает множество дескрипторов для вредоносных действий
dlllist.ndlls	Здесь указано общее количество загруженных библиотек DLL, при этом запутанная вредоносная программа может загружать дополнительные библиотеки DLL для выполнения своей полезной нагрузки или уклонения
svcscan.nactive	Эта функция фиксирует количество активных служб, которое запутанное вредоносное ПО может увеличить, регистрируя свои собственные службы или перехватывая существующие
handles.nport	Отслеживается количество дескрипторов портов - функция, которую скрытое вредоносное ПО может

	использовать для установления сетевых соединений или перехвата действий, связанных с сетью
malfind.uniqueInjections	Это число отражает количество обнаруженных уникальных внедрений в память, что является важным показателем наличия скрытого вредоносного ПО, поскольку оно часто использует внедрение в память для скрытности и сохранения
psxview.not_in_pslist	Указывается количество процессов, отсутствующих в списке процессов, что является потенциальным признаком запутанного вредоносного ПО, пытающегося скрыть свои процессы от стандартных списков
psxview.not_in_pspcid_list	Это количественная оценка процессов, не найденных в списке PSPCID, что может свидетельствовать о скрытом вредоносном ПО, использующем передовые методы, позволяющие оставаться незамеченным
psxview.not_in_pspcid_list_false_avg	Регистрируется среднее количество процессов, отсутствующих в списке PSPCID, что потенциально указывает на попытки запутанного вредоносного ПО избежать обнаружения на уровне ядра
pslist.nproc	Это общее количество запущенных процессов, на которое может повлиять запутанная вредоносная программа, поскольку она запускает дополнительные процессы для своих операций
handles.ndesktop	Отслеживается количество дескрипторов десктопов, и запутанное вредоносное ПО может манипулировать этой функцией, чтобы запускать процессы на изолированных десктопах для уклонения
malfind.commitCharge	Это количественно определяет стоимость фиксации обнаруженных инъекций памяти, что является важной особенностью для анализа, поскольку запутанное вредоносное ПО часто манипулирует памятью для выполнения кода и скрытности
psxview.not_in_session	Указано количество процессов, отсутствующих ни в одном сеансе, что, возможно, указывает на то, что вредоносная программа пытается изолировать свои процессы от пользовательских сеансов
handles.ndirectory	При этом отслеживается количество дескрипторов каталогов, которыми запутанная вредоносная программа может манипулировать для взаимодействия с каталогами файловой системы или мониторинга их состояния
psxview.not_in_csrss_handles	Количество процессов, не обнаруженных в обработчиках CSRSS, определяется количественно, что является потенциальным предупреждением для скрытого вредоносного ПО, поскольку оно может разорвать связи с критически важными системными процессами

psxview.not_in_pslist_false_avg	Это показывает среднее количество процессов, не найденных в списке процессов, что потенциально указывает на попытки запутанного вредоносного ПО остаться скрытым от обычного подсчета процессов
psxview.not_in_deskthrd	Указывается количество процессов, не найденных в потоке рабочего стола, - функция, с помощью которой запутанное вредоносное ПО может манипулировать, чтобы отделить свои процессы от пользовательских интерфейсов
malfind.protection	Это количественная оценка характеристик защиты обнаруженных внедрений в память, что является важной функцией для анализа, поскольку запутанное вредоносное ПО может манипулировать настройками защиты, чтобы выполнить вредоносный код, избегая обнаружения
pslist.avg_threads	Отслеживается среднее количество потоков на процесс, что может быть увеличено с помощью запутанного вредоносного ПО, поскольку оно создает дополнительные потоки для параллельного выполнения или манипулирования другими процессами
ldrmodules.not_in_init	Это позволяет подсчитать количество модулей, которые не находятся в инициализированном состоянии, что, возможно, указывает на запутанную вредоносную программу, пытающуюся нарушить обычные процедуры инициализации
ldrmodules.not_in_mem	Количество модулей, отсутствующих в памяти, определяется количественно, что является потенциальным признаком того, что запутанное вредоносное ПО выгружает модули после выполнения, чтобы избежать обнаружения
psxview.not_in_session_false_avg	Эта функция фиксирует среднее количество процессов, не обнаруженных ни в одном сеансе, что потенциально указывает на попытки вредоносного ПО изолировать свои действия от пользовательских сеансов
malfind.ninjections	Указывается общее количество обнаруженных внедрений в память, что является важным показателем для обнаружения скрытого вредоносного ПО, поскольку такие методы часто используются для выполнения кода и уклонения от него
svcscan.interactive_process_services	Это количественная оценка количества служб, запущенных в интерактивных процессах, на которые может быть нацелена запутанная вредоносная программа для запуска своих служб с повышенными привилегиями
psxview.not_in_ethread_pool	Отслеживается количество процессов, отсутствующих в пуле потоков, что является потенциальным сигналом

	для скрытого вредоносного ПО, использующего передовые методы обхода на уровне ядра
ldrmodules.not_in_load_avg	Это среднее количество незагруженных модулей, что, возможно, указывает на попытки вредоносного ПО манипулировать загрузкой модулей с целью уклонения от загрузки.
handles.nfile	Количество дескрипторов файлов определяется количественно, и запутанное вредоносное ПО может манипулировать этим количеством для взаимодействия с файлами или их скрытия
handles.ntimer	Эта функция отслеживает количество дескрипторов таймера, которые запутанная вредоносная программа может использовать для планирования действий или поддержания постоянства
callbacks.ngeneric	Предусмотрен подсчет общих обратных вызовов - функция, которую запутанное вредоносное ПО может использовать для мониторинга или перехвата системных действий без привязки к конкретным событиям
handles.nkey	Это количественная оценка количества дескрипторов разделов реестра, количество которых может быть увеличено из-за запутанной вредоносной программы, взаимодействующей с реестром для настройки, сохранения или хранения полезной нагрузки
svcscan.kernel_drivers	Отслеживается количество драйверов ядра, и запутанное вредоносное ПО потенциально нацелено на эту область для установки вредоносных драйверов или манипулирования загрузкой драйверов с целью уклонения от их использования
psxview.not_in_ethread_pool_false_avg	Это отражает среднее число процессов, не обнаруженных в пуле ETHREAD, что потенциально указывает на попытки запутанного вредоносного ПО остаться незамеченным на уровне ядра
handles.nhandles	Здесь указано общее количество открытых дескрипторов, что может быть увеличено из-за запутанного вредоносного ПО, поскольку оно открывает множество дескрипторов для своих вредоносных действий
svcscan.nservices	Это общее количество запущенных служб, на которое может повлиять запутанное вредоносное ПО, добавляя свои собственные службы или перехватывая существующие

Книга одного автора:

10. Dang Q-V (2024) Detecting obfuscated malware using graph neural networks. In: Shrivastava V, Bansal JC, Panigrahi BK (eds) Power engineering and intelligent systems, vol 1097. Springer, Singapore, pp 15–25.

26. Hossain MA (2023) Enhanced ensemble-based distributed denial-of-service (DDoS) attack detection with novel feature selection: a robust cybersecu- rity approach. Artif Intell Evol 4(2):165–186.

27. Dang Q-V (2022) Enhancing obfuscated malware detection with machine learning techniques. In: Dang TK, Küng J, Chung TM (eds) Future data and security engineering. Big data, security and privacy, smart city and indus- try 4.0 applications, vol 1688. Springer, Singapore, pp 731–738.

Книга двух авторов:

3. Brezinski K, Ferens K (2023) Metamorphic malware and obfuscation: a survey of techniques, variants, and generation kits. Secur Commun Netw 2023:1–41.

12. Haidros Rahima Manzil H, Manohar Naik S (2023) Detection approaches for android malware: taxonomy and review analysis. Expert Syst Appl.

15. Mezina A, Burget R (2022). Obfuscated malware detection using dilated con- volutional network. In: 2022 14th international congress on ultra modern telecommunications and control systems and workshops (ICUMT), pp 110–115.

19. Hossain MA, Islam MS (2023a) Ensuring network security with a robust intru- sion detection system using ensemble-based machine learning. Array.

20. Hossain MA, Islam MS (2023b) A novel hybrid feature selection and ensemble- based machine learning approach for botnet detection. Sci Rep 13(1):21207.

21. Mamdouh Farghaly H, Abd El-Hafeez T (2023) A high-quality feature selec- tion method based on frequent and correlated items for text clas- sification. Soft Comput 27(16):11259–11274.

23. Chen Z, Ren X (2023) An efficient boosting-based windows malware family classification system using multi-features fusion. Appl Sci 13(6):4060.

Книга трех авторов:

4. Gorment NZ, Selamat A, Krejcar O (2023) Obfuscated malware detection: impacts on detection methods. In: Nguyen NT, Boonsang S, Fujita H, Hnatkowska B, Hong T-P, Pasupa K, Selamat A (eds) Recent challenges in intelligent information and database systems, vol 1863. Springer, Cham, pp 55–66.

7. Finder I, Sheetrit E, Nissim N (2022) A time-interval-based active learning framework for enhanced PE malware acquisition and detection. Comput Secur 121:102838.

13. Lee K, Lee J, Yim K (2023) Classification and analysis of malicious code detec- tion techniques based on the APT attack. Appl Sci 13(5):2894.

17. Shafin SS, Karmakar G, Mareels I (2023) Obfuscated memory malware detec- tion in resource-constrained IoT devices for smart city applications. Sensors 23(11):5348.

22. Federici M, Ruhe D, Forré P (2023) On the effectiveness of hybrid mutual infor- mation estimation.

24. Ngo G, Beard R, Chandra R (2022) Evolutionary bagging for ensemble learning. Neurocomputing 510:1–14.

25. Vashishtha LK, Chatterjee K, Rout SS (2023) An ensemble approach for advance malware memory analysis using image classification techniques. J Inf Secur Appl 77:103561.

Книги четырех и более авторов:

1. Asghar HJ, Zhao BZH, Ikram M, Nguyen G, Kaafar D, Lamont S, Coscia D (2023) Use of cryptography in malware obfuscation (arXiv:2212.04008; Issue arXiv:2212.04008). http://arxiv.org/abs/2212.040087.

2.Bozkir AS, Tahillioglu E, Aydos M, Kara I (2021) Catch them alive: a malware detection approach through memory forensics, manifold learning and computer vision. Comput Secur 103:102166.

5. Beaman C, Barkworth A, Akande TD, Hakak S, Khan MK (2021) Ransomware: recent advances, analysis, challenges and future research directions. Comput Secur 111:102490.

6. Mukhtar BI, Elsayed MS, Jurcut AD, Azer MA (2023) IoT vulnerabilities and attacks: SILEX malware case study. Symmetry 15(11):1978.

8. Hossain Faruk MJ, Shahriar H, Valero M, Barsha FL, Sobhan S, Khan MA, Whit- man M, Cuzzocrea A, Lo D, Rahman A, Wu F (2021) Malware detection and prevention using artificial intelligence techniques. IEEE Int Conf Big Data (big Data) 2021:5369–5377.

9. Rudd EM, Krisiloff D, Coull S, Olszewski D, Raff E, Holt J (2023) Efficient malware analysis using metric embeddings. Digit Threats Res Pract.

11. Naeem MR, Khan M, Abdullah AM, Noor F, Khan MI, Khan MA, Ullah I, Room S (2022) A malware detection scheme via smart memory forensics for windows devices. Mob Inf Syst 2022:1–16.

14. Dugyala R, Reddy NH, Maheswari VU, Mohammad GB, Alenezi F, Polat K (2022) Analysis of malware detection and signature generation using a novel hybrid approach. Math Probl Eng 2022:1–13

16. Roy KS, Ahmed T, Udas PB, Karim MdE, Majumdar S (2023) MalHyStack: a hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intell Syst Appl 20:200283.

18. Carrier T, Victor P, Tekeoglu A, Lashkari A (2022) Detecting obfuscated malware using memory feature engineering. In: Proceedings of the 8th international conference on information systems security and privacy, pp 177–188

УДК *004.4’2*

ВИДЫ И ОСОБЕННОСТИ АРХИТЕКТУРНЫХ ПАТТЕРНОВ В РАЗРАБОТКЕ E-COM ПРИЛОЖЕНИЙ НА ЯЗЫКЕ SWIFT

М.А. Зотьева

Воронежский государственный университет

Введение

Архитектурные шаблоны являются основой хорошо спроектированного приложения. Они предоставляют собой решение проблемы конструирования в рамках повторяющегося контекста, облегчая разделение задач при разработке крупного программного обеспечения и улучшая удобство сопровождения и возможность повторного использования кода. В следствие чего уменьшаются трудозатраты на разработку сложного программного продукта, поскольку шаблонность действий облегчает коммуникацию между разработчиками, позволяет ссылаться на известные конструкции, снижает количество ошибок.

Тремя наиболее популярными архитектурными шаблонами являются Модель-Представление-Контроллер (MVC), Модель-Представление-Презентатор (MVP) и Модель-Представление-ViewModel (MVVM). Каждый из них имеет свои преимущества и предлагает решения для различных проектов и требований. Понимание этих шаблонов помогает разработчикам более эффективно проектировать программное обеспечение, создавать более качественные решения и гарантировать, что их проекты могут масштабироваться и адаптироваться к меняющимся требованиям.

В данной работе будут рассмотрены все три шаблона.

1. Виды и особенности архитектур

Шаблон MVC(Model-View-Controller) состоит из 3-х элементов:

1. Model (модель) – отвечает за данные и логику приложения. То есть отвечает за то “что” делает приложение.
2. View (представление) – элементы пользовательского интерфейса, которые используются контроллером для его работы. То есть отвечает за визуальную составляющую программного продукта.
3. Controller (контроллер) – управляющая логика, осуществляет взаимосвязь. То есть отвечает за то, как показать что-то на экране.

На рисунке 1 представлена схема архитектуры MVC.

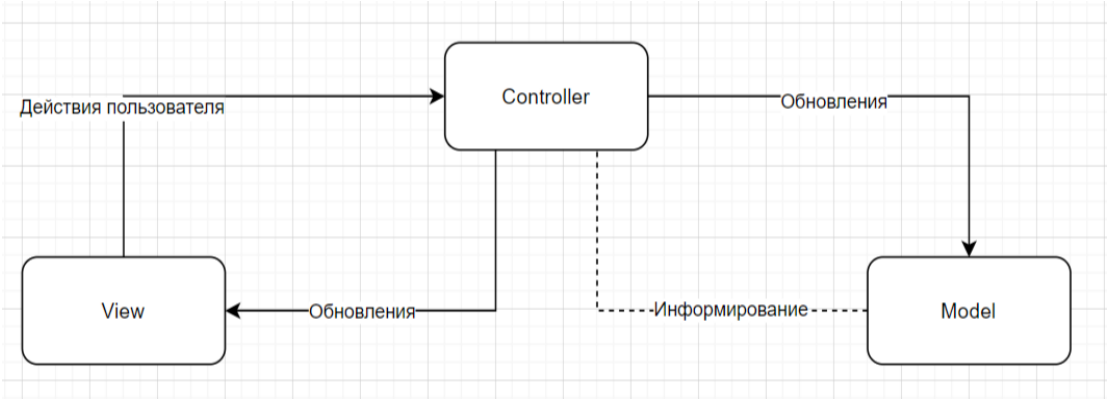


Рис. 1. Схема устройства архитектуры MVC.

Данная архитектура является классической. К её плюсам можно отнести:

1. Скорость разработки.
2. Наглядность.
3. Разделение модели и представления в разные части.

Среди недостатков выделяют, ту особенность, что контроллер и представление сильно связаны друг с другом, а это может приводить к проблеме, которая носит название «Massive ViewController» сильное разрастание Controller, что приводит к невозможности переиспользования и масштабируемости [1].

Данный вариант подходит для проектов, где нужно управлять набором данным, например, приложения для новостей, заметок, записей на мероприятия и т.д.

Шаблон MVP(Model-View-Presenter) также состоит из 3-х элементов:

1. Model(модель) – отвечает за данные и логику приложения.
2. View+ViewController (представление и контроллер) – отвечает за визуальную составляющую программного продукта.
3. Presenter (ведущий) – осуществляет координацию частей приложения.

На рисунке 2 представлена схема архитектуры MVP.

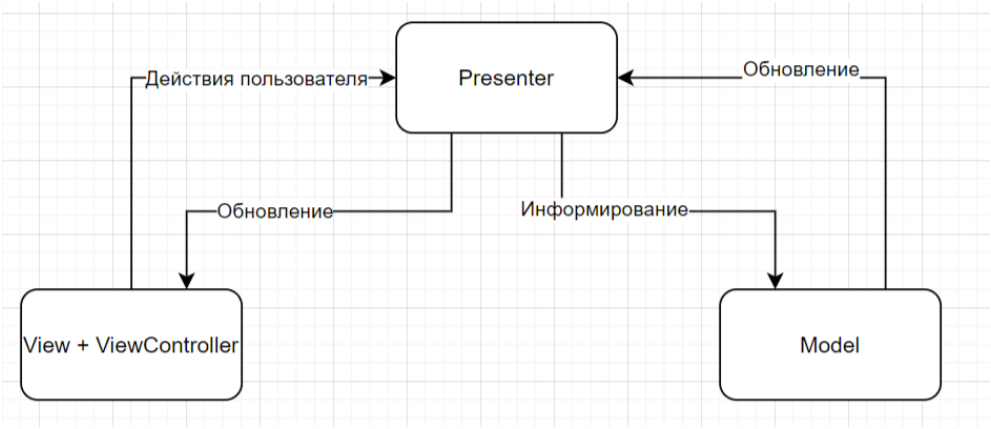


Рис 2. Схема устройства архитектуры MVP.

Данный шаблон отличается от MVC тем, что роль контроллера заменена ведущим, который выступает в качестве посредника между моделью и представлением, поэтому контроллер и представление могут быть также тесно связаны, но это не будет приводить к

проблеме, указанной в предыдущем пункте, так как логика по обновлению модели и представления вынесена в отдельный компонент [5].

Однако у такого подхода есть существенные минусы. К ним можно отнести необходимость в большом количестве однотипного кода, также ведущий не является компонентом повторного использования, то есть если у нас будет одна форма с тем же представлением и логикой, что и другая, мы не сможем повторно использовать тот же presenter [3].

Данный шаблон подходит для приложений с интенсивным взаимодействием пользователя, например, онлайн-чат, где пользователи активно общаются и взаимодействуют друг с другом.

Шаблон MVVM(Model-View-ViewModel) состоит из 3-х элементов:

1. Model(модель) – отвечает за данные.
2. View+ViewController (представление и контроллер) – отвечает за визуальную составляющую программного продукта.
3. View Model – отвечает за взаимодействие Model и View.

Такая архитектура направлена на разделение логики приложения, отображения и взаимодействия с данными. Взаимодействие Model и ViewModel происходит за счёт связывания данных и пользовательского интерфейса.[1]

На рисунке 3 представлена схема архитектуры MVC.

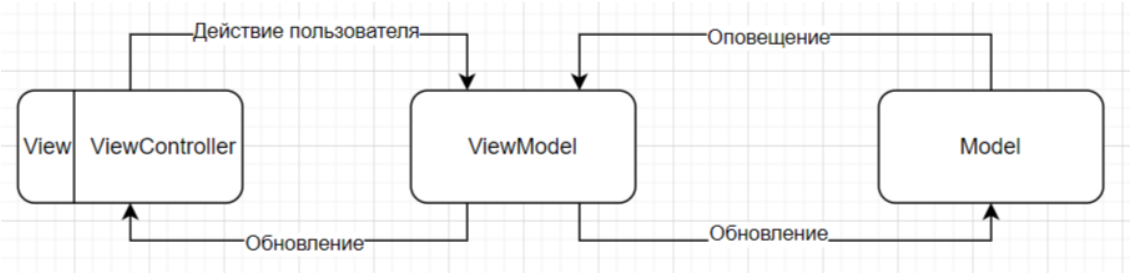


Рис 3. Схема устройства архитектуры MVVM.

Реализации паттернов MVP и MVVM внешне схожи, однако в MVVM связывание View и ViewModel осуществляется автоматически, в то время как в MVP для этих целей необходимо написать некоторый код. У MVC в отличие от этих паттернов возможностей по управлению View чуть больше.

Данный тип архитектуры подходит для приложений, которые разрабатываются для работы на различных платформах или используют большой набор данных и интерактивных элементов.

В конечном итоге, выбор архитектуры, безусловно, зависит от конкретных требований и характеристик проекта. Нет универсального решения, и каждый паттерн имеет свои сильные и слабые стороны. Зачастую придерживаться только одного паттерна – не всегда лучший выбор, так как каждый проект уникален, и его требования могут меняться в процессе разработки. В таких случаях чистое применение только одного паттерна может привести к недостаточной гибкости или даже невозможности удовлетворить требования проекта.

Заключение

В работе был проведен обзор современных стилей архитектур мобильных приложений. Каждый из трёх рассмотренных шаблонов имеет свои преимущества и недостатки, поэтому разработка приложений требует понимания архитектурных паттернов для эффективного проектирования и быстрой работы приложения.

Литература

1. К вопросу выбора оптимального языка программирования для разработки мобильного приложения / Мясоедова В. А – E-Scio, 2022. – 6 с. – URL: https://cyberleninka.ru/article/n/k-voprosu-vybora-optimalnogo-yazyka-programmirovaniya-dlya-razrabotki-mobilnogo-prilozheniya (Дата обращения: 11.02.2024).
2. Swift - язык, который изменит мир программирования / Фролов В.В – Актуальные проблемы авиации и космонавтики, 2020. – 7 с. – URL: https://cyberleninka.ru/article/n/swift-yazyk-kotoryy-izmenit-mir-programmirovaniya/viewer (Дата обращения: 01.03.2024).
3. Сравнение архитектур мобильных приложений на IOS / Романков С. В –Проблемы науки, 2022. – 5 с. – URL: https://cyberleninka.ru/article/n/sravnenie-arhitektur-mobilnyh-prilozheniy-na-ios-platforme (Дата обращения: 08.03.2024).
4. Википедия. – URL: https://ru.wikipedia.org/wiki/ (Дата обращения 13.03.2024).
5. Платформа для социальной журналистики. – URL: https://medium.com/@dev.omartarek/mvp-vs-mvvm-in-ios-using-swift-337884d4fc6f (Дата обращения: 10.01.2024).
6. Усов В.А. Swift. Разработка приложений на основе фреймворка UIKit / Усов В.А. – Перо, 2020. – 490 с.

УДК 004.021

МОДИФИКАЦИЯ КЛАССИЧЕСКОГО АЛГОРИТМА ВРЕМЕННОГО АНАЛИЗА НА ОСНОВЕ ПАРАМЕТРИЧЕСКОГО МЕТОДА ДЕФАЗИФИКАЦИИ

Ю. С. Зырянова

Воронежский государственный университет

Введение

Решение задач, связанных с управлением проектами, остается актуальным и важным для многих организаций. В современном бизнесе с каждым годом проекты становятся все более сложными и масштабными. В таких условиях эффективное управление проектами является ключевой составляющей для достижения поставленных целей, оптимизации использования ресурсов и снижения рисков. Актуальной становится стратегия развития методов, включающая разработку индивидуальных подходов к решению задач управления проектами. Эти подходы должны учитывать специфику деятельности и доступную информацию в организации. Задача календарного планирования играет ключевую роль в управлении проектами, так как именно от качества ее решения в значительной мере зависит успешное завершение проекта. Важными аспектами календарного планирования являются определение временных рамок для выполнения задач, выделение критических путей, нахождение резервов времени, а также определение времени, необходимого для реализации проекта. Цель статьи заключается в представлении модифицированного алгоритма временного анализа проекта в условиях, когда информация о продолжительности работ задана приближенно в форме нечетких чисел. Модификация заключается в использовании параметрического метода дефазификации для перехода от нечеткой задачи к классической.

1. Модели представления приближенной информации

1.1. Нечеткое число и его характеристики, типы нечетких чисел

Для описания объектов и явлений в условиях неопределенности широко используются понятия нечеткой переменной и нечеткой величины.

Нечеткая переменная задается тройкой (γ, A, U), где γ – название переменной, A – нечеткое подмножество универсального множества U (область определения α) с функцией принадлежности $\mu_A(x)$.

Нечеткая величина – это нечеткая переменная, определенная на множестве действительных чисел R.

При построении нечетких моделей используются два основных вида нечетких величин: нечеткий интервал и нечеткое число, которое в общем случае можно рассматривать как частный случай нечеткого интервала. Нечеткие числа и интервалы имеют ряд важных характеристик, которые используются для их сравнения, для оценки уровня неопределенности, для перехода от нечеткой информации к интервальной или обычной числовой. На рис. 1 изображено трапециевидное нечеткое число с функцией принадлежности $\mu_A(x)$ и его

основные характеристики. Заметим, что любое нечеткое число можно представить через совокупность его α-срезов, что позволяет перейти от нечеткой информации к интервальной.

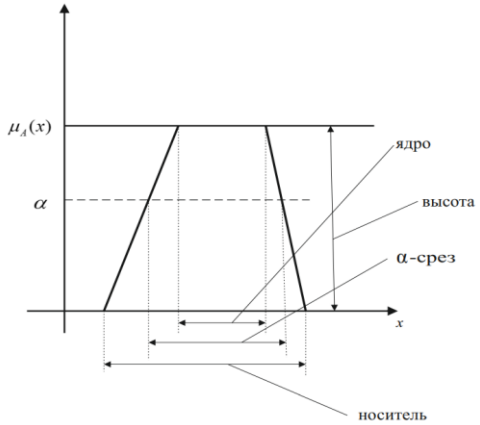


Рис. 1. Характеристики нечеткого числа

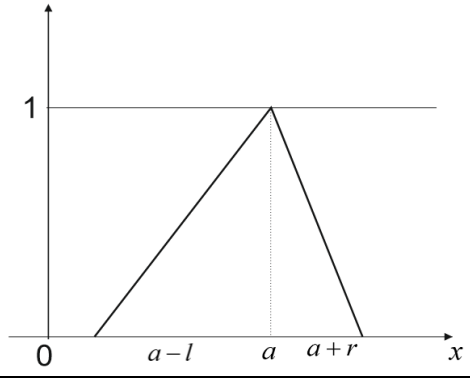
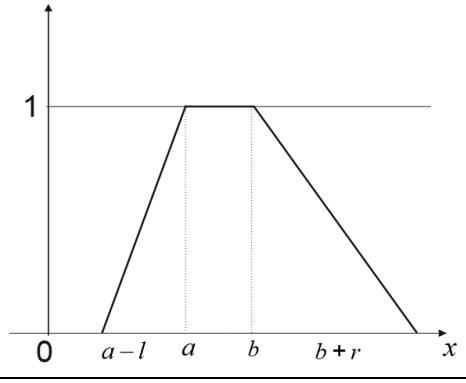
1.2 Сравнение нечетких чисел

В данной статье будем рассматривать нечеткие числа с прямоугольными функциями принадлежности (табл. 1).

Одной из проблем использования нечетких чисел является их сравнение. Для решения этой проблемы предложено значительное количество методов. Распространёнными являются методы, которые базируются на принципе дефазификации. Эти методы предполагают

Таблица 1

Простейшие типы нечетких чисел

l -левый коэффициент неопределенности r -правый коэффициент неопределенности a,b -модальные значения	Треугольное нечеткое число	Трапециевидное нечеткое число
		
Функция принадлежности	$\mu(x)=\begin{cases}1-\dfrac{a-x}{l}, если\ a-l\le x\le a,\\ 1-\dfrac{x-a}{r}, если\ a\le x\le a+r,\\ 0, иначе\end{cases}$	$\mu(x)=\begin{cases}1-\dfrac{a-x}{l}, если\ a-l\le x\le a,\\ 1, a\le x\le b\\ 1-\dfrac{b-x}{r}, если\ b\le x\le b+r,\\ 0, иначе\end{cases}$
Обозначение	$A=(a,l,r)$	$T=(a,b,l,r)$
α - срез	$A_\alpha=\left[a-l(1-\alpha),a+r(1-\alpha)\right]$	$T_\alpha=[a-(1-\alpha)l,b+(1-\alpha)r]$

преобразование нечеткого числа, которое имеет неопределенность, в скалярную величину путем его оценки и использование этой оценки для сравнения и упорядочения нечетких чисел в соответствии с их значимостью или приоритетностью [1]. Применение методов ранжирования, основанных на дефазификации, в контексте управления проектами, является очень полезным инструментом, так как способствует более точному и объективному анализу, а также помогает эффективно обрабатывать нечеткую информацию.

Для модификации классического алгоритма временного анализа проекта был выбран параметрический метод дефазификации, разработанный Ягером и Филевым [2]. За счет настройки параметров этот метод обеспечивает гибкость и настройку на информационную среду конкретного пользователя. Основывая свою работу на преобразовании нечеткого подмножества в связанное распределение вероятностей [3], Ягер и Филев вывели обобщенную формулу для класса оценочных функций и включили в данный класс две дополнительные функции – возрастающую и убывающую. В табл. 2 представлены оценочные функции для треугольного и трапециевидного нечетких чисел, на основе которых становится возможным провести их сравнение [4].

2. Модификация алгоритма определения временных параметров сетевого графика

На основе классического алгоритма определения временных параметров сетевого графика [5] был разработан модифицированный алгоритм временного анализа, основанный на

Таблица 2

Оценки $Val(F)$

	а) $f(\alpha)=\alpha^q$, $q\geq 0$ – строго возрастающая функция $f_{\uparrow}$	б) $f(\alpha)=(1-\alpha)^q$, $q\geq 0$ – строго убывающая функция $f_{\downarrow}$
△	$Val_{f_{\uparrow}}(F)=\nu\cdot\left(a-\frac{l}{q+2}\right)+(1-\nu)\cdot\left(b+\frac{r}{q+2}\right)$	$Val_{f_{\downarrow}}(F)=\nu\cdot\left(a-\frac{q+1}{q+2}\cdot l\right)+(1-\nu)\cdot\left(b+\frac{q+1}{q+2}\cdot r\right)$
△	$Val_{f_{\uparrow}}(F)=\nu\cdot\left(a-\frac{l}{q+2}\right)+(1-\nu)\cdot\left(a+\frac{r}{q+2}\right)$	$Val_{f_{\downarrow}}(F)=\nu\cdot\left(a-\frac{q+1}{q+2}\cdot l\right)+(1-\nu)\cdot\left(a+\frac{q+1}{q+2}\cdot r\right)$

принципе дефазификации и включающий в себя следующие этапы:

Шаг 1. Сформировать множества *fuzzyNumErl*, *fuzzyNumLst* — массивы нечетких треугольных чисел вида (a,l,r), представленные в табл. 3.

Шаг 2. Выбрать параметры метода сравнения ν и q, функцию f.

Шаг 3. Получить правильную нумерацию вершин сетевого графика, при этом исходное событие сети получает номер 0, а завершающее – номер n.

Таблица 3

Вид множеств fuzzyNumErl, fuzzyNumLst

0	1	…	n
(a_0,l_0,r_0)	(a_1,l_1,r_1)	…	(a_n,l_n,r_n)

Шаг 4. (*Определение ранних времен наступления событий*).

1) Положить $t_{erl}(0)=t_{erl}(1)=...=t_{erl}(n)$. Двигаясь по пронумерованной сети в порядке возрастания вершин, для каждой вершины j определить

$$t_{erl}(j) = \max_{i \in \Gamma^{-1}(j)} \left\{ Val_f(fuzzyNumErl(i) + t_{ij}) \right\},$$

где $\Gamma^{-1}(j) = \{i : i \to j\}$ — множество вершин, из которых дуги ведут в вершину j, а нечеткие треугольные числа складываются по формуле [6]:

$$(a, l_a, r_a) + (b, l_b, r_b) = (a + b, l_a + l_b, r_a + r_b)$$

2) Поместить в массив $fuzzyNumErl(j)$ соответствующее найденному максимуму $FuzzyNumErl(i) + t_{ij}$.

Шаг 5. (*Определение критического пути*).

Положить $T_{\max} = t_{erl}(n)$, где n – завершающее событие сети. Для определения критического пути выполнить следующие действия: из всех дуг, входящих в завершающее событие n, выделить те дуги (i, j), которые удовлетворяют условию $t_{erl}(j) - t_{erl}(i) = Val_f(t_{ij})$.

Затем рассматриваются те вершины, из которых выходят выделенные дуги, и снова из входящих в них дуг выделяются те, которые удовлетворяют тому же условию. Процесс продолжается до тех пор, пока не будет достигнуто исходное событие сети. Путь из исходного события в завершающее событие, составленный из выделенных дуг, является критическим.

Шаг 6. (*Определение поздних времен наступления событий*).

1) Положить $t_{lst}(n) = t_{erl}(n) = T_{\max}$. Двигаясь по сети от завершающего события в порядке убывания номеров вершин, определить для каждого i-го события

$$t_{lst}(i) = \min_{j \in \Gamma(i)} \left\{ Val_f(fuzzyNumLst(j) - t_{ij}) \right\},$$

где $\Gamma(i) = \{j : i \to j\}$ — множество вершин, в которые ведут дуги из i, а треугольные нечеткие числа вычитаются по формуле [6]:

$$(a, l_a, r_a)_{LR} - (b, l_b, r_b)_{LR} = (a - b, l_a + r_b, r_a + l_b)_{LR}$$

2) Поместить в массив $fuzzyNumLst(i)$ соответствующее найденному максимуму $FuzzyNumLst(j) - t_{ij}$.

3. Иллюстративный пример

Рассмотрим граф (рис. 2), в котором продолжительности работ заданы нечеткими треугольными числами вида (a, l, r), то есть работа из вершины 0 в вершину 1 задана как «приблизительно 3», из вершины 1 в вершину 3 «приблизительно 5» и т. д., а исходная информация о продолжительностях работ проекта задана в табл. 4.

Таблица 4

Продолжительность работ в форме треугольных нечетких чисел

(i, j)	(0,1)	(0,2)	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)
t_{ij}	(3,1,2)	(4,1,2)	(2,1,2)	(5,1,2)	(3,1,2)	(6,1,2)	(5,1,2)	(2,1,2)

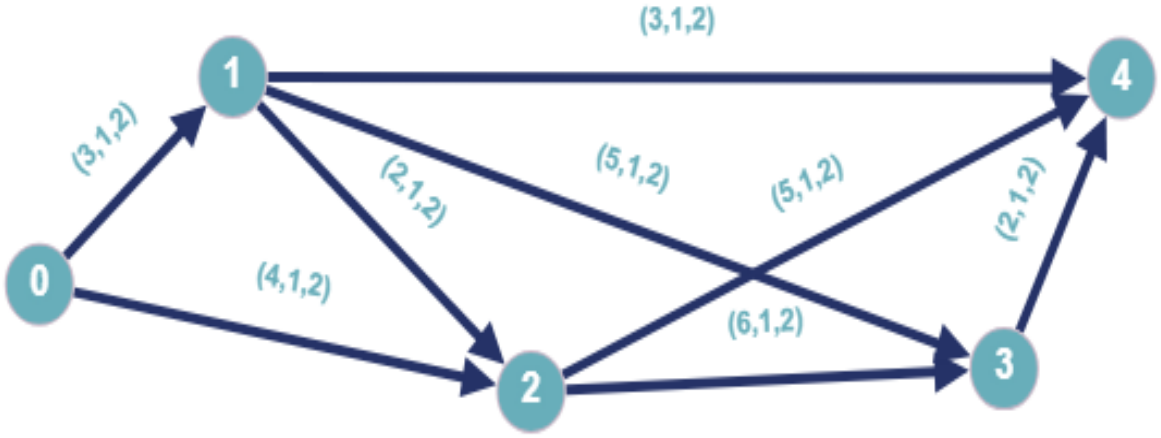


Рис. 2. Сетевой график

В табл. 5 показаны результаты работы алгоритма для расчета критического времени

Таблица 5

Работа алгоритма нахождения времени реализации проекта по ранним временам

j	$t_p(j)$	$FuzzyNumErl(i)+t_{ij}$	$t_{erl}(j)$	$FuzzyNumErl$
0	0	(0,0,0)	0	0 | 1 | 2 | 3 | 4 (0,0,0) | | | |
1	3	(3,1,2)	3.25	0 | 1 | 2 | 3 | 4 (0,0,0) | (3,1,2) | | |
2	5	(4,1,2); (5,2,4)	5.5	0 | 1 | 2 | 3 | 4 (0,0,0) | (3,1,2) | (5,2,4) | |
3	11	(8,3,4); (11,3,6)	11.75	0 | 1 | 2 | 3 | 4 (0,0,0) | (3,1,2) | (5,2,4) | (11,3,6) |
4	13	(6,2,4); (10,3,6); (13,4,8)	14	0 | 1 | 2 | 3 | 4 (0,0,0) | (3,1,2) | (5,2,4) | (11,3,6) | (13,4,8)

реализации проекта по ранним временам для сетевого графика на рис. 2, где t_p – ранние времена наступления событий, вычисленные с модальными значениями продолжительности работ, а t_{erl} – дефазифицированные значения в условиях неопределенности. Параметры метода были выбраны следующим образом: $\nu = \frac{1}{2}$, $q = 0$, $f_{\uparrow}$.

В результате получим, что $T_{\max} = t_{erl}(4) = 14$ — критическое время реализации проекта.

Рассмотрим вычисления подробнее: $t_{erl}(0)=0;$

$$t_{erl}(1)=Val_{f_\uparrow}(\,fuzzyNumErl(0)+t_{12})=Val_{f_\uparrow}\left((0,0,0)+(3,1,2)\right)=Val_{f_\uparrow}\left((3,1,2)\right)=$$

$$=\frac{1}{2}\cdot\left(3-\frac{1}{2}\right)+\frac{1}{2}\cdot\left(3+\frac{2}{2}\right)=3.25;$$

$$t_{erl}(2)=\max\left\{Val_{f_\uparrow}(\,fuzzyNumErl(0)+t_{02}),Val_{f_\uparrow}(\,fuzzyNumErl(1)+t_{12})\right\}=$$

$$=\max\left\{Val_{f_\uparrow}\left((0,0,0)+(4,1,2)\right),Val_{f_\uparrow}\left((3,1,2)+(2,1,2)\right)\right\}=$$

$$=\max\left\{Val_{f_\uparrow}\left((4,1,2)\right),Val_{f_\uparrow}\left((5,2,4)\right)\right\}=\max\left\{4.25,5.5\right\}=5.5;$$

$$t_{erl}(3)=\max\left\{Val_{f_\uparrow}(\,fuzzyNumErl(1)+t_{13}),Val_{f_\uparrow}(\,fuzzyNumErl(2)+t_{23})\right\}=$$

$$=\max\left\{Val_{f_\uparrow}\left((3,1,2)+(5,1,2)\right),Val_{f_\uparrow}\left((5,2,4)+(6,1,2)\right)\right\}=$$

$$=\max\left\{Val_{f_\uparrow}\left((8,2,4)\right),Val_{f_\uparrow}\left((11,3,6)\right)\right\}=\max\left\{8.5,11.75\right\}=11.75;$$

$$t_{erl}(4)=\max\left\{\begin{matrix}Val_{f_\uparrow}(\,fuzzyNumErl(1)+t_{14}),Val_{f_\uparrow}(\,fuzzyNumErl(2)+t_{24}),\\ Val_{f_\uparrow}(\,fuzzyNumErl(3)+t_{34})\end{matrix}\right\}=$$

$$=\max\left\{Val_{f_\uparrow}\left((3,1,2)+(3,1,2)\right),Val_{f_\uparrow}\left((5,2,4)+(5,1,2)\right),Val_{f_\uparrow}\left((11,3,6)+(2,1,2)\right)\right\}=$$

$$=\max\left\{Val_{f_\uparrow}\left((6,2,4)\right),Val_{f_\uparrow}\left((10,3,6)\right),Val_{f_\uparrow}\left((13,4,8)\right)\right\}=\max\left\{6.5,10.75,14\right\}=14.$$

Из полученных расчетов видно, что при выбранных параметрах дефазифицированное значение в условиях неопределенности отличается от точного на единицу. Для исследования критического времени реализации проекта $T_{\max}$ при изменении весового коэффициента ν оставим остальные параметры неизменными и рассмотрим различные значения параметра ν: 0.2, 0.5, 0.8, 1. В процессе анализа результатов вычислений был построен график в Excel, представленный на рис. 3. Из графика четко видно, что точное решение достигается при весе $\nu\in(0.6,0.7)$.

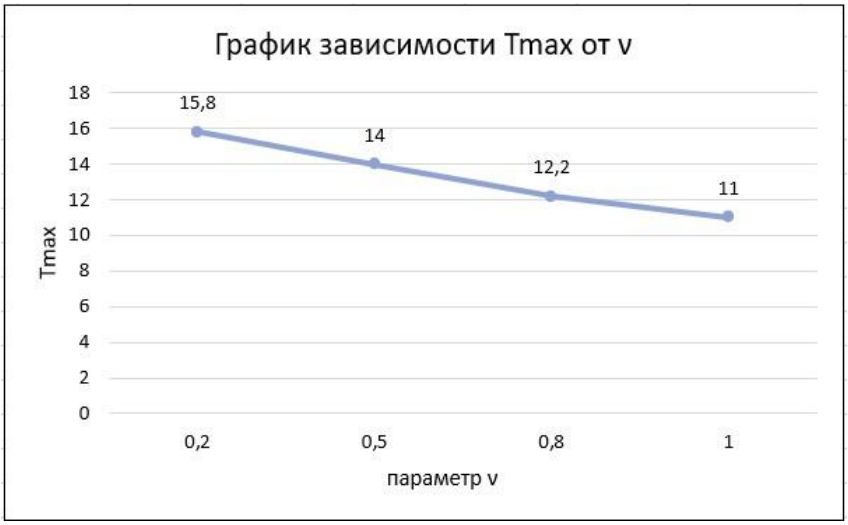


Рис. 3. График зависимости времени реализации проекта от коэффициента ν

Таким образом, модифицированный алгоритм временного анализа на основе параметрического метода дефазификации позволяет приблизиться к точным значениям временных характеристик проекта в условиях неопределенности при правильной для конкретной задачи настройке параметров. Однако следует учитывать, что при использовании данного метода остается небольшая погрешность, которая может быть индивидуально скорректирована в зависимости от определенных условий и требований проекта. Также важно отметить, что настройка параметров модифицированного алгоритма зависит от уровня неопределенности в проекте. Коэффициенты неопределенности могут влиять на выбор оптимальных весовых коэффициентов и тем самым влиять на точность предсказания временных параметров проекта. При наличии высокой степени неопределенности могут потребоваться более гибкие подходы к настройке параметров для достижения наиболее точных результатов при оценке временных характеристик.

Заключение

В рамках проведенного исследования была разработана модификация классического алгоритма календарного планирования, которая направлена на повышение эффективности его применения. Изменяя параметры ν и q, можно выбирать оптимальные способы упорядочения задач в зависимости от их целей и требований конкретной прикладной задачи. Дальнейшие планы включают в себя детальное исследование предложенного алгоритма с целью выявить, каким образом назначить параметры и какую функцию выбрать – убывающую или возрастающую в зависимости от величины коэффициентов неопределенности, а также выяснить влияние данных факторов на итоговый результат.

Литература

1. Заде Л.А. Роль мягких вычислений и нечеткой логики в понимании, конструировании и развитии информационных/интеллектуальных систем / Л.Ф. Заде // Новости искусственного интеллекта. –2001. - №2-3.

2. Yager R. R. On ranking fuzzy numbers using valuations / R. R. Yager, D. Filev // International Journal of Intelligent Systems, 1999. – №14. – P. 1249-1268.

3. Ягер Р.Р. Результат устранения нечеткости и выбор, основанный на нечетком множестве / Р.Р. Ягер, Д.П. Филев // Нечеткие множества и системы.–1993. - №55. – С. 255-272.

4. Леденев М.Ю., Шашкин А.И. Параметрические функции для сравнения нечетких чисел / М.Ю. Леденев, А.И. Шашкин // Актуальные проблемы прикладной математики, информатики и механики (г. Воронеж, 12-14 декабря 2022г.). – С. 1402-1406.

5. Леденев М.Ю., Сергиенко М.А. Алгоритм расчета нечетких и интервальных оценок временных параметров сетевой модели проекта [Электронный ресурс] / Леденев М.Ю., Сергиенко М.А.// Международный научно-исследовательский журнал – 2021. – № 6. – Режим доступа: https://research-journal.org/wp-content/uploads/2021/06/6-108-1.pdf#page=118. – (Дата обращения: 18.04.2023).

6. Пегат А. Нечеткое моделирование и управление / А. Пегат. - М. : Бином. Лаборатория знаний, 2009. – 798 с.

7. Леденева Т.М. Обработка нечеткой информации : учебное пособие / Т.М. Леденева. – Воронеж : Воронежский государственный университет, 2006. – 233 с.

УДК 681.5

ОЦЕНКА ЭФФЕКТИВНОСТИ РЕСУРСНОГО ОБЕСПЕЧЕНИЯ ПРОЕКТА НА ОСНОВЕ АППАРАТА СЕТЕЙ ПЕТРИ

А. В. Исаев

Воронежский государственный университет

Введение

В настоящее время, для успешного выполнения какого-либо проекта, необходимо календарное планирование. Оно включает в себя следующий комплекс действий: формирование последовательности выполнения работ, определение их продолжительности и потребностей в ресурсах, анализ результатов и оценку выполнения проекта в директивные сроки. Данное планирование осуществляется на этапе разработки проекта и является эффективным инструментом для управления им [1, 2].

Основой для исполнения проекта служит модель календарного планирования. В настоящем исследовании рассматривается строительный проект, представленный сетевым графом. Часть ресурсов проекта имеет недетерминированное распределение, для другой части - распределение ресурсов между проектами осуществляется из единого центра. Требуется провести анализ и оценить вероятность исполнения проекта в директивные сроки. Исследование его временных характеристик было решено проводить с использованием метода критического пути, основной задачей которого является нахождение пути с наибольшей продолжительностью из всех полных путей. Для исследования сетевого графа предложен механизм на основе аппарата раскрашенных сетей Петри, который позволит сымитировать выполнение работ календарного плана и оценить по результатам моделирования вероятность успешного выполнения проекта в директивные сроки.

Цель данного исследования заключается в создании алгоритмического и программного обеспечения для оценки эффективности распределения ресурсов в проекте, основанного на принципах сетевого планирования и аппарате сетей Петри.

Достижение этой цели требовало решения ряда задач, среди которых особое значение имеют:

1. разработка алгоритма оценки эффективности ресурсного обеспечения проекта, базирующегося на методах сетевого планирования и аппарате сетей Петри;
2. создание программного обеспечения для разработанного алгоритма.

1. Алгоритмическое и программное обеспечение для оценки эффективности ресурсного обеспечения проекта

Предполагается, что ресурсы, выделенные для осуществления проекта, разделены на две группы:

1. возвращаемые в центр после каждого использования;
2. выделяемые персонально на каждую работу в отдельности.

Требуется разработать алгоритм оценки эффективности ресурсного обеспечения проекта, позволяющий вычислить вероятность выполнения проекта в установленные сроки.

Приведем основные шаги выполнения алгоритма:

1. Вводится информация о каждой работе сетевого графа: номер работы, ее начало и конец, продолжительность;

2. Вычисляются временные характеристики проекта и критический путь выполнения работ в соответствии с методом критического пути;

3. Заполняется информация о потребности работ в возвращаемых в единый центр ресурсах. Для каждой работы задаются нижние границы, при которых может начаться работа в каждой определенной позиции $\underline{r_{hi}}$ и оптимальная величина ресурса, требуемая на выполнение данной работы $\overline{r_{hi}}$.

4. Заполняется информация о потребности работ в выделяемых на каждую работу в отдельности изначально ресурсах. Для каждой работы вводятся данные об интервалах по каждому типу ресурсов (r_j^{start}, r_j^{end}), трудоёмкости tr и качестве работ k_i. Считается, что каждый ресурс является случайной величиной, распределенной по равномерному закону с плотностью распределения:

$$F(x) = \begin{cases} \dfrac{1}{r_j^{end} - r_j^{start}}, & x \in \left[r_j^{start},\ r_j^{end}\right]; \\ 0, & x \notin \left[r_j^{start},\ r_j^{end}\right] \end{cases} \qquad (1)$$

5. Для начальной позиции определяются маркеры-переменные (цвета): маркер-флажок срабатывания перехода $D_a(j)$, маркеры начала и окончания текущей работы $S(i)$, $D(i)$, начальный временной маркер τ, маркер реального времени υ. Маркер $S(i)$ задает начало процесса. Он указывает на то, что выполнение работы в позиции началось и срабатывает лишь в случае, если на работу хватает ресурсов из единого центра. После срабатывания перехода активируется маркер-флажок окончания работы. Начальный временной маркер – переменная, представляющая собой время начала выполнения проекта. Маркер реального времени – определяет реальное время, прошедшее с начала выполнения проекта до текущего момента.

6. В качестве обязательных маркеров каждой позиции определяются две функции: время выполнения работы $T\left(r_1, \ldots, r_L\right)$ и качество выполнения работы $K\left(r_1, \ldots, r_L\right)$, которые являются функциями от L случайных ресурсов $r_1, \ldots, r_L$.

Функция времени задается следующей формулой:

$$T\left(r_1, \ldots, r_L\right) = \frac{tr}{f * r_i * k_i} \qquad (2)$$

где tr – трудоемкость;

r_i – выделенные ресурсы для данной работы;

k_i – заданный коэффициент качества для данной работы;

f – коэффициент перевода рабочих дней в календарные с учетом отпусков ($f \approx 0{,}66$).

Значения функции качества $K\left(r_1, \ldots, r_L\right)$ лежат в отрезке $[0;1]$. Функция формируется на основе оценок качества отдельных ресурсов для каждой работы:

$$K_j = \frac{r_j}{r_j^{opt}} \qquad (3)$$

где r_j – заданный ресурс, который имеется для данной работы;

r_j^{opt} – оптимальное значение ресурса, которое нужно для идеального выполнения работы.

Тогда итоговая функция качества определяется как

$$K(r_1, \dots, r_L) = \Pi_{j=1}^{L} K_j \qquad (4)$$

7. Задается общее имеющееся количество ресурсов проекта *Res*, которые возвращаются в центр управления после использования их для выполнения работ.

8. Вводится количество итераций и запускается механизм сетей Петри. Когда условный маркер-флажок активности приходит в определенную позицию, производится запрос к ресурсному центру проекта: если в центре имеется оптимальная величина ресурса r_j^{opt}, то именно оптимальная величина забирается на выполнение работы в данной позиции, в противном случае, если

$$\underline{r_{hi}} \le Res < \overline{r_{hi}} \quad (5)$$

то из центра берется количество ресурса равное *Res*. Работа начинается, то есть активируется маркер $S(i)$ начала работы, для нее генерируются значение функции времени выполнения работы $T(r_1, \dots, r_L)$, и значение $K(r_1, \dots, r_L)$ – качества выполнения работы. Если же ресурсов в центре для начала очередной работы не хватает, происходит ожидание возврата ресурсов какой-либо из работ, которые начались, но не закончились, то есть с активным маркером $S(i)$, но неактивным $D(i)$ и соответствующее увеличение маркера реального времени υ на величину, равную времени, прошедшему с начала момента ожидания. Когда все предшествующие переходу работы оказались выполнены, проводится первый контроль, представляющий собой сравнение маркера реального времени υ и позднего срока окончания $t_{le}(i, j)$ последней из завершившихся работ, предшествующих переходу. Если контроль не пройден, то есть $\upsilon > t_{le}(i, j)$, то происходит переход к следующей итерации механизма сетей Петри, а текущий переход помечается как несработавший. В случае же успешного прохождения контроля, проводится вторая проверка, представляющая собой сравнение реальной оценки качества текущей работы K_i с заданным минимальным порогом качества работ. В случае успешного прохождения обоих проверок, переход срабатывает, то есть активируется маркер флажок $D_a(j)$ и условный маркер-флажок активности перемещается к следующему во времени переходу.

9. С помощью имитационных механизмов сетей Петри оценивается вероятность выполнения проекта в директивные сроки. После последней итерации выводится таблица результатов и вероятность исполнения в директивные сроки определенного проекта с заданным ресурсным обеспечением.

2. Практическая реализация и результаты вычислительного эксперимента

Для осуществления алгоритма было разработано программное обеспечение, которое позволяет автоматизировать его выполнение. Созданное ПО способно не только производить расчеты требуемых параметров и величин, но и осуществлять контроль над ключевыми этапами работы проекта. Приложение предоставляет возможность ввода известных данных непосредственно в программу, а также проведение контроля над рассчитанными параметрами. Полученные результаты алгоритма будут использоваться для оценки конкретного проекта, что делает данное программное обеспечение не только удобным, но и необходимым инструментом в сетевом планировании проектов.

Программная реализация алгоритма выполнена в Microsoft Visual Studio 2019 на языке программирования C# с использованием платформы .NET Framework.

Рассмотрим работу программы на примере проекта строительства одноэтажного здания.

Сетевой граф проекта изображен на рисунке 1.

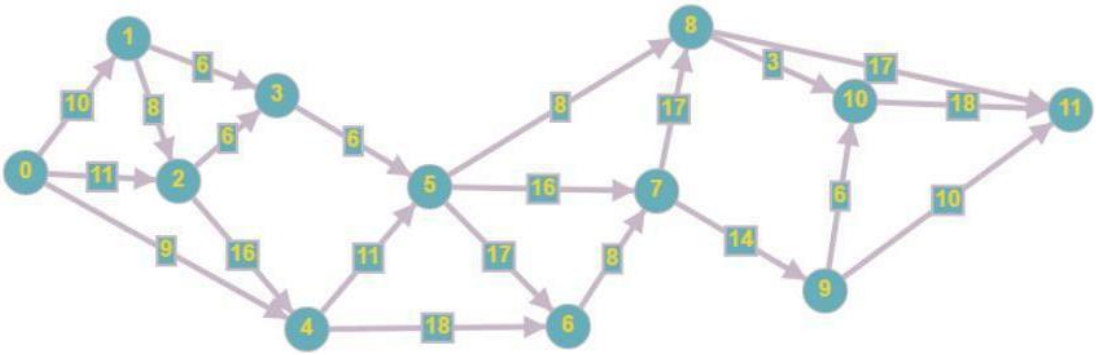


Рис. 1. Сетевой график проекта

После ввода в программу данных о работах проекта, срабатывает алгоритм нахождения критического пути, результат работы которого отображен на рисунке 2.

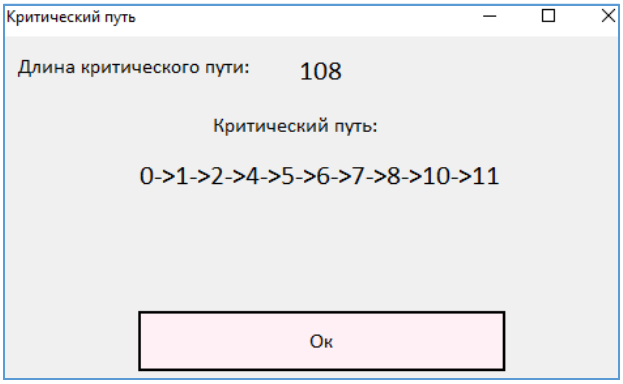


Рис. 2. Фрагмент программного обеспечения. Основная информация о результатах выполнения алгоритма нахождения критического пути

Далее, в соответствии с руководством пользователя, в программе задаются данные о потребности работ в возвращаемых в центр детерминированных ресурсах проекта. В качестве таких ресурсов рассматриваемого проекта выступают человеческие ресурсы, а именно количество разнорабочих, труд которых используется в работах проекта.

Для дальнейшего выполнения алгоритма производится ввод данных об интервалах по каждому типу ресурсов (r_j^{start}, r_j^{end}) , трудоемкости tr и качестве k_i работ.

План проведения имитационного эксперимента включает следующие этапы:

1. исследование влияния выделенных на выполнение всего проекта объемов возвращаемых ресурсов на вероятность исполнения проекта в директивные сроки;
2. исследование влияния выделенных объемов недетерминированных ресурсов для каждой работы на вероятность выполнения проекта в директивные сроки.

Далее, в соответствии с разработанным планом, в программу вводится имеющееся количество возвращаемых в центр проектных ресурсов, задается количество итераций

алгоритма. После выполнения последней итерации, выводится таблица результатов работы алгоритма. Она продемонстрирована на рисунке 3.

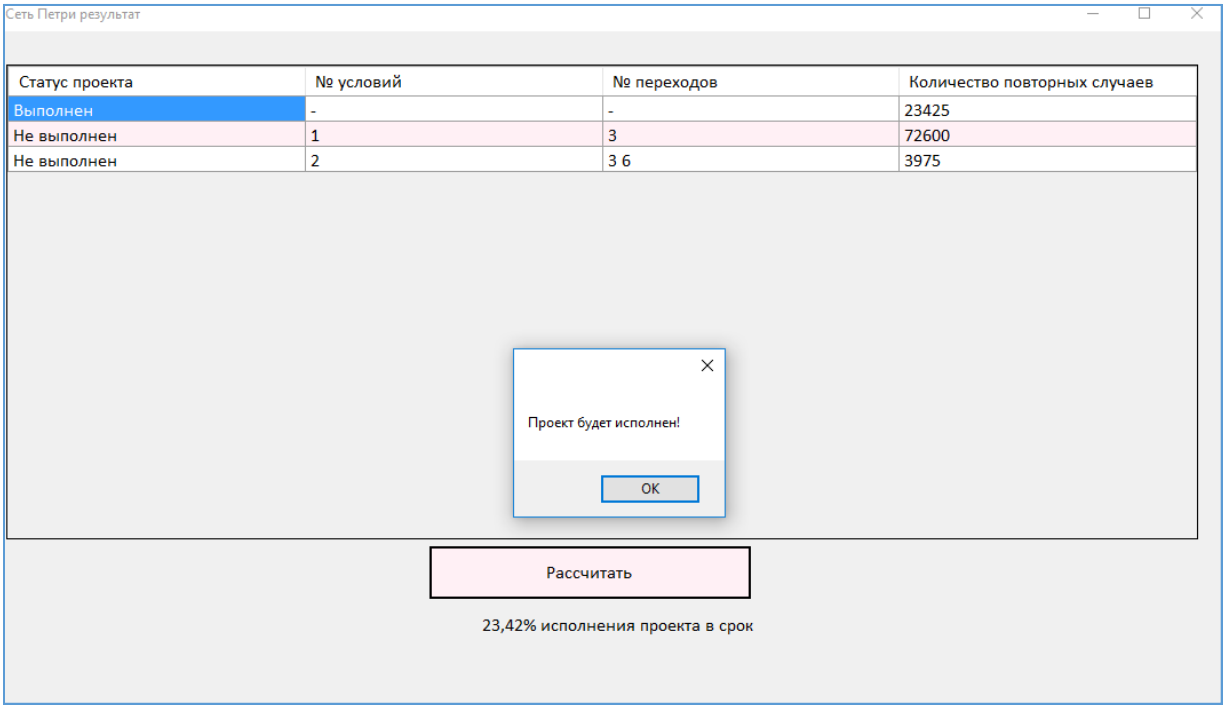


Рис. 3. Фрагмент программного обеспечения. Результат работы алгоритма с количеством разнорабочих проекта, равным 53 чел.

По результатам 100000 проверок вероятность выполнения в директивные сроки будет равна 23,42%. Причиной срыва выполнения проекта в 72600 случаях из 100000 являлось невыполнение временных ограничения и в 3975 случаях – невыполнение требований качества работ. Из полученной информации следует, что при таком ресурсном обеспечении вероятность исполнения проекта в директивные сроки низкая, и необходимо перераспределение ресурсов.

После был проведен эксперимент для выявления влияния повышения количества возвращаемого в центр ресурса, увеличения границ распределения недетерминированных ресурсов на вероятность исполнения проекта в директивные сроки. В рамках этого этого эксперимента количество возвращаемого ресурса в ресурсном центре было увеличено с 53-х до 100 единиц, границы распределения для недетерминированных ресурсов были повышены для каждой работы на 20%. Результат представлен на рисунке 4.

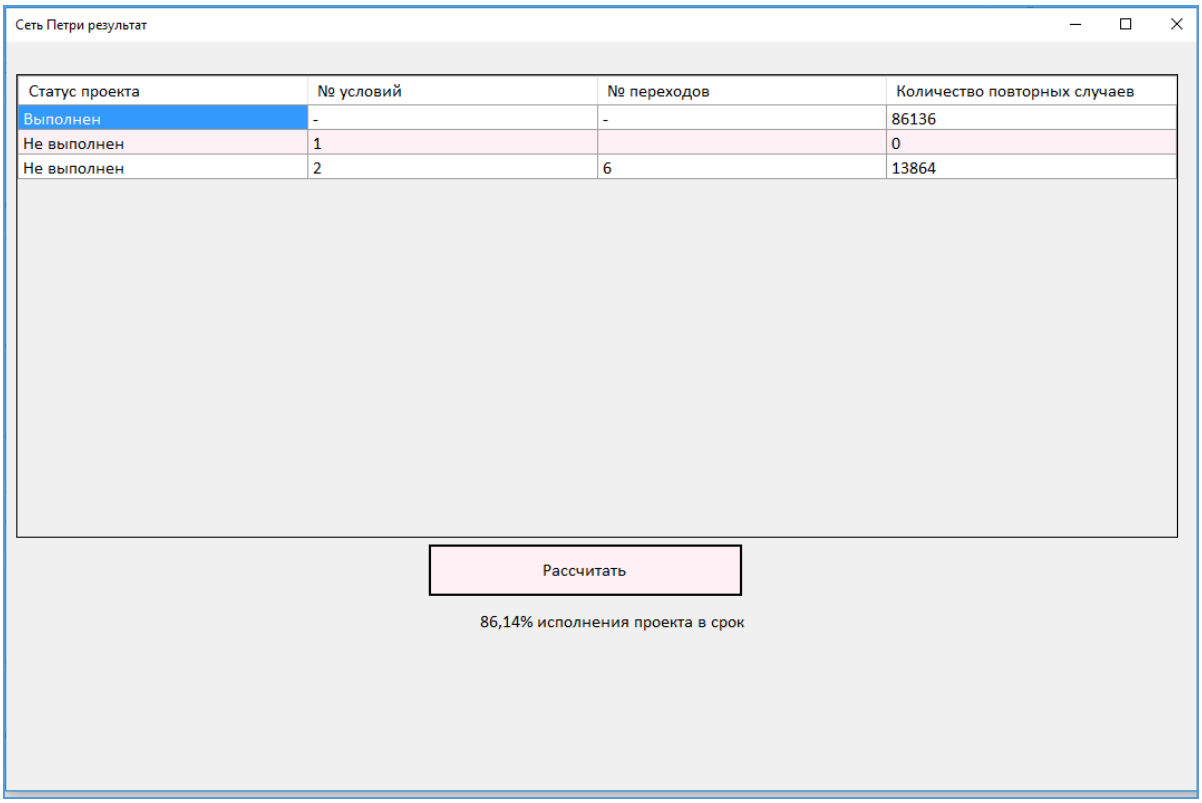


Рис. 4. Фрагмент программного обеспечения. Результат работы алгоритма с внесёнными изменениями.

Видим, что вероятность исполнения проекта в срок при таком распределении ресурсов повысилась до 86,14%. Причём на этот раз, контроль времени выполнения был пройден во всех случаях, то есть заданного количества возвращаемых ресурсов хватило на все параллельно выполняемые работы.

Таким образом, путём перераспределения возвращаемых в единый центр ресурсов, а также изменения границ распределения недетерминированных ресурсов, удалось повысить вероятность исполнения проекта в директивные сроки. Предложенный алгоритм позволяет имитировать работу системы при разном количестве ресурсов, и на основе приведённых расчётов, лицо принимающее решение может выбрать план распределения ресурсов, гарантирующий пороговую вероятность исполнения проекта в директивные сроки.

Заключение

В результате проведённого исследования был разработан алгоритм оценки эффективности ресурсного обеспечения проекта на основе аппарата сетей Петри, а также его программная реализация, позволяющая пользователю вводить информацию о проекте, рассчитывать определённые параметры и характеристики, анализировать проект и оценивать вероятность его исполнения в директивные сроки. Программное обеспечение реализовано на языке программирования C# и выполнено в Microsoft Visual Studio.

Литература

1. Боронина, Л. Н. Основы управления проектами : [учеб. пособие] / Л. Н. Боронина, З. В. Сенук ; М-во образования и науки Рос. Федерации, Урал. федер. ун-т. – Екатеринбург : Изд-во Урал. ун-та, 2015. — 112 с.
2. Питерсон Дж. Теория сетей Петри и моделирование систем: пер. с англ. / Дж. Питерсон – Москва : Мир, 1984. – 264 с.
3. Котов В.Е. Сети Петри. / В.Е. Котов – Москва : Наука, 1984. – 158 с.
4. Васильев, В.В. Сети Петри, параллельные алгоритмы и модели мультипроцессорных систем / В.В. Васильев, В.В. Кузьмук. – Киев: Наук. думка, 1990. – 216 с.
5. Лескин А. А. Сети Петри в моделировании и управлении / А. А. Лескин, П. А. Мальцев, А. М. Спиридонов – Л.: Наука, 1989. – 133 с.
6. Ломазова И. А. Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой / И. А. Ломазова –Москва : Научный Мир, 2004. – 208 с.
7. Кулагин В.П. Философия сетей Петри / В. П. Кулагин, В. Я. Цветков – Москва : Вестник МГТУ МИРЭА, 2014. – с. 18-38. .

УДК 004.932.2

ИССЛЕДОВАНИЕ АЛГОРИТМОВ СЕГМЕНТАЦИИ ДЛЯ ПОСТРОЕНИЯ ПОЛИГОНАЛЬНОЙ МОДЕЛИ ПО СНИМКАМ КТ КОСТЕЙ

В. А. Истомин, Е. В. Трофименко

Воронежский государственный университет

Введение

Задача сегментации — одна из основных задач обработки и анализа изображений, позволяющая разделять изображения на области, в каждой из которых пиксели обладают определенным критерием схожести: например, в одной области будут лежать пиксели, обладающие примерно одинаковым значением яркости и/или цвета.

Сегментация КТ снимков послойно представляет собой процесс, при котором изображение делится на различные регионы или слои, каждый из которых соответствует определенной анатомической структуре или типу ткани. Это особенно важно в медицинской визуализации для выделения определенных структур, таких как кости, органы или опухоли.

В данной статье проводится анализ алгоритмов сегментации, которые можно использовать для обработки данных из DICOM-файлов, в которые записываются результаты КТ-исследования: метод водоразделов, метод GrabCut и метод выращивания регионов; анализируется время работы всех рассмотренных алгоритмов, а также рассматривается возможность внедрения рассмотренных алгоритмов при построении полигональных моделей костей по снимкам компьютерной томографии.

1. Алгоритмы сегментации

1.1. Метод водораздела

В методе водораздела [1,2] изображение рассматривается как некоторая карта местности, где значение яркости пикселя представляют собой значение высот. Благодаря таким расчетам, пиксели с высокой интенсивностью образуют «горы», а с низкой – «впадины». Полученная с такими расчетами карта местности начинает заполняться водой: впадины будут постепенно заполняться, сливаясь с соседними, при этом место слияния будет обозначаться границей водораздела – именно по этим границам осуществляется разбиение изображения на сегменты.

Рассмотрение работы алгоритма в общем случае проводилось на нескольких изображениях. Результаты приведены на рис. 1.

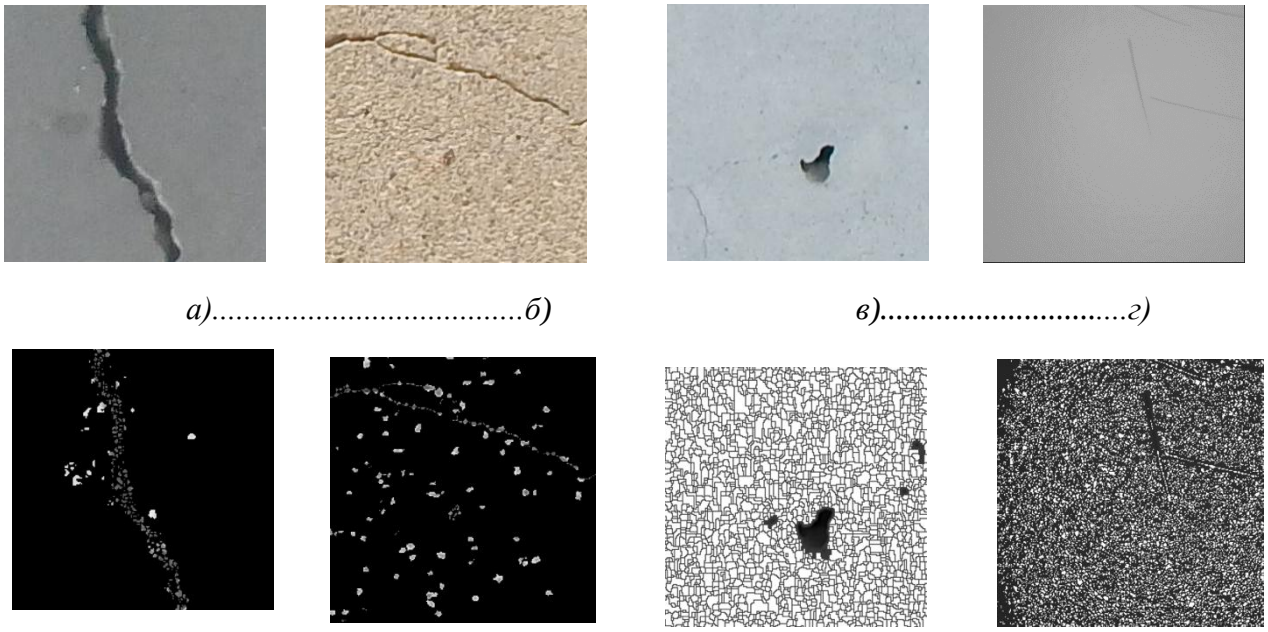
Исходя из полученных результатов можно сделать вывод о том, что метод водораздела частично справился с задачей выделения объектов на тестовых изображениях. Можно заметить, что на рис. 1е) алгоритм выделил несколько микрообъектов наряду с основным объектом (трещиной), несмотря на выраженную текстуру поверхности. Также, на рис. 1 з) метод водораздела не справился с выделением такого объекта, как царапина.

Также стоит добавить, что из-за особенности реализации алгоритма, нанесение на изображение границ мешает отфильтровать пиксели на картинке таким образом, чтобы на результирующем изображении остался только объект.

Рассмотрение работы алгоритма при выделении костей человека со снимка проводилось

для снимка предплечья и кисти человека. Тестовое изображение и результат сегментации приведен на рис. 2.

Исходя из полученных результатов можно сделать вывод о том, что метод водораздела хорошо справился с задачей выделения костной ткани на снимке, однако некоторые фрагменты кости он посчитал отдельными объектами. Тем не менее, точность очень высока.



а).....б) *в).....г)*

д).....е).....ж).....з)

Рис 1. Результаты сегментации методом водораздела. а-г) Изображения до сегментации, д-з) результаты после сегментации.

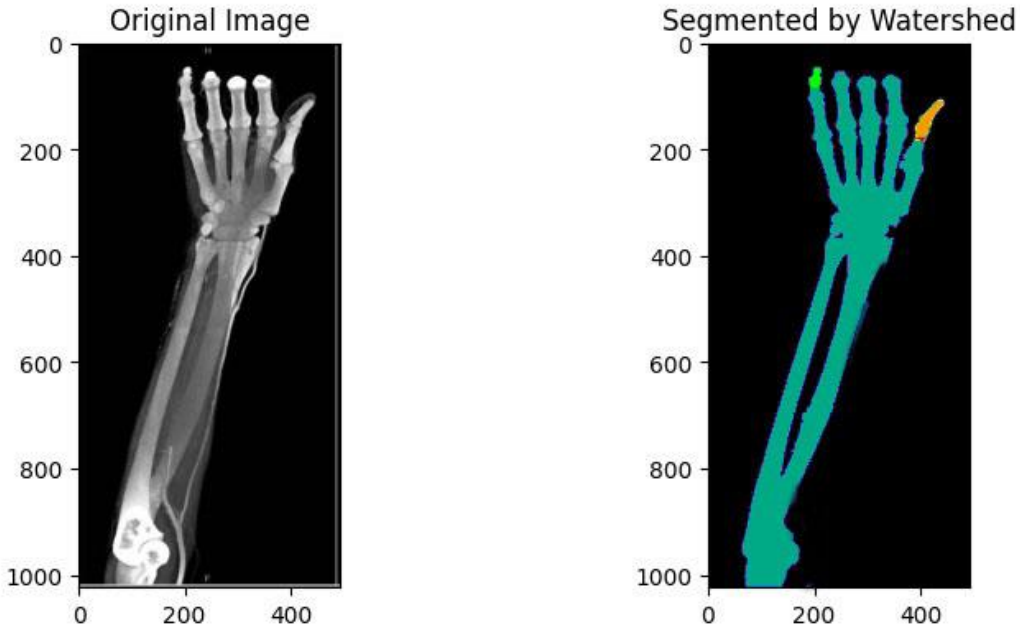


Рис 2. Результат сегментации снимка конечности методом водораздела

1.2. Метод GrabCut

Алгоритм GrabCut [3] — это полуавтоматический метод сегментации изображений. Он позволяет эффективно выделять объекты на изображениях, используя как минимальное взаимодействие пользователя, так и сложные алгоритмы анализа. Его особенность заключается в способности быстро и точно выделять передний план от фона на основе простых вводных данных пользователя и итеративных вычислений. GrabCut обычно используется в графических редакторах и приложениях для автоматической обработки изображений.

Идея алгоритма сегментации GrabCut заключается в автоматическом выделении интересующего объекта на изображении с минимальным вмешательством пользователя. Пользователь лишь указывает прямоугольную область, внутри которой находится объект, который нужно сегментировать от фона. Алгоритм использует эту информацию для инициализации модели и далее автоматически работает для определения границ объекта, используя модели цветового распределения для фона и переднего плана.

Алгоритм использует итеративный процесс, который постепенно уточняет границы между объектом и фоном, минимизируя заданную функцию энергии. Это позволяет достичь высокой точности сегментации.

Для представления цветового распределения как переднего плана, так и фона, GrabCut использует смешанные гауссовские модели. Это позволяет алгоритму адаптироваться к разнообразию цветов внутри обеих областей и более точно выделять границы объекта.

Результат работа алгоритма в общем случае приведен на рис. 3.

Несмотря на то, что алгоритм хорошо справляется с задачей сегментации, качество и точность сегментации сильно зависят от того, насколько корректно пользователь указал начальную область вокруг интересующего объекта. Если прямоугольник инициализации неправильно включает в себя части фона, которые сильно схожи с объектом по цвету или текстуре, это может привести к ошибкам в сегментации. Если фон изображения содержит цвета или текстуры, схожие с цветами объекта переднего плана, GrabCut может некорректно классифицировать эти области. Это видно на рис. 3 д). Также алгоритм может столкнуться с трудностями при наличии шума или сложных градиентов на изображении. Это видно на рис. 3 е). Результаты на рисунках 3 ж) и 3 з) говорят о том, что алгоритм способен точно выделять объекты на изображениях.

Рассмотрение работы алгоритма при выделении костей человека со снимка проводилось для снимка предплечья и кисти человека. Тестовое изображение и результат сегментации приведен на рис. 4.

Исходя из полученных результатов можно сделать вывод о том, что метод GrabCut частично справился с задачей выделения костной ткани на снимке. Часть кости, которая имеет меньшую яркость (интенсивность), алгоритм принял за мягкие ткани, а сосуды, напротив, отнес к костной ткани.

а) *б)* *в)* *г)*

д) *е)* *ж)* *з)*

Рис 3. Результаты сегментации GrabCut. а-г) Изображения до сегментации,д-з) результаты после сегментации.

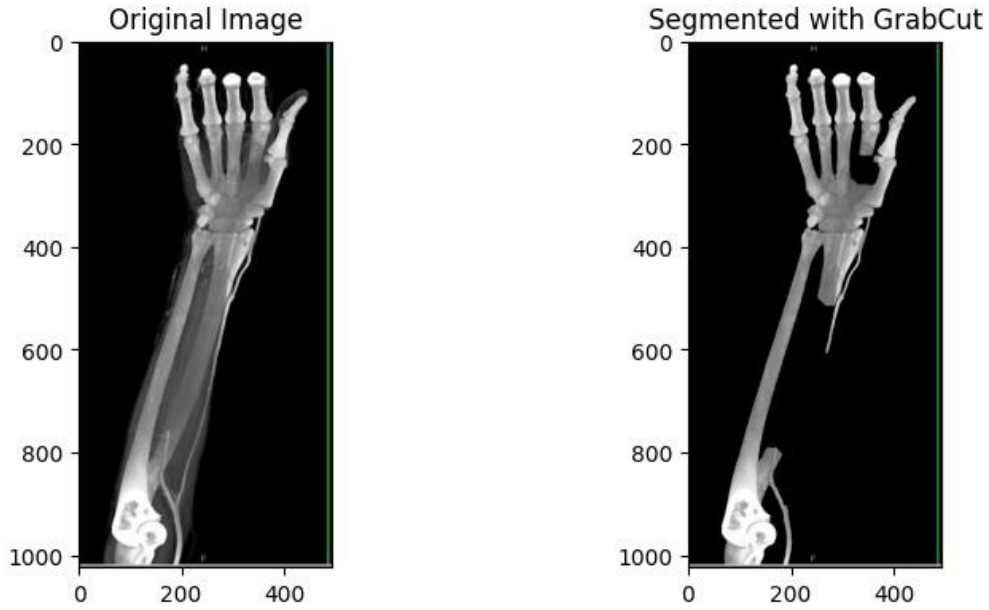


Рис 4. Результат сегментации снимка конечности методом GrabCut

1.3. Метод выращивания регионов

Идея метода выращивания регионов [4] заключается в следующем: сначала необходимо выбрать центры будущих регионов (сегментов). Это можно сделать произвольным образом или по заранее определенным правилам. Далее итеративно к каждому региону присоединяются соседние незанятые пиксели изображения, которые удовлетворяют некоторому заранее определенному условию. В данном алгоритме для проверки такого условия будет вычисляться разница яркостей между пикселями, которая в свою очередь сравнивается с пороговым значением, которое задается пользователем заранее. Пороговое значение здесь – гиперпараметр, который требует подбора для получения лучшего результата сегментации.

Процесс разрастания регионов не прекращается до тех пор, пока не останется ни одной точки изображения, которая могла бы быть присоединена к одному из уже выращенному региону.

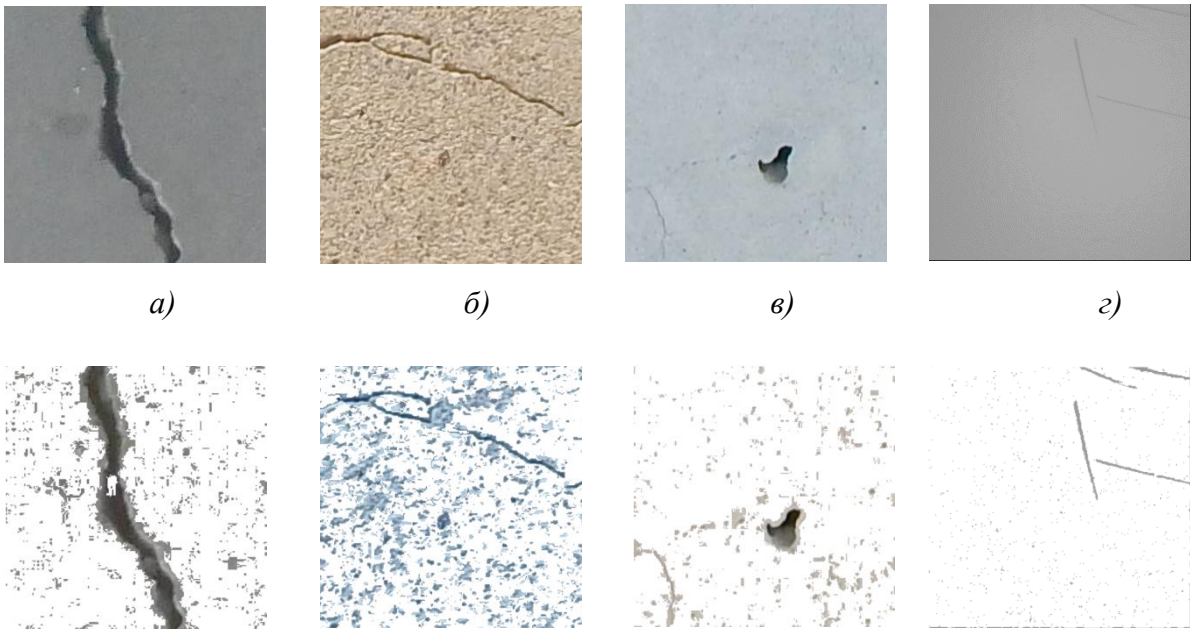
Результат работы алгоритма приведен на рис. 5.

Из полученных результатов видно, что алгоритм справился с задачей сегментации. Стоит выделить результат сегментации на рис. 5 з) – алгоритм также смог точно выделить объект без применения предварительной обработки. Однако, исходя из результатов на рис. 5 д) и рис. 5 е) можно сделать вывод о том, что алгоритм достаточно чувствителен к текстурным

особенностям поверхности.

Рассмотрение работы алгоритма при выделении костей человека со снимка проводилось для снимка предплечья и кисти человека. Тестовое изображение и результат сегментации приведен на рис. 6.

Исходя из полученных результатов можно сделать вывод о том, что метод выращивания регионов не полностью справился с задачей выделения костной ткани на снимке. Вся костная ткань выделена алгоритмом, но алгоритм также посчитал сосуды костной тканью.



а) *б)* *в)* *г)*

д) *е)* *ж)* *з)*

Рис 5. Результаты сегментации методом выращивания регионов. а-г) Изображения до сегментации, д-з) результаты после сегментации.

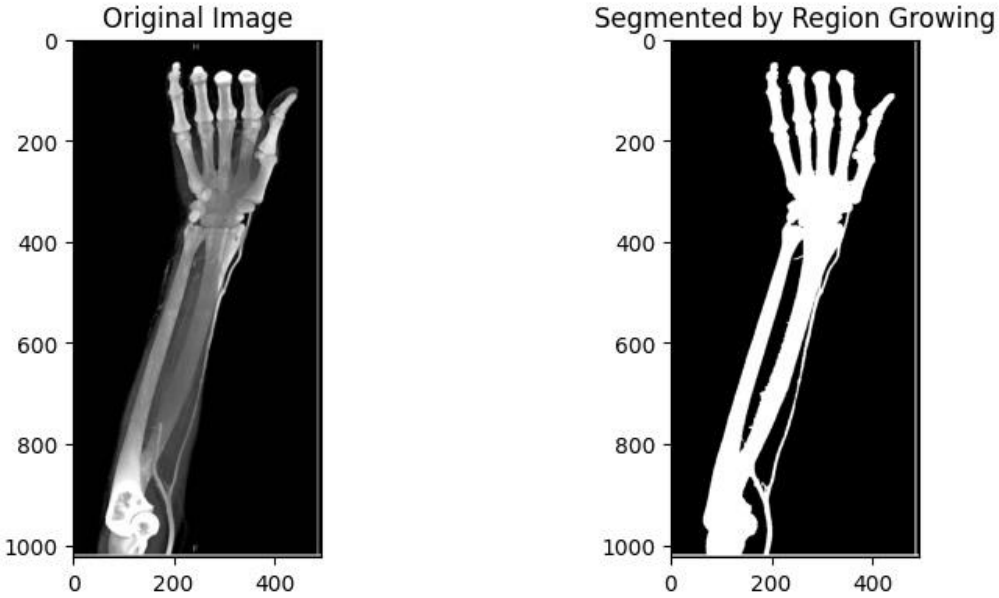


Рис 6. Результат сегментации снимка конечности методом выращивания регионов

2. Измерение времени работы алгоритмов

В ходе исследования были выбраны 3 алгоритма сегментации. Было проведено измерение времени работы алгоритмов. В табл. 1 представлено время работы рассмотренных алгоритмов (метод водораздела, метод GrabCut и метод выращивания регионов), рассчитанное как среднее время его работы для каждого из четырех тестовых изображений.

Таким образом, самым неэффективным по времени и результатам сегментации оказался метод водораздела.

Сильное временное различие между остальными алгоритмами можно объяснить тем, что метод водораздела требует большого количества вычислений и содержит несколько вложенных циклов, что значительно увеличивает алгоритмическую сложность метода.

Таблица 1

Время работы алгоритмов сегментации

Алгоритм	Время, с
Метод водораздела	17,75
Метод GrabCut	8,32
Метод выращивания регионов	6,76

Заключение

В результате проделанной работы были исследованы три алгоритма сегментации: метод водораздела, метод GrabCut и метод выращивания регионов. Исходя из полученных результатов можно сделать вывод о том, что метод GrabCut эффективнее справляется с выделением объекта на поверхности. Также хорошо справился метод водораздела. Из-за особенности реализации метода водораздела, четкие границы сегментов препятствуют фильтрации итогового изображения таким образом, чтобы на нем остался только выделенный дефект. В выделении костной ткани на снимках человеческого тела эффективнее всего оказался метод водораздела, он безошибочно выделил всю костную ткань, не выделив при этом ничего лишнего.

Также в ходе работы были проведены измерения времени работы алгоритмов сегментации. Метод водоразделов также оказался неэффективным и с точки зрения времени выполнения: в 2-3 раза дольше остальных.

Литература

1. Гонсалес, Р. С. Цифровая обработка изображений / Р. С. Гонсалес, Р. Е. Вудс. – 3-е изд., исп. и доп. – Москва:Техносфера, 2012. – 1104 с.

2. Цапаев, А. В. Методы сегментации изображений в задачах обнаружения дефектов поверхности / А. В. Цапаев, О. В. Кретинин // Компьютерная оптика. – 2012. – Т. 36. – № 3. – С. 448–452.

3. Boykov Y., Jolly M.-P. Interactive organ segmentation using graph cuts. In Medical Image Computing and Computer-Assisted Intervention, 2000. Pp. 276-286.

4. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Дж. Стокман; пер. с англ. – Москва: БИНОМ. Лаборатория знаний, 2006. – 752 с.

УДК 004.04

АНАЛИЗ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ЗАДАЧИ ПРОГНОЗИРОВАНИЯ ИЗМЕНЕНИЯ МАССЫ ТЕЛА

А. А. Казанин

Воронежский государственный университет

Введение

Алгоритмы машинного обучения находятся на пике своей популярности, проникая в различные сферы деятельности и привнося в них новые возможности и преимущества. От финансовых инвестиций до медицинских диагностик, от рекомендательных систем до анализа социальных сетей — используются повсеместно для решения разнообразных задач. Это объясняется их способностью обрабатывать и анализировать огромные объемы данных с высокой скоростью и точностью.

Прогнозирование считается одним из самых распространенных способов использования алгоритмов обучения из-за их значимости в различных областях. Способность предсказывать будущие события или тенденции на основе имеющихся данных имеет решающее значение для принятия стратегических решений в бизнесе, экономике, политике и других областях. Это позволяет компаниям предоставлять персонализированные услуги и продукты, соответствующие потребностям конечного пользователя.

Важность прогнозирования также обусловлена необходимостью адаптации к быстро меняющейся среде и принятия оперативных решений на основе актуальных данных. В условиях конкурентной борьбы умение предсказывать будущее становится ключевым фактором успешности. Таким образом, анализ алгоритмов обучения для задач прогнозирования не только актуален, но и остается одним из наиболее востребованных и направлений в области искусственного интеллекта и машинного обучения.

1. Постановка задачи

Требуется обучить модели, способные с высокой точностью прогнозировать изменения массы тела человека по заданным параметрам. Будем считать, что пользователь не имеет длительной истории наблюдений за этими параметрами (более распространенный случай).

Raw data представляют из себя набор характеристик, имеющих прямое или косвенное влияние на изменение массы тела человека [1, 2], а именно:

- id человека (на 1 человека может быть несколько записей);
- пол;
- возраст;
- рост;
- масса тела на текущий момент;
- порядковый номер дня измерений (относительно первого дня ведения записей для каждого конкретного человека);
- содержание дневного рациона [1];
- суточный расход калорий [1];
- тип тренировок [2] (фиксированный список вариантов);
- различные измерения тела [2];

- наличие существенных недомоганий (фиксированный список вариантов);
- субъективные оценки состояния на конец дня (в виде целых чисел).

2. Метрика оценки качества модели

Прежде всего следует упомянуть важную особенность предметной области. Raw data, используемые для прогнозирования, не являются обычным датасетом в контексте задач машинного обучения, они имеют важное отличие – на одного конкретного человека имеется группа записей (в контексте текущей задачи это измерения распределенные по дням). Такой подход позволяет гибко настроить взаимодействие пользователя с данными (например, меняя частоту измерений, что хорошо подойдет для людей, не готовых уделять приложению много времени).

Условимся, что из raw data была сформирована обучающая выборка, упорядоченная по id и порядковому номеру дня измерений.

Введем обозначения для модели:

n – количество признаков (столбцов);

l – общее количество измерений (строк);

$\overline{x}_i = (x_1^i, x_2^i, \ldots, x_n^i)$ – вектор измерений признаков (строка входных данных);

$X^l = \{\overline{x}_i\}_{i=1}^{l}$ – обучающая выборка (преобразованная из raw data, см. раздел 3);

$y_i = f(\overline{x}_i), \quad i = \overline{1,l}$ – целевое значение изменение массы тела за текущий день;

$a_i = f(\overline{x}_i), \quad i = \overline{1,l}$ – сгенерированное значение изменения массы тела за текущий день;

$L(a, \overline{x}_i)$ – функция потерь, в нашем случае:

$L(a, \overline{x}_i) = (a_i - y_i)^2, \quad i = \overline{1,l}$ – квадратичная ошибка;

$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l} L(a, \overline{x}_i)$ – метрика качества модели.

При текущих (классических) обозначениях метрика качества будет корректна для оценки модели в случае прогнозирования изменения массы тела за сутки. На деле, пользователю намного важнее видеть прогноз за более длительный промежуток времени. Поскольку классическая метрика не позволяет подобного, введем более объективную метрику.

Пусть $\overline{x}_1 = (x_1^1, x_2^1, \ldots, x_n^1)$ – текущие входные данные пользователя (для определенности считаем это первым днем измерений).

Тогда по ним можно получить a_1 – прогноз измененной массы тела на следующий день.

В таком случае, чтобы получить прогноз на 2-й день, потребуется выполнить прогноз $a_2 = f(\overline{x}_2)$, что приводит к проблеме, поскольку x_2 неизвестен.

Исправить эту проблему можно тем же методом, что используется в стохастическом градиентном спуске (в данном методе для оптимизации вычислений градиент заменяется псевдоградиентом что помогает кратно снизить количество вычислений и лишь немного потерять в сходимости). Чтобы выполнить замену, необходимо соблюсти условие незначительного отклонения заменяемого вектора и вектора, на который будет произведена замена. Иначе говоря, угол между этими векторами должен быть острым и в таком случае будет достигнута слабая потеря сходимости и сильно большей оптимизации вычислений.

В нашем случае речь идет о векторе наблюдений $\overline{x}_i$ на $\overline{x}_1$.

Еще раз рассмотрев датасет, можно сделать выводы о том, что в целом признаки меняются редко или незначительно, за исключением нескольких, которые способны значительно повлиять на точность, при замене замену $\overline{x}_i = \overline{x}_1$. Рассмотрим их.

Говоря о суточном расходе калорий, на практике, большую часть расходует поддержание жизнеспособности органов. Грубое решение – пренебречь изменениями, более оптимальное – программно усреднять значение для пользователя.

Также выделяется наличие существенных недомоганий, под которыми имеются ввиду заболевания, аномальный уровень стресса, высокая усталость и прочие вторичные факторы. Решение аналогично предыдущему случаю.

Таким образом, на ограниченном промежутке времени (в нашем случае не более 30 дней), при соблюдении оптимальных подходов, можно с достаточным уровнем точности полагать что:

$\overline{x}_i = \overline{x}_1 \Rightarrow a_i = a_1 \Rightarrow b_k = \sum_{i=1}^{k} a_1 = ka_1$ — изменение массы тела за k дней, $k = \overline{1,l}$

(равенство с первым вектором наблюдений сделано для примера, на практике можно положить произвольный вектор).

Дополнительно обозначим:

$\hat{y}_i^k = \sum_{i=1}^{k} y_i$ — целевое изменение массы тела за k дней, $k = \overline{1,l}$;

$X_k^l = \{\overline{x}_i \mid \left(\frac{(\overline{x}_i)_6}{k} = 1 \right) \wedge \left((\overline{x}_i)_5 = \sum_{s=1}^{k} (\overline{x}_s)_5 \right)\}_{i=1}^{l}$ — преобразованная обучающая выборка (согласно датасету из раздела 1, параметры по индексам: 5 – изменения массы, 6 – порядковый номер дня).

$p = \left| X_k^l \right|$ – количество людей в преобразованной выборке.

Получаем возможность вычисления оценки качества модели для произвольного количества дней (с учетом заданных ограничений):

$$\hat{Q}\left(a, X^l\right) = \frac{1}{p} \sum_{i=1}^{p} \hat{L}\left(a, \overline{x}_i\right) = \frac{1}{p} \sum_{i=1}^{p} \left(ka_1 - \hat{y}_i^k\right)^2 = \left(ka_1 - \hat{y}_i^k\right)^2$$

Относительно выборки X_k^l, это аналогично применению классической метрики, но с оговоркой что предсказание модели $\hat{a}_i = ka_1 = kf\left(\overline{x}_i\right)$.

В последствии, будем выполнять также простое математическое преобразование метрик для демонстрации успешности прогноза в процентах.

3. Подготовка обучающих выборок и функций оценок

Для применимости последующих алгоритмов выполним преобразования исходных измерений к обучающей выборке $X^l = \{\overline{x}_i\}_{i=1}^{l}$.

Для этого, а также для применимости оценки для произвольного временного промежутка, необходимо соблюсти условия:

1. Все категориальные [4] и строковые признаки приведены к числам.
2. Данные характеристики массы тела отражают не значение на конец дня, а его изменение за сутки.
3. Все данные упорядочены по id человека, и внутри этих кластеров повторно упорядочены по порядковому номеру дня.

Как указывалось ранее, для длительного прогноза формируется выборка:

$$X_k^l = \{\bar{x}_i \mid \left(\frac{(\bar{x}_i)_6}{k} = 1 \right) \wedge \left((\bar{x}_i)_5 = \sum_{s=1}^{k} (\bar{x}_s)_5 \right) \}_{i=1}^{l}$$

После, определяются функции вычисления метрик (рис. 2).
Дальнейшие вставки с кодом были написаны на языке Python в среде Google Colab.

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd

def dataset_init(): # Подготовка обучающего набора данных X^l
  url = "https://drive.google.com/uc?id=1Clmr3lJBZC4XzvsDhh9LszxewueYI3uU"
  global df
  df = pd.read_csv(url, delimiter=';')
  df = df.sort_values(by=['id', 'день'])
  previous_weights = {}
  for index, row in df.iterrows():
      current_id = row['id']
      current_weight = row['вес']
      if row['день'] == 1:
          df.at[index, 'вес'] = 0
      else:
          previous_weight = previous_weights.get(current_id, 0)
          weight_change = current_weight - previous_weight
          df.at[index, 'вес'] = weight_change
      previous_weights[current_id] = current_weight
  df['начало_сна'] = pd.to_datetime(df['начало_сна'], format='%H:%M').dt.hour \
  * 60 + pd.to_datetime(df['начало_сна'], format='%H:%M').dt.minute
  df['конец_сна'] = pd.to_datetime(df['конец_сна'], format='%H:%M').dt.hour \
  * 60 + pd.to_datetime(df['конец_сна'], format='%H:%M').dt.minute
  df = pd.get_dummies(df, columns=['пол', 'болезнь', 'женские_дни',
                                   'цель', 'тип_тренировки'])
  y = df['вес']
  X = df.drop(columns=['вес'])
  y = df['вес']
  return X, y

  k = 21

def create_Xkl_dataset(): # Подготовка обучающего набора данных X_k^l
    new_df = pd.DataFrame(columns=df.columns)
    weight_sum = 0
    for index, row in df.iterrows():
        if row['день'] <= k:
            weight_sum += float(row['вес'])
            new_row = row.copy()
            new_row['вес'] = weight_sum
            new_df.loc[index] = new_row
    Xk = new_df.drop(columns=['вес'])
    yk = new_df['вес']
    return Xk, yk
```

Рис. 1. Подготовка выборки

```python
def rate(y, y_test, y_pred): # Приведение метрики к виду в %
  mse = mean_squared_error(y_test, y_pred)
  print("- Среднеквадратичная ошибка (MSE):", mse)
  max_weight, min_weight = y.max(), y.min()
  max_mse = (max_weight - min_weight)**2
  success_rate = (1 - mse / max_mse) * 100
  print("- Успешность прогноза в %:", success_rate)

def full_rate(model, y, y_test, scaled=False): # Оценка метрик обученной модели
  y_pred = model.predict(X_test)
  print("Оценка прогноза за сутки")
  rate(y, y_test, y_pred)
  Xk, yk = create_Xkl_dataset()
  if scaled:
    scaler = MinMaxScaler()
    Xk_scaled = scaler.fit_transform(Xk)
    Xk = Xk_scaled.reshape(Xk_scaled.shape[0], 1, Xk_scaled.shape[1])
  yk_pred = k * model.predict(Xk)
  print(f"Оценка прогноза за k = {k} д.")
  rate(yk, yk, yk_pred)
```

Рис. 2. Функции вычисления метрик

4. Анализ алгоритмов в текущей предметной области

В данном разделе будут представлены примеры обучения моделей. Однако следует отметить, что задача оптимизации гиперпараметров для них не является объектом изучения данной статьи и не будет рассматриваться. Предполагается, что набор гиперпараметров для моделей был подобран исходя из эмпирических соображений.

4.1. Линейная регрессия

Линейная регрессия [3] — один из наиболее простых и понятных методов прогнозирования, основанный на линейной зависимости между входными признаками и целевой переменной. Подходит для анализа датасета из-за своей простоты, интерпретируемости и способности моделировать линейные зависимости между переменными. Модель обучается быстро, не требует сильных предположений о данных и может быть улучшена с помощью регуляризации. Однако она может оказаться недостаточно гибкой для моделирования сложных нелинейных связей в данных.

```python
X, y = dataset_init()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression()
model.fit(X_train, y_train)
full_rate(model, y, y_test)
```

```
Оценка прогноза за сутки
- Среднеквадратичная ошибка (MSE): 0.004094090018675636
- Успешность прогноза в %: 98.36236399252975
Оценка прогноза за k = 21 д.
- Среднеквадратичная ошибка (MSE): 82.01221881825651
- Успешность прогноза в %: 79.9009364723418
```

Рис. 3. Оценки линейной регрессии

4.2. Деревья решений

Деревья решений [3] — это модель машинного обучения, которая разбивает пространство признаков на множество прямоугольных областей и принимает решения на основе значений признаков. В отличие от регрессии, деревья решений могут обрабатывать как категориальные, так и числовые признаки без предварительного преобразования. Кроме того, деревья решений могут автоматически выявлять нелинейные зависимости в данных и строить разделяющие границы, которые могут быть сложными и нелинейными. Однако деревья решений могут быть склонны к переобучению, особенно на данных с большим количеством признаков или высокой размерностью.

```
from sklearn.tree import DecisionTreeRegressor

X, y = dataset_init()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
full_rate(model, y, y_test)
```

```
Оценка прогноза за сутки
- Среднеквадратичная ошибка (MSE): 0.0031666666666666852
- Успешность прогноза в %: 98.73333333333333
Оценка прогноза за k = 21 д.
- Среднеквадратичная ошибка (MSE): 83.25361904761894
- Успешность прогноза в %: 79.59670153719759
```

Рис. 4. Оценки дерева решений

4.3. Многослойный персептрон

Многослойный персептрон [3, 5] — это модель искусственной нейронной сети, состоящая из нескольких слоев нейронов (входной, скрытые слои и выходной). Каждый нейрон получает входные сигналы от предыдущего слоя, вычисляет их взвешенную сумму с учетом соответствующих весов и применяет активационную функцию для вычисления своего выходного значения. Многослойные персептроны могут обрабатывать как категориальные, так и числовые признаки без предварительного преобразования, способны выявлять сложные нелинейные зависимости, благодаря гибкой структуре сети и разнообразию активационных функций. Могут страдать от проблемы затухания градиента при обучении на глубоких сетях, требуют большого объема данных для эффективного обучения.

```
from sklearn.neural_network import MLPRegressor

X, y = dataset_init()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = MLPRegressor(activation='tanh', max_iter=7000,
                     hidden_layer_sizes=(90, 90, 20))
model.fit(X_train, y_train)
full_rate(model, y, y_test)
```

```
Оценка прогноза за сутки
- Среднеквадратичная ошибка (MSE): 0.037299673138998776
- Успешность прогноза в %: 85.08013074440048
Оценка прогноза за k = 21 д.
- Среднеквадратичная ошибка (MSE): 64.68207222690259
- Успешность прогноза в %: 84.14810503212856
```

Рис. 5. Оценки многослойного персептрона

4.4. Рекуррентная нейросеть с модификацией LSTM

Рекуррентные нейронные сети (RNN) [3, 5] — это класс нейронных сетей, способных обрабатывать последовательные данные, сохраняя внутреннее состояние или "память" о предыдущих входах. Они могут эффективно моделировать последовательности переменной длины и выражать зависимости во времени между входными данными. Однако обычные RNN имеют проблему исчезающего градиента при обучении на длинных последовательностях, что затрудняет обучение на длинных зависимостях.

LSTM (Long Short-Term Memory) [5] — это специальный тип рекуррентной нейронной сети, разработанный для преодоления проблемы исчезающего градиента и обучения долгосрочных зависимостей в данных. Они обладают дополнительными узлами памяти, которые помогают сохранять информацию на протяжении длительных временных интервалов. LSTM состоит из трех основных компонентов: ячейки памяти, фильтров входных ворот и фильтров выводных ворот, что делает их способными к запоминанию и использованию информации на длительные периоды времени.

```
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense

X, y = dataset_init()
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2)
X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))

model = Sequential()
model.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='sgd')
model.fit(X_train, y_train, epochs=15, batch_size=30, verbose=0)
full_rate(model, y, y_test, scaled=True)
```

```
2/2 [==============================] - 0s 7ms/step
Оценка прогноза за сутки
- Среднеквадратичная ошибка (MSE): 0.026734585123034917
- Успешность прогноза в %: 89.30616595078604
7/7 [==============================] - 0s 5ms/step
Оценка прогноза за k = 21 д.
- Среднеквадратичная ошибка (MSE): 59.885477943690745
- Успешность прогноза в %: 85.32362563873866
```

Рис. 6. Оценки LSTM

Заключение

В ходе анализа были исследованы факторы [1, 2], влияющие на задачу прогнозирования в текущей предметной области, рассмотрены алгоритмы оценки качества модели, потенциальные решения для составления прогнозов и проведены вычислительные эксперименты по обучению. Разнообразие алгоритмов и их результатов на различных временных промежутках позволяет подобрать точный прогноз изменения массы тела под текущие параметры и поведение пользователя, что в свою очередь, поможет ему сохранять мотивацию и двигаться к цели.

Литература

1. Association Between Diet, Physical Activity and Body Mass Index, Fat Mass Index and Bone Mineral Density of Soldiers of the Polish Air Cavalry Units — Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7019523
2. Effects of Exercise Interventions on Weight, Body Mass Index, Lean Body Mass and Accumulated Visceral Fat in Overweight and Obese Individuals: A Systematic Review and Meta-Analysis of Randomized Controlled Trials — Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7967650
3. Raschka, S., Mirjalili, V. Python Machine Learning. Third Edition / Sebastian Raschka, Vahid Mirjalili. — Birmingham - Mumbai: Packt Publishing Ltd., 2019. — 741 с.
4. Brownlee, J. Data Preparation for Machine Learning. / Jason Brownlee. — Edition: v1.1., 2020. — 381 с.
5. Haykin, S. Neural Networks and Learning Machines / Simon Haykin. — 3rd ed. — Upper Saddle River, New Jersey: Pearson Education, Inc., 2009. — 905 с.

УДК 004.4, 004.7

СОЗДАНИЕ СИСТЕМЫ ИНТЕЛЛЕКТУАЛЬНОГО ПОИСКА В СЕТИ ИНТЕРНЕТ С ИСПОЛЬЗОВАНИЕМ ИИ

А. И. Камышанов

Воронежский государственный университет

Введение

Создание интеллектуальной поисковой системы в интернете с использованием искусственного интеллекта является актуальной задачей в современном информационном мире. С ростом объема цифровой информации в сети Интернет возрастает потребность в разработке инновационных методов поиска и анализа данных. В данном контексте, использование технологий искусственного интеллекта становится ключевым фактором для создания эффективных и интеллектуальных поисковых систем.

Применение искусственного интеллекта в разработке поисковых систем позволяет автоматизировать процессы анализа и обработки информации, улучшить качество поисковых запросов и повысить релевантность получаемых результатов. Технологии машинного обучения, обработки естественного языка и алгоритмы рекомендаций играют важную роль в создании интеллектуальных поисковых систем, способных адаптироваться к изменяющимся запросам пользователей и эффективно обрабатывать огромные объемы информации.

В данной статье рассматривается процесс создания системы интеллектуального поиска в сети Интернет с использованием методов искусственного интеллекта.

1. Описание проекта «Система интеллектуального поиска»

Система интеллектуального поиска (СИП) – это веб-приложение, разработанное для эффективного и интеллектуального поиска информации в интернете с использованием технологий искусственного интеллекта (ИИ). СИП позволяет пользователям получать релевантные и точные результаты поиска, учитывая их запросы, предпочтения и контекст.

Приложение представляет собой клиентскую часть, которая обеспечивает удобный ввод данных для поиска и включает в себя следующий функционал:

1. Поиск и анализ информации: СИП осуществляет поиск по различным источникам информации в интернете, анализирует контент и предоставляет пользователю наиболее релевантные результаты.
2. Персонализированный поиск: СИП учитывает предпочтения пользователя и его предыдущий поиск для предоставления персонализированных рекомендаций и результатов поиска.
3. Улучшенное ранжирование результатов: Используя алгоритмы машинного обучения, СИП оптимизирует ранжирование результатов поиска, обеспечивая более точные и актуальные результаты.
4. Обработка естественного языка: СИП анализирует запросы пользователей на естественном языке, понимает их семантическую структуру и контекст для точного поиска информации.
5. Адаптивная фильтрация контента: СИП фильтрует полученную информацию в соответствии с интересами и запросами пользователя, обеспечивая более релевантный контент.

2. Этапы разработки

Разработка web-приложения была разделена на несколько этапов:

1. Определение требований: сначала необходимо определить требования и цели системы. Это включает в себя определение основных функций поиска, управления данными, а также учет пользовательских потребностей и предпочтений.
2. Исследование технологий: проведение исследований по существующим технологиям и методам в области искусственного интеллекта, машинного обучения, обработки естественного языка и поисковых систем.
3. Разработка и обучение моделей ИИ: создание и обучение моделей и алгоритмов искусственного интеллекта для выполнения функций поиска, анализа и ранжирования результатов.
4. Разработка пользовательского интерфейса: создание удобного и интуитивно понятного интерфейса для взаимодействия пользователей с системой, включая функционал ввода запросов и просмотра результатов.
5. Интеграция и тестирование: интеграция всех компонентов системы и проведение тестирования на различных этапах разработки для обеспечения качества и надежности работы.
6. Публикация проекта.

3. Анализ TensorFlow Framework

3.1. Краткое описание

Для разработки и обучения модели искусственного интеллекта (ИИ) в проекте интеллектуального поиска в сети интернет использовался фреймворк TensorFlow. TensorFlow – это открытая библиотека машинного обучения, разработанная компанией Google, предоставляющая инструменты для создания и обучения различных моделей искусственного интеллекта. При разработке были использованы следующие технологии:

1. Jupyter Notebook. Использование Jupyter Notebook позволило проводить эксперименты, анализировать данные и визуализировать результаты обучения. Были созданы отдельные ноутбуки для различных этапов процесса обучения, что обеспечило гибкость и удобство в работе.
2. TensorBoard. TensorBoard использовался для визуализации процесса обучения и мониторинга метрик производительности модели. Этот инструмент позволял анализировать графы вычислений, сравнивать результаты обучения и оптимизировать процесс.
3. GPU-ускорение. Для ускорения процесса обучения использовались графические процессоры (GPU). TensorFlow обеспечивает интеграцию с GPU, что позволяет ускорить вычисления, особенно для моделей с большим количеством параметров и объемом данных.

3.2. Инструментарий

Поиск был реализован на языке Python, поскольку Python идеально подходит для этой задачи, обеспечивая гибкость, простоту в использовании и богатый выбор библиотек для обработки данных и машинного обучения. Для хранения данных была использована база

данных MySQL. Данная СУБД была выбрана, поскольку MySQL легко интегрируется с Python и другими технологиями, используемыми в проекте, что обеспечивает эффективное взаимодействие между компонентами системы, а также обладает мощными механизмами безопасности, что защищает данные от несанкционированного доступа и взлома. Также необходимо учитывать, что разработка данного веб-приложения не предполагает хранение огромного количества данных, а следовательно использование платных, промышленных СУБД избыточно. MySQL является одной из наиболее производительных СУБД для небольших данных, а её кроссплатформенность исключает привязку к определенному типу устройств. Для работы с MySQL в Python доступны различные библиотеки, такие как pymysql или mysql-connector-python, которые позволяют легко выполнять запросы к базе данных, извлекать данные и взаимодействовать с ними.

Для работы с данными в базе данных MySQL был использован ORM Framework SQLAlchemy. Он предоставил удобный и абстрактный способ работы с базой данных на уровне объектов, что позволило избежать написания SQL запросов и упростило взаимодействие с базой данных. Основными плюсами данного фреймфорка являются:

1. Гибкость. SQLAlchemy позволяет работать с данными, представляя их в виде объектов Python, что делает код более читаемым и понятным.
2. Удобство использования. SQLAlchemy предлагает простой и интуитивно понятный API для выполнения различных операций с данными. Можно легко выполнять операции CRUD (Create, Read, Update, Delete).
3. Производительность. SQLAlchemy обеспечивает эффективную работу с базой данных, оптимизируя запросы и минимизируя количество обращений к ней.

4. Реализация проекта

4.1. Проблематика

При разработке Интеллектуальных Поисковых Систем (ИПС) встречаются несколько ключевых проблем. Прежде всего, сбор данных может стать сложной задачей из-за разнообразия источников и их нерегулярной структуры. Возникает слабая степень релевантности данных, что затрудняет точный поиск. Помимо этого, проблемы с ранжированием и хранением данных оказывают существенное влияние на эффективность ИПС. Применение искусственного интеллекта (ИИ) позволяет решить эти проблемы: алгоритмы машинного обучения и нейронные сети могут автоматизировать процесс сбора и структурирования данных, улучшить релевантность результатов поиска, оптимизировать ранжирование и создать эффективные системы хранения, способные обрабатывать большие объемы информации. Таким образом, ИИ играет ключевую роль в разрешении основных проблем при разработке ИПС, обеспечивая более точный и быстрый поиск для пользователей.

4.2. Разработка архитектуры приложения

Перед началом воплощения проекта была спроектирована его архитектура, определяющая структуру и компоненты. Архитектура приложения охватывала следующие элементы:

1. Веб-интерфейс (Web Interface): Этот компонент отвечает за взаимодействие с пользователем. Он принимает запросы от пользователей через веб-интерфейс и передает их обработчикам для дальнейшей обработки. Веб-интерфейс

отображает результаты поиска пользователю, предоставляя удобный способ взаимодействия с приложением.

2. Обработчики (Handlers): Обработчики принимают запросы от веб-интерфейса и взаимодействуют с моделями данных и сервисами. Они управляют операциями поиска и обработки запросов пользователей, обеспечивая точные и своевременные ответы.
3. Модели данных (Data Models): Эти компоненты представляют собой объекты данных, необходимые для работы приложения. Они включают в себя структуры для запросов пользователей, результаты поиска и информацию о пользователях. Модели данных обеспечивают единое хранилище информации для эффективной обработки запросов.
4. Сервисы (Services): Сервисы содержат основную бизнес-логику приложения. Они обрабатывают запросы от обработчиков, взаимодействуют с внешними сервисами, такими как нейронные сети для поиска, и управляют операциями поиска и обработки данных. Сервисы обеспечивают надежную и эффективную работу приложения, обеспечивая качественный интеллектуальный поиск в интернете.

4.3. Разработка базы данных

Была создана структура базы данных, в которой были определены следующие таблицы:

1. Пользователи (Users): содержит информацию о пользователях, включая логин, пароль и другие данные.
2. Запросы пользователя (Users): содержит информацию о запросах, отправленных пользователями, такие как текст запроса и дата.
3. Результаты поиска (Book application): содержит информацию о результатах поиска, включая название книги, идентификатор пользователя и дату.
4. История поиска (Roles): содержит записи о поисковых сессиях пользователей, включая дату и время начала сессии.

Заключение

В результате нашей работы было разработано веб-приложение, которое полностью соответствует поставленным задачам. После внедрения искусственного интеллекта (ИИ) в поиск и проведения сравнительного анализа между поиском с применением ИИ и без него. Результаты показали, что поиск с использованием искусственного интеллекта оказался более эффективным и точным, чем поиск без его применения.

Литература

1. TensorFlow: глубокое обучение искусственных нейронных сетей. – Режим доступа: https://www.tensorflow.org/guide (Дата обращения: 10.03.2024)
2. Python для анализа данных: наука о данных и машинное обучение с примерами кода на Python. – Режим доступа: **Ошибка! Недопустимый объект гиперссылки.** (Дата обращения: 10.02.2024)
3. "Глубокое обучение" от Иэна Гудфеллоу, Йошуа Бенжио и Аарона Курвиля. – Режим доступа: https://www.deeplearningbook.org/ (Дата обращения: 28.03.2023)

УДК 004.8

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ЯЗЫКА PYTHON ПРИ РАЗРАБОТКЕ УМНОГО ЗЕРКАЛА

К. Ю. Капшуков

Воронежский государственный университет

Введение

В век развития технологий люди научились изобретать различные устройства, позволяющие им всячески облегчать свою работу. На сегодняшний день почти у каждого изобретенного устройства появляется его улучшенная версия. В такие бытовые приборы как холодильники, колонки, а также автомобили и т.д. внедряются все возможные датчики и электроника, переводя их в разряд «умных» устройств.

Зеркала присутствуют в жизни человека с незапамятных времен. За весь срок своего существования они прошли путь от чрезвычайно простых до технически сложных устройств, способных выполнять далеко не одну первоначальную функцию отражения.

Интерактивные умные зеркала можно использовать как полноценные гаджеты, размещенные в удобном месте, таком как квартира, магазин, автомобиль или в офис. Их возможности ограничены только используемой операционной системой и установленным программным обеспечением.

В данной статье рассматриваются анализ существующих решений и особенности использования языка Python при разработке умного зеркала.

1. Описание проекта «Умное зеркало»

Умное зеркало — это продукт современных и прогрессивных технологий, являющий собой устройство с зеркальной поверхностью, за которой располагается монитор. Благодаря этому в выключенном состоянии устройство полностью похоже на зеркало, а после включения на отражающей поверхности появляется необходимая информация.

Устройство состоит из следующих компонентов:

1. Корпус. Металлическая, деревянная, пластмассовая рама для совмещения комплектующих устройства внутри.
2. Платформа. Компьютер или микроконтроллер с программным обеспечением для функционирования устройства.
3. Дисплей. Место вывода изображения.
4. Зеркало. Двухстороннее зеркало или стекло с пленкой.

Для разработки можно использовать относительно старые вещи: монитор, ноутбук, телевизор. Главное, удостовериться в их исправной работе.

Использование умного зеркало основывается на функции распознавания лица, что может пригодиться в бытовых вещах. Разработка алгоритма имеет несколько вариантов, но из сравнительного аналитического анализа можно выбрать удобный и функциональный.

Нотация IDEF0 дает представления об основных этапах разработки (рис. 1). Также показывает источники информации (материалы сети Интернет), где указаны существующие решения.

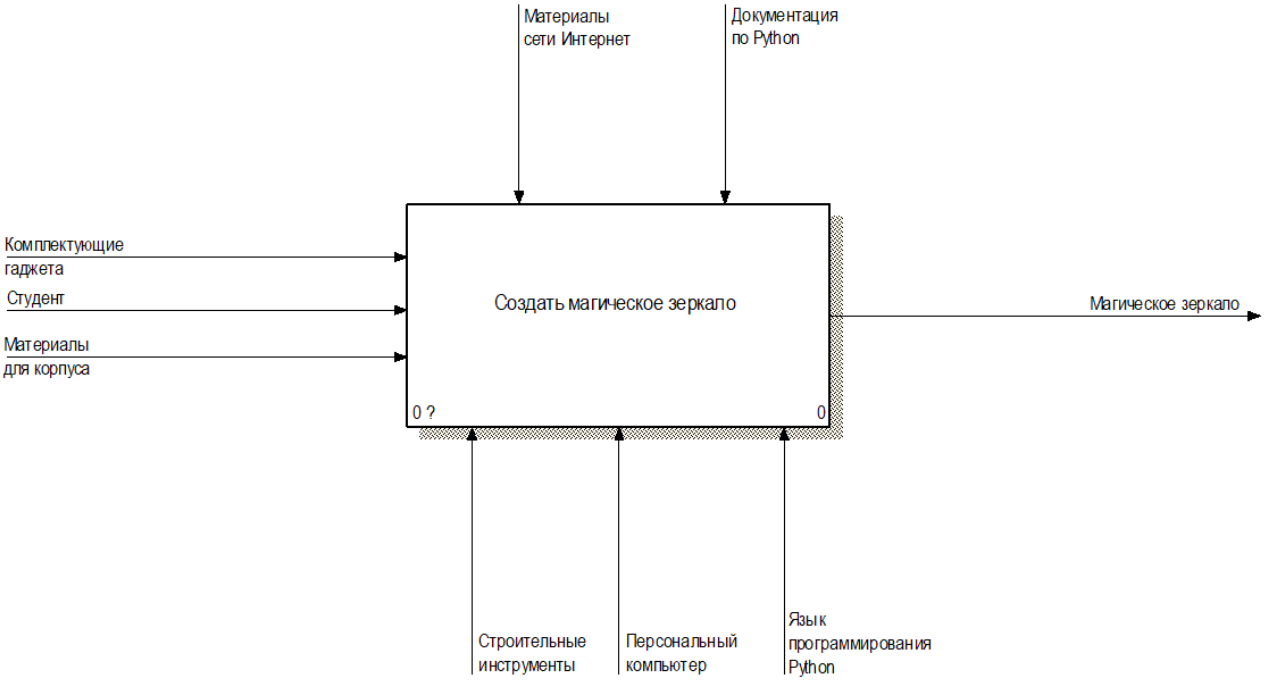


Рис. 1. Блок «Создать магическое зеркало»

2. Этапы разработки

Основные этапы разработки проекта представлены на рис. 2.

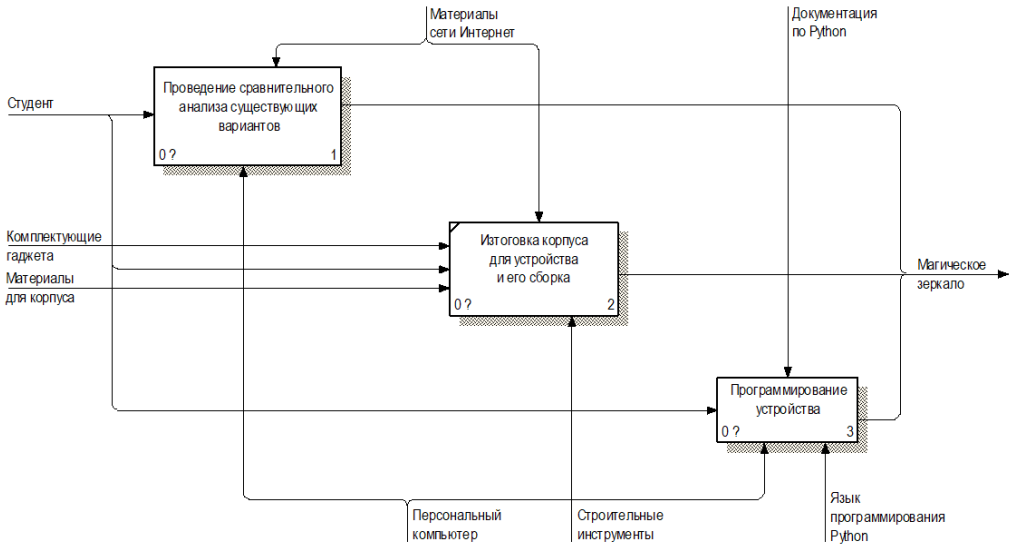


Рис. 2. Декомпозиция блока «Создать магическое зеркало»

Проведение сравнительного анализа необходимо, так как наличие огромного инструментария для создания устройства и варианты его применения тесно связаны.

3. Анализ библиотеки face_recognition

Распознавание лиц является хорошо изученной проблемой и широко используется как в индустрии, так и в научных кругах.

Каждый алгоритм машинного обучения принимает набор данных в качестве входных данных и учится на них. Алгоритм просматривает данные и выявляет закономерности. Есть несколько вещей, которые можно рассматривать как паттерн:

1. Высота/ширина лица.
2. Высота и ширина могут быть ненадежными, поскольку изображение может быть масштабировано по-разному. Однако даже после изменения масштаба неизменными остаются отношения - отношение высоты лица к ширине лица не изменится.
3. Цвет лица.
4. Ширина других частей лица, таких как губы, нос и так далее.

Машинное обучение может помочь с двумя вещами:

1. Получение вектора объекта: трудно вручную перечислить все признаки, потому что их много. Алгоритм машинного обучения может разумно обозначить многие из таких признаков. Например, сложными характеристиками могут быть: соотношение высоты носа и ширины лба. Человеку будет довольно сложно перечислить все такие функции;
2. Алгоритмы согласования: после того, как векторы признаков получены, алгоритм машинного обучения должен сопоставить новое изображение с набором векторов признаков, присутствующих в выборке.

Существует удивительно простая библиотека Python, которая инкапсулирует все, что мы рассматривали выше - создание векторов признаков из лиц и умение различать лица. Эта библиотека Python называется face_recognition и в глубине использует dlib — современный фреймворк C ++, который содержит несколько алгоритмов машинного обучения, которые помогают в написании сложных приложений на C ++.

Библиотека face_recognition в Python может выполнять большое количество задач, но для реализации алгоритма достаточно следующих:

- найти и управлять чертами лица в изображении;
- распознавать лица в реальном времени;
- идентифицировать лица на изображениях.

Заключение

Результатом исследования стал анализ средств языка программирования Python для разработки умного зеркала. На основе анализа была выбрана библиотека face_recogniton, набора функций которой будет достаточно для реализации проекта.

Литература

1. Документация face_recognition.: – Режим доступа: https://face-recognition.readthedocs.io/en/latest/ (Дата обращения 17.04.2024)
2. Документация Python – Режим доступа: https://pydocs.ru/ (Дата обращения 17.04.2024)
3. Умное зеркало.: – Режим доступа: https://habr.com/ru/companies/ruvds/articles/524166/ (Дата обращения 17.04.2024)
4. Разработка устройства «SMART MIRROR» – Режим доступа: https://cyberleninka.ru/article/n/razrabotka-ustroystva-smart-mirror/viewer (Дата обращения 18.04.2024)

УДК *004.4*

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПОДГОТОВКИ К ЭКЗАМЕНАМ

Д. Р. Качапин

Воронежский государственный университет

Введение

В современном мире экзамены являются неотъемлемой частью образовательного процесса и профессионального развития. Однако подготовка к экзаменам - сложный и трудоемкий процесс, требующий значительных затрат времени и сил. В связи с этим важно создавать мобильные приложения, которые оптимизируют процесс подготовки к экзаменам и повышают его эффективность. Такие приложения предоставляют пользователям удобный и систематизированный доступ к справочным материалам, тестам и другим полезным функциям, помогают систематизировать знания и уверенно сдать экзамен. Разрабатываемые мобильные приложения для iOS облегчат подготовку к экзаменам и повысят их процент успешного прохождения. Приложение автоматизирует выбор справочных материалов, разделенных по темам и заданиям, и позволяет оценивать и фильтровать их на основе оценок пользователя. Кроме того, в приложении предусмотрена возможность расширения теста коллективных идей для генерации новых идей на основе предложений внутренних сотрудников.

1. Функциональные требования

В приложении выделена одна роль для любого авторизовавшегося человека: пользователь.

Мобильное приложение обладает следующими функциональными возможностями:

1. Вход в личный кабинет пользователя;
2. Просмотр результатов изученного материала;
3. Отображение списка предложений;
4. Отображение списка всех тем;
5. Переход на экран выбора темы;
6. Просмотр таблицы результатов всех пользователей в личном кабинете;
7. Отображение справочного материала при необходимости;
8. Отображение информации о пользователе на отдельном экране;
9. Возможность фильтрации заданий и справочного материала;
10. Возможность закрыть приложение.

Исходные данные включают в себя:

1. Информацию о пользователе: логин, пароль и номер мобильного телефона;
2. Хранение информации о задании: тип задания, список тем, список справочных материалов.

1.1. Сценарий использования

Каждый пользователь имеет доступ к списку тем, у каждой темы есть общий справочный материал и банк тем (рис. 1).

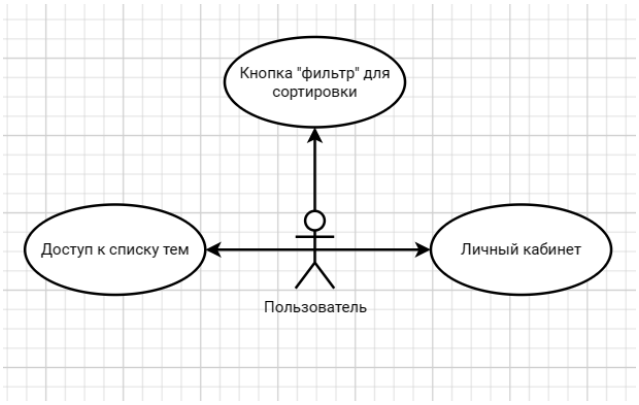


Рис. 1.

При активации кнопки "Фильтр" пользователь получает доступ к выпадающему списку, содержащему перечень возможных тем, тем со справочным материалом и практических заданий, которые можно решить. Затем в окне отображается материал, отфильтрованный по желанию пользователя (рис. 2).

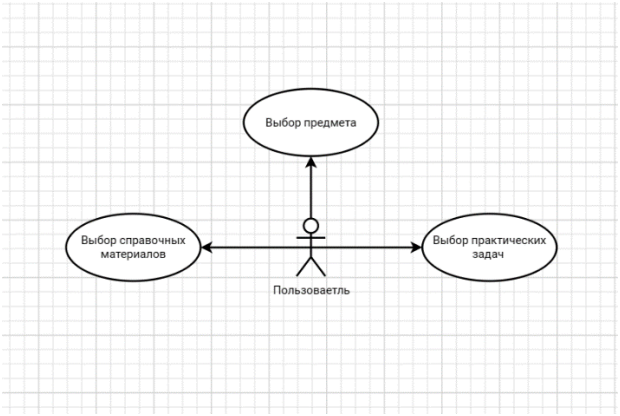


Рис. 2.

При нажатии на кнопку "Личный кабинет" отображаются персональные данные и статистика, связанная с изученным материалом. На этом экране можно добавить или удалить фотографию профиля, изменить имя и номер телефона (рис. 3).

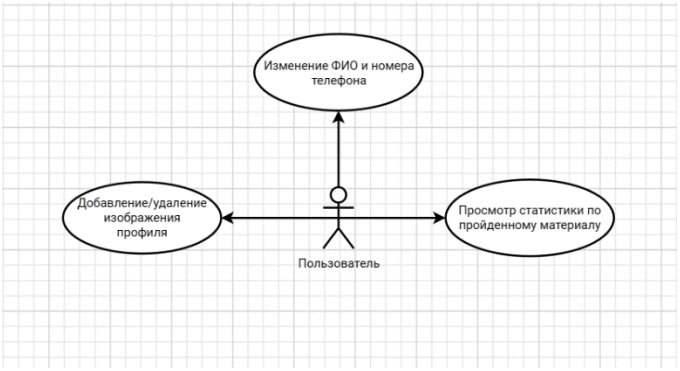


Рис. 3.

Ниже приведена диаграмма действий, в которой перечислены все возможные действия (рис. 4).

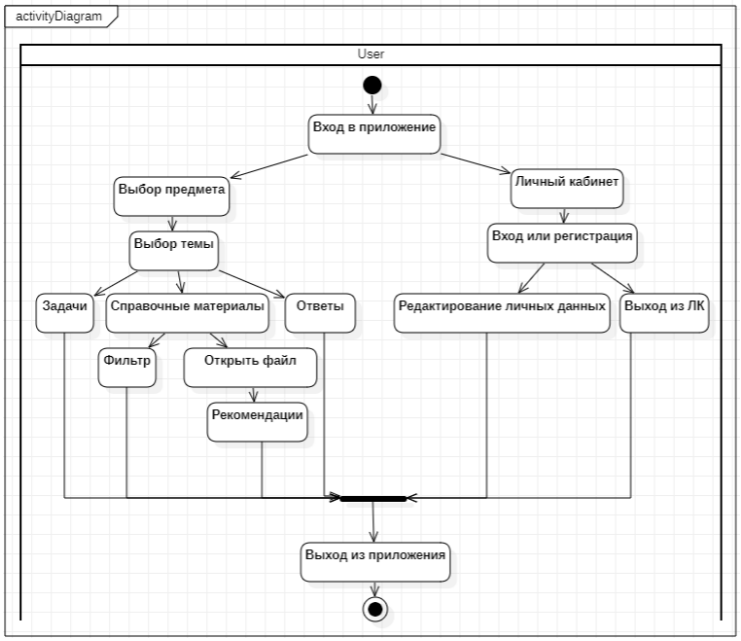


Рис. 4.

2. Реализация

Приложение было разработано в среде Xcode. В качестве языка программирования был выбран Swift, а в качестве API - REST API (рис. 5).

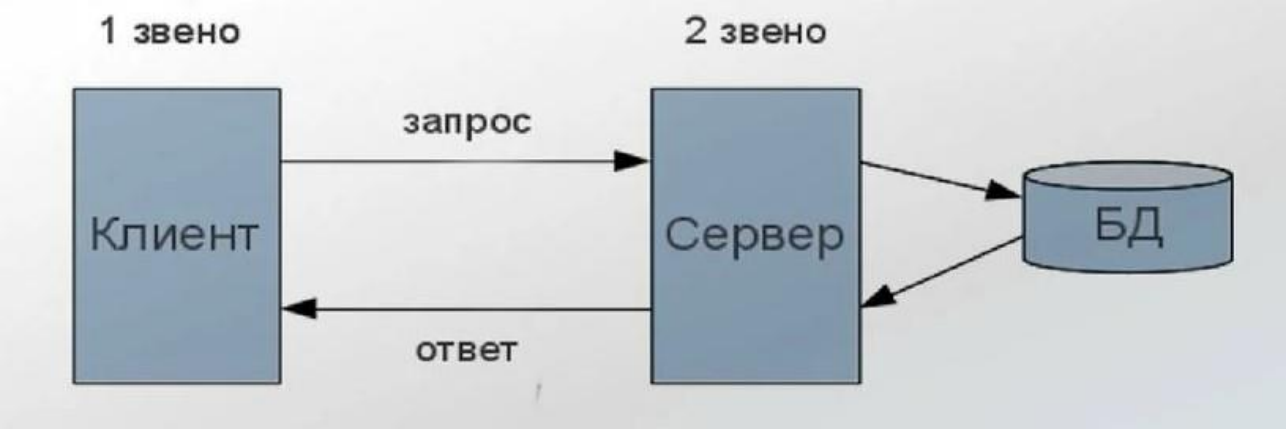


Рис. 5. Архитектура клиент серверного приложения [1]

REST был выбран потому, что он включает в себя простоту использования, независимость от языков программирования и платформ, масштабируемость и гибкость REST API широко используются в мобильных приложениях и других приложениях, которым необходимо обмениваться данными через интернет. Они широко используются при разработке мобильных и других приложений, которым необходимо обмениваться данными через интернет. Для создания пользовательских интерфейсов, обработки данных и взаимодействия с внешними сервисами также используются различные фреймворки и библиотеки.

Мобильные клиент-серверные приложения, использующие REST API и базы имеют многоуровневую архитектуру «клиент — сервер», в которой функция обработки данных вынесена на несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов [2]. Данное приложение имеет следующую архитектуру:

Клиентский уровень

1. Мобильное приложение, запускаемое на устройстве пользователя;
2. Предоставление графического пользовательского интерфейса и взаимодействие с пользователем;
3. Взаимодействие с REST API для получения и хранения данных.

Уровень REST API

1. Серверный компонент, предоставляющий набор маршрутов и методов для обработки HTTP-запросов;
2. Выполнение операции над данными в базе и возвращение результатов в формате JSON [3];
3. Обеспечение абстракции между клиентским уровнем и уровнем базы данных.

Уровень базы данных.

1. Хранилище данных для хранения данных приложения;
2. Реляционные (например, MySQL, PostgreSQL) или нереляционные (например, MongoDB, Cassandra) базы данных;
3. Управление данными и обеспечивает их целостность, доступность и безопасность.

Мобильное приложение отправляет HTTP-запросы на REST API, который взаимодействует с базой данных для выполнения операций с данными. Результаты или обновленные данные возвращаются в мобильное приложение через REST API. Такая архитектура обеспечивает разделение обязанностей и гибкость, поскольку клиентский уровень, REST API и база данных могут разрабатываться и развертываться независимо.

3. Реализация серверной части

Взаимодействие между мобильным клиентом и сервером в данном контексте происходит через REST API, который предоставляет набор маршрутов и методов, которые мобильный клиент может вызывать для выполнения различных операций [4].

Схема взаимодействия:

1. Мобильный клиент отправляет запрос: мобильное приложение отправляет HTTP-запрос на определенный маршрут REST API, указывая соответствующие данные или параметры запроса;
2. Обрабатывает запрос REST API: получает запрос REST API и анализирует маршрут и методы. Он извлекает из запроса необходимые данные;
3. REST API взаимодействует с базой данных: REST API выполняет соответствующие операции с базой данных, такие как вставка, выборка, обновление и удаление;
4. REST API возвращает ответ: когда операция завершена, REST API генерирует ответ и отправляет его обратно мобильному клиенту;
5. Мобильный клиент обрабатывает ответ: Мобильное приложение получает ответ и обрабатывает его в соответствии с логикой приложения. Оно отображает данные пользователю, обновляет локальные данные или выполняет другие действия.

3.1. Примеры взаимодействия

При создании пользователем новой задачи в приложении, мобильный клиент отправляет HTTP POST-запрос на маршрут '/tasks' в REST API: REST API обрабатывает запрос, создает новую запись в таблице задач в базе данных и возвращает ответ, содержащий идентификатор новой задачи. Мобильное приложение получает ответ и отображает созданную задачу в пользовательском интерфейсе. Такое взаимодействие позволяет мобильному клиенту взаимодействовать с данными на сервере через REST API, абстрагируясь от деталей базы данных и серверного приложения. Такая структура позволяет разрабатывать и развертывать мобильный клиент и сервер независимо друг от друга, обеспечивая гибкость и масштабируемость.

На диаграмме классов показана используемая структура базы данных (рис. 6).

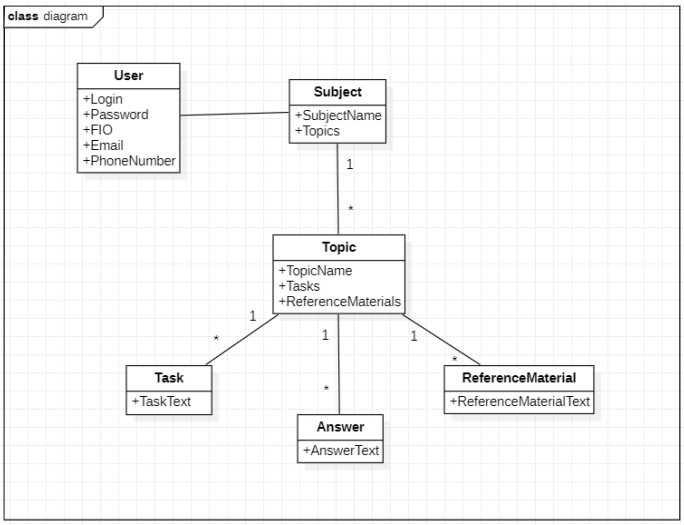


Рис. 6. Диаграмма классов

Заключение

Подводя итог, можно сказать, что создание этого мобильного приложения является важным шагом в области образования и личностного развития, предоставляя пользователям инновационный инструмент для совершенствования своих знаний и навыков. Благодаря своей функциональности и возможностям, приложение может успешно использоваться как в частных целях, так и в учебных заведениях, способствуя повышению качества обучения и развития навыков. В целом, разработка данного продукта создает новые возможности для обучения и преподавания, делая этот процесс более доступным, интересным и эффективным.

Литература

1. 2-Архитектура клиент-сервер. Режим доступа: https://studfile.net/preview/1882615/. –(Дата обращения: 15.04.2024).
2. База данных клиент-сервер. Режим доступа: https://flectone.ru/baza-danniyx-klient-server.html –(Дата обращения: 15.04.2024).
3. Простое клиент-серверное приложение для Android с нуля. Режим доступа: https://blog.illuzor.com/2018/10/03/client-server-app-from-the-scratch/?ysclid=lv6u3sf68z331321990. –(Дата обращения: 15.04.2024).
4. REST, что же ты такое? Понятное введение в технологию для ИТ-аналитиков. Режим доступа: https://habr.com/ru/articles/590679/. –(Дата обращения: 15.04.2024).

УДК 004.92

СОЗДАНИЕ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ АЛГОРИТМА ГЕНЕРАЦИИ ШУМА ПЕРЛИНА

Д. О. Квасов, О. А. Медведева

Воронежский государственный университет

Введение

Развитие современных компьютерных систем дает возможность использовать все больше и больше ресурсов для создания и демонстрации визуальной информации, что дает возможность увеличивать реалистичность объектов и повышает разнообразие вариантов итогового изображения. Одним из инструментов при создании подобного контента является процедурная генерация, которая позволяет не хранить огромные объемы данных, а воспроизводить их с помощью алгоритмов генерации по мере необходимости. Большую популярность в игровой индустрии получил алгоритм шума, разработанный Кимом Перлином.

Этот алгоритм применяют при генерации различных ландшафтов, рельефов и текстур, что говорит об универсальности метода. В данной статье будет представлено исследование зависимости визуальных характеристик полученных изображений от параметров, используемых при генерации шума.

Для создания шума на исходное изображение накладывается сетка, для каждого узла которой определяется случайный градиент. Далее значение шума рассчитывается для внутренних точек. Для этого определяются векторы от 4 ближайших вершин сетки до точки, в которой рассчитывается шум, и вычисляются скалярные произведения векторов и соответствующих градиентов. Итоговый результат значения шума получается интерполяцией полученных скалярных произведений.

1. Алгоритм генерации шума Перлина

Рассмотрим процедуру генерации шума. С этой целью сначала рассмотрим алгоритм для вычисления градиентов.

Алгоритм 1. Вычисление градиентов для узлов сетки.

Создаем неупорядоченный набор значений G от 0 до 255, где каждое число встречается единожды. Порядок значений не изменяется для всего алгоритма генерации шума.

Для каждой вершины сетки с целочисленными координатами $d(i, j)$ определяется градиент, $g_{ij} = g(p_{ij}),\ i, j = \overline{0,7},$ где

$$g_{ij} = \begin{cases} (0,1), & при\ p_{ij} = 0, \\ (1,0), & при\ p_{ij} = 0, \\ (-1,0), & при\ p_{ij} = 0, \\ (0,-1), & при\ p_{ij} = 0. \end{cases}$$

Здесь p_{ij} определяет вектор, соответствующий узлу сетки, и вычисляется по формуле:

$$p_{ij} = G_{n(d_{ij})} \bmod 4.$$

Номер элемента $n(d_{ij})$ набора G для вершины d_{ij} рассчитывается по формуле:

$$n(d_{ij}) = 16i + j.$$

Таким образом каждой точке сетки будет соответствовать определенный элемент из G. Обратим внимание на то, что порядок элементов G не изменяется и каждой точке сетки

соответствует определенное значение от 0 до 255, по которому определяется соответствующий градиент.

После задания градиентов для вершин сетки можно перейти к вычислению значений шума для каждой точки изображения.

Алгоритм 2. Вычисление значений шума для каждой точки изображения.

1) Для вычисления значения шума в точке с координатами (x, y) находятся 4 ближайшие к точке вершины сетки, образующие квадрат. Координаты вершин вычисляются по формуле:

$$d_{ij} = ([x]+i,[y]+j), \quad i, j = \overline{0,1}.$$

2) Введем локальные координаты точки в квадрате, для которой рассчитывается шум:

$$x_л = \{x\},$$
$$y_л = \{y\}.$$

3) Определим вектор от каждой вершины куба до точки:

$$v_{ij} = (x_л - i, y_л - j), \quad i, j = \overline{0,1}.$$

4) Для каждой вершины квадрата вычисляем градиент по Алгоритму 1, описанному выше:

$$g_{ij} = g_{[x]+i,[y]+j}, \quad i, j = \overline{0,1}.$$

где x, y – глобальные координаты точки, в которой вычисляется значение шума.

5) Считаем скалярное произведение векторов для каждой вершины:

$$s_{ij} = \langle g_{ij}, v_{ij} \rangle, \quad i, j = \overline{0,1}.$$

где $s = \langle g, v \rangle = x_g x_v + y_g y_v.$

6) Для каждой точки квадрата проводим интерполяцию по найденным скалярным произведениям и итоговое значение шума:

$$I(a,b,t) = a + (b-a) f(t).$$

При этом интерполяция проводится последовательно по координатам.

Интерполяция по x координате:

$$I_1 = I(s_{00}, s_{10}, x) = s_{00} + (s_{10} - s_{00}) f(x),$$
$$I_2 = I(s_{01}, s_{11}, x) = s_{01} + (s_{11} - s_{01}) f(x).$$

Интерполяция по y координате:

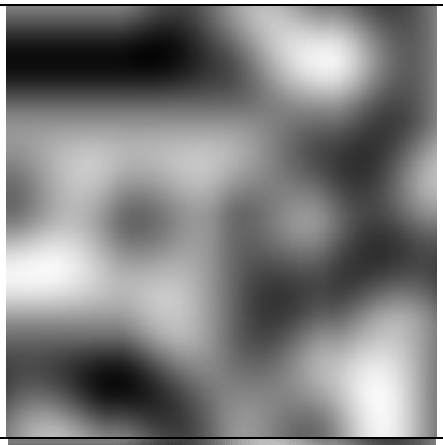
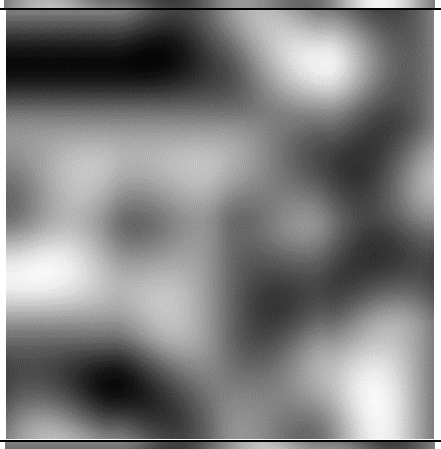
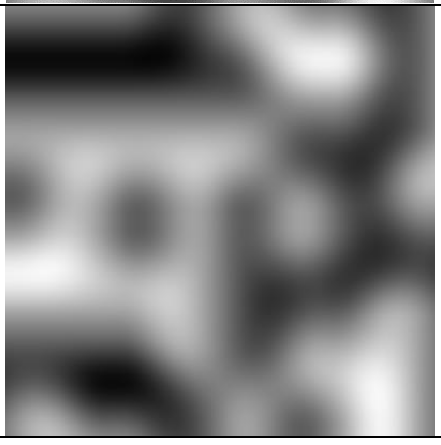
$$N = I(I_1, I_2, y) = I_1 + (I_2 - I_1) f(y) =$$
$$= (s_{00} + (s_{10} - s_{00}) f(x)) + (s_{01} + (s_{11} - s_{01}) f(x) - (s_{00} + (s_{10} - s_{00}) f(x))) f(y).$$

2.Вычислительный эксперимент

Для интерполяции можно использовать различные функции (Таблица 1). Рассмотрим некоторые из них. При генерации изображений будет применяться сетка $4 * 4$ и шаг 0.01. Таким образом, изображение шума будет иметь разрешение $400 * 400$ пикселей.

Таблица 1

Изображения шума в зависимости от функции интерполяции.

	Функция интерполяции	Итоговое изображение
1	$f(t) = t$	
2	$f(t) = (1 - \cos(t)) / 2$	
3	$f(t) = -2t^3 + 3t^2$	
4	$f(t) = 6t^5 - 15t^4 + 10t^3$	

Исследуя изображения, полученные при разных видах используемой функции, можно сделать следующие выводы:

1) Очевидно, что резкие переходы ближе к узлам сетки, образующиеся линейной интерполяцией (строка 1 Таблицы 1), отрицательно влияют на гладкость итогового изображения. Однако можем отметить, что внутри ячеек, образованных наложенной сеткой, переходы яркости плавные.

2) На изображении, полученном с помощью функции косинусной интерполяции, (строка 2 Таблицы 1) видно, что влияние клетчатой структуры становится менее заметно, однако при некоторых значениях все же выделяются пятна, образованные вдоль наложенной сетки.

3) При использовании кубической и косинусной интерполяций (строки 2 и 3 Таблицы 1) получаются визуально схожие изображения. В случае кубической интерполяции приближение к крайним значениям 0 и 1, что соответствует черному и белому шумам на изображении, задает менее интенсивный переход, чем в случае интерполяции с применением функции косинуса.

4) Избавиться от заметной клетчатой структуры с использованием вышеперечисленных функций интерполяции невозможно, влияние сетки визуально заметно. Поэтому для вычисления коэффициента интерполяции лучше использовать функцию пятой степени, предложенную Кимом Перлином:

$$f(t) = 6t^5 - 15t^4 + 10t^3$$

При этом при аргументах близких к 0 и 1 наблюдаются более плавные переходы цвета, что позволяет придать гладкость итоговому изображению (строка 4 Таблицы 1). Однако, в зависимости от целей, не исключено применение и других функций интерполяции

Исследуем влияние размерности сетки и шага, по которым строится изображение шума. Рассмотрим сетки размера 1×1, 4×4 и 8×8, а значения шага 0.0025, 0.01 и 0.25 (Таблица 2).

Таблица 2

Изображения шума в зависимости от размерности сетки и шага.

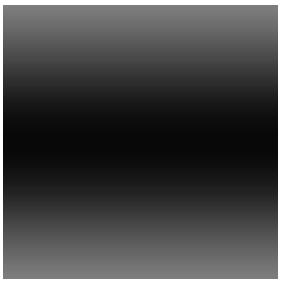
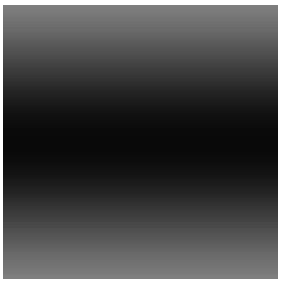
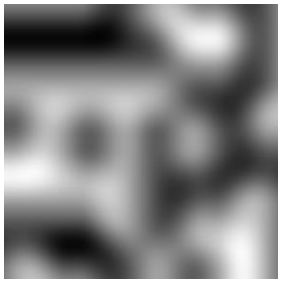
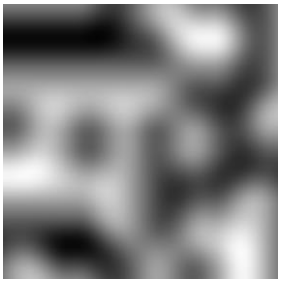
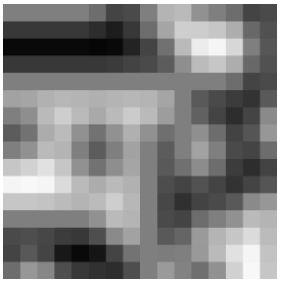
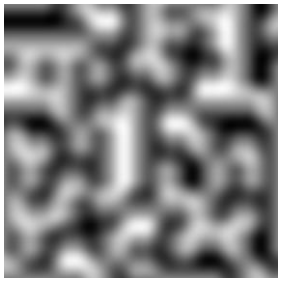
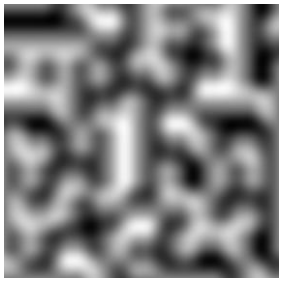
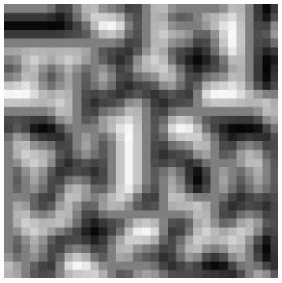
	Размер сетки	Изображение при шаге 0.0025	Изображение при шаге 0.01	Изображение при шаге 0.25
1	1×1			
2	4×4			

Таблица 2

Изображения шума в зависимости от размерности сетки и шага.

	Размер сетки	Изображение при шаге 0.0025	Изображение при шаге 0.01	Изображение при шаге 0.25
3	8×8			

Исследуя изображения, представленные в таблице 2, можно сделать следующие выводы:

1) В строке 1 Таблицы 2 изображение генерируется на основе 4 узлов сетки, что делает его изменения плавными даже при увеличении шага, это видно в 1 и 2 столбце, однако плавность теряется, когда разрешение становится сильно меньше.

2) При увеличении количества узлов изображения имеют больше перепадов (строки 2 и 3 Таблицы 2), частота которых тоже растет с увеличением накладываемой сетки.

Изображение с меньшей сеткой можно назвать приближенной частью изображения с большей сеткой. А изменение значения шага влияет на резкость перепадов. Однако обратим внимание на то, что структура шума остается неизменной. Это объясняется тем, что используется один и тот же набор значений G.

При изменении порядка значений G изменятся градиенты узлов сетки, и соответственно итоговые изображения будут отличаться по своей структуре, но на характер шума это не повлияет. Такие выводы можно сделать по рисункам, сгенерированным различными комбинациями значений G (рис. 1, 2).

Рис. 3. Изображение с помощью исходного набора G

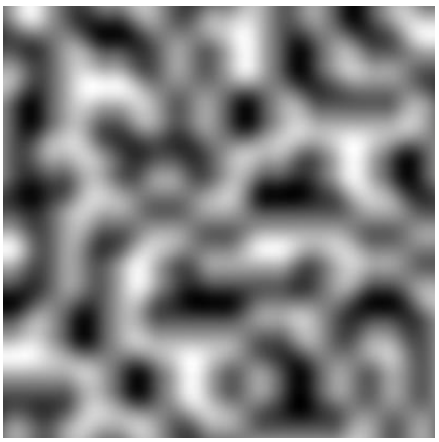


Рис. 2. Изображение с помощью измененного набора G

Заключение

Итак, алгоритм генерации шума Перлина является эффективным и гибким методом, позволяющим создавать разнообразные текстуры и изображения. Одной из особенностей этого алгоритма является возможность контролировать его параметры, что позволяет влиять на итоговый результат генерации. Изменение этих параметров позволяет создавать различные

варианты шума, что делает алгоритм Перлина удобным инструментом для создания визуального контента.

Литература

1. *Perlin, K.* Hypertexture. Computer Graphics. 1989.
2. *Perlin, K.* An Image Synthesizer. Computer Graphics. 1985.
3. *Perlin K.*, “Improved Noise”, International Conference on Computer Graphics and Interactive Techniques, Proceedings, 2002.

УДК 004.056

ОПРЕДЕЛЕНИЕ ФЕЙКОВЫХ НОВОСТЕЙ С ПОМОЩЬЮ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Е. С. Кенина, С. Ю. Болотова

Воронежский государственный университет

Введение

В современном мире дезинформация становится довольно значимой проблемой, затрагивающей различные аспекты общества. Развитие платформ социальных сетей способствует беспрецедентному ускорению распространения дезинформации. Это приводит ко многим негативным последствиям: подрыву доверия к средствам информации, поляризации общества, препятствию адекватного реагирования на новости.

Значительным подвидом дезинформации являются фейковые новости, представляющие собой ложную или искаженную информацию, созданную с целью ввести в заблуждение или манипулировать аудиторией. Это масштабное явление способно оказывать значительное влияние на общественное мнение и принятие важных решений, поскольку понять, что новость является фейковой, не всегда легко. Из-за большой вероятности человеческой ошибки для оперативной оценки и распознавания дезинформации требуется автоматизированный классификатор. Для этого могут быть использованы технологии в области машинного обучения, предоставляющие широкие возможности для анализа содержания новостных статей с целью эффективного обнаружения фейковых.

В настоящее время разработаны сложные модели для классификации фейковых новостей, позволяющие обрабатывать большие объемы данных, улавливать нюансы языка, анализировать содержание статьей и распознавать более тонкие признаки и закономерности.

В данной статье рассмотрены модели и методы классификации фейковых новостей.

1. Анализ задачи

Классификация фейковых новостей по данным категориям является сложным процессом, включающим в себя анализ источника информации, проверку фактов, оценку стиля речи и множество других факторов, что подчеркивает необходимость автоматизации данного процесса. Кроме того, существует несколько категорий, к которым можно было бы отнести ложные сведения: неверные факты, неверная интерпретация достоверной информации, псевдонаучные тексты, компиляции, состоящие в основном из чужих цитат и другие. В [1] приводится следующая классификация:

1) сатира, относящаяся к имитационным новостным программам, в которых обычно используется юмор или преувеличение, чтобы знакомить аудиторию с обновлениями новостей; пародия, которая вместо того, чтобы прямо комментировать текущие события с помощью юмора, играет на нелепости вопросов и подчеркивает их, придумывая полностью вымышленные новые истории;

2) фальсификация, которая относится к статьям, не имеющим фактической основы, но опубликованным в стиле новостных статей для создания легитимности (производитель товара часто имеет намерение ввести в заблуждение);

3) манипуляция, чаще с реальными изображениями или видео, чтобы создать ложное повествование;

4) реклама, распространяемая под видом настоящих новостных репортажей, а также выпуски со ссылкой на прессу, публикуемые как новости;

5) пропаганда, относящаяся к новостям, которые создаются политическим субъектом для влияния на общественное восприятие (открытая цель – принести пользу общественному деятелю, организации или правительству).

Однако, несмотря на то, что фейковые новости могут быть представлены в различных формах, все они направлены на манипулирование эмоциями, вследствие чего имеют общие черты и закономерности, которые могут быть выявлены с помощью технологий машинного обучения и обработки естественного языка (NLP). Это позволяет автоматически выявлять дезинформацию на основе ключевых слов, тональности текста, структуры предложений и других признаков.

2. Этапы решения задачи

Решение задачи классификации фейковых новостей можно разделить на несколько этапов.

На первом шаге необходимо получить датасет, представляющий собой всеобъемлющую подборку новостных статей, тщательно подобранных для предоставления сбалансированного и непредвзятого набора данных. Это имеет решающее значение для получения высококачественных обучающих данных и точных результатов. Для изучения фейковых новостей доступно несколько открытых наборов данных, эти наборы данных имеют существенные ограничения по размеру, категории или предвзятости. Чтобы устранить эти ограничения, можно объединить несколько существующих наборов данных, имеющих схожую структуру. Это уменьшит ограничения и предвзятость каждого набора данных.

В [2] отмечено, что:

- новостные статьи, содержащие от 450 до 550 слов, как правило, более надежны;
- общие тенденции указывают на то, что более короткие, но содержательные новости часто более правдивы;
- читаемость текста фейковых новостей хуже, чем реальных новостей;
- субъективность статей о фейковых новостях более значима, чем статей о реальных новостях;
- количество статей, представляющих реальные новости, больше, чем статей, представляющих фейковые новости.

Эти наблюдения дают ценную информацию о характеристиках фейковых и реальных новостных статей, что может сыграть важную роль в дальнейшей разработке и совершенствовании алгоритмов обнаружения фейковых новостей.

На следующем этапе происходит препроцессинг, или предварительная обработка текста новости. Здесь выполняются операции преобразования текстового заголовка новости в формат, необходимый для использования классификатора. К функциям препроцессора можно отнести:

- удаление пробелов, знаков препинания и спецсимволов с помощью регулярных выражений;
- приведение слов к нижнему регистру;
- токенизацию с целью разбиения текста на отдельные части – токены;
- удаление нерелевантных слов (стоп-слов), не играющих значительной роли;
- лемматизацию с целью приведения всех данных к нормальной словарной форме (для существительных и прилагательных это именительный падеж в единственном числе, для глаголов, причастий и деепричастий – инфинитив). Это уменьшает количество уникальных значений, что способствует улучшению результатов работы.

Поскольку алгоритмы машинного обучения не могут работать с сырым текстом, на следующем этапе полученные в результате препроцессинга данные нужно конвертировать в последовательности числовых идентификаторов каждого токена. Это называется извлечением признаков. Одним из наиболее простых и популярных способов извлечения признаков при работе с текстом является «мешок слов». Он описывает вхождения каждого слова в текст.

Чтобы использовать модель, необходимо определить словарь известных токенов и выбрать степень присутствия известных слов. При этом любая информация о порядке или структуре слов игнорируется. Иными словами, для модели важно знать только то, встречается ли знакомое слово в документе, но неважно, где конкретно. Сложность этой модели состоит в том, как определить словарь и как подсчитать вхождение слов.

Можно использовать другой способ создания словаря с помощью сгруппированных слов. Это изменит размер словаря и даст мешку слов больше деталей о документе. Такой подход называется «N-грамма». N-грамма – это последовательность каких-либо сущностей (слов, букв, чисел, цифр и т. д.). В контексте языковых корпусов, под N-граммой обычно понимают последовательность слов. Цифра N обозначает, сколько сгруппированных слов входит в N-грамму. В модель попадают не все возможные N-граммы, а только те, что фигурируют в корпусе.

На заключительном этапе для обнаружения характерных признаков каждой из категорий новостей и для формирования классификационной оценки отнесения новости к одной из категорий с определенным уровнем достоверности результата необходимо обучить нейронную сеть.

В основе классификатора новостных заголовков могут лежать следующие популярные архитектуры нейронных сетей, которые хорошо показали себя в задачах NLP и рекомендуются к использованию [3].

Сверточная нейронная сеть (Convolutional neural network, CNN) содержит один или более объединенных или соединенных сверточных слоев. CNN использует вариацию многослойного перцептрона. Сверточные слои используют операцию свертки для входных данных и передают результат в следующий слой. Эта операция позволяет сети быть глубже с меньшим количеством параметров. Они могут быть эффективными при обработке коротких текстов, таких как анализ тональности. В статье [4] автор описывает процесс и результаты задач классификации текста с помощью CNN. В работе представлена модель на основе word2vec, которая проводит эксперименты и демонстрирует хорошие результаты.

Рекуррентная нейронная сеть **(**RNN) - это тип нейросетей, который широко используется в NLP. Они способны учитывать последовательность слов в тексте и учиться на

основе контекста. Однако у RNN есть ограничения в обработке длинных последовательностей из-за проблемы исчезающего градиента.

Сеть долгой краткосрочной памяти (Long Short-Term Memory, LSTM) и *реккурентная нейронная сеть с управляемыми блоками* (Gated Recurrent Unit, GRU) – модификации RNN, спроектированные для решения проблемы исчезающего градиента и широко используются в NLP для анализа длинных последовательностей. LSTM создана для более точного моделирования временных последовательностей и их долгосрочных зависимостей, чем традиционная рекуррентная сеть. В работе [5] представлена модель для автоматической морфологической разметки, которая показывает точность 97.4 %. GRU является более легкой и вычислительно эффективной по сравнению с LSTM. Ее главное преимущество перед LSTM заключается в более низкой сложности и меньшем количестве параметров, что может быть важно при работе с ограниченными вычислительными ресурсами. Однако, стоит отметить, что LSTM всё равно остается более мощным в решении некоторых сложных задач, требующих учета долгосрочных зависимостей.

Двунаправленный LSTM, или BiLSTM, – это модель последовательности, которая содержит два уровня LSTM, один из которых предназначен для обработки входных данных в прямом направлении, а другой – для обработки в обратном направлении. Он обычно используется в задачах, связанных с NLP. Двунаправленный LSTM помогает машине лучше понимать эту взаимосвязь, чем однонаправленный LSTM. Это обеспечивает дополнительный контекст для сети и приводит к более быстрому и даже полному изучению проблемы, что важно для точной классификации фейковых новостей [6].

Для улучшения точности и сокращения времени обучения каждая из нейросетевых моделей может использовать заранее предобученную на русском корпусе слов модель BERT [7].

BERT – двунаправленная модель с transformer-архитектурой, заменившая собой последовательные по природе рекуррентные нейронные сети (LSTM и GRU), с более быстрым подходом на основе механизма внимания. Модель также предобучена на двух задачах без учителя – моделирование языковых масок и предсказание следующего предложения. Это позволяет использовать предобученную модель BERT, настраивая её под конкретные задачи, такие как определение эмоциональной окраски текста, вопросно-ответные системы и многие другие. В работе [8] приводятся достаточно высокие результаты трех обученных гибридных нейросетевых моделей – BERT-CNN, BERT-GRU, BERT-LSTM для идентификации фейковых новостей.

Полученные векторные представления передаются в нейросетевые слои в зависимости от конкретной модели для обнаружения характерных признаков. Таким образом, модели используют семантический анализ для выявления фейковой новости. Далее для обработки полученных признаков может быть использован полносвязный нейросетевой слой для улучшения точности распознавания. Кроме того, поскольку нейросетевые модели должны относить новости к одной из многих категорий, ключевой задачей является выполнение многоклассовой классификации. Для этого в конце каждой модели можно использовать полносвязный нейросетевой слой, содержащий столько нейронов, сколько существует классов, и каждый нейрон генерирует оценку вероятности для определенного класса в диапазоне от 0 до 1.

Заключение

В работе представлен обзор решений в области идентификации фейковых новостей, рассмотрены модели глубокого обучения, используемые для задачи классификации. В дальнейшем планируется практическая реализация предложенного решения и проведение статистической оценки полученных результатов с целью сравнения описанных нейросетевых архитектур.

Литература

1. Edson C., Tandoc Jr., Lim Z., Ling R. Fake News // Digital Journalism. – 2018. – № 6 (2). – С. 137–153.
2. Падалко Х., Хомко В., Чумаченко Д. (2024) Новый подход к классификации фейковых новостей с использованием моделей глубокого обучения на основе LSTM. Фронт. Большие данные 6:1320800. doi: 10.3389/fdata.2023.1320800
3. 7 архитектур нейронных сетей для решения задач NLP. – Режим доступа: https://neurohive.io/ru/osnovy-data-science/7-arhitektur-nejronnyh-setej-nlp/
4. Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.
5. Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. – Режим доступа: https://arxiv.org/pdf/1510.06168.pdf
6. Zeng, Z.-Y.; Lin, J.-J.; Chen, M.-S.; Chen, M.-H.; Lan, Y.-Q.; Liu, J.-L. A Review Structure Based Ensemble Model for Deceptive Review Spam // Information. 2019, №10 (7), 243. https://doi.org/10.3390/info10070243
7. BERT в двух словах: Инновационная языковая модель для NLP. – Режим доступа: https://habr.com/ru/companies/otus/articles/702838/
8. Тумбинская М.В., Галиев Р.А. Идентификация фейк-новостей с помощью веб-ресурса на основе нейронных сетей // Программные продукты и системы. 2023. Т. 36. № 4. С. 590–599. doi: 10.15827/0236-235X.142.590-599

УДК 004.93

ОТСЛЕЖИВАНИЕ ЛИЦЕВЫХ ПРИЗНАКОВ МЕТОДОМ КООРДИНАТНОЙ РЕГРЕСИИ НА ОСНОВЕ ГЛУБОКОГО ОБУЧЕНИЯ

Н. С. Кириллов, М. И. Быкова

Воронежский государственный университет

Введение

Создание трехмерной модели лица актера для каждого кадра видеозаписи актерской игры является сложным и многоступенчатым процессом. Одним из наиболее трудоемких и длительных этапов этого процесса, требующих человеческого вмешательства, является разметка ключевых лицевых признаков (в частности, контуров глаз, губ, бровей, носогубной складки и т.д.) на кадрах съемки актера. Эта разметка вместе с другими данными впоследствии используется для получения трехмерной модели лица.

Для достижения высокого качества лицевой анимации разметка должна соответствовать реальному положению лицевых признаков на изображении с приемлемой точностью. Кроме того, она должна сохранять согласованность как между различными кадрами одной записи, так и между различными записями, так как дрожания, скачки и прочие шумы мгновенно становятся заметны зрителю при проигрывании анимации и пагубно влияют на ее восприятие. Это требование также осложняет ручную разметку, в особенности, если ей занимается команда из нескольких человек.

1. Постановка задачи

Необходимо разработать и программно реализовать алгоритм отслеживания ключевых лицевых признаков на последовательности изображений с помощью глубоких нейронных сетей.

Обучающие данные состоят из пар изображение-разметка. Изображения извлекаются как кадры из видеозаписей, снятых на камеру шлема, который закреплен на голове актера во время съемки. Изображения черно-белые. Разметка представляет собой упорядоченные наборы точек, принадлежащие лицевым контурам. В данной статье рассматриваются следующие контуры:

– внешняя граница губы (для верхней и нижней губы);
– граница между внутренней частью губы и ротовой полостью, либо зубами (для верхней и нижней губы).

На рис. 1 представлен пример последовательности размеченных вручную кадров.

Решение поставленной задачи осложняется тем, что изображения могут содержать дефекты, такие как шум, потери при сжатии видео, расфокусировка камеры, размытие в движении, тряска и изменение положения шлема на голове актера (например, в активных сценах), изменение места съемки, фона или освещения. Эти дефекты должны обрабатываться алгоритмом с максимально возможной точностью и согласованностью между кадрами в зависимости от дефекта.

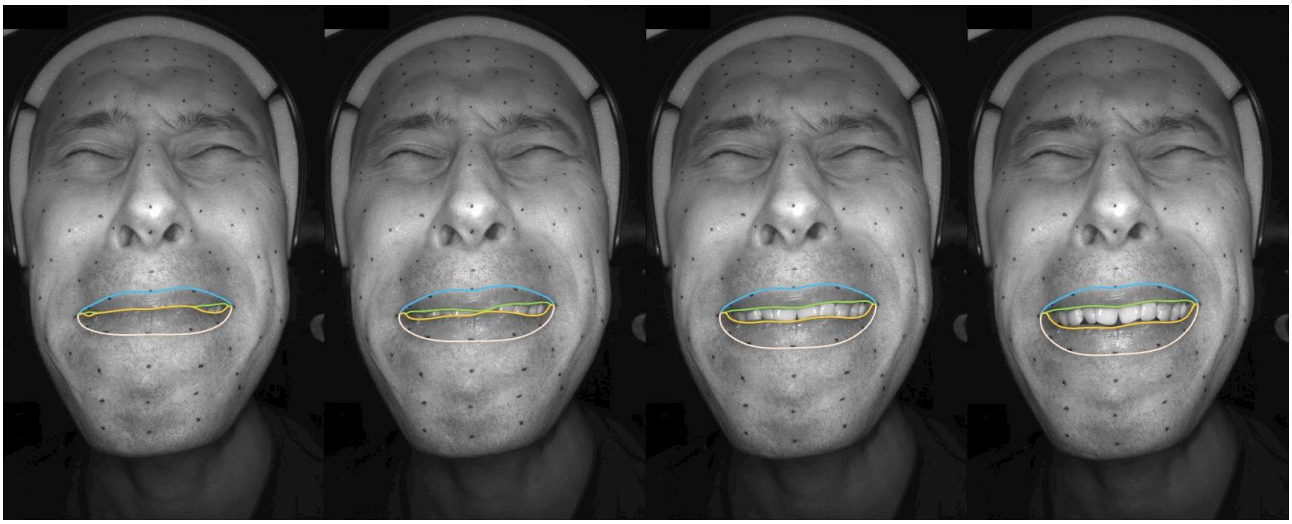


Рис. 1. Последовательность изображений с ручной разметкой

2. Анализ задачи

В данном пункте рассматриваются существующие варианты решения задачи отслеживания лицевых признаков, выделяются возможности и недостатки классических методов компьютерного зрения, а также описываются преимущества использования методов глубокого обучения.

2.1. Алгоритмы оценки оптического потока

Отслеживание точек на последовательности изображений является одной из известных задач классического компьютерного зрения. В частности, она может быть представлена как задача оценки оптического потока [3]. Задача оценки оптического потока заключается в предсказании движения пикселей изображения между двумя последовательными кадрами с учетом структуры объектов на изображении. Эта задача не имеет решения в общем случае, и алгоритмы оценки оптического потока вводят различные дополнительные ограничения, которые позволяют решить задачу. При этом основным является ограничение на постоянную яркость пикселей, относящихся к одним и тем же объектам на последовательных изображениях.

Так, например, алгоритм Лукаса-Канаде [1] использует предположение о том, что перемещение объектов между кадрами небольшое и близкое (почти одинаковое) для всех пикселей в некотором регионе вокруг рассматриваемой точки. Это позволяет составить систему уравнений оптического потока для всех пикселей этого региона и получить решение (вектор перемещения для целого региона) методом наименьших квадратов. Этот алгоритм является локальным (для оценки оптического потока используется только регион вокруг каждой точки) и разреженным (задача оценки оптического потока может решаться не для всех пикселей изображения, а для набора ключевых точек). Недостатком такого алгоритма является невозможность решить задачу для регионов, не содержащих достаточных признаков, которые позволяют однозначно определить, куда должен переместиться этот регион.

В свою очередь, алгоритм Хорна-Шунка [2] вводит глобальное ограничение на гладкость перемещений соседних пикселей. Таким образом, он позволяет заполнить значения перемещений для пикселей в областях объектов с однородной текстурой на основе перемещений граничных областей, ценой необходимости решать более сложную задачу для всех пикселей изображения.

Однако методы оценки оптического потока обладают большим недостатком, который не позволяет применить их в задаче отслеживания лицевых признаков, — это сложность в обработке появления и исчезновения объектов, что критично для отслеживания внутренних контуров губ (открытие и закрытие рта, появление зубов, десен или языка в кадре). Внутренние влажные поверхности губ и ротовая полость также могут вызывать сильные блики от окружающего света, что нарушает ограничение на постоянную яркость совпадающих объектов. Кроме того, отслеживание контуров губ осложняется широким спектром возможных деформаций, таких как сжатие и втягивание губ, открытие рта, поворот в одну из сторон, и др.

Сложные случаи для разметки изображены на рис. 2.

Рис. 2. Сложные выражения лица для разметки

2.2. Модели активного внешнего вида

Помимо алгоритмов оценки оптического потока среди решений, не полагающихся на машинное обучение, можно также выделить алгоритмы на основе численных оптимизаций и статистических моделей, таких как модели активного внешнего вида [6]. Эти модели основываются на подстановке набора ключевых точек под новое изображение согласно шаблонам — эталонным соответствиям между изображениями и наборами ключевых точек (то есть, изображениям с готовой разметкой).

Наличие эталонных примеров дает возможность корректно обрабатывать открытия и закрытия рта (и другие сложные ситуации) и, таким образом, позволяет применять эти алгоритмы для автоматической разметки лицевых контуров. Однако на практике необходимость в высокой точности разметки, меняющиеся условия съемки, а также наличие различных дефектов на изображениях приводит к необходимости размечать вручную большое количество шаблонов для каждой видеозаписи, чтобы получить удовлетворительный результат.

2.3. Методы глубокого обучения

Методы глубокого обучения основываются на применении многослойных нейронных сетей для решения различных задач. Процесс обучения представляет собой изменение параметров модели, которое приводит к наименьшему значению ошибки на обучающем наборе данных. Для этого применяется алгоритм обратного распространения ошибки [4], который заключается в вычислении градиентов функции ошибки по параметрам модели, и обновлении параметров методом градиентного спуска (и его модификациями).

Преимущество глубоких нейронных сетей заключается в способности автоматически извлекать из входных данных признаки и закономерности, релевантные для задачи, для которой используется модель. Большое количество достаточно репрезентативных обучающих данных позволяет модели сформировать обобщенное внутреннее представление, которое может давать удовлетворительную точность на тестовых данных.

2.4. Анализ существующих подходов

1. Алгоритмы оценки оптического потока.

Достоинства: отсутствие необходимости ручной разметки.

Недостатки: невозможность обработать появление зубов и языка; сложность в обработке больших деформаций; чувствительность к изменению окружения.

2. Модели активного внешнего вида.

Достоинства: возможность корректно обрабатывать открытие рта и большие деформации.

Недостатки: необходимость разметки большого количества эталонных примеров; чувствительность к изменению окружения.

3. Методы глубокого обучения.

Достоинства: возможность корректно обрабатывать открытие рта и большие деформации; устойчивость к изменению окружения.

Недостатки: необходимость большого количества чистых и разнообразных обучающих данных для получения качественной автоматической разметки.

3. Реализация

В данной работе был реализован алгоритм координатной регрессии на основе глубокого обучения. Этот алгоритм не использует геометрическую информацию (например, о связности между ключевыми точками), не вводит явные ограничения (например, на гладкость между соседями) и не требует ручной разметки эталонных примеров на этапе предсказания. Для реализации алгоритма использовался язык *Python* и библиотека глубокого обучения *Pytorch*.

На вход алгоритма координатной регрессии поступает изображение фиксированного размера, на котором в итоге будет получена разметка заранее определенных лицевых контуров. Алгоритм включает следующие шаги:

1. Извлечение признаков заранее определенных лицевых контуров из изображения с помощью обученной глубокой сверточной сети.
2. Преобразование полученных признаков в вектор.
3. Преобразование вектора признаков в вектор координат точек разметки с помощью обученной полносвязной сети.

Для извлечения признаков в алгоритме координатной регрессии используются модификации нейронных сетей, которые были разработаны для классификации изображений. Одной из наиболее известных моделей является *ResNet* [5]. Ее схема изображена на рис. 3.

Архитектура сетей для извлечения признаков строится таким образом, чтобы последовательно уменьшать пространственное разрешение изображения, извлекая из него на каждом слое более глобальные признаки, описывающие положение и форму контуров губ и не зависящие от фона, яркости и внешности остальной части лица. Такой подход обеспечивает устойчивость к смене освещения сцены и внешнего вида актера, дефектам изображения и видеозаписи и даже инородным объектам в кадре.

Использование полносвязной сети для преобразования вектора признаков в вектор координат, в свою очередь, обеспечивает зависимость между всеми ключевыми точками, так как координаты каждой точки получаются из взвешенной суммы всех признаков. При достаточном количестве и разнообразии качественных обучающих данных, это обеспечивает гладкую и правдоподобную форму контура, который получается в результате работы алгоритма, и предотвращает выбросы или шум в положении отдельных точек.

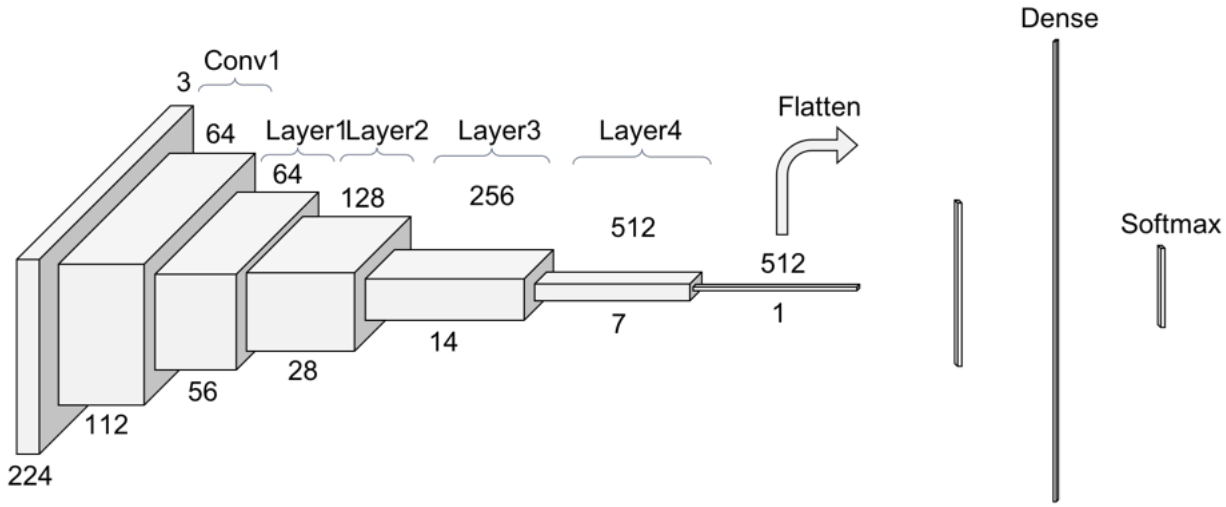


Рис. 3. Глубокая сверточная сеть для классификации изображений

4. Тестирование

Данный пункт посвящен демонстрации результатов работы обученной нейронной сети на тестовой выборке. Размер обучающей выборки составляет около 10000 примеров, размер тестовой выборки составляет около 2000 примеров. На рис. 4–6 изображены примеры полученной разметки для некоторых экстремальных выражений лица актера в сравнении с ручной разметкой (слева — автоматическая, справа — ручная).

На рис. 4 изображена сложная деформация контура губ, которая включает в себя поворот в трех измерениях.

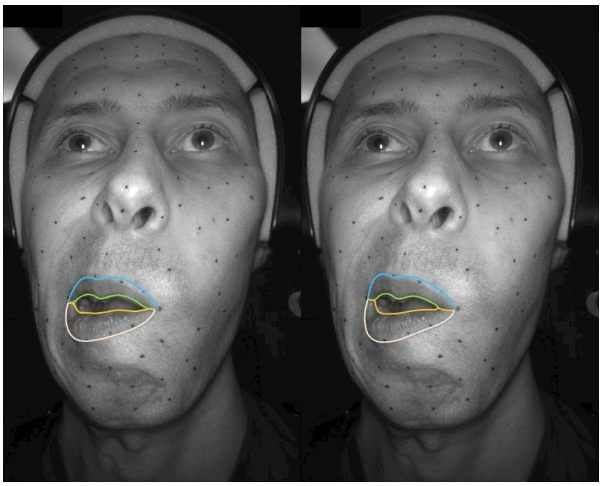


Рис. 4. Полученная разметка (слева) в сравнении с ручной (справа)

На рис. 5 показан пример с появлением зубов на изображении.

Рис. 5. Полученная разметка (слева) в сравнении с ручной (справа)

На рис. 6 изображено открытие рта, создающее сильную деформацию контура губ. На изображении также присутствуют зубы и язык.

Рис. 6. Полученная разметка (слева) в сравнении с ручной (справа)

На рис. 7–8 показана согласованность результатов работы алгоритма для соседних кадров. Внешние контуры губ сохраняют свое положение относительно маркеров, расставленных на губах актера во время съемки.

Рис. 7 демонстрирует устойчивость алгоритма к появлению зубов и языка при открытии рта.

Рис. 7. Полученная разметка для соседних кадров

Рис. 8 демонстрирует устойчивость алгоритма при сильных деформациях (широкое открытие рта), а также при появляющихся бликах на губах и языке.

Рис. 8. Полученная разметка для соседних кадров

Заключение

В результате проделанной работы был разработан алгоритм получения ключевых лицевых признаков (контуров губ) из изображения актера на основе координатной регрессии с использованием глубоких нейронных сетей.

Подход был успешно протестирован на новом наборе данных, отсутствующих в обучающей выборке. Был получен удовлетворительный, сравнимый по качеству с ручной разметкой результат работы алгоритма на изображениях различных эмоций актера. Была также проверена согласованность автоматической разметки на соседних кадрах.

Литература

1. B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pp 121–130, 1981.
2. B.K.P. Horn and B.G. Schunck. Determining optical flow. Artificial Intelligence, vol 17, pp 185–203, 1981.
3. David J. Fleet, Yair Weiss. Optical Flow Estimation. In N. Paragios, Y. Chen, and O. Faugeras (editors). Handbook of Mathematical Models in Computer Vision. Springer. pp. 237–257, 2006.
4. Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press. – URL: https://www.deeplearningbook.org (Дата обращения: 10.04.2024).
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. – URL: https://arxiv.org/abs/1512.03385 (Дата обращения: 10.04.2024).
6. Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active Appearance Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 6, june 2001.

УДК 53.531

О ФИЗИЧЕСКОЙ МОДЕЛИ ВЗАИМОДЕЙСТВИЯ ШАРА С СУКНОМ ДЛЯ СОЗДАНИЯ СИМУЛЯТОРА НАСТОЛЬНОЙ ИГРЫ «БИЛЬЯРД»

М. С. Киселев

Воронежский государственный университет

Введение

Бильярд – собирательное имя нескольких настольных игр. Любой бильярд – это настольная игра, в которой участники по очереди бьют особой деревянной палкой – кием – по шарам, обычно стараясь забить их в лузы. Побеждает тот, кто первым наберет нужное количество очков или забитых шаров [3]. По своей сути бильярд – игра, полностью основанная на физике взаимодействия различных компонент игры. Поэтому, для создания симулятора необходимо понимать физику игры.

Бильярд – одна из самых старых настольных игр. Первые упоминания и изображения бильярда датируются ещё XVII-м веком. Игра изучалась многие годы, и это позволяет нам создать реалистичную физическую модель для создания симулятора.

1. Описание задачи

Необходимо описать физику игры в бильярд. Для этого необходимо начать с описания взаимодействия шара с главным компонентом игры: сукном.

Взаимодействие шара с сукном подразумевает нахождение шара в состоянии покоя и движение шара по сукну. Движение шара по сукну, в свою очередь, можно разделить на 4 компонента: покой, вращение, качение и скольжение. Необходимо учесть каждый компонент движения шара.

2. Взаимодействие шар-сукно и состояние покоя

Очевидно, что шар трётся об сукно, тем самым замедляя своё движение. Однако, в зависимости от вращения шара, это трение может так же менять траекторию движения шара. Для начала всё же необходимо описать физику шара без учёта вращения.

На рис. 1 изображен движущийся бильярдный шар и силы, действующие на него.

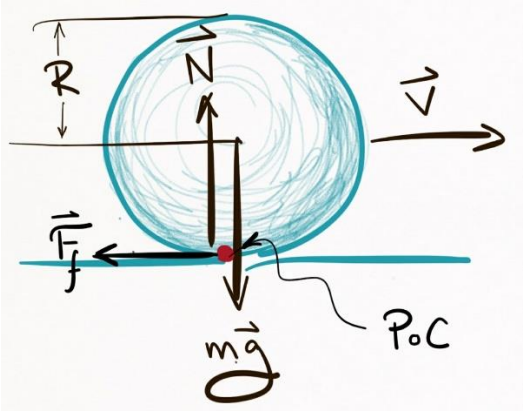


Рис. 1. Бильярдный шар в движении и действующие силы

На данном рисунке изображены скорость шара $\vec{v}$, его масса m и радиус R. Изображены так же гравитационная постоянная $\vec{g}$ и сила нормальной реакции $\vec{N}$.

Пусть у шара изначально нет вращения. Тогда шар двигается только горизонтально. На шар действует сила тяжести $m\vec{g}$, притягивая шар к столу. Существует так же обратная сила $\vec{N}$, без которой шар упадёт сквозь стол. То есть одна сила направленна вертикально вниз, а другая вертикально вверх. Это и создает силу трения шара о сукно. Когда шар движется, на него действует сила $\vec{F}_f$, направленная противоположно движению шара.

Сила $\vec{F}_f$ действует только на нижнюю точку шара (назовём её PoC). Это создаёт крутящий момент, так как замедляется только та самая нижняя точка шара, а верхняя движется по той же скорости.

Теперь рассмотрим шар с производным вращением. Тогда у шара может быть угловая скорость $\vec{\omega}$. Угловая скорость – физическая величина, характеризующая быстроту и направление вращения тела относительно оси вращения. Пусть шар может быть в произвольном состоянии (то есть у шара может быть произвольная скорость $\vec{v}$, угловая скорость $\vec{\omega}$ и смещение относительно какой-то точки $\vec{r}$. Эти три вектора полностью характеризуют текущее состояние шара). Также допустим, что у шара и сукна может быть только одна точка соприкосновения PoC.

То есть зная $\vec{v}$, $\vec{\omega}$, и $\vec{r}$, мы знаем текущее состояние шара. Модифицируя данные векторы $\vec{v}_0$, $\vec{\omega}_0$ и $\vec{r}_0$, мы меняем состояние шара.

Если шар в состоянии покоя, то он остаётся там же, где и был и на него не действуют ни линейная, ни угловая скорость.

Пусть t – время. Валидность описывает промежуток времени, при котором данные равенства действительны. Тогда:

Смещение:

$$\vec{r}(t) = \vec{r}_0 \tag{1}$$

Скорость:

$$\vec{v}(t) = \vec{0} \tag{2}$$

Угловая скорость:

$$\vec{\omega}(t) = \vec{0} \tag{3}$$

Валидность:

$$0 \le t < \infty \tag{4}$$

Важно помнить, что (1), (2) и (3) – векторные равенства. То есть каждое из них можно разбить на 3 отдельных скалярных равенства, по одному для каждого измерения. То есть (3), например, может быть записано как:

$$\begin{aligned} \omega_x(t) &= 0 \\ \omega_y(t) &= 0 \\ \omega_z(t) &= 0 \end{aligned} \tag{5}.$$

3. Вращение

Вращение – одно из частых состояний шара при игре в бильярд. При вращении у шара отсутствует линейный момент, но он вращается на одном месте.

Получается, у шара отсутствует $\vec{v}$ и он остаётся на месте $\vec{r}(t) = \vec{r}_0$. Однако, так как шар вращается, у него есть угловая скорость. Можно заметить, что шар вращается вокруг оси z, относительно системы координат, показанной на рис. 2.

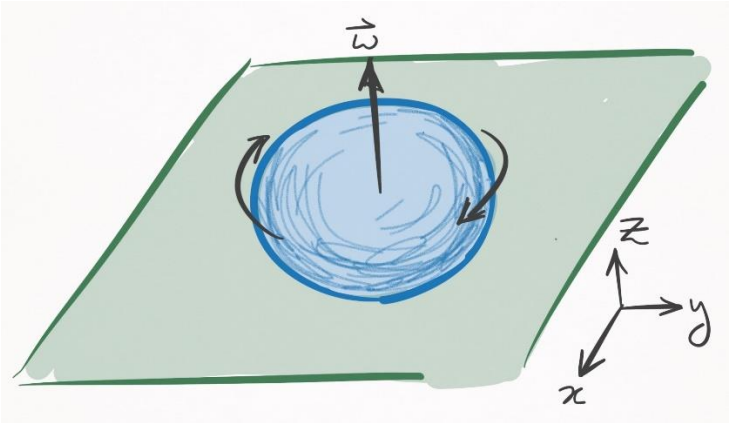


Рис. 2. Вращение шара вокруг оси z

Такое вращение не случайно. Это, можно сказать, условие состояния, так как если у шара будет угловая скорость по x или y, то будет существовать трение с сукном, которое, в свою очередь создаст линейную скорость. Но это противоречит тому, что шар вращается на месте. Так что $\omega_x(t)$ и $\omega_y(t)$ равны нулю.

До этого мы допускали, что шар и сукно имеют одну точку соприкосновения PoC. Это, однако, не может быть правдой, так как в таком случае у шара будет отсутствовать трение с сукном при вращении, так как относительная скорость между шаром и сукном будет равна 0. Фактически же шар замедляется, и как раз-таки из-за трения с сукном (и воздухом. Но его мы в данной модели игнорируем). Это происходит, потому что на самом деле сукно и шар взаимодействуют не в одной точке, а в какой-то области. Однако, в данной модели, вместо расчёта данной области, введём переменную μ_{sp}. Пусть это будет коэффициент силы трения между сукном и шаром при вращении вокруг оси z. Теперь можно описать все три векторных уравнения, необходимых для описания состояния шара при вращении.

Смещение:

$$\vec{r}(t) = \vec{r}_0 \tag{6}$$

Скорость:

$$\vec{v}(t) = \vec{0} \tag{7}$$

Угловая скорость:

$$\begin{aligned} \omega_x(t) &= 0 \\ \omega_y(t) &= 0 \\ \omega_z(t) &= \omega_{0z} - \frac{5\mu_{sp} g}{2R} t \end{aligned} \tag{8}$$

Валидность:

$$0 \le t \le \frac{2R}{5\mu_{sp} g} \omega_{0z} \tag{9}$$

Здесь R – радиус шара. Равенства валидны, пока шар не остановится. То есть $\omega_z(t) = 0$.

Это происходит при $t = (2R\omega_{0z}) / (5\mu_{sp} g)$.

4. Качение

Под качением подразумевается движение шара при сцеплении с сукном. То есть при таком движении нет скольжения.

Пусть шар катится в направлении x. На рис. 3 изобразим такое движение.

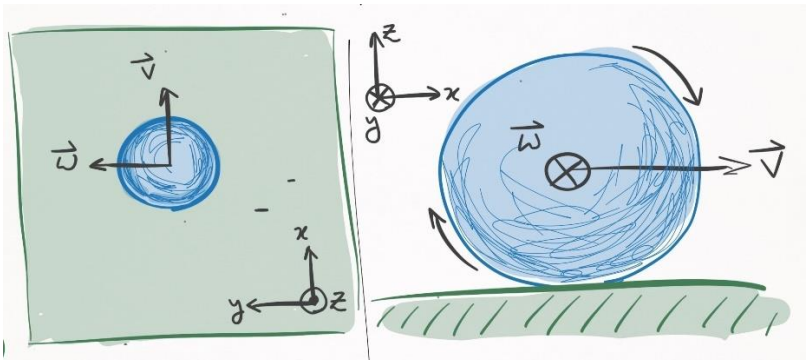


Рис. 3. Качение шара в направлении x

Слева показан вид сверху, а справа показан вид в профиль. Угловая скорость направлена по оси y.

Такой шар будет двигаться по прямой, пока не остановится. При движении он будет замедляться силой трения, что подразумевает линейное замедление с течением времени.

Пусть μ_r – коэффициент трения качения и $\hat{v}_0$ – вектор, указывающий направление, по которому движется шар.

$$\vec{v}(t) = \vec{v}_0 - \mu_r g t \hat{v}_0 \tag{10}$$

Шар так же будет смещаться в пространстве с течением времени.

$$\vec{r}(t) = \vec{r}_0 + \vec{v}_0 t - \frac{1}{2}\mu_r g t^2 \hat{v}_0 \tag{11}$$

Шар катится, если за один оборот вокруг оси y он проходит расстояние равное одной длине окружности ($2\pi R$). Иначе шар скользит.

Запишем далее 3 уравнения, необходимых для описания состояния шара при кручении. Записаны они будут относительно начального положения шара.

Смещение:

$$\begin{aligned} r_x(t) &= v_0 t - \frac{1}{2}\mu_r g t^2 \\ r_y(t) &= 0 \\ r_z(t) &= 0 \end{aligned} \tag{12}$$

Скорость:

$$\begin{aligned} v_x(t) &= v_0 - \mu_r g t \\ v_y(t) &= 0 \\ v_z(t) &= 0 \end{aligned} \tag{13}$$

Угловая скорость:

$$\omega_x(t) = 0$$
$$\omega_y(t) = \frac{v_x(t)}{R} \qquad (14)$$
$$\omega_z(t) = \omega_{0z} - \frac{5\mu_{sp} g}{2R} t$$

Валидность:

$0 \le t \le \frac{|\vec{v}_0|}{\mu_r g}$. Если $\frac{2R}{5\mu_{sp} g}\omega_{0z} < \frac{|\vec{v}_0|}{\mu_r g}$, то $\omega_z(t) = 0$ для $t > \frac{2R}{5\mu_{sp} g}\omega_{0z}$.

5. Скольжение

Качение можно описать как состояние, при котором относительная скорость $\vec{u}(t)$ между шаром и тканью в точке PoC равна 0. Зависит от двух компонентов: линейная скорость шара и скорость между шаром и сукном, существующее благодаря вращению шара. Их сумма и является $\vec{u}(t)$.

$$\vec{u}(t) = \vec{v}(t) + R\hat{k} \times \vec{\omega}(t) \qquad (15)$$

Если же $\vec{u}(t) \neq 0$, то имеет место скольжение.

Скольжение от качения отделяет то, что $\vec{u}(t)$ может указывать в любое управление на координатной плоскости xy. При этом сила трения будет направлена в противоположном направлении. Это показано на рис. 4.

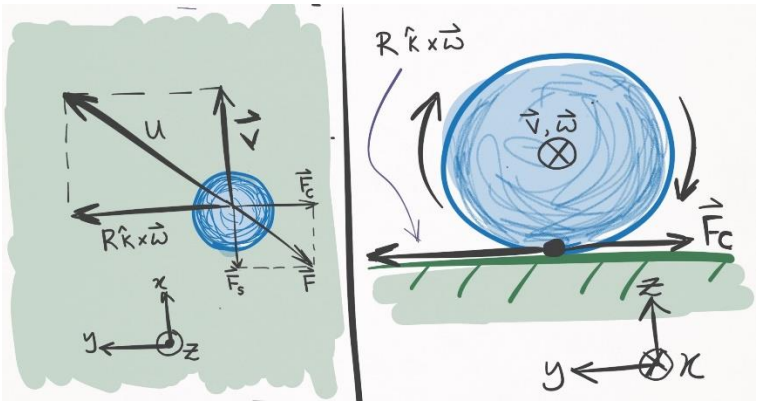


Рис. 4. Скольжение шара

На данном рисунке шар движется в направлении x. Угловая скорость также направлена в данном направлении. В данном случае шар будет понемногу откланяться вправо из-за силы трения, направленной противоположно $\vec{u}(t)$.

Теперь необходимо описать равенства необходимые для описания состояния шара при кручении. Записаны они будут относительно начального положения шара.

Смещение:

$$\vec{r}(t) = \vec{v}_0 t - \frac{1}{2}\mu_s g t^2 \hat{u}_0 \qquad (16)$$

Скорость:

$$\vec{v}(t) = \vec{v}_0 - \mu_s g t \hat{u}_0 \tag{17}$$

Угловая скорость:

$$\begin{aligned} \vec{\omega}_{xy}(t) &= \vec{\omega}_{0xy} - \frac{5\mu_{sp} g}{2R} t(\hat{k} \times \vec{u}_0) \\ \omega_z(t) &= \omega_{0z} - \frac{5\mu_{sp} g}{2R} t \end{aligned} \tag{18}$$

Валидность:

$0 \le t \le \frac{2}{7}\frac{u_0}{\mu_s g}$. Если $\frac{2R}{5\mu_{sp} g}\omega_{0z} < \frac{2}{7}\frac{u_0}{\mu_s g}$, то $\omega_z(t) = 0$ для $t > \frac{2R}{5\mu_{sp} g}\omega_{0z}$.

Заключение

В результате работы были рассмотрены возможности по созданию физической модели взаимодействия шара с сукном при создании симулятора настольной бильярдной игры. Были рассмотрены четыре основных случая взаимодействия.

Литература

1. *Вейланд С. Мерлоу*. The Physics of Pocket Billiards. / Вейланд С. Мэрлоу – 291 с.
2. *Хэн*. Dynamics in carom and three cushion billiards / Хэн // Journal of Mechanical Science and Technology (April 2005) – 976-986 с.
3. Киселев М.С. О создании симулятора настольной игры «бильярд». / Киселев М.С. // Сборник трудов Международной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 4-6 декабря 2023 г.). – Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2022. – С. 858-863 с.

УДК: 519.72

МОДИФИКАЦИЯ РАССТОЯНИЯ ЛЕВЕНШТЕЙНА ДЛЯ ПОИСКА В КАТАЛОГЕ ЛЕКАРСТВЕННЫХ ПРЕПАРАТОВ

Г. Д. Коваль
Воронежский государственный университет

Введение

Среди лекарственных препаратов часто встречается много схожих по названию. К тому же, почти каждое искомое средство обладает своей дозировкой и формой. По этим причинам возникают трудности при нахождении лекарств, увеличивается шанс ошибки. Целью данной работы является реализация модифицированного алгоритма Левенштейна для поиска лекарственных препаратов.

1. Понятие расстояния Левенштейна

Расстояние Левенштейна (или редакционное расстояние) — это мера различия двух последовательностей символов (строк) относительно минимального количества операций вставки, удаления и замены, необходимых для перевода одной строки в другую [1].

Алгоритм Левенштейна [1] позволяет получить численную оценку похожести строк. Идея алгоритма заключается в том, чтобы сосчитать минимальное количество операций удаления, вставки и замены, которые необходимо провести над одной из строк, чтобы получить вторую.

Пусть Str_1 и Str_2 — две строки длиной M и N соответственно, тогда расстояние Левенштейна $d(Str_1, Str_2)$ определяется следующим образом:

$$d(Str_1, Str_2) = D(M, N),$$

при этом элементы матрицы D вычисляются по следующей рекуррентной формуле:

$$D(i, j) = \begin{cases} 0; i = 1, j = 1 \\ i; i > 1, j = 1 \\ j; i = 1, j > 1 \\ \min\left(D(i, j-1)+1, D(i-1, j)+1, D(i-1, j-1)+m(S_1[i], S_2[j])\right); i > 1, j > 1 \end{cases}$$

где $m(S_1[i], S_2[j]) = \begin{cases} 1; S_1[i] \neq S_2[j] \\ 0; S_1[i] = S_2[j] \end{cases}$, $\min(a, b, c)$ возвращает наименьший из аргументов.

Рассмотрим формулу подробнее. Шаг по i означает удаление из первой строки, по j — вставку в первую строку, шаг по обоим индексам означает замену символа или отсутствие изменений. Редакционное расстояние между двумя пустыми строками равно нулю. Очевидно, что для получения пустой строки из строки длиной j, нужно совершить i операций удаления, а для получения строки длиной j из пустой, нужно произвести j операций вставки. В нетривиальном случае нужно выбрать минимальную «стоимость» из трёх вариантов. Вставка/удаление будет в любом случае стоить одну операцию. Замена же может не понадобиться, если символы равны, так как тогда шаг по обоим индексам будет бесплатным. Формализация этих рассуждений приводит к формуле, указанной выше. Более строгое доказательство приведено в [2].

Очевидно, справедливы следующие утверждения:

1) $d(S_1, S_2) \geq \left\| |S_1| - |S_2| \right\|$;

2) $d(S_1, S_2) \le \max(|S_1|, |S_2|)$;

3) $d(S_1, S_2) = 0 \Leftrightarrow S_1 = S_2$.

Рассмотрим пример. Построим матрицу преобразования слова «Текст» в «Тост». Изначально матрица выглядит, как показано в табл. 1.

Заполнение значений в матрице происходит по следующей формуле:

Таблица 1

Изначальное состояние матрицы

		Т	о	с	т
	0	1	2	3	4
Т	1				
е	2				
к	3				
с	4				
т	5				

$$a(i, j) = \min\left(a(i-1, j)+1; a(i, j-1)+1 a(i-1, j-1) + if\left(S[i] = S[j], 0, 1\right)\right),$$

где выражение $if\left(S[i] = S[j], 0, 1\right)$ возвращает 0, если буквы, стоящие в соответствующих позициях одинаковы, и 1, если различаются.

Таким образом, чтобы получить значение элемента надо знать значения его соседей сверху, снизу и по диагонали. Для элемента $a(1,1)$ получаем

$$a(1,1) = \min\left(a(0,1)+1; a(1,0)+1; a(0,0) + if\left(S[1] = S[1], 0, 1\right)\right) = 0,$$

$$a(1,2) = \min\left(a(0,2)+1; a(1,1)+1; a(0,1) + if\left(S[2] = S[2], 0, 1\right)\right) = 1,$$

… и т.д.

В итоге имеем матрицу, значение элемента $a(n, m)$ которой равно расстоянию Левенштейна от S_1 = Тост до S_2 = Текст (табл. 2).

2. Программная реализация

Таблица 2

Конечное состояние матрицы

		Т	о	с	т
	0	1	2	3	4
Т	1	0	1	2	3
е	2	1	1	2	3
к	3	2	2	2	3
с	4	3	3	2	3
т	5	4	4	3	2

В качестве средства разработки использовался язык программирования Python. Код, используемый в данной работе, был написан в редакторе Visual Studio Code. Для решения поставленной задачи был реализован набор некоторых функций.

Функция ***get_levenhstein_distance*** (Листинг 1) принимает на вход две строки и вычисления расстояние Левенштейна.

Очевидно, что качество работы алгоритма зависит от конкретных данных. В частности, алгоритм учитывает перестановки символов и слов местами, поэтому является

Листинг 1

```
def levenshtein_distance(str1, str2):
    m = len(str1)
    n = len(str2)
    dist = [[0] * (n + 1) for i in range(m + 1)]

    for i in range(1, m + 1):
        dist[i][0] = i

    for j in range(1, n + 1):
        dist[0][j] = j

    for j in range(1, n + 1):
        for i in range(1, m + 1):
            if str1[i - 1] == str2[j - 1]:
                cost = 0
            else:
                cost = 1
            dist[i][j] = min(dist[i - 1][j] + 1,
                             dist[i][j - 1] + 1,
                             dist[i - 1][j - 1] + cost)

    return dist[m][n]
```

регистрозависимым. К тому же в названиях лекарственных препаратов часто встречаются сокращения и единицы измерения, не имеющие значения для сравнения. В связи с этим возникает необходимость предобработки данных. Для предварительной обработки строк была написана функция ***normalize*** (Листинг 2), которая принимает на вход строку, приводит её к нижнему регистру, сортирует слова в алфавитном порядке, удаляет «мусорные» слова, лишние пробелы, знаки препинания.

Листинг 2

```
def normalize(str):
    str = ' '.join(''.join(i if i not in extended_punctuation
else ' ' for i in a_string).lower().split())

    word_list = str.split()

    black_list = ['амп', 'капс', 'табл', 'мл', 'ед', ' мг', 'шт',
'уп', 'фл', 'д п', 'р р' 'в в', 'в м']
    word_list = [word for word in word_list if word not in
black_list]

    word_list.sort()
    sorted_str = ' '.join(word_list)

    return sorted_str
```

Возможны случаи, когда строки отличаются только одним словом, но алгоритм выдаёт достаточно сильное расхождение. Это связано с тем, что при сравнении двух строк слова занимают разные позиции в алфавитном порядке, поэтому сильно влияют на результат. Для решения этой проблемы необходим алгоритм, который будет учитывать исключённые из сравнения части текста. Для каждого исключённого общего слова введём свой «вес». Тогда, при вычислении сходства, будет суммироваться вес каждого одинакового слова, а не их количество. Если какое-то слово длиннее, то важность у него будет больше, чем у короткого слова. Небольшой вес у коротких слов поможет избежать проблем, создаваемыми короткими словами, которые часто не несут полезной нагрузки. Для исключения из сравнения всех одинаковых слов была написана функция ***get_wight*** (Листинг 3), которая принимает на вход две строки и рассчитывает суммарный «вес».

Листинг 3

```
def get_weight(str1, str2):

    len1 = len(str1)
    len2 = len(str2)

    words1 = str1.lower().split()
    words2 = str2.lower().split()

    unique_words1 = list(set(words1) - set(words2))
    unique_words2 = list(set(words2) - set(words1))

    common_words = list(set(words_1) & set(unique_words1))

    weight = 0
    for word in common_words:
        word_len = len(word)
        weight_in_str1 = word_len / len1
        weight_in_str2 = word_len / len2
        word_weight = (weight_in_str1 + weight_in_str2) / 2

        weight += word_weight

    return weight
```

Заключение

Таким образом, был рассмотрен алгоритм Левенштейна и разработана его модификация, которая предназначена для поиска в каталоге лекарственных препаратов, которая включает в себя предобработку входных строк и присвоение веса схожим словам. Полученные результаты могут быть использованы, например, при разработке логистических информационных систем для складов медицинских товаров.

Список литературы

1. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов / В.И. Левенштейн // Доклады АН СССР, 1965. – №163:4. – С. 845-848.

2. Гасфилд Д. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология / Пер. с англ. И.В. Романовского. – СПб.: Невский Диалект; БВХ-Петербург. 2003. – 654 с.

УДК 004.94

ОБЗОР UNITY ФРЕЙМВОРКА ДЛЯ СОЗДАНИЯ ПРОСТРАНСТВЕННЫХ ЧЕТЫРЕХМЕРНЫХ ИГР

А. Д. Колесникова, Е. В. Трофименко

Воронежский государственный университет

Введение

Четвертое пространство — это не время, как привыкли думать многие, а настоящее измерение пространства, которое работает так же, как и первые три, которые широко известны. Для человека измерения выше третьего сложны для понимания, но для компьютеров, которые с легкостью оперируют математическими формулами, в этом нет ничего сложного.

Четырехмерное пространство — это уникальная математическая концепция, которая открывает новые возможности для понимания мира вокруг нас. Хотя мы не можем физически взаимодействовать с четырехмерным пространством, идеи, которые оно подразумевает, имеют широкое применение в научных и инженерных областях, таких как физика, математика, компьютерное моделирование, инженерия и дизайн.

В настоящей статье рассматривается фреймворк для Unity, который позволяет работать с четырехмерными объектами.

1. Понятие 4D пространства.

Четырехмерное пространство строится следующим образом — каждое следующее пространство можно получить из предыдущего путем клонирования, параллельного переноса и соединения соответствующих точек [1–3]. На рисунке 1 представлен переход от одномерного пространства к четырехмерному, где:

- черным обозначено предыдущее пространство;
- красным выделено скопированное и параллельное сдвинутое предыдущее пространство;
- синим отмечены соединения соответствующих точек.

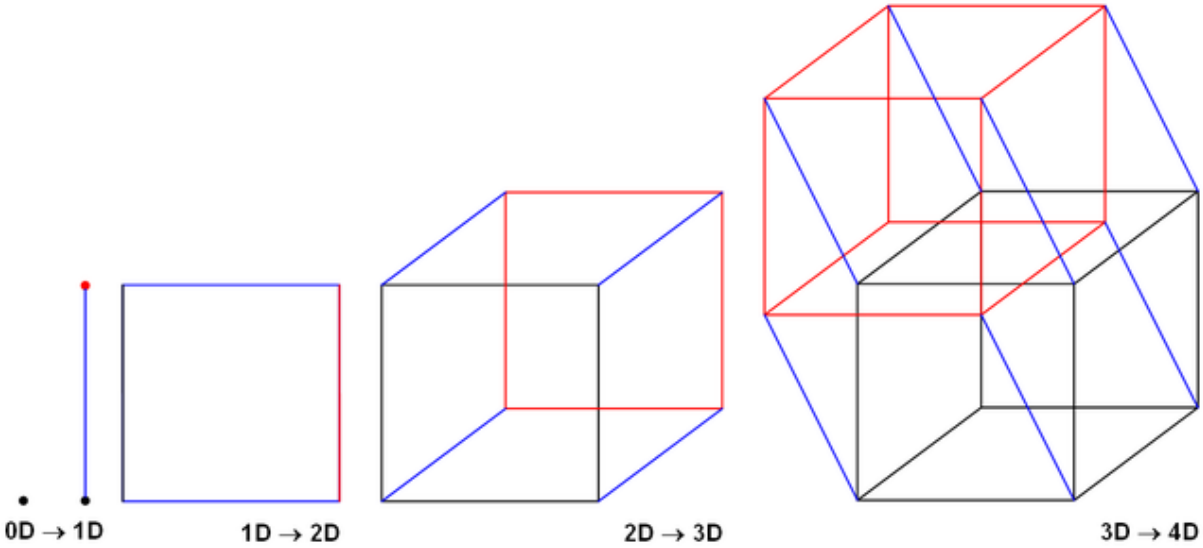


Рис. 1. Преобразование одного пространства в другое

Одним из способов визуализации объектов высших измерений является их пересечение с объектами с меньшей размерностью. Рассмотри на примере 2D и 3D. Например, возьмем трехмерный куб и пропустим его через двумерную плоскость, как это показано на рисунке 2. Здесь мы увидим одну из граней куба, а за несколько проходов - сможем рассмотреть их все. В нашем случае вместо того, чтобы брать 2D-срез 3D-объекта, мы будем брать 3D-срез 4D-объекта.

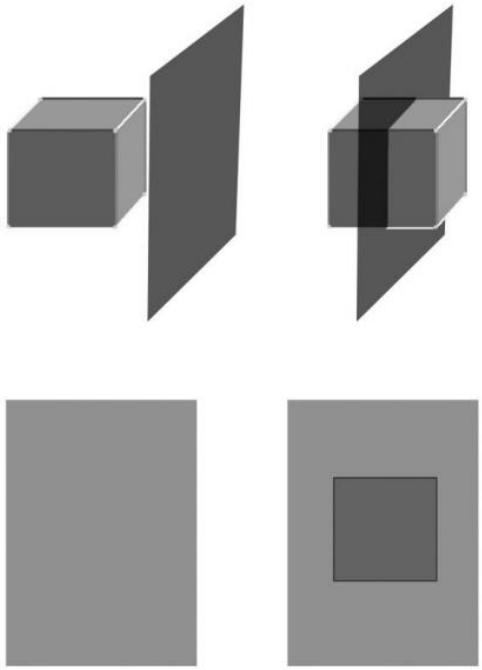


Рис. 2. Пересечение трехмерным кубом двухмерной плоскостью

2. Возможности фреймворка

Чтобы обеспечить возможность работы с четырехмерными объектами был создан фреймворк для Unity [4]. Он добавляет для редактирования четырехмерных фигур интерфейс, который добавляет объектам четвертую координату w (рис. 3).

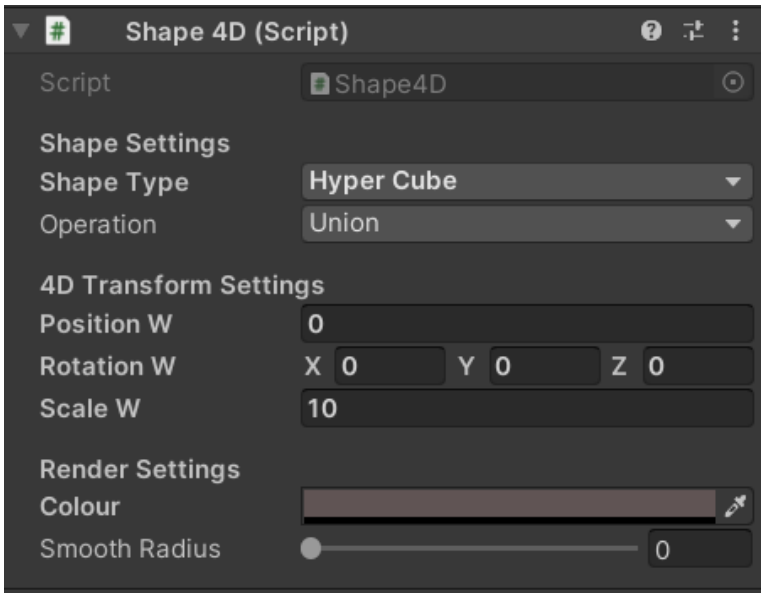


Рис. 3. Интерфейс взаимодействия с четырехмерными объектами

Данный фреймворк устанавливает для w определенное значение, затем отображает полученный для этих координат срез. Так как любая 4D фигура в разрезе будет представлять собой набор из 3D фигур, то мы будем видеть трехмерные фигуры, которые будут меняться в зависимости от значения w.

3. Метод рендеринга Raymarching

Для отрисовки среза четырехмерного объекта в данном фреймворке используется Raymarching [5–6]. Raymarching — это метод рендеринга сцен в реальном времени, который заключается в проецировании лучей на объекты со стороны наблюдателя и выполняется для каждого пикселя. Таким образом, мы выпускаем лучи через каждый пиксель изображения и, если луч оказался на поверхности объекта красим этот пиксель в цвет объекта.

Для определения пересечения луча с четырехмерными объектами требуется использование полей расстояний со знаком (signed distance fields). Поле расстояний — это функция, получающая на вход точку и возвращающая кратчайшее расстояние от этой точки до поверхности каждого объекта в сцене. Поле расстояний со знаком дополнительно возвращает отрицательное число, если точка находится внутри объекта.

Чтобы достигнуть объекта можно было бы каждый раз делать по очень маленькому шагу вдоль луча, но мы можем "шагать" более эффективно, используя алгоритм «трассировки сферами». Вместо того, чтобы на каждой итерации делать крошечный шаг, мы делаем максимальный возможный шаг. Этот максимально возможный шаг равен минимальному расстоянию из полученных (с помощью SDF) для каждого объекта на сцене.

На рис. 4 представлена визуализация Raymarcher с использованием поля расстояний со знаком, где:

- красными точками показаны точки каждого шага;
- синими кругами показаны области, которые гарантированно не содержат объектов (потому что они находятся в результатах функции поля расстояний);
- пунктирными зелеными линиями показаны истинные кратчайшие векторы между каждой точкой шага и сценой.

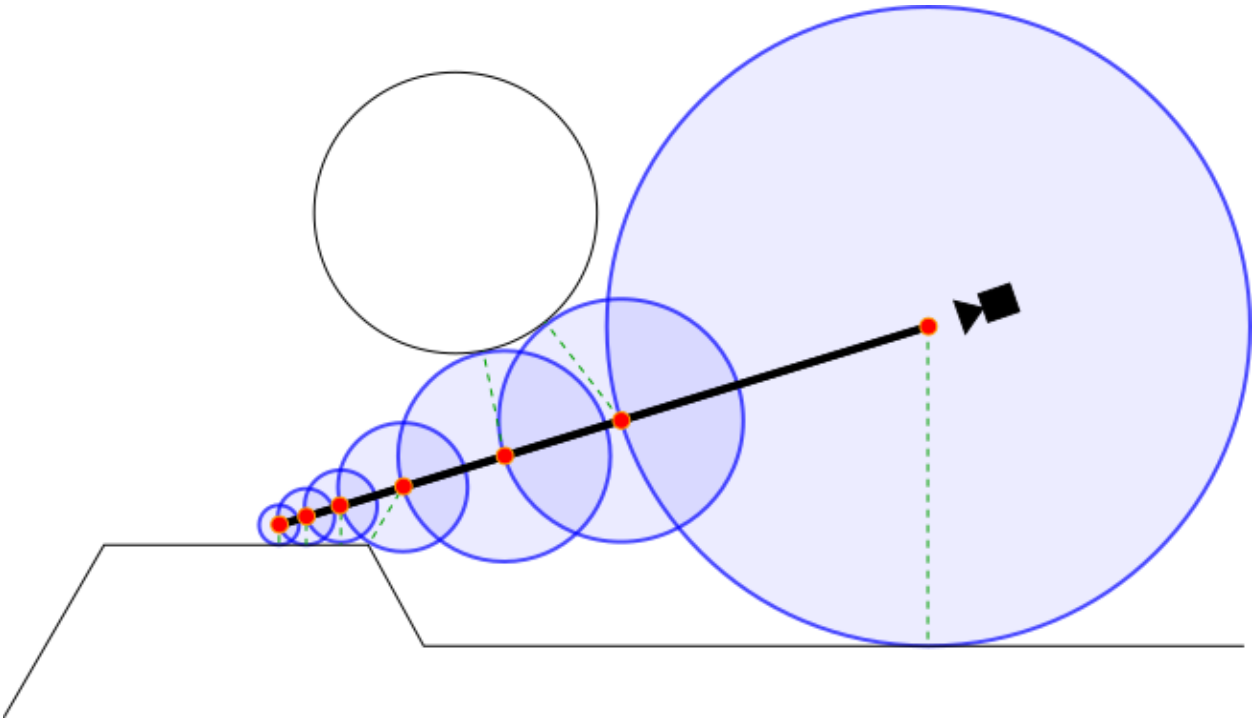


Рис. 4. Визуализация Raymarcher с использованием поля расстояний со знаком

4. Улучшение производительности

Современным компьютерам сложно обрабатывать большое количество четырехмерных объектов из-за чего производительность игры сильно падает. Данный фреймворк предлагает оптимизацию работы приложения за счет снижения качества изображения.

Разработчиком было решено использовать две камеры, одна из которых делает рендер текстуры с низким разрешением, а другая (основная) камера смотрит на эту текстуру (рис. 5). Добиться улучшения качества при снижении расширения помогает шейдер ячеек, за счет которого расплывающееся изображение приобретает более четкие очертания и яркие цвета.

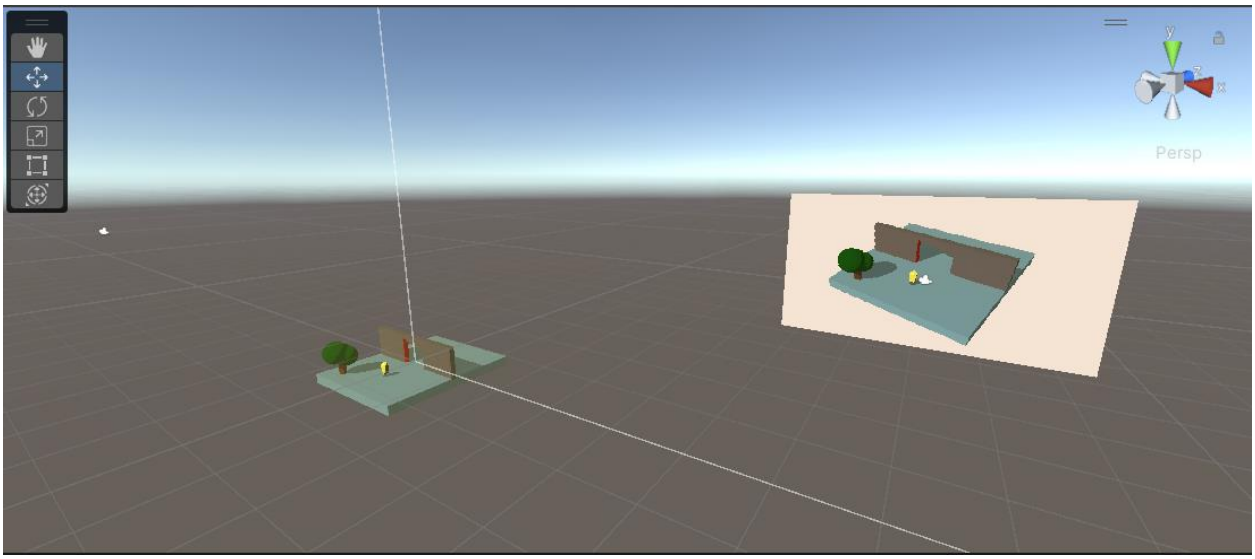


Рис. 5. Расположение камер для улучшения производительности

Заключение

В данной статье был обозрен фреймворк для создания пространственных четрехмерных игр на Unity. Так же был рассмотрен метод рендеринга Raymarching и способ улучшения производительности четырехмерной игры.

Литература

1. Парадокс четырехмерного пространства. Что, зачем и как? – Режим доступа: https://habr.com/ru/articles/727856/ – (Дата обращения: 10.04.2023).

2. Удивительные кубы Хинтона, которые позволяют каждому увидеть четырехмерный мир – Режим доступа: https://habr.com/ru/articles/727838/ – (Дата обращения: 10.04.2023).

3. Четырехмерный рендеринг: особенности, проблемы, варианты решения – Режим доступа: https://habr.com/ru/articles/113485/ – (Дата обращения: 10.04.2023).

4. 4D-Raymarching — Режим доступа: https://github.com/Jellevermandere/4D-Raymarching (Дата обращения: 10.04.2023).

5. Гамбетта, Г. Компьютерная графика. Рейтрейсинг и растеризация. / Г. Гамбетта – СПб : Питер, 2022. – 224 с.

6. Фролов, В. А. Компьютерная графика / В. А. Фролов ; Фонд Вольное Дело, 2020. – 380 с.

УДК 003.26.09

ИСТОРИЯ РАЗВИТИЯ РОССИЙСКОЙ КРИПТОГРАФИИ

А. В. Комнатный

Воронежский государственный университет

Введение

Статья посвящена изучению истории становления криптографии в России, которая является многогранной и интересной, но до сих пор малоизученной в виду того, что многие аспекты остаются засекреченными и по сей день. В ней рассматриваются этапы развития криптографии, а именно: моноалфавитные шифры, полиалфавитные шифры, электромеханическое шифрование, математическая криптография и методы, присущие каждому из них, а также использование криптографии в современном мире.

1. Криптография до XX в.

Данный период в развитии криптографии характеризуется моноалфавитными и полиалфавитными шифрами.

Считается, что родоначальником криптографии на Руси является Петр 1, но существуют свидетельства того, что первые шифры появились задолго до начала его правления. Так переписчик Псковского «Апостола» 1307 года Домид увековечил своё имя, приписав его к «Апостолу» двумя видами тайнописи — цифровой системой (замена букв и слогов цифрами, иногда усложнялась математическими операциями) и акростихом-первобуквием (осмысленный текст, сложенный из начальных букв каждой строки стихотворения) [6, 8]**.** Так же использовались шифры простой замены.

Литорея – замена одних букв алфавита другими. Она разделялась на простую и мудрую. Простая, иначе называемая тарабарской грамотой, — это замена одних согласных букв на другие. Гласные буквы оставались без изменений. В усложнённом варианте буквы в строках располагались в случайном порядке.

В 19 веке начали широко использоваться биграммныем шифры, биклавные шифры – многозначной замены.

2. Развитие криптографии в Советский период

Серьезное развитие криптография получила с появлением электромеханических устройств.

М-100 "Спектр"

Теоретическую основу шифровальной техники в СССР заложил инженер И. П. Волосок в 1930 году. Впоследствии он стал ведущим конструктором отечественной шифротехники довоенного и послевоенного периодов. Им был предложен принцип наложения случайной последовательности знаков на комбинацию знаков открытого текста [5]. Данный способ шифрования также называется аддитивным шифром. В нем используется сложение по модулю исходного текста с гаммой, представленных в численном виде. Для современного шифрования можно пользоваться кодировками, такими как ASCII или UTF-8. Гамма же либо выбирается заранее, либо генерируется при помощи регистра сдвига

"ЕС"

Помимо телеграфных сообщений, как способ передачи информации на фронте и в тылу, использовалось телефонирование. Его также было необходимо шифровать. 1935 - 1936 годах на заводе "Красная заря" было создано устройство автоматического засекречивания телефонных переговоров - инвертор "ЕС", и налажен его выпуск для каналов телефонной высокочастотной связи. Через год был налажен выпуск шифратора "ЕС-2", которая подключалась непосредственно к аппаратуре ВЧ-связи. К 1940 году завод выпустил 262 аппарата., в основном с инверсией спектра.

С-1 "Соболь" и "Соболь-П"

Наряду с производством "ЕС", в 1938 году создание шифратора для засекречивания речевых сигналов с повышенной стойкостью к дешифрированию [7]. Во второй половине того же года была завершена разработка и проведены испытания аппаратуры сложного засекречивания С-1. В данной аппаратуре уже использовалась система временных и частотных перестановок. Выпущенный же в 1942 году "Соболь-П" помимо перестановок использовал шифратор - термографическую ленту со случайно нанесенными перфорациями.

Следующим шагом для отечественной криптографии стало развитие вычислительное техники. Со временем появилась необходимость шифровать информацию, представленную на цифровых носителях. Были разработаны различные стандарты защиты информации. Одним из них был ГОСТ 28147-89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования» [1]. Этот стандарт является примером классического DES подобного блочного алгоритма, основанного на ячейках сети Фейстеля. Длина блока 64 бита, ключа 256 бит.

В этом шифре используются таблицы замены или так называемые S боксы.

- Каждый блок 64 бита делится на два подблока
- Левый подблок Ai складывается по модулю 2^32 с раундовым ключом
- Левый подблок проходит через S-бокс.
- Биты левого подблока сдвигаются на 11 позиций (циклический сдвиг).
- Левый подблок складывается с правым по модулю 2
- Подблоки меняются местами

Поскольку этот шифр работает с блоками строго определенной длины может возникнуть вопрос, а что же делать, если нужно зашифровать более длинной сообщение. Для этого в ГОСТ [1] описаны следующие режимы работы:

1. Режим простой замены: сообщение разбивает на блоки по 64 бита и каждый блок шифруется отдельно. В случае неполноты последнего блока, он дополняется в одном из режимов дополнения блока.

2. Режим гаммирования называется так потому, что в нем используется гамма - псевдослучайная последовательность, которая в каждом раунде складывается по модулю 2 с открытым текстом. Гамма образуется из синхропосылки S - псевдослучайной последовательности, которая изменяется с каждой итерацией.

3. Режим гаммирования с обратной связью. Алгоритм похож на режим гаммирования, однако гамма формируется на основе предыдущего блока зашифрованных данных, так что результат шифрования текущего блока зависит также и от предыдущих блоков.

4. Режим имитовставки. В этом режиме вырабатывается имитовставка - дополнительный блок фиксированной длины, зависящий от исходного текста и ключей. Такой небольшой блок нужен для подтверждения того, что в шифротекст случайно или преднамеренно не были внесены искажения, — то есть для проверки целостности.

3. Криптография на современном этапе развития

На данном этапе развития используется математическая криптография. Основой Российской криптографии являются 4 основных блока: цифровая подпись, хеш-функция, шифры и режимы работы шифров.

ГОСТ 34.12—2015 (Кузнечик) имеет длину блока 128 бит и длину ключа 256 бит [4].

В этом стандарте помимо S перестановок были использованы линейные преобразования L. Они основываются на функции умножения чисел в конечном поле (или поле Галуа) над неприводимым полиномом.

Так же тут осуществляется всего 10 раундов.

В этом стандарте были добавлены новые режимы шифрования:

1. Режим простой замены с зацеплением Особенностью данного режима работы является то, что каждый блок открытого текста, за исключением первого, складывается с предыдущим результатом шифрования.

2. Режим обратной связи по шифротексту - для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным результатом шифрования предыдущего блока.

ГОСТ 34.10-2018 описывает алгоритмы формирования и проверки электронной цифровой подписи с помощью операций в группе точек эллиптической кривой и функции хеширования [2]. Длины секретного ключа 256 бит и 512 бит.

У пользователя есть сообщение, ключ подписи и ключ для проверки подписи. Ключ проверки подписи является открытым, для того, чтобы любой получатель смог убедиться в достоверности подписи. То есть если получателю удается получить равенство с помощью открытого ключа проверки подписи, принадлежащего определенному отправителю, то он может быть уверен, что сообщение подписывалось ключом подписи именно этого отправителя.

ГОСТ 34.11-2018 посвящен функции хеширования [3]. В данном документе содержится описание алгоритма вычисления хеш-функции, известной из предыдущих версий стандарта, как Стрибог.

Стрибог принимает на вход сообщение произвольной длины, которое впоследствии разбивается на блоки размером 512 бит (с дополнением блоков при необходимости). После чего входные данные преобразуются в хеш-код фиксированной длинны 256 или 512 бит.

Заключение

Российская криптография прошла немалый путь от самых простых способов защиты данных, таких как шифры простой замены, перестановок, биграммные шифры и т.д., до передовых – блочные шифры, электронно-цифровые подписи и хэширование. Сейчас криптография обладает всеми необходимыми инструментами для обеспечения безопасности данных.

Литература

1. ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования : дата введения 01.07.1990. – Москва : ИПК Издательство стандартов, 1996. – 28 с.

2. ГОСТ 34.10-2018 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи : дата введения 01.06.2019. – Москва : Стандартинформ, 2018. – 20 с.

3. ГОСТ 34.11-2018 Информационная технология. Криптографическая защита информации. Функция хэширования : дата введения 01.06.2019. – Москва : Стандартинформ, 2019. – 23 с.

4. ГОСТ Р 34.12-2015 Информационная технология. Криптографическая защита

информации. Блочные шифры : дата введения 01.01.2016. – Москва : Стандартинформ, 2018. – 15 с.

5. Кузьмина, Н. В. Из истории криптографии / Н. В. Кузьмина // Вестник российских университетов. Математика. – 2003. – Т. 3, № 1. – С. 217-218.

6. Лузгин, Д. А. История криптографии / Д. А. Лузгин, А. Р. Галимов, О. И. Свиридов // Экономика и социум. – 2018. – № 12 (55). – С. 737-739.

7. Пчелинцева, Н. В. К вопросу применения криптографии / Н. В. Пчелинцева, И. В. Чепраков, А. А. Гущина // Наука и образование. – 2022. – Т. 5, № 2. – С. 15-23.

8. Токарева, Н. Н. Об истории криптографии в России / Н. Н. Токарева // Прикладная дискретная математика. – 2018. – № 4 (18). – С. 82-105.

УДК 004.43

АКТУАЛЬНОСТЬ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ 1С: ПРЕДПРИЯТИЕ

Д.С. Конюхова, Д.В. Борисенков

Воронежский государственный университет

Введение

В настоящее время перед большинством компаний стал актуальный вопрос: как автоматизировать бизнес-процессы? Сейчас каждая организация имеет возможность использовать самые новые технологии ведения электронного учёта, автоматизации бизнес-процессов, систем электронных расчётов. С их помощью компании могут эффективнее использовать свои ресурсы, а также улучшить показатели управленческой деятельности. Помимо всего этого, технологии автоматизации позволяют повысить финансовые показатели организации путём рационализации производственного процесса.

В наше время у каждого человека есть при себе маленький переносной компьютер – смартфон, а мобильные приложения в свою очередь становятся все популярнее. Мобильные приложения были созданы для работы в них через смартфоны или же через планшетные компьютеры. Главным преимущество мобильных приложений является возможность работать в любом месте в любое время, при всем при этом функционал не сильно отличается от работы на обычном стационарном компьютере. На данный момент есть две наиболее популярные мобильные операционные системы – Android и iOS, также можно отметить и Windows, но сейчас крайне мало смартфонов на этой операционной системе.

1. Почему 1С? Актуальность.

Широкие возможности системы “1С: Предприятие” обусловлены её архитектурой. Система состоит из платформы и конфигурации – прикладных программ. Простыми словами, платформа – это фундамент приложения, а конфигурация в свою очередь отвечает за функциональность. Конфигурации разрабатываются на базе платформы и не могут существовать в отрыве от неё. При этом платформа только одна, а конфигураций – сколько угодно. Такой способ позволяет платформе быть очень гибкой. При этом главное удобство разработки на данной платформе в том, что имеются типовые решения, которые можно доработать под нужды конкретного заказчика. Но все чаще у предприятий появляется необходимость в удалённом доступе к приложению 1С.

Приведём несколько типичных примеров:

1. Есть пиццерия, откуда ежедневно курьеры вывозят десятки заказов. Когда-то давно бухгалтерия велась от руки с помощью листа бумаги и ручки. Не так давно начали использовать стационарные компьютеры, которые уже позволяли сразу внести всю информацию в приложение, но такой вариант имел тот недостаток, что курьеру для внесения всей необходимой информации нужен был доступ к компьютеру. Вариант решения – нанять диспетчера, который будет передавать всю информацию курьеру и заказчику, что и делалось лет 10 назад. Оператор здесь является третьим звеном, что невыгодно для бизнеса. Для этого и

были разработаны мобильные приложения, позволяющие без использования стационарного компьютера посмотреть всю необходимую информацию. И теперь нет нужды оплачивать лишнее рабочее место, что позволит увеличить доход бизнеса как минимум на зарплату оператора, но самое главное, что такое приложение упростит заказ еды из заведения, ведь больше нет необходимости ждать ответа оператора.

2. Самый простой и короткий пример. Директору необходимо посмотреть рабочие документы, чтобы решить срочный вопрос, но в офисе его по какой-то причине нет, теперь всё это есть прямо в мобильном телефоне.

3. Офис крупной компании по заготовке леса находится в крупном городе, а заготовка леса производится далеко за чертой города. Компании требуется удалённый доступ к приложению для того, чтобы вносить новые заказы, давать обратную связь о готовности того или иного заказа, хранить информацию о наличии тех или иных заготовок.

Таких примеров можно привести достаточно много. В услугах удалённого доступа к платформе нуждаются многие профессии: риелторы, застройщики, работники складов, руководители предприятий и организаций любых отраслей и так далее. Из этого всего становится ясно, для чего необходима разработка мобильных приложений 1С.

Итак, перечислим преимущества предлагаемого подхода:

1. Ускорение бизнес-процессов
2. Повышение эффективности работы сотрудников
3. Отсутствие привязанности к офису

Так почему же при огромном выборе платформ стоит выбрать 1С? На это есть ряд причин:

1. Программные продукты компании 1С являются самыми популярными в России. Большинство российских организаций используют продукты этой компании.

2. Приложения на платформе 1С являются многофункциональными благодаря своим конфигурациям, которые в свою очередь могут интегрироваться с мобильным приложением.

3. Разработка на 1С дешевле, чем на любом другом языке

4. Скорость разработки. Благодаря типовым конфигурациям, которые достаточно подправить для конкретных целей, разработка занимает в разы меньше времени, чем разработка на любой другой платформе или традиционном языке программирования.

5. Одно и то же разработанное приложение будет работать, как на Android так и на iOS.

6. Возможность в короткие сроки внести изменения в работу приложения.

7. Нет никаких проблем с поиском сотрудника, который будет использовать приложение, так как на рынке труда их достаточно

2. Описание мобильного приложения

Была поставлена задача реализации взаимодействия между курьером интернет-магазина, работающим на мобильном устройстве (телефоне или планшете Android), и офисом самого интернет-магазина, где работает офисное приложение на стационарном компьютере.

Офисное приложение будет одно, вариантом мобильного приложения, которые могут с ним взаимодействовать, будет приложение мобильного клиента.

В офисном приложении реализованы следующие функции:

- При поступлении товаров на склад менеджер интернет-магазина создает приходные накладные. Суммы по товару автоматически пересчитываются при изменении его цены и количества. Поступившие товары учитываются в разрезе складов, размеров и цветов.

- Полученные товары заносятся в справочник товаров. Для каждого товара можно выбрать картинку, которая находится в справочнике хранимых файлов.

- Менеджер устанавливает цены товаров на определенную дату, по которым они будут продаваться. Эти цены хранятся в периодическом регистре сведений.

- При создании заказа менеджер вводит данные клиента, дату доставки, отмечает важность и статус заказа (*Открыт*). Адрес доставки заполняется автоматически из данных клиента. При заполнении заказа актуальные цены товаров подставляются в заказ. Суммы по строкам табличной части автоматически пересчитываются при изменении товара, его цены и количества.

- Менеджер интернет-магазина создает заказ со статусом *Открыт*, затем указывает курьера, который будет обслуживать заказ, и устанавливает статус заказа *В работе*. После выполнения заказа курьером менеджер проверяет заказ со статусом *Выполнен* и присваивает ему статус *Закрыт*. Заказанные товары учитываются в разрезе складов, размеров и цветов.

- Редактирование ранее созданных заказов возможно только в том случае, если заказ имеет статус *Открыт* или *В работе*. Заказ со статусом *Выполнен* можно только закрыть, а заказ со статусом *Закрыт* можно только просматривать.

- Список заказов отображается с условным оформлением в зависимости от статуса заказа и его важности.

- В офисном приложении реализована работа с хранимыми файлами: выбор файла из локальной файловой системы, запись в справочник хранимых файлов, чтение файла из справочника и его открытие или запись в файловую систему пользователя. В основном справочник хранимых файлов используется для хранения и выбора картинок товаров.

- На основе закрытых заказов создаются расходные накладные со списком товаров, которые клиент купил. Кроме того, менеджер может ввести расходную накладную вручную. Суммы по товару автоматически пересчитываются при изменении товара, его цены и количества. Проданные товары учитываются в разрезе клиентов и курьеров.

• Для анализа работы интернет-магазина в офисе строятся различные отчеты: об остатках товаров на складах в разрезе цветов и размеров, о причинах отказа от товаров и о продажах товаров клиентам.

В мобильном приложении будут реализованы следующие функции:

• Курьер получает из интернет-магазина список заказов в порядке их обслуживания. Затем он забирает заказанные товары со склада.

• Курьер открывает первый заказ из этого списка и определяет местоположение клиента, сделавшего заказ, на карте.

• Звонит клиенту и сообщает о том, что он к нему едет. Возможно, он не дозванивается или задерживается, тогда курьер может послать клиенту SMS.

• Приезжает к клиенту. После знакомства с товарами клиент какие-то позиции заказа берет, а какие-то — нет.

• Курьер отмечает факт отказа от товаров в заказе. При этом у возвращенных товаров указывается причина отказа.

• В случае несоответствия товара по цвету или качеству курьер может сделать его фото- и видеосъемку, а также записать аудиоотзыв клиента о заказе.

• Курьер может добавить новый заказ на этого же или нового клиента, если возникла такая необходимость (например, нужно перезаказать тот же товар, но другого размера).

• В случае необходимости курьер может создавать для себя напоминания (например, напоминание о звонке клиенту), которые будут появляться в указанное время или повторяться периодически. Кроме того, курьер может отправить клиенту письмо по электронной почте.

• После окончания обслуживания заказа курьер ставит отметку о его выполнении (присваивает заказу статус *Выполнен*).

• При необходимости курьер может сформировать и проанализировать данные в отчетах. Причем эти отчеты могут быть построены как на данных мобильного приложения, так и удаленно в офисном приложении и переданы курьеру через веб-сервис.

• Интернет-магазин может посылать курьеру сообщения — например, если в офисе изменилась важная для курьера информация.

• Затем курьер находит следующий по порядку заказ в списке обслуживания, открывает данные клиента, сделавшего заказ, и т. д.

3. Возможности мобильных устройств

Для автоматизации в мобильном клиенте будет добавлена возможность курьера делать фото/видео съемку товаров, записывать аудиоотзыв клиента. Для этого необходима реализация мультимедийных возможностей.

В форму были добавлены кнопки: СделатьАудиоЗапись, СделатьВидеоЗапись, СделатьФотоСнимок, которые будут работать только в мобильном клиенте. Видимость и доступность этих кнопок прописывается в коде, как

```
#Если НЕ МобильныйКлиент Тогда
   Элементы.СделатьАудиозапись.Видимость = Ложь;
   Элементы.СделатьВидеозапись.Видимость = Ложь;
   Элементы.СделатьФотоснимок.Видимость = Ложь;
#Иначе
   Если ТолькоПросмотр = Ложь Тогда
      Элементы.СделатьАудиозапись.Доступность = СредстваМультимедиа.ПоддерживаетсяА
      удиозапись();
      Элементы.СделатьВидеозапись.Доступность = СредстваМультимедиа.ПоддерживаетсяВ
      идеозапись();
      Элементы.СделатьФотоснимок.Доступность = СредстваМультимедиа.ПоддерживаетсяФо
      тоснимок();
   Иначе
      Элементы.СделатьАудиозапись.Доступность = Ложь;
      Элементы.СделатьВидеозапись.Доступность = Ложь;
      Элементы.СделатьФотоснимок.Доступность = Ложь;
   КонецЕсли;
#КонецЕсли
```

В этом обработчике устанавливается, что команды для работы со средствами мультимедиа будут видимы только при работе в мобильном клиенте *(#Если МобильныйКлиент*) и доступны только в том случае, если данные в форме хранимого файла можно редактировать и на мобильном устройстве поддерживаются мультимедийные возможности.

Таким образом, в форме хранимого файла, открытой на стационарном компьютере, пользователь не увидит команд для работы со средствами мультимедиа, как будто их и нет вообще (рис. 3)

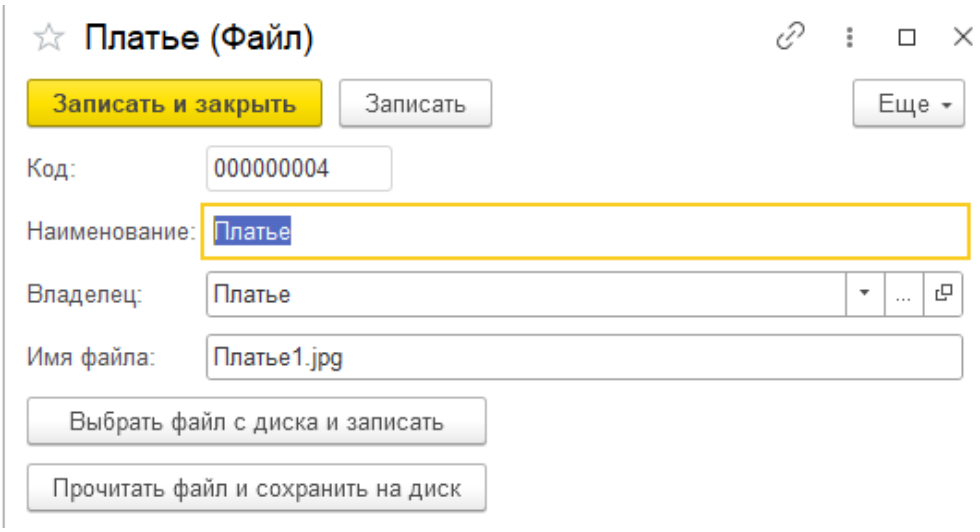


Рис. 3. Форма справочника «Хранимые файлы» на стационарном компьютере

В мобильном клиенте курьеру будут доступны все команды для работы с хранимыми файлами (рис. 4)

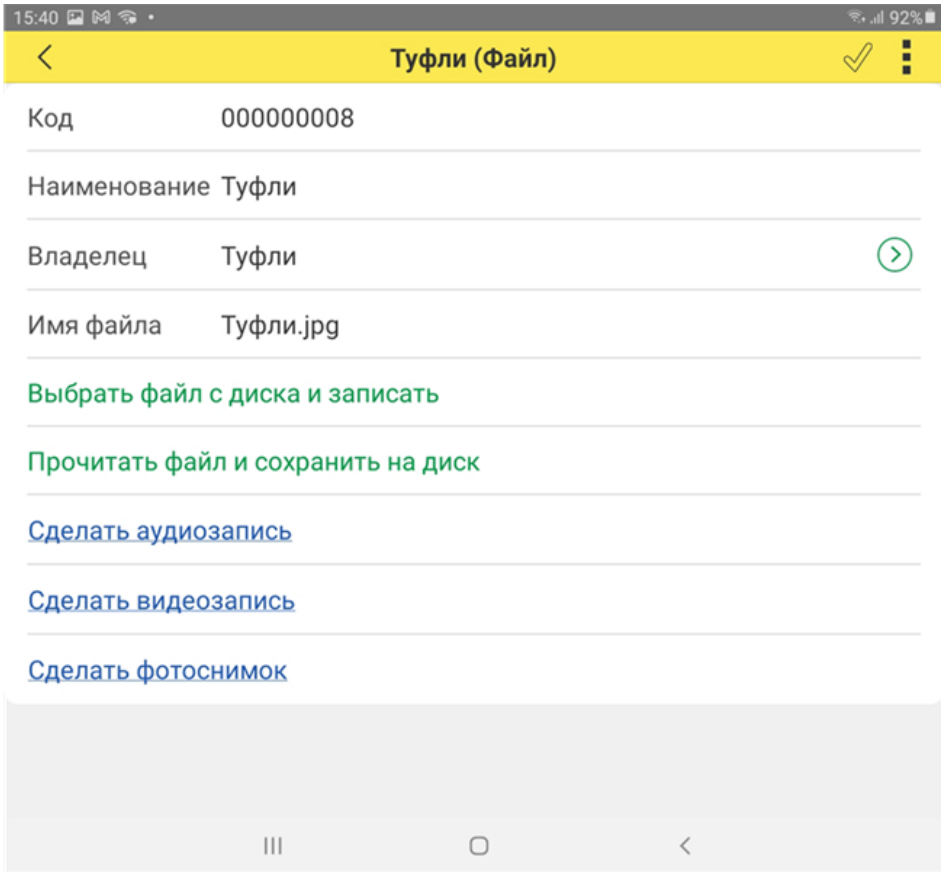


Рис. 4. Форма хранимого файла на телефоне

Например, возможна такая ситуация. Если курьер доставляет клиенту товар с браком, то открыв карточку товара, на закладке *Файлы* курьер может добавить фото товара, на котором будет зафиксирован этот брак.

Заключение

Мобильные устройства прочно вошли в нашу жизнь, сегодня они используются не только для звонков и отправки смс-сообщений, но и позволяют использовать богатый функционал для развития бизнеса. Созданное компанией 1С мобильное приложение существенно расширила возможности программного продукта, она может работать на мобильных операционных системах: Android 13 и выше. Мобильный клиент представляет собой единую базу, как для стационарного компьютера, с возможными ограничениями, так и для Android. С помощью этого варианта платформы можно собрать мобильное приложение, используя которое, «удаленный» пользователь на своем мобильном устройстве сможет работать с информационной базой так же, как если бы он работал в офисе за стационарным компьютером. При этом он будет вносить данные в ту же информационную базу, подключаться к которой сможет только онлайн.

Если проводить аналогию с платформой для настольных компьютеров, то такое мобильное приложение является аналогом тонкого клиента, работающего с информационной базой, опубликованной на веб-сервере.

Таким образом, с помощью этого варианта платформы можно достаточно легко и быстро, с минимальными усилиями разработчика собрать мобильное приложение, имеющее всю функциональность офисного приложения и работающее с информационной базой в режиме онлайн.

В данной работе были изучены актуальные источники по работе с мобильным клиентом. Предметной областью данной задачи является менеджер, курьер и клиент. Задача находится в процессе реализации, которая требует использования технологий:

- Мультимедиа;
- Геолокация;
- Звонки;
- Смс-сообщения;
- Электронная почта;
- Push-уведомления.

Литература

1. Катаев, С. М. Программирование в 1С: Предприятие 8.3 / С. М. Катаев, Ю. А. Сергиенко. - Санкт-Петербург : Питер, 2014.- 304 с.

2. Автоматизированные информационные системы в экономике : учебное пособие / Г. Г. Куликов, Е. А. Дронь, М. А. Шилина, Ю. О. Багаева : Уфимск. гос. авиац. техн. ун-т. – Уфа : УГАТУ, 2013. – 186 с.

3. Гвоздева, Т. В. Проектирование информационных систем : учебное пособие / Т. В. Гвоздева, Б. А. Баллод. – Ростов-н/Д: Феникс, 2014 – 508с.

4. Заботина Н. Н. Проектирование информационных систем. Учебное пособие для студентов высших учебных заведений, обучающихся по специальности 080801 "Прикладная информатика (по областям) и другим экономическим специальностям" / Н. Н. Заботина – Москва : ИНФРА-М, 2013 – 329с.

5. Видеокурс «Разработка мобильных приложений под Android на платформе 1С» [Электронный ресурс]. – Режим доступа : http://servicebook.pro/uchebnyi-centr/video/courses/videokurs- razrabotkamobilnyh-prilozhenij-pod-android-na-platforme-1s-predpriyatie-8-3- na-primereprilozheniya-mobilnye-finansy/?page=all (дата обращения: 20.02.2018).

6. Радченко М. Г. Архитектура и работа с данными «1С: Предприятия 8.2». Серия «1С: Профессиональная разработка» / М. Г. Радченко, Е. Ю. Хрусталева. – Москва : ООО «1С-Паблишинг», 2011. – 268 с.

7. Рыбалка В. В. Пример быстрой разработки мобильного приложения на платформе «1С: Предприятие 8.3» / В. В. Рыбалка. – Москва : ООО «1С-Паблишинг», 2014. – 329 с.

УДК 004

Обновление сайта кафедры САиУ

В. В. Корнеева

Воронежский государственный университет

Введение

В настоящей научно-исследовательской работе представлен новый, улучшенный веб-сайт кафедры САиУ с использованием современных методов разработки веб-ресурсов, включая конструкторы сайтов. Обновление веб-сайта кафедры учебного заведения является актуальной задачей, направленной на улучшение взаимодействия с различными заинтересованными сторонами, включая студентов, преподавателей и потенциальных студентов.

Целью исследования является оценка эффективности использования конструктора сайтов в процессе обновления веб-сайта кафедры учебного заведения. В работе произведен анализ процесса выбора и адаптации шаблона, функциональность и эстетика обновленного веб-ресурса, а также его воздействие на пользователей.

Данное исследование имеет практическую значимость для учебных учреждений, стремящихся улучшить свое онлайн-присутствие, а также для разработчиков веб-ресурсов, которые могут использовать полученные результаты для оптимизации процесса создания и обновления сайтов с использованием конструкторов.

1. Анализ доступных ресурсов для обновления сайта

Конструкторы веб-сайтов представляют собой инструменты, разработанные для облегчения и быстроты создания веб-сайтов для пользователей, не обладающих знаниями в области программирования или информационных технологий.

Приведем наиболее популярные конструкторы сайтов с краткими характеристиками:

- LPGenerator [1] — профессиональный конструктор посадочных страниц, интернет-магазинов небольших масштабов, а также квиз-сайтов. Имеет строгую коммерческую направленность и служит цели роста прибыли для бизнеса.
- Tilda [2] — лидер в создании одностраничных сайтов. В пределах бесплатной функциональности пользователям доступно создание одного сайта с использованием готовых шаблонов или Zero Block — встроенного редактора.
- ReadyMag [3]: основная особенность этого конструктора — огромный выбор анимаций, которые можно настроить под свой вкус. Есть сетка для подбора количества колонок. В конструкторе предусмотрена адаптивная верстка для мобильных устройств.
- Setup [4] — конструктор со своей библиотекой шаблонов. На нем можно сделать сайт-визитку, лендинг или собрать простой магазин. Но в пределах бесплатного доступа функционал сильно ограничен.
- uCoz [5] официально можно назвать один из отцов всего Рунета благодаря широкой функциональности бесплатной версии: удобный визуальный редактор, простое подключение домена, много модулей для настройки ресурса под свои предпочтения.
- Google Sites [6] — очередной продукт корпорации, призванный улучшить жизнь пользователей интернета. Этот конструктор отлично подойдет для новичков: в нем есть около 13 шаблонов и возможность собрать свой вариант шаблона с нуля. Не нужна дополнительная регистрация — достаточно просто авторизоваться в аккаунте Google, чтобы получить доступ к

конструктору.

2. Сравнение версий сайта

Перейдем к этапу сравнения обновленной и старой версий сайта. Для этого будем рассматривать рисунки, представленные ниже. На них можно заметить, что частично изменилось цветовое оформление сайта, но основная стилистика бирюзовых акцентов осталась неизменной.

Более детальный анализ начнем с вкладки «Главная» (рис. 1, 2). В новой версии сайта (рис. 2) на этой странице появился текст, в котором приветствуются посетители сайта САиУ, приводится краткая информация о кафедре, описываются основные цели создания кафедры.

Далее следует рассмотреть следующий раздел (рис. 3, 4): «О кафедре» - страница, где представлена история кафедры и интересные факты о ней. Можно заметить, что особых изменений, кроме дизайна и актуализации некоторых данных о кафедре, не наблюдается. Но блок стал более информативным и интересным.

Следующий раздел – «Кафедра в лицах» (рис. 5): он делится на два подраздела. На первом располагаются карточки сотрудников, работающих на кафедре, вместе с фотографиями и занимаемыми должностями, а также скрываемое поле, с информацией о бывших сотрудниках кафедры, которые оставили свой след в научной, учебной и воспитательной работе кафедры. На втором разделе планируется организовать карточки выпускников кафедры. Данные для этого раздела находятся в процессе сбора и обработки. Поэтому на сайте выводится сообщение: «Этот раздел в данный момент находится в разработке. Приносим свои извинения».

На вкладке «Образование» (рис. 6, 7) можно заметить существенные изменения: более подробную информацию представляют подстраницы «Бакалавриат», «Магистратура», «Аспирантура». Также эти вкладки получили возможность по щелчку мыши свернуть и развернуть содержимое каждого раздела.

Появился новый раздел на сайте - «Студентам» (рис. 8, 9), в котором находятся два подраздела: «Методическая копилка» и «Расписание».

«Методическая копилка» содержит в себе необходимые для обучения справочные материалы, разработанные преподавателями кафедры и другими учеными. «Расписание» переместилось из раздела «Обучение» (рис. 6).

Все представленные на станице документы хранятся в Гугл Диске, что обеспечивает возможность легкого доступа к справочным материалам и быстрого обновления информации в расписании.

С «Главной страницы» можно открыть список «Полезные ссылки», такие как сайт ВГУ, сайт ПММ, сайт Зональной научной библиотеки ВГУ (рис. 10), и, щелкнув по выбранному элементу списка перейти на соответствующий интернет-ресурс. Кроме того, можно посетить этот раздел и с помощью кнопок также перейти по указанным ссылкам (рис. 11).

Раздел «Контактная информация» (рис. 12, 13) приобрел новое оформление: появилась интеграция Гугл Карт, с указанием местоположения главного корпуса ВГУ. Это поможет новым посетителям сайта узнать, маршрут до университета. Также это раздел приобрел фиксированное положение – он доступен в нижней части любой страницы сайта, что позволяет пользователям оперативно задавать вопросы о кафедре.

В новой версии сайта появилась возможность просмотра с мобильных устройств, благодаря внедрению адаптации страниц. Это позволит всем желающим посещать сайт кафедры с любых доступных устройств и не испытывать неудобств с просмотром страниц.

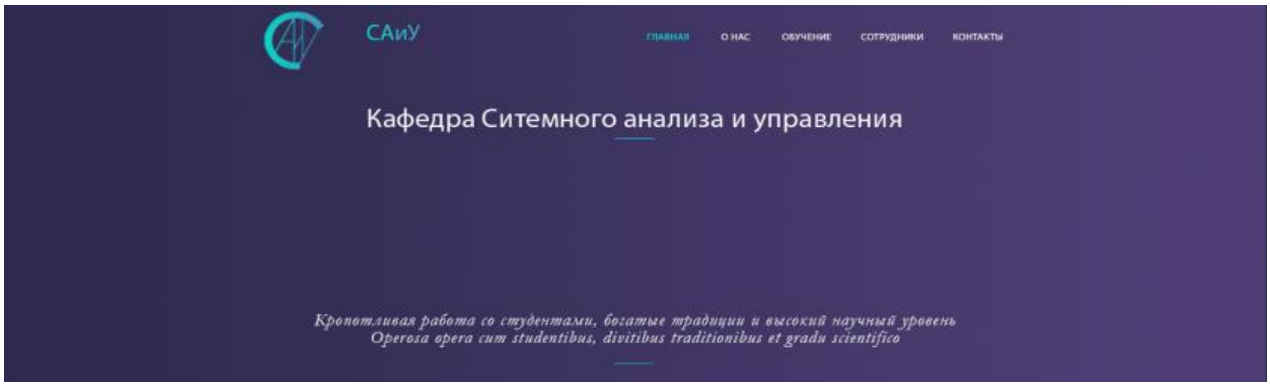


Рис. 1 Главная страница старой версии сайта САиУ

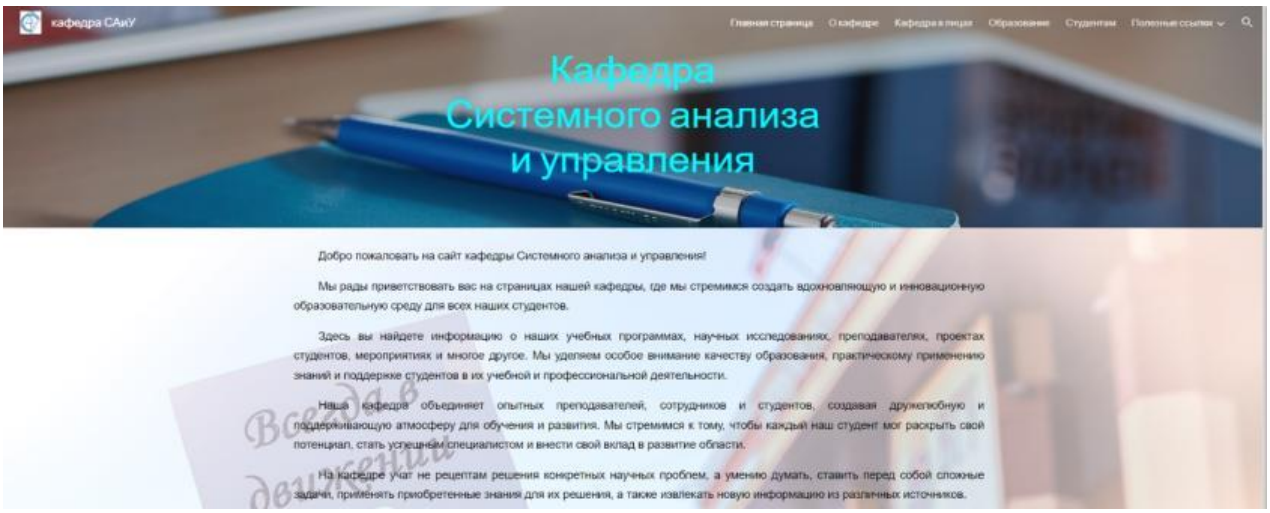


Рис. 2 Главная страница новой версии сайта САиУ

О НАС

Полное название кафедры – кафедра Системного анализа и управления (кафедра Нелинейных колебаний)
Дата образования – 1971 год

Кафедра системного анализа и управления фактически старше факультета ПММ, так как она уже существовала, правда под другим названием, еще на математико-механическом факультете. Тогда она называлась кафедрой обыкновенных дифференциальных уравнений. На факультете ПММ кафедра существует с 1971 года. До недавнего времени это была кафедра нелинейных колебаний.

Возглавлял кафедру со дня основания и по 2003 год ученик М.А. Красносельского, доктор физ.-мат. наук, профессор Перов Анатолий Иванович: член Американского математического общества, академик Академии нелинейных наук, сорosовский профессор, его имя включено в энциклопедию «Who's Who in Sience and Ingineering». Им подготовлено 26 кандидатов и 6 докторов наук.

В настоящее время заведует кафедрой ученик А.И.Перова доктор физ.-мат.наук, профессор ЗадорожнийВ.Г. Преподавательскийсостав кафедры, как правило,формировался из учеников А.И. Перова.

В разные годы на кафедре работали Боровских А.В., Жукова Г.С., Кравец О.Я., Кузнецов А.А., Минаев В.А., Проценко О.Б.,

Рис. 3 Вкладка «О нас» старой версии сайта САиУ

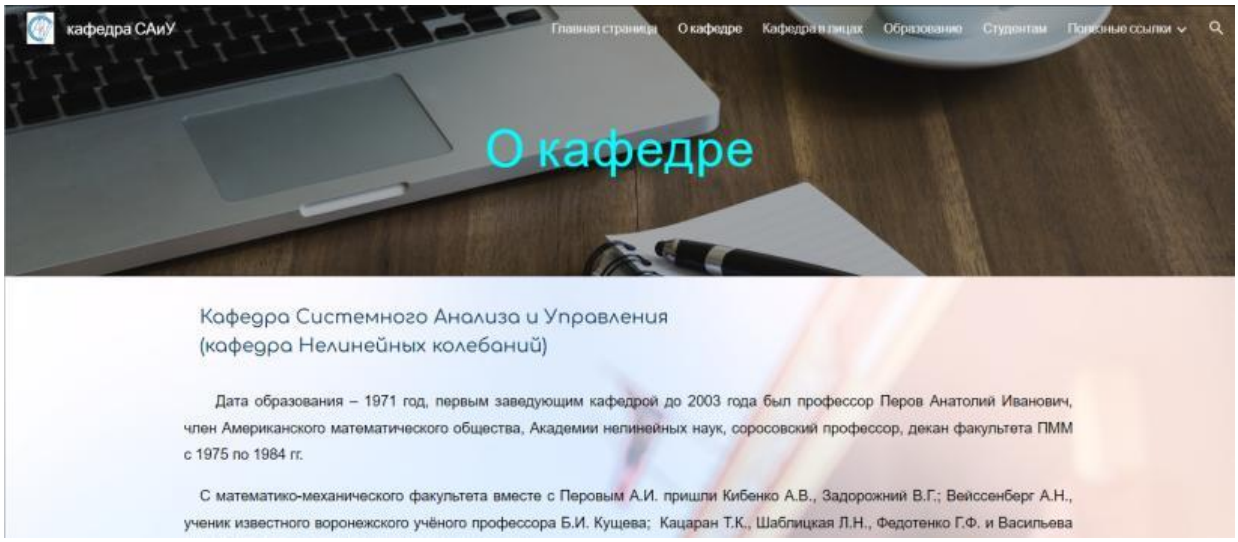


Рис. 4 Вкладка «О кафедре» новой версии сайта САиУ

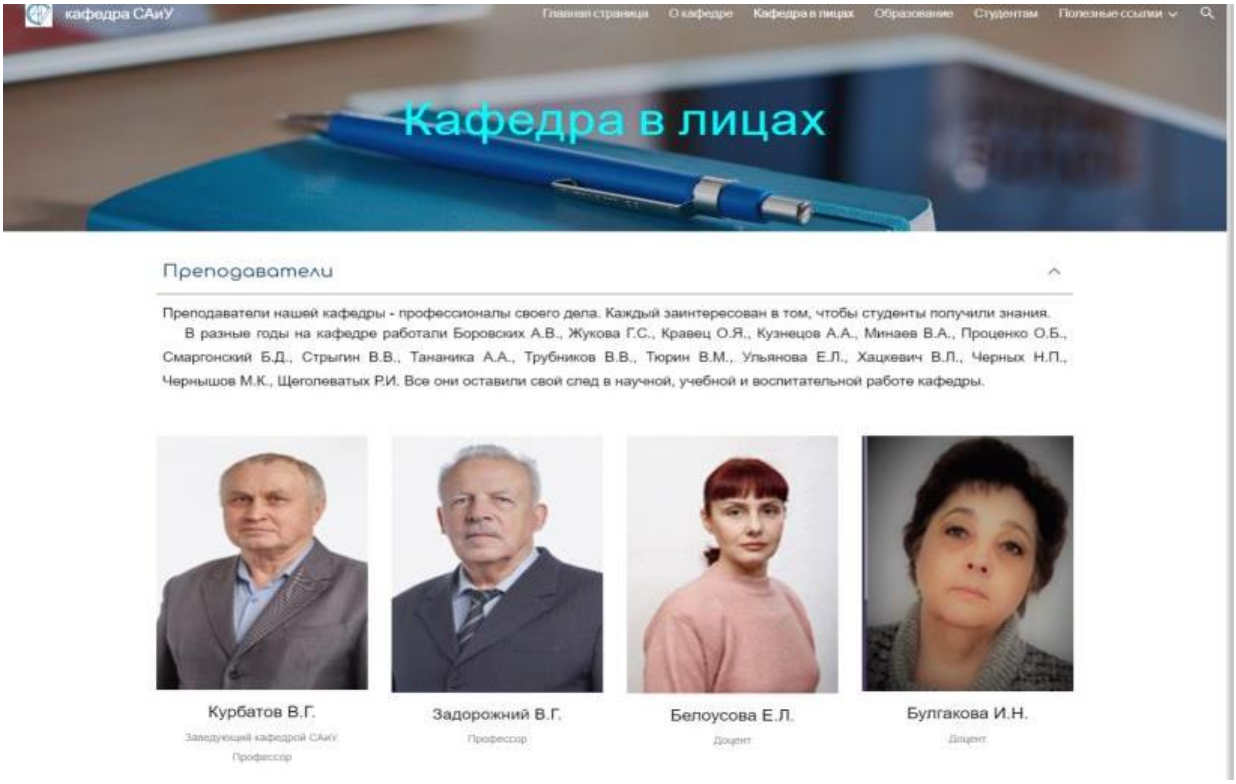


Рис. 5 Вкладка «Кафедра в лицах» новой версии сайта САиУ

День недели	ФИО преподавателя	Задорожний В.Г.	Перов А.И.	Баскаков А.Г.	Курбатов В.Г.	Белоусова Е.П.	Булгакова И.Н.	Коструб И.Д.	Костина Л.Н.	Кабанцова Л.Ю.	Корчагина Е.В.	Гусева Е.Ю.
Понедельник	8:00 - 9:35	436					320	214				
	9:45 - 11:20	436				410П	305	437 433		504П		
	11:30 - 13:05					410П	404П		304	504П		
	13:25 - 15:00					410П				504П		
	15:10 - 16:45											
	16:55 - 18:30											
	18:40 - 20:00											
	20:10 - 21:30											

Рис. 6 Вкладка «Обучение» старой версии сайта САиУ

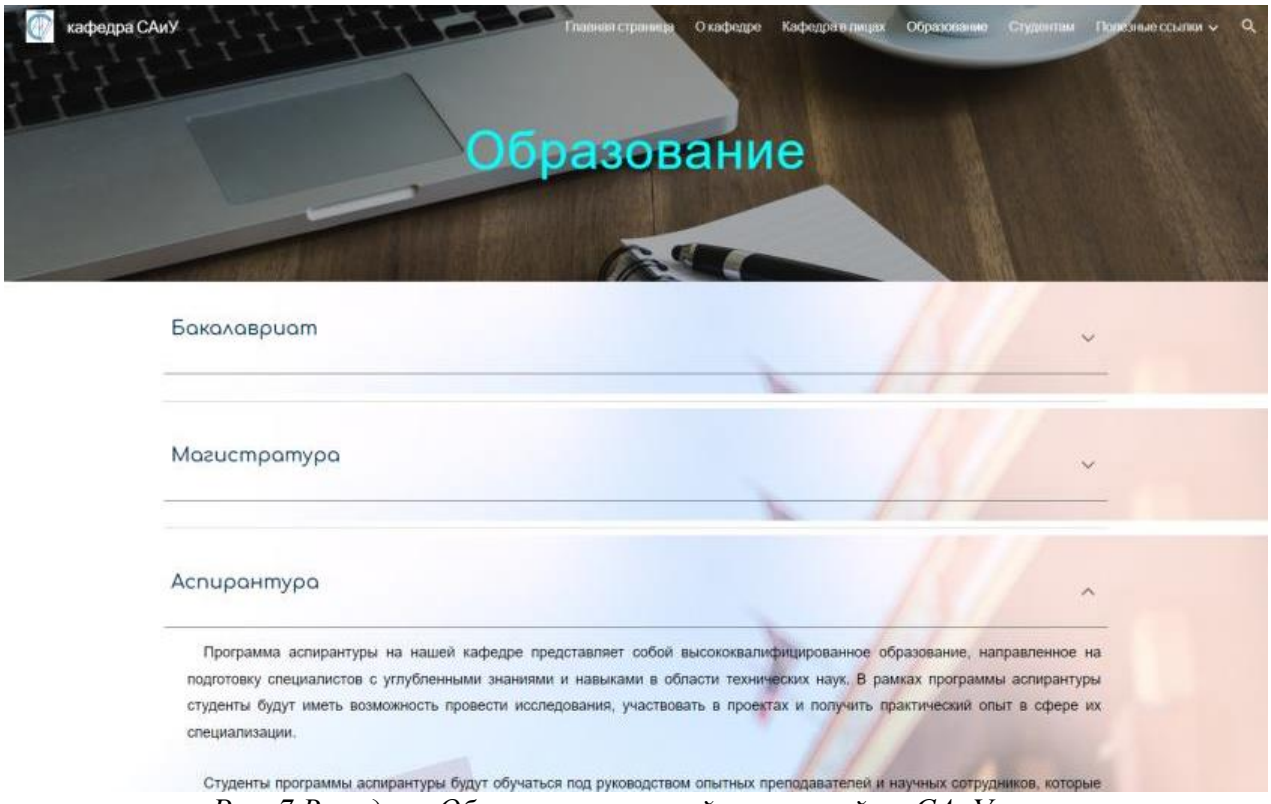


Рис. 7 Вкладка «Образование» новой версии сайта САиУ

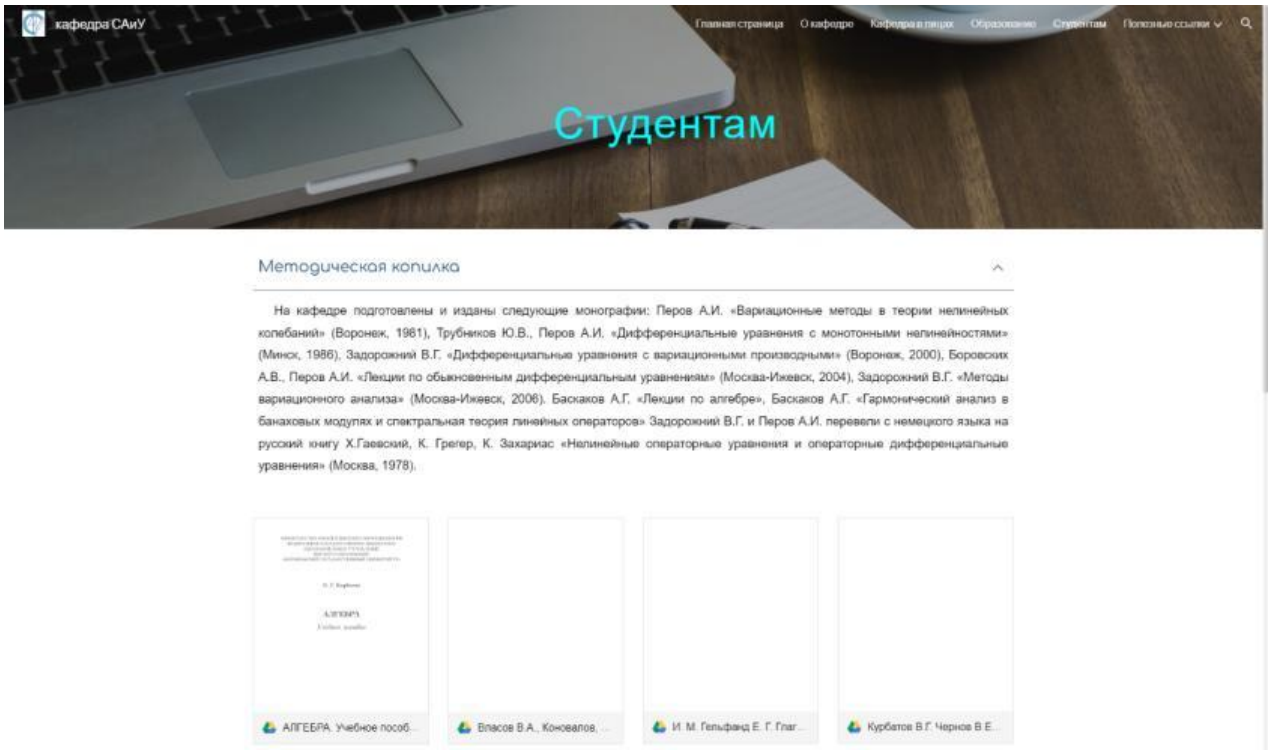


Рис. 8 Вкладка «Студентам» новой версии сайта САиУ

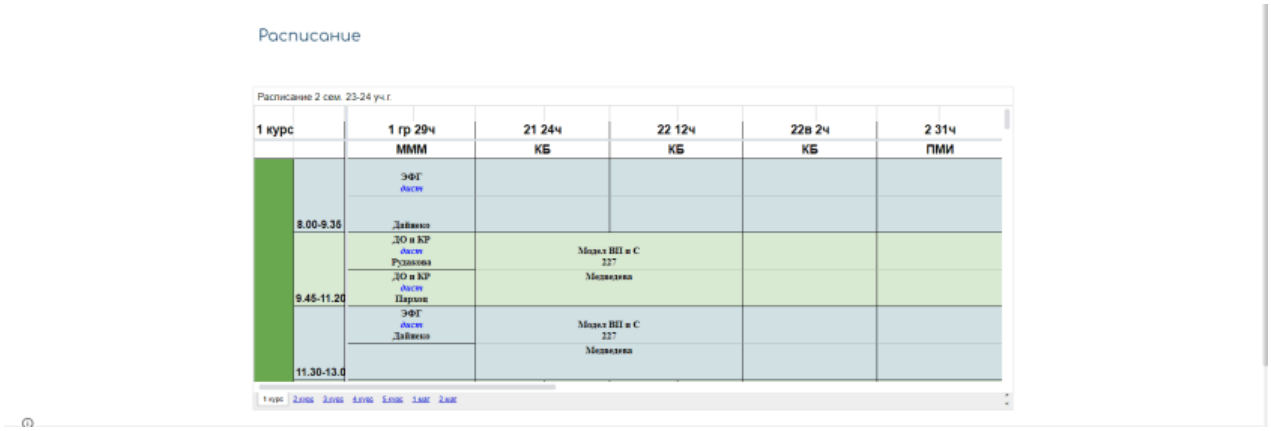


Рис. 9 Вкладка «Студентам» новой версии сайта САиУ

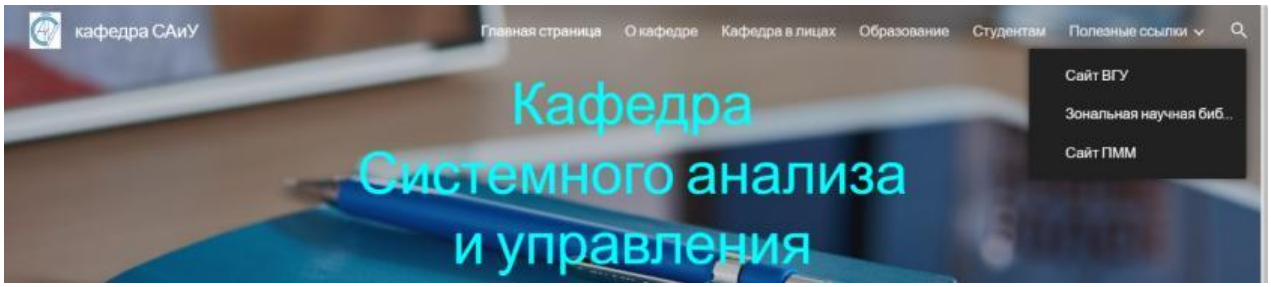


Рис. 10 Список «Полезные ссылки» новой версии сайта САиУ

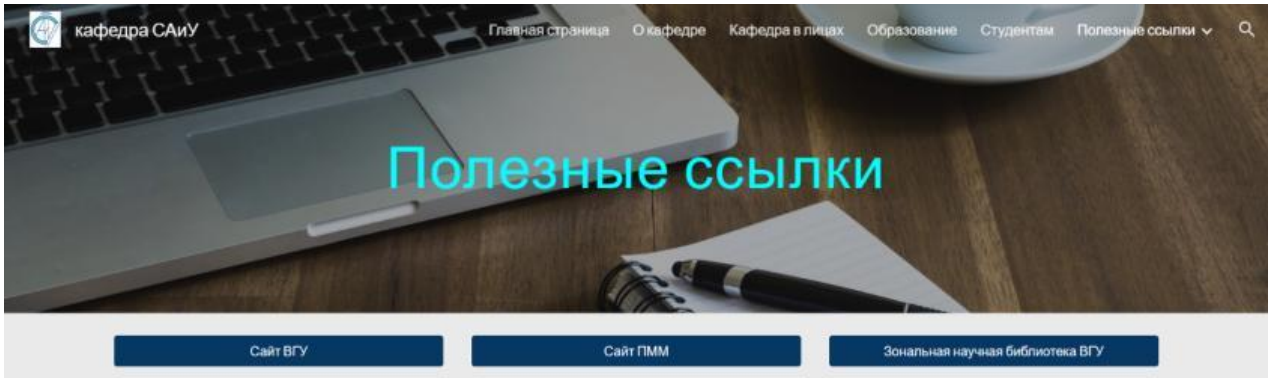


Рис. 11 Вкладка с кнопками «Полезные ссылки» новой версии сайта САиУ

Рис. 12 Раздел «Контактная информация» старой версии сайта САиУ

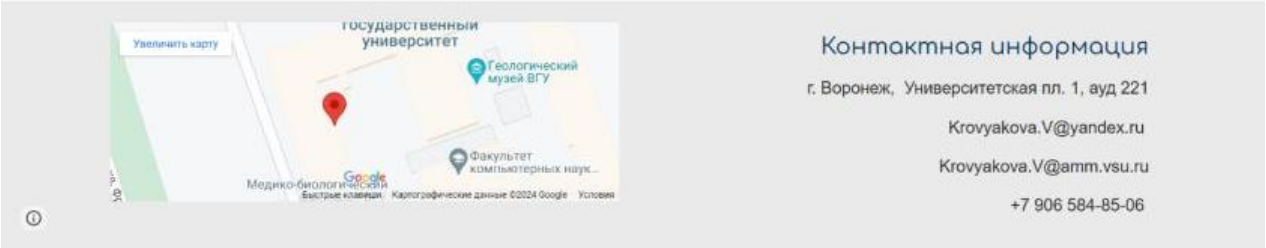


Рис. 13 Раздел «Контактная информация» новой версии сайта САиУ

Заключение

В данной научной работе было подробно описано обновление сайта кафедры учебного заведения с использованием конструктора сайтов. Использование конструктора сайтов позволило значительно упростить процесс разработки и обновления веб-ресурса, снизить затраты на найм веб-разработчиков и ускорить внедрение изменений.

Результаты работы показали, что обновленный сайт стал более современным, удобным для пользователей и функциональным. Введение нового раздела «Студентам», в котором представлены подразделы «Методическая копилка» и «Расписание», обновление разделов «Образование» и «Кафедра в лицах», интеграция Гугл-карт помогли привлечь внимание посетителей и повысить их вовлеченность.

Таким образом, использование конструктора сайтов для обновления веб-ресурса кафедры учебного заведения оказалось эффективным и позволило достичь поставленных целей в сокращенные сроки.

Литература

1. Сайт конструктора LPGenerator: https://lpgenerator.ru/
2. Сайт конструктора Tilda: https://tilda.cc/ru
3. Сайт конструктора ReadyMag: https://readymag.com/
4. Сайт конструктора Setup: https://www.setup.ru/
5. Сайт конструктора uCoz: https://www.ucoz.ru/
6. Сайт конструктора Google Sites: https://sites.google.com

УДК 519.16

ПОИСК ТОЧКИ ПЕРЕХОДА: БЫСТРЫЙ ПОИСК ПУТИ A* ДЛЯ СЕТОК С ОДНОРОДНОЙ СТОИМОСТЬЮ И ЕГО ПРИМЕНЕНИЕ

К.Р. Корчагина, О.В. Авсеева

Воронежский государственный университет

Введение

Алгоритмы поиска пути играют важную роль во многих областях компьютерных наук, от компьютерной графики до искусственного интеллекта и робототехники. Они используются для решения задач, которые включают поиск кратчайшего пути между двумя точками в некотором пространстве. Пространство, в котором происходит поиск, обычно представляется в виде графа. Вершины графа представляют собой точки или места, а ребра графа представляют собой пути или дороги между этими точками. Веса на ребрах графа могут представлять расстояния или стоимости перемещения между вершинами.

Целью алгоритма поиска пути является нахождение кратчайшего или наиболее экономичного пути от начальной вершины до конечной вершины. Такие алгоритмы используются во многих приложениях. Например, в системах навигации для автомобилей они используются для определения самого быстрого или самого короткого пути до места назначения. В компьютерных играх они помогают в определении пути, которым должны следовать персонажи. В области искусственного интеллекта они используются для планирования действий и оптимизации решений [1].

Алгоритм A* был разработан как усовершенствование оригинального алгоритма поиска по графам Эдсгера Дейкстры, который широко известен как алгоритм Дейкстры [1,2]. A* смог достичь этого благодаря использованию эвристики для разумного выбора путей, которые быстрее приведут к цели. Это позволяет отсеивать соседние объекты с более высокой стоимостью, что приводит к обнаружению оптимального пути.

Этот алгоритм оказал значительное влияние на разработку игр и поиск путей роботами. Новые подходы могут еще больше улучшить его производительность. Используя расширение траектории, точки перехода и уменьшение симметрии, алгоритм поиска точек перехода может стать будущим искусственного интеллекта в играх и робототехнике [3].

В настоящей статье рассматривается процесс реализации поиска игрока врагами в игре на движке Unity3D с помощью алгоритма Поиска точки перехода.

1. Описание алгоритма

Поиск точек перехода (JPS) – это уникальный алгоритм, который ускоряет поиск путей на сетчатых картах с равномерной стоимостью. JPS работает быстрее и мощнее, и может последовательно ускорять поиск на порядок и более. JPS не требует предварительной обработки или дополнительных затрат памяти и легко комбинируется с большинством существующих методов ускорения, включая абстракцию и эвристику памяти.

Поиск пути с поиском точки перехода JPS может быть реализован как оптимизация алгоритма A* с небольшими изменениями. JPS отлично работает на больших открытых участках карты. Именно в этих открытых областях JPS может пропустить или перепрыгнуть

через большое количество промежуточных узлов, которые в противном случае были бы рассмотрены с помощью традиционного алгоритма A*. Распознавание симметрии позволяет JPS также устранить многие другие потенциальные узлы. Уделяя немного больше внимания вычислениям на каждом рассмотренном узле, JPS может исключить большое количество потенциальных узлов пути.

Далее опишем механические детали и алгоритмические свойства поиска точки перехода.

JPS [3] может быть описан в терминах двух простых правил сокращения, которые применяются рекурсивно во время поиска: одно правило предназначено для прямых шагов, а другое – для диагональных движений. Ключевая интуитивная идея в обоих случаях заключается в том, чтобы сократить набор ближайших соседей вокруг узла, стремясь доказать, что существует оптимальный путь (симметричный или иной) от родительского узла до каждого соседнего узла, и этот путь не предполагает посещения текущего узла. На рисунке 1 проиллюстрирована основная идея.

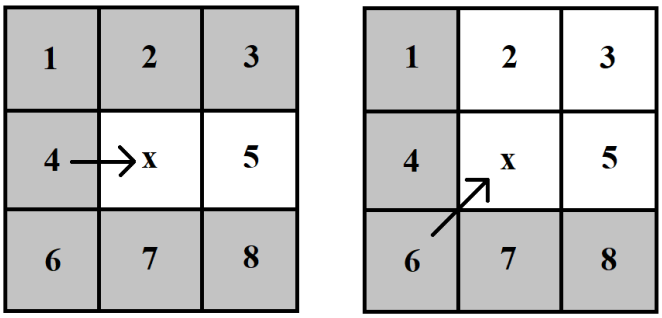


Рис. 1. Отсеченный сосед

X – текущая рассматриваемая точка. Стрелка указывает направление перемещения от его родительского узла, которое может быть прямым или диагональным. В обоих случаях мы можем сразу удалить все серые соседние узлы, потому что оптимальным путем к ним можно добраться от родительского узла x, без необходимости проходить через узел x.

Мы будем называть набор узлов, которые остаются после отсечения, настоящими соседями текущего узла. На рисунке 1 они отмечены белым цветом. В идеале набор настоящих соседей учитывается только во время просмотра. Однако в некоторых случаях, в зависимости от того, какой из этих наборов будет использоваться, наличие препятствий может означать, что нам также нужно рассмотреть небольшой набор из k дополнительных узлов. Эти узлы являются вынужденными соседями текущего узла. На рисунке 2 проиллюстрирована эта идея.

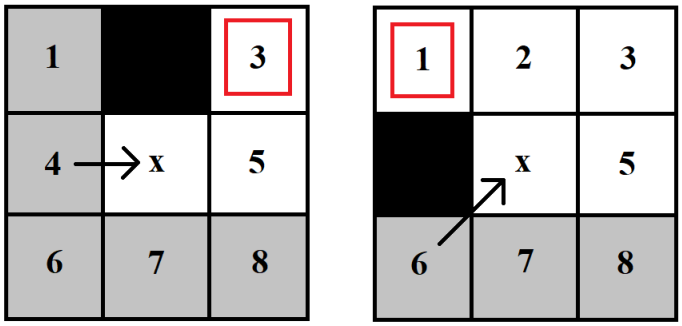


Рис. 2. Принужденный сосед

X – текущая рассматриваемая точка. Стрелка указывает направление перемещения от родительского узла – прямое или диагональное. Когда x находится рядом с препятствием, выделенные соседи не могут быть отсечены; любой альтернативный оптимальный путь от родительского элемента x к каждому из этих узлов блокируется.

Мы применяем эти правила сокращения во время поиска следующим образом: вместо генерации каждого настоящего и принудительного соседа рекурсивно сокращаем набор соседей вокруг каждого такого узла. Цель состоит в устранении симметрий рекурсивным методом, "перепрыгивая" через все узлы, до которых оптимальным образом можно добраться путем, не заходящим в текущий узел. Останавливаем рекурсию, когда натыкаемся на препятствие или, когда находим так называемую точку перехода – преемника.

Точки перехода интересны тем, что у них есть соседи, до которых невозможно добраться альтернативным симметричным путем: оптимальный путь должен проходить через текущий узел.

Детали алгоритма рекурсивного сокращения довольно просты: чтобы обеспечить оптимальность, нам нужно только задать порядок обработки настоящих соседей (прямые переходы перед диагональными). На рисунке 3 приведены два примера алгоритма сокращения в действии. В первом случае мы определяем точку перехода, повторяя движение по прямой; во втором случае мы определяем точку перехода, повторяя движение по диагонали.

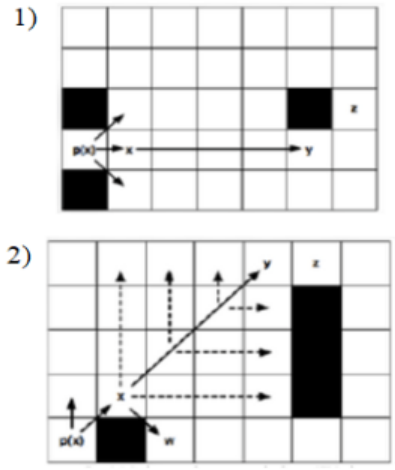


Рис. 3. 1) Прямые переходы; 2) Диагональные переходы

X – текущая рассматриваемая точка, p(x) является его родительским узлом.

1) Рекурсивно применяем правило отсечки и определяем y как точку перехода, которая следует за x. Этот узел интересен тем, что у него есть соседний узел z, до которого невозможно добраться оптимальным образом, кроме как по пути, который ведет к x, а затем к y. Промежуточные узлы никогда явно не генерируются и даже не вычисляются.

2) Рекурсивно применяем диагональные правила отсечки и определяем y как точку перехода, следующую за x. Перед каждым шагом по диагонали мы сначала повторяем прямую (пунктирные линии). Только если обе прямые рекурсии не выявляют точку перехода, снова переходим по диагонали. Узел w, который является просто вынужденным соседом x, генерируется как обычно.

2. Реализация

Для поиска игрока врагами в игре на движке Unity3D реализованы:

1. класс для точек, с помощью которых будет определяться путь;
2. класс, в котором будут прописаны настройки поиска пути;
3. класс, для хранения информации о мире, карте;
4. функция получения маршрута.

Задача была реализована с использованием языка программирования C#.

Для реализации функции получения маршрута необходимы два основных алгоритма: Определения приемника и Функция прыжка [4,5].

```
1. successors(x) ← ∅
2. neighbours(x) ← prune(x, neighbours(x))
3. for all n ∈ neighbours(x) do
4.       n ← jump(x, direction(x, n), s, g)
5.       add n to successors(x)
6. return successors(x)
```

Рис. 4. Определение приемника

Этот алгоритм описывает процесс поиска преемника для текущей точки в пространстве. Сначала множество соседей, непосредственно примыкающих к текущей точке x, обрезается (строка 2). Затем, вместо добавления каждого соседа n в множество преемников (successors) для x, мы пытаемся "перепрыгнуть" к точке, которая находится дальше, но лежит относительно направления x к n (строки 3-5).

Например, если ребро (x, n) представляет собой движение по прямой вправо от x, то мы смотрим на точку прыжка непосредственно справа от x. Если такая точка существует, то она добавляется вместо n в набор преемников. Если же до точки прыжка дойти не получается, то преемник не добавляется. Процесс продолжается до тех пор, пока не будут обработаны все соседи, и затем алгоритм возвращает список всех преемников для x (строка 6).

Чтобы найти отдельные преемники для точки прыжка, будем использовать алгоритм на рисунке 5. Он требует точки отсчета x, направления движения d, а также начальной точки s и целевой точки g.

```
1.  n ← step(x, d)
2.  if n – препятствие или находится вне графа then
3.        return null
4.  if n = g then
5.        return n
6.  if ∃ n' ∈ neighbours(n) такие что n' – принуждённый then
7.        return n
8.  if d – диагональное then
9.        for all i ∈ {1,2} do
10.             if jump(n, di, s, g) не является null then
11.                   return n
12. return jump(n, d, s, g)
```

Рис. 5. Функция прыжка

Алгоритм пытается определить, имеет ли x точку для прыжка среди преемников, перемещаясь в направлении d (строка 1). Затем он проверяет, удовлетворяет ли точка n критериям точки прыжка. Если это так, то n обозначается как точка прыжка и возвращается (строки 5, 7 и 11).

Если n не является точкой прыжка, алгоритм рекурсивно повторяется, и новая точка отсчета становится n (строка 12). Рекурсия останавливается, когда встречается препятствие и никакие дальнейшие действия не могут быть предприняты (строка 3). Перед каждым диагональным шагом алгоритм должен обнаружить точки прыжка в прямых направлениях (строки 9-11).

В результате работы описанного выше алгоритма получается сцена, представленная на рисунках 6-7. На рисунке 6 показано начальное положение врага. Далее на рисунке 7 показан поиск игрока врагом и обход врагом препятствий.

Рис. 6. Результат работы программы

Рис. 7. Результат работы программы

Заключение

В данной статье была рассмотрена задача реализации поиска игрока врагами в игре на движке Unity3D с помощью алгоритма Поиска точки перехода. Был рассмотрен сам алгоритм, который использует алгоритм A* в своей основе. Были рассмотрены алгоритмические свойства, которые показывают преимущества JPS над алгоритмом A*.

Список использованных источников

1. Нильсон Н. Искусственный интеллект: методы поиска решений = Problem-solving Methods in Artificial Intelligence / Пер. с англ. В. Л. Стефанюка; под ред. С. В. Фомина. — М.: Мир, 1973. — 272 с.
2. Лорьер Ж.-Л. Системы искусственного интеллекта / Пер. с фр. и ред. В. Л. Стефанюка. — М.: Мир, 1991. — 547 с.
3. Harabor, D. (2011, Aug 26). Shortest Path – Fast Pathfinding via Symmetry Breaking. Режим доступа: https://harablog.wordpress.com/2011/08/26/fast-pathfinding-via-symmetry-breaking/ (Дата обращения: 14.04.2024).
4. Рассел С. Дж. Искусственный интеллект: современный подход = Artificial Intelligence: A Modern Approach / Рассел С. Дж., Норвиг, — Пер. с англ. и ред. К. А. Птицына. — 2-е изд.. — М.: Вильямс, 2006. — 953 с.
5. Botea A. Near Optimal Hierarchical Path-Finding. Journal of Game Development / ., Muller M., Schaeffer J. // Journal of Game Development – 2004, vol. 1, issue 1, pp. 7–28.

УДК 004.031.42

РАЗРАБОТКА АЛГОРИТМА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ НА ОСНОВЕ ЛИНГВИСТИЧЕСКОГО ПОДХОДА

А.И. Косарыч

Воронежский государственный университет

Введение

Рекомендательные системы – самый распространенный тип интеллектуальных информационных систем, который широко применяется в сфере электронной коммерции. Существует четыре основных типа рекомендательных систем, базирующихся на следующих технологиях: фильтрация, основанная на контенте; коллаборативная фильтрация; фильтрация, основанная на знаниях; гибридные рекомендательные системы. Рекомендательные системы помогают пользователям находить подходящие товары с помощью рекомендаций, основанных на информации из различных источников, при этом в большинстве случаев информация является числовой. Однако для повышения степени выразительности, а, следовательно, и точности предлагаемых рекомендаций целесообразно использовать лингвистическую модель, которая актуальна для обработки экспертной информации. Именно такая информация, полученная от пользователей, обрабатывается в рекомендательных системах. К основным процессам в рекомендательных системах относятся: *процесс профилирования*, направленный на создание профиля пользователя – информационной структуры, которая выражает предпочтения пользователя на основе представленных им примеров; *процесс разработки рекомендаций* – система измеряет сходство между профилем пользователя и элементами из базы данных о товарах, а затем ранжирует эти объекты в соответствии с показателем схожести и рекомендует наиболее схожие. Цель статьи заключается в разработке алгоритмов, реализующих указанные процессы.

1.Алгоритм формирования профиля

На текущем этапе система собирает и анализирует предпочтения пользователя, чтобы знать, какой для него требуется товар. Пусть рекомендательная система имеет базу данных, в которой товары-объекты из множества $A=\{a_1,...,a_m\}$ описаны с помощью набора атрибутов $C=\{c_1,...,c_n\}$ так, что каждому объекту a_j соответствует векторная оценка $f_j = f\left(a_j\right)=\left(v_1^j,...,v_n^j\right)$, $j=\overline{1,m}$, при этом каждая частная оценка формируется в некоторой лингвистической шкале S_k, включающей $k+1$ термов в форме нечетких чисел определенного типа. Лингвистические шкалы, используемые для формирования базы данных, будем называть базовыми. Количество термов в шкале определяет ее гранулярность. Чем больше гранулярность, тем лингвистические оценки являются менее расплывчатыми и более точными в некотором смысле. Векторную оценку f_j будем называть *вектором полезности* товара a_j [1]. Данные описания могут бы получены либо от экспертов, либо на основе опросов, проведенных в отношении этих товаров. Заметим, что каждый атрибут может быть оценен в

«своей» лингвистической шкале в соответствии с существующей степенью знаний о нем. Таким образом, база знаний содержит лингвистическое описание товаров, из которых пользователь выбирает нужный ему.

1.1 Формирование начального профиля

Для создания профиля пользователя, необходимо определить его набор предпочтений на основе выбора примеров предпочтительных товаров-объектов. Данные примеры образуют множество, которое можно рассматривать как обучающее. Пусть $U = \{u_1, ..., u_k\}$ – множество предпочтительных примеров товаров, выбранных пользователем, и для некоторого i имеем $u_i = a_j$, т.е. в качестве примера выступает товар a_j. Этот элемент описан в базе данных с помощью вектора полезности $f_j = f(a_j) = (v_1^j, ..., v_n^j)$, в котором $v_l^j \in S_k$, k – количество термов в лингвистической шкале. Данный пример определяет начальный профиль пользователя, который мы обозначим как $P_r^0 = (p_1^r, ..., p_n^r)$, где $p_s^r = v_s^j$ для всех $s = \overline{1,n}$, $r \in \{1, ..., L\}$, L – количество предпочтительных примеров, указанных пользователем $P^0 = \{P_1^0, ..., P_L^0\}$. Таким образом, указывая на предпочтительные товары, пользователь формирует свой начальный профиль P^0, при этом описания примеров совпадают с теми, которые имеются в базе данных [2].

1.2 Модификация начального профиля

После определения начального профиля пользователя P^0 система предоставляет ему возможность изменять одно или несколько значений его профиля в целях совершенствования процесса вынесения рекомендаций. Вероятно, знания пользователя о заданном атрибуте предпочтительного примера отличаются от знаний специалистов о нем, т.е. лингвистическая оценка какого-либо атрибута предпочтительного примера требует уточнения от пользователя с использованием другой лингвистической шкалы. В этом случае система предоставляет пользователю возможность использовать другой набор лингвистических термов, более подходящий для выражения его знаний об атрибуте предпочтительного примера. Модификация начального профиля позволяет уточнить предпочтения пользователя и на их основе выдать наиболее релевантные рекомендации. Таким образом, в начальном профиле для некоторых предпочтительных примеров пользователь может изменить частные оценки, используя другие лингвистические шкалы, отличающиеся от базовых степенью гранулярности, т.е. количеством используемых лингвистических термов. При переходе к другой лингвистической шкале описание примеров из начального профиля меняется с учетом новой лингвистической шкалы, используемой пользователем. Тем самым, будет сформирован новый профиль пользователя $\tilde{P} = \{\tilde{P}_1, ..., \tilde{P}_L\}$, где в каждом примере $\tilde{P}_r$ частная оценка $\tilde{p}_j^r$ будет изменена на оценку в лингвистической шкале $\tilde{S}_d$ с количеством термов d. Если пользователь не изменил оценку, то $\tilde{p}_j^r = p_j^r$ [3].

1.3 Унификация лингвистических шкал

В результате модификации начального профиля будет сформирован индивидуальный профиль пользователя, включающий совокупность примеров (товаров), которые соответствуют

предпочтениям пользователя в наибольшей степени, при этом оценки атрибутов будут оценены в лингвистических шкалах с различным количеством термов. В различных шкалах один и тот же терм имеет различный семантический смысл, так что информация для выработки рекомендаций является разнородной. Понятно, что для обработки такой информации и обеспечения сопоставимости ее необходимо унифицировать, т.е. привести к некоторой универсальной шкале. *Универсальная лингвистическая шкала* может быть выбрана из уже существующих или построена, но в любом случае ее размерность должна быть не меньше, чем размерность уже используемых для оценки шкал.

Для перехода из шкалы S_j в универсальную шкалу S_U будем использовать *функцию трансформации* $\tau_{S_j \to S_U} : S_j \to F(S_U)$, такую что

$$\tau_{S_j \to S_U}(S_i^j) = \left\{\left(S_k^U / \alpha_k^i\right)\right\}_{k=\overline{0,N}},$$

$$\tau_{S_j \to S_U}(S_i^j) = \left\{\left(S_k^U / \alpha_k^i\right)\right\}_{k=\overline{0,N}},$$

$$\alpha_k^i = \max_y \min\left\{\mu_{S_i^j}(y), \mu_{S_k^U}(y)\right\},$$

где $\mu_{S_i^j}(y), \mu_{S_k^U}(y)$ – функции принадлежности нечетких множеств, определяющих термы $S_i^j \in S^j$ и $S_k^U \in S_U$ соответственно, $F(S_U)$ – нечеткое подмножество универсальной лингвистической шкалы S_U [4].

Пусть эксперт E_j в процедуре оценивания использует шкалу $S_j = \left\{S_0^j, S_1^j, S_2^j, S_3^j, S_4^j\right\}$, которая содержит 5 термов, причем каждому терму соответствует нечеткое треугольное число

$$S_0^j = (0,0,0.25), S_1^j = (0,0.25,0.5), S_2^j = (0.25,0.5,0.75), S_3^j = (0.5,0.75,1), S_4^j = (0.75,1,1).$$

Предположим, что универсальная шкала содержит 7 термов

$$S_U = \left\{S_0^U, S_1^U, S_2^U, S_3^U, S_4^U, S_5^U, S_6^U, S_7^U\right\}$$

также с треугольными функциями принадлежности

$$S_0^U = (0,0,0.16), S_1^U = (0,0.16,0.34), S_2^U = (0.16,0.34,0.5), S_3^U = (0.34,0.5,0.66),$$

$$S_4^U = (0.5,0.66,0.84), S_5^U = (0.66,0.84,1), S_6^U = (0.84,1,1).$$

После применения преобразования $\tau_{S_j \to S_T}$ получим нечеткое подмножество шкалы S_U. Например, для термов S_0^j и S_1^j функции принадлежности будут иметь следующий вид:

$$\tau_{S_j \to S_U}\left(S_0^j\right) =$$

$$= \left\{\left(S_0^U / 1\right), \left(S_1^U / 0.58\right), \left(S_2^U / 0.18\right), \left(S_3^U / 0\right), \left(S_4^U / 0\right), \left(S_5^U / 0\right), \left(S_6^U / 0\right)\right\},$$

$$\tau_{S_j \to S_U}(S_1^j) =$$

$$= \left\{\left(S_0^U / 0.39\right), \left(S_1^U / 0.85\right), \left(S_2^U / 0.85\right), \left(S_3^U / 0.39\right), \left(S_4^U / 0\right), \left(S_5^U / 0\right), \left(S_6^U / 0\right)\right\}.$$

На рис.1 изображены обе шкалы S^j (сплошная линия) и S^U (пунктирная линия).

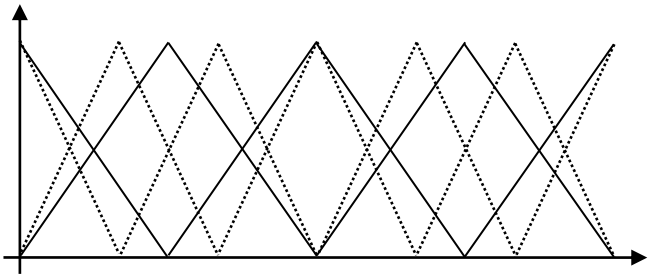


Рис. 1 Шкалы S^j и S^U

На рис. 2 представлен график функций принадлежности полученного нечеткого множества для терма S_0^j.

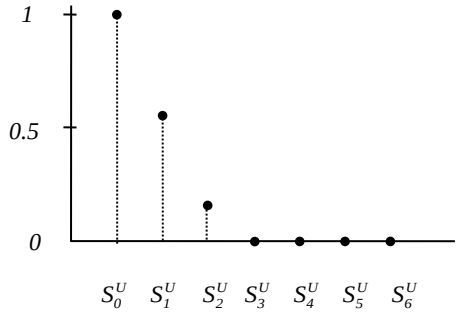


Рис.2 График функции для терма S_0^j

Будем считать, что универсальная лингвистическая шкала представляет собой множество термов $S^U = \{S_0^U, ..., S_N^U\}$, где $N+1$ – размерность лингвистической шкалы.

2. Алгоритм формирования рекомендаций

На этой стадии система вычисляет, насколько близки товары-объекты к профилю пользователя, с помощью измерения сходства. Для выполнения этого этапа система оценивает сходство между всеми элементами базы данных $A = \{a_1, \ldots, a_n\}$ и профилем пользователя после следующих шагов:

1. *Вычисление сходства между каждым объектом и профилем пользователя:* система вычисляет степень сходства между профилем пользователя и элементами базы данных.

2. *Предоставление рекомендаций:* наконец, система предлагает пользователю наиболее подходящие товары-объекты, то есть наиболее близкие к профилю пользователя.

2.1 Вычисление сходства между профилем пользователя и товарами из базы данных

Пусть $a_j \in A$ – некоторый товар-объект из базы данных, ему соответствует векторная оценка $f_j = f(a_j) = (v_1^j, ..., v_n^j)$, где каждая частная оценка v_i^j есть нечеткое подмножество универсальной лингвистической шкалы вида $\{\mu_{v_i^j}(k) / S_k^U\}_{k=\overline{0,N}}$. Выберем товар $\tilde{P}_r$ из профиля пользователя. В универсальной лингвистической шкале каждая частная оценка $\tilde{p}_l^r$ также представляет собой нечеткое подмножество $\{\mu_{\tilde{p}_l^r}(k) / S_k^U\}_{k=\overline{0,N}}$. Здесь $\mu_{v_i^j}(k)$ и $\mu_{\tilde{p}_l^r}(k)$ – степени принадлежности соответствующих оценок терму S_k^U. Задача сводится к определению степени сходства или несходства данных нечетких множеств.

Заметим, что для вычисления данных характеристик можно использовать различные метрики, в том числе известные функции расстояния. Рассмотрим подход, основанный на дефазификации.

Пусть A – нечеткое множество с функцией принадлежности $\mu_A : \square \to [0,1]$, и существует интервал $[a,b] \subseteq \square$ такой, что $Supp(A) = [a,b]$.

Дефазификацией называется отображение $D : \mu_A \to \square$, которое каждой функции принадлежности μ_A ставит в соответствие число $D(\mu) \in [a,b]$. Таким образом, дефазификация устанавливает связь между нечетким множеством и некоторым числовым значением, которое принадлежит области определения функции принадлежности нечеткого множества [4]. В настоящее время существует несколько методов дефазификации, которые используются в приложениях для вычисления нечетких выводов. Наиболее распространенными являются следующие:

- *метод центра тяжести*

$$D(\mu) = \frac{\int_{\min}^{\max} x \cdot \mu(x)\,dx}{\int_{\min}^{\max} \mu(x)\,dx}$$

или в случае дискретного представления множеств

$$D(\mu) = \frac{\sum_{i=1}^{n} x_i \cdot \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)};$$

- *метод центра площади*

$$D(\mu) = \left\{ u : \int_{\min}^{u} \mu(x)\,dx = \int_{u}^{\max} \mu(x)\,dx \right\};$$

- *метод левого (правого) модального значения*

$$D(\mu) = \min\{x_m\} \ \left(D(\mu) = \max\{x_m\}\right),$$

где x_m – модальное значение нечеткого множества.

Поскольку в нашем случае нечеткое множество определено на дискретном универсальном множестве, то метод дефазификации должен быть модифицирован.

Пусть на множестве термов $\{S_h\}$ лингвистической шкалы определено нечеткое множество $A = \{\alpha_1 / S_0, ..., \alpha_N / S_N\}$. Центральным значением нечеткого множества A назовем величину

$$cv(A) = \frac{\sum_{k=0}^{N} \alpha_k \cdot ind(S_k)}{\sum_{k=0}^{N} \alpha_k},$$

где $ind(S_k) = k$.

Пусть заданы два вектора $X = (x_1, ..., x_n)$ и $Y = (y_1, ..., y_n)$, компоненты которых являются нечеткими подмножествами множества термов лингвистической шкалы. Для каждой пары одноименных компонент, которые представляют собой оценки атрибутов товаров, можно вычислить центральное значение. Степень сходства этих значений вычислим по формуле

$$sim(x_i, y_i) = 1 - \frac{|cv(x_i) - cv(y_i)|}{N+1},$$

тогда *коэффициентом сходства* двух векторов назовем усредненное значение

$$Sim(X, Y) = \frac{1}{n}\sum_{i=1}^{n} sim(x_i, y_i).$$

Заметим, что для вычисления коэффициентов сходства можно использовать другие средние.

Результатом данного шага является множество коэффициентов сходства, которые определяют степень соответствия товаров из базы данных запросам пользователя.

2.2 Разработка рекомендаций

На данном этапе система ранжирует товары согласно значениям коэффициентов сходства. Лучшими будут те, которые ближе к профилю пользователя, то есть те, которые имеют наибольшее значение коэффициента сходства. Система будет рекомендовать топ-N товаров, которые достигают заданного порога, т.е. если один из топ-N товаров слишком далек от профиля пользователя (его степень сходства меньше порога), то этот товар не будет включен в рекомендацию [2].

Заключение

В данной статье был представлен алгоритм рекомендательной системы на основе лингвистического представления информации, причем такая информация формируется пользователем в индивидуальной лингвистической шкале, которая затем преобразуется в универсальную шкалу. Преимущество такого представления заключается в том, что мы можем собрать информацию пользователя, которая обычно связана с восприятием или вкусами, без потери выразительности или точности. Кроме того, сформирована гибкая модель работы с информацией, в которой каждый атрибут может быть оценен с помощью наиболее подходящего набора лингвистических термов, а пользователи могут использовать наборы лингвистических термов в соответствии со своими знаниями или предпочтениями.

Научный руководитель доцент, доктор техн. наук, профессор кафедры математических методов исследования операций ВГУ, Бондаренко Юлия Валентиновна.

Литература

1. Кокачев В.А. Рекомендательные системы в контексте технологий больших данных: диссертация / В.А. Кокачев. – Санкт-Петербург: Санкт-петербургский государственный университет, 2018. – Режим доступа: https://dspace.spbu.ru/bitstream/11701/12104/1/Kokachev_V.pdf (дата обращения: 14.04.2024).

2. Ricci F. Recommender systems handbook / F. Ricci, L. Rokach, B. Shapira, P. Kantor. – New York: Springer US, 2015. – 1008 p

3. Martinez L. A knowledge based recommender system with multigranular linguistic information / L. Martinez [and etc.] // International Journal of Computational Intelligence Systems. – 2008. – Vol.1, № 3. – P. 225-236.

4. Леденева Т.М. Обработка нечеткой информации: учебное пособие / Т.М. Леденева. – Воронеж: Воронежский государственный университет, 2006. – 233 с.

УДК 519.8 : 004.942

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТОМОВ ПРИ МОДЕЛИРОВАНИИ ОСОБЕЙ И ОКРУЖАЮЩЕЙ СРЕДЫ ДЛЯ СИМУЛЯЦИИ ИСКУССТВЕННОЙ ЖИЗНИ

В. С. Котенко

Воронежский государственный университет

Введение

Генетические алгоритмы могут улучшать решения реальных задач, используя упрощённый закон естественного отбора биологических видов и организмов. Генетические алгоритмы могут использоваться для интерактивного управления различными процессами, следить за оптимальной работой мультипроцессоров, их можно применять для решения некоторых экономических задач в бизнесе или техническом производстве, в том числе задач прогнозирования. С помощью генетических алгоритмов можно обучать виртуальные машины ездить или виртуальных людей ходить без чёткого описания этого процесса программным кодом, поэтому их так же можно применить для обучения робототехнических устройств корректному движению [1].

Одна из важнейших задач при работе с генетическими алгоритмами — моделирование виртуальных объектов, которые будут обучаться, а также моделирование среды, в которой будет проходить естественный отбор. Настоящая работа посвящена вопросу применения генетических алгоритмов в задаче симуляции искусственной жизни.

1. Основные понятия

Матрица мира — это модель окружающей среды, состоящей из квадратных ячеек (рис. 1). Ячейка может быть пустой, на ней может располагаться ресурс, преграда или особь.

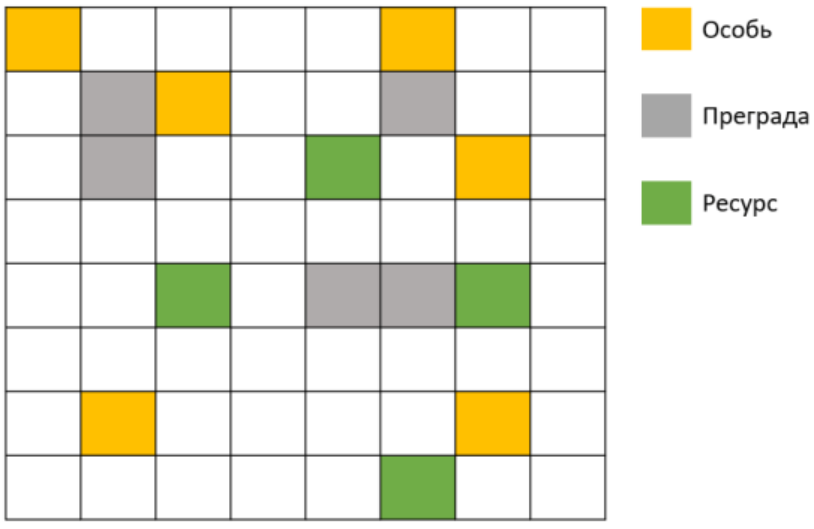


Рис. 1. Модель окружающей среды

Особь — это модель живого организма, которая перемещается по матрице мира. Она может наступать на ресурсы для их сбора, но не может пройти сквозь препятствие или другую особь. Границы мира также считаются преградами и выход за пределы среды невозможен.

Популяция — это список из особей. В процессе развития особи популяции конкурируют между собой за ресурсы, а самые успешные становятся родителями для особей следующей популяции.

На этапе моделирования мира задаются такие параметры, как количество всех особей, количество ресурсов и плотность препятствий [2].

2. Моделирование особи

Особь характеризуется набором своих генов [3]. Гены хранятся в односвязном кольцевом списке и являются неотрицательными целыми числами. Размер списка и диапазон значений генов зависит от количества действий, которые способна выполнять особь. Настоящая работа рассматривает модель, способную на три основных действия: движение, изучение, ожидание. Пример списка генов приведён на рис. 2.

Значение	**8**	**14**	**35**	**11**	**…**	**24**
Индекс	0	1	2	3	…	35

Рис. 2. Список генов особи

Значение активного гена определяет действие, которое будет выполнять особь. Активным геном является тот, на котором в ход особи установлен указатель. Список доступных действий:

• 0 – 7: переход особи на одну из восьми окружающих её ячеек (рис. 3).

• 8 – 15: проверка одной из 8 окружающих особь ячеек на наличие там другой особи, препятствия или ресурса.

• 16 – 35: особь выжидает. Указатель сдвигается на количество ячеек, соответствующее значению гена, при этом считается, что после 35 стоит 0.

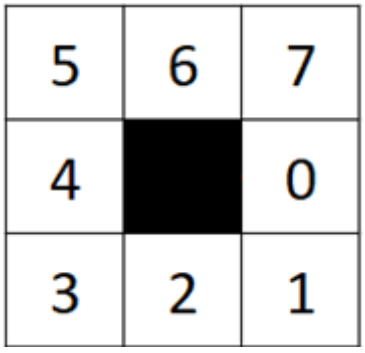


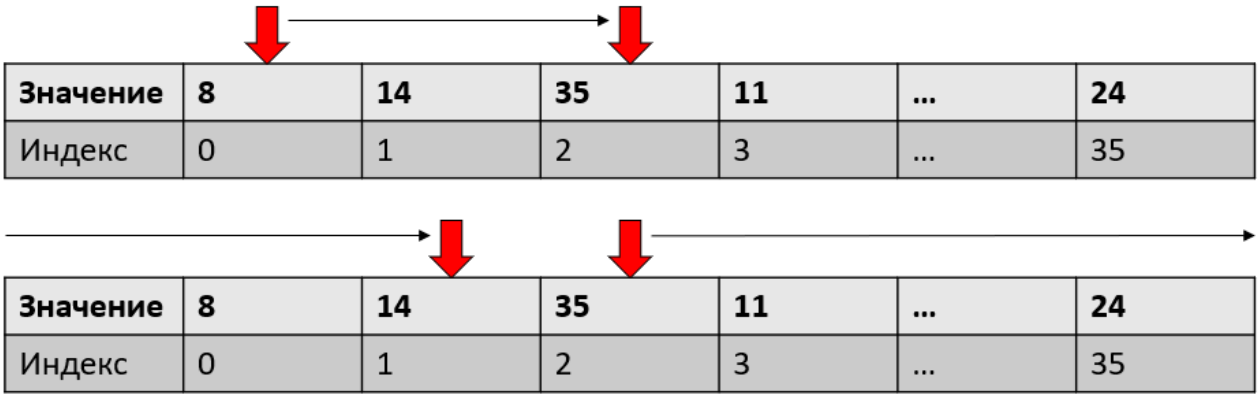
Рис. 3. Направления перемещения особи

Результат команд перемещения и проверки зависит от внешней среды, поэтому реакция также должна отличаться. Существует четыре результата для этих команд:

1. На целевой клетке расположен ресурс. Указатель сдвигается на 1 ячейку.
2. На целевой клетке построена стена. Указатель сдвигается на 2 ячейки.
3. Особь сталкивается с другой особью. Указатель сдвигается на 3 ячейки

4. Целевая клетка пустая. Указатель сдвигается на 4 ячейки.

Пример работы указателя приведён на рис. 4. При этом можно наблюдать следующее: указатель изначально расположен на нулевом индексе, особь двигается в соответствующую сторону, сталкивается с другой особью и указатель сдвигается на два индекса. В свою следующую активацию особь выжидает, а указатель движется на 35 индексов и устанавливается на первый.



Значение	**8**	**14**	**35**	**11**	**...**	**24**
Индекс	0	1	2	3	...	35

Значение	**8**	**14**	**35**	**11**	**...**	**24**
Индекс	0	1	2	3	...	35

Рис. 4. Функция указателя

У особей есть параметр голода, который увеличивается сам по себе на величину, зависящую от совершаемого особью действия, а уменьшается при поглощении ресурсов. В данной работе голод является основным параметром фитнес-функции [4, 5].

4. Эволюция популяции

Для заполнения матрицы генов первого поколения реализованы два способа: случайное заполнение и алгоритм BFS. С его помощью для каждого бота ищется ресурс рядом с ним, после чего матрица генов заполняется значениями, позволяющими боту достигнуть найденный ресурс.

Популяция живёт, пока живо больше половины особей, после чего оставшаяся половина скрещивается между собой для создания новой популяции.

Функцией приспособленности является параметр голода, чем он меньше, тем более приспособленной считается особь. Мутация – замена числа в случайной ячейке списка генов на другое случайное число. Для репродукции используется четыре типа кроссинговеров: одноточечный, двухточечный, равномерный и циклический [6, 7].

5. Анализ результатов вычислительного эксперимента

Так как в симуляции ограниченное количество ресурсов, оптимальным результатом работы алгоритма является «все ресурсы поглощены, все особи мертвы».

В виду отсутствия точного решения задачи, для оценки эффективности эволюции особей в процессе эволюции будет сравниваться эффективность выживания самой первой популяции с популяцией, прошедшей сто циклов скрещивания одноточечным кроссинговером и мутаций.

В табл. 1 используются средние данные после 100 запусков симуляции.

Таблица 1

Влияние количества преград и ресурсов

	Количество преград	Количество ресурсов	Количество оставшихся ресурсов	Количество прожитых ходов
Первая популяция	10	10	7,3	65,8
Сотая популяция	10	10	5,6	92,5
Первая популяция	30	10	8,5	50
Сотая популяция	30	10	7,4	100,7
Первая популяция	10	20	16,5	63,3
Сотая популяция	10	20	10,1	107,8
Трёхсотая популяция	10	10	5,9	89,8

В среднем после эволюции популяции живут и действуют в полтора раза эффективнее, увеличение количества ресурсов действительно влияет на продолжительность жизней популяции, а увеличение числа препятствий понижает эффективность первой популяции, но популяции после эволюции успешно приспосабливаются к этому.

В следующих экспериментах исследуется влияние различных параметров на эволюционный процесс. Успешность выживания оценивается по количеству ходов, совершённых особями. Все приведённые значения являются усреднёнными на основе 20 проведённых запусков проекта.

Вычислительный эксперимент №1 (см. табл. 2):

- размер мира – 15;
- количество еды – 30;
- количество стен – 10;
- количество поколений – 10;
- погибшие особи считаются препятствием;
- случайный мир для каждого метода скрещивания.

Таблица 2

Экспериментальные данные №1

	1	2	3	4	5
Одноточечный	100	86.65	95.4	101.35	103
Двухточечный	95.1	104.65	99.6	105	91.45
Равномерный	97.25	99.55	110.4	88.25	118.35
Циклический	84.8	110.2	119.35	83.5	113.45

Вычислительный эксперимент №2 (см. табл. 3):

- погибшие особи не считаются препятствием;
- одинаковый мир для каждого метода.

Таблица 3

Экспериментальные данные №2

	Размер мира – 10 Количество особей – 20 Количество еды – 25 Количество преград – 25	Размер мира – 20 Количество особей – 60 Количество еды – 100 Количество преград – 100	Размер мира – 30 Количество особей – 120 Количество еды – 200 Количество преград – 200
Одноточечный	91.7	102.45	100.95
Двухточечный	89.75	100.6	96.8
Равномерный	100.4	121.4	123.4
Циклический	108.0	118.0	130.65

На основе полученных данных можно сделать следующие выводы: выбор метода скрещивания на данном этапе не оказывается значительного влияния на развитие эволюции, но, в среднем, двухточечный и циклический метод показывают немного лучшие результаты.

Вычислительный эксперимент №3 (см. табл. 4):

- размер мира – 15;
- количество особей – 16;
- количество еды – 20;
- количество стен – 30;
- погибшие особи не считаются препятствием;
- случайный мир для каждого метода скрещивания.

Таблица 4

Сравнение методов заполнения генов начальной популяции

	BFS		Случайные значения	
Поколение	1	10	1	10
Одноточечный	52.15	99.55	72.15	101.6
Двухточечный	51.5	86.0	70.55	99.85
Равномерный	64.8	99.25	76.1	113.45
Циклический	62.45	113.35	80.95	115.3

Данный эксперимент позволяет сделать вывод, что поиск в ширину не подходит для текущих правил мира и эволюции.

Заключение

По результатам проделанной работы можно сделать следующие выводы. Представленные модели окружающей среды и живых организмов корректно реагируют на различные параметры симуляций, что показывает перспективность дальнейшего развития этого проекта. Благодаря его модульной структуре и лёгкости масштабирования возможен переход проекта к более сложным симуляциям экосистем с несколькими конкурирующими популяциями.

Изменения в параметрах мира и популяции влияют на стратегии выживания особей. Таким образом, наблюдение за генами особей из выживших популяций позволяет изучать, как проходит адаптация к разным особенностям окружающей среды.

Работа выполнена под руководством кандидата физ.-мат. наук, доцента кафедры

вычислительной математики и прикладных информационных технологий Воронежского государственного университета Королькова Олега Геннадьевича.

Литература

1. Генетический алгоритм: Википедия. – URL: https://ru.wikipedia.org/wiki/Генетический_алгоритм (дата обращения 19.04.2024).
2. ReinLife: GitHub. – URL: https://github.com/MaartenGr/ReinLife (дата обращения 19.04.2024).
3. Курейчик В.В. Теория эволюционных вычислений / В.В. Курейчик, В.М. Курейчик, С.И. Родзин. – Москва: Физматлит, 2012. – 260 с.
4. Генетический алгоритм. Просто о сложном: Хабр. – URL: https://habr.com/ru/articles/128704/ (дата обращения 19.04.2024).
5. Лекция 1: Введение. Основы генетических алгоритмов: ИНТУИТ. – URL: https://intuit.ru/studies/courses/14227/1284/lecture/24168?page=6 (дата обращения 19.04.2024).
6. Генетический алгоритм: Youtube. – URL: https://www.youtube.com/watch?v=SfEZSyvbj2w (дата обращения 19.04.2024).
7. Moraglio, A. Geometric crossover for thepermutation representation / A. Moraglio, R. Poli // Intelligenza Artificiale. – 2011. – Т. 5. – С. 49–63.

УДК: 004.08+004.048

АНАЛИЗ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ АВТОМАТИЗАЦИИ КЛИЕНТСКОГО СЕРВИСА

Г. Ю. Котляров

Воронежский государственный университет

Введение

В наше время клиентский сервис играет ключевую роль в успешной деятельности любого бизнеса. Клиенты ожидают быстрых, эффективных и персонализированных решений для своих потребностей, и именно в этом контексте становится ясной важность непрерывного совершенствования клиентского опыта.

На протяжении последних десятилетий клиентский сервис претерпел значительные изменения, прежде всего благодаря цифровой трансформации и внедрению современных технологий. С развитием интернета и мобильных устройств клиенты получили новые возможности для взаимодействия с компаниями: от онлайн-чатов и электронной почты до социальных сетей и мессенджеров. Это привело к росту ожиданий клиентов и усилению конкуренции между компаниями за их внимание и лояльность [1].

В условиях растущей нагрузки на клиентские сервисы и необходимости обеспечить высокое качество обслуживания при сокращении затрат автоматизация становится неотъемлемой частью стратегии многих компаний. Искусственный интеллект, в частности, играет ключевую роль в этом процессе, предоставляя возможность автоматизации рутинных операций, анализа данных и предоставления персонализированных решений для клиентов. Технологии ИИ, такие как чат-боты, виртуальные ассистенты и системы машинного обучения, позволяют компаниям значительно улучшить эффективность своих клиентских сервисов и повысить удовлетворенность клиентов [2].

1. Основные понятия

1.1. Определение искусственного интеллекта

Искусственный интеллект (ИИ) представляет собой область компьютерных наук, которая занимается созданием систем и программ, способных выполнять задачи, обычно требующие человеческого интеллекта. В широком смысле, ИИ включает в себя различные подходы и техники, такие как машинное обучение, обработка естественного языка, компьютерное зрение и много другое. Основная цель искусственного интеллекта заключается в создании систем, способных обучаться из опыта, адаптироваться к новым ситуациям и принимать решения, аналогичные решениям человека [2].

1.2. Краткий обзор истории ИИ в клиентском сервисе

Идеи, лежащие в основе искусственного интеллекта, имеют долгую историю, начиная с появления понятий и методов в 1950-х годах. Однако, применение ИИ в клиентском сервисе стало широко распространяться только в последние десятилетия, с появлением технологий и возможностей.

В начале использование ИИ в клиентском сервисе ограничивалось простыми автоматизированными системами, такими как IVR (Interactive Voice Response) системы и базовые чат-боты. Однако, с развитием машинного обучения и обработки естественного языка, возможности ИИ значительно расширились. Сегодня компании могут использовать продвинутые алгоритмы машинного обучения для анализа данных клиентов, создания персонализированных рекомендаций и предсказания поведения клиентов.

История ИИ в клиентском сервисе свидетельствует о постоянном развитии и усовершенствовании технологий, направленных на улучшение опыта клиентов и повышение эффективности обслуживания [3].

2. Технологии ИИ в клиентском сервисе

2.1. Использование чат-ботов и виртуальных ассистентов

Чат-боты и виртуальные ассистенты являются одними из наиболее распространенных приложений искусственного интеллекта в клиентском сервисе. Чат-боты представляют собой программы, которые могут автоматически взаимодействовать с пользователями через текстовые или голосовые сообщения, симулирую разговор с человеком. Они могут быть использованы для решения широкого спектра задач, включая предоставление информации о продуктах или услугах, обработку заказов, помощь в решении проблем и много другое. Виртуальные ассистенты, такие как Siri от Apple, Google Assistant, Amazon Alexa или Алиса от Яндекса, представляют собой более продвинутые версии чат-ботов, способные выполнять более сложные задачи и взаимодействовать с пользователем в более естественной форме [4].

2.2. Применение машинного обучения для анализа данных клиентов

Машинное обучение играет ключевую роль в анализе данных клиентов и предоставлении персонализированных рекомендаций. С его помощью компании могут анализировать большие объемы данных о клиентах, такие как история покупок, предпочтения, поведение и т. д., чтобы выявить паттерны и тенденции, которые могут помочь в улучшении обслуживания. Например, алгоритмы машинного обучения могут использоваться для прогнозирования предпочтений клиентов, определения их потребностей и предоставления рекомендаций по продуктам или услугам, которые наиболее вероятно заинтересуют конкретного клиента [4].

Эти технологии искусственного интеллекта не только повышают эффективность клиентского сервиса, но и способствуют улучшению опыта клиентов, предоставляя им быструю и персонализированную поддержку.

3. Преимущества автоматизации клиентского сервиса с помощью ИИ

3.1. Улучшение качества обслуживания

Автоматизация клиентского сервиса с помощью искусственного интеллекта позволяет компаниям обеспечить более высокого качество обслуживания за счет предоставления быстрых и точных решений. Чат-боты и виртуальные ассистенты, основанные на ИИ, могут оперативно отвечать на вопросы клиентов и предоставлять информацию о продуктах или услугах 24/7 без задержек и ошибок, что способствует улучшению удовлетворенности клиентов.

3.2. Сокращение времени ожидания ответа

Использование чат-ботов и автоматических систем ответов на запросы клиентов позволяет существенно сократить время ожидания ответа. Вместо того, чтобы ждать в очереди на связь с оператором контакт-центра, клиенты могут сразу получить ответ на свой вопрос или решение проблемы. Это сокращает время ожидания и ускоряет процесс обслуживания.

3.3. Повышение эффективности работы сотрудников

Автоматизация клиентского сервиса с помощью ИИ также приводит к повышению эффективности работы сотрудников. Задачи, которые ранее требовали бы большого количества времени и ресурсов, теперь могут быть автоматизированы, что позволяет сотрудникам сконцентрироваться на более сложных и важных задачах. Например, благодаря автоматическому анализу данных клиентов с использование машинного обучения, сотрудники могут получать ценные инсайты о предпочтениях и потребностях клиентов, что помогает им лучше понимать и удовлетворять потребности клиентов.

4. Вызовы и ограничения

4.1. Этические соображения при использовании ИИ

Внедрение искусственного интеллекта в клиентский сервис вызывает ряд этических вопросов и забот. Одним из основных вопросов является прозрачность и объяснимость принимаемых ИИ-системой решений. Например, при использовании алгоритмов машинного обучения для принятия решений о предоставлении кредита или определения цены страховки важно, чтобы клиенты понимали, какие факторы влияют на эти решения и как они могут повлиять на них. Еще одним важным аспектом является проблема защиты данных клиентов и конфиденциальности. Сбор, хранение и использование больших объемов персональных данных клиентов требует строго соблюдения законов о защите данных и нормативных актов о конфиденциальности [2].

4.2. Технические препятствия и ограничения

Помимо этических вопросов, существуют и технические препятствия и ограничения при использовании искусственного интеллекта в клиентском сервисе. Одним из таких ограничений является недостаточная точность и надежность алгоритмов ИИ, особенно в контексте анализа естественного языка или обработки изображений. Это может привести к неправильным решениям или неполным ответам, что негативно отразится на удовлетворенности клиентов. Кроме того, некоторые задачи могут быть трудно автоматизированы из-за их сложности или неоднозначности, что ограничивает применение ИИ в определенных областях клиентского сервиса.

5. Кейс-стади и примеры из практики

5.1. Анализ успешных примеров автоматизации клиентского сервиса

Одним из успешных примеров автоматизации клиентского сервиса с помощью искусственного интеллекта является применение чат-ботов в сфере розничной торговли. Например, компания Sephora успешно внедрила чат-бота в мессенджере Facebook для консультирования клиентов по вопросам косметики и предоставления персонализированных

рекомендаций о продуктах. Чат-бот помогает клиентам быстро находить нужную информацию и делать покупки, что улучшает опыт покупателей и увеличивает продажи компаний [7].

Другим примером успешной автоматизации клиентского сервиса с использование ИИ является система обработки запросов в банковской сфере. Многие банки внедрили автоматизированные системы обработки запросов через онлайн-банкинг или мобильные приложения, что позволяет клиентам быстро получать информацию о своих счетах, совершать переводы и выполнять другие операции без необходимости посещения банковских отделений или звонка в контакт-центр [6].

5.2. Неудачные попытки автоматизации

Одним из примеров неудачных попыток автоматизации клиентского сервиса с использование ИИ является случай с компанией Microsoft в 2016 году. Компания запустила твиттер-бота под названием Tay, который был предназначен для общения с пользователями и обучения на основе их сообщений. Однако, из-за недостаточной фильтрации контента и возможности влияния злонамеренных пользователей, бот был использован для распространения неприемлемых сообщений и был отключен в течение 24 часов после запуска. Из этого случая был сделан вывод о необходимости тщательного контроля и обучения искусственных интеллектуальных систем перед их внедрением в реальную среду, а также о важности обеспечения безопасности и этичности использования ИИ в клиентском сервисе [1].

6. Будущее ИИ в клиентском сервисе

6.1. Прогнозы развития технологий

Прогнозы развития искусственного интеллекта в клиентском сервисе указывают на то, что технологии автоматизации и умных систем будут продолжать развиваться и усовершенствоваться. Ожидается, что в ближайшие годы ИИ будет все шире использоваться для улучшения качества обслуживания, оптимизации рабочих процессов и создания персонализированных взаимодействий. Прогнозируется также увеличение интеграции ИИ с другими технологиями, такими как интернет вещей (IoT) и аналитика данных, для создания еще более эффективных и интеллектуальных систем.

6.2. Потенциальные нововведения и их влияние на отрасль

Потенциальным нововведением в области клиентского сервиса с использованием искусственного интеллекта является расширение возможностей автоматизации и адаптации систем под индивидуальные потребности клиентов. Применение технологий глубокого обучения и нейронных сетей позволит создавать более гибкие и интеллектуальные системы, способные адаптироваться к изменяющимся потребностям и предпочтениям клиентов.

Также ожидается развитие технологий обработки естественного языка (NLP) и распознавания речи, что позволит создавать более естественные и продуктивные взаимодействия между клиентами и автоматизированными системами.

Заключение

В результате проведенного исследования были выявлены ключевые аспекты использования искусственного интеллекта в клиентском сервисе, а также его преимущества, вызовы и перспективы.

Анализ текущего состояния клиентского сервиса показал, что автоматизация с

использованием искусственного интеллекта играет все более важную роль в современных бизнес-процессах. Чат-боты, виртуальные ассистенты и системы машинного обучения становятся неотъемлемой частью клиентского обслуживания, обеспечивая более высокое качество обслуживания, сокращение времени ожидания ответа и повышение эффективности работы.

Литература

1. Microsoft 2016. Tay: Microsoft's AI chatbot. Режим доступа: https://en.wikipedia.org/wiki/Tay_(chatbot)
2. Ли К. и Чжан Л. 2017. Этические аспекты использования искусственного интеллекта в обслуживании клиентов. Журнал «Этика и технологии».
3. Баум Э. Б. 2018. Искусственный интеллект в клиентском сервисе: текущее состояние и перспективы. Журнал «Технологии обслуживания».
4. Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. Harvard Business Review, 96(1), 108-116.
5. Li, C., Wang, Y., Liu, Z., & Wang, Y. (2019). Customer service chatbots: State of the art and future directions. Decision Support Systems, 113, 1-15.
6. KPMG. (2020). Customer first: Digital Transformation in Retail Banking. Retrieved from: https://kpmg.com/xx/en/home/insights/2020/01/customer-first-insights-banking.html
7. Sephora introduces chatbot that can reserve appointments. Режим доступа: https://www.cosmeticsbusiness.com/news/article_page/Sephora_introduces_chatbot_that_can_reserve_appointments/122631

УДК 004.023

РЕШЕНИЕ ДВУХКРИТЕРИАЛЬНОЙ ЗАДАЧИ ДВУМЕРНОЙ УПАКОВКИ ПРИ ПОМОЩИ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

В. О. Котов, О. В. Авсеева

Воронежский государственный университет

Введение

Проблема упаковки в контейнеры является важной задачей в различных отраслях промышленности и логистики. Она заключается в распределении набора прямоугольных объектов в контейнеры определённого размера с учётом их геометрических параметров и времени обработки. В данной статье мы рассмотрим практическую задачу упаковки деталей из листового металла на предприятии, используя генетические алгоритмы для оптимизации процесса.

1. Описание задачи

1.1. Постановка

На предприятии имеется база данных с информацией о деталях для изготовления, а также склад с листами металла различных характеристик. Для каждой детали указаны геометрические размеры, время обработки и срочность изготовления. Задача состоит в том, чтобы эффективно распределить детали по листам металла таким образом, чтобы минимизировать количество использованных листов и время обработки.

Технологически процесс раскроя металла на предприятии заключается в следующем:

1. В базе данных предприятия хранится информация о деталях, которые необходимо изготовить.

2. На складе хранятся листы металла с различными характеристиками (к ним, в частности, относятся их геометрические размеры, толщина, сорт), на которых в дальнейшем должны быть размещены детали для резки.

3. Для каждой детали указан в том числе временной интервал её обработки и изготовления. Требуется из базы данных выбрать согласно приоритету обработки те детали, которые можно уместить на тот или иной лист.

4. После распределения деталей по листам необходимо выяснить

- имеются ли не заполненные полностью листы;
- достаточно ли листов на складе.

В первом случае незаполненные до конца листы либо откладываются до получения новых заказов, либо обрабатываются по принципу "как есть" — в этом случае на склад поступает обрезок листа нового размера.

Во втором случае требуется сформировать заказ на закупку дополнительных листов у поставщика.

5. После закупки новых листов обработку не распределенных деталей следует повторить (см. п. 3).

1.2. Математическая модель

Математическая модель сформулированной задачи представляет собой задачу упаковки некоторого количества элементов (изготавливаемых деталей) в минимальное число контейнеров (листов металла). При этом постановка задачи может быть сформулирована следующим образом.

Дано множество контейнеров размера V набор n элементов для упаковки с размерами $a_1, a_2, ..., a_n$. Найти целое число контейнеров B и $B-$разбиение $S_1 \cup ... \cup S_B$ множества $\{1,...,n\}$ такое, что $\sum_{i \in S_k} a_i \le V, \forall k = 1,..., B$ и число контейнеров B было минимально.

Введём обозначения:

$$y_i = \begin{cases} 1, \text{если контейнер } i \text{ использован}, \\ 0, \text{в противном случае}, \end{cases} i = 1,...,n \quad (1)$$

$$x_{ij} = \begin{cases} 1, \text{если элемент } j \text{ упакован в контейнер } i, \\ 0, \text{в противном случае}, \end{cases} \quad (2)$$

$$i = 1,...,n,\ j = 1,...,m$$

Тогда задача запишется в виде:

$$B = \sum_{i=1}^{n} y_i \to \min \quad (3)$$

при

$$B \ge 1 \quad (4)$$

$$\sum_{j=1}^{n} a_j x_{ij} \le V y_i, \forall i = 1,..., n \quad (5)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j = 1,..., n \quad (6)$$

$$y_i \in \{0,1\}, i = 1,..., n \quad (7)$$

$$x_{ij} \in \{0,1\}, i = 1,..., n,\ j = 1,..., n \quad (8)$$

Условие (4) гарантирует ненулевое решение. Условие (5) означает, что суммарный размер деталей, упакованных в i-й контейнер, не превышает его размера. Условие (6) означает, что j-й элемент должен быть упакован только в один контейнер.

Соотношения (3)–(8) действительно представляют собой классическую математическую модель задачи упаковки в контейнеры. Однако, в нашем случае у каждой детали помимо размеров есть ещё один параметр — это время выполнения заказа. Следовательно, контейнеры и упакованные в них детали должны быть распределены таким образом, чтобы при выполнении заказа были соблюдены указанные сроки. То есть в результате все контейнеры должны быть упорядочены, и последовательность контейнеров соответствовала срокам выполнения заказа для содержащихся в них деталей. Иными словами, первый контейнер, представляющий собой лист стали, должен содержать детали с наименьшими сроками выполнения, а последний – с наибольшими.

Учтём вышеописанные условия в математической модели. Указание сроков изготовления деталей устанавливает над элементами для упаковки отношение частичного порядка $\pm$.

Пусть мы имеем следующее упорядочение множества из n элементов:

$$p_1 \pm ... \pm p_i \pm ... \pm p_j \pm ... \pm p_n \quad (9)$$

В данном упорядочении элемент p_1 имеет наибольший приоритет, а p_n — наименьший.

Введём для каждой пары элементов (a_i, a_j) оценку r_{ij} отражающую степень предпочтения элемента a_i элементу a_j и равную расстоянию между этими элементами в упорядочении (9).

Если элемент a_i стоит на i -м месте в ряду (9), а элемент a_j — на j -м, то

$$r_{ij} = j - i, r_{ji} = i - j \quad (10)$$

Таким образом, для упорядочения (9) имеем симметричную матрицу предпочтений R размерности $n \times n$, на главной диагонали которой расположены нули.

Очевидно, что элементы должны распределяться по контейнерам согласно приоритету.

Это условие можно формализовать, введя дополнительный критерий, означающий близость элементов, принадлежащих одному контейнеру, с точки зрения оценки предпочтений r_{ij}:

$$P = \sum_{j=1}^{m} \left(y_j \sum_{i=1}^{n} \sum_{k=1}^{n} x_{ij} x_{kj} |r_{ij}| \right) \to \min \quad (11)$$

Таким образом, получаем двухкритериальную задачу (3)–(8), (11).

2. Генетические алгоритмы

Генетические алгоритмы являются эвристическими методами оптимизации, вдохновлёнными принципами естественного отбора и эволюции в биологическом мире. Они работают на основе понятий хромосом, генов, популяции и операторов скрещивания и мутации. Генетические алгоритмы показывают хорошую эффективность при решении задач оптимизации с большим пространством поиска и множеством локальных оптимумов.

В данном разделе представлено применение NSGA-II (Non-dominated Sorting Genetic Algorithm II)[2] — одного из наиболее эффективных и популярных генетических алгоритмов многокритериальной оптимизации - для решения задачи упаковки в контейнеры.

NSGA-II является улучшенной версией классического алгоритма NSGA, предложенного Куршом и др. в 2002 году. Он работает на основе концепции недоминирующей сортировки и генетической эволюции. NSGA-II эффективно решает задачи многокритериальной оптимизации, обеспечивая набор Парето-оптимальных решений, то есть решений, которые нельзя улучшить по одному критерию без ухудшения по другому.

NSGA-II подходит для решения задачи упаковки в контейнеры из-за его способности работать с множеством критериев и возможности обеспечения множества оптимальных решений в виде недоминирующего фронта Парето.

Процесс работы NSGA-II можно описать следующим образом(см. Рис. 1):

1. **Инициализация популяции**:

Начальная популяция P_t создаётся случайным образом.

2. **Оценка приспособленности**:

Каждой хромосоме x_i в популяции присваивается вектор критериев приспособленности

$$F(x_i) = (f_1(x_i), f_2(x_i), ..., f_m(x_i)) \quad (12)$$

где m - где количество критериев.

3. **Недоминирующая сортировка**:

Применяется недоминирующая сортировка для разделения хромосом на набор недоминирующих фронтов Парето. Для этого определяются следующие понятия:

- **Доминирование**: Хромосома xi доминирует над xj, если

$$\forall k : f_k(x_i) \le f_k(x_j) \quad (13)$$

и существует хотя бы один критерий, при котором неравенство строгое.

- **Недоминирующий фронт**: Недоминирующий фронт - это множество хромосом, которые не доминируются другими хромосомами популяции.
- **Ранг хромосомы**: Ранг хромосомы xi определяется как минимальное число такое, что xi недоминируется любой хромосомой из P_t

4. **Селекция**:

Выбираются хромосомы из набора недоминирующих фронтов Парето для создания новой популяции Q_t следующего поколения.

5. **Скрещивание и мутация**:

Применяются операторы скрещивания и мутации к выбранным хромосомам из Q_t, чтобы создать новое поколение P_{t+1}.

6. **Повторение**:

Шаги 2-5 повторяются до достижения критерия останова.

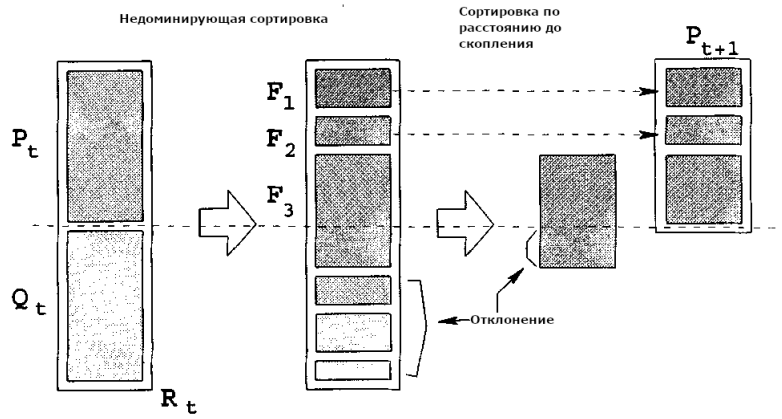


Рис 1. Алгоритм работы NSGA-II

Алгоритмические выражения

- **Доминирование:** Хромосома xi доминирует над xj тогда и только тогда, когда:

$$\forall k : f_k(x_i) \le f_k(x_j) \quad (14)$$

$$\exists k : f_k(x_i) < f_k(x_j) \quad (15)$$

- **Ранг хромосомы**: Ранг хромосомы xi определяется как:

$$rank(x_i) = \min_{j:x_i \text{доминируетнад } x_j} (1 + rank(x_j)) \quad (16)$$

Если xi не доминирует ни над одной хромосомой, ее ранг равен 1.

- **Селекция**:

Выбор хромосомы из популяции P_t для создания Q_t основан на турнирном отборе, где каждый раз случайно выбираются две хромосомы, и та из них, которая имеет более низкий ранг, выбирается для включения в Q_t.

Эти операции повторяются до тех пор, пока не будет достигнут критерий останова, например, заданное количество итераций или сходимость алгоритма.

3. Решение задачи двумерной упаковки генетическим алгоритмом

Для исследования эффективности различных методов решения задачи двумерной упаковки мы использовали тестовую выборку, сгенерированную с помощью нормального распределения, учитывающую длину, ширину и приоритет каждого элемента.

Применяя библиотеку Pymoo [1], в частности алгоритм NSGA-II (Non-dominated Sorting Genetic Algorithm II) [2], мы решали многокритериальную задачу упаковки с учётом двух основных критериев: минимизации количества использованных контейнеров и максимизации заполненности каждого контейнера.

Ниже приведена таблица с данными тестовой выборки:

Таблица 1

Тестовая выборка

Элемент	**Длина**	**Ширина**	**Приоритет**
Элемент1	5	3	2
Элемент2	8	3	4
…	...	...	...
Элемент20	3	4	1

Полученные упаковки были также сравнены с результатами, полученными при использовании других методов, таких как NSGA-II и Brute Force. Ниже приведены результаты сравнения:

Таблица 2

Результаты работы различных подходов

Метод	**Количество использованных контейнеров**	**Заполненность контейнеров**
NSGA2	6	80%
Brute Force	7	75%
Greedy	9	60%

Исходя из сравнения результатов, можно сделать следующие выводы:

1. Генетический алгоритм NSGA-II демонстрирует лучшую эффективность по сравнению с Brute Force и Greedy. Он способен находить оптимальные упаковки с учётом приоритетов и обеспечивает высокую заполненность контейнеров.

2. Метод Brute Force, хотя и обеспечивает оптимальное решение, требует значительно больше вычислительных ресурсов и времени из-за полного перебора всех возможных вариантов упаковки.

3. Метод Greedy, хотя и является простым и быстрым в реализации, оказывается менее эффективным при решении данной задачи из-за своей жадной стратегии выбора.

Заключение

В данной статье была рассмотрена задача упаковки элементов с учётом их приоритетов и размеров. Использование генетического алгоритма с множественными критериями оптимизации позволило эффективно решить эту задачу. Полученные результаты демонстрируют способность алгоритма адаптироваться к различным наборам элементов и находить оптимальные упаковки с учётом их приоритетов.

Одним из преимуществ генетического алгоритма является его способность сохранять разнообразие исследуемых решений, что позволяет обнаружить различные варианты упаковки. Также стоит отметить, что использование генетического алгоритма позволяет учитывать несколько критериев одновременно, что особенно важно при решении многокритериальных задач

Доцент, канд. техн. наук, доцент кафедры математического обеспечения ЭВМ ВГУ, Авсеева Ольга Владимировна.

Литература

1. pymoo: Multi-objective Optimization in Python. – Режим доступа: https://pymoo.org/index.html. – (Дата обращения: 14.03.2024).

2. Comparative Study of Multiobjective Genetic Algorithms. – Режим доступа: https://www.researchgate.net/publication/228845090_Comparative_Study_of_Multiobjective_Genetic_Algorithms. – (Дата обращения: 11.02.2024).

3. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский; Пер. с польск. И. Д. Рудинского. – М.: Горячая линия -Телеком, 2006. – С. 123–274.

4. Антипина, Е.В. Алгоритм решения задачи многоцелевой оптимизации на основе кинетической модели химической реакции / Е.В. Антипина, С.А. Мустафина, А.Ф. Антипин // Автометрия. – 2021. – № 6. – С. 124–131.

5. Прохорова, И. А. Применение генетических алгоритмов при решении многокритериальных задач / И. А. Прохорова, С. С. Аверьянова // Наука ЮУрГУ: материалы 72-й научной конференции. Секции экономики, управления и права. – 2020.

УДК 004.9

ПРИМЕНЕНИЕ ВЕБ-ПРИЛОЖЕНИЙ В КАЧЕСТВЕ ИНТЕРФЕЙСОВ ДЛЯ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ

С. П. Кракова, В. М. Гудков, К. Г. Резников

Воронежский Государственный Университет

Введение

Существует множество библиотек, фреймворков и утилит для разработки веб-приложений [1]. Первичный набор технологий для разработки состоит из языка разметки гипертекста HTML, каскадных таблиц стилей CSS и языка программирования JavaScript. HTML определяет структуру веб-страницы, CSS стилизует внешний вид, а JavaScript отвечает за логику работы веб-приложения [2]. Однако при разработке все более комплексных веб-приложений специалисты сталкиваются с постоянным повышением сложности и увеличивающимся размером исходного кода. Для формализации имеющихся проблем при разработке рассмотрим структуру, содержащую основные этапы реализации [3].

Разработка веб-приложения проходит через этапы создания дизайна, разработки серверной и клиентской части веб-приложения, тестирования и развертывания. Развитие технологического прогресса и внедрение новейших технологий способствовало развитию искусственного интеллекта. Современный модели машинного обучения могут быть развернуты как веб-приложение, но все еще имеют ряд ограничений. Многие исследователи сталкиваются с проблемами при развертывании моделей глубокого обучения в браузере, поскольку некоторые модели не могут быть разработаны для выполнения операций на стороне клиента, кроме того, высокопроизводительный сервер с аппаратным ускорением (GPU, TPU, специализированный CPU) отличается от внешней среды в браузере с ограниченными ресурсами [4].

Рассмотрим различное применение современных компьютерных технологий, информационных технологий (ИТ), при разработке веб-приложений с использованием моделей машинного обучения для дальнейшего анализа прогресса компьютерных технологий в разработке веб-приложений. По результатам анализа, будут проведены исследования современных проблем разработки веб-приложений и представлены методы машинного обучения для упрощения процесса разработки веб-приложений.

1. Анализ исследований различных подходов в интеграции веб-приложений и машинного обучения

Веб-браузеры и веб-приложения привлекли внимание множества исследователей искусственного интеллекта, поскольку они позволяют производить вычисления на стороне клиента и не требуют адаптировать кодовую базу под конкретные платформы. Конфиденциальность, доступность и низкая задержка взаимодействия являются основными преимуществами использования веб-браузеров [1, 2, 4]. Доступ браузера к программным и аппаратным компонентам на стороне клиента, включая веб-камеру, микрофон и акселерометр позволяет легко использовать полученные данные в качестве входных параметров для моделей машинного обучения. Благодаря такой связи пользовательские данные могут храниться на устройстве, сохраняя при этом конфиденциальность, что позволяет создавать персонализированные программы глубокого обучения в таких областях, как медицина и

образование [5].

В работе [6] для облегчения разработки, обучения и развертывания, авторами предложен инновационный подход WebFed на основе языка программирования JavaScript с новой структурой обучения на основе возможностей браузера.

Некоторые моделей машинного обучения предназначены для радиологии, которые могут разворачиваться через облачные платформы и локальные инфраструктуры для использования в современных больницах. Ряд аспектов применения методов глубокого обучения в задачах радиологии приведен в работе [7], где представлены сложные модели глубокого обучения с использованием выделенных высокопроизводительных систем, так и без них в веб-браузере.

Не менее актуально исследование интеграции инструментов искусственного интеллекта, в частности генеративных языковых моделей (GPT, с англ. generative pre-trained transformer) OpenAI для чат-бота в проект на основе WordPress [8].

2. Преимущества и недостатки использования веб-приложений в качестве интерфейсов для моделей машинного обучения

Выделим основные преимущества и недостатки использования веб-приложений в качестве интерфейсов.

Во-первых, такие приложения имеют зависимость от интернет-соединения, что означает, что пользователь не сможет полноценно использовать приложение в отсутствие интернета. Это может быть особенно проблематично в случае решения продолжительных по времени задач или в ситуациях, где интернет-соединение нестабильно или недоступно.

Кроме того, использование веб-приложений может ограничить функциональность и производительность интерфейса, поскольку веб-технологии, такие как HTML и JavaScript, имеют свои ограничения и могут не обладать той же производительностью, что и настольные приложения.

Другой недостаток заключается в том, что веб-приложения могут стать объектом атак со стороны злоумышленников. Поскольку веб-приложения доступны через интернет, они могут быть подвержены различным угрозам безопасности, таким как введение вредоносного кода, перехват данных, фишинг и другие виды атак. При создании веб-приложений необходимо уделять большое внимание безопасности и предпринимать соответствующие меры для защиты данных и пользователей от потенциальных угроз.

Также следует отметить, что использование веб-приложений требует наличия разработчиков, которые обладают навыками разработки веб-приложений. Это может означать дополнительные затраты на найм и обучение соответствующих специалистов. Кроме того, веб-приложения могут требовать регулярного обновления и поддержки, чтобы сохранить свою функциональность и совместимость с различными веб-браузерами и платформами.

Несмотря на эти недостатки, использование веб-приложений в качестве интерфейсов также имеет свои преимущества. Хоть, с одной стороны, необходимость подключения к сети интернет является минусом, но это позволяет приложениям быть доступными с любого устройства с подключением к интернету. Кроме того, веб-технологии позволяют создавать адаптивные интерфейсы, которые могут корректно отображаться и работать на различных устройствах с разными размерами экранов и разрешениями. Это повышает удобство использования и доступность приложения для широкого круга пользователей.

В проведенной научной исследовательской работе, принадлежащей Калифорнийскому университету, было разработано веб-приложение с использованием модели глубокого обучения для сегментации изображений на базе YOLO (You Only Look Once). Развертывание данной модели на клиентской стороне при помощи JavaScript и TensorFlow.js привело к

снижению нагрузки на сервер при той же точности сегментации.

Также стоит обратить внимание на примеры успешного использования веб-приложений для моделей машинного обучения в реальных условиях, в том числе проект Teachable Machine от Google, который предоставляет возможность пользователям обучать модели машинного обучения прямо в браузере. Этот подход позволяет пользователям создавать модели, способные классифицировать и распознавать объекты на основе их персональных данных. Современные модели машинного обучения часто используются в качестве веб-приложений, что облегчает их использование и распространение. Это открывает новые возможности для разработчиков и исследователей, предоставляя им возможность создавать более эффективные и доступные решения.

Зачастую портирование разработок машинного обучения в новые среды требует значительных усилий и времени, из-за несовместимости существующих фреймворков с окружениями для развертывания. Это связано с тем, что фреймворки глубокого обучения обычно разрабатываются для использования на мощных машинах с графическими процессорами, что ускоряет процесс обучения, но затрудняет его применение на устройствах с ограниченными ресурсами. Кроме того, создание веб-приложений на JavaScript может быть сложным для разработчиков, привыкших к использованию Python, что дополнительно осложняет процесс. Одной из сложностей также является ограниченный выбор общедоступных пакетов и встроенных функций JavaScript для глубокого изучения, что ограничивает доступность и возможности разработчиков по сравнению с Python.

Таким образом, применение веб-приложений в качестве интерфейсов для моделей машинного обучения имеет ряд преимуществ, таких как упрощение использования API, охват большого числа пользователей, снижение нагрузки на сервер и оперативный отклик. Однако такой подход также несет проблемы, связанные с повышенными требованиями к безопасности, сложностями с адаптацией моделей к нужным форматам для целевых платформ и возможностями непроизводительных платформ.

3. Использование TensorFlow.js

Использование TensorFlow.js вместе с API-интерфейсами TensorFlow Python позволяет создавать модели машинного обучения, обладающие сильными сторонами обоих платформ. Однако, в отличии от TensorFlow Python, TensorFlow.js может быть легко распространен и выполнен на любой платформе без необходимости установки, что делает его более удобным и доступным. Особенно следует отметить, что TensorFlow.js обладает высокоуровневым API для моделирования, что облегчает развертывание предварительно обученных моделей в браузере. Кроме гибкости, необходимой для веб-приложений машинного обучения, TensorFlow.js также обладает высокой производительностью благодаря использованию аппаратного ускорения. Это связано с тем, что библиотека TensorFlow.js использует механизм ускорения, который может быть доступен в современных веб-браузерах, что позволяет выполнять высокопроизводительное машинное обучение и множество вычислений на JavaScript [7, 8]. Современные веб-браузеры имеют API WebGL (кроссплатформенный API для 3D-графики в браузере), который изначально предназначался для ускорения рендеринга 2D и 3D-визуальных изображений на веб-страницах. Позже он был перепрофилирован для параллельных вычислений в нейронных сетях в TensorFlow.js[9]. Это означает, что интегрированная видеокарта может ускорить операции машинного обучения, устраняя при этом потребность в специальных видеокартах, например NVIDIA, GTX, необходимых для нативных фреймворков.

TensorFlow.js содержит коллекцию предварительно обученных моделей Google, показанных в таблице 1. Подготовленные модели на основе команд готовы выполнять различные задачи, например: идентификация объектов, сегментация изображения,

распознавание голоса и категоризация токсичности текста.

Таблица 1

Модели TensorFlow.js

Название	Применение	Описание
Компьютерное зрение		
MobileNet	Классификация изображений	Модель, обученная с помощью базы данных ImageNet, содержит 1,2 миллиона обучающих изображений и 1000 классов объектов.
Coco ssd	Обнаружение объектов	Модель, обученная на наборе данных COCO, может классифицировать и выделять элементы из 80 разных категорий на изображениях.
DeepLab	Семантическая сегментация	Модель способная попиксельно локализировать объекты с определением их класса.
Распознавание лиц и поз		
Blazeface	Распознавание лиц	Модель распознавания лица на основе архитектуры Single Shot Detector, способна обнаруживать одно или несколько лиц на фотографии.
HandPose	Отслеживание рук	Состоит из двух моделей: детектора ладони и модели, отслеживающей пальцы на скелете руки. Предусматривает 21 ключевую точку для каждой идентифицированной руки отдельно.
Pose-detection	Отслеживание тела	Унифицированный набор инструментов для обнаружения поз, использующих одну из трех моделей (MoveNet, BlazePose и PoseNet) для обнаружения нетипичных поз и быстрых телодвижений в режиме реального времени. – MoveNet – быстрая и высокоточная модель, определяющая 17 ключевых точек тела; – BlazePose может определять 33 ключевые точки; – PoseNet может определять несколько поз, каждая из которых содержит 17 ключевых точек.
BodyPix	Сегментация частей тела	Набор моделей сегментирующих 24 компонентов тела с изображения или видео в реальном времени, работает для нескольких людей. Его конструкция основана либо на MobileNetV1 (менее точный, но более быстрый вариант), либо на ResNet50 (более точный, но медленный вариант).
Обработка естественного языка		
BERT	Вопрос-ответные системы	Модель ответов на вопросы на естественном языке в заданном контексте. Обучена на основе набор данных SQuAD 2.0.
Toxicity	Анализ тональности	Модель, обученная с помощью набора данных комментариев, который включает более 2 миллионов комментариев, которые могут классифицировать текст от «Очень токсичный» до «Очень здоровый».
USE	Векторизация	Модель может сжимать текст в 512-мерное

	текстов	векторное представление, которое можно использовать, например для классификации или поиска схожих по смыслу текстов.

Модели можно использовать в существующем виде или применять как базовые при трансферном обучении.

4. Результаты анализа и методы машинного обучения для упрощения процесса разработки веб-приложений

В последние годы разработка веб-приложений стала одной из основных задач сферы информационных технологий. Однако, процесс разработки веб-приложений остается достаточно сложным и трудоемким. В связи с этим, активное использование методов машинного обучения становится все более популярным для упрощения этого процесса. Рассмотрим подробнее внедрение машинного обучения для упрощения процесса разработки веб-приложений.

Первым из варианта использования моделей может являться обучающая платформа, основанная на MobileNet. Данная технология позволяет пользователям обучать классификационные модели машинного обучения. Такое веб-приложение использует видео, изображения и звук с устройств для построения модели, которая выявляет тенденции и закономерности в медиаресурсах. Примером использования такой платформы является обучение пользователей с помощью интерактивного онлайн-объяснения и распознавания изображений. Например, пользователи могут сфотографировать предмет, а алгоритм глубокого обучения на веб-сайте сможет идентифицировать его , так же может быть в виде развивающей платформы как визуальной веб-среды программирования, предоставляющей интерфейс с использованием графических блоков (Blocky40) с возможностью создания интерактивной визуализации.

Кроме того, существуют платформы для реабилитации и мониторинга предназначенные для облегчения оценки риска падения и мониторинга реабилитации людей, такие приложения способны помогать улучшать и стабилизировать состояние здоровья. Например, в работе [10] использовался PoseNet для программ помощи в реабилитации в домашних условиях, где скелетное отслеживание использовалось для идентификации и мониторинга движений пациентов при выполнении реабилитационных упражнений перед веб-камерой. После того, как пациенты завершали реабилитационные мероприятия, врачи могли исследовать и оценивать показатели в разные дни, чтобы узнавать темпы выздоровления. Интерфейс контроля результатов полученных с помощью моделей машинного обучения представлен на рисунке 1.

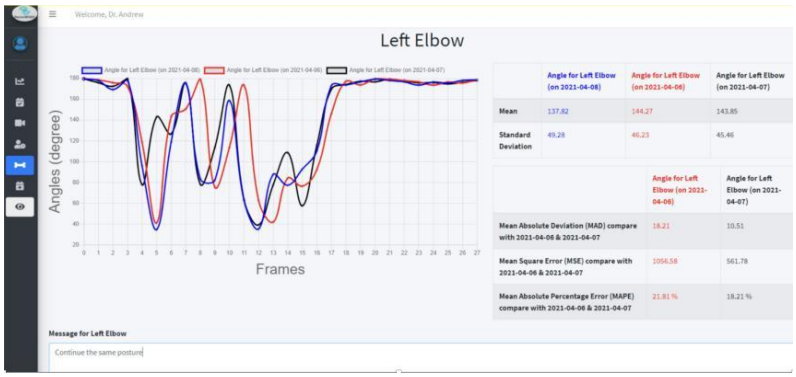


Рис. 1. Интерфейс контроля результатов

Далее рассмотрим приложения для отслеживание жестов, это такое веб-приложение, которое реализует 3D-жестовый ввод, используя оценку позы и взаимодействие с пользователем. Так, Scroobly46 и PoseAnimator47 отражают пользовательские движения в своих анимациях с помощью моделей машинного обучения Facemesh и PoseNet. Когда пользователь двигается, алгоритм машинного обучения изменяет отображаемую на экране анимацию[11].

Обнаружение выражений лица — это веб-приложения с открытым исходным кодом, например, Face-api.js, которое позволяет распознавать эмоции. Использование такого веб-приложения может быть полезным для мониторинга эмоционального состояния пациентов во время психиатрического лечения. Для распознавания лица используется предиктор, основанный на 68 ориентирах, одним из его достоинств является его размер, он составляет менее 200 Кб. Кроме того, на основе этого приложения в рамках предотвращения несанкционированного доступа создана система, использующая методы распознавания лиц и выявления ориентиров для выполнения задач верификации личности и обнаружения живого человека[8].

Веб-интерфейсные приложения для глубокого обучения также применяются для централизованного взаимодействия между удаленными пользователями. Используя веб-инструменты, можно создать онлайн-систему для интеграции изображений отпечатков пальцев, что позволяет организовать сотрудничество для выполнения задач, требующих использования конфиденциальных данных. В данном случае, анализ, осуществляемый с использованием TensorFlow.js, позволяет проводить предварительную обработку данных [12].

В целом, предварительно обученные модели TensorFlow.js предоставляют мощные инструменты для решения задач компьютерного зрения и обработки изображений. Они могут быть использованы в различных областях, таких как реклама, медицина, безопасность и фитнес и многих других. Эти модели сочетают в себе высокую точность и эффективность, что делает их идеальными для встраивания в веб-приложения.

Заключение

Использование веб-приложений как интерфейсов для моделей машинного обучения имеет ряд преимуществ, которые объясняются широким охватом, снижением затрат на сервере, своевременным реагированием и повышенными условиями конфиденциальности.

Проведенный анализ применения моделей глубокого обучения в разработке веб-приложений позволяет сделать несколько предположений. При создании приложений с применением методов глубокого обучения, важно учитывать не только стандартные этапы разработки, но и специфические шаги, связанные с применением таких методов.

Проанализированы ключевые особенности применения языков программирования, таких как Python и JavaScript для браузерных веб-приложений, а также использование TensorFlow.js для создания моделей машинного обучения, в том числе совместимых с API-интерфейсами TensorFlow Python.

Таким образом, использование веб-браузеров и веб-приложений привлекает внимание исследователей из-за их способности выполнять вычисления на стороне клиента и не требовать адаптации под различные платформы, что обеспечивает высокую конфиденциальность, доступность и низкую задержку взаимодействия.

Список литературы

1. Резников, К. Г. Разработка веб-приложения для визуализации кинематических

поверхностей с использованием трассировки лучей / К. Г. Резников, С. Н. Медведев // Вестник факультета прикладной математики, информатики и механики. – Воронеж: Издательский дом ВГУ, 2021.

2. Dinh, D., & Wang, Z. (2020). Modern front-end web development: how libraries and frameworks transform everything. https://urn.fi/URN:NBN:fi:amk-2020060416939

3. Сотник, С., Шакурова, Т., & Ляшенко, V. (2023). Development Features Web-Applications. https://openarchive.nure.ua/handle/document/21600

4. Goh, H. A., Ho, C. K., & Abas, F. S. (2022). Front-end глубокие яркие web apps разработки и развития: a review. Applied Intelligence, 1-23. https://doi.org/10.1007/ s10489-022-04278-6

5. Andersson, V., & Roll, E. (2020). Front-end study and application of modern web-app technologies with the aim of improving an existing system (Bachelor's thesis, NTNU). https://hdl.handle.net/11250/2672185

6. Lian, Z., Yang, Q., Zeng, Q., & Su, C. (2022, May). Webfed: Cross-platform federated learning framework основанный на web browser with local differential privacy. In ICC 2022-IEEE International Conference on Communications (pp. 2071-2076). IEEE. https://doi.org/10.1109/ ICC45855.2022.9838421

7. Jodogne, S. (2023). Client-Side Application of Deep Learning Models Through Teleradiology. Studies in Health Technology and Informatics, 302, 997-1001. https://doi.org/ 10.3233/SHTI230325

8. Tosic, D. (2023). Artificial Intelligence-driven web development and agile project management using OpenAI API and GPT technology: A detailed report on technical integration and implementation of GPT models in CMS with API and agile web development for quality user-centered AI chat service experience. urn:nbn:se:miun:diva-48446

9. Kanber, B. (2018). Hands-on machine learning with JavaScript: solve complex computational web problems using machine learning. Packt Publishing Ltd. ISBN 9781788990301

10. Chua, J., Ong, L. Y., & Leow, M. C. (2021). Telehealth using PoseNet-based system for in-home rehabilitation. Future Internet, 13(7), 173. https://doi.org/10.3390/fi13070173

11. Smilkov, D., Thorat, N., Assogba, Y., Nicholson, C., Kreeger, N., Yu, P., ... & Wattenberg, M. M. (2019). TensorFlow.js: Machine Learning for the Web and Beyond. Processes from Management and Systems, 1, 309-321. https://arxiv.org/abs/1901.05350

12. Cai, S., Bileschi, S., Nielsen, E. D., & Chollet, F. (2020). Deep Learning with JavaScript: Neural networks in TensorFlow. js. Manning Publications. ISBN 9781617296178

УДК 004.738.5

ИНТЕГРАЦИЯ IOT И TELEGRAM БОТОВ В УМНЫЕ СИСТЕМЫ КОНТРОЛЯ ФИЗИЧЕСКОГО ДОСТУПА

С. П. Кракова, С. Ю. Болотова

Воронежский государственный университет

Введение

В современное время, когда вопросы безопасности становятся всё более актуальными, важно использовать передовые технологии для обеспечения контроля физического доступа к объектам. Одним из таких инновационных решений является применение Интернета вещей (IoT), который позволяет управлять умными устройствами на расстоянии, тем самым повышая уровень защиты объектов. Интеграция IoT в системы контроля доступа обеспечивает возможность постоянного мониторинга в реальном времени и быстрого реагирования на любые угрозы, при обнаружении попыток несанкционированного доступа система автоматически активирует защитные механизмы и немедленно оповещает о происходящем. Такая система обладает высокой адаптивностью и масштабируемостью, эти качества делают ее идеальным выбором для городских условий, где требования к безопасности постоянно растут.

1. Организация IoT

Для организации IoT необходимо выполнить два шага. На первом шаге осуществляется взаимодействие каждого устройства с сервером. Далее следует выбрать протокол связи, который будет использоваться для взаимодействия между устройствами IoT и сервером, таких как MQTT, UDP, HTTP и другие. Каждый из протоколов имеет свои особенности, преимущества и ограничения. Важно, чтобы каждое подключаемое устройство поддерживало тот протокол связи, который выбран для взаимодействия с сервером.

Другим подходом может выступать взаимодействие с сервером через устройство, называемое хабом (от англ. hub). Оно собирает и отправляет данные на каждое устройство в системе по удобному для каждого интерфейсу связи, из популярных и доступных на контроллерах интерфейсах на данный момент выделяются: проводные – RS232, CAN; беспроводные – WIFI, Bluetooth, ZigBee, 433МГц.

Выбор подходящего протокола связи, такого как MQTT, UDP, HTTP и другие, является критически важным этапом в разработке системы Интернета вещей, поскольку каждый из протоколов обладает уникальными техническими характеристиками, преимуществами и ограничениями, что существенно влияет на производительность и безопасность системы. В отличие от использования хаба, который выступает в роли центрального устройства, собирающего и пересылающего данные на каждое подключенное устройство, применение единого протокола связи для всех устройств обеспечивает гомогенную среду, где все устройства могут взаимодействовать на одном языке, упрощая таким образом интеграцию новых устройств и обслуживание системы.

Это также упрощает масштабирование системы, так как добавление новых устройств не

требует изменений в основной инфраструктуре или дополнительных настроек на центральном хабе, что, в свою очередь, уменьшает зависимость от работы одного устройства и снижает риск точки отказа в системе. Таким образом, несмотря на удобство использования хаба в определенных ситуациях, выбор подходящего протокола связи для непосредственного взаимодействия устройств с сервером предлагает значительные преимущества в плане масштабируемости, надежности, эффективности и безопасности системы.

1.1. Определение протокола связи

При выборе платформы для IoT критически важным аспектом является определение протокола связи, который будет использоваться для обеспечения взаимодействия устройств с сервером. Ниже представлены некоторые из наиболее распространенных протоколов:

MQTT — это сетевой протокол обмена сообщениями, работающий по принципу "издатель-подписчик" на основе TCP/IP. Для обеспечения безопасности данных необходимо использовать SSL-шифрование. Протокол поддерживается множеством микроконтроллеров и платформ управления домашней автоматикой, таких как Home Assistant и OpenHAB, и легко конфигурируется на серверах.

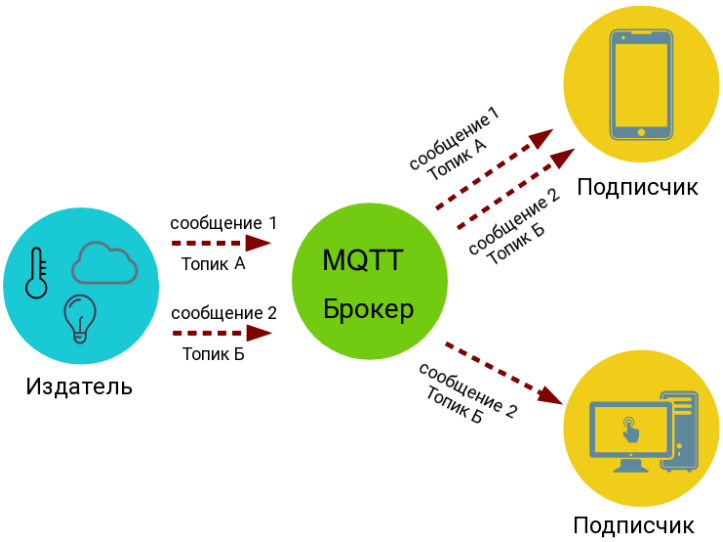


Рис. 1. Схема работы протокола MQTT

UDP — это транспортный протокол, который осуществляет передачу данных без подтверждения их приема, что увеличивает скорость передачи за счет снижения надежности. Данные, передаваемые по UDP, обычно защищены шифрованием AES-GCM, что избавляет от необходимости дополнительной организации шифрования. Этот протокол известен своей простотой и поддерживается многими устройствами и платформами.

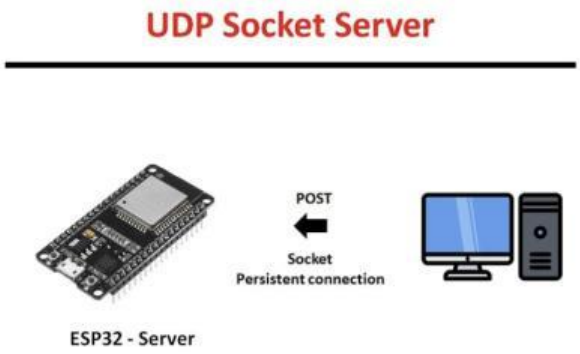


Рис. 2. UDP сокет-сервер на базе ESP32

На рисунке 2 представлена схема работы UDP сокет-сервера, реализованного на микроконтроллере ESP32. Схема показывает, как сервер на базе ESP32 поддерживает постоянное соединение через сокет и принимает POST-запросы от клиентского устройства. UDP протокол обеспечивает быструю передачу данных за счет отсутствия подтверждения получения, что снижает задержки, но уменьшает надежность передачи.

HTTP — протокол передачи данных, который используется для взаимодействия между пользовательскими приложениями и серверами. Для защиты данных необходимо применять дополнительные меры шифрования, например SSL. HTTP относительно сложен в реализации на микроконтроллерах, но поддерживается системами вроде Home Assistant и OpenHAB.

На рисунке представлена схема работы HTTP-сервера, взаимодействующего с различными клиентскими устройствами, включая микроконтроллер ESP32, мобильные устройства и ПК/ноутбуки. HTTP-сервер находится в облаке и обрабатывает запросы от клиентов, обеспечивая передачу данных через интернет. Это позволяет устройствам в IoT-системе обмениваться информацией и взаимодействовать друг с другом посредством HTTP-протокола.

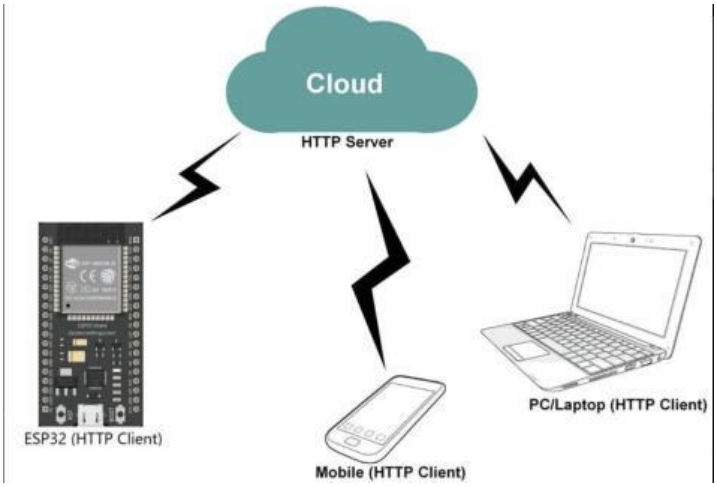


Рис. 3. Схема работы HTTP

Для систем, которые включают сервер с программным обеспечением для визуализации

данных и автоматизации процессов, рекомендуется выбрать протокол MQTT, поскольку он является наиболее популярным в применениях умного дома. Для его функционирования необходим MQTT-брокер. В данном случае был выбран WQTT, который не ограничивает количество подключений, предоставляет русскоязычный интерфейс и предлагает доступные тарифы на подписку.

2. Интеграция с Telegram ботом и программные решения для управления

Telegram боты, действующие как виртуальные пользователи, становятся ключевым элементом в системах контроля доступа. Основное преимущество использования Telegram ботов заключается в том, что они значительно упрощают процессы управления и мониторинга систем контроля доступа, используя мессенджер для отправки команд и данных.

Telegram боты становятся ключевым элементом систем контроля доступа, взаимодействуя с пользователями и устройствами, что упрощает управление и мониторинг. Интеграция через платформы, такие как Home Assistant, расширяет возможности систем, обеспечивая высокую безопасность данных.

Благодаря готовым решениям, разработка систем для визуализации данных и автоматизации процессов становится более доступной. Эти платформы поддерживают широкий спектр протоколов связи и позволяют легко интегрировать Telegram боты, предлагая обширные возможности для анализа и отображения данных, а также автоматизации операций. Гибкость и масштабируемость этих систем позволяют адаптировать их к различным типам умных домов и коммерческих автоматизационных проектов, обеспечивая надежное и адаптивное решение для управления умными устройствами.

3. Создание и настройка Telegram бота

Для начала интеграции Telegram в систему умного дома Home Assistant, первоначально требуется создать бота в Telegram с помощью BotFather. Данный процесс включает регистрацию нового бота и получение уникального API-токена, который затем используется в конфигурационном файле Home Assistant для установления связи с ботом. Эффективное использование бота обеспечивает автоматическую отправку уведомлений о событиях в доме, таких как активация сигнализации, изменения температуры или включение систем безопасности.

На практике, интеграция с Home Assistant позволяет пользователям разрабатывать автоматизированные сценарии, например, включение освещения при входе в дом или отправка уведомлений при обнаружении посетителей у входной двери. Это не только увеличивает удобство использования, но и повышает уровень безопасности, предоставляя возможности контроля доступа в реальном времени.

Использование Telegram в качестве интеграционного компонента системы умного дома предоставляет ряд значительных преимуществ. Пользователи могут получать мгновенные уведомления о любых изменениях или событиях, что способствует оперативному реагированию. Бот предоставляет интерактивную платформу для приема команд и их выполнения, что значительно расширяет функциональные возможности системы. Благодаря поддержке всех основных платформ, Telegram является универсальным и доступным инструментом для интеграции в системы умного дома.

Telegram бот выполняет функцию пользовательского интерфейса, позволяя пользователям отправлять команды для управления замком и получать уведомления о событиях доступа. Использование MQTT в качестве протокола передачи данных улучшает надежность и оперативность коммуникаций между умными устройствами и сервером, обеспечивая быструю реакцию на команды пользователя и высокую степень безопасности передаваемой информации.

Таким образом, интеграция Telegram бота и MQTT в систему контроля доступа значительно повышает уровень удобства и эффективность управления безопасностью, обеспечивая непрерывное и точное взаимодействие между пользователями и умными устройствами. Эта архитектура позволяет не только оперативно реагировать на события, но и гибко настраивать параметры безопасности в зависимости от потребностей пользователей.

Заключение

Интеграция IoT и Telegram ботов в умные дома повышает безопасность и удобство управления. Использование протоколов MQTT и других технологий обеспечивает адаптивность и масштабируемость систем, позволяя эффективно реагировать на угрозы и автоматизировать процессы.

Литература

1. Официальная документация OpenHAB [Электронный ресурс] // URL: https://www.openhab.org/docs (дата обращения: 28.08.2023).

2. Официальная документация Home Assistant [Электронный ресурс] // URL: https://www.home-assistant.io (дата обращения: 28.08.2023).

3. Боксел, Дж. Изучаем Arduino. 65 проектов своими руками [Текст] / Дж. Боксел; пер. с анг. А. Киселев, ред. Н. Римицан – СПб : Питер, 2019. – 400 с.

4. «Интернет вещей» - что это такое и как применять IoT в реальном бизнесе. URL: https://rb.ru/longread/iot-cards/.

5. Семь российских проектов в области «Интернета вещей» // rb.ru/list/iot-7/.

6. Инструкция по установке Home Assistant [Электронный ресурс] URL: https://www.home-assistant.io/installation/ (дата обращения: 20.04.2024).

УДК 004

ПОСТАНОВКА ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТНЫХ СРЕДСТВ С УЧЕТОМ ГРУЗОПОДЪЕМНОСТИ ДЛЯ ВЕРОЯТНОСТНОГО ПРОГНОЗА СЛОЖНОСТИ

Д. А. Крамаренко

Воронежский государственный университет

Введение

Настоящая работа посвящена постановке задачи маршрутизации с транспортных средств с учетом грузоподъемности для дальнейшего вероятностного прогноза сложности.

В современном мире транспортная отрасль играет ключевую роль в обеспечении эффективной и безопасной транспортировки грузов. Одной из основных задач в этой сфере является задача маршрутизации транспортных средств. Эффективное планирование и оптимизация маршрутов движения транспортных средств имеет прямое влияние на экономическую эффективность и конкурентоспособность транспортных предприятий. Кроме того, при построении маршрутов возникают проблемы нехватки транспорта, ограничение грузоподъемности транспорта.

Большой интерес представляют временные затраты для индивидуальной задачи маршрутизации с учетом грузоподъемности. Встает задача исследования подходящего алгоритма решения задачи маршрутизации транспорта, а также определения величины, представляющую сложность индивидуальной задачи. В перспективе, на основе данных условий, будет проводиться вероятностный прогноз сложности индивидуальной задачи в зависимости от размерности ее матрицы стоимости.

1. Задача маршрутизации с учетом грузоподъемности

Для решения задачи маршрутизации с учетом грузоподъемности рассмотрим ее математическую модель [1].

Изначальное имеется некоторое количество единиц транспорта с ограниченной грузоподъемностью. Дан один склад (депо), а также некоторое количество потребителей. Для каждой единицы транспортного средства необходимо составить маршрут, который начинается и заканчивается в депо, и при этом на каждом маршруте сумма груза не должна превышать грузоподъемность транспортной единицы.

Пусть граф G – совокупность множеств вершин V и ребер E, между которыми определено отношение инцидентности. Вершинами графа являются пункты потребления, а длины ребер d задают расстояние между этими пунктами. Мощность множеств $|V(G)|=n$ и $|E(G)|=n(n-1)/2$. Имеется величина v_0, обозначаемая как склад (депо). Каждый пункт потребления имеет неотрицательный спрос c_i. На складе имеется некоторое количество транспортных единиц m с грузоподъемностью q. Граф является неориентированным, а его маршруты – простые цепочки с неповторяющимися вершинами. Маршруты включают в себя несколько вершин, сумма весов ребер в маршруте k – пройденное расстояние одним транспортным средством (рис. 1).

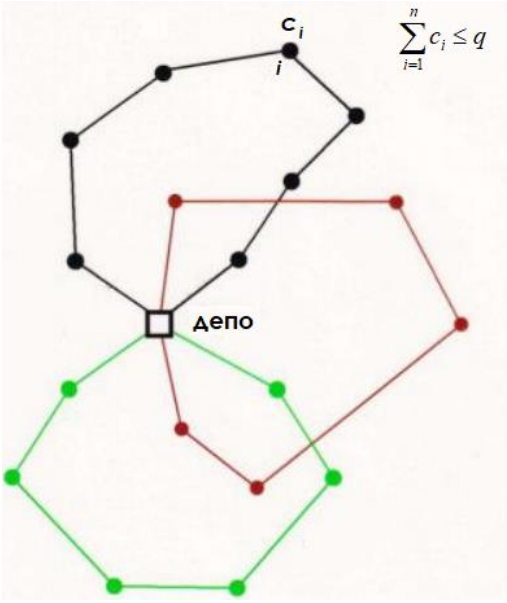


Рис. 1. Граф задачи маршрутизации транспорта с учетом грузоподъемности

Задача состоит в минимизации пройденного расстояния всеми имеющимися транспортными средствами:

$$\sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n} d_{ij} X_{ij}^{k} \to \min \tag{1}$$

При нахождении решения задачи важно, чтобы оно удовлетворяло поставленным ограничениям:

$$\sum_{i=1}^{n} c_i \sum_{j=0}^{n} X_{ij}^{k} \le C_k \quad \forall k = 1, \ldots, m, \tag{2}$$

$$\sum_{j=1}^{n} X_{0j}^{k} \le 1 \quad \forall k = 1, \ldots, m, \tag{3}$$

$$\sum_{i=1}^{n} X_{i0}^{k} \le 1 \quad \forall k = 1, \ldots, m, \tag{4}$$

$$\sum_{k=1}^{m}\sum_{i=0}^{n} X_{ij}^{k} = 1 \quad \forall i = 1, \ldots, n, \tag{5}$$

$$\sum_{i=0}^{n} X_{ih}^{k} - \sum_{j=0}^{n} X_{hj}^{k} = 0 \quad \forall h = 1, \ldots, n, \forall k = 1, \ldots, m, \tag{6}$$

$$X_{ij}^{k} \in \{0,1\} \quad \forall i, j = 0, \ldots, n, \forall k = 1, \ldots, m, \tag{7}$$

$$X_{ii}^{k} = 0 \quad \forall i = 0, \ldots, n, \forall k = 1, \ldots, m, \tag{8}$$

2. Алгоритм решения и показатель сложности задачи

Транспортная задача маршрутизации с учетом грузоподъемности является NP-трудной и обобщает задачу коммивояжера и задачу о рюкзаке. Задачи средних размерностей могут быть решены точно. Многие точные алгоритмы основаны на поиске по дереву решений (рис. 2).

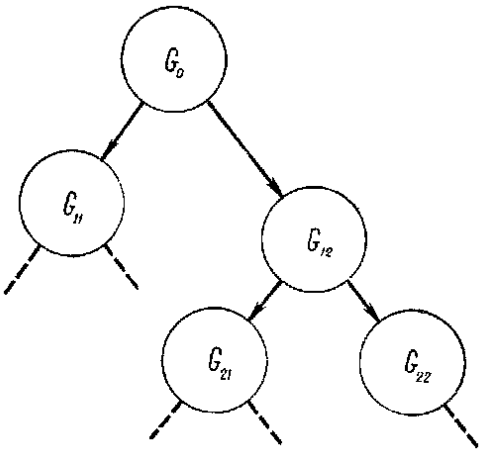


Рис. 2. Метод ветвей и границ для задачи маршрутизации

Метод ветвей и границ заключается в разбиении задачи на упрощенные подзадачи, путем фиксирования значения переменной ветвления. Эти подзадачи решается тем же способом, разветвляется другая переменная. На каждом шаге проверяется, нет ли лучшего решения, чем то, известное в настоящее время, которое можно найти в этой ветке. Одним из методов ветвей и границ для задачи маршрутизации является метод K-дерева [2].

Пусть количество используемого транспорта равно K. Транспортная задача маршрутизации моделируется как задача определения K-дерева, то есть связного графа с K циклами, такого, что вершина v_0 имеет степень $2K$, степени вершин-потребителей равны двум и выполняются дополнительные ограничения на грузоподъемность транспортного средства.

Опираясь на работу М. И. Фомичева [3], под сложностью индивидуальной задачи, заданной матрицей A, удобно принять величину $c(A)$, которая равна числу порожденных методом ветвей и границ вершин дерева решений. Данная характеристика не будет зависима от мощности вычислительной машины, а также позволит изучить задачу вне зависимости от программных реализаций и аппаратных конфигураций.

Заключение

В данной статье сформулирована математическая модель задачи маршрутизации транспорта с учетом грузоподъемности. Приведены варианты методов решения данной задачи, определена величина, описывающая ее сложность. Вероятностный анализ решений задач различных размерностей поможет спрогнозировать временные затраты для индивидуальной задачи маршрутизации. Разработанный в дальнейшем программный комплекс позволит провести множество вычислительных экспериментов и в зависимости от размерности индивидуальной задачи спрогнозировать временные затраты.

Литература

1. Смирнов К. С. Решение задачи маршрутизации транспорта в системе «1С:Предприятие 8.3» / К. С. Смирнов // ФГАОУ ВО «Южно-Уральский государственный университет», 2016. – 61 с. – URL: https://dspace.susu.ru/xmlui/handle/0001.74/11077 (дата обращения: 19.04.2024)

2. Чеписова Е. С. Методы ветвей и сечений для задачи маршрутизации с ограничением по грузоподъемности / Е. С. Чеписова // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2012. – № 2. – С. 68–72.

3. Фомичев М. И. Вероятностный прогноз сложности индивидуальных задач коммивояжера на основе идентификации распределения сложности по экспериментальным данным / М. И. Фомичев, М. В. Ульянов, Г. Н. Жукова, В. А. Головешкин // Автоматика и телемеханика. – 2018. – № 7. – С. 149–166.

УДК 004.623

АНАЛИЗ СПОСОБОВ ОБНОВЛЕНИЯ ДАННЫХ В КОНЕЧНЫХ ОБЪЕКТАХ БАНКОВСКОГО КОРПОРАТИВНОГО ХРАНИЛИЩА ДАННЫХ

Д. А. Кретинина

Воронежский государственный университет

Введение

Объём информации, обрабатываемый банковским корпоративным хранилищем данных (КХД), постоянно растёт. Архитекторы и разработчики обязаны предусмотреть возможность увеличения количества информации в таблицах, служащих источниками данных для КХД, и, как следствие, увеличение количества времени, необходимого на их загрузку в хранилище и проведения дальнейших расчетов для получения данных, требующихся бизнес-пользователям.

В случае выбора неоптимального способа обновления для объекта, процесс расчета данных может занять несколько десятков часов, а в исключительных случаях – завершиться с ошибкой, связанной с нехваткой места в оперативной памяти. Задержки обновления хранилища данных влекут за собой сбои в работе сотрудников Банка, так как у них нет информации, с которой они могли бы работать, а также штрафы, которые накладываются на руководство Банка за несвоевременное предоставление отчетов в Центральный Банк [1].

Конечными объектами в КХД являются объекты детального слоя, как правило, хранящие данные в том виде, в котором они поступают из таблиц систем-источников, и объекты витринных слоев, расчёт которых происходит по заданной бизнес-логике.

1. Способы обновления данных в таблицах детального слоя

По типу загрузки данных из систем-источников в КХД делятся на два типа:

- архивный (полная выгрузка данных из таблицы системы-источника);
- инкрементальный (загрузка только изменившихся данных).

1.1. Полная выгрузка данных из таблицы системы-источника

Архивная выгрузка данных используется:

- для первоначальной инициализационной загрузки данных в КХД на установленную дату;
- для повторной загрузки ранее загруженных данных в КХД;
- для регулярной загрузки данных в КХД из таблиц СИ небольшого объёма [2].

При таком подходе каждый раз при обращении к таблице-источнику происходит забор всех имеющихся в ней данных. Этот способ подходит для регулярной загрузки сравнительно небольших таблиц (до нескольких миллионов записей) и удобен тем, что не требует дополнительного анализа при организации инкремента и не оказывает дополнительную нагрузку на источник наложением фильтра на отбор записей.

После загрузки полного среза данных источника в таблицу типа «SNP» — таблицу, совпадающую по атрибутному составу с таблицей системы-источника и предназначенную для хранения снимка текущей выгрузки источника — происходит сравнение этих данных с данными, полученными в предыдущей загрузке. В результате сравнения получается набор

записей, атрибуты которых были изменены, добавлены или удалены с момента предыдущей загрузки. По полученному набору записей выполняется актуализация таблицы-зеркала источника с обязательной отметкой времени полученного изменения и флагом операции («I» – вставка новой строки, «D» – удаление строки, «U» – изменение строки).

Преимущество данного метода состоит в его простоте и в том, что он не зависит от исходной системы. Недостатками являются высокие накладные расходы, связанные с необходимостью повторного сравнения для неизмененных данных при каждой загрузке.

Сравнение может быть выполнено:

- при помощи поэлементного сравнения обеих версий;
- при помощи сравнения хэш-функции от конкатенации атрибутов загружаемого среза с хэш-функциями, вычисленными и сохраненными в прошлых загрузках.

При выборе способа с использованием хэш-функции накладываются дополнительные расходы на вычисление и хранение значений этой функции.

Применимость того или иного алгоритма определяется эффективностью по каждой отдельной таблице. В случае небольшого количества сравниваемых атрибутов эффективнее будет поэлементное сравнение за счет отсутствия дополнительных вычислений хэш-функции. Для таблицы, с большим количеством полей может оказаться эффективнее вычисление хеш-функции по всем атрибутам и сравнение её с предыдущим значением для этой записи. Эффективность также зависит от самой выбранной хэш-функции. Таким образом, перед реализацией и применением одного из этих алгоритмов, необходимо проводить их сравнительное тестирование.

1.2. Выделение дельты для загрузки данных из таблицы системы-источника

Инкрементальная выгрузка используется для процессов регулярной загрузки данных в КХД объёмных таблиц.

Выделение дельты для загрузки данных из системы-источника можно разделить на два подхода:

1. Анализ времени изменения. Если бизнес-система гарантирует, что при каждом обновлении данных обновляется и поле, содержащее время изменения данных, то этот метод является очень эффективным для идентификации измененных записей.

2. Выделение по инкрементальному счетчику. Если есть гарантия того, что данные в таблице базы данных только добавляются (использовалась только команда INSERT) и они содержат однозначный идентификатор транзакции, то для идентификации измененных записей (записей, добавленных в таблицу с момента последней транзакции) необходимо знать номер последней транзакции. Указанный метод хорошо подходит для обработки транзакционных таблиц фактов [3].

Оба этих подхода ускоряют загрузку данных с источника, однако имеют два существенных недостатка.

Первый минус заключается в том, что в КХД отсутствует возможность отследить удаление (а во втором случае и изменение) записи, что может вызвать расхождения с источником, и, как следствие, неверные данные в витринах КХД. Часто КХД вынуждено забирать данные не по специальному атрибуту, которое обновляется при каждом изменении записи на источнике, а по какому-нибудь бизнес-полю, например, по дате звонка. Таким образом, если КХД отбирает из таблицы СИ данные о звонках, которые произошли после момента последней загрузки, данные об изменении старых звонков в хранилище не попадут. В таком случае можно расширить дельту и отбирать данные «внахлёст», чтобы одна и та же строка попадала в отбор несколько загрузок подряд, и было возможно отследить её изменение. Глубина такого забора определяется исходя из нагрузочного тестирования (необходимо

проверить, какой объем данных загрузка сможет обработать за приемлемый отрезок времени) и после коммуникации с представителями СИ (необходимо, чтобы они пояснили, на какой глубине хранения могут меняться данные). Способом решения проблемы с отслеживанием удаленных строк может быть присылаемая источником отдельно информация о строках, удаленных за отчётный период (как правило, за последние сутки). Такие строки в КХД также пометятся удалёнными.

Второй недостаток заключается в следующем: приходится полагаться на то, что система-источник на своей стороне создаст и будет в дальнейшем поддерживать работу полей, отвечающих за выделение инкремента со стороны КХД. На практике этот минус встречается достаточно редко, так как если система-источник работает с большими объемами данных, как правило, у неё есть способ однозначно идентифицировать каждую строку числом из последовательности. Это же число можно использовать в качестве инкрементального счётчика в КХД. Если всё же в таблице СИ нет возможности создать такой идентификатор и отсутствует поле, позволяющее определить время изменения записи, от инкрементальной загрузки придётся отказаться или применить её нестандартную модификацию, когда при изменении и добавлении записи в системе-источнике изменённые и добавленные ключи записываются в техническую очищаемую таблицу, на основе которой происходит загрузка в КХД только добавленных или обновленных строк.

2. Анализ способов обновления данных в таблицах витринного слоя

Ключевым требованием к загрузке таблиц витринного слоя является то, что витрина должна быть доступна в любой момент времени. Это означает, что, когда бы бизнес-пользователи ни обратились к объекту, он всегда должен существовать и быть непустым. Если обращение произошло в момент пересчета данных, который еще не завершился, пользователи должны видеть состояние витрины на момент предыдущего расчёта. Как только текущий расчёт завершится, им станет доступна обновлённая информация.

Выделяется два способа обновления данных загрузка объектов витринного слоя:

- полный пересчёт;
- расчёт по дельте.

2.1. Полный пересчет данных

В случае расчета таблиц витринного слоя, как и в случае с детальными данными, основным преимуществом полного пересчёта данных является простота реализации.

Полный пересчет данных представляет собой полную очистку предыдущего состояния и расчет актуальных данных за новый отчётный период. Так как операция вставки может занимать длительное время, доступность витрины обеспечивает механизм переключения синонимов.

Данный механизм заключается в создании двух таблиц с идентичным атрибутным составом и двух синонимов к ним. Один из синонимов, название которого совпадает с названием витрины, будет указывать на актуальный срез данных, а второй «неактивный»– на предыдущий. Во время начала расчёта витрины очищается и заполняется новыми данными «неактивный» синоним, в то время как синоним, совпадающий с названием витрины, доступен для пользователей и содержит в себе предыдущий расчёт. Сразу после окончания вставки в «неактивный» синоним происходит их переключение, и теперь «основной» синоним указывает на таблицу с наиболее свежим расчётом, а второй синоним указывает на предыдущий расчёт.

Такой подход имеет ряд преимуществ:

- разработчику всегда доступен предыдущий срез данных, это облегчает поиск ошибок и атрибутов, определяющихся случайным образом (например, если в отборе данных используется сортировка по полю, которое не уникально в таблице);
- наличие предыдущего среза данных позволяет упростить написание логики атрибутов, которые зависят от своего предыдущего значения;
- полный пересчёт данных в таблице облегчает исправление ошибок разработчиков: достаточно исправить код и запустить расчёт повторно;
- полный пересчёт данных имеет стабильное время выполнения: вне зависимости от количества обновлений в таблицах более низкого уровня пересчёт таблицы не зависнет относительно своих предыдущих расчётов;
- простота реализации.

Недостатками полного пересчёта данных является следующее:

- объем данных в таблицах детального слоя растёт постепенно, но постоянно, вместе с этим будет постоянно расти количество времени, требуемое для полного пересчёта данных в витрине;
- постоянный пересчёт даже тех данных, в которых не было изменений;
- в КХД хранится не только актуальное состояние, но и состояние на момент предыдущего расчёта, если таблица содержит в себе большой объём данных, это может существенно влиять на количество доступной памяти в хранилище;

Таким образом, полный пересчёт данных в витрине подойдёт для небольших объектов (до 10 Гб данных или нескольких десятков миллионов строк), в которых часто происходят обновления существующих данных или необходимо использовать предыдущее состояние для текущего расчёта.

2.2. Выделение дельты для загрузки данных

В случае, если актуальный срез данных объекта витринного слоя занимает большой объём, часто происходит вставка новых строк, а обновление существующих – редко, возможно перевести объект с полного пересчёта на пересчёт по дельте изменений.

Выделение дельты для витрины сильно отличается от выделения дельты для загрузки данных в таблицу детального слоя. Ниже представлены этапы перевода объекта витринного слоя на расчёт по дельте.

1. Полный инициализирующий расчёт. Несмотря на то, сколько времени занимает полная загрузка данных в витрину, она должна быть выполнена хотя бы один раз при первом расчёте объекта. В дельнейшем при плановой загрузке объекта полный пересчёт проводиться не будет. Если полный пересчёт не удается провести в принципе, следует искать ошибку в логике соединений или оптимизировать запрос с помощью хинтов (дополнений к стандарту SQL, которые указывают движку базы данных, как выполнить запрос [4]).

2. Выделение изменившихся ключей из таблиц, участвующих в расчёте. На данном шаге отдельно отбираются ключи, по которым произошли изменения. В процессе разработки обычно именно этот этап занимает большую часть времени и требует наиболее тщательного тестирования.

3. Удаление из текущего состояния витрины обновлённых ключей. При помощи операции DELETE из витрины удаляются ключи, по которым в таблицах детального слоя прошли изменения.

4. Вставка в витрину новых и измененных записей из таблиц детального слоёв. Код этого шага является повторением запроса для полного расчёта объекта с одним важным дополнением: в запросе присутствует строгое соединение с таблицей, в которой собраны новые

и обновлённые ключи таблицы. Таким образом, пересчёт проводится не полного объёма данных, а только того, что действительно изменилось.

При пересчёте витрины по дельте она, так же, как и при использовании механизма переключения синонимов, доступна в любой момент, но во время расчёта существует промежуток времени, когда данные в витрине не консистентны (не согласованы данных друг с другом, противоречивы [5]). Это состояние, когда уже выполнено удаление из витрины обновлённых ключей, но вставка обновлённых значений ещё не завершена. Чтобы минимизировать время, которое витрина находится в этом состоянии, создается ещё одна временная очищаемая таблица, аналогичная по атрибутному составу основной витрине, и добавляется новый шаг перед удалением строк со вставкой во временную таблицу добавленных и обновленных строк по логике из четвёртого этапа. Таким образом, сам четвёртый этап можно заменить на вставку всех строк из новой временной таблицы, что быстрее пересчёта.

Основным преимуществом такого подхода к обновлению данных в витринах является отсутствие необходимости в полном пересчёте. Несмотря на наличие дополнительных шагов (выделение дельты изменений, удаление из витрины по изменившимся ключам), расчёт по дельте часто проходит быстрее.

Однако разработчику стоит помнить о том, что, если в дельту попадет большое количество ключей, которые необходимо обновить, расчёт объекта займёт неожиданно большое количество времени. Например, при обновлении справочника, значения из которого используется в каждой строке витрины, расчёт по дельте займёт гораздо больше времени, чем полный пересчёт, потому что перед тем, как пересчитать все строки в витрине, будет выполнено удаление всех существующих строк.

Таким образом, расчёт по дельте подходит для объектов, полный пересчёт которых занимает слишком много времени и памяти, при этом помимо текущего времени полного расчёта необходимо проанализировать темпы роста данных в таблицах детального слоя, на которых строится объект. Например, если на момент разработки полный расчёт занимает сравнительно небольшое время, но влияющая таблица детального слоя растет ежедневно на несколько миллионов записей, разработчик должен предусмотреть расчёт по дельте чтобы не допустить зависания полного пересчёта в будущем.

Заключение

Обновление данных в объектах как детального, так и витринного слоя КХД можно разделить на два подхода: обновление всего объекта целиком и обновление ограниченного набора записей. Каждый из этих способов имеет свои преимущества и недостатки, и часто сложно предсказать заранее, какой будет более эффективен в каждом конкретном случае.

При разработке нового объекта разработчик должен проверить возможность использования обоих подходов и принять решение о применении какого-либо способа, основываясь на результатах тестирования.

Информация о научном руководителе: к.ф.-м.н., доцент Барановский Е. С.

Литература

1. Налоговый кодекс Российской Федерации (часть первая) от 31.07.1998 № 146-ФЗ (ред. от 26.02.2024) – Режим доступа: https://www.consultant.ru/document/cons_doc_LAW_19671/4871a92ac4bb0f9b7d4ae1bc9923cdb81a6d69cf/// – (Дата обращения: 31.03.2024)

2. Попова-Коварцева Д. А. Основы проектирования баз данных. / Д. А. Попова-Коварцева, Е. В. Сопченко – Самара: Изд-во Самарского университета, 2019. – 112 с.
3. Инкрементная загрузка больших объемов данных в Data Lake – Режим доступа: https://bigdataschool.ru/blog/how-to-implement-data-pipeline-for-incremental-loading.html – (Дата обращения: 20.03.2024).
4. Мишра С. Секреты Oracle SQL. / С. Мишра, А. Бьюли – СПб : Символ Плюс, 2003. – 368 с.
5. Консистентность данных – Режим доступа: https://studfile.net/preview/3707302/ page:11/ – (Дата обращения: 31.03.2024).

УДК 004.716

ИНСТРУМЕНТЫ РАЗРАБОТКИ ПО ДЛЯ ПРОГРАММНО-ОПРЕДЕЛЯЕМОГО РАДИО В ЗАДАЧАХ РАДИОКОНТРОЛЯ СПУТНИКОВЫХ СИГНАЛОВ

Е. В. Кривошлыкова

Воронежский государственный университет

Введение

Спутники – одни из ключевых элементов современных систем радиосвязи. Комплекс задач, реализующих радиоконтроль спутниковых сигналов является важнейшим компонентом в обеспечении безопасности и надежности космических систем связи и навигации. Эти задачи направлены на обеспечение мониторинга, анализа сигналов и предотвращение возможных внешних воздействий, таких как помехи или вмешательства. Любая система спутниковой связи является специализированным инструментом, поэтому одной из проблем радиоконтроля таких космических систем является обеспечение универсальности этого вида аппаратуры. Одним из вариантов обеспечения универсальности системы спутникового радиоконтроля стало использование программно-определяемых радиосистем (SDR), однако в силу специфики области применения, к таким SDR решениям предъявляется ряд дополнительных технических требований. В статье рассмотрено обобщение этих технических требований включая аппаратную и программную часть SDR-систем спутникового радиоконтроля.

1. Программно-определяемое радио

Программно-определяемое радио (SDR) представляет собой технологию, при которой обработка радиосигналов реализуется программным обеспечением, а аппаратная составляющая является лишь «преобразователем среды», т.е. трансформирующим принимаемый радиосигнал в набор его числовых характеристик, которые далее анализируются программным обеспечением. Это позволяет достичь гибкости, масштабируемости и эффективности в процессе разработки и использования радиосистем. В контексте радиоконтроля спутниковых сигналов SDR играет важную роль, обеспечивая возможность адаптации и модификации радиосистем в реальном времени. Благодаря программной реализации основных радиотехнических функций, таких как фильтрация, модуляция, демодуляция и обработка сигналов, SDR позволяет инженерам эффективно мониторить и анализировать спутниковые сигналы.

1.1. Роль SDR в радиоконтроле

Одним из основных преимуществ SDR в радиоконтроле является его способность работать с различными типами сигналов и протоколов связи без необходимости изменения аппаратных компонентов. Это позволяет легко адаптировать систему для работы с различными спутниковыми сигналами и устройствами, что особенно важно в условиях быстро меняющейся технологической среды космической навигации и связи. Таким образом, SDR представляет собой неотъемлемый элемент в арсенале инструментов для радиоконтроля спутниковых сигналов, обеспечивая гибкость, масштабируемость и высокую производительность в выполнении радиотехнических задач.

2. Требования к аппаратной составляющей SDR-системы

Так как любая SDR-система – совокупность программных и аппаратных объектов, то технические требования к них также состоят из двух частей: требованиям, предъявляемым к аппаратной части и программной части.

2.1.Требования в SDR-приемнику

Спутниковые системы радиосвязи являются очень «плотными» с точки зрения количества информации, передаваемой в рабочих полосах сигналов. Это достигается использованием высоких несущих частот радиосигналов и сигнальных созвездий высоких порядков (QAM-64, PSK-16 и т.д.). Для обработки таких сигналов необходимы АЦП с высокой разрядностью (от 12 бит и выше).

Большая часть спутниковых систем связи сосредоточена в таких частотных диапазонах как Ка-диапозон, L-диапазон (т.е. от 10 ГГц и выше). С применением понижающих частоту конверторов, она отображается в диапазон от 1 ГГц до 4 ГГц. Таким образом, применяемые для спутникового радиоконтроля SDR приемники, должны иметь возможность работы с сигналами в диапазоне от 1 до 4 ГГц.

Сигналы спутниковых систем – широкополосные, например, спутниковое телевидение, использующее протокол DVB-S и DVB-S2 имеет полосу в 8 МГц. Т.е. для демодуляции DVB-S/S2 сигналов, мгновенная полоса SDR-приемника должна быть не менее 8 МГц. Среди спутниковых систем гражданского назначения самым широкополосным является сигнал системы Starlink с шириной полосы 160 МГц.

Таким образом требования к приемнику системы радиоконтроля следующие:

- верхняя граница диапазона частот (F_{max}): не менее 4 ГГц (при использовании дополнительных конверторов в транспондерах);
- разрядность АЦП (M): не ниже 12 бит;
- мгновенная полоса обработки сигнала (F_{tsx}): не ниже 160 МГц.

Поток оцифрованных данных от такого приемника будет составлять не менее 228 Мбайт/сек .

2.2. Требования к ПЭВМ обработки сигналов

Для определения технических характеристик ПЭВМ для обработки спутниковых сигналов необходимо ответить на вопрос: «На сколько глубоко должен осуществляться радиоконтроль?». Если задача контроля – отслеживание изменений в панорамном спектре (т.е. отслеживании частотно-временных и энергетических характеристик сигналов), то особых требований к ПЭВМ не предъявляется. В этом случае осуществляется перестройка центральной частоты используемого SDR-приемника с последующим «сшиванием» полос обработки сигналов, от начальной частоты до конечной.

Однако ситуация кардинально меняется, если мы хотим исследовать сигнал глубже, а именно узнать его тип модуляции, что передается и какой помехоустойчивый код используется. В этом случае у нас предъявляются очень серьезные требования к оперативной памяти и непосредственно центральному процессору. При медленном процессоре обработка данных занимает достаточно много времени. Это приведет к заполнению дисковой системы и утрате большого количества данных. Потеряется целостность передаваемой информации и оцифрованного сигнала. При медленной дисковой подсистеме наблюдается схожий эффект. Поступившие в оперативную память оцифрованные данные попадают в оперативную память,

затем они кешируются на жесткий диск. Данные пишутся с меньшей скоростью, чем поступают в оперативную память, поскольку HDD существенно медленнее ОЗУ. В конечном результате полезный сигнал будет утерян. Для эффективной обработки сверхширокополосных сигналов применяют методы многопроцессорной обработки сигналов или обработку радиосигналов на массивах GPU. Объем оперативной памяти выбрается таким образом, чтобы жесткий диск успевал кешировать и записывать данные со скоростью, при которой определенный процент оперативной памяти всегда оставался свободным для предотвращения переполнения.

2.3. Требования к программной составляющей SDR-системы

Спутниковые сигналы являются сложными по своей структуре в силу используемых типов модуляций, манипуляций и сигнально-кодовых конструкций.

Сигнально-кодовая конструкция представляет собой сочетание типа модуляции (или манипуляции) и помехоустойчивого кода (напр. Кодов Рида-Соломона). Отсюда вытекают два основных условия: необходимость поддержки сложных модуляторов и демодуляторов, а также алгоритмов расширения спектра для борьбы с различными искажениями. Когда речь идет об алгоритмах борьбы с переотражениями сигнала, в первую очередь упоминается OFDM (Orthogonal Frequency Division Multiplexing) – специальный метод расширения спектра, разработанный для преодоления таких переотражений. В некоторых источниках его называют отдельным видом модуляции, но на самом деле это расширение модуляции, а не модуляция сама по себе. Для многоканальных спутниковых систем передачи данных используется не OFDM, а COFDM (Coded Orthogonal Frequency Division Multiplexing) – это метод разделения сигналов по кодовому принципу. Он позволяет абонентам, работающим в одном частотном домене, использовать математические методы, такие как автокоррелируемые функции Уолша и другие, для разделения сигналов.

В настоящее время существуют различные среды разработки, способные поддерживать работу с технологиями для моделирования систем спутникового радиоконтроля. GNU Radio и Phosphor SDR являются бесплатными средами разработки. GNU Radio – один из ключевых инструментов разработки программно-определяемого радио. Он предоставляет широкий набор инструментов и библиотек для разработки радиосистем на языке Python. Программа позволяет инженерам реализовывать различные радиотехнические алгоритмы, создавать и обрабатывать радиосигналы в реальном времени. Благодаря открытой архитектуре и активному сообществу разработчиков, GNU Radio широко используется в задачах радиоконтроля спутниковых сигналов. Широкие возможности для моделирования как бесплатно, так и в проприетарной среде может предоставить нам MATLAB. Он содержит мощные инструменты для моделирования и анализа радиосистем. Simulink (графический инструментарий MATLAB) позволяет создавать графические модели систем радиосвязи, что значительно упрощает проектирование и отладку. Богатый выбор библиотек и инструментов, доступных в MATLAB, делают его популярным выбором среди инженеров, работающих в области радиоконтроля.

Другие важные инструменты SoapySDR и gr-osmosdr: эти открытые библиотеки и фреймворки обеспечивают доступ к программно-определяемым радиоустройствам и интеграцию с другими инструментами разработки. SoapySDR и gr-osmosdr позволяют разработчикам легко взаимодействовать с радиоаппаратурой различных производителей и создавать гибкие системы радиоконтроля. Перечсленные среды разработки обладают необходимыми характеристиками для работы с современными технологиями и позволяют проводить как полунатурное моделирование, так и более сложные исследования в области спутникового радиоконтроля**.**

Инструменты разработки программно-определяемого радио (SDR) играют ключевую роль в обеспечении эффективного радиоконтроля спутниковых сигналов. Гибкость, масштабируемость и высокая производительность, предоставляемые этими инструментами, делают их необходимыми компонентами в арсенале инженера-радиотехника.

С помощью перечисленных инструментов инженеры могут разрабатывать и отлаживать системы радиосвязи и навигации. Благодаря активному развитию и поддержке этих инструментов со стороны сообщества разработчиков, радиоконтроль спутниковых сигналов становится более доступным и эффективным. Инструменты разработки SDR не только улучшают качество и надежность спутниковых систем связи и навигации, но и играют важную роль в развитии космической отрасли в целом, обеспечивая безопасность и надежность космических миссий и повышая эффективность использования космического пространства.

Литература

1. Банников И. М. Радиоприёмные устройства и радиоприёмные комплексы перспективных узлов коротковолновой связи / В.А. Березовский, М. М. Валеев, Г. К. Хазан // Международная научно-техническая конференция «Радиотехника, электроника и связь, РЭиС-2011», -2011. -С. 121-125.

2. Николашин Ю. Л. Перспективные методы повышения помехоустойчивости декаметровых радиолиний / П. А. Будко, Е. С. Жолдасов, Г. А. Жуков // Наукоемкие технологии в космических исследованиях Земли, №1. -2004. - С. 30-37.

3. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов. / Р. Блейхут// - 1989. -С. 448.

4. Тиан З. Адаптивные фильтры для коммуникаций, основанных на разреженности: применение к формированию спектра / З. Тиан, Д. Б. Джианнакис // Журнал IEEE по обработке сигналов. -2011. -Т. 28, № 6. -С. 113-128.

5. Лопес-Бените М. Архитектуры программно определяемого радио: обзор /М. Лопес-Бените, Ф. Касадеваль // Журнал «Обзоры и учебники по связи» IEEE. -2013. -Т. 15, № 4. -С. 1691-1717.

УДК 004

РАЗРАБОТКА ИНТЕРФЕЙСА МОБИЛЬНОГО ПРИЛОЖЕНИЯ-ОРГАНАЙЗЕРА “Achievement”

С. П. Крохина

Воронежский государственный университет

Введение

В современном мире, из-за роста деловой загруженности, людям необходимо организовывать более быстро и тщательно свою жизнь. Следовательно разработка мобильных приложений по самоорганизации/планированию актуальна по таким причинам как:

1. Растущая потребность в эффективности: органайзеры предлагают возможность оптимизировать план действий, управлять задачами и поддерживать баланс между различными сферами жизни.
2. Удобство и персонализация: органайзеры предлагают широкий выбор средств, которые позволяют пользователям адаптировать их под себя, что делает их более удобными и функциональными.
3. Улучшение психологического состояния: органайзеры помогают людям чувствовать себя более организованными и контролирующими свою жизнь, что снижает уровень стресса и способствует психологическому комфорту.

Также мобильное приложение всегда под рукой, так как практически каждый человек носит с собой смартфон. Напоминания и уведомления позволяют человеку не забыть о событиях и задачах, что помогает, например, развивать полезные привычки, эффективно планировать время. Также приложение позволяет проанализировать свою активность, чтобы в дальнейшем было удобнее корректировать свои задачи.

Существует множество приложений такого типа, однако большинство из них позволяют только записать задачу и обозначить срок, что является не очень продуктивным способом достижения целей, поэтому необходимо разработать приложение – органайзер с дополнительной функциональностью, с помощью которой человек сможет эффективно самоорганизовываться.

1. Постановка задачи

Требуется разработать интерфейс кроссплатформенного мобильного приложения-органайзера «Achievement», при помощи которого пользователь сможет записывать свои задачи, подзадачи, отмечать их выполненными, определять приоритет, срок окончания, а также на основе этих данных генерировать эффективный план выполнения поставленных задач. Также пользователь сможет просматривать отчёт по своей активности в виде удобных графиков и сможет увидеть общий рейтинг, показывающий, насколько качественно пользователь выполняет задачи и достигает свои цели.

2. Основная функциональность приложения

Основные действия пользователя при работе с приложением представлены на use-case диаграмме (рис. 1).

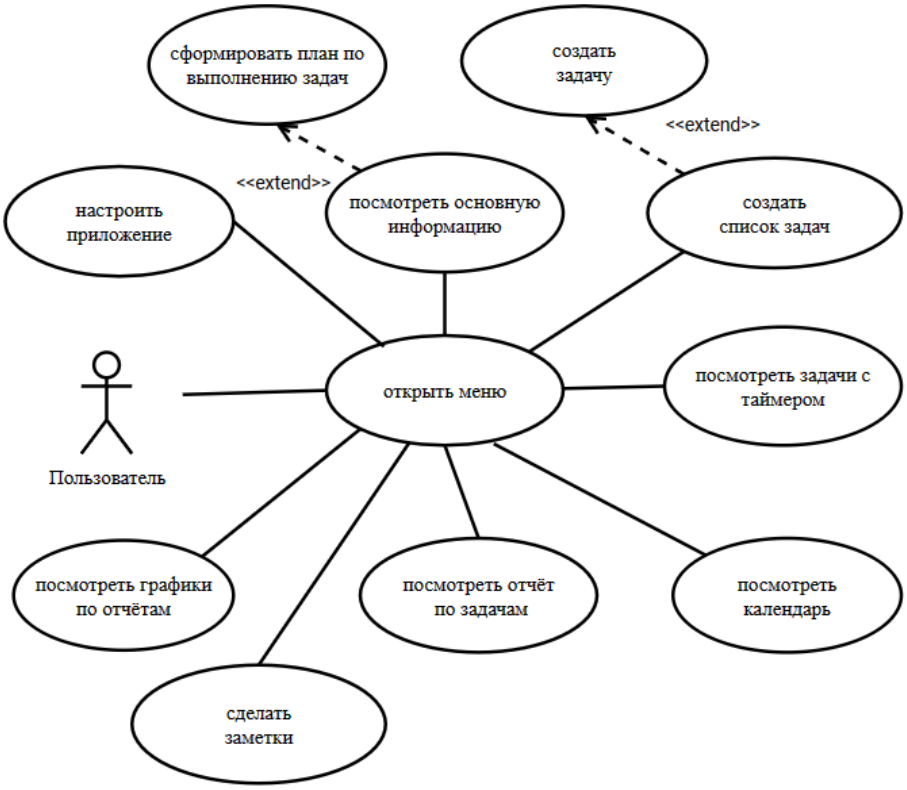


Рис. 1. Основные функции приложения

На основе диаграммы предполагается построить меню, в котором пользователю будут доступны различные вкладки приложения с соответствующими функциями.

3. Интерфейс приложения

Для удобства пользователя был создан Navigation Drawer (рис. 2) – меню, которое открывается сбоку (можно настроить, справа или слева) [3]. В нём находятся основные вкладки приложения, такие как:

1. Главное – содержит основную вкладку о задачах, также содержит вкладку, которая позволяет создать эффективный план с учётом параметров задач.
2. Списки – содержит вкладки со сферами жизни, внутри каждой вкладки могут быть еще подвкладки, а в них уже соответствующие задачи.
3. Трекеры – содержит задачи, для которых требуется отследить время, в течение которого задачи выполняются.
4. Календарь – календарь, в котором можно удобно смотреть что и когда нужно сделать, а также быстро добавить задачу.
5. Заметки – заметки с указанием даты, когда заметка была создана.
6. Достижения – выполненные задачи, анализ и отчётность.
7. Графики – общая вкладка для всех графиков.
8. Настройки – настройки и персонализация.

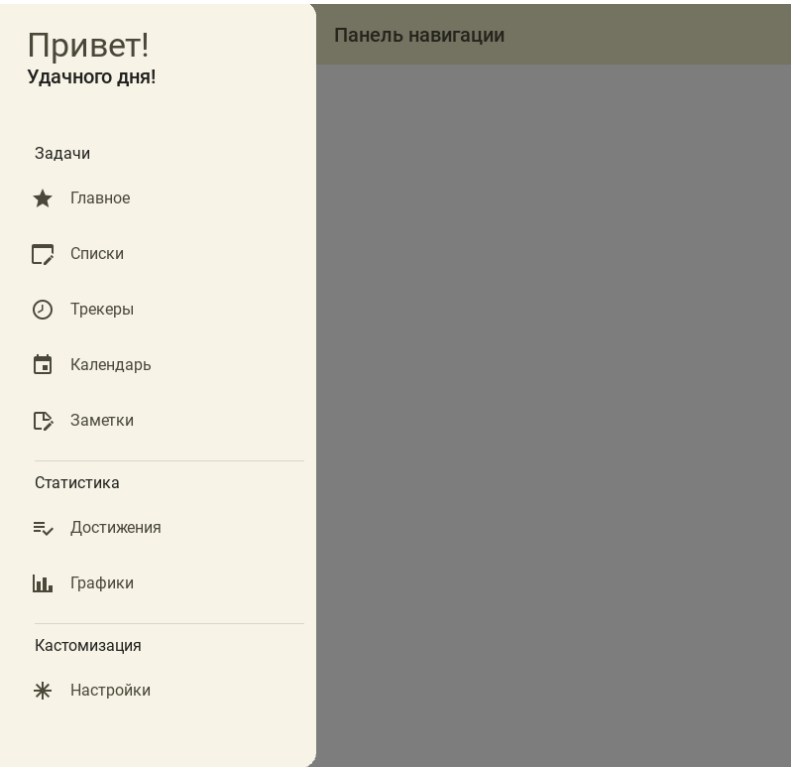


Рис. 2. Меню приложения

Рассмотрим теперь каждую вкладку отдельно.

2.1. Главное

В данной вкладке будет отображаться общая информация о задачах. В Главном три дополнительные вкладки: информация на ближайшее время (сегодня и завтра), планирование и подробный отчёт за сегодняшний день. Вкладки внизу экрана сделаны в виде Bottom Navigation – навигация в нижней части экрана [2].

2.2. Списки

Тут две вкладки: первая – отдельное меню с названиями списков, каждый из которых пользователь может создать, удалить и кастомизировать. Можно создавать подсписки, а в них уже задачи; вторая – будет отображать список всех задач, которые можно сортировать по различным параметрам и осуществлять поиск.

2.3. Трекеры

В этой вкладке два раздела: в первом задачи, для которых таймер отслеживает время их выполнения, во втором задачи, для которых необходимо регулярное напоминание в течение дня. Выглядит аналогично вкладке “Списки”, но с другими иконками и соответственно содержимым.

2.4. Календарь

Вкладка Calendar предназначена для наглядного отображения дат и запланированных

задач на эти даты: при нажатии на конкретную дату, будет высвечиваться список дел на этот день [1, 2]. Тут же можно быстро добавить задачу или пометку. Выбор даты реализован при помощи инструмента MDDatePicker [3].

2.5. Заметки

Данная вкладка представляет собой текстовое поле с обозначением даты, когда заметка была записана. Заметки можно ассоциировать с задачами. Также имеется возможность добавлять напоминания к заметкам.

2.6. Достижения

Здесь можно будет отследить весь прогресс за любой выбранный пользователем промежуток времени. В этом разделе отмечается сколько задач выполнено, не выполнено, анализируется их сложность, важность и другие параметры. На основе этих данных формируется общая оценка деятельности пользователя.

2.7. Графики

На основе раздела Achievement формируются различные графики за определённый временной промежуток времени. Например, можно отследить с какой скоростью и какие задачи были выполнены; в каком списке какое было выполнено количество задач.

2.8. Настройки

Данный раздел предназначен для настройки приложения, в котором имеется возможность будет настроить тему экрана (тёмная/светлая), можно будет открыть раздел “помощь” – небольшое руководство по пользованию. Также здесь отображены данные о приложении, его версия, контакты автора и т. д.

Заключение

В ходе работы был разработан дизайн меню приложения-органайзера “Achievement” в соответствии с требованиями, представленными на use-case диаграмме (рис. 1). Также был разработан внешний вид для каждой вкладки меню. В итоге получился многофункциональный удобный интерфейс с простым дизайном для мобильного приложения-органайзера, что позволяет человеку ставить цели и эффективно их достигать.

Литература

1. Ulloa, R. Kivy – Interactive Applications and Games in Python / R. Ulloa. — Second Edition. — Packt Publishing Ltd., 2015. — 250 с. — ISBN 978-1-78528-692-6.

2. Phillips, D. Creating Apps in Kivy / D. Phillips. — O’Reilly Media, Inc., 2014. — 125 с. — ISBN 978-1-491-94667-1.

3. Постолит, А. Разработка кроссплатформенных мобильных и настольных приложений на Python: Практическое пособие / А. Постолит. — 2022. — 676 с.

УДК 004.94

МЕТОД СИМУЛЯЦИИ И АНИМАЦИИ ВОДНОЙ ПОВЕРХНОСТИ НА ОСНОВЕ ВОЛН ГЕРСТНЕРА

А. С. Крутько

Воронежский государственный университет

Введение

Поверхность воды – один из наиболее сложных и удивительных объектов природы, которые часто требуется воссоздать в виртуальной среде. Симуляция волн на поверхности воды – важный аспект визуальных эффектов и компьютерной графики, который находит применение в различных областях, от игровой индустрии до кинематографа. В данной работе будет рассмотрена одна из реализаций, использующая модель волн Герстнера для симуляции и анимации волн на поверхности воды.

1. Постановка задачи

В данной работе под глубокой водой будет пониматься такое тело воды, для которого глубина тела воды значительно больше высоты волны.

Необходимо описать реализацию шейдера, использующего волны Герстнера [1] для создания симуляции и анимации движения волн на поверхности глубокой воды. Данная реализация должна соответствовать следующим требованиям:

1. Необходимо обосновать почему были сделаны решения в сторону использования тех или иных формул, используемых в вычислениях шейдера;
2. Должны быть описаны параметры, на основе которых работает шейдер, чтобы пользователь мог с точностью выстроить нужную ему форму и поведение волн;
3. Должна быть предусмотрена корректность работы освещения с геометрией объекта;
4. Работа шейдера должна происходить в режиме реального времени, а обработка всех вычислений и отрисовка поверхности воды не должны занимать много ресурсов устройства.

2. Описание шейдера

2.1. Теоретическая основа шейдера

Волны представляют собой колебания. Простейшим видом описания колебаний с помощью математических функций является описание через синусоиду. Используя данный факт, зададим высоту волны через синус и построим следующее соотношение для координат, где P – вектор, определяющий координаты вершины плоскости:

$$P = \begin{bmatrix} x \\ \sin(x) \\ z \end{bmatrix}, \quad (1)$$

У волны существует несколько различных параметров, от которых зависит её форма и внешний вид. Введем дополнительный параметр – длину волны λ, получим зависимость

количества волн от данного параметра для функции $\sin x$. Так как полный период функции $\sin(x)$ равен 2π, то необходимо разделить аргумент на длину волны λ, а затем умножить на 2π. В результате данных преобразований получим новый параметр k, определяющий количество волн в периоде 2π:

$$k = \frac{2\pi}{\lambda}, \tag{2}$$

Помимо длины волны необходимо также задать её амплитуду. Для задания амплитуды волны необходимо умножить синус на определенное число. Введем новый параметр α для управления амплитудой волны. Аналогично для скорости волны введем ещё один дополнительный параметр c. Чтобы вершины плоскости двигались, а не находились в стационарном положении необходимо для них задать зависимость ещё от одного параметра – промежуток времени между двумя кадрами t.

Используя вышеперечисленные параметры, можно получить значение, определяющее высоту волны. Запишем новый вектор, определяющий координаты вершин:

$$P = \begin{bmatrix} x \\ asin(k(x-ct)) \\ z \end{bmatrix}, \tag{3}$$

В компьютерной графике для корректного взаимодействия освещения с объектами необходимы данные о нормали, бинормали и касательной каждой вершины геометрии данных объектов (рис. 4).

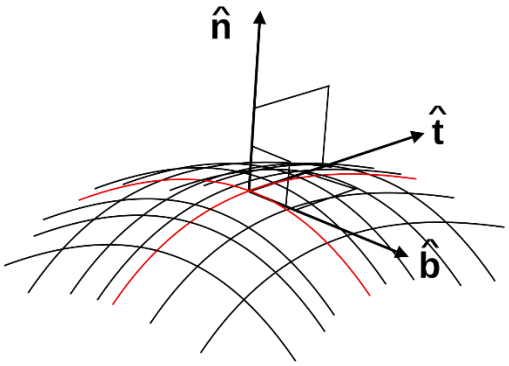


Рис. 4. Расположение векторов нормали, бинормали и касательной на вершине геометрии

Необходимо с использованием формул для трёхгранника Френе [2] пересчитать нормаль, касательную и бинормаль с учетом нового вектора координат вершин (3).

Для поверхности, построенной с помощью (3), касательная описывается следующим образом:

$$T = P' = \begin{bmatrix} x' \\ \alpha \sin(k(x-ct))' \\ z' \end{bmatrix}, \tag{4}$$

В нашем случае аргумент функции синуса тоже функция. Для упрощения записи $a\sin(k(x-ct))$ можно записать как $P_y = asin(f(x))$, где $f(x) = k(x-ct)$ (для удобства записи далее просто f). Посчитав сложную производную, получим:

$$T = \begin{bmatrix} 1 \\ k\alpha \cos f \\ 1 \end{bmatrix}, \tag{5}$$

Для вывода итогового вектора касательной необходимо нормализовать вектор, полученный в (5).

Исходя из свойств трехгранника Ферне [2] вектор бинормали можно записать как единичный вектор $B = (1,0,1)$. Согласно формулам Ферне вектор нормали является векторным произведением двух векторов $B = (1,0,1)$ и $T = (1, k\alpha \cos f, 1)$:

$$N = \begin{bmatrix} -k\alpha \cos(f) \\ 1 \\ -k\alpha \cos(f) \end{bmatrix}, \tag{6}$$

Волны, строящиеся из синусоид, простые – они не задают форму настоящих волн. Используем волны Герстнера (рис. 5) [1] для симуляции и анимации реалистичных больших волн на плоскости.

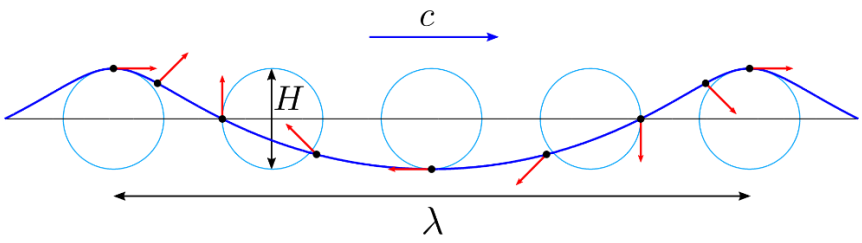


Рис. 5. Схематическое представление волн Герстнера

Как можно заметить на рис. 5 для волн Герстнера движение каждой точки поверхности описывается в виде окружности, закрепленной в определенной точке фиксации. Кривая на рисунке описывает непосредственно волну, а векторы, направленные из точек – траекторию движения самих точек поверхности. При приближении гребня волны точка движется вместе с ним, после его прохождения она движется назад к новому гребню волны.

Чтобы получить из синусоидальной волны (3) волну Герстнера необходимо преобразовать её в окружность. Это можно сделать путем введения выражения $ka\cos(f)$, которое в совокупности с $ka\sin(f)$ будет описывать окружность. Также необходимо добавить для осей x и z смещение на соответствующие изначальные позиции вершины тем самым задав их как точки фиксации. Запишем координаты x и z вершины геометрии как сумму вышеперечисленного:

$$P = \begin{bmatrix} x + ka\cos(f) \\ ka\sin(f) \\ z + ka\cos(f) \end{bmatrix}, \tag{7}$$

Так как были изменены координаты вершин, необходимо пересчитать касательные и нормали, аналогично (4) и (5) считаем производные и получаем вектор касательной:

$$T = \begin{bmatrix} 1 - k\alpha \sin(f) \\ k\alpha \cos(f) \\ 1 - k\alpha \sin(f) \end{bmatrix}, \tag{8}$$

Можно заметить, что для вектора нормали описанного в (6) координатам x и z соответствуют координата y в (5), умноженная на -1. Для нового вектора нормали все аналогично:

$$N = \begin{bmatrix} -k\alpha\cos(f) \\ 1-k\alpha\sin(f) \\ -k\alpha\cos(f) \end{bmatrix}, \quad (9)$$

На данном этапе волны принимают желаемый вид, но не учитывается ситуация, при которой значения амплитуды слишком велики. Например, значение координат x и z для вектора касательной T могут стать отрицательными если $k\alpha > 1$, в данной ситуации вектор касательной будет направлен в обратную сторону, в результате чего на волнах будут видны закругления с разрывом (рис. 6). Чтобы избежать подобного, необходимо использовать иную форму записи амплитуды волны, например, в некоторых случаях для волн можно проследить следующую зависимость между длиной волны и амплитудой:

$$\alpha = \frac{e^{kb}}{k}, \quad (10)$$

В данном случае параметр b отвечает за поверхностное давление. Таким образом, чем сильнее давление, тем более плоскими будут волны. В случае $b = 0$ мы получаем амплитуду $\alpha = \frac{1}{k}$, что задаёт гребни волны с углом в 0° прежде геометрия волны начнёт ломаться в вершине гребня волны как это показано на рис. 6.

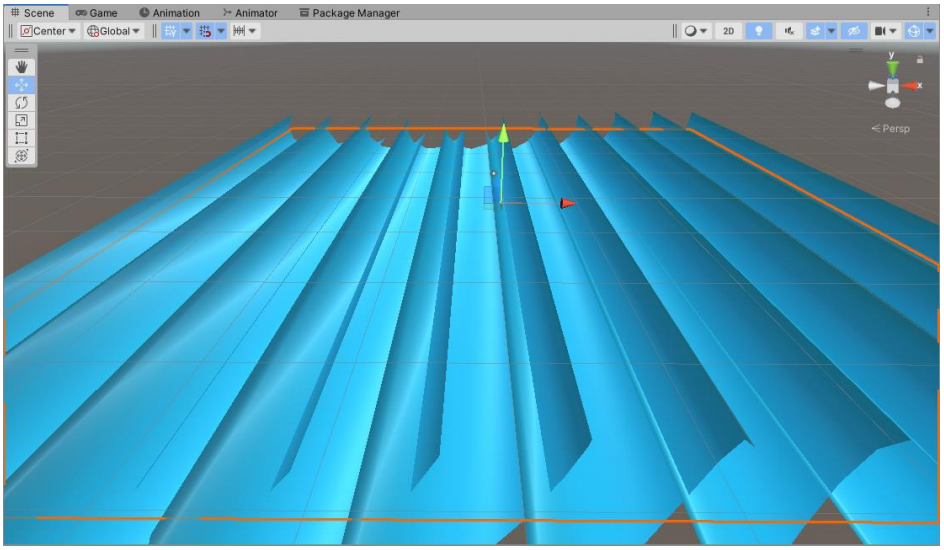


Рис. 6. Закругления с разрывом на гребнях волн

Итак, необходимо заменить параметр, отвечающий за амплитуду волны. Это можно сделать путем её задания через степень крутизны – параметр s, таким образом $\alpha = \frac{s}{k}$. Перепишем (7), (8) с использованием данного параметра:

$$P = \begin{bmatrix} x + \frac{s}{k}\cos f \\ \frac{s}{k}\sin f \\ z + \frac{s}{k}\cos f \end{bmatrix}, \quad (11)$$

$$T = \begin{bmatrix} 1 - s\sin f \\ s\cos f \\ 1 - s\sin f \end{bmatrix}, \quad (12)$$

Нормаль задаётся аналогично (9).

В реальности, скорость волн не задаётся отдельно, она зависит от других параметров волны, в частности количества волн. Так как в данной работе рассматриваются волны в глубокой воде, то их скорость можно описать следующим образом [3]:

$$c = \sqrt{\frac{g}{k}}, \quad (13)$$

где $g = 9.81$ – сила земного притяжения.

Для задания направления движения волн будем использовать единичный вектор $D = \begin{bmatrix} D_x \\ D_z \end{bmatrix}$. С учетом данного вектора f принимает вид: $f = k(D_x x + D_z z - ct)$. Аналогично просчитывается и положение вершины путем умножения косинусов в x и z координатах на соответствующие значения.

$$P = \begin{bmatrix} x + D_x \frac{s}{k}\cos f \\ \frac{s}{k}\sin f \\ z + D_z \frac{s}{k}\cos f \end{bmatrix}, \quad (14)$$

Для данных значений необходимо будет пересчитать векторы касательной T и бинормали B, вектор нормали можно получить через векторное произведение векторов (15) и (16).

$$T = \begin{bmatrix} 1 - D_x^2 s\sin f \\ D_x s\cos f \\ -D_x D_z s\sin f \end{bmatrix}, \quad (15)$$

$$B = \begin{bmatrix} -D_x D_z s\sin f \\ D_z s\cos f \\ 1 - D_z^2 s\sin f \end{bmatrix}, \quad (16)$$

В реальности на поверхности воды почти никогда не бывает лишь одной волны, зачастую водная поверхность представляет собой сложную комбинацию из множества волн, для решения подобной проблемы необходимо представить координату вершины волны в виде суммы. Рассмотрим на примере координаты x:

$$P_x = x + \sum_{i=1}^{n} D_{ix} \frac{s_i}{k_i} \cos f_i,\ f_i = k(D_i \cdot \begin{bmatrix} x \\ z \end{bmatrix} - ct) \qquad (17)$$

Таким образом, можно задавать множество волн, каждая из которых будет по-своему влиять на поверхность и придавать ей желаемый пользователем вид. На рис. 9 можно заметить, как две волны меняют геометрию поверхности, тем самым симулируя движение волн по поверхности воды.

2.2 Пример работы шейдера

На основе теории, описанной в предыдущем разделе, был реализован шейдер в графическом движке Unity, используя built-in систему обработки графики, на языке ShaderLab. Разберём на примере работу данного шейдера. Рассмотрим плоскость, состоящую из квадратов единичного размера площадью 10×10 единиц. В данном случае для нее период волны будет равен $\frac{10}{2\pi} \approx 1.59$. У функции синуса длина волны будет равна её полному периоду, а именно $2\pi \approx 6.28$. На рис. 7 представлен пример генерации синусоидальной волны для подобной плоскости.

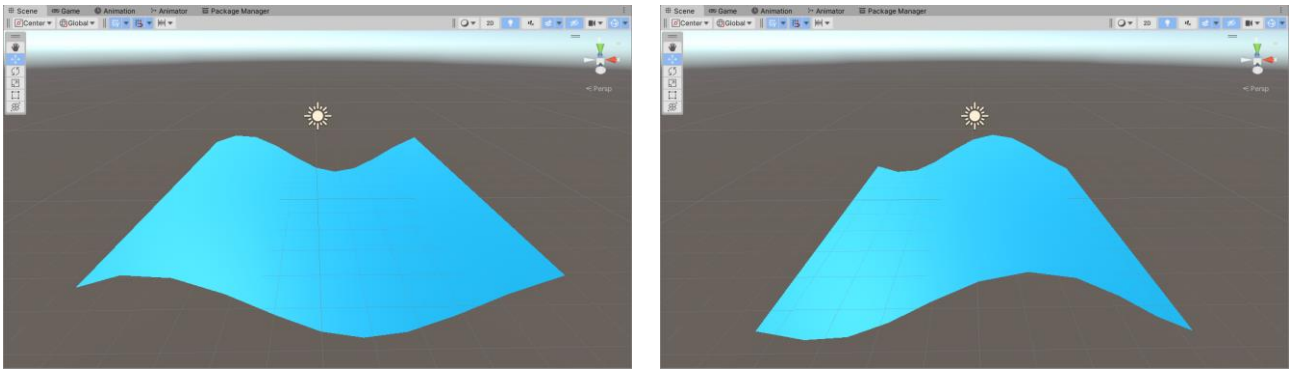


Рис. 7. Результат работы шейдера

Как можно заметить, освещение сцены взаимодействует с данной волной некорректно, свет отражается от поверхности так, будто данная поверхность – обычная плоскость. Это можно исправить, применив новые нормали (6). На рис. 8 можно увидеть разницу в отражении света от плоскости до использования новых нормалей и после.

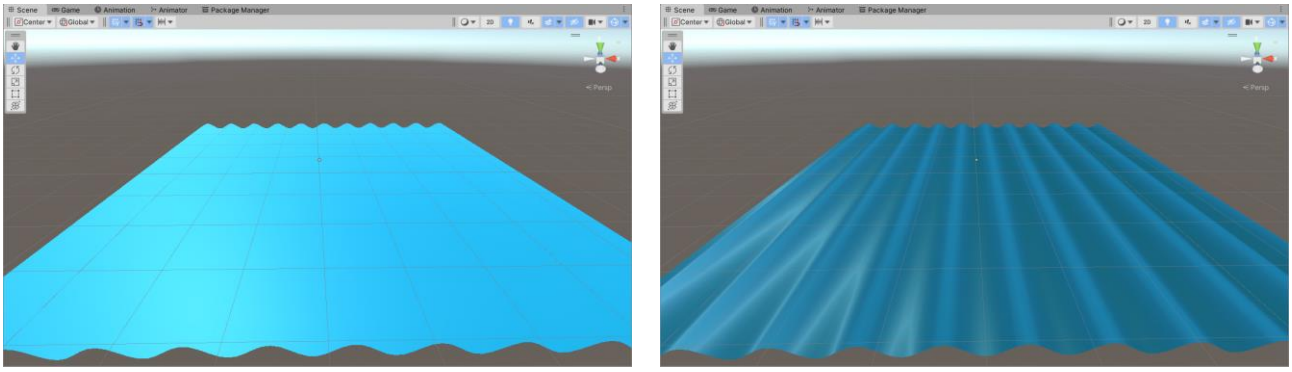


Рис. 8. Сравнение работы шейдера до введения новых вычислений и после

Чтобы задать волнам более естественный вид, воспользуемся комбинацией волн, реализуемой с помощью выражения (17). Зададим две волны со следующими параметрами:

1. Длина волны в $\lambda = 60$, вектор направления $D = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$, степень крутизны $s = 0.5$;
2. Длина волны в $\lambda = 31$, вектор направления $D = \begin{bmatrix} 0.86 \\ 0.51 \end{bmatrix}$, степень крутизны $s = 0.50$;

С данными параметрами получим волны со следующими характеристиками:

1. Период волны $\frac{60}{2\pi} \approx 9.5$, амплитуда $\alpha \approx 0.05$, скорость $c \approx 1$;
2. Период волны $\frac{31}{2\pi} \approx 4.9$, амплитуда $\alpha \approx 0.1$, скорость $c \approx 1$;

На рис. 9 можно увидеть результат работы шейдера с данными параметрами.

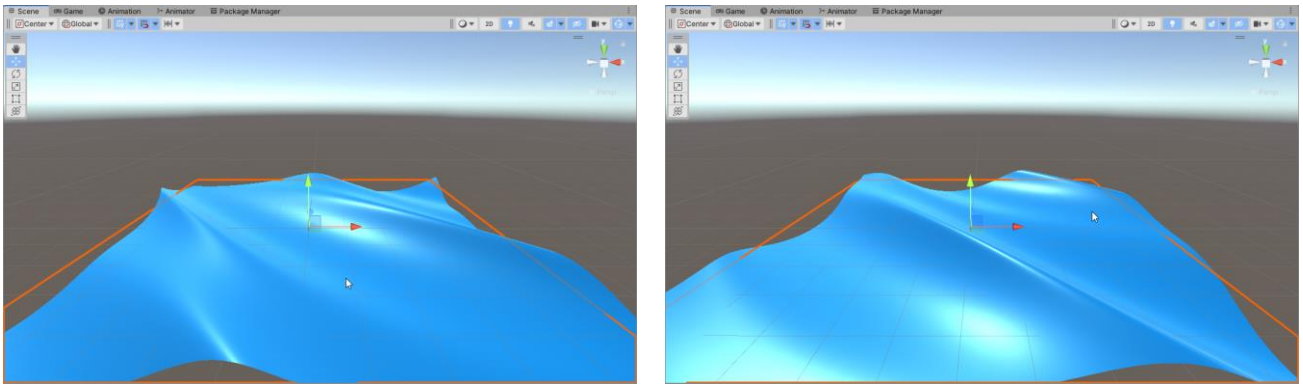


Рис. 9. Итоговый результат работы шейдера

Заключение

В заключении работы выделим основные результаты:

1. Рассмотрен один из методов реализации анимации и симуляции водной поверхности.
2. Приведено детальное описание параметров и зависимостей между ними и частями шейдера, с помощью чего пользователь может настроить желаемую сцену.
3. Обработка приблизительно 124100 полигонов занимает у видеокарты примерно 0.4 мс (Рис. 10), что указывает на низкую нагрузку алгоритма на систему.

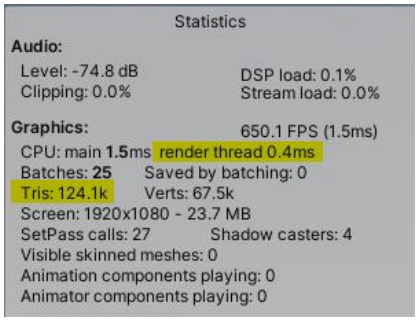


Рис. 10 Статистика производительности сцены игрового движка

Можно сделать вывод что в результате проведенной работы были выполнены все поставленные задачи. Реализация данного шейдера показывает основные концепции при

решении подобного рода задач, а результат его работы можно использовать в проектах, не требующих большого количества деталей и упора в реалистичность с точки зрения визуальной составляющей сцены игрового движка.

Литература

1. Абрашкин А.А. Волны Герстнера и их обобщения в гидродинамике и геофизике / А.А. Абрашкин, Е.Н. Пелиновский // Успехи физических наук: методические заметки т. 192, №5 / А.А. Абрашкин, Е.Н. Пелиновский – Нижний Новгород, 2021.
2. Игнатьев Ю.Г. Дифференциальная геометрия кривых и поверхностей в евклидовом пространстве. Учебное пособие. IV семестр. – Казань, 2013, - 204 с.
3. E. Buckingham // On Physically Similar Systems; Illustrations of the Use of Dimensional Equations. Phys. Rev. 4 / E. Buckingham // American Physical Society. – 1914. – С. 345-376
4. Manual — NVIDIA Flex 1.1.0 documentation – URL: https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/flex/manual.html (дата обращения 11.04.2024).
5. J. Tessendorf: Simulating Ocean waters, Simulating nature: realistic and interactive techniques, course 47 – 2001.
6. M. Kass, G. Miller: Rapid, Stable Fluid Dynamics for Computer Graphics, Advanced Technology Group Apple Computer, – 1990г
7. Chapter 1. Effective Water Simulation from Physical | NVIDIA Developer – URL: https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-1-effective-water-simulation-physical-models (дата обращения 11.04.2024)
8. Unity - Manual: Unity User Manual 2022.3 (LTS) – URL: https://docs.unity3d.com/Manual/index.html (дата обращения 14.04.2024)

УДК 004.932.2

РАЗРАБОТКА МОБИЛЬНОГО ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ПОДБОРА ФИЛЬМОВ И СЕРИАЛОВ

Н. С. Крючков, Е. В. Трофименко

Воронежский государственный университет

Введение

Зачастую друзья или партнеры сталкиваются с проблемой невозможности найти чего-то подходящего обоим для просмотра на вечер. Бесконечные просмотры трейлеров, споры и разногласия могут затянуться, а желание что-либо смотреть и вовсе отпадает. В связи с этим было разработано веб-приложение, позволяющее быстро, легко и просто подобрать то, что понравится обоим пользователям.

Основной целью данной статьи является описание приложения, которое позволит аутентифицированным пользователям быстро и просто подбирать фильм или сериал для просмотра по определенным требованиям.

В статье будут рассмотрены различные технологии, которые позволят реализовать веб-приложение, уделив особое внимание библиотекам React JS и Node JS для разработки самого веб-приложения и базе данных PostgreSQL для хранения данных.

1. Архитектура приложения

При разработке приложения были сформулированы следующие требования:

Приложение должно содержать два модуля:

1. «Пользователь».
2. «Поиск совпадений».

Модуль «Пользователь» обладает следующими возможностями:

1. Аутентификация по логину и паролю.
2. Регистрация.
3. История найденных совпадений.
4. Смена пароля.
5. Сохранение истории партнеров, с которыми был поиск

Модуль «Поиск совпадений» обладает следующими возможностями:

1. Создание пары между двумя пользователями с помощью уникального кода.
2. Настройка фильтров для более точного поиска.
3. Поиск совпадений посредством выбора пользователями того, что они хотят посмотреть.
4. Вывод найденного совпадения.

2. Анализ функциональности

На этапе проектирования была выделена одна роль для любого зарегистрировавшегося/авторизовавшегося человека: пользователь. У каждого пользователя изначально имеется доступ к подключению к другому пользователю и возможность поиска

совпадений. Для того, чтобы начать поиск совпадений, пользователям нужно подключиться друг к другу с помощью уникального для каждого пользователя кода. После подключения дается возможность выбрать подходящие фильтры для сужения зоны поиска (рис. 1).

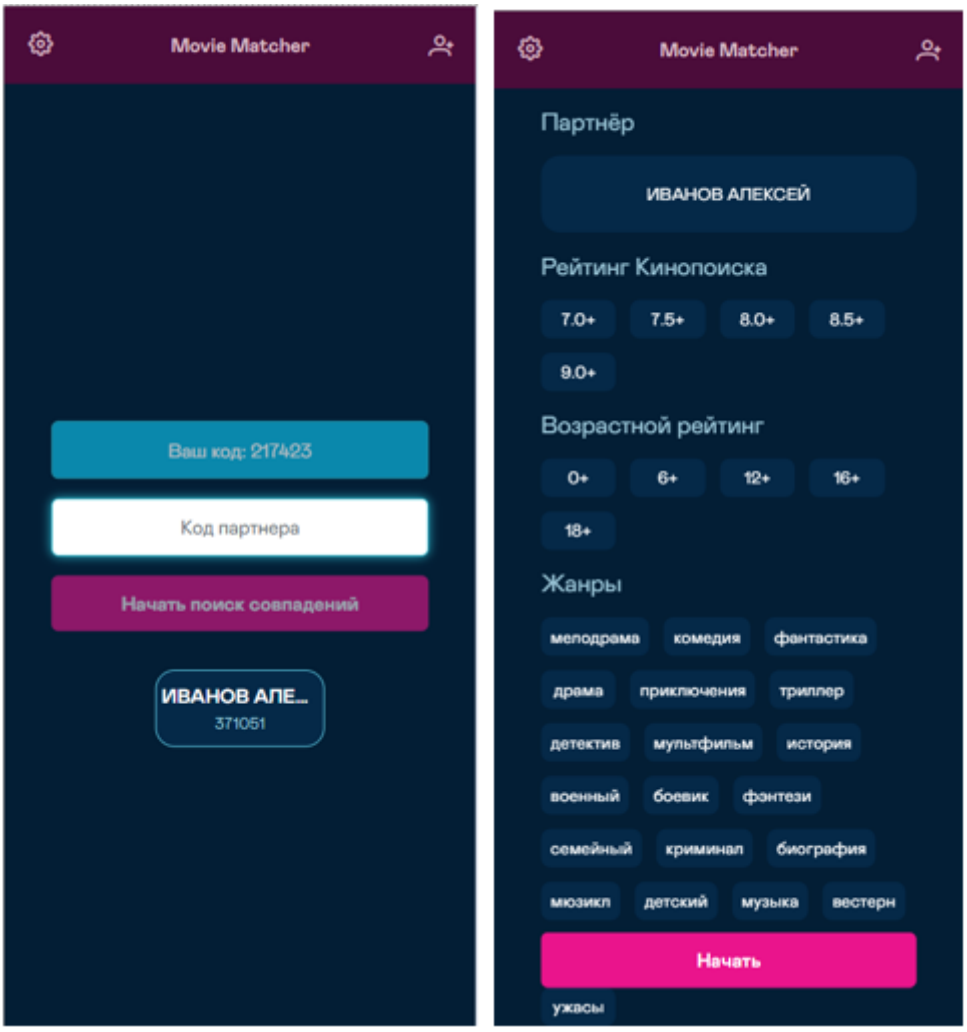


Рис. 1. Подключение пользователей

Далее у пользователей на экране будут появляться фильмы и сериалы из сгенерированной подборки. Пользователь должен делать выбор, хочет он смотреть предложенную картину или нет. На основе выбора «хочу» между двумя пользователями в режиме реального времени идет поиск совпадений. Как только совпадение будет найдено, оно появится на экране (рис. 2).

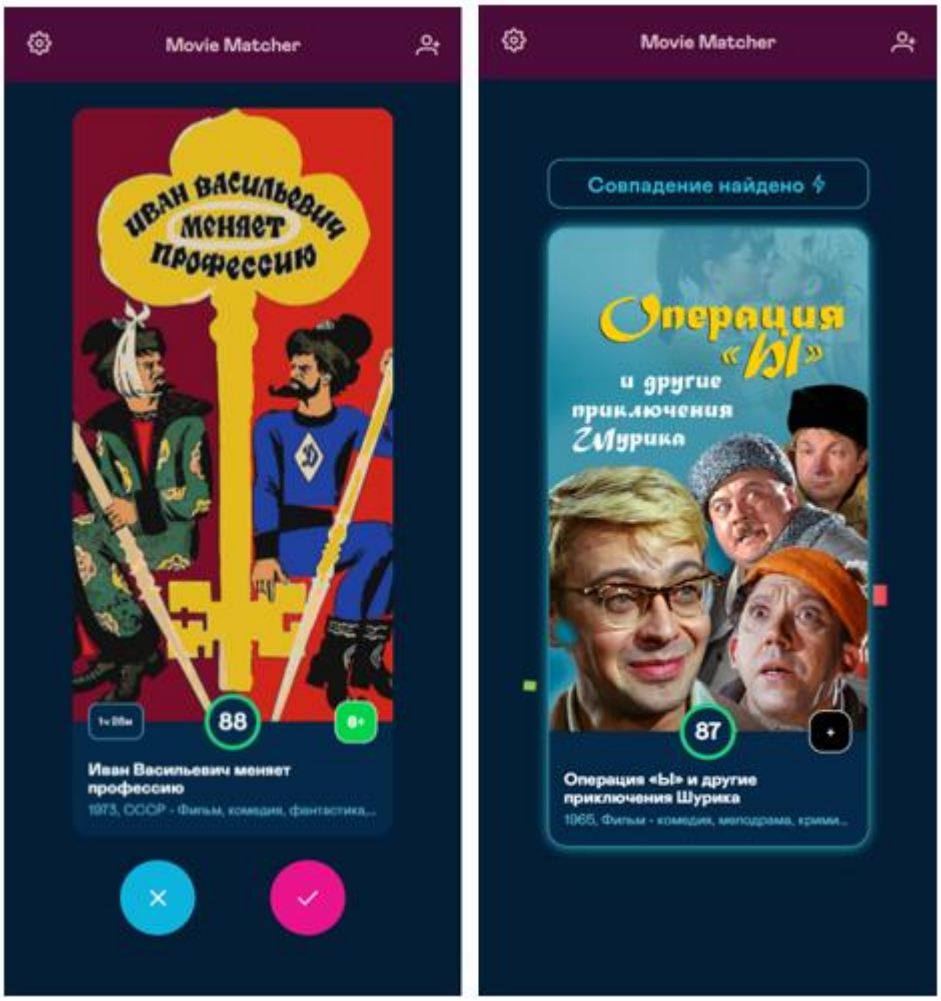


Рис. 2. Поиск совпадений

Затем пользователь сможет увидеть подобранный фильм в своей истории просмотра (рис. 3)

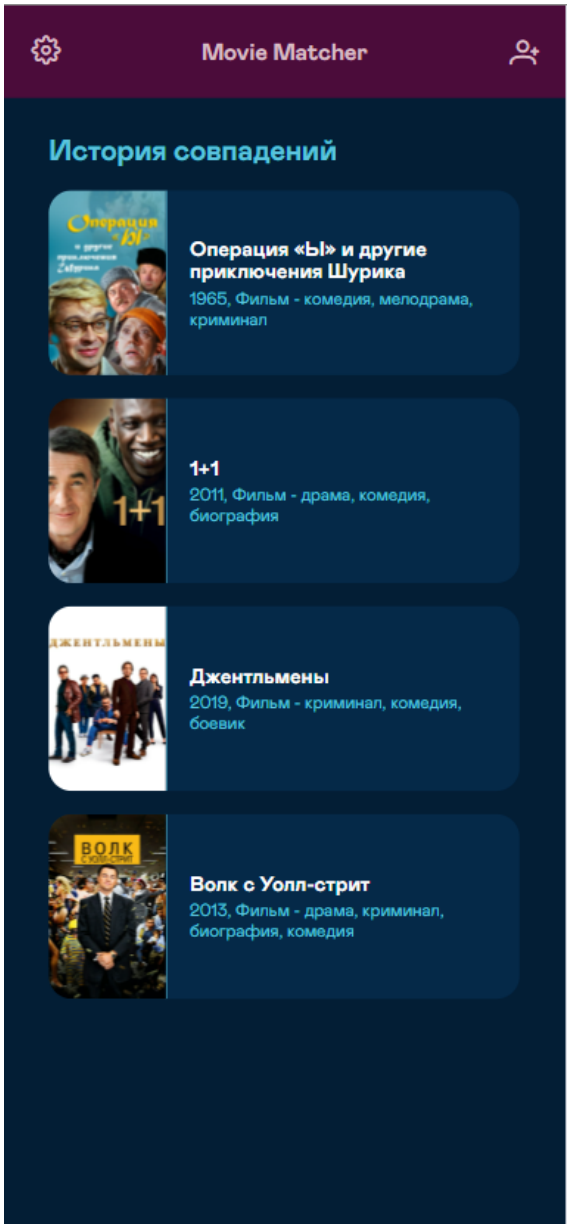


Рис. 3. История просмотра

Веб-приложение разрабатывалось в среде WebStorm. JetBrains WebStorm — интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains, разработанная на основе платформы IntelliJ IDEA. WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями [5].

В качестве языка программирования был выбран JavaScript [6]. Использовалась библиотека React JS для клиентской части, и серверное окружение NodeJS для серверной части. React JS был выбран за счет быстроты его работы, кроссплатформенности, а так же большому количеству библиотек для него [1]. NodeJS был выбран как оптимальный вариант написания небольшой и отказоустойчивой серверной части на языке JavaScript [2]. Для хранения данных была выбрана документированная система управления базами данных – PostgreSQL [4]. Моделями, хранимыми в базе, являются сущности “users” и “movies” (рис. 4).

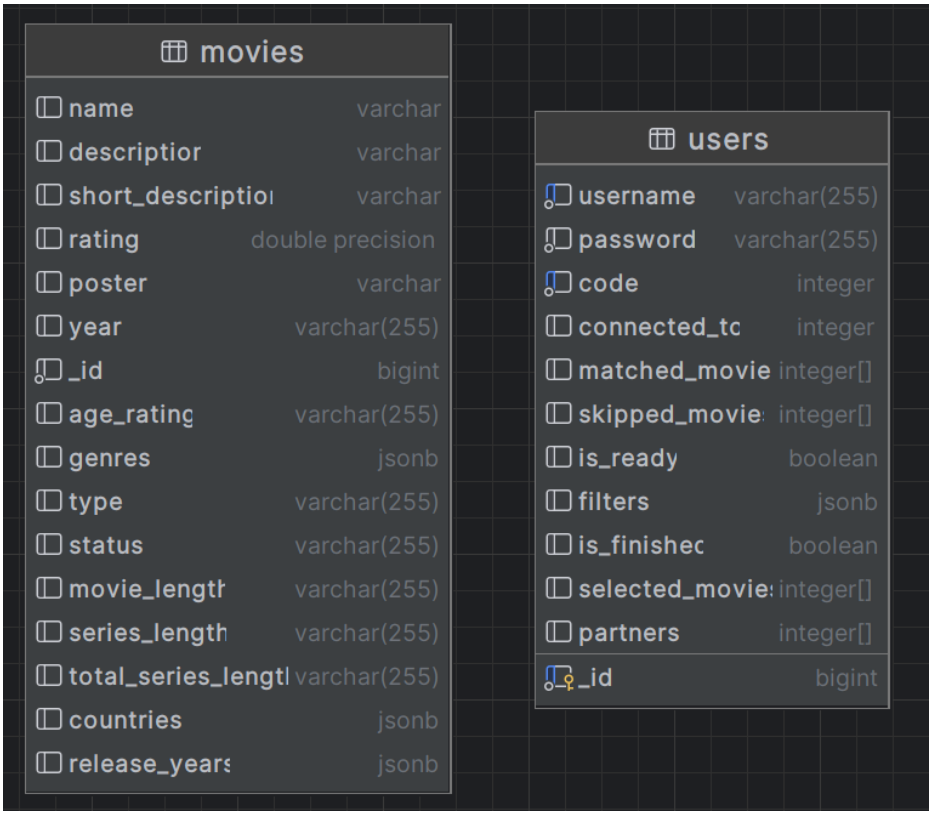


Рис. 4. Схема базы данных

Заключение

В результате проделанной работы было создано веб-приложение, которое отвечает всем требованиям постановки задачи. Полученное веб-приложение рассчитано на широкую аудиторию и позволяет каждому найти, что посмотреть, по интересам.

Литература

1. Тиленс Т.М. React в действии: учеб пособие / Т.М. Тиленс 1-е изд. СПб.: Питер, 2019. 368 с.
2. Node.js в действии: учеб. пособие / К. Симпсон [и др.]; под ред. Н. Райлих 2-е изд. СПб.: Питер, 2018. 432 с.
3. FullStack React, Энтони Аккомаццо, 2020.
4. Postgresqltutorial [Электронный ресурс]. – Режим доступа: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/.
5. Jetbrains [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/idea/
6. Hackreactor [Электронный ресурс]. – Режим доступа https://www.hackreactor.com/blog/what-is-javascript-used-for

УДК 004.8

ВНЕДРЕНИЕ ЯЗЫКОВЫХ МОДЕЛЕЙ RUGPT-3 И RUGPT-3.5 ПРИ РАЗРАБОТКЕ ЧАТ-БОТА

Ю. Д. Кузнецова

Воронежский государственный университет

Аннотация: В статье рассмотрены методы машинного обучения, основанные на внедрении языковых моделей ruGPT-3 и ruGPT-3.5, для задачи генерации текста на поставленный вопрос от пользователя. Для каждой из моделей была передана самостоятельно составленная база данных вопросов-ответов про факультет ПММ. Для обучения и подключения моделей использована библиотека transformers. В исследовании применяются методы вывода текста: Greedy Search, Beam Search, Top-k сэмплирование, Top-p сэмплирование, сэмплирование с температурой. В результате приведены примеры генерации ответа от модели и их оценки.

Ключевые слова: машинное обучение, языковые модели, генерация текста, перплексия.

Введение

В современном мире, где технологии развиваются с невероятной скоростью, чат-боты становятся все более популярными как инструменты для взаимодействия с пользователями. Они используются в различных сферах, от обслуживания клиентов до предоставления персонализированной информации. Одним из ключевых элементов успешной работы чат-бота является его способность понимать и генерировать текст, который звучит естественно и имеет смысл для пользователя. В этом контексте, внедрение ruGPT-3 и ruGPT-3.5 разработанные на основе архитектуры GPT (Generative Pre-trained Transformer), представляет собой значительный шаг вперед, обеспечивающий высококачественное взаимодействие между чат-ботом и его пользователями.

Особенности использования таких языковых моделей в разработке чат-ботов включают в себя их способность к обучению без учителя, что позволяет моделям адаптироваться к специфическим задачам и контекстам использования. Кроме того, эти модели обладают высокой степенью гибкости, позволяя разработчикам настраивать их параметры для более качественной генерации текста.

Цель предполагаемого исследования заключается в анализе ответов чат-бота на часто задаваемые вопросы о факультете ПММ, их уместности, правильности и креативности посредством внедрения языковых моделей ruGPT-3 и ruGPT-3.5. Генерация текста была осуществлена несколькими методами с помощью настройки их параметров. Оценка полученных ответов от моделей оценивалась с помощью метрики переплексии. Работа была реализована на языке программирования Python, потому что имеет доступные и полезные библиотеки для задач машинного обучения. Запуск и обучение моделей были осуществлены в Google Colaboratory (Colab), бесплатном сервисе от Google, который позволяет создавать и запускать Python-код в облачной среде.

1. Исходные данные

Была вручную составлена небольшая база вопросов и ответов на тему поступления на факультет ПММ. Данные формата .xlsx включают в себя информацию о проходных баллах, кафедрах, направлениях, заочной форме обучения, а также формулировки приветствия и непонимания заданного текста от пользователя. Всего 166 строк парных вопросов и ответов. Некоторые вопросы брались с официального сайта факультета. Имеются два столбца в данном датасете: Questions – вопросы, которые могут поступить от пользователя, Answers – ответы, с помощью которых модель будет реагировать на вопрос. На рис. 1 представлены случайные строки полученной таблицы:

	Questions	Answers
157	Когда выдадут пропуск?	В начале сентября.
107	номер пмм	Телефон: +7 (473) 2-208-271
44	функции	Привет! Загляни в главное меню или напиши свой...
76	расскажи про ммио	Кафедра ведет подготовку студентов ПММ по спец...
55	Кафедра мипа	Подготовка специалистов в области механики в В...

Рис. 1 Пример некоторых строк из датасета

2. Общие сведения об языковых моделях GPT-3 и ее русскоязычных версиях ruGPT-3 ruGPT-3.5

Развитие технологий и стремление к получению более качественных систем обработки и генерации текста привели к созданию GPT-3 (Generative Pre-trained Transformer 3), одной из самых мощных и продвинутых моделей искусственного интеллекта, разработанная компанией OpenAI. Она представляет собой нейронную сеть, обученную на огромном объеме текстовых данных, чтобы генерировать человекоподобные тексты в ответ на запросы.

После того как OpenAI предоставила доступ к API-функциональности, предоставляемую другими программами или сервисами — GPT-3 для разработчиков, была создана русскоязычная версия ruGPT-3 от компании Сбер 22 октября 2020 года.

Для создания ruGPT-3 были выполнены следующие шаги:

1. Был взят исходный код GPT-2, внедрены в него идеи из опубликованной научной статьи GPT-3.
2. Для создания ruGPT-3 необходимо было предобучить модель на корпусе текстов размером в 600 ГБ, 90% из которых были на русском языке. В набор данных для обучения были включены русская и английская Википедия, корпус русской литературы, некоторые русскоязычные сайты, а также снимки GitHub и Stack Overflow.
3. После предобучения модели на русском языке, был осуществлен процесс тонкой настройки и дообучения модели на более специфических данных и задачах, чтобы модель могла лучше соответствовать потребностям русскоязычных пользователей.
4. Затем были проведены тесты и оптимизации для улучшения качества генерации текстов на русском языке, а также для обеспечения стабильной работы модели. Модель ruGPT-3 Large, содержит 760 млн параметров.

Версия ruGPT-3.5, продолжая развитие, расширила количество параметров до 13 миллиардов, что делает ее одной из крупнейших моделей для русского языка на данный момент. Модель была обучена в два этапа. Сначала прошло обучение примерно полтора месяца на 300 гигабайтах данных, включающих книги, энциклопедии, научные статьи, социальные ресурсы и другие источники. Затем было проведено дообучение («дотрейн») на 110 гигабайтах данных, включающих затем дополнительно обучена на 100 Гб кода и юридических документов.

2.1 Архитектура GPT-3

GPT-3 использует технологию *трансформеров* — мощную и гибкую архитектуру нейронных сетей для обработки последовательностей данных. Оригинальная архитектура трансформеров включает в себя энкодер и декодер. Кодировщик преобразует векторизованную последовательность с позиционной информацией. Декодировщик расшифровывает ее в виде новой последовательности.

Последующие исследования показали, что можно использовать только один стек блоков трансформера – декодер или энкодер соответственно, увеличивая их количество и обучая модель на больших объемах текстовых данных. Так GPT-3 построена с использованием 96 блоков декодера. Ее принцип работы состоит в генерировании одного токена за раз при применении, основываясь на авторегрессии – каждый созданный токен добавляется к последовательности входных данных для модели.

Для начала работы модели передается стартовый токен, у нее есть только один входной токен, поэтому первая траектория остается единственной активной. Токен успешно проходит через все слои декодера и преобразуется в вектор, который получает коэффициент относительно словаря - всех слов, которые модель знает. GPT-3 способна обрабатывать 2048 токена.

Первоначально находится векторное представление слова в общей матрице эмбеддингов, которая входит в обученную модель. В строках расположены наборы чисел, отражающие слова и улавливающие их часть значений, среди которых находится стартовое слово.

Далее для учета порядка слов в последовательности используется механизм — *positional encoding (позиционное кодировани*е*)*, который позволяет трансформерам «видеть» порядок входных токенов и кодировать позиции слова внутри эмбеддинга. Таким образом, позиция кодируется в виде вектора, который добавляется к эмбеддингу токена.

После произведения поиска эмбеддинга слова и добавление вектора позиционного кодирования для позиции, первый блок обрабатывает токен с помощью механизма внутреннего внимания (Self-Attention). Внутреннее внимание помогает модели понять релевантные слова для того, чтобы оценить контекст для каждого слова, прежде чем приступить к его обработке (прохождению через нейронную сеть).

GPT-3 использует чередующиеся плотные (dense) и разреженные (sparse) слои внутреннего внимания в своей архитектуре. Эти слои играют ключевую роль в обработке входных данных и генерации выходных последовательностей.

В плотных слоях каждый токен в последовательности обрабатывается параллельно, и модель не учитывает контекст других токенов при обработке текущего. Это позволяет модели быстро обрабатывать большие объемы данных, но может привести к потере важного контекста, который мог бы быть захвачен в разреженных слоях.

В разреженных же слоях модель учитывает контекст других токенов при обработке текущего токена. Этот процесс позволяет модели лучше понимать структуру и смысл входных данных, но может быть более медленным из-за необходимости обработки каждого токена последовательно. Так, плотные и разреженные слои дополняют друг друга при их чередовании.

После данной обработки высылается результирующий вектор вверх по стеку для последующих блоков декодера. Так, процесс повторяется в каждом из них.

После того как верхний блок модели выдает выходной вектор, модель умножает его на матрицу эмбеддингов и получается распределение вероятностей для следующего токена. Процесс будет повторяться до тех пор, пока не будет сгенерирован весь контекст или пока не появится токен конца предложения.

GPT-3 сильно превосходит своего предшественника благодаря своей надежной работе и

значительному количеству параметров, содержащих текст разнообразных тематик.

2.2 Обучение модели ruGPT-3 и применение ruGPT-3.5

Для решения задачи генерации текста ответа на непредвиденные вопросы для начала будем использовать языковую модель ruGPT-3, которую необходимо обучить на датасете с информацией о ПММ.

Перед тем, как передать базу данных в модель, будут созданы списки кортежей, где каждый кортеж содержит пару "вопрос-ответ", затем применено форматирование с добавлением меток «Вопрос:» и «Ответ:» и сохранение форматированных данных в .txt файл. Такой способ изменения исходных данных поможет ruGPT-3 при обучении разбивать текстовые данные на отдельные части, улучшая дальнейший вывод.

Для обучения использована библиотека transformers версии 4.24.0. для работы с предобученными моделями трансформеров, благодаря которой в дальнейшем:

- Импортируются два класса: GPT2LMHeadModel для работы с моделью ruGPT-3 (т.к. она в большинстве основана на GPT-2) и GPT2Tokenizer — для токенизации текста. Модель, которую будем загружать - rugpt3small_based_on_gpt2 от Sberbank AI.
- Создается класс TextDataset для создания датасета из текстовых файлов, используя указанный токенизатор и класс DataCollatorForLanguageModeling, подготавливающий даталодер, который будет использоваться для подготовки данных в процессе обучения.
- Используется Trainer для обучения модели и TrainingArguments для настройки параметров обучения.

После создания экземпляра Trainer, вызывается метод train(), который запускает процесс обучения модели с использованием заданных параметров и данных. Использование класса Trainer позволяет упростить процесс обучения модели, автоматизируя многие аспекты, такие как подготовка данных, обучение и сохранение модели.

Так, модель была постепенно обучена всего на 750 эпохах. Часть процесса обучения модели можно наблюдать на рисунке 2:

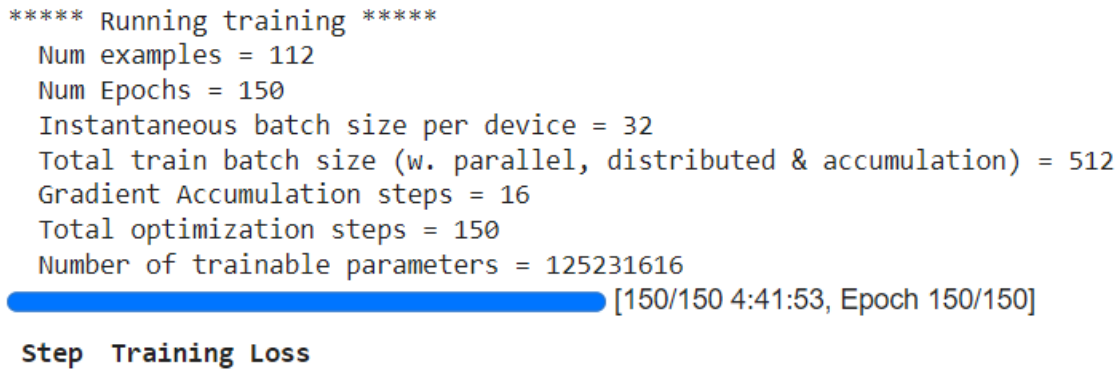


Рис. 2 Процесс обучения модели ruGPT-3

Языковая модель ru-GPT3.5 предоставляет возможность передачи контекста вместо обучения на датасете с парами вопросов и ответов, так как является усовершенствованной версией. Для этого в переменную, содержащую вопрос, будем также добавлять форматированный .txt файл.

3. Методы вывода текста для языковых моделей

После того, как модель получает вероятность следующего токена, происходит процесс

генерации текста, осуществление которого возможно реализацией следующих методов.

Greedy Search — это жадный алгоритмический подход к решению задач оптимизации, который на каждом шаге выбирает наилучший вариант из доступных, не учитывая возможные последствия этого выбора.

На каждом шаге алгоритм выбирает токен, у которого максимальная вероятность, основываясь на текущем состоянии. После выбора наилучшего варианта алгоритм обновляет текущее состояние, применяя выбранный токен. Процесс выбора и обновления повторяется до тех пор, пока не будет достижение локального оптимума. При наличии преимуществ данного метода – быстродействие и простота – существенными недостатками для решения задачи являются ситуации, когда генерация застревает в локальных минимумах, после чего появляются повторяющиеся фрагменты текста, а также когда модель обретает недальновидность. Ведь возможно упущение наиболее вероятных последовательностей, которые были скрыты на предшествующих шагах работы алгоритма.

Beam Search является вариацией предыдущего алгоритма, но вместо выбора одного лучшего варианта на каждом шаге, он сохраняет наиболее вероятные последовательности, называемые лучами, и продолжает поиск в этих направлениях.

Алгоритм beam search начинает работу с первого токена и на каждом шаге увеличивает свой поиск, добавляя к текущим последовательностям следующий наиболее вероятный символ. После этого алгоритм отбирает только те последовательности, которые имеют наивысшую общую вероятность, сохраняя их как возможные результаты. Таким образом, пути генерации разветвляются, получаются несколько вариантов текста. В конечном итоге, выбирается тот вариант, который имеет наилучшую метрику оценки.

Рассмотренные методы имеют недостаток — отсутствие человечности в результирующем выходе модели. *Вероятностное сэмплирование с температурой* решает эту проблему, т.к. имеет параметр, который влияет на распределение вероятностей, назначающиеся каждому возможному следующему слову. Сэмплирование является процессом случайного выбора следующего слова на основе его вероятности встретиться в данной последовательности. На низких значениях параметра модель становится более детерминированной, выбирая наиболее вероятные слова, в то время как на высоких значениях допускается больше свободы для выбора менее вероятных, но более креативных слов. Однако при слишком большом значении температуры токены выбираются наугад, что может привести к некорректным и неправдивым результатам.

Сэмплирование с ограничением маловероятных токенов (Top-k сэмплирование) — это метод, который позволяет контролировать разнообразие генерируемого текста, ограничивая выборку только наиболее вероятными токенами, тем самым улучшая качество результата. Модель сначала выбирает k наиболее вероятных токенов из распределения вероятностей, предсказанного на текущем шаге генерации текста. Затем следующий токен выбирается случайным образом из этих K токенов. Этот процесс повторяется на каждом шаге генерации текста.

В случае *Nucleus Sampling (Top-p сэмплирование)* выбираются только наиболее вероятные токены, пока их суммарная вероятность не превысит значение p. Затем из этого подмножества выбирается следующий токен случайным образом, но только из тех вариантов, которые были выбраны на предыдущем шаге. Таким образом, подмножество слов, из которых выбирается новый токен последовательности, будет динамически меняться на каждом шаге алгоритма.

4. Оценка выводов модели

Рассмотрим примеры генерации текста в таблице 1 на вопрос «как ты думаешь, сложно

ли учиться на ПММ?» после обучения ruGPT-3 для каждого алгоритма, при этом в каждом из них будут изменяться параметры, а дальше оценен выход модели.

Оценка качества генерации текста для языковых моделей является не до конца решенной задачей, ведь решающую роль в определении текста уместным, смысловым отводится человеческому фактору, а не программе. Существуют оценки BLEU, ROUGE или METEOR, однако для них нужна информация эталонного текста. Для оценки данных алгоритмов при изменении параметров было решено использовать метрику перплексию, которая измеряет неопределенность модели при генерации текста. Она может быть вычислена как обратное значение логарифма вероятности следующего слова.

Таблица 1

Оценка генерации текста для ruGPT-3

Метод	**Вывод и оценка**
Greedy Search	Ответ: Сложно, особенно если поступать на бюджет. Но если поступать на бюджет, то можно, если поступать не на бюджет, а на бюджетное место, то можно. Перплексия для Greedy Search равна: 38.223
Beam Search	Ответ: Сложно, особенно если поступать на бюджет. Но если поступать на бюджет, то можно, если поступать не на бюджет Перплексия с числом лучей 1: 7.391 Ответ: Сложно, если не знаешь языка Перплексия с числом лучей 2: 6.668 Ответ: Сложно! Но если регулярно заниматься, не пропускать занятия и учить материал в течение семестра, сдать экзамены становится Перплексия с числом лучей 3: 6.283
Top-k сэмплирование	Ответ: Сложно, но можно. Перплексия с top_k 10: 10.847 Ответ: Сложно, если не знаешь языка Перплексия с top_k 15: 12.464 Ответ: Сложно, из-за ограниченности бюджета. Перплексия с top_k 25: 17.5
Top-p сэмплирование	Ответ: Сложно, но если захотеть, то всё получится. Перплексия с top_p 0.7: 7.515 Ответ: Сложно, но если регулярно заниматься, не пропускать занятия и учить материал в течение семестра, сдать экзамены становится Перплексия с top_p 0.8: 7.403 Ответ: Сложно, если не знаешь английского Перплексия с top_p 0.92: 7.521
Сэмплирование с	Ответ: Иногда возникает ощущение, что ответ тебе не известен, но если постараться, то всё поймёшь.

температурой	Перплексия с температурой 1.2: 8.571 Ответ: Сложно, но я не совсем понимаю твою ситуацию. На ПММ учиться сложно, поэтому там сложно. Перплексия с температурой 1.4: 7.53 Ответ: Обычно обучение проходит в университете на платной основе. Расходы включают: пересылку документов Перплексия с температурой 1.8: 33.313

Greedy Search не имеет параметров, которые можно изменять, поэтому вывод будет один. Вывод получился несвязным, о чем свидетельствует высокое значение перплексии.

Для сэмплирования с температурой были взяты параметры 1.2, 1.4, 1.8. Первые два вывода имеют оценки 8,571 и 7,530. Можно сказать, что значения относительно низкие, для второго случая, вероятно, из-за появления слова «ПММ» в ответе, метрика лучше. Для третьего вывода перплексия очень высока. Это указывает на то, что модель генерирует текст с очень низкой вероятностью, текст не имеет смысла или не соответствует ожидаемому стилю.

Для лучевого поиска взяты параметры лучей 1, 2 и 3. Самый лучший результат показал последний вариант, т.к. его перплексия меньше всего.

Top-k сэмплирование и Top-p сэмплирование показали относительно средний результат.

Для ruGPT-3.5 выполним вывод генерации текста на тот же вопрос: «как ты думаешь, сложно ли учиться на ПММ?» и оценку переплексии. В таблице 2 можно увидеть, что результаты метрики для каждого ответа получаются относительно высокими и равными между собой. Это может говорить о том, например, что модели требуется больше передаваемого контекста.

Таблица 2

Оценка генерации текста для ruGPT-3.5

Метод	**Вывод и оценка**
Greedy Search	Ответ: да, сложно. Перплексия для Greedy Search равна: 16.421
Beam Search	Ответ: Сложно. Перплексия с числом лучей 1: 19.015 Ответ: да, сложно. Перплексия с числом лучей 2: 19.046 Ответ: нет, не сложно. Перплексия с числом лучей 3: 19.015
Top-k сэмплирование	Ответ: я так думаю, что ПММ-овцы - люди очень трудолюбивые и упорные. Они все-таки стараются учиться хорошо :) Перплексия с top_k 10: 16.39 Ответ: очень сложно. Это очень, очень, очень сложно! Перплексия с top_k 15: 16.39 Ответ: нет, не сложно. Если учиться.

	Перплексия с top_k 25: 16.359
Top-p сэмплирование	Ответ: на ПММ учиться сложно. Но, если ты хочешь стать программистом, тебе все равно придется учить математику. А если не хочешь, Перплексия с top_p 0.7: 16.421 Ответ: да, сложно. Особенно в первое время. Перплексия с top_p 0.8: 16.359 Ответ: Не сложно, потому что очень-очень редко задается вопрос с неоднозначным ответом. Перплексия с top_p 0.92: 16.421
Сэмплирование с температурой	Ответ: Мне лично интересно было на ПММ с первого же дня. Думаю, любой, кто на ПММ хоть один день был, учиться хотел Перплексия с температурой 1.2: 16.39 Ответ: думаю, что не очень. Учиться трудно потом. =) Учиться труднее всего где-то с 5ого - 7го раза и после Перплексия с температурой 1.4: 16.421 Ответ: трудно или просто было на ПММ? Вообще ПММ – это, наверное, наиболее сложное из того, что я прошла в этой школе в Перплексия с температурой 1.8: 16.421

Заключение

В ходе данного исследования были описаны исходные данные, рассмотрены модели языковые модели ruGPT-3 и ruGPT-3.5, приведены методы генерации текста, а затем оценены с помощью метрики перплексии.

Как упоминалось ранее, нет точной оценки генерации текста, и сама переплексия может оказаться неточной. Например, при некоторых выводах она имеет высокую оценку, но при этом ответ на вопрос является осмысленным и уместным, если рассматривать его с человеческого фактора. Улучшение качества генерации текста, путем настройки, комбинирования параметров, расширения базы ответов и вопросов, и ее оценки является актуальными задачами, которые можно рассмотреть и решить в будущем.

Литература

1. Саваш Йылдырым Осваиваем архитектуру Transformer / Саваш Йылдырым, Мейсам Асгари-Ченаглу – Москва: ДМК Пресс, 2022. – 320 с.

2. Салып Б. Ю. Декодирующие методы генерации текста в больших языковых моделях / Б.Ю. Салып// Вестник науки и образования. 2023. №10 (141)-2.

3. Ураев Д. А. Классификация и методы создания чат-бот приложений / Д. А. Ураев// Collection of scientific articles. LXIV international correspondence scientific and practical conference (Boston, 20-21 ноября 2019 г.). – Boston: Problems of science – 2019. – С. 30-33.

4. ruGPT-3.5 13B – Режим доступа: https://huggingface.co/ai-forever/ruGPT-3.5-13B. (Дата обращения 12.04.2024)
5. Generation – Режим доступа: https://huggingface.co/docs/transformers/v4.18.0/en/main_classes/text_generation. (Дата обращения 14.04.2024)

УДК 004

ПРИЛОЖЕНИЕ «КАЛЬКУЛЯТОР НАЛОГОВ»

К.В. Лаврентьева

Воронежский государственный университет

Аннотация. Представлены результаты разработки веб-приложения «Калькулятор налогов», функциональность, технологии и преимущества. Приложение предоставляет узкоспециализированные возможности в налоговой сфере.
Ключевые слова: веб-приложение, калькулятор налогов, налоговое законодательство.

Введение

В современном мире, где информация доступна в один клик, особенно важно, чтобы она была структурирована и проста для восприятия. Это касается и такой важной темы, как налоги.

Налоги – это основа экономики любой страны, и знание налогового законодательства необходимо как для граждан, так и для организаций.

Существующие информационные системы, такие как «Консультант Плюс» или «Гарант» достаточно объемны и не рассчитаны только на какой-то выделенный круг задач. Поэтому целью работы является разработка веб-приложения «Калькулятор налогов», которое позволит:

- Узнать о принципах и методах налогообложения в РФ.
- Получить доступ к актуальному законодательству.
- Рассчитать сумму налоговых платежей.

Веб-приложение будет простым в использовании и доступным для всех. Оно станет ценным инструментом для:

- Граждан, которые хотят самостоятельно разобраться в своих налоговых обязательствах.
- Предпринимателей, которым необходимо вести бухгалтерский учет.
- Студентов, изучающих экономику и право.

Актуальность работы заключается в обеспечении доступности законодательства для налогоплательщиков. Для этого необходимо простое в использовании веб-приложения «Калькулятор налогов». Веб-приложение «Калькулятор налогов» сделает информацию о налогах доступной и понятной для всех.

1. Функциональность разработки

«Калькулятор налогов» реализован в виде веб-приложения.

Возможности узкоспециализированы и направлены на деятельность в налоговой сфере, с целью облегчить поиск нужной правовой информации для посетителей сайта.

Пользователи делятся на две категории: зарегистрированные и незарегистрированные. Лица, не имеющие регистрацию, могут воспользоваться калькулятором налогов и получить необходимые расчеты.

Также можно посмотреть актуальные изменения налогового законодательства. В разработанном веб-приложении предусмотрена возможность просмотра необходимой правовой документации.

На рис. 1 представлена модель карты сайта. Сайт состоит из 9 страниц, включая главную страницу. Пользователь может выбрать раздел и получить более подробную информацию.

Рис. 1. Карта сайта

На главном экране расположены кнопки для быстрого перехода на необходимые страницы (рис. 2).

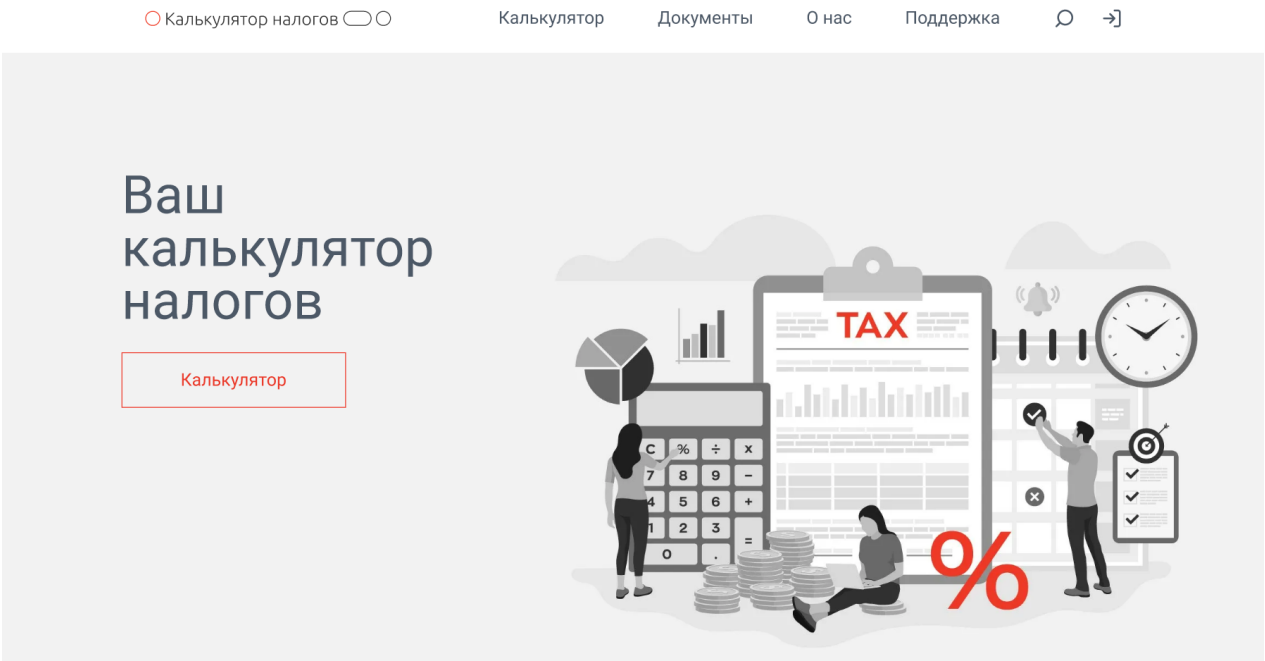


Рис. 2. Главный экран веб-приложения «Калькулятор налогов»

Сам калькулятор представлен в виде окна с выбором необходимых факторов (рис. 3). Для выполнения расчета необходимо нажать на кнопку «Рассчитать».

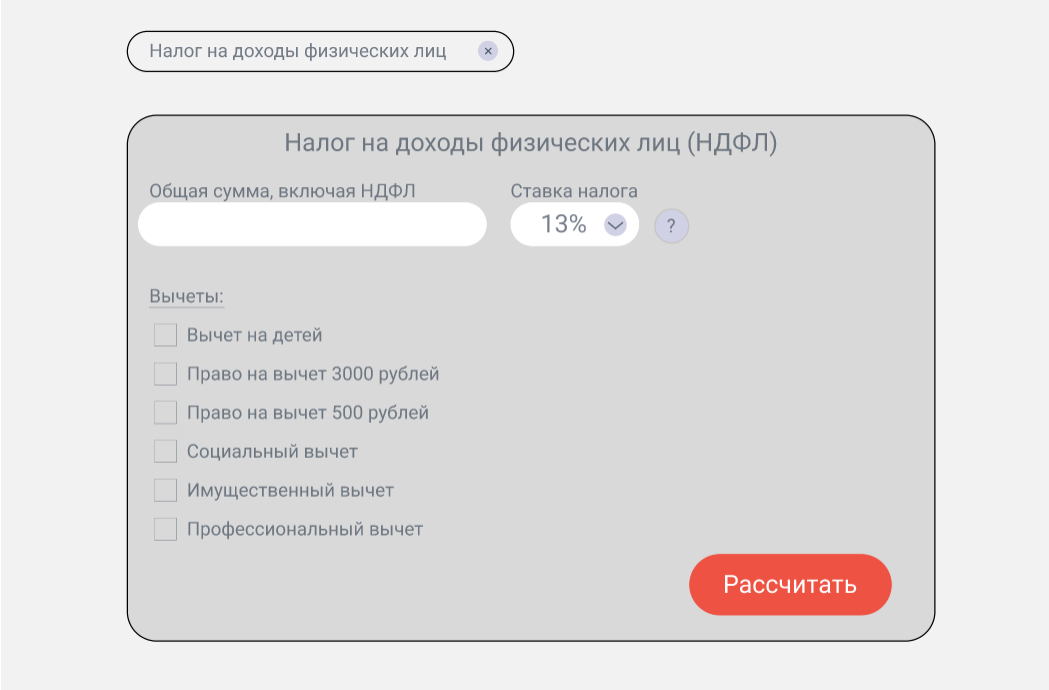


Рис. 3. Пример калькулятора

В основе справочной системы лежит банк данных. В свою очередь, банк данных делится на базы данных, формирующие весь информационный массив. Для организации баз данных используются типовые подходы:

1. Ориентация на информационные потребности пользователей.
2. Группировка информации в соответствии с различными классификационными признаками, например: кодексы, конституционные законы, федеральное и региональное законодательство, официальные и неофициальные документы, отраслевое законодательство, нормы регламенты, стандарты; комментарии специалистов.

Обновление и пополнение информационного банка системы осуществляется на ежедневной основе за счет появления новых документов, а также формирования комментариев, ссылок, ответов на вопросы. Для обновления данных необходимо наличие технических возможностей и соответствующего пользовательского соглашения между клиентом и разработчиком информационной системы.

Заключение

Веб-приложение «Калькулятор налогов» станет помощником в мире налогообложения. Использование приложения позволит:

- Экономить время и деньги.
- Оптимизировать налоговые платежи.
- Избежать ошибок и штрафов.
- Быть в курсе всех изменений в законодательстве.
- Получить профессиональную поддержку.

Литература

1. Конституция Российской Федерации: принята всенародным голосованием 12 декабря 1993 года с изменениями, одобренными в ходе общероссийского голосования 01.07.2020. – Москва : Омега-Л, 2021. - 39 с.
2. Налоговый кодекс Российской Федерации (часть первая) от 31 июля 1998 года N 146-ФЗ [Электронный ресурс] – URL: https://www.consultant.ru/document/cons_doc_LAW_19671/
3. Козырин А. Н. Введение в российское налоговое право: Учебное пособие. — М. : Институт публично-правовых исследований, 2014. — 302 с.

УДК 004.42

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ «ПОМОЩНИК ЧИТАТЕЛЯ»

С. А. Лавров, Т. В. Курченкова

Воронежский государственный университет

Введение

В современном мире многие люди любят читать книги. Чтение позволяет получить новые знания, расширить кругозор и развить воображение.

Книги часто содержат мудрые мысли, которые могут оказаться полезными. Однако, при чтении большого числа книг, бывает трудно сохранять все цитаты в памяти.

В статье рассматривается разработка веб-приложения, позволяющего сохранять информацию о цитатах и делиться ими.

1. Постановка задачи

Разработать веб-приложение по сохранению и публикации цитат из книг. Программа должна содержать следующую функциональность:

- аутентификация пользователей;
- авторизация пользователей на основе ролей (авторизованный пользователь, модератор, администратор);
- операции для создания, получения, изменения, удаления пользователей и их цитат;
- операции для изменения статуса цитат (приватный, в ожидании опубликования, публичный);
- операции для оценки цитат пользователями.

2. Логическая модель данных

При создании веб-приложения была проанализирована предметная область. В результате анализа предметной области была разработана модель данных.

Логическая модель данных в нотации Баркера представлена на рис. 1.

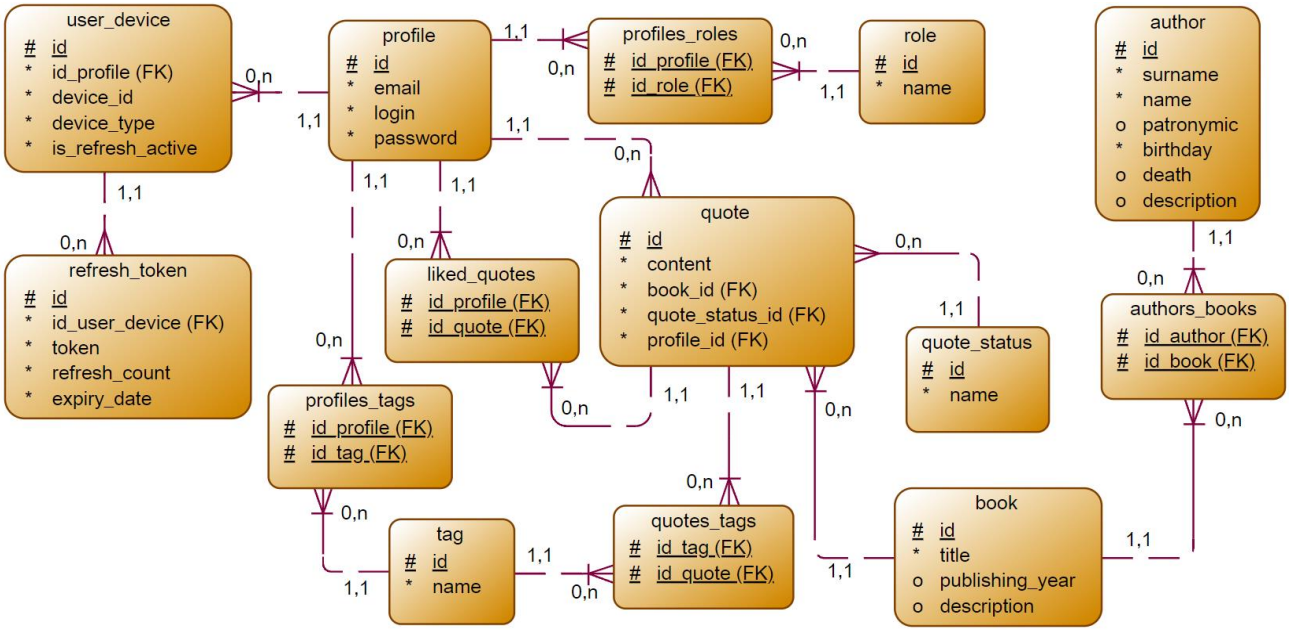


Рис. 1. Логическая модель данных

Сущность *Автор* (author) содержит информацию об авторах: ФИО, дата рождения, дата смерти, описание.

Сущность *Книга* (book) содержит информацию о книгах: название, год издания, описание.

Сущность *Цитата* (quote) содержит информацию о цитатах: содержание.

Сущность *Статус цитаты* (quote_status) содержит информацию о статусе цитат: название.

Сущность *Тег* (tag) содержит информацию о тегах, которые могут быть у цитат: название.

Сущность *Роль* (role) содержит информацию о ролях пользователей: название.

Сущность *Профиль* (profile) содержит информацию о профилях пользователей: адрес электронной почты, имя пользователя, пароль.

Сущность *Устройство пользователя* (user_device) содержит информацию об устройствах, с помощью которых пользователи используют приложение. Данная сущность необходима для того, чтобы выход из аккаунта происходил только на конкретном устройстве, используемом пользователем, а не на всех его устройствах.

Сущность *Токен обновления* (refresh_token) содержит информацию о токенах, которые используются для аутентификации пользователей.

3. Общее описание приложения

Для создания серверной части приложения был использован Java фреймворк Spring Boot [1]. Клиентская часть приложения создана при помощи JavaScript фреймворка Next.js [2].

Внешний вид главной страницы приложения представлен на рис. 2. Структура публичных цитат включает в себя: содержание, название книги, список авторов, список тегов; имя пользователя, опубликовавшего цитату; кнопка *Лайк*, отражающая количество лайков.

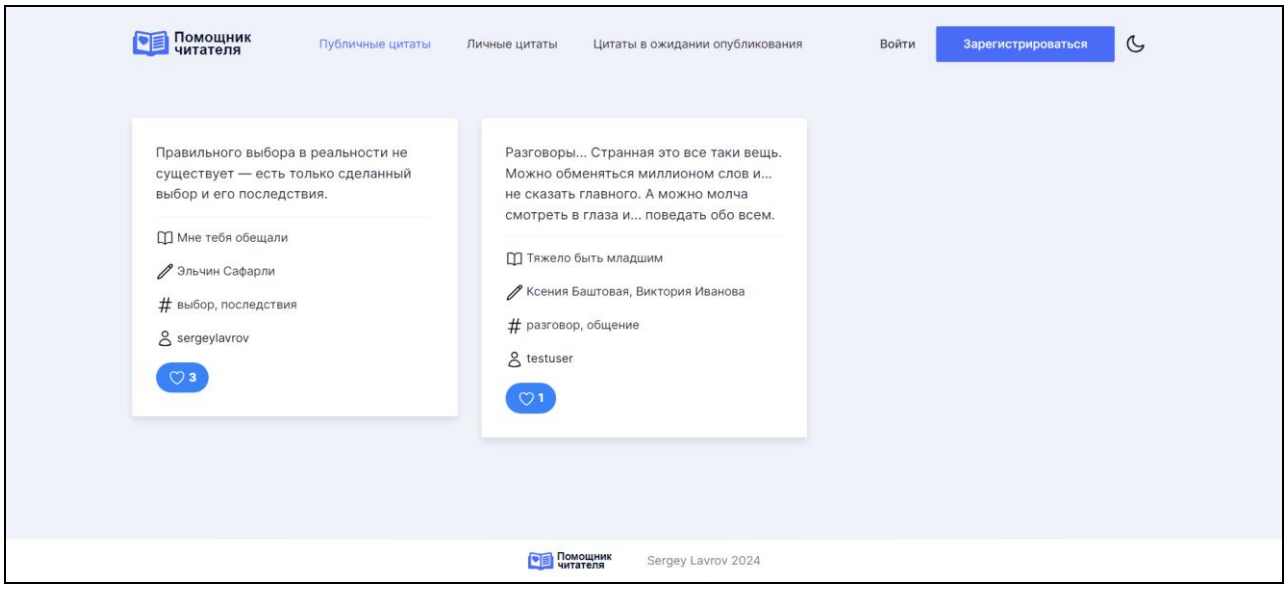


Рис. 2. Главная страница приложения

Приложение имеет возможность включения тёмной темы для комфортного использования, например, в ночное время. Внешний вид главной страницы приложения с включённой тёмной темой представлен на рис. 3. Для создания тёмной темы был использован CSS фреймворк Tailwind CSS [3].

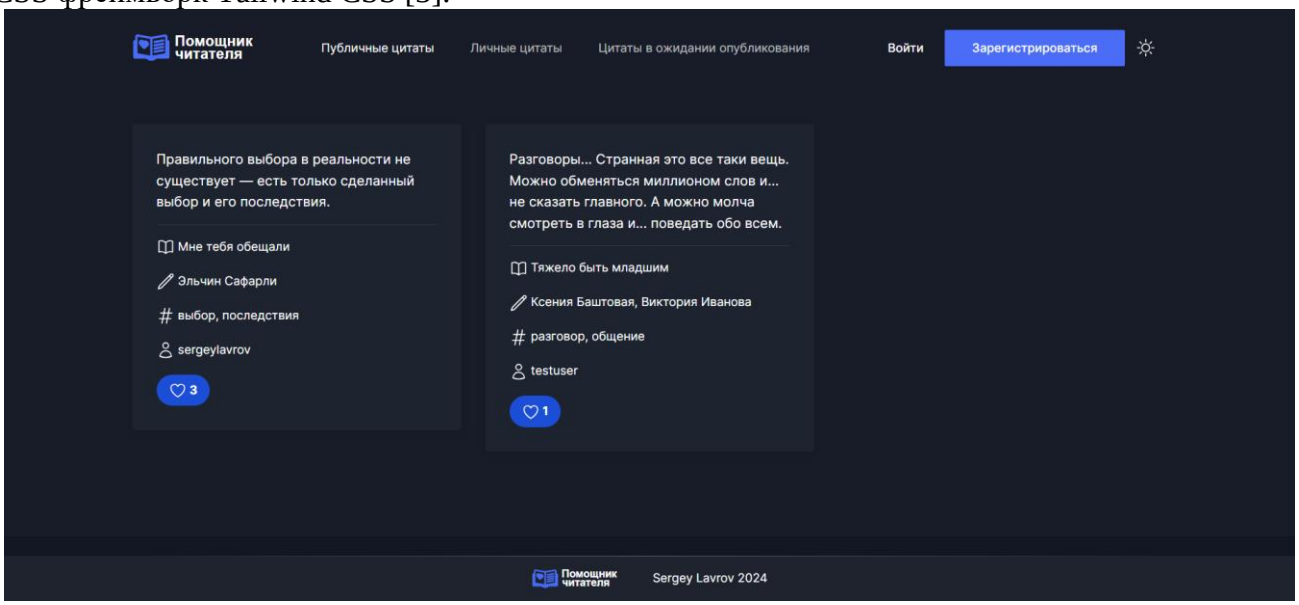


Рис. 3. Тёмная тема

Внешний вид формы регистрации аккаунта и формы входа в аккаунт представлен на рис. 4 и на рис. 5 соответственно.

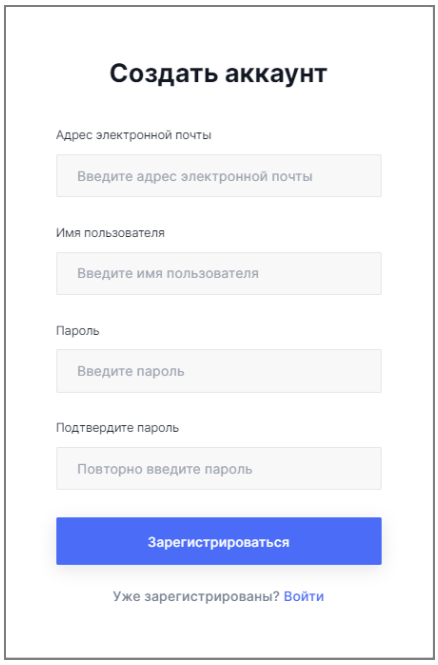


Рис. 4. Форма регистрации аккаунта

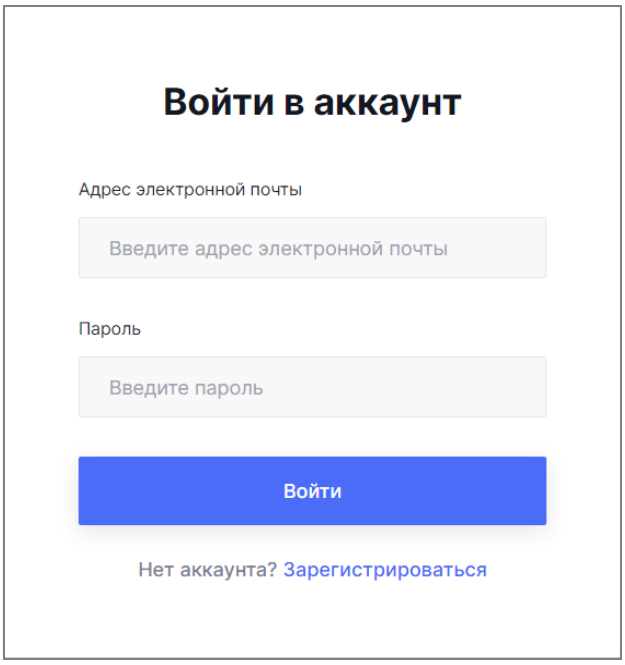


Рис. 5. Форма входа в аккаунт

После входа в аккаунт в правом верхнем углу приложения появляется меню профиля. При наведении на меню профиля открывается подменю, содержащее настройки аккаунта и кнопку выхода из аккаунта. Внешний вид меню профиля при наведении на него представлен на рис. 6.

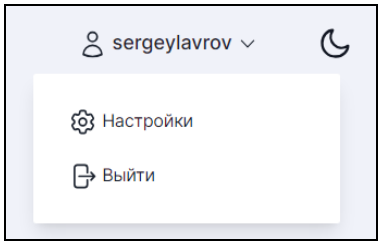


Рис. 6. Меню профиля

При нажатии на кнопку *Настройки* открывается форма настроек аккаунта, внешний вид которой представлен на рис. 7.

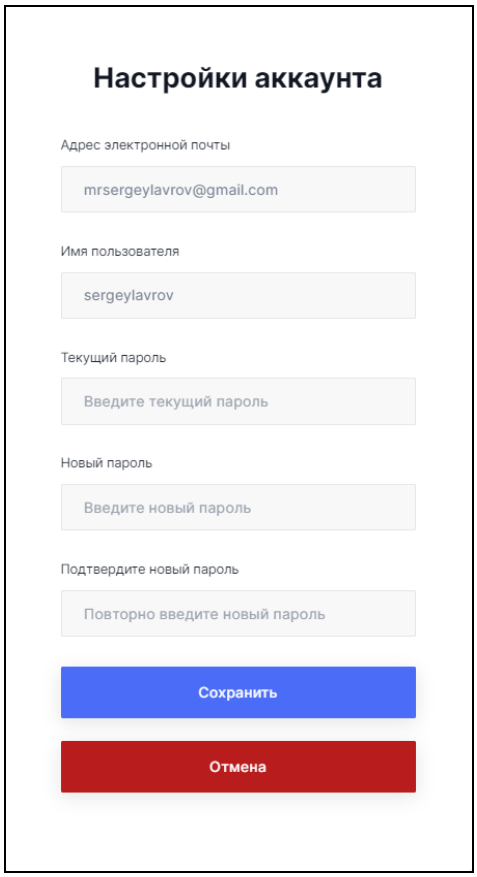


Рис. 7. Форма настроек аккаунта

Внешний вид страницы *Личные цитаты* представлен на рис. 8. В отличие от цитат с публичным статусом, цитаты с приватным статусом не имеют кнопки *Лайк*, но имеют такие кнопки, как *Опубликовать*, *Редактировать* и *Удалить*. Цитаты со статусом *В ожидании опубликования* имеют соответствующую пометку.

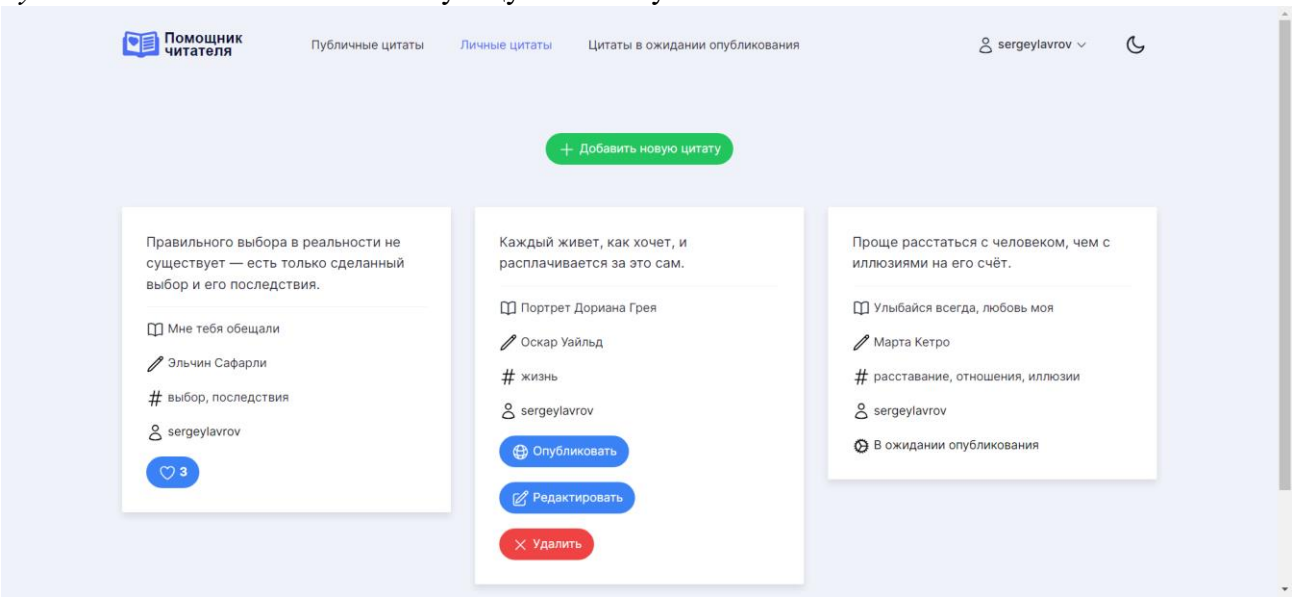


Рис. 8. Личные цитаты

Внешний вид формы добавления цитаты и формы изменения цитаты представлен на рис. 9 и на рис. 10 соответственно.

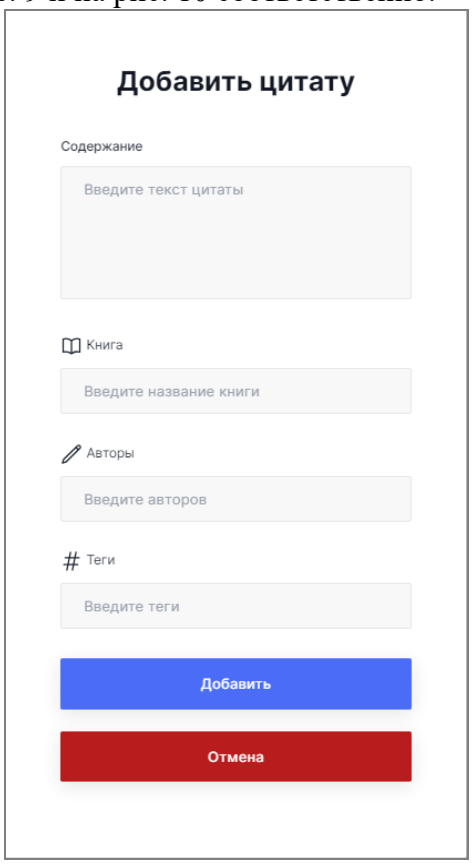


Рис. 9. Форма добавления цитаты

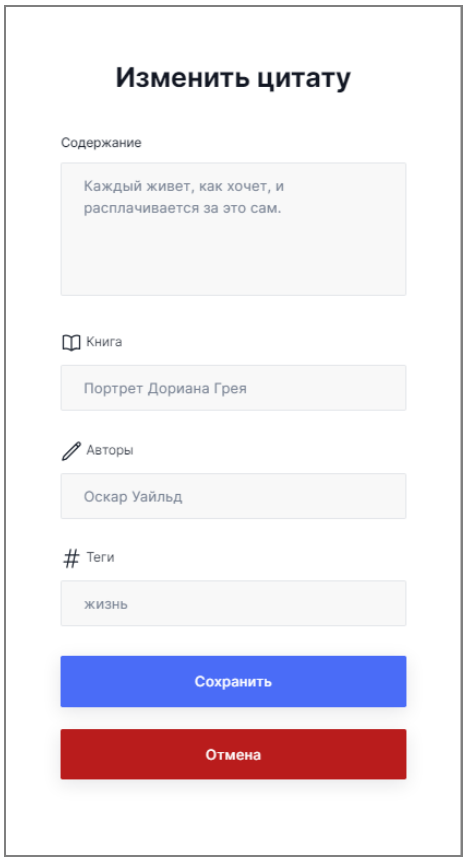


Рис. 10. Форма изменения цитаты

К странице *Цитаты в ожидании опубликования* имеют доступ пользователи с ролями *Модератор* или *Администратор*. Внешний вид цитат на данной странице представлен на рис. 11. Модераторы или администраторы могут опубликовать или отклонить эти цитаты.

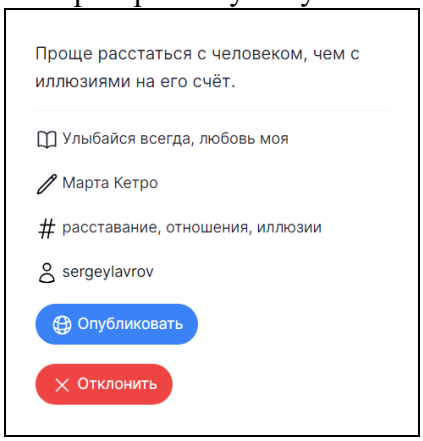


Рис. 11. Цитата в ожидании опубликования

Заключение

Было разработано веб-приложение, позволяющее пользователям сохранять цитаты из литературных произведений и делиться ими. Проставление лайков под цитатами позволяет определить наиболее популярные и делает их более релевантными в приложении. Цитаты,

опубликованные для совместного просмотра, являются достоверными, поскольку проходят проверку модераторами или администраторами. Использование приложения позволяет не забывать цитаты из интересующих произведений, вдохновляться цитатами, опубликованными другими пользователями, а также мысленно возвращаться в запомнившиеся моменты прочитанных произведений.

Литература

1. Spring Boot Reference Documentation. – Режим доступа: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle. – (Дата обращения: 19.04.2024).

2. Next.js Docs. – Режим доступа: https://nextjs.org/docs. – (Дата обращения: 19.04.2024).

3. Tailwind CSS Docs Dark Mode. – Режим доступа: https://tailwindcss.com/docs/dark-mode. – (Дата обращения: 19.04.2024).

УДК 004.912

РАЗРАБОТКА ВЕБ-СЕРВИСОВ С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ И FASTAPI

Д. Ф. Лебединский , К. Г. Резников

Воронежский государственный университет

Введение

Методы машинного и глубокого обучения все чаще применяются в разработке веб-сервисов – это рекомендательные системы в приложениях доставки и киносервисах, различные фильтры для фотографий, использующие методы компьютерного зрения, различные модели для прогнозирования временных рядов, в также все сильнее набирающие популярность чат-боты с нейронными сетями, которые так или иначе умеют работать с текстами на естественном языке.

В связи с этим, изучение базовых технологий и инструментов, которые используются в разработке подобных сервисов, является актуальной темой для многих разработчиков программного обеспечения.

Целью данной научной работы будет написать ML веб-сервис, а также продемонстрировать инструменты, которые могут помочь интегрировать вашу модель машинного обучения в веб сервис для дальнейшего. Данный сервис должен показать, на сегодняшний день сущетсвует множество инструментов, которые позволяют очень быстро внедрить машинное обучение в приложения и показать какие технологии они используют.

1. О модели машинного обучения

В качестве модели машинного обучения была выбрана линейная регрессия, обученная на датасете House Sales in King County, USA (kaggle.com). Датасет содержит признаки и стоимость жилья в округе Кинг, штат Вашингтон, США. При анализе данных были сконструированы новые признаки, и в конечном итоге признаки выглядят так (рис. 1):

```
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   bedrooms                    21613 non-null  int64
 1   bathrooms                   21613 non-null  float64
 2   sqft_living                 21613 non-null  int64
 3   sqft_lot                    21613 non-null  int64
 4   floors                      21613 non-null  float64
 5   waterfront                  21613 non-null  int64
 6   view                        21613 non-null  int64
 7   condition                   21613 non-null  int64
 8   grade                       21613 non-null  int64
 9   sqft_above                  21613 non-null  int64
 10  sqft_basement               21613 non-null  int64
 11  yr_built                    21613 non-null  int64
 12  yr_renovated                21613 non-null  int64
 13  zipcode                     21613 non-null  int64
 14  lat                         21613 non-null  float64
 15  long                        21613 non-null  float64
 16  sqft_living15               21613 non-null  int64
 17  sqft_lot15                  21613 non-null  int64
 18  square                      21613 non-null  int64
 19  cent                        21613 non-null  int64
 20  distance_to_seattle_center  21613 non-null  float64
 21  baths_res_bedr              21613 non-null  float64
 22  avg_neighbor_area           21613 non-null  float64
 23  season_spring               21613 non-null  uint8
 24  season_summer               21613 non-null  uint8
 25  season_winter               21613 non-null  uint8
dtypes: float64(7), int64(16), uint8(3)
memory usage: 4.0 MB
```

Рис. 1 – сводная информация по данным.

Общие метрики качества можно назвать приемлемыми, они получились следующими (рис. 2):

```
RMSE: 198750.38422811573
R2: 0.7387050251251113
```

Рис. 2 – метрики качества предсказания модели.

График остатков (рис. 3):

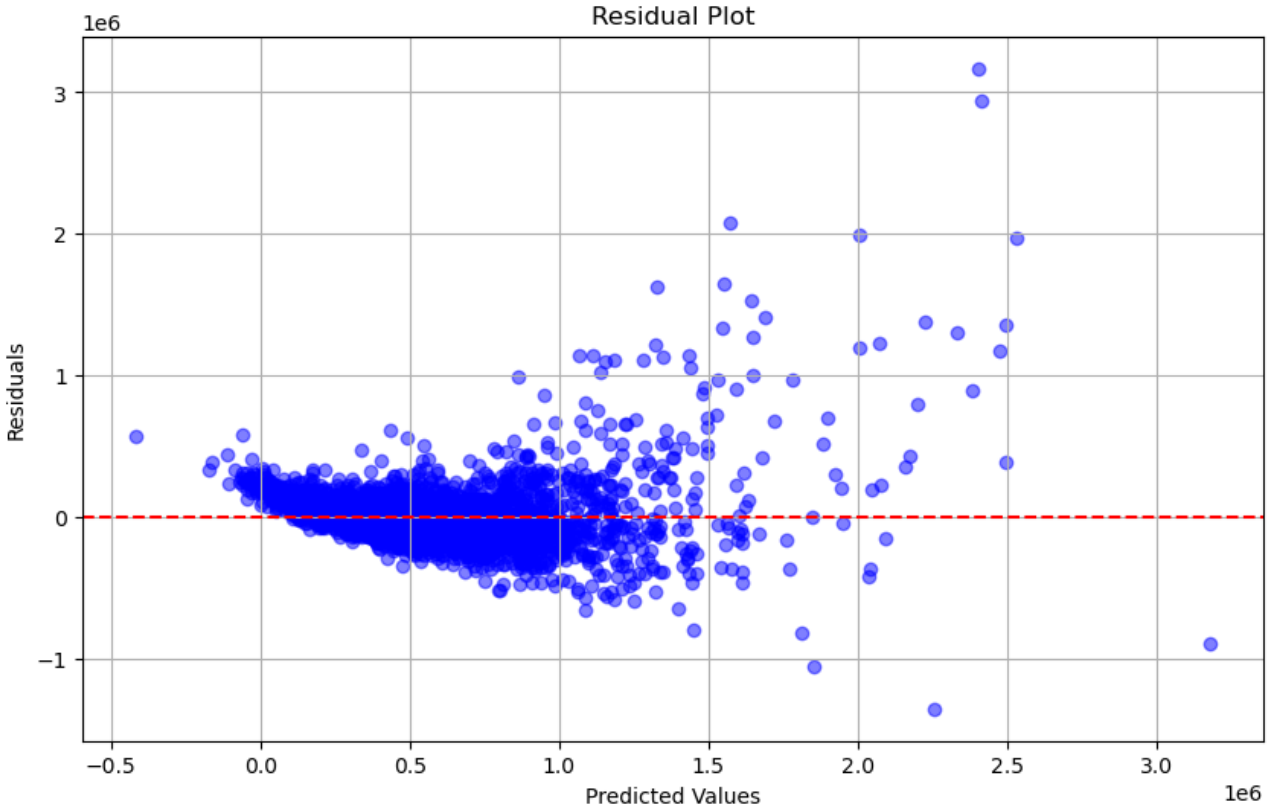


Рис. 3 – график остатков.

Загружать модель можно с помощью Joblib – это библиотека для сериализации объектов в файлы и обратной выгрузки в память. Эта библиотека предоставляет простой интерфейс для загрузки и восстановления таких сложных объектов как Numpy массивы, модели машинного обучения sklearn, pandas DataFrame и тому подобные объекты. Joblib поддерживает параллельную обработку данных для загрузки и сохрания в обход GIL, что обеспечивает высокую производительность даже при работе с большими моделями. Функция **dump** нужна для сохранения модели в файл, **load** – для загрузки из файла обратно в объект.

2. Построение веб-приложения с помощью FastAPI

FastAPI – это современный, мощный фреймворк для написания асинхронных веб-API на языке Python. Асинхронность – одно из главных преимуществ фреймворка, он хорошо интегрируется с такими библиотеками как aiohttp, а его синтаксис тесно связан с синтаксисом нативных асинхронных корутин в Python (**async/await** синтаксис) и с встроенным модулем asyncio.

FastAPI построен на основе Pydantic – это мощный и высокопроизводительный инструмент для валидации данных (к примеру, приходящих в API со стороны клиента) и их сериализации.

Для запуска FastAPI требуется сервер, и для этого отлично подходит Uvicorn – веб-сервер ASGI. ASGI можно рассматривать как связующее звено, он позволяет асинхронным Python серверам и приложениям взаимодействовать друг с другом. Он повторяет множество

решений из WSGI, и обычно представляется как его преемник с поддержкой асинхронности асинхронностью. Его можно изобразить с помощью следующей диаграммы (рис. 4):

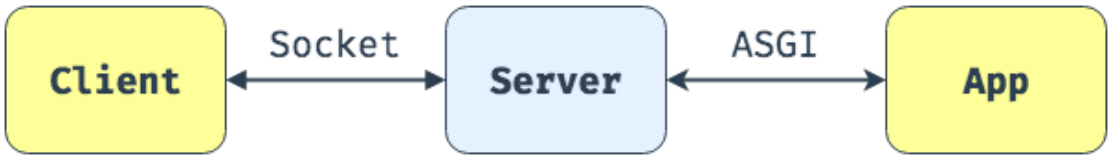


Рис. 4 – упрощенная диаграмма структуры ASGI сервера.

ASGI состоит из двух компонентов:

1) Сервер протокола – слушает сокеты на наличие сообщений и создает соединение при необходимости.

2) Приложение – существует внутри сервера протокола, для каждого соединения создается свой экземпляр приложения, который обрабатывает приходящие сообщения.

Более детализированная диаграмма (рис . 5):

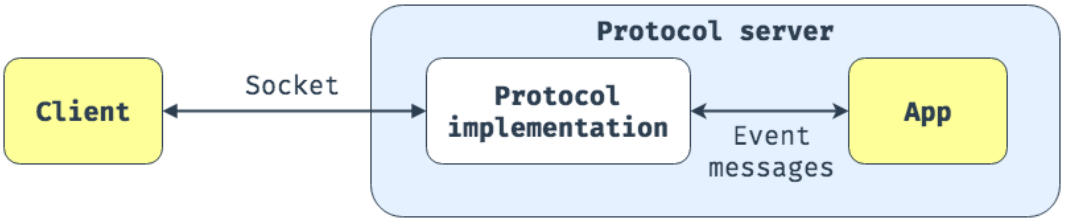


Рис. 5 – общая диаграмма структуры ASGI сервера.

Однако, стоит отметить, асинхронность не всегда может увеличить производительность ML-сервисов – это связано с тем, что обработка данных моделью машинного обучения относятся к задачам CPU bound, в то время как асинхронность может увеличить производительность во время ожидания при задачах ввода/вывода.

3. Реализация.

Загрузим модель в директорию, находящуюся внутри проекта приложения. Для реализации нам понадобится одна конечная точка – **/predict**. Это будет HTTP метод POST – данные будут приходить со стороны клиента в файле json, сериализация – при помощи упомянутого ранее Pydantic. Для этого опишем класс HouseFeatures, который является наследником класса BaseModel.

После корректного описания метода **predict**, можем запустить и протестировать приложение.

Так выглядит интерфейс нашего метода predict в swaggerUI (рис. 6):

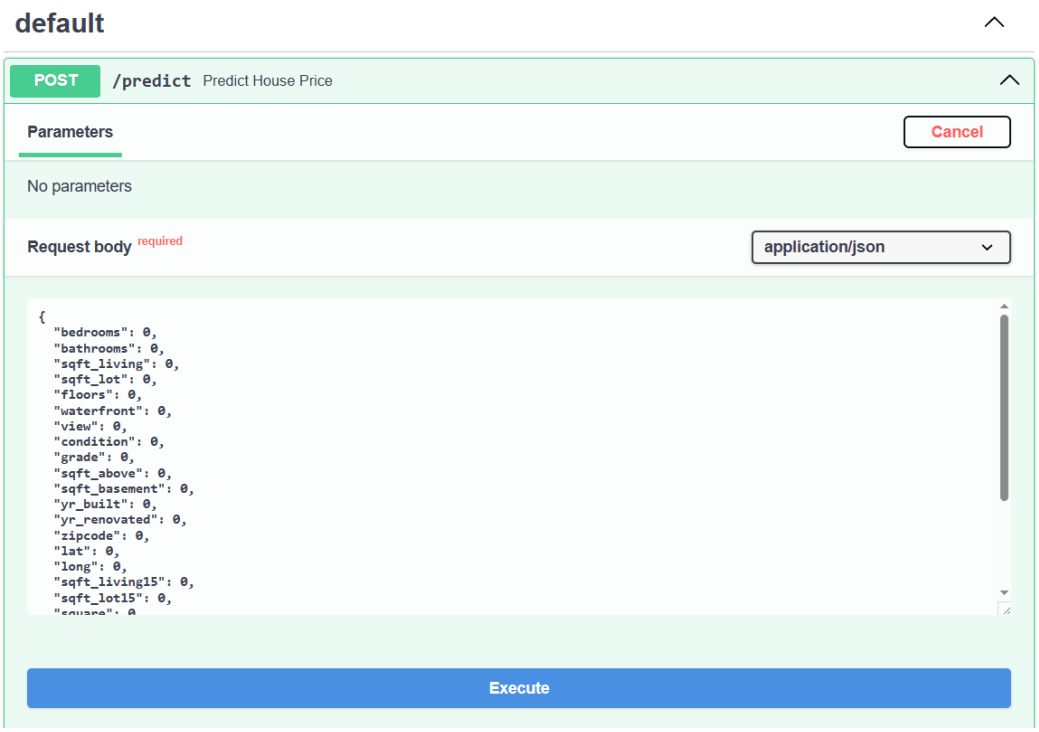


Рис. 6 – интерфейс метода predict в swagger.

Передадим в метод json файл с признаками заготовленного дома. Метод выполнился с ответом 200, что говорит о том, что критических проблем не возникло.

Ответ модели, на которую поступил POST-запрос с данными признаков в формате json, при реальном значении – 325000 $ (рис. 7):

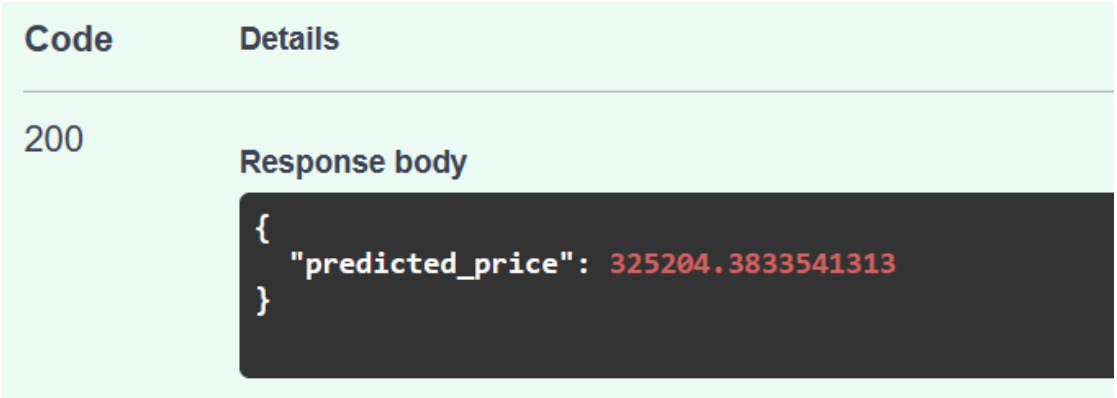


Рис. 7 – ответ сервиса, вычисленный интегрированной в нее моделью машинного обучения.

Заключение

При написания работы были рассмотрены многие инструменты на языке Python, которые могут помочь нам создавать небольшие веб-сервисы, основанные на алгоритмах машинного обучения. Был пройден значительный путь разработки подобного сервиса – от создания модели машинного обучения до выбора инстументов загрузки модели, сериализации данных и написания веб-API приложения. В данной работе были продемонстрированы основные инструменты и подходы, которые используются при итегрировании машинного обучения в веб-сервисы, а также подходы, которые не имеет смысла использовать в таких приложениях (например асинхронное программирование).

Список использованных источников

1. Харрисон М. Машинное обучение. Карманный справочник. — Диалектика, 2020 г., 320 с.
2. Интернет-ресурс: `FastAPI (tiangolo.com).`
3. Интернет-ресурс: https://joblib.readthedocs.io/en/stable/

УДК 658.711

МОДЕЛИРОВАНИЕ УЧЕТНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ТЕРМИНАЛЬНО-СКЛАДСКИМ КОМПЛЕКСОМ НА БАЗЕ 1С:ПРЕДПРИЯТИЕ 8.3

М. С. Левченко , И. Н. Булгакова

Воронежский государственный университет

Введение

Рост популярности контейнерных перевозок, увеличение спроса на поставки контейнерных грузов, увеличение мультимодальных перевозок требуют новых подходов к организации работы терминально-складских комплексов, ориентированных на широкое применение современных методов управления. Целью исследования является разработка управленческой системы контейнерно-терминального комплекса, ориентация которого идёт на учет всех затрат, которые задействованы в отправке и загрузке груза, на их грамотное распределение в зависимости от выбора временных рамок, а также учет запасов при частичной загрузке. Реализация задач исследования достигнута на основе инструментов программного продукта «1С. Предприятие 8.3», а именно не создание дополнительных модификаций для существующих решений, а создание новой конфигурации, что позволит задать решению свою структуру.

1.1. Описание проблемной области

Терминально-складские комплексы являются, по сути, элементами контейнерной транспортной системы и размещаются в пунктах взаимодействия разных видов транспорта. Выделяют следующие виды контейнерных терминалов: грузовые, грузосортировочные и сортировочные терминалы.

В своем исследовании мы будем опираться на анализ процессов, происходящих на грузосортировочных терминалах, которые выполняют все функции грузовых пунктов, а также осуществляют сортировку транзитных контейнеров, т.е. предоставление складских услуги по перегрузки с одного на другое транспортное средство.

Контейнер загружается на транспортное средство с помощью вилочного погрузчика, бокового погрузчика, ричтрака, контейнерного домкрата или другого оборудования. Развитие системы грузоперевозок с использованием стандартных контейнеров привело к возникновению новой формы транспортного обслуживания - предложение контейнеров, ролл-трейлеров, автотрейлеров, автотягачей, погрузчиков в аренду как грузовладельцам, экспедиторам, так и перевозчикам. Поэтому при анализе издержек, связанных с работой терминально-складского комплекса, необходимо учитывать все возможные варианты.

Используемое транспортно-перегрузочное и складское оборудование напрямую влияет на основные технико-экономические показатели терминально-складского комплекса: наличие и использование складских площадей, время переработки одного контейнера, суточное количество перерабатываемых контейнеров, количество единиц техники, количество персонала и т.д. [2].

Функционирование терминально-складского комплекса рассматривается на конкретном горизонте планирования, который определяется как период, в течении которого определен спрос на формирование контейнеров с определенным видом продукции. Предлагаемый в [1,2] подход к разработке стратегии загрузки и отправки контейнеров предполагает учет всех затрат,

которые возникают при обработке, загрузке и отправке груза, их грамотное распределение по этапам работ.

К издержкам, возникающим при управлении терминально-складского комплекса относят:

на хранение одного контейнера в рассматриваемый период;

на эксплуатацию погрузочно-разгрузочного оборудования в рассматриваемом периоде с учётом амортизационных отчислений;

на аренду погрузочно-разгрузочного оборудования;

на автоперевозки собственным транспортом;

на аренду автотрейлера;

на проведение визуального осмотра груженого/порожнего контейнера в вагоне/автомобиле, его внешнего состояния, наличия исправного запорно-пломбировочного устройства и соответствие номера, указанного в перевозочном документе;

на букировку;

на поддержание температурного режима рефрижераторных контейнеров;

на очистку, промывку, ветеринарно-санитарную обработку и дезинфекциу вагонов/контейнеров после выгрузки грузов на дезинфекционно-промывочных пунктах и станциях;

на оформление и выдачу сертификата VGM (верификация массы груженых контейнеров).

на транспортно-экспедиционное и стивидорное обслуживание,

на перемещение груженого контейнера в пределах терминала для осуществления отбора проб, таможенного или иного досмотра;

на перемещение груженого контейнера в пределах терминала для осуществления досмотра с возможной выгрузкой и сортировкой по наименованиям;

на перемещение груженого контейнера в пределах терминала для определения фактической массы;

на перегруз по варианту контейнер-склад или в обратном направлении.

На российском рынке существует ряд специализированных компаний, занимающихся разработкой профессионального программного обеспечения для контейнерных терминалов. Так, БизАрт Групп [5] разработано решение «Контейнерное экспедирование и логистика» на базе 1С: Предприятие, предназначенное для автоматизации учета транспортно-экспедиторских предприятий, которые занимаются контейнерными перевозками любыми видами транспорта. Компания SeaData [6] представила операционную систему управления контейнерным и грузовым терминалом, тарификацию услуг и контроль оплат. Управление контейнерным и портовым терминалом — программное решение, разработанное AXELOT [4] для автоматизации управления всеми операциями с грузами, транспортом и техникой на территории портовых и контейнерных терминалов. И, наконец, компания «СОЛВО» [7] является одним из лидирующих российских разработчиков программного обеспечения SCE (Supply chain execution или исполнение цепей поставок в режиме реального времени): системы класса WMS (Warehouse Management System) и TOS (Terminal Operating System) для автоматизации процессов на складах, контейнерных, балкерных и прочих портовых терминалах, грузовых дворах железнодорожных станциях и др. Но в рамках приведенных программных решений возможно построение только стоимостной модели, в то время как существует необходимость совместить ее с процедурой управления обработки контейнеров [3].

1.2 Моделирование учетной системы управления терминально-складским комплексом

На основании динамического моделирование системы управления терминально-складским комплексом, представленном в [1], предлагается программное решение для внутреннего учета загруженных и отправленных контейнеров, себестоимости операций, связанных с загрузкой и отправкой, координации временных параметров работы терминала, регулировании видов статей затрат на платформе «1С:Предприятие 8.3». В качестве результата работы представлена конфигурация «Терминально-складской комплекс».

Интерфейс данной конфигурации (рис.1) представлен в виде раздела «Администрирование», в котором содержатся основные документы, справочники и отчеты.

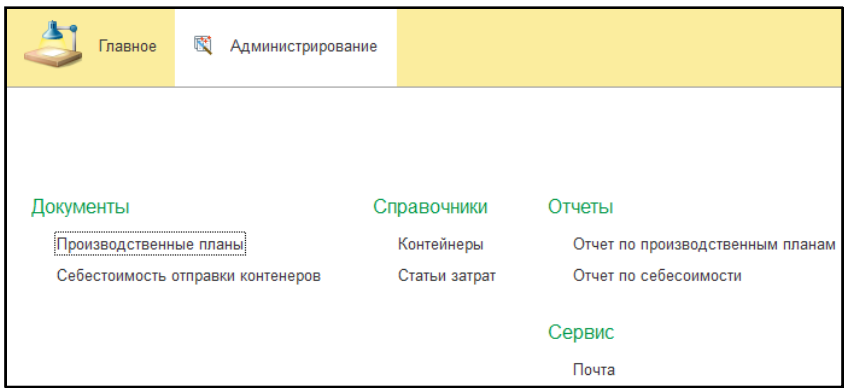


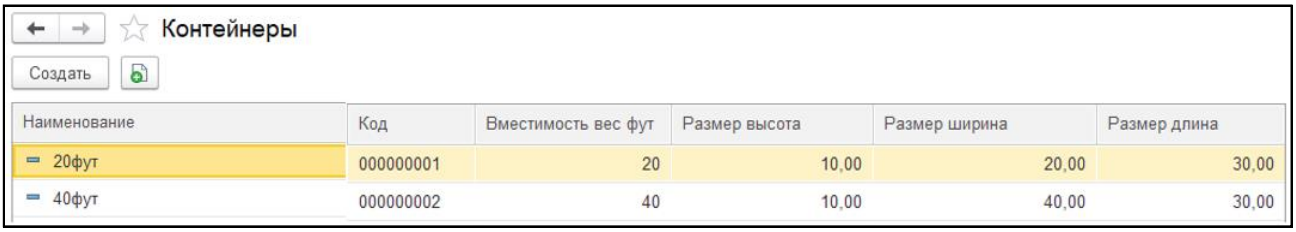
Рис. 1. Программный интерфейс

В программе содержатся два справочника: «Контейнеры» и «Статьи затрат».

В соответствии со стандартом ISO 668-1995 (E) различают 4 размерные группы контейнеров:

1. 1CC, 1C, 1CX — длина 20 футов, 6096 мм.
2. 1AAA, 1AA, 1A, 1AX — 40 футов, 12192 мм.
3. 1D, 1DX — 10 футов, 3048 мм.
4. 1BBB, 1BB, 1B, 1BX — 3 фута, 914 мм.

В качестве примера будем рассматривать первые два типа контейнеров. Вид заполненного справочника «Контейнеры» представлен на рисунке 2:



Наименование	Код	Вместимость вес фут	Размер высота	Размер ширина	Размер длина
20фут	000000001	20	10,00	20,00	30,00
40фут	000000002	40	10,00	40,00	30,00

Рис.2. Справочник «Контейнеры»

Справочник «Контейнеры» имеет реквизиты: «Наименование», «Код», «Вместимость вес фут», «Размер высота», «Размер ширина» и «Размер длина». Высота и длина у двух контейнеров совпадает, тогда как ширина отличается и соответствует вместимости.

Справочник «Статьи затрат» (рис.3) содержит реквизиты «Наименование» и «Код». Информация по затратам, занесенная в справочник, будет в дальнейшем использована при формировании учетной цены поставщика и фактической себестоимости операций терминально-складского комплекса:

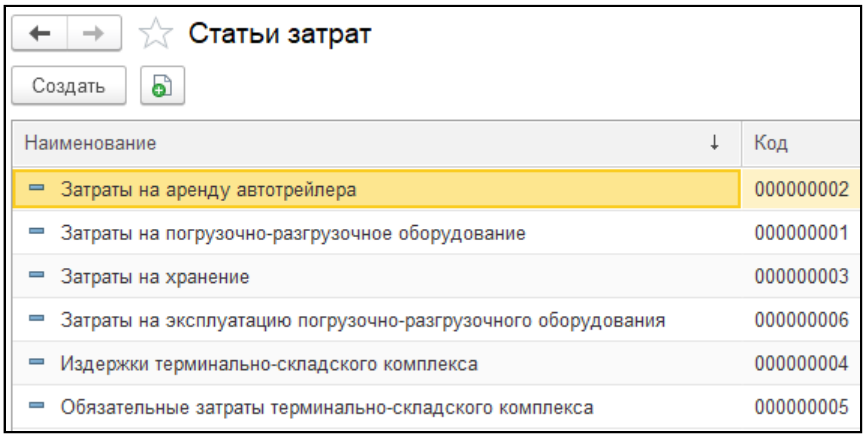


Рис. 3. Справочник «Статьи затрат»

Следующий этапом после заполнения пользователем справочников, необходимо сформировать первый документ «Себестоимость отправки контейнеров». Для этого необходимо знать информацию о плановом периоде отправки (раннюю и позднюю даты), а также периоды оперативного управления. Например, если считать периодом плановой отправки календарный месяц, то за период оперативного управления возможно принять неделю. Тогда для расчета себестоимости отправки контейнеров необходимо задать ровно четыре документа с датами каждой недели, следующая неделя начнется после последней даты предыдущей. Отличие данных документов друг от друга будет также состоять в выборе типа контейнера. Отчетливо эта связь будет прослеживаться при составлении документа «Производственный план». Рассмотрим пример, в котором зададим производственный план и для двух типов контейнеров - 20- футовых и 40- футовых, поэтому на плановый период в один месяц с оперативным периодом одна неделя предполагается сформировать восемь документов «Себестоимость отправки контейнеров», по четыре на каждый вид контейнера.

Для формирования документа необходимо заполнить четыре вкладки в табличной части: «Расшифровка затрат отправка», «Расшифровка затрат загрузка», «Расшифровка затрат отправка подрядчик», «Расшифровка затрат загрузка подрядчик» (рис.4). Стоит отметить, что документ «Себестоимость отправки контейнеров» включает в себя одновременно два типа издержек – связанных с загрузкой и с отправкой. Период планирования предполагается с 01.03.2024 по 28.03.2024. Затраты на каждую неделю в рассматриваемом примере будем считать одинаковыми. Данные, заполняемые в табличной части документа, суммируется в итоговых полях, каждое из которых соответствует вкладке табличной части. При этом «Затраты на погрузочно-разгрузочное оборудование», «Затраты на эксплуатацию погрузочно-разгрузочного оборудования», «Затраты на аренду автотрейлера» соотносятся с процедурой отправки контейнеров, а затраты «Обязательные затраты терминально-складского комплекса», «Затраты на хранение», «Издержки терминально-складского комплекса» соотносятся с процедурой загрузки контейнеров.

Итоговые документы, отражающий затратную модель по всем оперативным периодам, приведен на рис. 5

Заключительный документ – «Производственный план». Согласно нему, будут отправляться и загружаться контейнеры. Первым действием при создании документа будет выбор даты начала периода плана. В данном примере это будет 01.03.2024, так как именно с этой даты формируется себестоимость в документах «Себестоимость отправки контейнеров». Период плана – месяц, единица плана – неделя. Общий объем - 10 000,00 единиц товара в тоннах. С помощью опции «Заполнить период» в табличной части будет проведено распределение на периоды оперативного управления. Плановые индикаторы по отправке

контейнеров можно

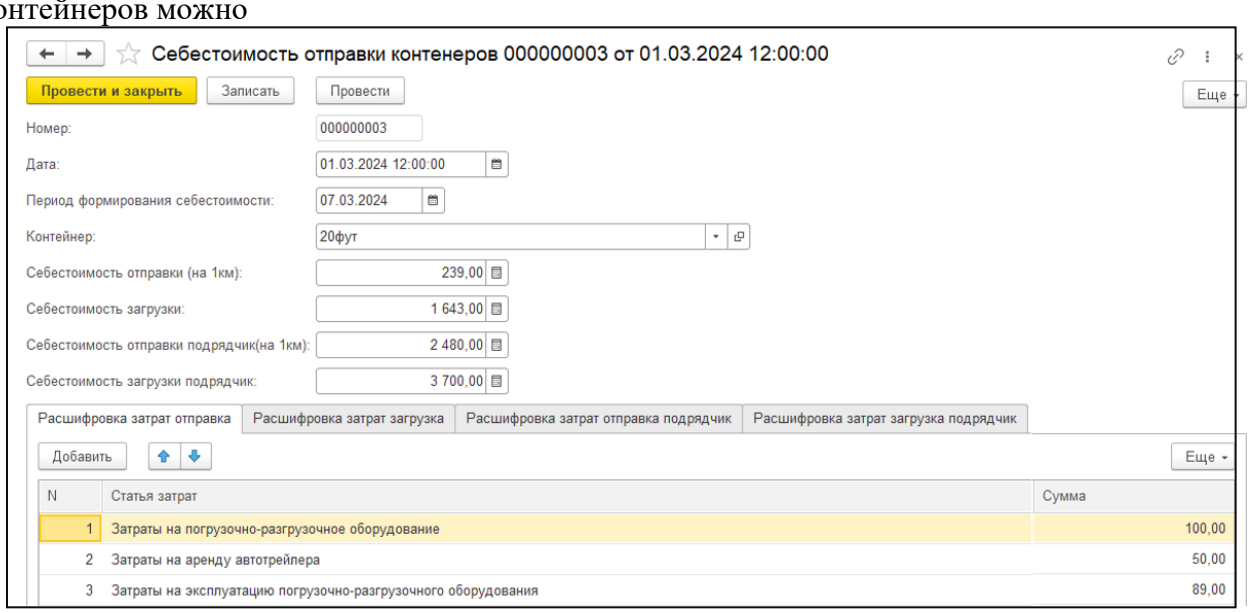


Рис.4. Формирование себестоимости процедуры отправки контейнеров

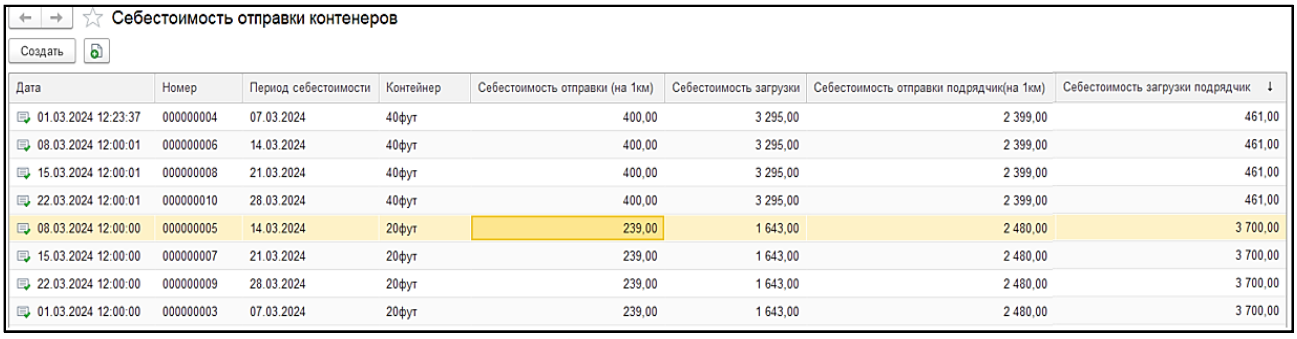


Рис.5. Документы «Себестоимость отправки контейнеров»

ввести вручную или использовать опцию «Распределить объемы», которая позволяет задать равномерный план. При ручном распределении важно учитывать, что в табличной части все объемы в сумме не должны превысить производственную программу планового периода.

В случае превышении общего объема в колонке «Объем превышении» отобразится излишек загруженных, но неотправленных контейнеров в оперативном периоде (рис.6), который позднее будет отражен в запасах, т.е. тех контейнеров, которые ждут своей отправке на складе

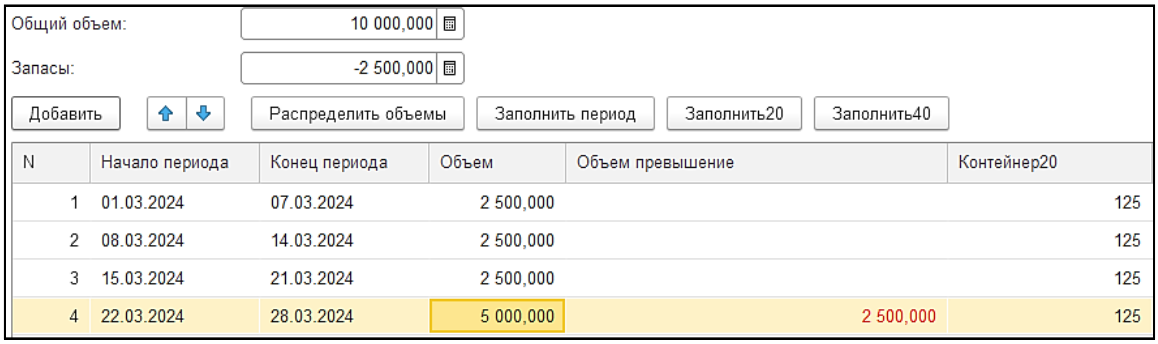
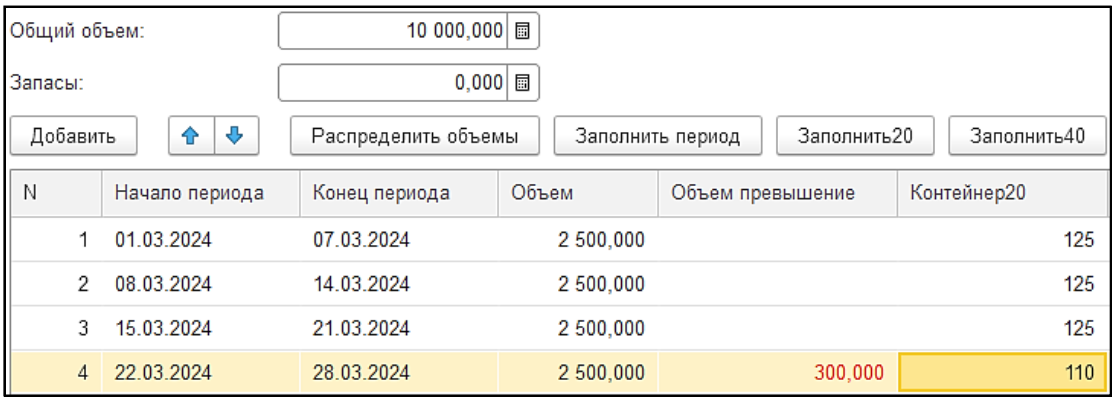


Рис.6. Вид контроля превышения общего объема в документе «Производственный план»

Стоит обратить внимание, что в поле «Запасы» сформировался отрицательный запас.

Это логично, так как в программе других документов, которые могли бы сформировать запас

Процесс внутренней загрузки самих контейнеров можно организовать как вручную, так и с помощью опций «Заполнить20» и «Заполнить40». При вводе вручную можно создавать ситуацию, при которой в один и тот же оперативный период часть товара будет отправлена контейнерами 20 футов, а другая часть будет отправлена контейнерами 40 футов. Как и в случае с распределением объемами недобор и перебор товарных единиц будет контролировать столбец «Объем превышение», как на рисунке 7.



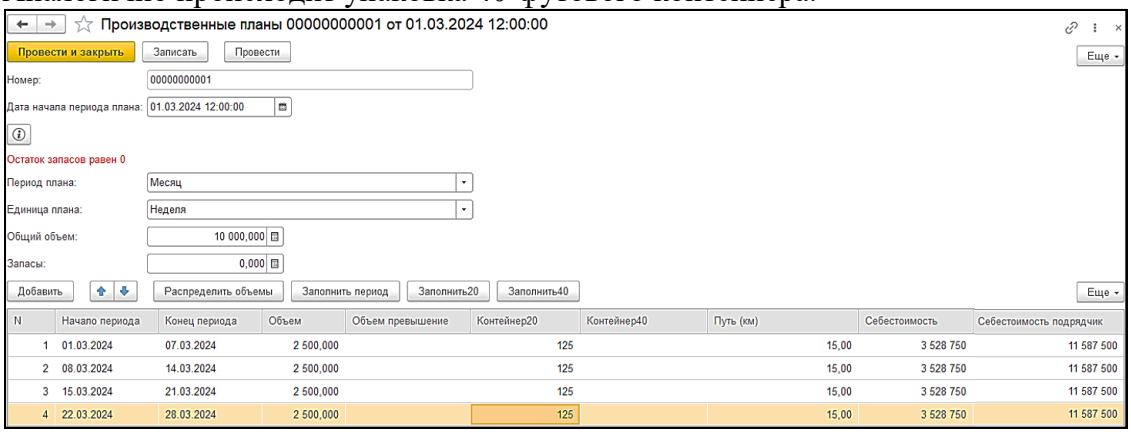
N	Начало периода	Конец периода	Объем	Объем превышение	Контейнер20
1	01.03.2024	07.03.2024	2 500,000		125
2	08.03.2024	14.03.2024	2 500,000		125
3	15.03.2024	21.03.2024	2 500,000		125
4	22.03.2024	28.03.2024	2 500,000	300,000	110

Рис. 7. Вид контроля недобора контейнеров в документе «Производственный план»

В случае недобора контейнеров будет зафиксировано то количество объема, который не распределен по контейнерам, что является прямой ошибкой, так как объем уже распределен на загрузку, но сама загрузка не состоялась и, таким образом, данная ситуация не сформирует запас – товар считается списанным и фактически отправленным. Для перевозки 2 500 тонн еженедельного объема загрузки и отправки понадобиться (рис.8) 125 контейнеров вместительностью 20 футов (что соответствует 20-ти тоннам).

Себестоимость не будет сформирована полностью, если не учесть издержки на перевозку. Поэтому фиксируя длину маршрута перевозки и стоимость выполнения операции собственными силами или с помощью подрядчика, получаем недостающие издержки в процессе управления терминалом (рис.8).

Аналогично происходит упаковка 40-футового контейнера.



N	Начало периода	Конец периода	Объем	Объем превышение	Контейнер20	Контейнер40	Путь (км)	Себестоимость	Себестоимость подрядчик
1	01.03.2024	07.03.2024	2 500,000		125		15,00	3 528 750	11 587 500
2	08.03.2024	14.03.2024	2 500,000		125		15,00	3 528 750	11 587 500
3	15.03.2024	21.03.2024	2 500,000		125		15,00	3 528 750	11 587 500
4	22.03.2024	28.03.2024	2 500,000		125		15,00	3 528 750	11 587 500

Рис.8. Производственный план для 20-футовых контейнеров

В заключении полезно сформировать отчет «Анализ расчета себестоимости», который. позволяет подробно отследить в динамике процесс формирования суммарных издержек обслуживания одного контейнера в течении всего планового периода. На рис.9 отражен подробный пример анализа в оперативном периоде планирования:

07.03.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000003 от 01.03.2024 12:00:00	1 643	3 700	239	2 480
Затраты на аренду автотрейлера			50	800
Затраты на погрузочно-разгрузочное оборудование			100	1 000
Затраты на хранение	555	400		
Затраты на эксплуатацию погрузочно-разгрузочного оборудования			89	680
Издержки терминально-складского комплекса	200	300		
Обязательные затраты терминально-складского комплекса	888	3 000		
Себестоимость отправки контенеров 000000004 от 01.03.2024 12:23:37	3 295	461	400	2 399

Рис.9. Пример анализа себестоимости по оперативному периоду планирования

За весь плановый период отчет может иметь вид (рис.10):

Анализ расчета себестоимости

Параметры: Начало периода: 29.02.2024 0:00:00
Конец периода: 01.02.2025 0:00:00

Период себестоимости Ссылка Статья затрат	Сумма загрузка	Сумма загрузка подрядчик	Сумма отправка	Сумма отправка подрядчик
07.03.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000003 от 01.03.2024 12:00:00	1 643	3 700	239	2 480
Себестоимость отправки контенеров 000000004 от 01.03.2024 12:23:37	3 295	461	400	2 399
14.03.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000005 от 08.03.2024 12:00:00	1 643	3 700	239	2 480
Себестоимость отправки контенеров 000000006 от 08.03.2024 12:00:01	3 295	461	400	2 399
21.03.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000007 от 15.03.2024 12:00:00	1 643	3 700	239	2 480
Себестоимость отправки контенеров 000000008 от 15.03.2024 12:00:01	3 295	461	400	2 399
28.03.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000009 от 22.03.2024 12:00:00	1 643	3 700	239	2 480
Себестоимость отправки контенеров 000000010 от 22.03.2024 12:00:01	3 295	461	400	2 399
07.12.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000011 от 24.03.2024 23:53:10	3 295	461	400	2 399
Себестоимость отправки контенеров 000000012 от 24.03.2024 23:54:14	1 643	3 700	239	2 480
14.12.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000013 от 24.03.2024 23:54:49	3 295	461	400	2 399
Себестоимость отправки контенеров 000000014 от 24.03.2024 23:54:56	1 643	3 700	239	2 480
21.12.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000015 от 24.03.2024 23:55:07	3 295	461	400	2 399
Себестоимость отправки контенеров 000000016 от 24.03.2024 23:56:14	1 643	3 700	239	2 480
28.12.2024	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000017 от 24.03.2024 23:56:22	3 295	461	400	2 399
Себестоимость отправки контенеров 000000018 от 24.03.2024 23:56:29	1 643	3 700	239	2 480
07.01.2025	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000019 от 25.03.2024 0:12:16	3 295	461	400	2 399
Себестоимость отправки контенеров 000000020 от 25.03.2024 0:13:07	1 643	3 700	239	2 480
14.01.2025	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000021 от 25.03.2024 0:12:34	3 295	461	400	2 399
Себестоимость отправки контенеров 000000022 от 25.03.2024 0:12:57	1 643	3 700	239	2 480
21.01.2025	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000023 от 25.03.2024 0:14:22	3 295	461	400	2 399
Себестоимость отправки контенеров 000000024 от 25.03.2024 0:14:29	1 643	3 700	239	2 480
28.01.2025	4 938	4 161	639	4 879
Себестоимость отправки контенеров 000000025 от 25.03.2024 0:14:37	3 295	461	400	2 399
Себестоимость отправки контенеров 000000026 от 25.03.2024 0:14:43	1 643	3 700	239	2 480
Итого	59 256	49 932	7 668	58 548

Рис.10. Отчет «Анализ расчета себестоимости»

Таким образом, предлагаемая учетная система управления терминально-складским комплексом позволяет вести комплексный учет расходов предприятия, расширяет возможности аналитики результатов деятельности, дает возможность получать детальную и своевременную информация по отправкам грузов и может быть использована в целях планирования затрат на отправки и прогнозирования доходов.

Заключение

В заключении необходимо отметить, что увеличение спроса на контейнерные перевозки, на поставки контейнерных грузов, возрастание мультимодальных перевозок требуют новых способов к формированию терминально-складского комплекса, направленных на применение актуальных методов управления, включение в эти методы количественного обоснования принятого решения. Создание внутренних учетных систем, направленных на четкое и слаженное отслеживание приходящих, уходящих и залеживающихся единиц продукции способствует автоматизации и прозрачности работы логистических компаний, что

облегчает процесс отслеживания материальных и информационных потоков сотрудникам компании.

Литература

1. Булгакова И.Н. Динамическое моделирование системы управления терминально-складским комплексом / И.Н.Булгакова, М.С.Левченко // Естественно-гуманитарные исследования. Международный журнал. - 2024. - № 1 (51) – С.364-371.

2. Хлебородов В.С. Анализ эффективности существующих систем организации контейнерных терминалов при использовании различного транспортно-грузового оборудования / В.С. Хлебородов, С.Н. Корнилов // Современные проблемы транспортного комплекса России. 2012. №2. С.238-251.

3. Шаповалова М. А. Выбор схемы организации работы контейнерного терминала / Я. Я. Эглит, К. Я. Эглите, М. А. Шаповалова, К. А. Кирносов // Транспортное дело России. - 2021. - № 6. - С. 144-147.

4. AXELOT. Решения для логистики. – Режим доступа: https://www.axelot.ru/about/ – (Дата обращения: 07.04.2024).

5. BizArt Group. 1С учет на контейнерном терминале. – Режим доступа: https://bizart-group.ru/nashi-razrabotki/uchet-v-agentirovanii-sudov/ – (Дата обращения: 07.04.2024).

6. SeaData. Операционная система управления контейнерным и грузовым терминалом. – Режим доступа: https://www.seadata.ru/product/cterminal-tos/ – (Дата обращения: 07.04.2024).

7. SOLVO.Автоматизация грузовых терминалов, портов, многопрофильных перегрузочных комплексов и логистических центров. – Режим доступа: https://www.solvo.ru/solutions/ports-terminals/ – (Дата обращения: 07.04.2024).

УДК *004*

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ УСТРОЙСТВА «НАЛИВАТОР»

А. Н. Леденев

Воронежский государственный университет

Аннотация. В данной научной статье рассматривается разработка мобильного приложения для управления устройством под названием «Наливатор», предназначенным для точного разлива жидких напитков в стаканчики. Приложение разработано с целью обеспечить удобное взаимодействие с устройством через мобильные устройства под управлением операционных систем iOS и Android, с интуитивно понятным интерфейсом.
Ключевые слова: мобильное приложение, управление устройством, разлив напитков, кроссплатформенная разработка, Flutter, Bluetooth 4.0.

Введение

Наливатор представляет собой технически совершенное устройство, предназначенное для разлива любых жидких напитков в стаканчики с высокой точностью (до миллилитра). Устройство обладает настройками стороны налива, которые позволяют определить направление разлива, а также автоматическим и ручным режимами работы. В дополнение к этому, «Наливатор» оснащен счетчиком общего и сбрасываемого «пробега», что позволяет контролировать количество разлитых напитков [1].

Управление устройством осуществляется с помощью энкодера, расположенного на корпусе, но это не всегда удобно. Для облегчения процесса использования устройства было разработано решение в виде мобильного приложения, способного функционировать на операционных системах iOS и Android. Разработка данного приложения направлена на минимизацию неудобств, связанных с использованием традиционных методов управления, и обеспечение более удобного и доступного способа взаимодействия с устройством.

Разработанное мобильное приложение для управления устройством «Наливатор» отличается минималистичным и интуитивно понятным интерфейсом, который специально адаптирован для максимального удобства использования. Основной упор сделан на простоту и легкость навигации, что делает приложение доступным для широкого круга пользователей, вне зависимости от их уровня технической подготовки.

Разработка и реализация мобильного приложения для управления устройством «Наливатор» на платформах iOS и Android является ключевым аспектом данного исследования, и его технические детали, функциональные возможности и потенциал будут подробно рассмотрены в дальнейших разделах настоящей научной работы.

1. Выбор платформы для разработки приложения

Выбор оптимальной платформы для разработки мобильного приложения является первоочередным этапом, определяющим успешность проекта. В данном разделе мы рассмотрим и проанализируем три популярные платформы: Kotlin, Flutter и Xamarin, выделим их основные плюсы и минусы.

Kotlin, являющийся официальным языком разработки для платформы Android, предлагает совместимость с Java и удобный синтаксис. Однако, его ограниченная поддержка

платформы iOS и отсутствие нативного подхода к ней делают его менее привлекательным в контексте кроссплатформенной разработки.

Flutter, с другой стороны, выделяется кроссплатформенностью, быстрой разработкой и высокой производительностью. Несмотря на некоторые ограничения, такие как меньшее сообщество разработчиков и ограничения в доступе к нативным функциям устройств, Flutter обеспечивает эффективное и удобное средство для разработки качественных приложений для различных платформ.

Xamarin, использующий C# и .NET, обеспечивает нативный интерфейс и производительность. Однако, его относительная ограниченность инструментов и библиотек делают его менее привлекательным выбором.

Учитывая характеристики платформ для разработки мобильных приложений, выбор падает на Flutter в качестве наилучшего решения для разработки приложения, так как важны следующие факторы.

Во-первых, кроссплатформенность: поскольку проект направлен на широкую аудиторию пользователей как на устройствах iOS, так и на устройствах Android, необходимо выбрать платформу, которая обеспечит эффективную кроссплатформенную разработку. Flutter предлагает решение этой задачи, позволяя создать один код, который работает на обеих платформах без необходимости писать отдельный код для каждой из них.

Во-вторых, важное значение имеет большое сообщество разработчиков. Для успешного развития и поддержки проекта необходимо наличие активного и разнообразного сообщества, обеспечивающего доступ к обширным ресурсам, ответам на вопросы и поддержке. Flutter обладает широким и быстрорастущим сообществом. Это облегчает решение возникающих задач и ускоряет процесс разработки.

В-третьих, для эффективной разработки приложения важно иметь доступ к качественной документации, которая объясняет основные концепции и возможности платформы. Flutter предлагает полноценную и простую в понимании документацию, что делает процесс изучения и использования этой технологии более эффективным и приятным для разработчиков.

2. Интерфейс приложения

При разработке приложения был уделен особый акцент на создание удобного и интуитивно понятного пользовательского интерфейса (UI), который обеспечивает простоту использования для различных категорий пользователей.

При запуске приложения пользователь встречает главный экран, который является центральной точкой взаимодействия. На данном экране размещен удобный слайдер, предназначенный для выбора объема напитка, а также кнопка «Налить», которая отправляет команду с выбранным объемом на устройство «Наливатор». (рис. 1)

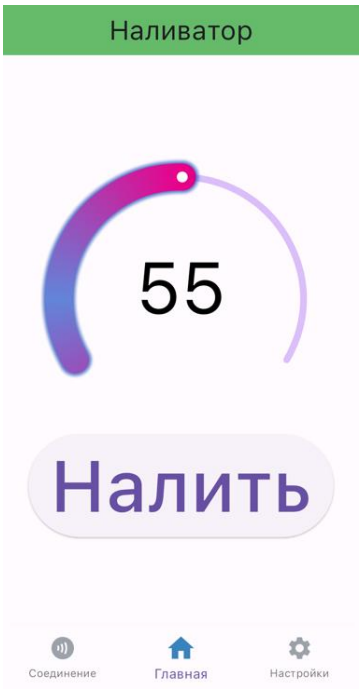


Рис. 1. Главный экран

Экран «Соединение» предназначен для обнаружения и отображения доступных устройств Bluetooth. Пользователю предоставляется возможность запустить процесс поиска новых устройств путем нажатия соответствующей кнопки. Это позволяет удобно управлять подключением к устройству и обеспечивает простоту в процессе установления соединения. (рис. 2)

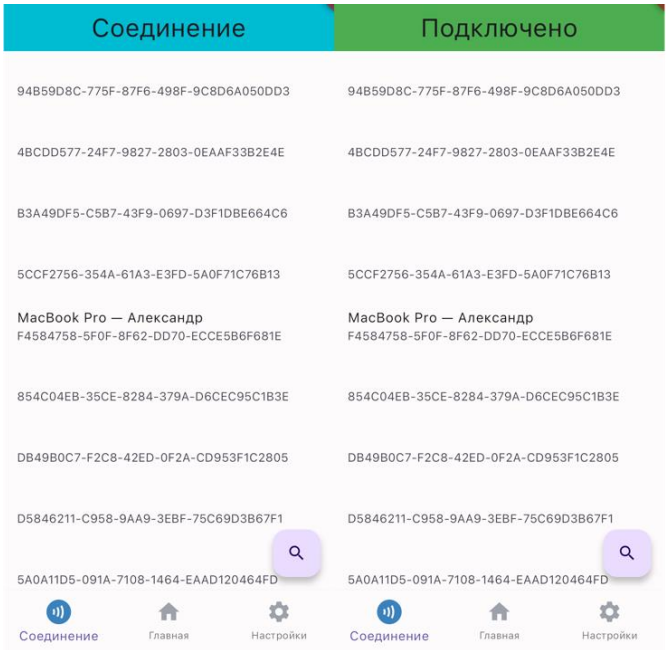


Рис. 2. Экран «Соединение»

На экране «Настройки» представлены различные параметры, позволяющие пользователю настраивать основные параметры: выбор стороны налива и настройку автоматического режима в соответствии с его потребностями. (рис. 3)

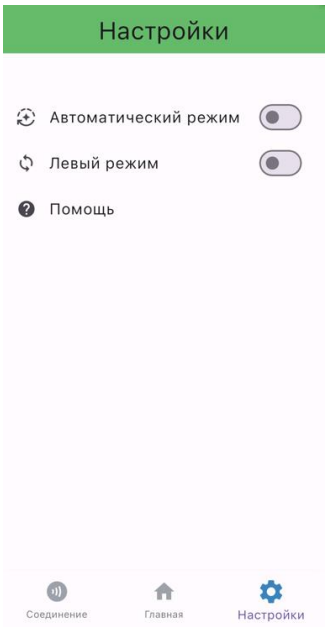


Рис. 3. Экран «Настройки»

Навигация между экранами осуществляется с помощью меню, расположенного в нижней части экрана. Такой вид навигации широко распространен в мобильных приложениях и интуитивно понятен для большинства пользователей, что обеспечивает удобство перемещения между различными функциональными частями приложения и повышает общую доступность интерфейса.

3. Bluetooth 4.0 (BLE)

Bluetooth 4.0, также известный как Bluetooth Low Energy (BLE) или Bluetooth Smart, представляет собой беспроводной протокол передачи данных, разработанный с целью обеспечения низкого энергопотребления и расширения возможностей беспроводной связи. Одной из ключевых его особенностей является его расширенный радиус действия по сравнению с предыдущими версиями Bluetooth. Это позволяет устройствам поддерживать стабильное соединение на больших расстояниях без значительной потери качества связи.

При интеграции функциональности Bluetooth в приложение важным моментом стало обеспечение совместимости аппаратной части с требованиями платформы iOS, которая поддерживает только Bluetooth версии 4.0 или Bluetooth Low Energy (BLE). Для этого в аппаратной части устройства «Наливатор» был добавлен модуль HM-10, который обеспечивает возможность использования Bluetooth 4.0.

Модуль HM-10 представляет собой компактное и удобное решение, позволяющее осуществлять беспроводное подключение и обмен данными по протоколу Bluetooth версии 4.0. (рис. 4)

Программная часть реализована с использование библиотеки «Flutter Blue», которая предоставляет необходимые методы и функционал для взаимодействия с устройствами, поддерживающими Bluetooth. Эта библиотека обладает набором методов, позволяющих осуществлять подключение к устройствам и передачу данных между приложением и устройством.

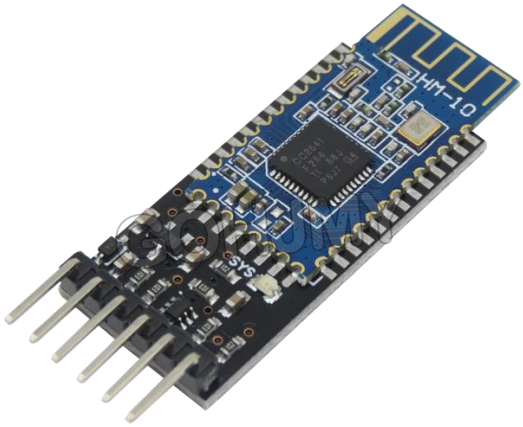


Рис. 4. Модуль HM-10

Заключение

В результате работы получаем полностью работоспособное приложение, которое отвечает требованиям потребителя. Тестирование показало, что ошибок в ходе работы минимальное количество и устройство полностью готово к эксплуатации.

На всем путем от идеи до реализации мобильного приложения для устройства «Наливатор» стояли различные трудности и вызовы, требующие тщательного анализа и поиска оптимальных решений. Однако одной из наиболее серьезных трудностей было интегрирование Bluetooth 4.0 в приложение, особенно с учетом совместимости с устройствами iOS. Этот процесс требовал особого внимания к выбору подходящей технологии и библиотеки, способной обеспечить стабильное и эффективное взаимодействие с устройством через Bluetooth.

В результате, приложение было успешно разработано и реализовано, представляя собой инструмент управления устройством «Наливатор» дистанционно. Благодаря интеграции Bluetooth 4.0 и разработке интуитивно понятного пользовательского интерфейса, приложение обеспечивает простоту и удобство взаимодействия с устройством даже на расстоянии. Разработанное приложение успешно решает задачи облегчения взаимодействия с устройством, предоставляя пользователям удобный и эффективный способ управления устройством дистанционно. Это открывает новые возможности для использования устройства в различных сферах, таких как общественное питание, мероприятия и домашнее использование, делая его более доступным и удобным для широкого круга пользователей.

В данной статье был представлен полный обзор всех этапов разработки, начиная от выбора платформы и проектирования интерфейса, и заканчивая интеграцией Bluetooth 4.0. Преодоление трудностей и успешная реализация проекта стали возможны благодаря систематическому подходу, глубокому анализу и эффективному использованию современных технологий и инструментов разработки

Литература

1. Леденев А. Н. Умное устройство для разлития напитков «Наливатор» / А. Н. Леденев // Математика, информационные технологии, приложения: сб. тр. межвуз. науч. конференции молодых ученых и студентов (Воронеж, 27 апреля 2023 г.) - Воронеж : Научная книга, 2023. – С. 253–256.
2. Алеев А. Быстрый старт Flutter-разработчика/ А. Алеев, – Ridero, 2020. – 170 с.

3. Flutter Documentation. – Режим доступа: https://docs.flutter.dev. – (Дата обращения: 30.01.2024).
4. Flutter Blue Documentation. – Режим доступа: https://pub.dev/packages/flutter_blue. – (Дата обращения: 13.02.2024).
5. Flutter на практике. Прокачиваем навыки мобильной разработки с помощью открытого фреймворка от Google / под ред. Д. А. Мовчан. – Москва : ДМК-Пресс, 2020. – 328 с.

УДК 519.246.8

Применение методов машинного обучения для прогнозирования инфляции

А. В. Лепендин

Воронежский государственный университет

Введение

В исследовании рассматривается применение методов машинного обучения для прогнозирования индекса потребительских цен, который на официальном уровне является индикатором инфляции в России.

Повышение цен на товары и услуги всегда является неприятным сюрпризом для покупателя, поскольку напрямую влияет на уровень его благосостояния. Темпы изменения цен носят непостоянный характер, цена может не измениться ни разу или изменится несколько раз за период и при этом на разную величину. Индекс цен отражает среднюю динамику, ему присваивается значение, равное единице или 100 в некоторый базисный период, значения индекса в другие периоды времени представляет собой среднее пропорциональное или процентное изменение по сравнению с базисным периодом. Индекс цен удобно использовать для сравнения цен в разных странах, регионах, городах на один и тот же момент времени.

Индекс потребительских цен берет начало своей истории в восемнадцатом веке, в 70-х годах девятнадцатого столетия были предложены индексы Ласпейреса и Пааше. Для расчета ИПЦ существует множество формул, которые могут давать разный результат, большинство экономистов сходятся во мнении, что формула индекса должна относится к классу гиперболических индексов, чтобы обеспечить аппроксимацию индекса стоимости жизни.

Популярным индексом является также индекс Лоу, при вычислении которого считается процентное изменение общей стоимости заданной корзины между двумя сравниваемыми периодами. В индексе Лоу можно использовать любой набор корзины. Существует пара особых случаев, если корзина относится к базисному периоду цен, то говорят об индексе Ласпейреса, в случае, когда корзина относится к другому периоду времени, то говорят об индексе Пааше.

Известно, что ИПЦ не охватывает цены на активы, инвестиционные товары, товары, потребляемые предприятиями и органами государственного управления, однако корреляция этого индекса с общей инфляцией остается высокой из-за того, что потребительские расходы составляют большой процент от совокупных конечных расходов, в расчетах используется потребительская корзина базового года

В России данные по ИПЦ публикуются Федеральной службой государственной статистики, в качестве базового периода выступает предыдущий месяц или декабрь предыдущего года [7].

На основе набора данных о ценах на товары и услуги можно сформировать временные ряды. Временной ряд —это ряд значений некоторого статистического показателя, взятый в хронологическом порядке времени, в динамике, значения ряда проиндексированы на основе атрибута даты и времени. Временные ряды анализируется с использованием различных методов и моделей, применяются нейросетевые модели и многопроцессорная обработка данных. Прогнозирование с использованием временных рядов позволяет получить значения показателя для будущих периодов времени, что является важным отличием от статичных данных [9]. В работе рассматривается прогнозирование индекса потребительских цен и

исследуется влияние на инфляцию различных факторов, таких как цена на топливо, ставка центрального банка, курс доллара к рублю.

Актуальность исследования обусловлена необходимостью принятия верных и своевременных решений для минимизации рисков и повышения стабильности экономики. Индекс потребительских цен является ключевым показателем для принятия экономических решений, таких как корректировка заработной платы и социальных пособий [6].

1. Исходные данные, способы расчета инфляции

На момент начала исследования имеются данные по более чем 12 миллионам единиц товаров и услуг. Особенность набора данных состоит в том, что частота наблюдения цен для разных товаров неоднородна, частота фиксации цены каждого отдельного товара непостоянна.

Описание полей набора данных приведено в табл. 1.

Таблица 1.

Описание полей набора данных

WebPriceId	Уникальный номер товара/услуги
DateObserve	Дата наблюдения
StockStatus	Статус товара/услуги на дату наблюдения (InStock – в продаже, OutOfStock – отсутствует в продаже)
CurrentPrice	Цена товара/услуги на дату наблюдения (Если StockStatus = OutOfStock – значение отсутствует)

Длина временных рядов в целом имеет нормальное распределение с длинным хвостом, это отражено на рисунке 1, по оси x указывается количество наблюдений (изменений цены) в одном временном ряду (ед.), по оси y — количество товаров с соответствующим количеством наблюдений (млн. ед.)). Около 10 млн. товаров из 12 имеющихся в базе имеют не более 10 изменений цены, причем 6,7 млн товаров — не более 2 наблюдений.

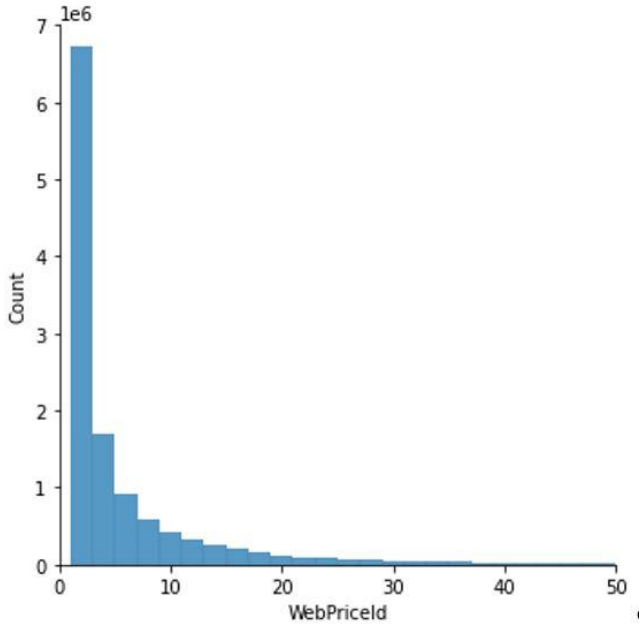


Рис. 1. Распределение временных рядов

При рассмотрении более 8 миллионов наблюдений, выяснилось, что в них содержится информация по более чем 3 миллионам товаров и услуг, анализировать весь набор данных и

строить по нему прогноз нецелесообразно (рис. 2):

```
print("Размерность набора данных –", data.shape)
print("Количество записей с уникальным id, т. е. количество товаров –",  data.WebPriceId.nunique())
```

```
Размерность набора данных – (8578975, 4)
Количество записей с уникальным id, т. е. количество товаров – 3506102
```

Рис. 2. Малое количество записей для количества товаров

Индекс потребительских цен товара или услуги рассчитывается по формуле (1):

$$ИПЦ = \frac{P_t}{P_{t-1}} - 1, \quad (1)$$

где P_t — цена на товар/услугу на конец месяца t, P_{t-1} — цена на товар/услугу на конец месяца $t-1$.

Индекс цен на группу товаров рассчитывается по формуле (2):

$$ИПЦгр.т. = \frac{\sum_{i=1}^{N} ИПЦтовара_i}{N} \quad (2)$$

где N — количество товаров в группе.

Расчет ИПЦ на группу товаров можно проводить иначе, с помощью индекса Лоу. В этом случае ИПЦ рассчитывается как отношение сумм цен на группу товаров на конец месяца t и $t-1$. Индекс показывает на сколько изменилась цена некоторой «продуктовой корзины», формула (3):

$$ИПЦгр.т. = \frac{\sum_{i=1}^{N} P_{ti}}{\sum_{i=1}^{N} P_{t-1i}} - 1 \quad (3)$$

Способ расчета ИПЦ влияет на конечные результаты, пример такого случая приведен на рис. 3, при подсчете ИПЦ на группу товаров по формуле (2) получили повышение цен на 4%. Однако, при таком подходе не учитывается стоимость каждого товара и, соответственно стоимость суммы товаров. Если учесть стоимость товаров, вести расчет по формуле (3), то получим снижение цен на 11.4%.

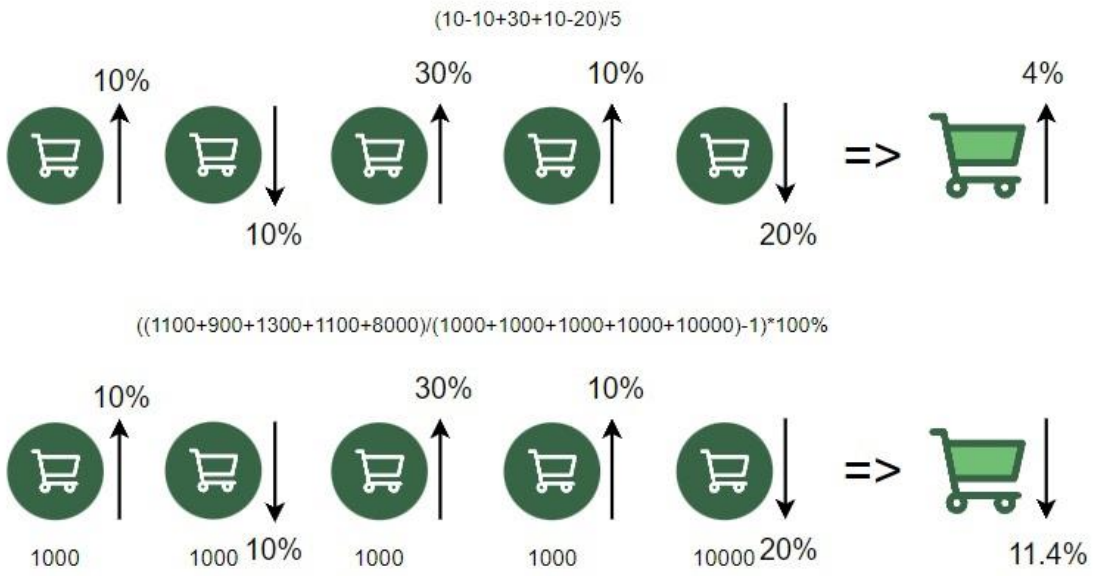


Рис. 3. Способы расчета ИПЦ на группу товаров

2. Предобработка данных

Работа с нерегулярными временными рядами имеет определенные сложности при анализе и моделировании, и требует специальных методов и техник, способных учитывать нерегулярность наблюдений.

Определяется методика заполнения пропусков в данных, анализируются способы исключения выбросов (ошибки измерения, аномальные события и др.).

Пропуски в данных, вызванные отсутствием товара в наличии, заполняются последней наблюдаемой ценой.

Существует много моделей и методов для прогнозирования временных рядов, но большинство из них, такие как ARIMA и ее модификации, статистические модели (Statsforecast), рекуррентные (RNN) или сверточные (CNN) нейронные сети рассчитаны на регулярные интервалы времени между наблюдениями. Для эффективного обучения подобных моделей необходимо проведение ресемплирования данных. Ресемплирование — это процесс изменения частоты наблюдений во временном ряде путем агрегации или интерполяции существующих данных на новые временные интервалы. Такой процесс применяется не только, если исходные данные имеют нерегулярные интервалы, но и, если требуется привести данные к более низкой или высокой частоте для анализа или моделирования. Пример приведения данных к более низкой частоте представлен на рис. 4, данные по дням переведены в данные по месяцам.

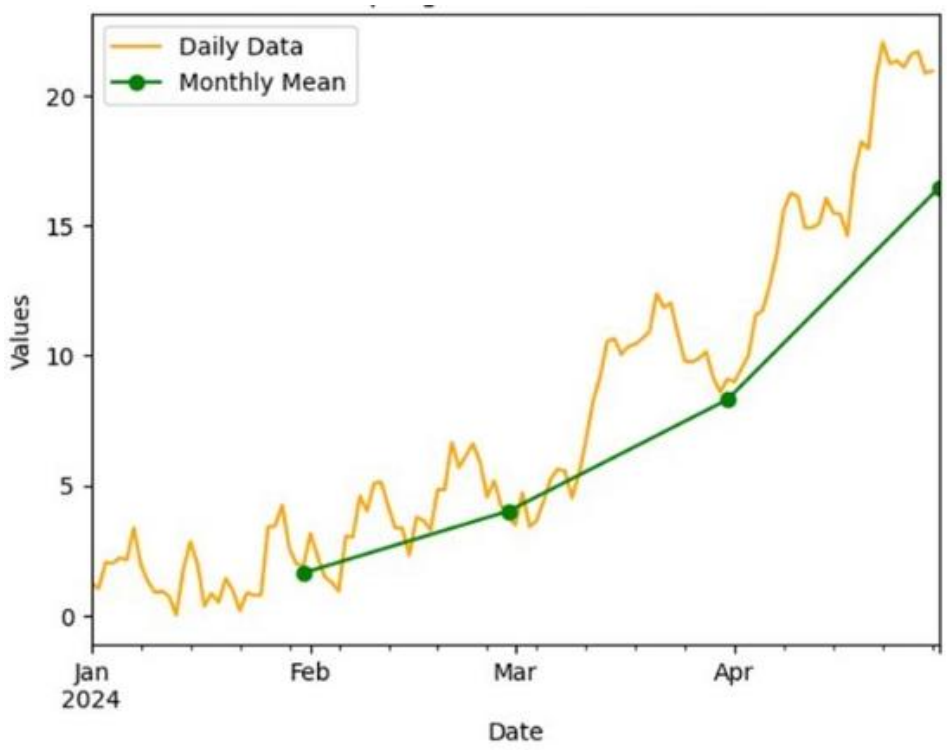


Рис. 4. Пример ресемплирования данных

Пример ресемплирования (данные группируются в дневные временные интервалы, пропуски заполняются с помощью линейной интерполяции) данных по товару с WebPriceId=4 представлен на рис. 5.

Рис. 5. Пример ресемплирования данных

В исследовании проводится также работа по исключению выбросов в данных, выбросы в данных возникают в аномальных ситуациях, при ошибках в наблюдениях. Для товаров и услуг аномальной ситуацией может оказаться очень высокая скидка. Обычно товар стоит около 100 руб. но было одно наблюдение, когда он стоил 10 руб., пример приведен на рис.6:

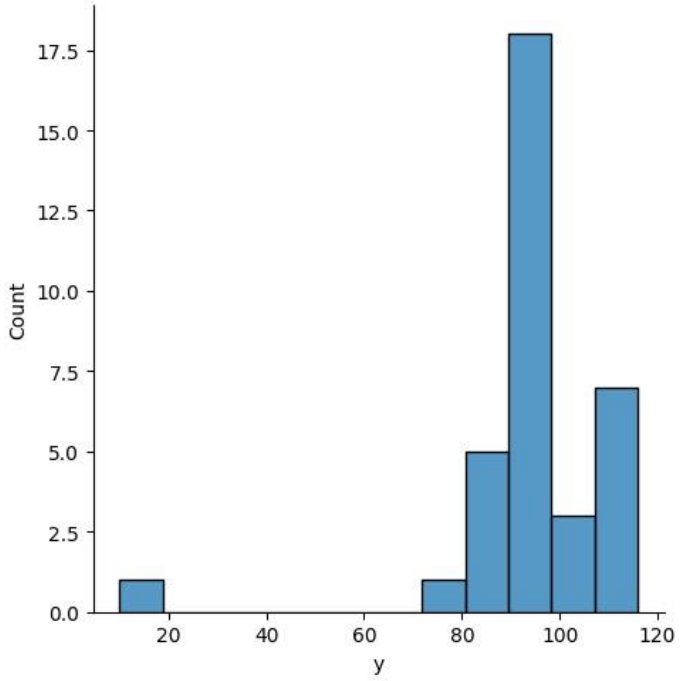


Рис. 6. Пример выбросов в данных

Для того, чтобы очистить данные от выбросов в машинном обучении существуют различные методы: межквартильный размах (IQR), стандартное отклонение, метод DBSCAN. В связи с особенностями данных цен товаров, их распределения, цена на один товар может не меняться все время наблюдений или цены не принимают вид нормального распределения, в некоторых сценариях упомянутые выше методы могут привести к исключению большого количества записей, которые на самом деле не относятся к выбросам. Метод DBSCAN же обычно используется в решении задач кластеризации, когда формируются однородные сразу по нескольким признакам группы. [1]

Было принято решение удалять наблюдения методом, адаптированным под специфику задачи: исключаются строки, в которых значение в столбце 'CurrentPrice' больше, чем в три раза превышает среднее значение 'CurrentPrice' товара с тем же уникальным номером товара/услуги и удалять строки, в которых значение в столбце 'CurrentPrice' меньше чем в три раза среднего значения 'CurrentPrice' с тем же уникальным номером товара/услуги.

3. Применение нейронных сетей для прогнозирования временных рядов

Рассмотрим проблему применения рекуррентной нейронной сети для предсказания временного ряда.

Для записей по одному из товаров создается временной ряд с заданным размером скользящего окна, далее набор данных разбивается на обучающую и тестовую выборки, форма набора данных изменяется, чтобы соответствовать требованиям входа LSTM (3D массивы).

Далее создается модель нейронной сети, в последовательный стек слоев добавляется

слой LSTM с 4 нейронами, полносвязный (плотный) слой с одним нейроном, модель компилируется с функцией потерь — среднеквадратическая ошибка и оптимизатором — адаптивная оценка момента.

Модель обучается на обучающей выборке в течение 10 итераций и размером пакета 1, на обучение нейронной сети по данным временного ряда одного товара потребовалось более 20 секунд (рис. 7):

```
Epoch 1/10
621/621 - 3s - loss: 0.1096 - 3s/epoch - 4ms/step
Epoch 2/10
621/621 - 1s - loss: 0.0116 - 1s/epoch - 2ms/step
Epoch 3/10
621/621 - 1s - loss: 0.0081 - 987ms/epoch - 2ms/step
Epoch 4/10
621/621 - 1s - loss: 0.0047 - 1s/epoch - 2ms/step
Epoch 5/10
621/621 - 1s - loss: 0.0020 - 1s/epoch - 2ms/step
Epoch 6/10
621/621 - 1s - loss: 4.1979e-04 - 986ms/epoch - 2ms/step
Epoch 7/10
621/621 - 1s - loss: 3.6552e-05 - 973ms/epoch - 2ms/step
Epoch 8/10
621/621 - 1s - loss: 1.7643e-05 - 980ms/epoch - 2ms/step
Epoch 9/10
621/621 - 1s - loss: 1.7548e-05 - 1s/epoch - 2ms/step
Epoch 10/10
621/621 - 1s - loss: 1.7016e-05 - 1s/epoch - 2ms/step
Потребовалось времени на обучение модели (10 итераций): 22.494597994999822 секунд
```

Рис. 7. Обучение модели нейронной сети

Время, затраченное на обучение нейронной сети по данным одного товара сопоставимо с тем, что было получено в дальнейшем при обучении модели Prophet и расчете инфляции по данным более 50 товаров. Изменение параметров нейронной сети, добавление различных слоев, в том числе dropout, который применяется для предотвращения переобучения, в большинстве случаев увеличивает время обучения и ухудшает точность на обучающей и тестовой выборках. Визуализация реальной и предсказанной цены товара (рис. 8):

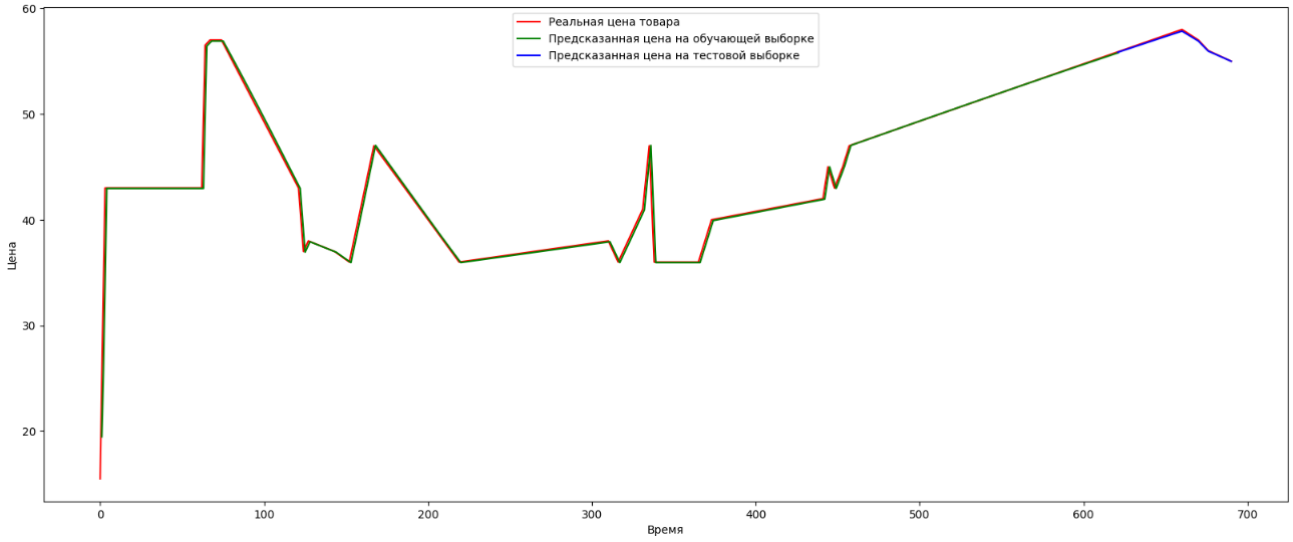


Рис. 8. Визуализация предсказания цены рекуррентной нейронной сетью

3. Использование библиотеки Prophet для прогнозирования временных рядов

Модель для прогнозирования временных рядов, которая практически не подвержена отрицательному эффекту выбросов в данных и отсутствующих данных — это Prophet.

Аддитивную модель Prophet, можно представить в виде формулы (4):

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \quad (4)$$

Слагаемые в этой формуле — это сезонные компоненты, тренд, аномальные дни и ошибка, неучтённая моделью, соответственно.

После обучения модели для каждого товара предсказывается цена на выбранную дату и на дату спустя месяц. Результаты расчета инфляции за апрель 2021 года и январь 2022 года на основании 10000 наблюдений представлены на рисунках 9 и 10:

```
print("В среднем инфляция за месяц составила:", mean(InflationPerMonthArr1),"%")
print('Время, потребовавшееся для прогнозирования цен на товары в цикле =' ,testtime1,"секунд" )
print('Инфляция рассчитывается за месяц, следующий за выбранная датой: ' ,prevmonth )

В среднем инфляция за месяц составила: 1.600487246715331 %
Время, потребовавшееся для прогнозирования цен на товары в цикле = 186.2921826839447 секунд
Инфляция рассчитывается за месяц, следующий за выбранная датой:  2021-03-31
```

Рис. 9. Результаты расчета инфляции за апрель 2021 года на основании 10000 наблюдений

```
print("В среднем инфляция за месяц составила:", mean(InflationPerMonthArr1),"%")
print('Время, потребовавшееся для прогнозирования цен на товары в цикле =' ,testtime1,"секунд" )
print('Инфляция рассчитывается за месяц, следующий за выбранная датой: ' ,prevmonth )

В среднем инфляция за месяц составила: 0.8304291368266918 %
Время, потребовавшееся для прогнозирования цен на товары в цикле = 166.10291695594788 секунд
Инфляция рассчитывается за месяц, следующий за выбранная датой:  2021-12-31
```

Рис. 10. Результаты расчета инфляции за январь 2022 года на основании 10000 наблюдений

Предлагается сравнить полученные значения инфляции с данными, публикуемыми Росстатом [10] (рисунок 11) [10]:

Индексы потребительских цен на товары и услуги[1)] по Российской Федерации в 1991-2023[*)]гг.

К содержанию

на конец периода, в %

	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
к концу предыдущего месяца																																	
январь	106,20	345,30	125,80	117,90	117,77	104,11	102,34	101,51	108,38	102,33	102,76	103,09	102,40	101,75	102,62	102,43	101,68	102,31	102,37	101,64	102,37	100,50	100,97	100,59	103,85	100,96	100,62	100,31	101,01	100,40	100,67	100,99	100,84
февраль	104,80	138,00	124,70	110,82	111,02	102,79	101,54	100,89	104,13	101,04	102,28	101,16	101,63	100,99	101,23	101,66	101,11	101,20	101,65	100,86	100,78	100,37	100,56	100,70	102,22	100,63	100,22	100,21	100,44	100,33	100,78	101,17	100,46
март	106,30	129,90	120,10	107,41	108,94	102,80	101,43	100,64	102,79	100,64	101,86	101,08	101,05	100,75	101,34	100,82	100,59	101,20	101,31	100,63	100,62	100,58	100,34	101,02	101,21	100,46	100,13	100,29	100,32	100,55	100,66	107,61	100,37
апрель	163,50	121,70	118,70	108,49	108,47	102,16	100,96	100,38	103,03	100,89	101,79	101,16	101,02	100,99	101,12	100,35	100,57	101,42	100,69	100,29	100,43	100,31	100,51	100,90	100,46	100,44	100,33	100,38	100,29	100,83	100,58	101,56	100,38
май	103,00	111,90	118,10	106,91	107,93	101,60	100,94	100,50	102,22	101,75	101,78	101,69	100,80	100,74	100,80	100,48	100,63	101,35	100,57	100,50	100,48	100,52	100,66	100,90	100,35	100,41	100,37	100,38	100,34	100,27	100,74	100,12	
июнь	101,20	119,10	119,90	106,00	106,66	101,17	101,10	100,08	101,91	102,55	101,62	100,53	100,80	100,78	100,64	100,28	100,95	100,97	100,60	100,39	100,23	100,89	100,42	100,62	100,19	100,36	100,61	100,49	100,04	100,22	100,69	99,65	
июль	100,60	110,60	122,39	105,33	105,38	100,72	100,93	100,17	102,82	101,79	100,45	100,72	100,71	100,92	100,46	100,67	100,87	100,51	100,63	100,36	99,99	101,23	100,82	100,49	100,80	100,54	100,07	100,27	100,20	100,35	100,31	99,61	
август	100,50	108,60	126,00	104,62	104,56	99,79	99,86	103,67	101,16	100,98	100,01	100,09	99,59	100,42	99,86	100,19	100,09	100,36	100,00	100,55	99,76	100,10	100,14	100,24	100,35	100,01	99,46	100,01	99,76	99,96	100,17	99,48	
сентябрь	101,10	111,50	123,00	107,96	104,46	100,33	99,70	138,43	101,48	101,32	100,60	100,40	100,34	100,43	100,25	100,09	100,79	100,80	99,97	100,84	99,96	100,55	100,21	100,65	100,57	100,17	99,85	100,16	99,84	99,93	100,60	100,05	
октябрь	103,50	122,90	119,50	115,00	104,72	101,20	100,17	104,54	101,37	102,11	101,09	101,07	101,00	101,14	100,55	100,28	101,64	100,91	100,00	100,50	100,48	100,46	100,57	100,82	100,74	100,43	100,20	100,35	100,13	100,43	101,11	100,18	
ноябрь	108,90	126,10	116,39	114,61	104,56	101,88	100,61	105,67	101,23	101,52	101,36	101,61	100,96	101,11	100,74	100,63	101,23	100,83	100,29	100,81	100,42	100,34	100,56	101,28	100,75	100,44	100,22	100,50	100,28	100,71	100,96	100,37	
декабрь	112,10	125,20	112,50	116,44	103,20	101,42	100,96	111,61	101,26	101,64	101,60	101,54	101,10	101,14	100,82	100,79	101,13	100,69	100,41	101,08	100,44	100,54	100,51	102,62	100,77	100,40	100,42	100,84	100,36	100,83	100,82	100,78	
к декабрю предыдущего года																																	
декабрь	260,40	2608,84	939,90	315,14	231,30	121,81	111,03	184,43	136,53	120,18	118,58	115,06	111,99	111,73	110,92	109,00	111,87	113,28	108,80	108,78	106,10	106,57	106,47	111,35	112,91	105,39	102,51	104,26	103,04	104,91	108,39	111,94	102,06[2)]

[1)] в соответствии с Федеральным планом статистических работ, утвержденным распоряжением Правительства Российской Федерации от 6 мая 2008 г. № 671-р, Росстатом разрабатывается показатель **«Индекс потребительских цен» (ИПЦ)**, который используется в качестве одного из основных показателей, характеризующих уровень инфляции в Российской Федерации;

[2)] Апрель 2023 г. в % к декабрю 2022 г.

Руководствуясь теорией статистики, для получения ИПЦ за произвольный период необходимо перемножить все входящие в этот временной промежуток индексы, характеризующие изменение цен в отчетном периоде по сравнению с предыдущим. Так, например, индекс потребительских цен по Российской Федерации за период апрель 2022 г. - сентябрь 2022 г. рассчитывается следующим образом:

101,56 * 100,12 : 100 * 99,65 : 100 * 99,61 : 100*99,48:100*100,05:100= 100,46%

Обращаем Ваше внимание, что в январе 1998 г. была проведена деноминация, в результате которой произошло уменьшение масштаба цен в 1000 раз.

Методология расчета ИПЦ размещена на сайте Росстата: Главная страница / Статистика / Официальная статистика / Цены. инфляция/ Методология

Рис. 11. ИПЦ по данным Росстата

При сравнении рассчитанных выше значений с официальными данными Росстата, можно заметить, что полученный в работе результат за апрель 2021 года, равный 101.6%, превышает 100.58% из таблицы Росстата. Если же сравнивать показатели ИПЦ за январь 2022 года, то результат 100.83% близок к данным Росстата — 100.67%.

Увеличим количество наблюдений и товаров для расчета инфляции, включим 50000 наблюдений и вновь вычислим инфляцию за январь 2022 года (рис. 12):

```
print("В среднем инфляция за месяц составила:", mean(InflationPerMonthArr1),"%")
print('Время, потребовавшееся для прогнозирования цен на товары в цикле =' ,testtime1,"секунд" )
print('Инфляция рассчитывается за месяц, следующий за выбранная датой: ' ,prevmonth )
print('Количество товаров и услуг, участвующих в расчете', len(groups_by_ticker))
```

```
В среднем инфляция за месяц составила: 1.9960487530883722 %
Время, потребовавшееся для прогнозирования цен на товары в цикле = 606.4397473335266 секунд
Инфляция рассчитывается за месяц, следующий за выбранная датой:  2021-12-31
Количество товаров и услуг, участвующих в расчете 1087
```

Рис. 12. Результаты расчета ИПЦ за январь 2022 года на основании 50000 наблюдений

Приведенные выше результаты получены при вычислении ИПЦ по формуле (2). Далее приведем результаты расчетов ИПЦ для корзины товаров по формуле (3).

Если провести ресемплирование данных 303 товаров, то можем получить динамику общей стоимости (суммы) этих товаров (рис. 13):

Рис. 13. Динамика суммы цен на товары с октября 2020 г. по июнь 2022 г.

Проводится расчет ИПЦ корзины товаров с использованием многопроцессорной обработки. Прогноз модели строится на 90 дней вперед. В анализ попадают только те товары, по которым имеется прогноз на выбранный месяц. Время обучения модели сокращается (117 вместо более 160 секунд для 10000 наблюдений), на рисунке 14 приведены результаты расчётов май 2022 года.

```
ticker
4       3.021562
6       2.031455
7       1.236226
8       2.069500
16      2.510959
          ...
297    -1.275008
299     0.291996
300    -2.169875
301   -11.719765
302     0.007079
Name: percentage_change, Length: 90, dtype: float64
Инфляция в среднем за май 2022 составила: 0.2209820737871952 %
Время на обучение моделей для 303 товаров: 117.38280916213989
```

Рис. 14. Прогнозируемая инфляция за май 2022 г.

Рассчитаем также инфляцию за июнь 2022 года, результат отражен на рисунке 15:

```
ticker
2       0.585689
3     -60.211798
4       2.838330
6       1.926782
7       0.069647
          ...
298    -4.337187
299     0.281754
300    -2.146454
301   -12.847392
302     0.006850
Name: percentage_change, Length: 156, dtype: float64
Инфляция в среднем за июнь 2022 составила: -0.48147476776002496 %
Время на обучение моделей для 303 товаров: 117.38280916213989
```

Рис. 15. Прогнозируемая инфляция за июнь 2022 г.

Возвращаясь к таблице Росстата, обратим внимание на значение индекса потребительских цен в мае и июне 2022 года, моделям удалось показать хорошее приближение к динамике ИПЦ (рис. 16).

	январь	февраль	март	апрель	май	июнь	июль
2024	100,86	100,68					
2023	100,84	100,46	100,37	100,38	100,31	100,37	100,63
2022	100,99	101,17	107,61	101,56	100,12	99,65	99,61
2021	100,67	100,78	100,66	100,58	100,74	100,69	100,31
2020	100,40	100,33	100,55	100,83	100,27	100,22	100,35
2019	101,01	100,44	100,32	100,29	100,34	100,04	100,20
2018	100,31	100,21	100,29	100,38	100,38	100,49	100,27
2017	100,62	100,22	100,13	100,33	100,37	100,61	100,07

Рис. 16. ИПЦ по данным Росстата

С увеличением скорости работы моделей можно попробовать загрузить большее количество данных. Рассмотрим 100000 записей, содержащих информацию по динамике цен 1885 товаров. Предсказание строится на 90 дней вперед, в итоговых результатах учитываются товары, по которым имелись данные на даты, не более чем на 90 дней отстоящие от предсказываемых. Получили предсказание инфляции за июль 2022 года, которое расходится с результатами от Росстата, рис. 17:

```
ticker
2          0.601688
3       -156.375144
4          2.851992
7          0.071919
15        -2.630441
             ...
1877       2.890831
1878      -8.259846
1881       2.214011
1883      -5.419174
1884      -0.090550
Name: percentage_change, Length: 1019, dtype: float64
Инфляция на корзину товаров за июль 2022 составила: 0.6161644718624947 %
Время на обучение моделей для 1885 товаров: 875.1429650783539 секунд
Количество учтенных в расчете инфляции товаров/услуг: 1019
```

Рис. 17. Прогнозируемая инфляция за июль 2022 г.

Результаты предсказания ИПЦ по моделям не всегда бывают адекватными, сильно зависят от выборки товаров. Для прогнозирования инфляции следует учитывать большее количество факторов, не только динамику цен на некоторую группу товаров. Было решено проанализировать влияние на инфляцию различных факторов, глобальных и локальных.

4. Анализ влияния локальных и глобальных факторов на ИПЦ

Идея состоит в том, чтобы:

1) Подобрать такие факторы, для которых можно сформировать временные ряды и осуществить прогноз.

2) Обучить регрессионную модель на основе имеющихся данных, где зависимая переменная — инфляция в месяц, а независимые переменные (предикторы) — это помесячные значения факторов.
3) Значения факторов, предсказываемые с помощью временных рядов, использовать в обученной регрессионной модели для предсказания инфляции.

В задачах регрессии для оценки модели обычно используется среднеквадратическое отклонение (5) и коэффициент детерминации (6):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2, \quad (5)$$

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}. \quad (6)$$

После поиска и обработки данных, приведения их к требуемому формату, получим таблицу (рис.18): цена на нефть марки Brent, ставка центрального банка РФ, курс доллара к рублю, дата, инфляция за месяц по данным Росстата.

	Oil	rate	Close	date	inflation
0	50.30	12.50	33.422295	2009-04-30	0.69
1	64.98	12.00	31.866024	2009-05-31	0.57
2	68.11	11.50	31.003659	2009-06-30	0.60
3	70.08	11.00	31.496465	2009-07-31	0.63
4	69.02	10.75	31.819557	2009-08-31	0.00
..	...	...	...	...	...
172	87.29	12.00	95.389346	2023-08-31	0.28
173	95.86	13.00	96.330428	2023-09-30	0.87
174	86.82	15.00	97.176854	2023-10-31	0.83
175	81.72	15.00	90.486181	2023-11-30	1.11
176	77.69	16.00	90.962143	2023-12-31	0.73

[177 rows x 5 columns]

Рис. 18. Данные по факторам с 2009 г. по 2023 г.

Поскольку инфляция характеризует изменение цен за некоторый период, то было решено для цены на топливо и курса доллара к рублю рассчитать процентное изменение за период, в нашем случае — месяц (рис. 19).

	rate	inflation	Oil_pc	Usd_pc
1	12.00	0.57	0.291849	-0.046564
2	11.50	0.60	0.048169	-0.027062
3	11.00	0.63	0.028924	0.015895
4	10.75	0.00	-0.015126	0.010258
5	10.00	-0.03	-0.046363	-0.019252

Рис. 19. Первые 5 строк в измененном наборе данных

Построим матрицу корреляции (рис. 20).

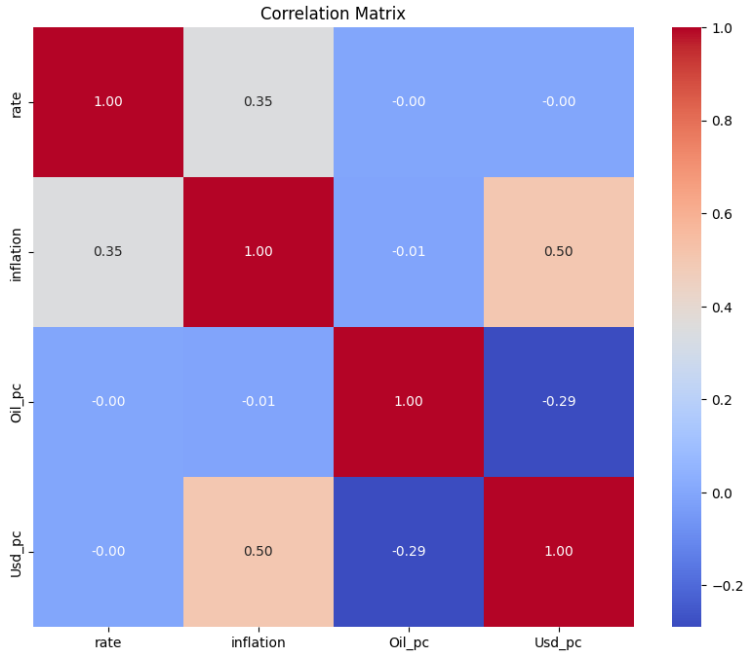


Рис. 20. Матрица корреляции

Заметим, что существует положительная корреляция между инфляцией и ставкой ЦБ на уровне 0.35, а также между инфляцией и изменением курса доллара на уровне 0.5. Корреляция не обязательно означает, что изменение одной переменной вызовет изменение другой переменной. Это подтверждается при построении таких регрессионных моделей как: дерево решений, случайный лес, градиентный бустинг. Проводилась стандартизация данных, подбирались различные параметры моделей, чтобы не допустить переобучения. Во всех случаях наблюдалась следующая закономерность: высокий коэффициент детерминации на обучающей выборке и около нулевой на тестовой выборке (рис. 21).

```
# Calculate evaluation metrics
mae = mean_absolute_error(y_test, predictions2)
mse = mean_squared_error(y_test, predictions2)
r2 = r2_score(y_test, predictions2)

# Print evaluation metrics
print(f'Средняя абсолютная ошибка на тестовой выборке: {mae}')
print(f'Среднеквадратическая ошибка на тестовой выборке: {mse}')
print(f'Коэффициент детерминации на тестовой выборке: {r2}')
```

```
Средняя абсолютная ошибка на обучающей выборке: 0.12598226950354613
Среднеквадратическая ошибка на обучающей выборке: 0.03668483510638298
Коэффициент детерминации на обучающей выборке: 0.8257685845341202
Средняя абсолютная ошибка на тестовой выборке: 0.5799861111111111
Среднеквадратическая ошибка на тестовой выборке: 1.6258707152777783
Коэффициент детерминации на тестовой выборке: 0.03477173032937031
```

Рис. 21. Расчет оценок для регрессионной модели

Поведение инфляции в краткосрочном периоде может быть близко аппроксимировано с помощью простых моделей, основанных только на временном ряде инфляции. Прогнозирование инфляции с использованием других макроэкономических переменных в качестве предикторов имеет серьезные ограничения, связанные, во-первых, с потенциально

большим количеством информативных предикторов, во-вторых, с продолжительностью имеющихся временных рядов. Если количество наблюдений в ряду слишком мало, особенно при большом количестве предикторов, то любые модели зачастую подстраиваются к случайным закономерностям в обучающей выборке. Вне обучающей выборки модель дает неточный прогноз.

Заключение

В результате проведения исследования были решены следующие задачи: проанализирована предметная область, изучены основные аспекты формирования цен на товары и услуги; проанализированы факторы, оказывающие влияние на изменение цен товаров и услуг; проанализированы существующие методы расчета инфляции; разработан алгоритм обработки большого объема данных для перевода данных во временные ряды; разработано программное обеспечение на языке python для построения прогнозов по предложенным временным рядам; проведен вычислительный эксперимент на основе разработанного программного обеспечения, проведен анализ результатов эксперимента; проанализированы глобальные и локальные факторы, оказывающие влияние на инфляцию.

Литература

1. Бринк Х. Машинное обучение / Х. Бринк, М. Феверолф, Дж. Ричардс. – Санкт-Петербург : Питер, 2017. – 336 с.
2. Коэльо Л.П. Построение систем машинного обучения на языке Python / пер. с англ. А. А. Слинкина / Л. П. Коэльо, В. Ричарт. – Москва : ДМК Пресс, 2016. – 302 с.
3. Рашка С. Python и машинное обучение. – Москва : ДМК Пресс, 2017. – 418 с.
4. Лимановская О. В. Основы машинного обучения: учебное пособие / О. В. Лимановская, Т. И. Алферьева. – 2-е изд. – Москва : ФЛИНТА, 2022. – 88 с.
5. Воронов, М. В. Системы искусственного интеллекта : учебник и практикум для вузов / М. В. Воронов, В. И. Пименов, И. А. Небаев. – Москва : Юрайт, 2022. – 256 с.
6. Фергюсон А. Когда деньги умирают. Кошмар гиперинфляции / пер. с англ. С. Э. Борич – Минск : Попурри, 2012. – 320 с.
7. Социально-экономическое положение России. – Москва : Федеральная служба государственной статистики, – 2022. – 313с.
8. Теория и практика машинного обучения / В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов. – Ульяновск : УлГТУ, 2017. – 290 с.
9. Хайндман Р. Прогнозирование : принципы и практика / пер. с англ. А. Логунов / Р. Хайндман, Дж. Атанасопулос. – Москва : ДМК Пресс, 2023. – 382 с.
10. Федеральная служба государственной статистики. – Режим доступа: https://rosstat.gov.ru. – (Дата обращения: 01.12.2022).

УДК 519.876.5

О ФОРМАХ ЗАВИСИМОСТИ ФУНКЦИОНАЛЬНОГО ОТКЛИКА ОТ ЧИСЛЕННОСТИ ХИЩНИКА В МОДЕЛЯХ ХИЩНИК-ЖЕРТВА.

Д. В. Логунов

Воронежский государственный университет

Введение

Многие из наиболее интересных явлений в природе связаны с взаимодействием между организмами. Эти взаимодействия часто тонкие, косвенные и их трудно обнаружить. Например, есть взаимодействия, при которых один организм полностью или частично поглощает другой. Сюда входят такие взаимодействия как хищник-жертва, травоядное-растение и паразит-хозяин. Эти связи являются основными способами передачи энергии по пищевым цепям. Они являются важным фактором экологии популяций, определяя смертность добычи и рождение новых хищников.

Одним из наиболее распространенным подходом к построению математических моделей биологии является их описание с помощью дифференциальных уравнений. Математические модели и логика предполагают, что связанная система хищника и жертвы должна циклически повторяться:

- число хищников увеличивается, когда число жертв велико,
- число жертв сокращается из-за хищничества,
- количество хищников уменьшается, а жертвы восстанавливаются, и так до бесконечности.

В настоящей статье будет проанализирована и рассмотрена с точки зрения приложений система уравнений Лотки-Вольтерры. Эти уравнения были разработаны для описания динамики биологических систем, в которых самым важным аспектом является взаимодействие хищников и их жертв, а конкуренция, болезни и другие факторы игнорируются. С точки зрения математики, уравнения Лотки-Вольтерры представляют собой систему нелинейных дифференциальных уравнений первого порядка. Важным фактором в понимании того, что происходит с решениями этой системы с течением времени, является анализ критических точек. Такие точки соответствуют постоянным решениям или положениям равновесия.

1. Общая форма уравнения хищник-жертва

Модели хищник-жертва, которые будут изучены в этой диссертации, следуют двум общим принципам. Первый из них заключается в том, что динамика популяции может быть разложена на процессы рождения и смерти,

$$\frac{dN}{dt} = рождаемость - смертность$$

Второй принцип состоит в принципе сохранения массы [1], утверждающем, что хищники могут расти только как функция того, что они съели. С учетом этих двух принципов мы можем записать каноническую форму системы хищник-жертва,

$$\frac{dN}{dt} = f(N) - g(N,P) - \mu_N(N)N$$
$$\frac{dP}{dt} = eg(N,P) - \mu_P(P)P \quad (1)$$

где: f(N) - прирост численности добычи на душу населения; - функциональный отклик g(N, P) (количество съеденной добычи на хищника за единицу времени [2]); темп естественной смертности μ_P и μ_N.

Численный отклик (прирост численности хищников как функция потребления) считается в этой системе пропорциональным функциональному отклику. Обратите внимание, что указанная выше система становится системой Лотка-Вольтерра, когда $f(N) = r$, $g(N,P) = aN$, $\mu_N(N) = 0$ и $\mu_p(N) = \mu$. В литературе было предложено множество других форм для функций f, g и $\mu_x, x \in \{N,P\}$. В частности, функциональный отклик провоцировал творческие работы, например работа [3], перечисляющих более 50 моделей.

Обычно считается, что хищничество является основной причиной смерти для добычи. В этом случае можно пренебречь $\mu_N(N)$, то есть принять как 0 (пока существует хищник). Я буду следовать этому подходу в данной статье, заменив прирост численности стандартным логистическим ростом и также рассматривая смертность хищников как постоянную, как в уравнениях Лотки-Вольтерра, получая уравнения

$$\frac{dN}{dt} = r\left(1 - \frac{N}{K}\right)N - g(N,P)P$$
$$\frac{dP}{dt} = eg(N,P)P - \mu P \quad (2)$$

Традиционно функциональный отклик в системе (2) предполагается функцией только от обилия добычи g(N, P)=g(N), например

$$g(N) = aN \qquad (\textit{Лотка})$$
$$g(N) = \frac{aN}{1 + ahN} \quad (\textit{Холинг})$$

Назовем этот случай функциональным откликом, зависящем от добычи. Таким образом, как и в случае Лотки-Вольтерра, она полностью определяется параметрами хищников. Графически это выражается в вертикальной изоклине хищников. Увеличение предельной вместимости K (обогащения) изменит только изоклину добычи и приведет к дестабилизации равновесия, так называемому парадоксу обогащения [4]. Традиционный подход, основанный на зависимости от добычи, в контексте системы (2), всегда приводит к модели хищник-жертва, контролируемой сверху, которая дестабилизируется при обогащении.

Предыдущий анализ показывает, где мы должны изменить нашу систему (2), чтобы нарушить этот шаблон: функциональный отклик должен также зависеть от численности хищников. Назовём этот случай зависимостью от хищника. Обычно более высокая плотность хищников приводит к более частым встречам между ними. Это может привести к потере эффективности хищничества либо просто из-за времени, утраченного на обнаружение того, что встреченный организм не является добычей (это время не тратится на поиск другой добычи), либо из-за активного взаимодействия между хищниками. Следовательно, численность хищников должна отрицательно влиять на функциональный отклик.

$$\frac{dg(N,P)}{dP} < 0$$

2. Формы зависимости функционального отклика от хищника

Зависимость хищников от функционального отклика была введена много раз. Первое предположение было о том, что частота атак (количество съеденных жертв) должна уменьшаться с увеличением плотности хищников. Они предложили экспоненциальное убывание со скоростью m.

$$a = \alpha P^{-m} \tag{3}$$

Параметр m можно интерпретировать как коэффициент взаимодействия. Чем сильнее хищники мешают друг другу, тем больше должно быть m. Хотя изначально он использовался только для объяснения частоты нападений, включение его в систему (2) также показало стабилизирующий эффект. Есть две технические проблемы с этой формулировкой. Во-первых, существует незначительная проблема размерности, которая может быть устранена путем введения фиктивного параметра с размерностью (представляющего одного хищника) и переписывания (3) как

$$a = \alpha \left(\frac{P}{u_p} \right)^{-m}$$

Вторая проблема более серьезная и касается математического анализа систем, использующих этот тип частоты нападений: из-за экспоненты часто невозможно явно найти точку равновесия, что делает анализ более затруднительными (требующий использования теории неявных функций или других более сложных техник).

Обе проблемы можно избежать, введя зависимость от хищника так, как предложено Беддингтоном [5],

$$g(N,P) = \frac{aN}{1 + ahN + cP} \tag{4}$$

Где h является временем взаимодействия с добычей (таким же, как и в обычном функциональном отклике типа II Холлинга) и эмпирической константой. Беддингтон [5] вывел эту модель на основе поведенческих механизмов, предполагая, что хищник тратит время либо на поиск добычи, либо на взаимодействие с добычей, либо на взаимодействие с другими хищниками (распознавание, прямой контакт). в этом контексте является произведением частоты встреч хищника и времени взаимодействия хищника. Систему (2) вместе с этим функциональным откликом можно анализировать обычным образом, решая уравнения для равновесий и проверяя стабильность (например, с помощью критериев Раута-Гурвица). При этом было отмечено, что в случае $1 + ahN \square cP$ система становится контролируемой донором. Таким образом, функциональный отклик (4) предоставляет простой способ внедрения признаков снизу вверх в систему хищник-жертва. Однако его редко используют в математических исследованиях или при применении к реальным экологическим системам. Вероятные причины заключаются в том, что параметр сложно оценить, а математический анализ не так прост, как, например, в случае с функциональными откликами Лотки-Вольтерра или типа II Холлинга, потому что он является функцией двух отдельных аргументов и требует использования частных производных.

Более простой способ включения зависимости от хищника был предложен в [6]. Там представили функциональный отклик как функцию только одного аргумента, но теперь этим аргументом является соотношение добычи к хищнику.

$$g(N,P) = g\left(\frac{N}{P} \right) \tag{5}$$

Это особый случай зависимости от хищника, и модели, использующие этот тип функционального отклика, обычно называются моделями, зависящими от соотношения.

3. Экспериментальное доказательство существования зависимости от отношения

Рождером Ардити и Хенни Саем [7] был проведен ряд экспериментов с двумя видами хищных кладоцер, D_1 и D_2, питающихся водорослями. Водоросли выращивали в отдельном сосуде и добавляли с постоянной скоростью в первый сосуд с хищником. Выход из этого сосуда (только с водорослями, хищники не могли переходить в следующий сосуд) направлялся во второй сосуд с хищниками и так далее. Гипотеза заключалась в том, что в случае вертикальной изоклины хищника (зависимость от добычи) хищник должен был выживать только в первом сосуде, пожирая водоросли до определенного уровня, который не мог бы поддерживать популяции хищников в следующих сосудах. С другой стороны, если изоклина хищника наклонена, хищники, вероятно, смогут выживать также в последовательных сосудах. Хищники, выбранные для эксперимента, имели особое пространственное поведение: D_1 равномерно плавали в среде, в то время как D_2 старалась находиться около стенок сосуда. D_1 следовало предсказанному паттерну вертикальной изоклины хищника, в то время как D_2 сохранялся в нескольких последовательных сосудах. В следующем этапе D_1 не позволяли перемещаться по всему сосуду, пока D_1 равномерно распределяли с помощью частого перемешивания. При такой настройке D_1 сохранялся в нескольких сосудах, тогда как D_2 существовал только в первом. Эти эксперименты ясно демонстрируют, что пространственная гетерогенность может вызвать наклонную изоклину хищника. Однако они не показывают, что изоклина хищника проходит через начало, как это предсказывают модели, зависящие от соотношения. На рис. 1 это иллюстрируется на примере трех последовательных сосудов. Модель качественно предсказывает один и тот же паттерн: количество хищников в равновесии уменьшается геометрически от одного сосуда к следующему.

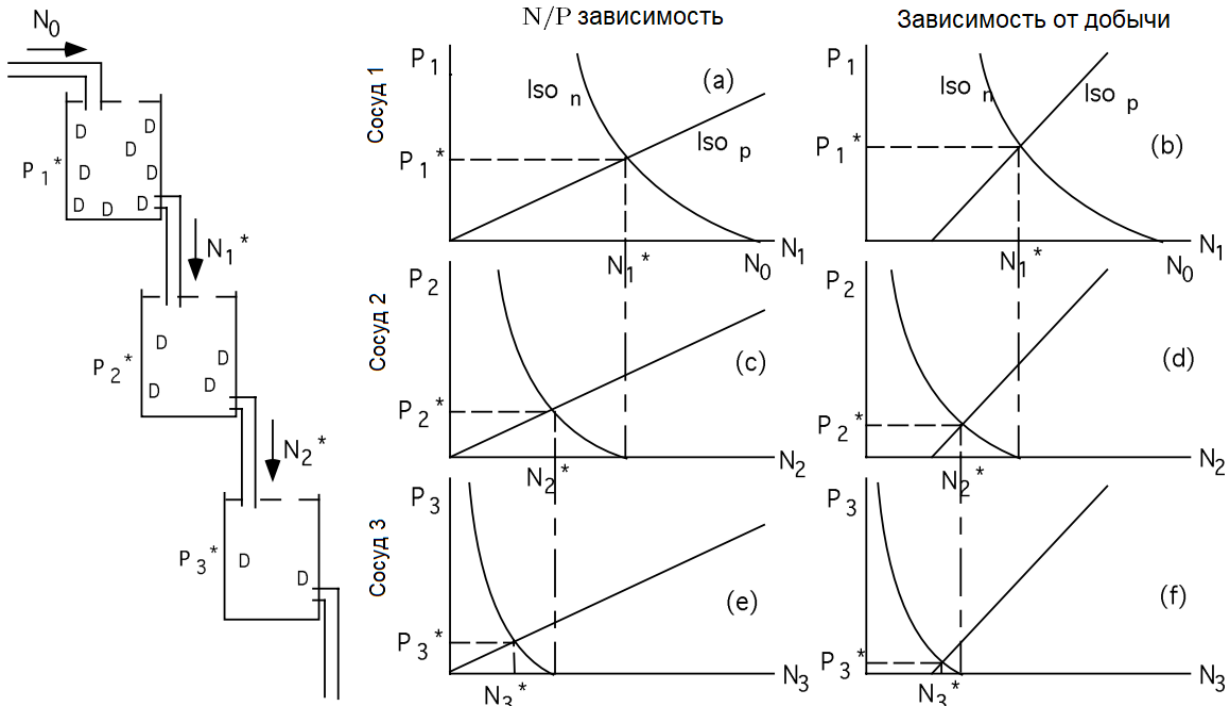


Рис. 4. Прогнозы равновесных количеств хищников в цепных экспериментах с сосудами

Заключение

На сегодняшний день теория зависимости от соотношения остается предметом активных исследований в области экологии и математической биологии. Этот подход к моделированию взаимодействия хищник-жертва или более сложных пищевых цепей и сетей питания вызывает интерес и обсуждения среди ученых и практиков.

Несмотря на значительное количество теоретических исследований и экспериментов, поддерживающих применимость зависимости от коэффициента в моделях хищник-жертва, этот подход все еще не получил широкого признания как общепринятый теоретический подход. Некоторые ученые рассматривают его как полный абсурд, в то время как другие опасаются, что он может отвлечь внимание от более общих форм зависимости хищника. Критики также отмечают, что хищники могут потенциально выживать при произвольно низких уровнях ресурсов, однако для этого нет экспериментальных данных. Поэтому статистические методы сравнения, такие как нелинейная регрессия в рамках вероятностной модели, или разработка экспериментов, способных опровергнуть оба типа изоклин, вертикальные и проходящие через начало координат, требуются для дальнейших исследований.

Есть также некоторые проблемы с математическим пониманием динамики моделей зависимости от соотношения, особенно в области начала координат, где модели не определены. Тщательное математическое понимание модели может указать как на слабые стороны, так и на потенциально сильные черты.

Несмотря на эти сложности, зависимость от соотношения остается перспективным исследовательским направлением. В дальнейших исследованиях необходимо сравнивать зависимость от коэффициента с данными на равных основаниях, дальше разрабатывать методы моделирования и исследовать возможные альтернативы включения зависимости хищника в моделирование.

Научный руководитель: доцент, доктор физико-математических наук, профессор кафедры математического и прикладного анализа ВГУ, Шишкина Элина Леонидовна.

Литература

1. Ginzburg, L. R. Assuming reproduction to be a function of consumption raises doubts about some popular predator-prey models / L. R. Ginzburg // Journal of Animal Ecology – 1998 – №2 – С. 325–327.

2. Solomon, M. E. The natural control of animal populations / M. E. Solomon // Journal of Animal Ecology – 1949 – №1 – С. 1–35.

3. Bastin G. On-line Estimation and Adaptive Control of Bioreactors / G. Bastin, D. Dochain. – Belgium : Elsevier, 1990. – 379 с.

4. Rosenzweig M. L Paradox of enrichment: destabilization of exploitation ecosystems in ecological time / M. L. Rosenzweig // Science – 1971 – №3969 – С. 385–387.

5 Beddington J. R. Mutual interference between parasites or predators and its effect on searching efficiency / J. R. Beddington // Journal of Animal Ecology – 1975 – №44 – С. 331–340.

6. Arditi R. Coupling in predator-prey dynamics: Ratio-Dependence / R. Arditi, L. R. Ginzburg // Journal of Theoretical Biology – 1989 – №139 – С. 311–326.

7. Arditi, R. Empirical evidence of the role of heterogeneity in ratiodependent consumption / R. Arditi, H. Saïah // Ecology – 1992 – №73 – С. 1544–1551.

УДК 519.161

ЖАДНЫЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ ТРЁХИНДЕКСНОЙ ПЛАНАРНОЙ ЗАДАЧИ О НАЗНАЧЕНИЯХ

Р.А. Лысенко

Воронежский государственный университет

Введение

Задача о назначениях является одной из фундаментальных задач линейного программирования. Она возникает во многих сферах жизни, когда необходимо сопоставить две группы объектов друг другу с наибольшей эффективностью: назначение работников на места, распределение машин на производства и т.д. Задача формулируется следующим образом: пусть имеется квадратная матрица затрат C размера $N \times N$, необходимо выбрать N элементов так, чтобы их сумма была минимальна (максимальна). При этом в каждой строке и каждом столбце может быть выбран только один элемент.

Введём квадратную матрицу переменных X размера $N \times N : x_{ij} = \begin{cases} 1, \textit{если элемент выбран} \\ 0, \textit{иначе}, \end{cases}$

где $i,\ j = \overline{1, N}$.

Тогда математическую модель задачи можно представить в виде:

$$L(x) = \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} x_{ij} \to \min,$$

$$\sum_{j=1}^{N} x_{ij} = 1,\ i \in \overline{1, N},$$

$$\sum_{i=1}^{N} x_{ij} = 1,\ j \in \overline{1, N}.$$

Для классической задачи о назначениях известны точные полиномиальные алгоритмы решения: венгерский алгоритм, метод ветвей и границ и т.д. Дальнейшие исследования и расширение задачи привели к рассмотрению многоиндексных задачи о назначениях. При числе индексов три и более задачи о назначениях принадлежит к классу NP трудных задач – для них не известен точный полиномиальный алгоритм решения. Полный перебор всех возможных комбинаций назначений может быть вычислительно сложным или даже невозможным при больших размерах входных данных. В связи с этим становятся актуальными разработка и исследование приближенных алгоритмов для их решения.

Целью данной статьи является разработка жадного алгоритма для решения планарной трёхиндексной задачи о назначениях.

1. Трёхиндексные задачи о назначениях

Наиболее распространёнными и изученными являются аксиальная и планарная трёхиндексные задачи о назначениях. Они находят широкое применение при решении задач распределения ресурсов, планирования, многоцелевого отслеживания и т.д.

1.1. Аксиальная трёхиндексная ЗОН

Пусть имеется кубическая матрица затрат C размера $N \times N \times N$, аксиальная трёхиндексная ЗОН состоит в таком выборе N элементов, чтобы их сумма была минимальна(максимальна). При этом в каждой плоской матрице при фиксировании одного из индексов (рис.1) может быть выбран только один элемент.

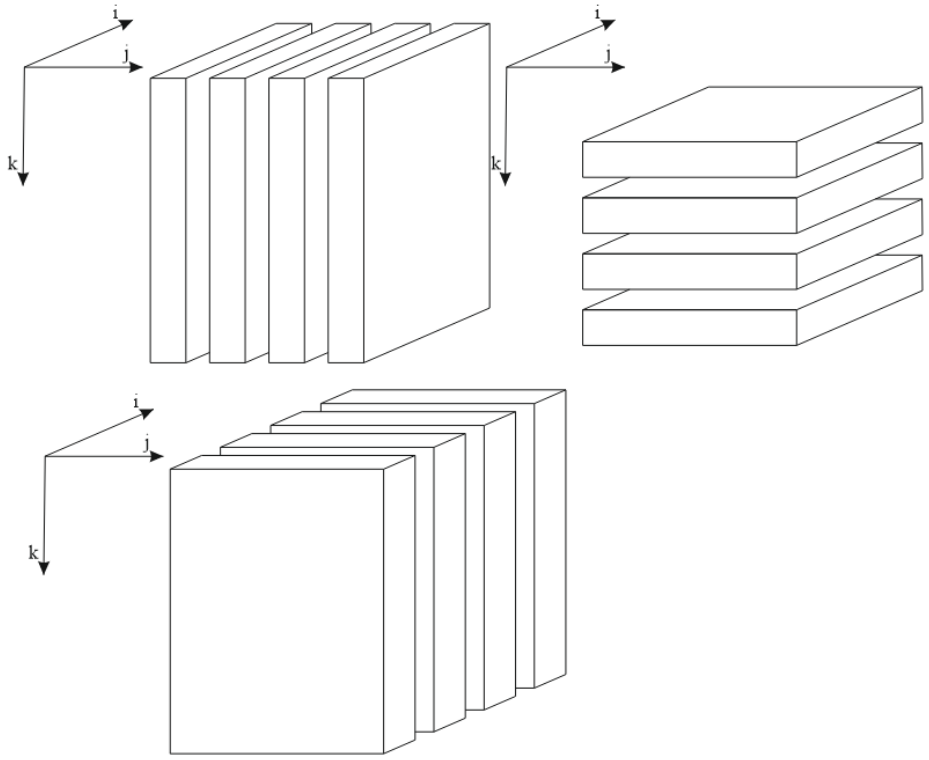


Рис. 1. Плоские матрицы при фиксировании одного из индексов

Введём кубическую матрицу x размера $N \times N \times N$:

$$x_{ij}^{k} = \begin{cases} 1, \textit{если элемент выбран} \\ 0, \textit{иначе}, \end{cases}$$

где $i,\ j = \overline{1, N}$.

Тогда математическую модель задачи можно представить в виде:

$$L(x) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} c_{ij}^{k} k_{ij}^{k} \rightarrow \min,$$

$$\sum_{k=1}^{N}\sum_{j=1}^{N} x_{ij}^{k} = 1,\ i \in \overline{1, N},$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}^{k} = 1,\ k \in \overline{1, N},$$

$$\sum_{k=1}^{N}\sum_{i=1}^{N} x_{ij}^{k} = 1,\ j \in \overline{1, N}.$$

1.2. Планарная трёхиндексная ЗОН

Пусть имеется кубическая матрица затрат C размера $N \times N \times N$, планарная трёхиндексная ЗОН состоит в таком выборе N^2 элементов, чтобы их сумма была минимальна(максимальна). При этом в каждой строке матрицы при фиксировании двух индексов(рис.2) может быть выбран только один элемент.

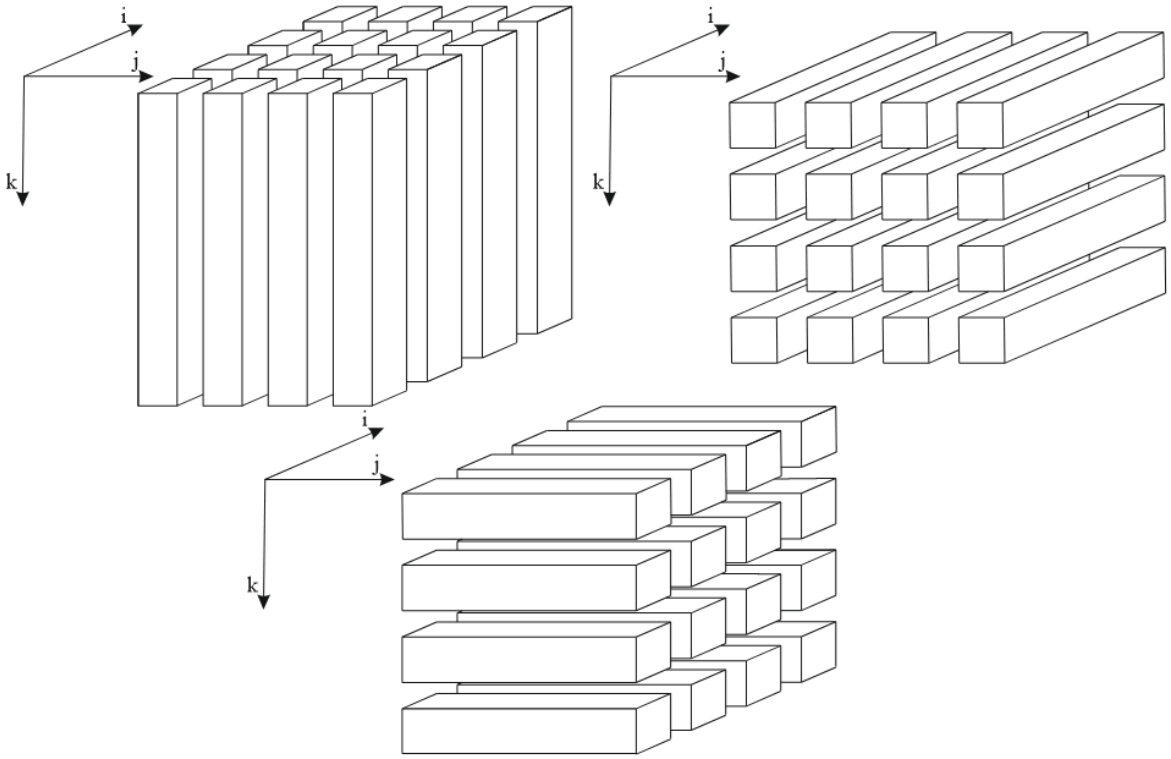


Рис. 2. Строки матрицы при фиксировании при фиксировании двух индексов

Введём кубическую матрицу переменных x размера $N \times N \times N$:

$$x_{ij}^k = \begin{cases} 1, \textit{если элемент выбран} \\ 0, \textit{иначе}, \end{cases}$$

где $i,\ j = \overline{1, N}$.

Тогда математическую модель задачи можно представить в виде:

$$L(x) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} c_{ij}^k x_{ij}^k \ \to \min,$$

$$\sum_{k=1}^{N} x_{ij}^k = 1,\ i,\ j \in \overline{1, N},$$

$$\sum_{j=1}^{N} x_{ij}^k = 1,\ i, k \in \overline{1, N},$$

$$\sum_{i=1}^{N} x_{ij}^k = 1,\ j, k \in \overline{1, N}.$$

2. Жадный алгоритм для решения трёхиндексной планарной ЗОН

Жадными называют семейство алгоритмов, основывающихся на выборе локально оптимальных вариантов: алгоритм Крускала для поиск остовного дерева минимального веса в графе, алгоритм Прима поиск остовного дерева минимального веса в связном графе и т.д. Жадные алгоритмы находят точное решение для задач, в которых сумма локальных оптимумов приводит к глобально оптимальному решению. Также применение жадных алгоритмов целесообразно при нахождении приближенного решения для задач с большой вычислительной сложностью.

2.1. Жадный алгоритм решения задачи о назначениях

Несмотря на то, что для классической задачи о назначениях известны точные алгоритмы решения, при больших размерностях затруднителен поиск точного решения, тогда становится оправданным применение приближенных алгоритмов.

Ниже представлен жадный алгоритм для её решения:

Дано:

C – квадратная матрица затрат $N \times N$.

Введем обозначения:

$J = \{1,..,N\}$ – множество доступных значений индекса j.

Алгоритм 1. Жадный алгоритм решения задачи о назначениях

Цикл. Для $i = \overline{1,N}$.

Шаг 1. Нахождение минимума при фиксации одного из индексов

$$\min_{j \in J} c_{ij} = c_{ij*}$$

$$x_{ij*} = 1$$

Шаг 2. Удаление индекса из множества доступных для выбора

$$J = J / \{j*\}$$

Конец цикла.

2.2. Жадный алгоритм решения трёхиндексной планарной задачи о назначениях

Дано:

C – кубическая матрица затрат $i \times j \times k$.

Введем обозначения:

$S_{\substack{j \\ j=\overline{1,N}}} = \{1,..,N\}$ – множества доступных значений индекса k при фиксации индекса j

$K = \{1,..,N\}$ – множество доступных значений индекса k при фиксации индекса i

$J = \{1,..,N\}$ – множество доступных значений индекса j при фиксации индекса i

Алгоритм 2. Жадный алгоритм для решения трёхиндексной планарной задачи о назначениях

Шаг 1. Положим индексы $i,\ j$ равными:

$i = 1,\ j = 1$

Цикл. Для $j = \overline{1, N}$.

Шаг 2.1. Нахождение минимума при фиксации двух индексов $i,\ j$:

$\min\limits_{k \in J} c_{ij}^{k} = c_{ij}^{k*}$

$x_{ij}^{k*} = 1$

Шаг 2.2. Удаление индекса из множества доступных для выбора

$K = K / \{k*\}$

$S_j = S_j / k*$

Конец цикла.

Шаг 3. Обновляем множество K:

$K = \{1, .., N\}$

Цикл. Для $i = \overline{2, N}$.

Цикл. Для $j \in J$.

Шаг 4.1.1. Фиксируем индекс $j = j*$:

$\min\limits_{j \in J} |S_j| = S_{j*}$

Шаг 4.1.1. Нахождение минимума при фиксации двух индексов $i,\ j$:

$\min\limits_{k \in J, k \in S_j} c_{ij}^{k} = c_{ij}^{k*}$

$x_{ij}^{k*} = 1$

Шаг 4.2. Удаляем из множества J индекс j, из множества S_j индекс $k*$, из множества K индекс $k*$:

$J = J \,/\, j$

$S_j = S_j / k*$

$K = K / k*$

Конец цикла.

Шаг 5. Обновляем множеств J и K:

$J = \{1, .., N\}$

$$K = \{1,..,N\}$$

Конец цикла.

Рассмотрим пример работы жадного алгоритма. Возьмем матрицу затрат C (Рис.3). Зелёным помечаются выбранные элементы, жёлтым - элементы недоступные для выбора. Ход работы алгоритма (Рис.4 – Рис.19) и полученное решение (Рис.20):

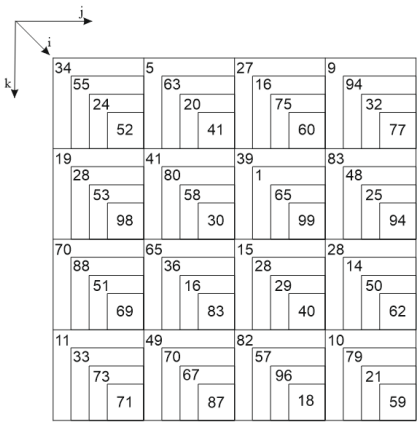


Рис.3 Кубическая матрица затрат

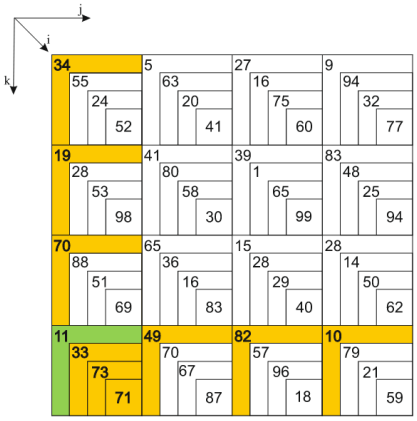


Рис.4 $i=1,\ j=1$

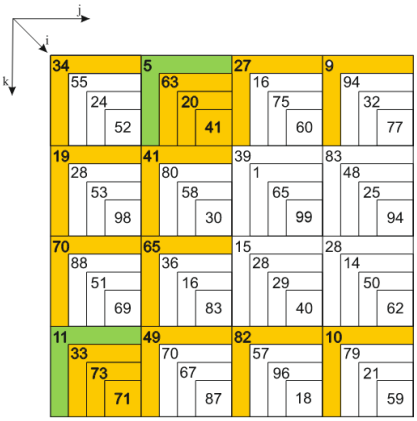


Рис.5 $i=1,\ j=2$

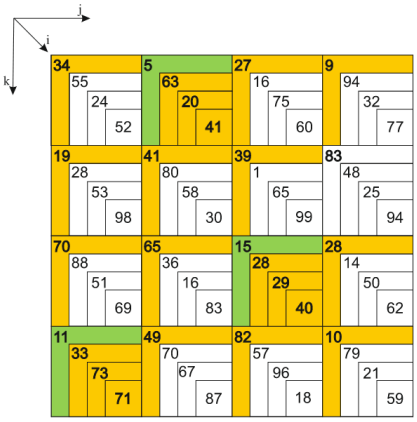


Рис.6 $i=1,\ j=3$

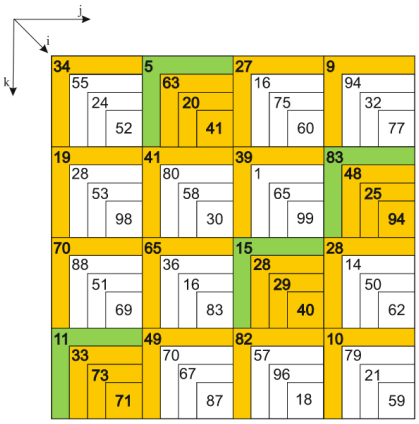


Рис.7 $i=1,\ j=4$

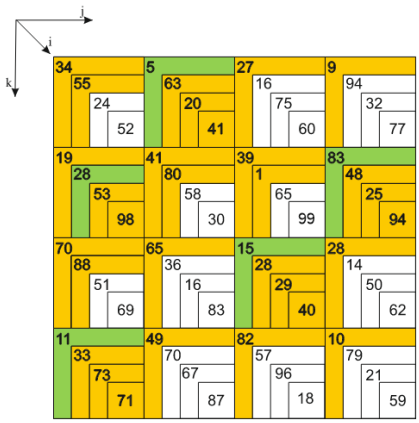


Рис.8 $i=2,\ j=1$

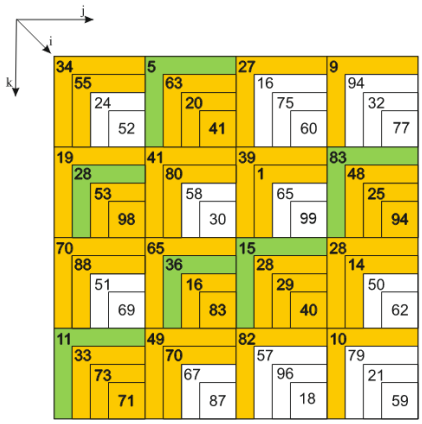


Рис.9 $i=2,\ j=2$

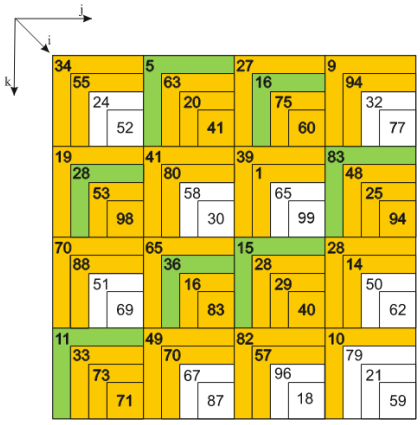


Рис. 10 $i = 2,\ j = 3$

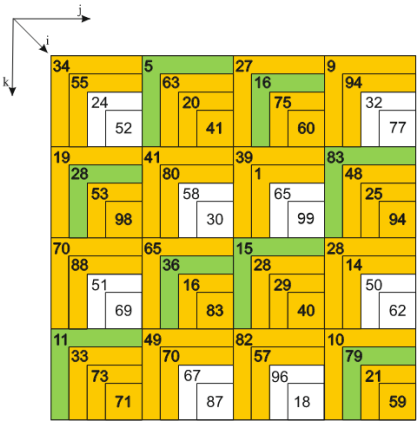


Рис. 11 $i = 2,\ j = 4$

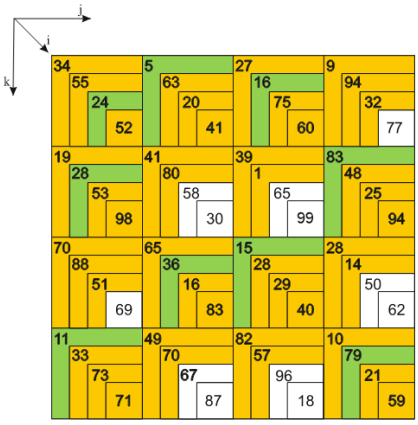


Рис. 12 $i = 3,\ j = 1$

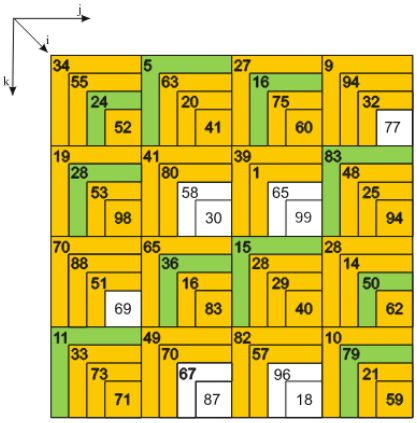


Рис. 13 $i = 3,\ j = 2$

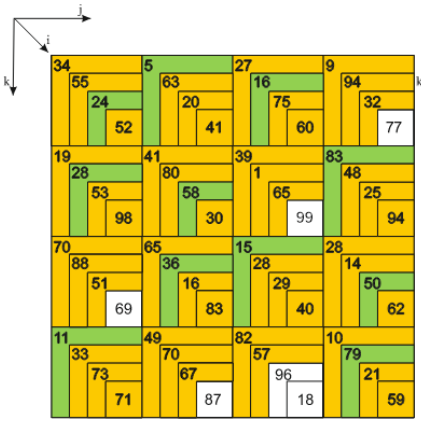


Рис. 14 $i = 3,\ j = 3$

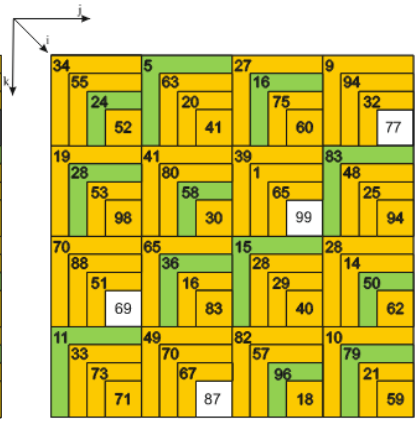


Рис. 15 $i = 3,\ j = 4$

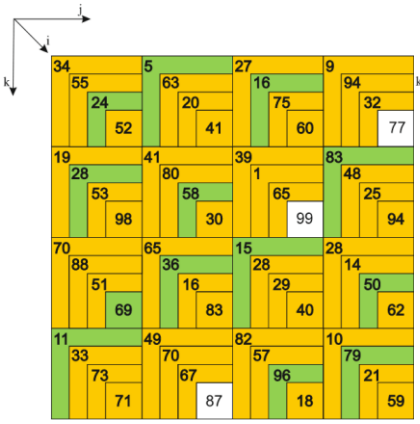


Рис. 16 $i = 4,\ j = 1$

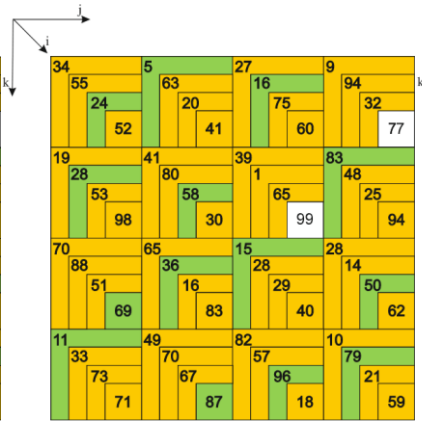


Рис. 17 $i = 4,\ j = 2$

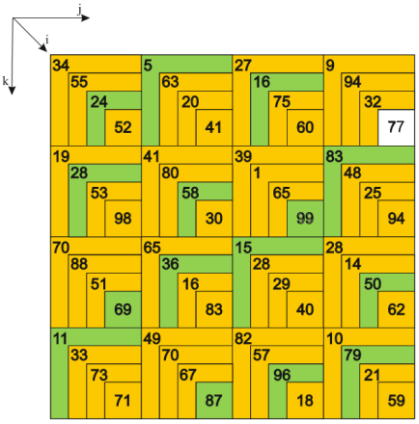


Рис. 18 $i = 4,\ j = 3$

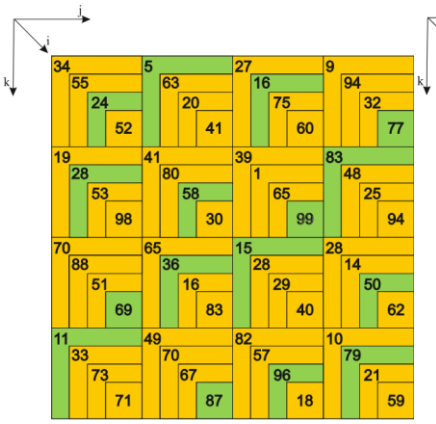


Рис. 19 $i = 4,\ j = 4$

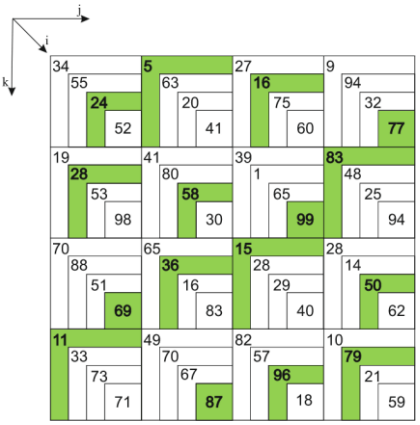


Рис. 20 $i = 1,\ j = 2$

Алгоритм не нашел оптимального решения, но нашел приближенное с суммой 833. Как можно заметить, жадный алгоритм имеет недостаток – на последних шагах остается меньше элементов для выбора, поэтому чем ближе к концу работы, тем большую ошибку он может привнести в решение.

Заключение

Был разработан жадный алгоритм для решения трёхиндексной планарной задачи о назначениях и показан ход его работы.

Литература

1. Трегубов, А. Г. Вероятностный подход к решению трехиндексной аксиальной задачи о назначениях / А. Г. Трегубов, С. Н. Медведев // Вестник Воронеж. гос. ун-та. Сер. Системный анализ и информационные технологии. – 2015. – № 4. – 12 с.

2. Гимади, Э. Х. Аксиальные трехиндексные задачи о назначениях и коммивояжера: быстрые приближенные алгоритмы и их вероятностный анализ / Э. Х. Гимади, А. И. Сердюков // Изв. Вузов. Математика. – 1999. – № 12. – С. 19–25.

УДК 004.42

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ «TASK PLANIFY»

Д. В. Любавин, Т. В. Курченкова

Воронежский государственный университет

Введение

В мире, насыщенном информацией, с учетом постоянно ускоряющегося темпа жизни, общество постоянно сталкивается с потоком новых задач и обязанностей. Будь то проекты на работе, учебные задания или личные цели, эффективное управление временем становится ключом к успеху и достижению поставленных целей. Часто ощущается нехватка времени и средств для эффективной организации задач. В поисках решения этой проблемы многие обращаются к различным методикам и инструментам, которые помогли бы справиться с этим вызовом.

В статье рассматривается разработка веб-приложения, которое призвано упростить процесс планирования и управления задачами. Это приложение предлагает интуитивно понятный интерфейс, позволяющих пользователям эффективно организовывать свое время и достигать поставленных целей.

1. Постановка задачи

Разработать веб-приложение по планированию задач. Программа должна содержать следующую функциональность:

- аутентификация пользователей;
- авторизация пользователей;
- операции для создания, получения, изменения, удаления пользователей, задач и временных блоков;
- операции для создания, получения, изменения до 5 помодоро таймеров;
- валидация данных;
- логирование исключений;
- хранение паролей в хешированном виде.

2. Логическая модель данных

В процессе разработки веб-приложения был проведён анализ предметной области, включающий изучение основных требований и потребностей пользователей. На основе этого анализа была разработана модель данных, которая отражает структуру и взаимосвязи между различными элементами системы. Эта модель данных служит основой для построения функциональности приложения и обеспечивает эффективное взаимодействие между его компонентами.

Логическая модель данных в нотации Баркера представлена на рис. 1.

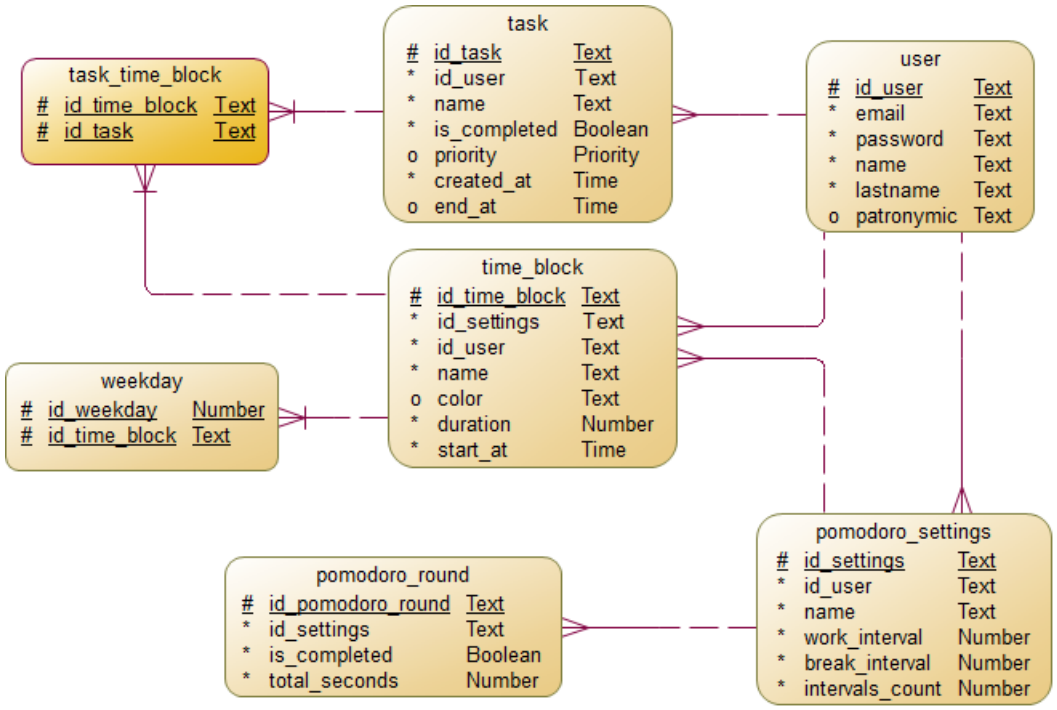


Рис. 1. Логическая модель данных

Сущность *Пользователь* (user), содержит информацию о пользователях: адрес электронной почты, пароль, ФИО пользователя.

Сущность *Задача* (task), содержит информацию о задачах пользователя: название задачи, выполнена ли задача, приоритет задачи, время создания задачи, время, когда задача должна быть выполнена.

Сущность *Временной блок* (time_block), содержит информацию о временных промежутках, в которых нужно выполнить задачи: название, цвет, промежуток времени в минутах, когда начнется.

Сущность *День недели* (weekday), содержит днях недели: номер дня недели.

Сущность *Настройка помодоро* (pomodoro_settings), содержит настройки помодоро таймера: название, время в минутах рабочего раунда, время в минутах перерыва, количество интервалов таймера.

Сущность Раунд п*омодоро* (pomodoro_round), содержит информацию о раунде помодоро таймера: завершился ли раунд работы или отдыха, количество прошедших секунд с начала раунда.

3. Общее описание приложения

Для создания серверной части приложения был использован JavaScript фреймворк NestJS [1]. Клиентская часть приложения создана при помощи JavaScript фреймворка Next.js [2]. Стили для компонентов использовались из CSS фреймворка Tailwind CSS [3].

Чтобы начать пользоваться приложением необходимо войти или зарегистрироваться. Внешний вид формы регистрации аккаунта и формы входа в аккаунт представлен на рис. 2 и на рис. 3 соответственно.

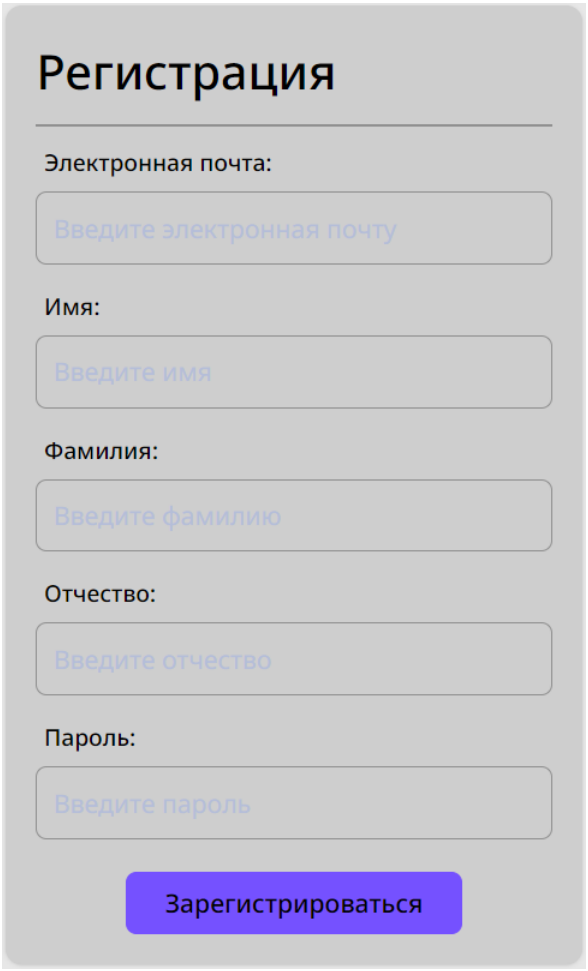


Рис. 2. Форма регистрации аккаунта

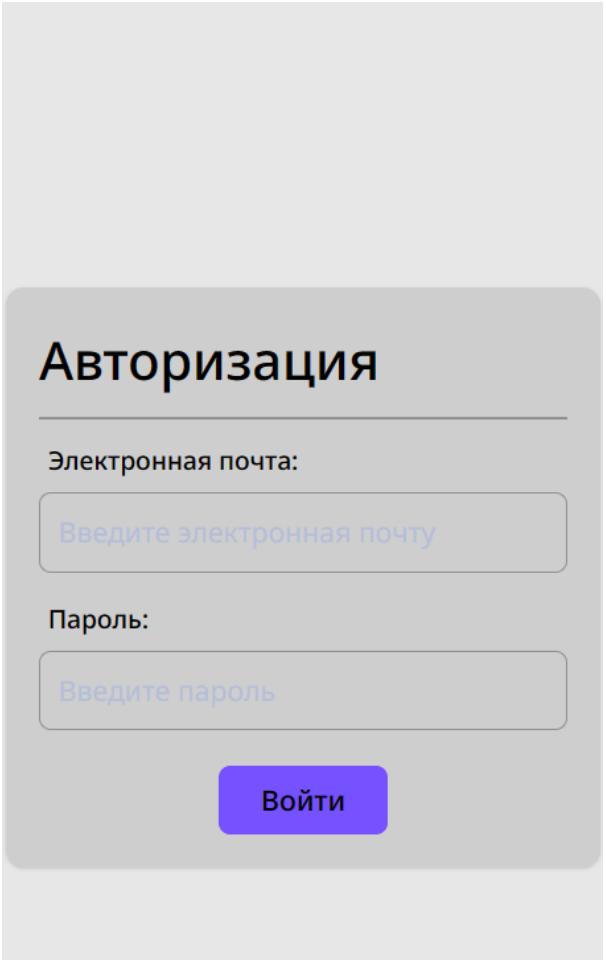


Рис. 3. Форма входа в аккаунт

После входа в аккаунт попадаем на страницу с задачами. Внешний вид страницы с задачами приложения представлен на рис. 4. Структура задач включает в себя: название, дата, когда задача должна быть выполнена, ее приоритет.

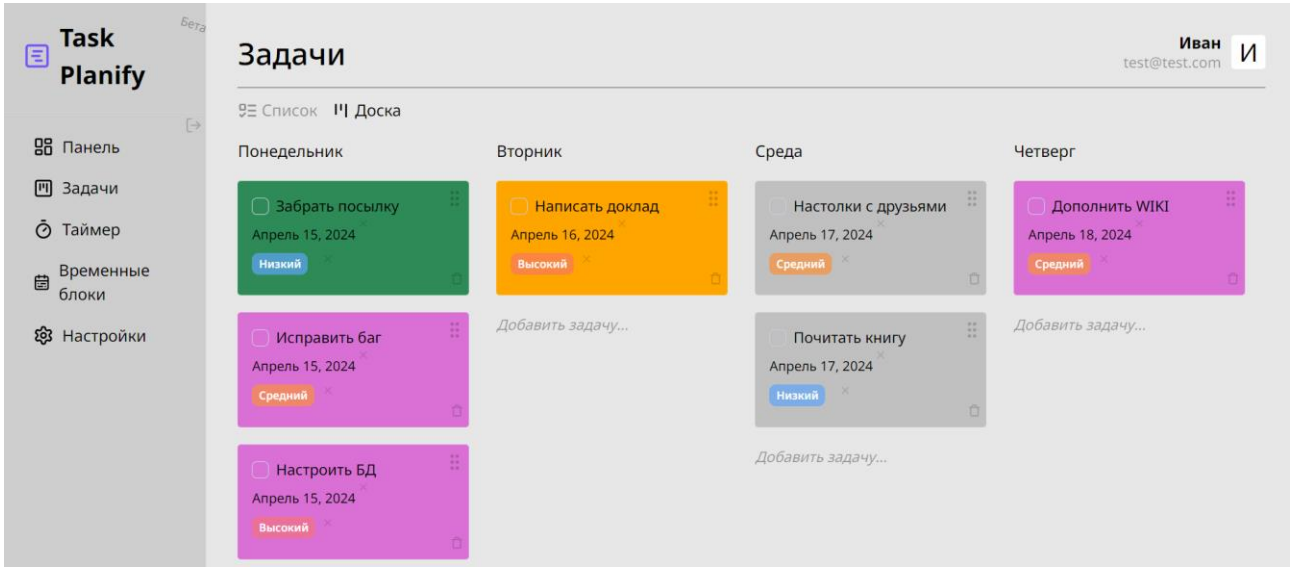


Рис. 4. Страница с задачами приложения

При нажатии на кнопку *Временные блоки* открывается страница с временными блоками, внешний вид которой представлен на рис. 5.

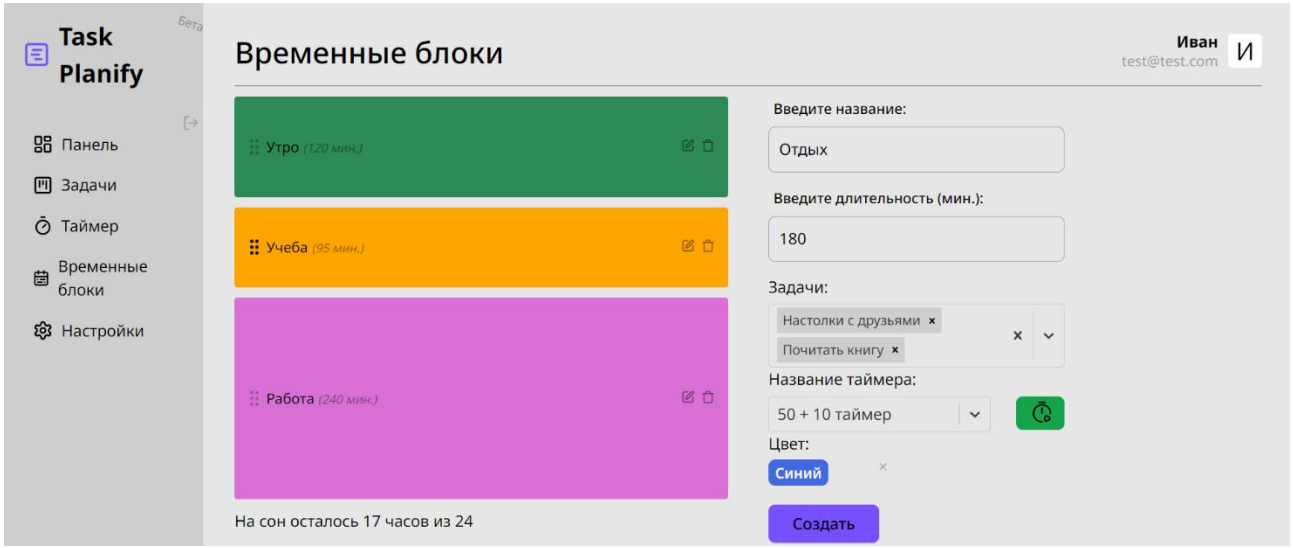


Рис. 5. Страница временных блоков

При нажатии на кнопку *Настройки* открывается страница настроек аккаунта, внешний вид которой представлен на рис. 6.

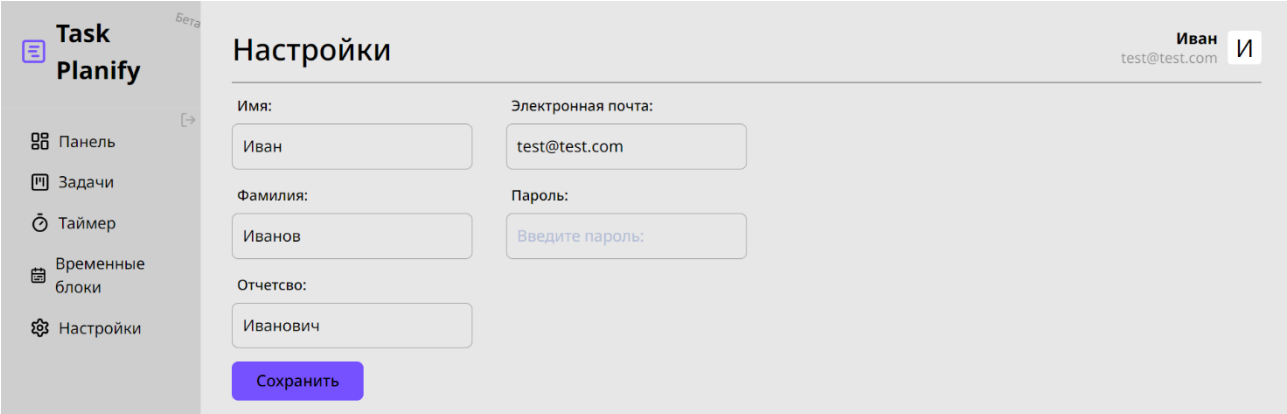


Рис. 6. Страница настроек аккаунта

Заключение

Разработанное веб-приложение представляет собой удобный и интуитивно понятный инструмент, который помогает пользователям эффективно организовывать свои задачи и достигать поставленные цели. Благодаря его востребованной функциональности и простому пользовательскому интерфейсу, пользователи могут легко создавать, отслеживать и управлять своими задачами, повышая свою производительность и улучшая качество своей жизни.

Литература

1. Nest.js Documentation. – Режим доступа: https://docs.nestjs.com/. – (Дата обращения: 19.04.2024).
2. Next.js Docs. – Режим доступа: https://nextjs.org/docs. – (Дата обращения: 19.04.2024).
3. Tailwind CSS Docs. – Режим доступа: https://tailwindcss.com/docs/installation. – (Дата обращения: 19.04.2024).

УДК 519.6

Атаки на муравьиный алгоритм

Е. М. Макарова

Воронежский государственный университет

Введение

В современном мире, где компьютерные алгоритмы играют ключевую роль в различных сферах деятельности, защита от атак и взломов становится все более актуальной задачей. Одним из популярных методов оптимизации, используемых в различных областях, является муравьиный алгоритм. Однако, как и любой другой алгоритм, он подвержен различным видам атак, направленных на его нарушение или изменение параметров для достижения нежелательных результатов.

В данной статье будут рассмотрены основные виды атак на муравьиный алгоритм и способы защиты от них. Также будут проанализированы возможные уязвимости алгоритма и его параметров, которые могут быть использованы злоумышленниками для проведения успешных атак. Цель - помочь разработчикам и исследователям понять уязвимости муравьиного алгоритма и принять меры по защите от потенциальных угроз.

1. Муравьиный алгоритм

1.1. Общая схема муравьиного алгоритма

Алгоритм муравьиной колонии предложен в 1992 году бельгийским ученым Марко Дориго.

Основная идея алгоритма - моделирование поведения муравьев, социальных насекомых, живущих большими колониями. Несмотря на достаточно простые функции, которые выполняет отдельный муравей поведениие колонии в целом оказывается достаточно разумным.

В настоящее время муравьиный алгоритм используется для решения широкого класса задач дискретной оптимизации: маршрутизация, раскраска графа, календарное планирование, оптимизация сетевых графов и другие.

Рассмотрим непосредственно общую схему муравьиного алгоритма:

0 шаг. Представить задачу в виде набора компонент и переходов между ними - взвешенный неориентированный граф.

1 шаг. Создание муравьев.

Муравей - агент. Должен иметь память(например, запоминать маршрут).

Определяется стартовая точка, куда помещается муравей.

В зависимости от задачи муравьи помещаются в одну точку, в разные с повторением, в разные без повторений.

Так же на этом этапе на дугах задается начальный уровень феромона, положительное небольшое число. Это необходимо для подсчета вероятности на следующем шаге.

2 шаг. Поиск решения.

Вероятность перехода из вершины i в j на итерации t:

$P_{ij}^t = \frac{(\tau_{ij}^t)^\alpha (\frac{1}{c_{ij}})^\beta}{\sum_{j \in J} (\tau_{ij}^t)^\alpha (\frac{1}{c_{ij}})^\beta}$, где J - доступные вершины

τ_{ij}^t - уровень феромона на ребре i - j

c_{ij} - эврестическое расстояние между i и j

α, β - некоторые константы

$\alpha = 0$ - выбор происходит только на основе расстояний, т.е. жадный алгоритм с вероятностями выбора

$\beta = 0$ - выбор на основе феромона

3 шаг. Обновление феромона.

$\tau_{ij}^{t+1} = (1-\rho)\tau_{ij}^t + \sum_{k \in A_{ij}} \frac{Q}{L_k}$,

ρ - интенсивность испарения феромона

L_k - длина пути, пройденного κ-ым муравьем

A_{ij} - множество муравьев, прошедших по ребру i - j

Q - общее количество феромона у марвьев (настраиваемы параметр)

$\frac{Q}{L_k}$ - феромон, откладываемый κ-ым муравьем на ребро i - j

4 шаг. Дополнительные действия.

Рассмотрим исполнение алгоритма локального поиска.

Этапы решения задачи:

1. Представить задачу в виде взвешенного неориентированного графа.
2. Задать пармаетры алгоритма:

- количество муравьев
- α, β
- Q; лучше выбирать в соответствии с длиной оптимального пути ($Q \approx L^*$)
- $\rho \in [0,1]$, $\rho \approx 1$ потеря решений, $\rho \approx 0$ - локальная оптимизация

3. Выбрать разновидность муравьиного алгоритма и применить
4. Настроить параметры алгоритма

Также важным является момент обновления феромона:

1. Последовательный алгоритм:

- обновление после каждого муравья
- обновление в конце итерации

2. Параллельный алгоритм:

- обновление после каждого ребра
- обновление в конце алгоритма

1.2. Задача Коммивояжера

Муравьиный алгоритм - это метаэвристический алгоритм оптимизации, вдохновленный поведением муравьев при поиске кратчайшего пути к источнику пищи. Он может быть

применен для решения задачи коммивояжера, которая заключается в нахождении оптимального маршрута, проходящего через все города (вершины) ровно один раз и возвращающегося в исходный город.

Применение муравьиного алгоритма для решения задачи коммивояжера включает следующие шаги:

1. Инициализация феромонов на ребрах графа и случайное размещение муравьев в разных городах.

2. Движение муравьев от одного города к другому, принимая решения на основе феромонов и эвристической информации.

3. Обновление феромонов на ребрах после каждого цикла движения муравьев.

4. Повторение шагов 2-3 до достижения критерия останова (например, заданного числа итераций).

Муравьиный алгоритм обладает способностью находить близкое к оптимальному решение задачи коммивояжера за сравнительно короткое время. Однако, он может потребовать тщательной настройки параметров для достижения лучших результатов.

Рассмотрим подробнее применение муравьиного алгоритма для решения задачи коммивояжера.

В данной задаче граф уже имеется и перейдет к заданию остальных оставляющих:

1) Память муравья - маршрут

2) $\frac{1}{c_{ij}} = d_{ij}$ - видимость города

3) Вероятность перехода κ-ого муравья из города i в j

$$P_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(d_{ij})^{\beta}}{\sum_{\ell \in J_k}(\tau_{ij})^{\alpha}(d_{ij})^{\beta}}, \forall j \in J_k$$

J_k - доступные вершины для k-ого муравья

4) Вероятностный переход

5) Обновление феромона

Пусть L_k - его длина, T_k - маршрут муравья k, $Q \approx L^*$, $Q = C \cdot L^*$, тогда муравей обновляет феромон следующим образом:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, (i,j) \in T_k \\ 0, (i,j) \notin T_k \end{cases}$$

ρ - испарение феромона

$\tau_{ij}^{t+1} = (1-\rho)\tau_{ij}^{t} + \Delta\tau_{ij}^{t}$, где $\Delta\tau_{ij}^{t} = \sum_{k=1}^{m}(\Delta\tau_{ij}^{k})^{t}$, m - количество муравьев.

1.3. Параметры муравьиного алгоритма и атаки в природе

Параметры муравьиного алгоритма могут включать следующие:

1. **Число муравьев (ants)**: Количество муравьев, которые будут перемещаться по графу. Большее число муравьев может способствовать более широкому исследованию пространства решений, но при этом увеличивается вычислительная сложность.

2. **Коэффициент испарения феромонов (evaporation_rate)**: Скорость испарения феромонов на ребрах графа. Этот параметр влияет на скорость сходимости алгоритма и

избегание локальных оптимумов.

3. **Коэффициенты важности феромонов и эвристики (pheromone importance, heuristic_importance)**: Веса, которые учитывают влияние феромонов и эвристической информации при принятии решения о следующем шаге муравья.

4. **Параметры для обновления феромонов (pheromone_deposit, pheromone_decay)**: Как быстро феромоны откладываются на ребрах и как они испаряются. Эти параметры влияют на интенсивность обновления феромонов.

5. **Вероятность переключения между эксплорацией и эксплойтацией (exploration_probability)**: Вероятность выбора случайного пути вместо выбора на основе феромонов и эвристики. Позволяет избежать застревания в локальных оптимумах.

6. **Количество итераций (iterations)**: Число итераций алгоритма перед остановкой. Может быть задано как фиксированное число или зависеть от других критериев останова.

Эти параметры могут быть настроены экспериментально для достижения лучших результатов в конкретной задаче. Оптимальные значения параметров могут зависеть от размера задачи, структуры графа, требуемой точности решения и других факторов.

В природе муравьиный алгоритм, который вдохновлен поведением муравьев при поиске кратчайшего пути к источнику пищи, также может подвергаться различным видам атак и воздействий.

Некоторые из возможных атак на муравьиный алгоритм в природе включают:

1. **Изменение концентрации феромонов**: Некоторые виды хищных насекомых могут изменять концентрацию феромонов на тропах, чтобы привести муравьев к ловушке или отвлечь их от источника пищи.

2. **Уничтожение феромонов**: Химические вещества или другие агрессивные методы могут быть использованы для уничтожения феромонов на тропах, что затруднит муравьям находить путь к пище.

3. **Манипуляция эвристикой**: Некоторые хищники могут создавать ложные следы или изменять окружающую среду, чтобы искажать эвристическую информацию, которую муравьи используют для принятия решений о направлении движения.

4. **Блокирование доступа к пище**: Хищные насекомые могут блокировать доступ к источнику пищи или создавать препятствия на тропах, чтобы затруднить доступ муравьев к ресурсам.

5. **Маскировка и ловушки**: Некоторые хищники могут использовать маскировку или ловушки, чтобы скрыть свое присутствие и атаковать муравьев внезапно.

В природе муравьи развивают различные стратегии и адаптации для справления с такими видами атак. В контексте вычислительных алгоритмов также можно разрабатывать методы защиты от подобных атак, например, путем усиления робастности алгоритма, добавления случайности в принятие решений или использования дополнительных проверок и балансировки параметров.

2. Формирование атак на муравьиный алгоритм

Атаки на муравьиный алгоритм могут формироваться различными способами и стратегиями, которые направлены на нарушение его эффективности или введение искажений в принятие решений.

Некоторые из основных способов формирования атак на муравьиный алгоритм включают:

1. **Искажение информации**: Атакующие могут изменять информацию о феромонах на тропах, чтобы исказить принятие решений муравьев о выборе пути. Например, они могут увеличивать или уменьшать значения феромонов на определенных тропах, чтобы привести

муравьев по невыгодному пути.

Атака «Искажение информации» на муравьиный алгоритм направлена на изменение информации о феромонах на тропах, что может привести к искажению принятия решений муравьев о выборе оптимального пути. Эта атака может существенно повлиять на работу алгоритма и привести к нежелательным результатам. Вот как атака «Искажение информации» воздействует на муравьиный алгоритм:

- **Искажение привлекательности путей**: Атакующие могут увеличивать или уменьшать значения феромонов на определенных тропах, чтобы искусственно увеличить или уменьшить привлекательность этих путей для муравьев. Это может привести к тому, что муравьи выберут неоптимальные пути из-за искаженной информации о феромонах.

- **Отклонение от оптимального пути**: Изменение информации о феромонах может привести к тому, что муравьи будут отклоняться от оптимального пути и выбирать менее эффективные варианты. Это может снизить производительность алгоритма и увеличить время нахождения оптимального решения.

- **Потеря сходимости**: Искажение информации о феромонах может вызвать потерю сходимости алгоритма, то есть возможность найти оптимальное решение. Муравьи могут застрять на невыгодных путях из-за искаженной информации, не смогут обнаружить оптимальный путь и не достигнут желаемого результата.

Для защиты от атак «Искажение информации» можно использовать методы проверки целостности данных, контроля доступа к информации о феромонах, случайные элементы в выборе пути муравьев и другие техники, которые помогут предотвратить искажение информации и обеспечить эффективную работу муравьиного алгоритма.

2. **Блокирование троп**: Атакующие могут блокировать доступ к оптимальному пути или к источнику пищи, создавая препятствия или уничтожая тропы, что затрудняет передвижение муравьев.

Атака «Блокирование троп» на муравьиный алгоритм направлена на блокирование доступа муравьев к определенным тропам или путям, чтобы искусственно изменить принятие решений и выбор оптимального пути. Вот как атака «Блокирование троп» воздействует на муравьиный алгоритм:

- **Исключение оптимальных путей**: Атакующие могут блокировать доступ муравьев к оптимальным путям, которые ведут к решению задачи. Это может привести к тому, что муравьи будут вынуждены выбирать менее эффективные пути из-за отсутствия доступа к оптимальным тропам.

- **Искажение принятия решений**: Блокирование троп может исказить информацию о доступных путях и привлекательности троп для муравьев. Это может привести к тому, что муравьи будут делать неправильные выборы из-за ограничения доступа к определенным тропам.

- **Увеличение времени поиска решения**: Блокирование троп может увеличить время, которое муравьи тратят на поиск оптимального решения. Из-за ограничения доступа к некоторым путям им придется искать альтернативные варианты, что может замедлить процесс поиска оптимального решения.

Для защиты от атак «Блокирование троп» можно использовать методы обнаружения блокирования троп, автоматическое обходное планирование для муравьев при блокировке путей, установление временных ограничений на блокировку троп и другие техники, которые помогут обеспечить эффективную работу муравьиного алгоритма даже при попытках блокирования троп.

3. **Атаки на параметры алгоритма**: Атакующие могут проводить атаки на параметры алгоритма, такие как скорость испарения феромонов, коэффициенты влияния феромона и эвристики, чтобы нарушить работу алгоритма.

Атака «Атака на параметры алгоритма» направлена на изменение параметров муравьиного алгоритма для искажения его работы и результатов. Вот как эта атака воздействует на муравьиный алгоритм:

- **Искажение принятия решений**: Атакующие могут изменить параметры алгоритма таким образом, чтобы исказить информацию о привлекательности путей для муравьев. Это может привести к тому, что муравьи будут делать неправильные выборы из-за искаженной информации о путях.

- **Ухудшение производительности**: Изменение параметров алгоритма может привести к ухудшению его производительности. Например, если увеличить коэффициент испарения феромона, то муравьи могут терять информацию о пройденных путях быстрее, что затруднит им поиск оптимального решения.

- **Изменение сходимости**: Изменение параметров алгоритма может повлиять на скорость сходимости алгоритма к оптимальному решению. Некорректные параметры могут замедлить процесс поиска оптимального решения или даже привести к тому, что алгоритм не найдет оптимальное решение вовсе.

Для защиты от атак «Атака на параметры алгоритма» можно использовать методы контроля и проверки параметров алгоритма, шифрование параметров для предотвращения их изменений, автоматическую настройку параметров в процессе работы алгоритма и другие техники, которые помогут обеспечить безопасность и эффективность работы муравьиного алгоритма.

Таким образом, для защиты от подобных атак на муравьиный алгоритм можно использовать методы усиления безопасности и робастности алгоритма, добавление проверок целостности данных, случайных элементов в принятие решений и другие техники защиты от внешних воздействий.

Заключение

Муравьиный алгоритм является мощным методом оптимизации, вдохновленным поведением муравьев в природе. Он нашел широкое применение в решении различных задач, включая известную задачу коммивояжера. Однако, как и любой другой алгоритм, муравьиный алгоритм подвержен атакам, которые могут исказить его результаты и повлиять на его эффективность.

В данной статье были рассмотрены общая схема муравьиного алгоритма, его применение к задаче коммивояжера, основные параметры алгоритма и способы их настройки, а также атаки на муравьиный алгоритм в природе и их влияние на работу алгоритма.

Формирование атак на муравьиный алгоритм требует понимания его работы и параметров. Атаки могут быть направлены на изменение информации о привлекательности путей для муравьев, ухудшение производительности алгоритма или изменение сходимости к оптимальному решению. Для защиты от таких атак необходимо использовать методы контроля параметров, шифрование информации и другие техники безопасности.

В целом, муравьиный алгоритм остается эффективным методом оптимизации, но для его успешного применения необходимо учитывать возможные уязвимости и защищать его от потенциальных атак.

Литература

1. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях, 2003, №4, с.70-75.

2. МакКоннелл Дж. Основы современных алгоритмов. — М.: Техносфера, 2004 — 368 с.

3. Thompson, Jonathan. Ant Colony Optimization. //www.orsoc.org.uk/region/regional/swords/swords.ppt
4. Barker T. and Von Haartman M. Ant Colony Optimization. //courses.washington.edu/inde510/510/Ant Colony Optimization3.ppt
5. www.cosc.brocku.ca/Offerings/3P71/Lectures/ACO.ppt

УДК 004.7

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ SIGNALR В СРАВНЕНИИ С АНАЛОГАМИ

И. С. Мартовецкий

Воронежский государственный университет

Введение

В настоящее время интерактивные веб-приложения и сервисы испытывают большой спрос на реализацию обмена данными в режиме реального времени. Сюда относится передача данных без необходимости перезагрузки страниц, уведомления о событиях немедленно после их возникновения, а также многопользовательская инфраструктура, как в чатах и онлайн-играх. Одним из инструментов, позволяющих реализовывать подобную интерактивность, является технология SignalR от Microsoft.

1. Краткий обзор SignalR

SignalR — это библиотека для ASP.NET, предназначенная для добавления функциональности обмена данными в режиме реального времени в веб-приложения. Она облегчает обмен данными между сервером и клиентами, делая возможными такие задачи, как синхронизация данных, мгновенные уведомления о изменениях на стороне сервера без задержек или необходимости обновления страницы[1].

Одной из ключевых особенностей SignalR является поддержка различных транспортных механизмов для связи между клиентом и сервером. Работать она может с такими протоколами, как WebSockets, который является современным и эффективным способом обмена данными, long polling, если браузер клиента не поддерживает WebSockets, и Server-Sent Events.

2. Протоколы обмена данными

В SignalR поддерживаются три протокола обмена данными[1].

2.1. WebSockets

WebSockets — протокол, который обеспечивает двунаправленное, полодуплексное взаимодействие между клиентом и сервером через длительное TCP-соединение. Это позволяет веб-приложениям отправлять и получать информацию мгновенно и эффективно без необходимости повторного устанавливания соединения или ожидания ответа сервера на прочие запросы.

Процесс установления WebSockets начинается с обычного HTTP-запроса, который называется рукопожатием (handshake). Во время рукопожатия клиент и сервер обмениваются заголовками, которые обновляют соединение с HTTP до протокола WebSockets. После успешного рукопожатия исходное соединение преобразуется в WebSockets-канал[2].

Протокол способен поддерживать постоянное соединение, по которому можно одновременно передавать сообщения в обе стороны. Это очень полезная особенность для приложений, которым нужны обновления в режиме реального времени, таких как игры, социальные сети или порталы для совместной работа в реальном времени.

WebSockets требует для работы сравнительно новых версий браузеров и поддержки сервером. Так Google Chrome поддерживает работу с WebSockets начиная с версии 16, Mozilla Firefox начиная с версии 11, Safari начиная с версии 7, Opera начиная с версии 12.1, Internet Explorer поддерживает работу с протоколом с ограничениями, начиная с версии 10. В случае с мобильными версиями браузеров поддержка iOS Safari начинается с версии 4.2, Firefox Mobile начиная с версии 11, Opera Mobile начиная с версии 12.1[5].

2.2. SSE

SSE (Server-Sent Events) — Технология, позволяющая серверу инициировать отправку данных клиенту, подключенному через HTTP[7]. В отличие от WebSockets, SSE поддерживает только одностороннюю связь, то есть сервер может отправлять данные клиенту, но не наоборот. Эта особенность делает SSE идеальным выбором для сценариев, где необходимо непрерывное однонаправленное вещание информации, такого как ленты новостей, обновления социальных сетей или уведомления о событиях в реальном времени.

Для установления соединения с использованием SSE клиент создает HTTP запрос к серверу, который затем остается открытым на неопределенное время. В ответ сервер может отправлять поток данных в формате текста или специально форматированных сообщений, которые транслируются через это открытое соединение. Благодаря этому, данные от сервера могут доставляться клиенту без задержки, как только они появляются, избегая необходимости постоянного переустанавливания соединения.

Технология SSE проста в реализации и хорошо поддерживается современными веб-браузерами. Сейчас ее поддерживают браузеры Opera с версии 10.6, Google Chrome, Safari версии 5 и выше[3]. Технология позволяет отправлять текстовые данные и обладает встроенными механизмами для автоматического переподключения в случае временной потери связи. Браузер автоматически пытается восстанавливать соединение, отправляя повторные запросы на сервер через определенные временные интервалы.

Одним из основных преимуществ SSE является низкая нагрузка на сервер. Поскольку соединение поддерживается исключительно для отправки данных и не занимает его в простое, сервер может эффективно обслуживать большее количество клиентов одновременно, по сравнению с методами, требующими постоянных запросов на обновление данных. Также SSE легко встраивается в существующую инфраструктуру HTTP и не требует специальной поддержки за его пределами, что упрощает задачу работы с прокси-серверами и брандмауэрами, которые иногда блокируют более сложные протоколы вроде WebSockets[8-9].

Однако SSE имеет ограничения. Это отсутствие встроенной поддержки двунаправленной связи и возможные ограничения при работе с некоторыми прокси-серверами, которые могут прервать открытое соединение из-за тайм-аута.

Для проверки этого ограничения был проведен эксперимент. Создан простой сервер, предоставляющий эндпоинт для подключения по SSE и отправляющий в заданный период времени сообщение клиенту, и простой клиент, который подключается к этому эндпоинду и получает сообщения. Также было настроено логирование обрыва соединения. Сервер был развернут в инфраструктуре Kubernetes, для чего был создан Deployment, управляющий репликами приложения, Service для доступа внутри k8s и Ingress для доступа извне. Для Ingress настройка proxy_read_timeout была выставлена на небольшое значение в 10 секунд (по умолчанию устанавливается 60 секунд). В результате, если сервер не отправлял сообщение в течение более чем 10 секунд, прокси разрывал соединение и клиент заново запрашивал установление соединения.

2.3. Long Polling

Long Polling — техника обмена данными, используемая в веб-разработке для имитации непрерывного соединения между клиентом и сервером. В отличие от традиционного опроса, при котором клиент периодически запрашивает обновления, в технике Long Polling запрос клиента остается открытым на сервере до тех пор, пока сервер не будет готов отправить данные, или не будет достигнут установленный таймер ожидания.

Когда клиент выполняет запрос к серверу с использованием Long Polling, сервер проверяет наличие новых данных, которые могут быть отправлены немедленно. Если новые данные доступны, сервер отвечает на запрос, отправляя их обратно клиенту. Если же данных для немедленной отправки нет, сервер удерживает запрос, ожидая появления новой информации. Как только ожидаемые данные появляются, сервер отправляет их клиенту, после чего клиент почти сразу инициирует новый запрос[11].

Этот метод подходит для сред, где соединение в реальном времени нужно, но использование более современных протоколов невозможно из-за ограничений браузера или сетевых настроек. Также Long Polling может эффективно применяться в случаях, когда сообщения передаются нечасто или регулярность обмена информацией не критически важна.

Однако у Long Polling есть свои недостатки, включающие задержку из-за интервала опроса и потенциальную нагрузку на сервер, который должен поддерживать большое количество открытых соединений. Это может стать проблемой при масштабировании приложения на большое количество пользователей, так как каждое соединение занимает ресурсы сервера.

2.4. Автоматический выбор протокола

SignalR поддерживает автоматический выбор протокола обмена данными. Этот процесс начинается с отправки запроса от клиента к серверу. Сервер, в свою очередь, оценивает принятый запрос и анализирует возможности клиента. Если клиент и сервер оба поддерживают работу с WebSockets, и нет никаких препятствий для этого типа соединения, то выбор обычно падает на WebSockets благодаря их эффективности и низкой задержке в общении.

В случае если применение WebSockets невозможно, SignalR рассматривает альтернативные методы. Так, может быть выбран Server-Sent Events как вариант, позволяющий отправлять данные клиенту в одностороннем порядке. Если и SSE не поддерживаются, то в игру вступает Long Polling, который обладает наибольшей совместимостью, но при этом подразумевает большую нагрузку и задержку[4].

Клиент и сервер вместе определяют, какой из протоколов будет использован в конкретном случае, и этот выбор основывается на текущем состоянии сетевого окружения и совместимости устройств. Весь процесс прозрачен для разработчиков, которым не требуется заботиться о ручной настройке протокола, что значительно упрощает процесс разработки приложений.

3. Масштабируемость

SignalR первоначально не была разработана с учетом горизонтального масштабирования. Это означает, что когда приложение развертывается на одном сервере, все работает нормально, но, когда есть необходимость использовать несколько серверов, возникает проблема поддержания согласованности и синхронизации сообщений между разными узлами.

SignalR предоставляет решения для масштабируемости приложения посредством использования обратных шлюзов (backplanes) и групповых перевозчиков (group managers). Это

позволяет приложению SignalR поддерживать состояние и распространять сообщения между клиентами, подключенными к разным серверам[12].

Примерами решения для горизонтального масштабирования SignalR являются Redis, Azure SignalR Service, SQL Server, Service Bus.

4. Архитектурные ограничения

Для использования SignalR в приложении на .NET не требуется принципиально менять структуру и архитектуру приложения. Следовательно, ограничений при использовании библиотеки не накладывается.

5. Использование SignalR с технологиями, отличными от .NET

На данный момент не существует официальной реализации сервера SignalR для платформ, отличных от .NET. Однако клиентская библиотека SignalR доступна не только для .NET, но и для JavaScript, что позволяет легко интегрировать её в любое web-приложение, независимо от используемых на сервере технологий. Для создания клиентских приложений, которые могут подключаться к серверу SignalR, можно использовать JavaScript в браузере, а также различные JavaScript-фреймворки и библиотеки, такие как Angular, React или Vue.js.

Кроме того, существуют клиентские библиотеки SignalR для нативных платформ, таких как Java для Android приложений и Swift для iOS приложений. Это означает, что разработчики мобильных приложений могут использовать SignalR для установления взаимодействия с сервером в реальном времени в своих приложениях, не ограничиваясь .NET экосистемой[14-15].

Также можно встретить сторонние библиотеки и примеры реализации клиентов для SignalR на других языках программирования, таких как Python, PHP и C++. Это расширяет спектр возможностей для интеграции SignalR в разнообразные рабочие среды, где серверные компоненты могут быть написаны на языках, отличных от .NET. Однако следует учитывать, что такие реализации могут не всегда иметь такую же полную поддержку и совместимость функционала, как оригинальные библиотеки от Microsoft[16-17].

6. Аналоги и альтернативы SignalR

Для реализации взаимодействия в реальном времени существует несколько альтернатив SignalR.

6.1. WebSockets

WebSockets — это низкоуровневое API, которое позволяет открыть полнодуплексный интерактивный канал связи между пользовательскими агентами и сервером. WebSockets предоставляют большой контроль над процессом обмена сообщениями, но также требуют более сложной ручной настройки и управления соединениями[18].

Так как низкоуровневая реализация обычно требует больше времени на разработку и тестирование, WebSockets используется при наличии высоких требований к производительности.

SignalR позволяет разработчикам не концентрироваться на ручном управлении соединениями, в то время как с WebSockets придётся иметь дело со всеми аспектами протокола

на более низком уровне.

6.2 SSE

SSE (Server-Sent Events) — технология, позволяющая серверу отправлять автоматически обновлённые данные браузеру в формате event-stream. Для использования SSE в ASP.NET без SignalR можно воспользоваться нативным HTTP-потоком ответа, который обеспечивается ASP.NET. В этом случае разработчик самостоятельно должен управлять посылкой обновлений и форматом данных. SSE работают поверх обычного HTTP-соединения и используют MIME-тип text/event-stream, благодаря которому браузер понимает, что соединение должно оставаться открытым для дальнейшей передачи данных[19].

Использование SSE напрямую требует более тщательного управления соединениями и обработки ошибок, и не предоставляет столько удобств и автоматизации, сколько SignalR. Тем не менее, для простых сценариев или когда требуется точный контроль над соединением и форматом данных, прямое использование SSE может быть хорошим решением.

6.3. Socket.IO

Socket.IO — это библиотека, позволяющая обмениваться данными между браузерами и сервером в реальном времени. Это решение используется преимущественно в Node.js-среде и также обеспечивает автоматический выбор транспортного протокола в зависимости от возможностей клиента. Однако, в отличие от SignalR, Socket.IO не поддерживает протокол SSE[20].

При интеграции в приложение на .NET могут возникнуть трудности из-за различия стека. Хоть и есть решения для использования библиотеки с языком C#, но они не дают такой же производительности, как в случае с использованием нативных технологий.

6.4. Firebase Realtime Database и Firestore

Firebase Realtime Database и Firestore (от Google) — это облачные решения для создания приложений с реальным временем без необходимости управления инфраструктурой для веб- и мобильных платформ. Они предоставляют гибкие и масштабируемые решения для синхронизации данных в режиме реального времени между клиентами и сервером. Такие решения предлагают значительные преимущества в удобстве масштабирования и поддержки, т.к. забота о базовой инфраструктуре ложится на плечи облачного провайдера.

Использование платформы Firebase может существенно повлиять на архитектуру приложения, особенно в случае .NET приложений, поскольку оно вносит изменения в то, как приложение взаимодействует с данными, аутентификацией и бизнес-логикой.

Сервисы Realtime Database и Cloud Firestore предоставляют решения NoSQL для хранения и синхронизации данных в реальном времени. Это означает, что разработчики должны проектировать структуру данных исходя из нереляционной модели, что отличается от традиционных SQL баз данных. Кроме того, взаимодействие с базой данных происходит на клиентской стороне через предоставляемые Firebase SDK, что меняет концепцию традиционного бэкенда и делает клиент более автономным[21].

7. Безопасность и автономность

В контексте безопасности SignalR подразумевает выполнение стандартных практик, таких как шифрование сообщений при помощи протоколов таких как TLS, что обеспечивает

защиту данных, передаваемых между клиентом и сервером. При интеграции с методами аутентификации и авторизации, которые предлагает платформа .NET, разработчики должны гарантировать, что доступ к функциональности SignalR контролируется на основе проверенных учетных данных пользователя[22].

8. Выбор на основе сценариев использования

Если требуется немедленная двусторонняя связь в приложении, запускаемом в среде .NET, то SignalR будет предпочтительным вариантом благодаря интеграции с ASP.NET Core и поддержке полноценного .NET клиента.

Для сценариев, в которых акцент делается на высокую конкурентность и производительность, может подойти использование чистых WebSockets без дополнительных абстракций, если команда готова уделить больше времени низкоуровневой настройке соединений.

Платформы, такие как Firebase Realtime Database и Firestore, могут привлекать тем, что предоставляют готовые решения для быстрой разработки приложений со взаимодействием в реальном времени, особенно если приложение ориентировано на мобильные устройства или если предпочтительнее облегчить нагрузку на backend-разработку.

Заключение

SignalR занимает свою уникальную нишу на рынке технологий в реальном времени, обеспечивая удобный инструментарий для разработчиков в .NET экосистеме. Несмотря на наличие альтернативных решений, SignalR продолжает оставаться актуальной технологией благодаря своей интеграции, удобству использования и масштабируемости.

Выбор технологии для обмена данными в реальном времени во многом зависит от специфических требований проекта и существующего стека технологий. SignalR — мощный и удобный вариант для проектов .NET, обеспечивающий богатый функционал и удобство в разработке. С другой стороны, существуют альтернативные технологии, каждая из которых имеет свои преимущества и может быть более подходящей в определенных условиях или средах разработки. Вывод конкретных рекомендаций требует анализа конкретного проекта, его требований, доступных ресурсов и уровня комфорта команды с различными технологиями

Литература

1. Общие сведения о ASP.NET Core SignalR – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/signalr/introduction?view=aspnetcore-8.0. – (Дата обращения: 15.04.2024).

2. WebSocket: разбираем как работает – Режим доступа: https://habr.com/ru/sandbox/171066. – (Дата обращения: 15.04.2024).

3. Создание приложений реального времени с помощью Server-Sent Events – Режим доступа: https://habr.com/ru/articles/120429. – (Дата обращения: 15.04.2024).

4. Конфигурация SignalR ASP.NET Core – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/signalr/configuration?view=aspnetcore-8.0&tabs=dotnet. – (Дата обращения: 15.04.2024).

5. Поддержка API WebSocket в браузерах: список и обходные пути – Режим доступа: https://sky.pro/wiki/html/podderzhka-api-web-socket-v-brauzerakh-spisok-i-obkhodnye-puti. – (Дата обращения: 15.04.2024).

6. SignalR vs. Socket.IO: which should you choose in 2024? – Режим доступа: https://ably.com/topic/signalr-vs-socketio#signal-r-advantages-and-disadvantages. – (Дата обращения: 15.04.2024).

7. Server-sent events – Режим доступа: https://ru.wikipedia.org/wiki/Server-sent_events – (Дата обращения: 15.04.2024).

8. Using server-sent events – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events. – (Дата обращения: 15.04.2024).

9. Server-Sent Events: пример использования – Режим доступа: https://habr.com/ru/articles/519982. – (Дата обращения: 15.04.2024).

10. Module ngx_http_proxy_module – Режим доступа: https://nginx.org/en/docs/http/ngx_http_proxy_module.html?&_ga=2.202432209.113540198.1713643696-59753647.1713643604#proxy_read_timeout. – (Дата обращения: 15.04.2024).

11. Long polling – Режим доступа: https://javascript.info/long-polling. – (Дата обращения: 15.04.2024).

12. Масштабирование приложений ASP.NET Core SignalR с помощью службы Azure SignalR – Режим доступа: https://learn.microsoft.com/ru-ru/azure/azure-signalr/signalr-concept-scale-aspnet-core. – (Дата обращения: 15.04.2024).

14. Клиент JavaScript ASP.NET Core SignalR – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/signalr/javascript-client?view=aspnetcore-8.0&tabs=visual-studio. – (Дата обращения: 15.04.2024).

15. Клиент Java ASP.NET Core SignalR – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/signalr/java-client?view=aspnetcore-8.0. – (Дата обращения: 15.04.2024).

16. Сторонняя библиотека SignalR для Python – Режим доступа: https://github.com/mandrewcito/signalrcore. – (Дата обращения: 15.04.2024).

17. Сторонняя библиотека SignalR для PHP – Режим доступа: https://github.com/alexwight/php-signalr-client. – (Дата обращения: 15.04.2024).

18. Поддержка WebSockets в ASP.NET Core – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/fundamentals/websockets?view=aspnetcore-8.0. – (Дата обращения: 15.04.2024).

19. [ASP.NET Core] Try Server-Sent Events – Режим доступа: https://dev.to/masanori_msl/aspnet-core-try-server-sent-events-5db2. – (Дата обращения: 15.04.2024).

20. Socket.IO Introduction – Режим доступа: https://socket.io/docs/v4. – (Дата обращения: 15.04.2024).

21. Introduction to Firebase in .NET – Режим доступа: https://code-maze.com/dotnet-firebase. – (Дата обращения: 15.04.2024).

22. Introduction to SignalR Security – Режим доступа: https://learn.microsoft.com/en-us/aspnet/signalr/overview/security/introduction-to-security. – (Дата обращения: 15.04.2024)

УДК 004.912

СУММАРИЗАЦИЯ ТЕКСТОВ БОЛЬШОЙ ДЛИНЫ

М.В. Масленникова

Воронежский государственный университет

Введение

На сегодняшний день объем текстовой информации в сети Интернет превышает объем, который человек способен прочитать за продолжительность нескольких десятков, а может и сотен жизней, и с каждым днем это число возрастает экспоненциально. Актуальные новости, статьи по интересам, развлекательный контент — объемы этой, уже повседневной текстовой информации не сопоставимы со временем, доступным человеку. В таких реалиях текст должен быть кратким, емким и разделенным на легко воспринимаемые абзацы. Реализация этого — одна из задач автоматического реферирования текста. Автоматическое реферирование текста позволяет сократить его, при этом сохранив всю необходимую, главную информацию.

За последний год резко возросла популярность и общедоступность моделей языкового прогнозирования, но, при этом, объем текста, с которым такие коммерческие модели способны работать, сильно ограничен. Адаптация алгоритма под реферирование больших текстов улучшит эффективность таких моделей и позволит обрабатывать тексты быстрее, с меньшей нагрузкой на сервер и с меньшей вероятностью ошибки.

1. Постановка задачи

Целью работы является определение оптимального метода сегментирования текста для его автоматической суммаризации.

Для достижения поставленной цели были поставлены следующие задачи:

1. Рассмотреть основные алгоритмы автоматической суммаризации текста и проанализировать их;
2. Собрать корпус текстов для проведения исследования;
3. Разработать программу, суммаризирующую текст на основе его начала или конца;
4. Провести сравнительный анализ полученных результатов с известными.

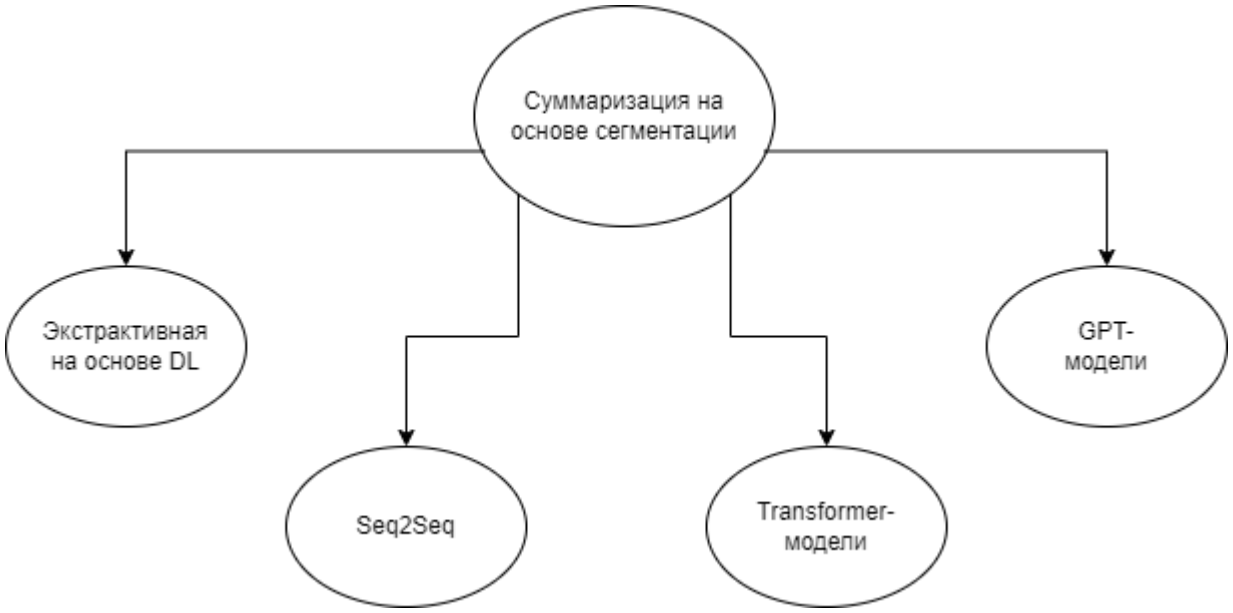


Рисунок 1. Задачи работы.

2. Основные виды алгоритмов автоматической суммаризации текста

2.1. Экстрактивные алгоритмы

2.1.1. Экстрактивная суммаризация на основе векторных представлений

1. Предобработка текста: разделение текста на предложения, предложений на токены; лемматизация токенов.
2. Нахождение векторного представления для каждого слова (модель Word2Vec).
3. Нахождение векторного представления для каждого предложения путем нахождения отношения суммы векторов каждого слова к длине предложения.
4. Построение графа на основе матрицы схожести. Устройство графа аналогично устройству графа в п.1.1.1.
5. Ранжирование предложений по значимости и сортировка их в порядке, соответствующем исходному тексту.

2.1.2. Экстрактивная суммаризация на основе глубокого обучения

Алгоритм извлекающей суммаризации на основе глубокого обучения [6, с. 28]:

1. Предобработка текста: разделение текста на предложения, предложений на токены; лемматизация токенов.
2. Построение словаря уникальных токенов, с присвоением каждому из них уникального индекса.
3. Присвоение вектора каждому токену в словаре, при этом каждый вектора имеет малую размерность относительно самого словаря.
4. Преобразование последовательности векторов yi токенов xi.
5. Вычисление представления токенов xi с учетом вектора yi всех токенов на i-1.
6. Использование архитектур Seq2Seq для генерации выходной последовательности.

В комментарий к методам извлекающей суммаризации можно сказать, что опора только на токенизацию и лемматизацию текста при его компьютерной обработке может вызвать неточности или ошибки в итоговом результате.

Серьезным недостатком подходов 2.1.1. и 2.1.2. является их неспособность различения омографов (классический пример из азбуки: “ключ” как отмычка для двери и “ключ” как родник, источник воды). В зависимости от исполнения программа может счесть их за одну и ту же единицу или счесть инстанции использования одной и той же единицы как двух разных. Также к проблемам опоры на токенизацию и лемматизацию стоит отнести проблему выявления лексем, состоящих из двух и более словоформ. К таким лексемам можно отнести фразовые глаголы английского языка или некоторые будущие формы глаголов в русском языке.

Еще одной проблемой метода можно выделить опору на отдельные слова, а не на общую семантику предложения в выборе фраз и предложений для итогового реферата. В таком случае предложения в полученном тексте могут быть мало связаны между собой и требовать дальнейшей обработки.

2.2. Абстрактивные алгоритмы

Абстрактивные методы суммаризации основаны на образовании нового текста, являющегося кратким содержанием исходного. Согласно [7], “абстрактные методы генерируют текст краткими фразами, которые семантически согласуются с изначальным документом и содержат его важнейшую информацию. Результаты работы таких решений похожи на то, как люди пересказывают тексты…”

2.2.1. Модель Sequence to Sequence (Seq2Seq)

Модель *Sequence to sequence* — модель глубокого обучения, принимающая на вход одну последовательность элементов, и возвращающая на выходе другую. Модели Seq2Seq пользуются большой популярностью в сфере машинного перевода, являясь частью таких крупных автоматических переводчиков как Google Translate.

Внутри модели энкодер и декодер. “Энкодер обрабатывает каждый элемент входной последовательности, переводит полученную информацию в вектор, называемый контекстом. После обработки всей входной последовательности энкодер пересылает контекст декодеру, который затем начинает генерировать выходную последовательность элемент за элементом.” [9]. Слово, как элемент входной последовательности, должно быть представлено вектором через эмбеддинг-алгоритм.

Эмбеддинг — представление текста, в котором слова, имеющие одинаковое значение, имеют сходное векторное представление [11]. Эмбеддинги содержат в себе семантическую информацию о слове.

Рекуррентные нейронные сети последовательны в своей работе, обрабатывая предложения слева направо или справа налево в зависимости от языковой модели. “Считывая по одному слову за раз, RNN приходится делать несколько шагов, чтобы принять решения насчет слов, расположенных друг от друга на расстоянии. В примере [I arrived at the bank after

crossing the river] RNN определит значение bank как отмель, только после считывания каждого слова между bank и river шаг за шагом, последовательно. … Чем больше таких шагов требует принятие решения, тем тяжелее обучить RNN принятию таких решений” [13].

Несмотря на то, что вышеописанная проблема разрешения неоднозначности рассмотрена на примере из сферы машинного перевода, сложность обучения модели разрешению неоднозначности на больших расстояниях все еще актуальна, что особенно остро для языков вроде русского, где синтаксис имеет свойство нагружаться придаточными предложениями.

2.2.2. Transformer

Архитектура Transformer была представлена исследователями Google в 2017 году как модель, решающая проблемы и недостатки RNN и CNN. В отличие от них, трансформер-модель обрабатывает все элементы в цепочке параллельно, а не последовательно.

Архитектуры Transformer особенно популярны в сфере машинного перевода, они используются в таких крупных системах перевода как Яндекс.Переводчик и Google Translate.

В основе Transformer лежит механизм attention, описанный в [Vaswani et al, 2017]. Attention моделирует связь всех слов в предложении вне зависимости от их расположения путем конкатенации параллельных векторов attention каждого слова в предложении. Полученный вектор затем уменьшается через линейное преобразование и на выходе остается векторное представление слова предложения из векторных представлений всего остального предложения. Такой процесс может наслаиваться сам на себя, используя результаты прошлой итерации в последующей. Так Transformer модель выстраивает эмбеддинг каждого слова.

Декодер Transformer-а функционирует по схожему алгоритму, но генерирует по одному слову за раз. Генерация слова учитывает как уже сгенерированные слова предложения, так и векторы attention, полученные при энкодинге.

Архитектура Transformer на данный момент лежит в основе всех перспективных программных продуктов, связанных не только с суммаризацией текста, но и с его генерацией и машинным переводом. Примером тому являются нашумевшие GPT-модели.

Заключение

На данном этапе исследовательской работы были изучены основные методы автоматической суммаризации текстов, структура Transformer-моделей, применяемых для автоматического реферирования.

В дальнейших перспективах исследования выбрать жанр текстов для анализа и составить корпус для проведения исследования, составить программу, реферирующую текст на основе его начала или конца, используя разные алгоритмы суммаризации, а также провести сравнительный анализ полученных результатов.

Литература

1. Суммаризация текста: подходы, алгоритмы, рекомендации и перспективы // Хабр URL: https://habr.com/ru/articles/514540/ (дата обращения: 20.10.2023).

2. Извлекаем суть новости. Опыт Яндекса // Хабр URL: https://habr.com/ru/companies/yandex/articles/586634/ (дата обращения: 28.11.2023).

3. Постановка задачи автоматического реферирования и методы без учителя // Хабр URL: https://habr.com/ru/articles/595517/ (дата обращения: 28.11.2023).

4. Извлекающие методы автоматического реферирования // Хабр URL: https://habr.com/ru/articles/595597/ (дата обращения: 28.11.2023).

5. Секреты генерирующего реферирования текстов // Хабр URL: https://habr.com/ru/articles/596481/ (дата обращения: 28.11.2023).

6. Егунова А. И., Комаров Р. С., Вечканова Ю. С., Егунова О. И., Сидоров Д. П., Шибайкин С. Д., Никулин В. В. Анализ алгоритмов и решений для автоматической генерации подводок новостных статей в соцсетях с использованием искусственного интеллекта // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. 2023. №1.

7. Белякова А.Ю., Беляков Ю.Д. Обзор задачи автоматической суммаризации текста // ИВД. 2020. №10 (70).

8. Вашкевич Е.К. Токенизация в NLP // 56-я Научная Конференция Аспирантов, Магистрантов и Студентов БГУИР. - Минск: Белорусский государственный университет информатики и радиоэлектроники, 2020. - С. 67-68.

9. Визуализируя нейронный машинный перевод (seq2seq модели с механизмом внимания) // Хабр URL: https://habr.com/ru/articles/486158/ (дата обращения: 19.11.2023).

10. Как работает ChatGPT: объясняем на простом русском эволюцию языковых моделей с T9 до чуда // Хабр URL: https://habr.com/ru/companies/ods/articles/716918/ (дата обращения: 19.11.2023).

11. What Are Word Embeddings for Text? // Machine Learning Mastery URL: https://machinelearningmastery.com/what-are-word-embeddings/ (дата обращения: 19.11.2023).

12. Transformer — новая архитектура нейросетей для работы с последовательностями // Хабр URL: https://habr.com/ru/articles/341240/ (дата обращения: 17.01.2024).

13. Transformer: A Novel Neural Network Architecture for Language Understanding // Google Research URL: https://blog.research.google/2017/08/transformer-novel-neural-network.html (дата обращения: 17.01.2024).

14. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need // Advances in neural information processing systems. - 2017. - С. 30-45.

15. Ахметов И. Разработка метода для информативного экстрактивного реферирования научных текстов на английском языке: дис. д-р. филос. наук: 8D06101. - Алматы, 2022. - 147 с.

УДК 004.9

КОНЦЕПЦИЯ ОСНОВНЫХ ФУНКЦИЙ СИСТЕМЫ УПРАВЛЕНИЯ ОТНОШЕНИЯМИ С КЛИЕНТАМИ ДЛЯ ОНЛАЙН-ШКОЛ

К. Л. Матюшевский

Воронежский государственный университет

Введение

Система управления взаимоотношениями с клиентами (CRM)— прикладное программное обеспечение для компаний, предназначенное для автоматизации стратегий взаимодействия с клиентами, в частности, для контроля и повышения уровня продаж.

В отчете компании Forrester [1] аналитик Уильям Бэнд опросил около 150 различных компаний, пытаясь выявить проблемы использования CRM-систем. Он обнаружил, что 18% его респондентов столкнулись с проблемами, непосредственно связанными с неадекватными стратегиями внедрения CRM-решений.

На экономические результат внедрения CRM влияют такие факторы, как специализированная на определенном бизнесе функциональность, а также, обучение и понимание ее персоналом. Поэтому, работоспособная CRM-стратегия должна отвечать принципам SMART, а именно: быть конкретной, измеримой, достижимой, значимой и ограниченной во времени. Также стратегию следует связать с наиболее важными бизнес целями компании. CRM не является универсальным решением, а лишь помогает компании повышать прозрачность в управлении бизнесом. В этом случае сама система становится неотъемлемой частью дальнейшего развития компании [2].

Внедрение CRM позволяет компаниям улучшить работу с клиентами на разных этапах. При этом, упрощается и ускоряется процессы обслуживания клиентов, и как следствие - продаж. Благодаря использованию CRM, компании могут более оперативно взаимодействовать с клиентами, получая и предоставляя им нужную информацию и решать текущие вопросы в процесс формирования продажи или поддержки. Напротив, отсутствие CRM негативно влияет на бизнес показатели компании.

По статистике, компания без CRM теряет от 20 до 50% обращений, поступающих с сайта, почты, телефона. Это звонки в нерабочее время и выходные, забывчивость менеджеров, потери на передаче информации. А временные затраты персонала на выполнение действий, необходимых для работы с клиентом, при внедрении CRM-системы сокращаются, в среднем, на 20-30% за счет встроенных шаблонов документов, хранения истории, работы с задачами [3]

Применение CRM-системы обеспечивает комплексный подход к управлению клиентскими отношениями, позволяя более эффективно анализировать и использовать полученные данные о клиентах для принятия более обоснованных решений и формирования персонализированных предложений. В результате, улучшается уровень удовлетворенности клиентов и, как следствие, повышается лояльность к бренду. Внедрение CRM-системы способствует улучшению коммуникации на всех уровнях организации и повышает ее конкурентоспособность на рынке, что в свою очередь приводит к росту продаж и увеличению прибыли [4].

Но не каждая CRM подойдет любому бизнесу. Крупным компаниям нужны такие функции как улучшение взаимодействия между отделами продаж, маркетинга, финансов и выявление новых возможностей и направлений развития. Организациям поменьше, не нужны

эти функции, для них они избыточны, усложняют функциональность, которая должна быть нацелена именно на улучшение процесса принятия решений, обеспечение единой и актуальной базы данных о клиентах пришедших из разных источников, ускорение процессов обработки клиентов и повышение конверсии из заявки в конечную продажу. В статье предполагается рассмотреть путь реализации вышеуказанных функций для онлайн-школ, к которым относятся онлайн-школы иностранных языков, курсов программирования, психологов и онлайн-консультантов, то есть любых образовательных платформ, где есть воронка продаж и после первичной продажи будут продолжаться однотипные вопросы, действия и покупки услуг со стороны ученика.

1. *Постановка задачи*

Реализацию CRM-системы для онлайн-школ следует тесно рассматривать с основным набором функций, которые необходимы именно для такого формата образовательной платформы начиная от привлечения заказчиков (учеников, клиентов, слушателей) и заканчивая начислением вознаграждения и премий для исполнителей (репетиторов, консультантов, коучей). Ни одна из существующих популярных CRM полностью не удовлетворяет узким потребностям онлайн-школ: ни операционные CRM, где в основном акцент делается на работе с клиентской базой и этап взаимодействия с заказчиками, ни маркетинговые, где хранится информация не только о продажах, но и о рекламных кампаниях (данные по акциям, промокодам, программам лояльности, бонусам и скидкам). Поэтому, стоит задача разработки и реализации CRM, сочетающими все необходимые функции и, простоту использования.

2. *Основные функции CRM для онлайн-школ*

Первоочередной задачей для онлайн-школы является привлечение заказчиков (учеников, клиентов, слушателей) по оптимальной, но при этом рентабельной цене. Заказчики получают доступ к информации об онлайн-школе через поисковые системы или от разнородных рекламных систем. В CRM по каждой заявке в профиле должна сохраняться и накапливаться информацию об источнике трафика, и предоставляться аналитические данные стоимости клиента по каждому из источников трафика, длительности обучения ученика из данного источника, среднему доходу на ученика с данного рекламного канала. Эти данные невозможно получить из системы ЯндексМетрика.

Информацию о каждой полученной заявке (потенциальном ученике) или уже обучающемся ученике необходимо редактировать, обновлять, прикреплять к ней исполнителей. Это должна быть отдельная страница/вкладка, чтобы менеджер в удобной, компактной форме имел возможность понять текущий статус заявки. Сюда можно добавить и информацию по попыткам связи менеджера с учеником для системы аналитики и контроля уже самих менеджеров, оперативности и четкости их работы. На эту же вкладку следует вынести и информацию о последних полученных уроках ученика с заказчиком. Эта информация крайне важна для менеджера и ее также невозможно получить из типовых CRM.

Распределение ролей в CRM для онлайн-школ (рис. 1) достаточно ограничить ролями владельца, администратора, ученика и репетитора. Внутреннее наполнение и функциональность кабинетов будет отличаться.

На отдельной странице или вкладке владелец и менеджер должны иметь возможность видеть полный список заявок и учеников с акцентом на статусы обработки, чтобы своевременно реагировать на возникающие проблемы или состояния пользователя, например, прошедшее тестовое занятие или нулевой баланс.

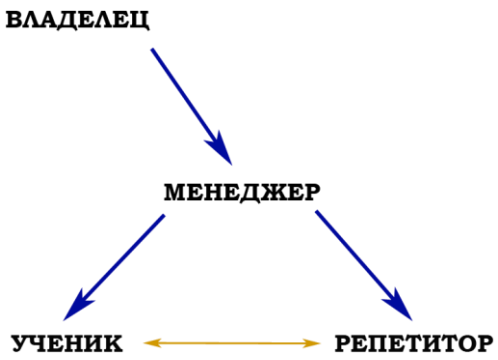


Рис.1. Роли, схема контроля, доступа к аналитике и взаимодействия в CRM-системе

Внутри профиля ученика помимо вышеуказанных функций, необходимо создать систему экспресс сообщений через сайт или приложение и один или несколько популярных мессенджеров. Данные функций есть во множестве CRM, но расфокусировать внимание менеджеров на отдельные функции в разных системах нельзя, ведь необходима именно сквозная аналитика и поддержка взаимоотношений.

Важной функцией, которая необходима как владельцу, так и менеджерам, является контроль работы исполнителей и анализ их эффективности. В разных тематиках онлайн школ данные метрики могут быть разными, но несомненно и менеджер, и владелец должны иметь возможность делать выводы и эффективности и полезности каждого отдельного исполнителя, что на прямую будет влиять на выручку и чистую прибыль организации.

Владелец онлайн-школы по данным, предоставленным CRM будет видеть конверсию работы менеджеров, что позволит привязать доход или премию к данной метрике.

Что касается маркетинговых акций (промокоды, программы лояльности), то CRM должна предоставлять аналитику по окупаемости ученика, пришедшего из каждого рекламного канала, что позволит владельцу понимать окупаемость рекламы и лучше взаимодействовать со специалистами по рекламе.

Заключение

Результат предложенной концепции реализации CRM для онлайн школ может стать отправной точкой для онлайн-школ разной тематики, со своими специфическими бизнес особенностями. Несомненно, описанную функциональность можно дополнить и другими функциями, и взаимоотношениями ролей пользователей. При этом, рекомендуется реализовать именно свой вариант CRM на одном из популярных языков программирования, например, Java или PHP, а не пытаться интегрировать несколько разнородных систем, собирая по кусочкам похожую функциональность.

Литература

1. Интернет-ресурс: Forrester [Электронный источник] – Режим доступа: http://blogs. forrester.com/william_band/12–03–01-dont_let_ crm_pitfalls_trip_you_up (Дата обращения: 11.02.2024)

2. Алексеев К.Н. Инженерный подход. Управление маркетингом, основанном на данных // Издательские решения. – 2019. С. 43–51

3. Интернет-ресурс: Amber-soft [Электронный источник] – Режим доступа: https://amber-soft.ru/products/amber-crm / (Дата обращения: 09.04.2024)

4. Ребрин, М. С. CRM-система как инструмент повышения эффективности бизнеса / М. С. Ребрин. — Текст : непосредственный // Молодой ученый. — 2023. — № 22 (469). — С. 211-216. — URL: https://moluch.ru/archive/469/103481/ (дата обращения: 10.04.2024).

УДК 004

ПРОГНОЗИРОВАНИЕ ДЕПРЕССИИ С ПОМОЩЬЮ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Ю. А. Мацнева

Воронежский государственный университет

Введение

С развитием технологий, когда носимые устройства становятся неотъемлемой частью повседневной жизни, а сбор данных становится более доступным и точным, анализ физической активности становиться актуальной задачей в области медицины. Проблемы психического здоровья связаны с нарушениями во внутренних биологических системах. Это сложные системы, и, поскольку связи между данными двигательной активности и настроением еще недостаточно хорошо изучены, изменения в этих системах трудно обнаружить. Депрессия и биполярное расстройство - это эпизодические расстройства настроения, при которых патологическое состояние и здоровое состояние могут пониматься как различные стабильные состояния, разделенные внезапными изменениями. Кроме того, исследования [1] показывают, что состояние депрессии связанно со снижением двигательной активности в дневное время, а также с повышением активности в ночное время по сравнению со здоровыми людьми. Снижение двигательной активности отмечается и при биполярных депрессиях, а также отмечается повышенная изменчивость уровней активности. С помощью методов машинного обучения можно улучшить понимание влияния физической активности на здоровье и создать инновационные подходы в диагностике психических расстройств.

Целью данного исследования является разработка эффективной модели прогнозирования наличия депрессии у пациентов на основе данных активности, с использованием методов машинного обучения. В данном исследовании используется большой набор записей двигательной активности 23 пациентов с униполярной и биполярной депрессией для предсказания наличия заболевания. На основе этих данных обучено несколько различных моделей, таких как рекуррентная нейронная сеть, сверточная нейронная сеть, гибридная нейронная сеть и случайный лес.

1. Исходные данные

В данном исследовании используется набор The Depression Dataset [5] взятый с сайта Kaggle – площадки для организации конкурсов по исследованию данных. В качестве входных данных была использована информация о 23 пациентов с униполярной и биполярной депрессией. Для каждого пациента представлены сенсорные данные за несколько дней непрерывных измерений. Пять пациентов в период сбора данных находились на стационарном лечении, 18 - на амбулаторном. Степень тяжести текущей депрессии оценивалась врачом по шкале оценки депрессии Монтгомери-Асберга в начале и в конце записи двигательной активности. Кроме того, в набор данных включены записи двигательной активности 32 человек, не страдающих депрессией (контрольная группа), среди которых 23 сотрудника больницы, 5 студентов и 4 бывших пациента без текущей психиатрической симптоматики. Все файлы представлены в формате csv. Пример информации о пациентах собранной в файле "scores.csv"-приведен на рис. 1.

.

	number	days	gender	age	afftype	melanch	inpatient	edu	marriage	work	madrs1	madrs2
0	condition_1	11	2	35-39	2.0	2.0	2.0	6-10	1.0	2.0	19.0	19.0
1	condition_2	18	2	40-44	1.0	2.0	2.0	6-10	2.0	2.0	24.0	11.0
2	condition_3	13	1	45-49	2.0	2.0	2.0	6-10	2.0	2.0	24.0	25.0
3	condition_4	13	2	25-29	2.0	2.0	2.0	11-15	1.0	1.0	20.0	16.0
4	condition_5	13	2	50-54	2.0	2.0	2.0	11-15	2.0	2.0	26.0	26.0

Рис. 1. Общий вид выборки данных файла scores.csv

2. Разведочный анализ

Данные из файлов с активность пациентов были приведены к виду, представленному на рис. 2. для возможности дальнейшей работы с ними в качестве входных значений моделей.

```
[  73    0    0 ... 1002   57  111]
[231  15   0 ...   0   0 620]
[  0 293 479 ... 184 184 184]
[347 347 347 ... 151 283  86]
[  8 121  66 ...   0   0   0]
[   0    0    0 ...  334 1558  582]
[ 83 239 111 ...   0   0   3]
[ 80 146 151 ...  17  45   5]
[  18   32   17 ...  131 1290  971]
```

Рис. 2. Пример данных

На рис. 3 приведен пример данных, построенных для одного пациента в течение 24 часов. Анализируя активность групп таким образом можно выявить любые закономерности или аномалии.

Рис. 3. Данные за день для первого пациента

На рис. 4. представлена средняя активность для пациентов обоих групп. Можно сделать вывод, что средняя активность пациентов, страдающих депрессией, ниже в течении дня, однако в ночное время она немного превосходит активность контрольной группы. Эти данные соответствуют теоретическим сведениям, приводимыми в исследовании [1], и являются ожидаемыми.

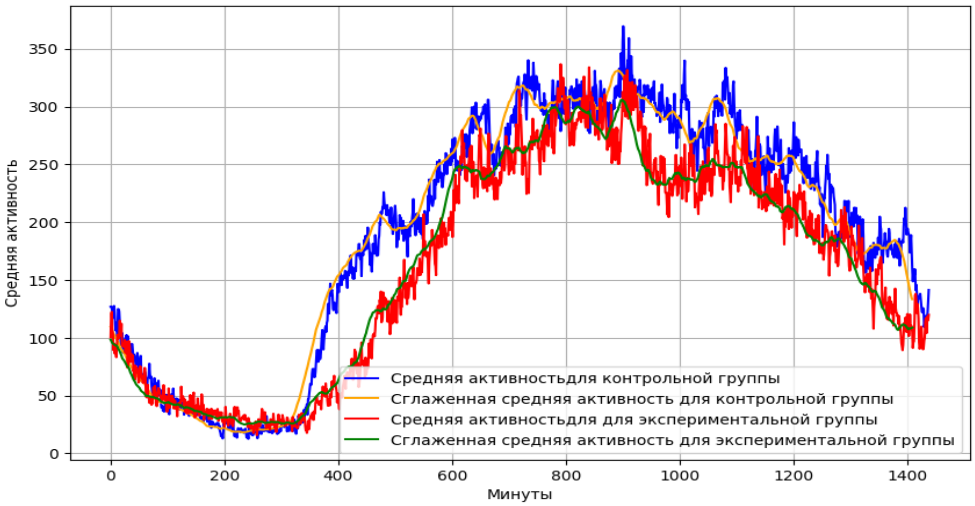


Рис. 4. Средняя активность для двух групп

3. Материалы и методы

В качестве образцовой модели был взят случайный лес. Случайные леса широко используются в практике машинного обучения для задач классификации из-за своей эффективности и способности обрабатывать сложные датасеты. Модель работает следующим образом:

1. Создаются случайные подвыборки с возвращением из обучающего набора данных. Это означает, что одни и те же данные могут входить в подвыборку несколько раз, а другие вообще не попасть.
2. Для каждой выборки строится отдельное дерево решений. Каждое дерево обучается на своей уникальной подвыборке данных. При построении каждого узла дерева случайным образом выбирается подмножество признаков. Это способствует разнообразию деревьев и уменьшает корреляцию между ними.
3. Когда требуется классифицировать новый объект, каждое дерево в лесу голосует за предполагаемый класс. Класс, получивший максимальное количество голосов, считается окончательным предсказанием случайного леса.

На основе полученных результатов можно будет оценить эффективность нейронных сетей, чья точность должна быть выше за счет более сложной системы.

Нейронные сети обладают способностью автоматически извлекать сложные зависимости и паттерны из данных. В контексте анализа активности, где могут присутствовать сложные взаимосвязи между различными видами двигательной активности, нейронные сети могут быть эффективны для выявления этих зависимостей. Также они отлично работают с большими объемами данных и имеют достаточно гибкую архитектуру. Также, с учетом быстрого развития области машинного обучения, появляется потенциал для создания новых, более эффективных моделей, способных обеспечивать высокую точность прогнозирования.

Сверточные нейронные сети (CNN) изначально разработаны для обработки изображений, но они также показали выдающиеся результаты в обработке последовательных данных [2]. В частности, одномерные сверточные слои могут извлекать пространственные шаблоны из последовательных данных, что делает их применимыми к задачам анализа временных рядов. Сверточная нейронная сеть (CNN) состоит из входного слоя, по крайней мере, одного сверточного слоя (convolutional layer), за которым следуют активационные (нелинейные) и слои пулинга (pooling layers), а также по крайней мере один полносвязанный слой. Для сверточного слоя локальная область предыдущего слоя связывается с ядром свертки

для автоматического изучения локальных и краткосрочных признаков. Операция свертки функционирует как извлекатель признаков, совмещенный с активационным слоем. Карты признаков должны быть свернуты перед прохождением через один или несколько слоев с полной связью. Наконец, слой softmax вычисляет вероятность каждого класса.

Рекуррентные нейронные сети (RNN) являются мощным классом архитектур глубокого обучения, способных работать с последовательными данных. Ключевая особенность RNN - наличие циклических соединений, позволяющих передавать информацию от одного временного шага к следующему. Обучение RNN происходит с использованием метода обратного распространения ошибки. LSTM — тип рекуррентной нейронной сети, способный обучаться долгосрочным зависимостям в данных. Проектирование LSTM использует вентили для описания временной корреляции между текущей и накопленной информацией. Модель LSTM аналогична архитектуре CNN: Входные данные передаются в слои LSTM, и вывод последнего слоя LSTM отправляется на полносвязанный слой. Слой softmax генерирует вероятность классификации.

Гибридные модели RNN-LSTM представляют собой комбинацию рекуррентных и долгосрочных краткосрочных памятных (LSTM) слоев. LSTM слои обладают способностью запоминать долгосрочные зависимости в данных и предотвращать проблему затухающего градиента. Использование гибридных моделей RNN-LSTM позволяет сочетать преимущества обеих архитектур и повышает качество прогнозирования временных рядов [2].

В данном исследовании применяются и сравниваются эти модели для прогнозирования наличия депрессии у пациентов на основе данных активности. Для построения модели использовалась библиотека TensorFlow, которая предоставляет гибкие инструменты для создания различных архитектур нейронных сетей, включая сверточные, рекуррентные и комбинированные модели, а также легко интегрируется с другими инструментами и библиотеками Python, такими как NumPy, Pandas и Matplotlib, что упрощает процесс предварительной обработки данных и визуализации результатов.

Для обучения моделей нейронных сетей сначала необходимо произвести нормализацию данных. Этот процесс заключается в приведении всех признаков к одному диапазону значений. В данном исследовании для нормализации данных используется метод Min-Max Scaling при помощи класса MinMaxScaler из библиотеки sklearn.preprocessing. Сначала данные преобразуются в нужный формат, а после проводится нормализация с помощью метода fit_transform для обучающего набора данных и метода transform для тестового набора. Это позволяет масштабировать значения каждого признака таким образом, чтобы они находились в диапазоне от 0 до 1. Нормализация помогает ускорить обучение нейронной сети, так как она уменьшает разброс значений и облегчает процесс оптимизации. Это особенно важно для функции активации, так как значения входов, близкие к 0, обеспечивают более стабильные градиенты при обратном распространении ошибки. Кроме того, нормализация помогает предотвратить насыщение функции активации, что может произойти, если входные данные имеют большой разброс значений.

4. Результаты и обсуждение

Для начала исследование проводилось на модели случайного леса. Его результаты приведены на рис. 5. Дальнейшие эксперименты с нейронными сетями будут ориентированы на достижение или превышение точности, полученной с использованием модели случайного леса. Нейронная сеть будет считаться успешной, если ее точность равна или превышает значение 0,8.

```
[[113   9]
 [ 30  43]]
              precision    recall  f1-score   support

           0       0.79      0.93      0.85       122
           1       0.83      0.59      0.69        73

    accuracy                           0.80       195
   macro avg       0.81      0.76      0.77       195
weighted avg       0.80      0.80      0.79       195
```

Рис.5. Матрица ошибок

В данном исследовании лучшие результаты были получены на рекуррентной нейронной сети. На рис. 6. приведена ее архитектура. При тестировании самые высокие результаты были получены при значениях batch_size = 16 и функции потерь 'mean_squared_error'. Точность модели составила 0.989, а значение функции потерь 0.011.

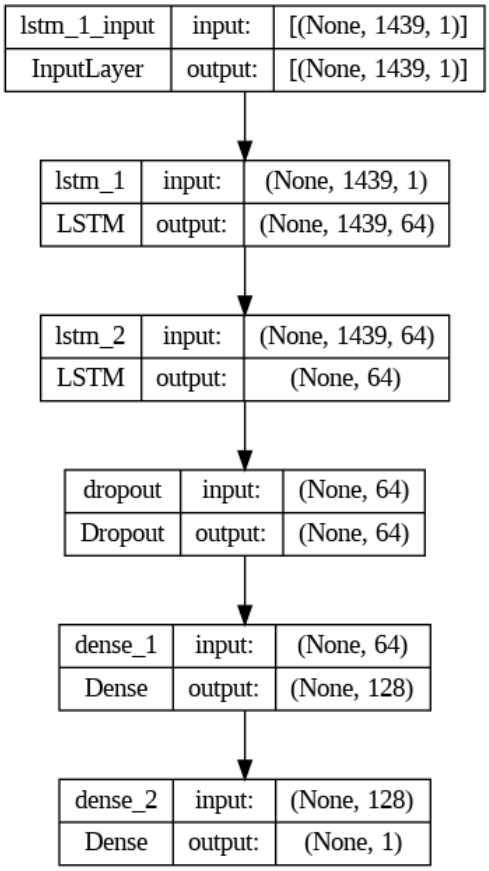


Рис. 6. Графическое представление рекуррентной модели

Вопреки всем преимуществам гибридной нейронной сети в задаче распознавания человеческой деятельности, описанной в статье Sakorn Mekruksavanich [4], в конкретной задаче этот подход не оказался наиболее эффективным. Полученная точность модели с одним сверточным и двумя LSTM слоями при значениях batch_size = 16 и функции потерь 'mean_squared_error' составила 0,917, а значение функции потерь 0.077. Тем не менее, следует

отметить, что представленная а рис. 7. модель не является исчерпывающей, и возможно, при других параметрах она проявит более высокую эффективность.

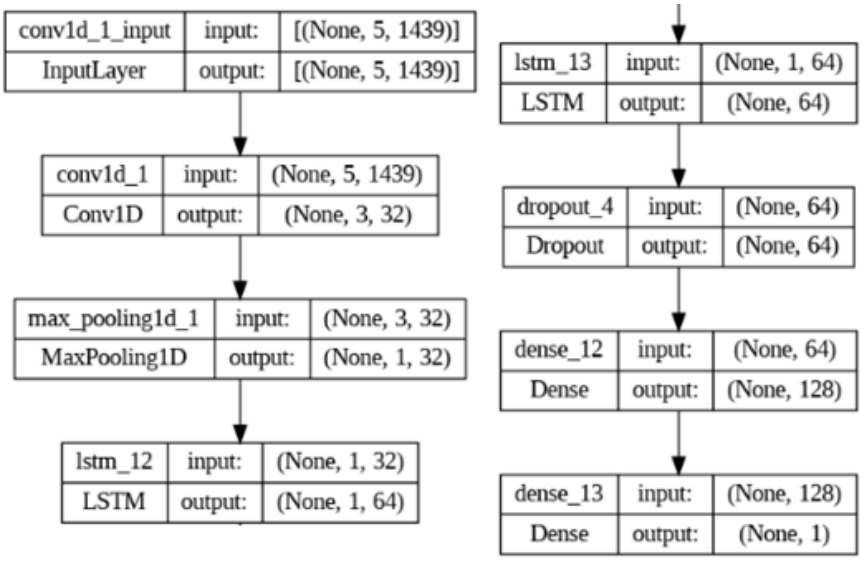


Рис. 7. Графическое представление сверточной модели

Сверточная модель оказалась хуже рекурентной, но все еще лучше гибридной. Для модели (рис. 8.) состоящей из пары слоев свертки и сжатия с 32, и парой с 64 нейронами, была получена точность 0.953 и значение функции потерь 0,0181. В результате эксперимента лучшими значениями параметров оказались batch_size = 16 и функция потерь 'binary_crossentropy'.

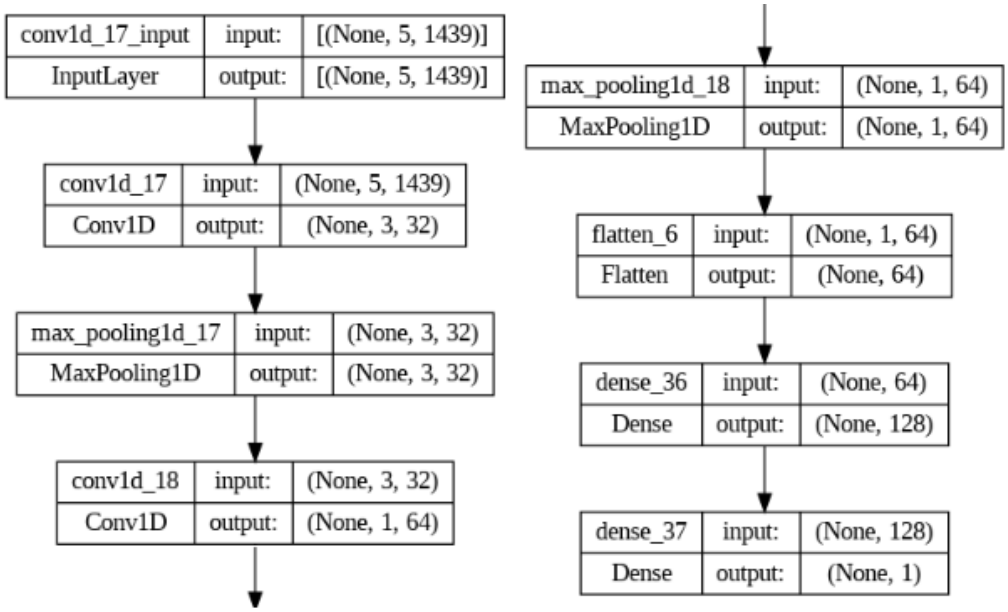


Рис. 8. Графическое представление гибридной модели

Следует отметить, что все модели превзошли результаты случайного леса, а потому могут считаться успешными. Они демонстрируют достаточно высокую точность и могут быть использованы в качестве основы для дальнейших исследований или практических применений.

Заключение

В исследовании были разработаны и протестированы модели прогнозирования наличия депрессии у пациентов на основе данных активности. Был проведен их анализ и выявлены лучшие параметры. Разработанный алгоритм демонстрирует высокую точность и может помочь в установлении болезни для ее своевременного лечения.

В связи с ограниченным доступом к медицинским данным невозможно проведение всесторонних исследований в области депрессии и ее диагностики на основе физической активности. Однако, проведение подобных исследований становится все более важным, учитывая растущую значимость мониторинга физической активности и здоровья в целом. С увеличением объема данных возможно улучшение моделей и алгоритмов, а также их доработка для более точного определения типа депрессии и выбора наиболее эффективного метода лечения для каждого конкретного случая. Таким образом, дальнейшие исследования в этой области имеют большое значение и могут привести к разработке инновационных методов диагностики и лечения психических расстройств.

Литература

1. A Motor Activity Database of Depression Episodes in Unipolar and Bipolar Patients / Garcia-Ceja E, Riegler M, Jakobsen P, Tørresen J, Nordgreen T, Oedegaard KJ, Fasmer O. B. //конф. – Амстердам: Конференция по мультимедийным системам, 2018 – С.12-15;

2. Enhanced Hand-Oriented Activity Recognition Based on Smartwatch Sensor Data Using LSTMs / Sakorn Mekruksavanich, Anuchit Jitpattanakul,Phichai Youplao,Preecha Yupapin // Symmetry – 2020 – № 12 – С. 1570;

3. Margin-Based Deep Learning Networks for Human Activity Recognition / Lv, Tianqi, Xiaojuan Wang, Lei Jin, Yabo Xiao, and Mei Song // Sensors – 2020 – № 7 – С. 1871

4. Motor activity as biomarker for depression // (Engl) – Режим доступа: https://www.kaggle.com/code/gianmarcoguarnier/motor-activity-as-biomarker-for-depression. – (дата обращения: 01.04.2024);

УДК 004.8

РЕЗУЛЬТАТЫ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ БАНКОВСКИХ КАРТ С ПОМОЩЬЮ МОБИЛЬНОГО УСТРОЙСТВА

А. Е. Мащенко, С.Ю. Болотова

Воронежский государственный университет

Введение

Банковские карты уже давно прочно вошли в нашу жизнь, постепенно все больше вытесняя наличные деньги. Развитие коммуникационных технологий и эквайринга позволяет без лишних проблем внедрить безналичную оплату практически в любую организацию. Банковские карты позволяют быстро оплатить купленный в Интернете товар, просто введя платежные реквизиты. Однако, современные гаджеты с их голосовыми ассистентами, модулями автозаполнения, подсказками, интерпретацией, распознаваниями отучили пользователей вводить множество данных механически. Ведь это действительно не очень удобно, когда приходится вводить 12-19 цифр, особенно когда речь идет о покупке в не самых комфортных окружающих условиях.

В работе [1] были предложены и рассмотрены методы распознавания текстовой информации с помощью мобильного устройства, работающего под управлением операционной системы iOS, на примере задачи распознавания ценника. При решении задачи распознавания было выделено две подзадачи: определение на изображении границ прямоугольной области - ценника и выделение границы символа на изображении и его последующей его обработкой с применением методов машинного обучения. В [2] было реализовано три механизма определения прямоугольной границы ценника:

1) средствами iOS с помощью встроенных возможностей фреймворка Vision для определения прямоугольных областей;
2) с помощью модели машинного обучения, обученной на нахождение прямоугольных областей на изображении;
3) с помощью ручной настройки границы, в рамках которой будет проводиться распознавание текста.

В данной работе приводятся результаты применения рассмотренных ранее методов для решения задачи сканирования банковской карты. Описываются результаты статистической обработки данных, полученных выполнением множества тестов с банковскими картами, имеющими различные способы нанесения и размещения символов на поверхности карты. Получены показатели времени и точности результатов использования описанных выше механизмов, а также представлены итоги их сравнительного исследования.

Сравнительный анализ

Общепринятой классификации банковских карт не существует, однако, все карты можно сгруппировать:

- по типу платежной системы;
- по способу нанесения данных;
- по расположению данных на карте.

В нашей стране чаще всего выпускаются карты платежных систем МИР, VISA, MasterCard. Однако, для тестирования удалось найти карту American Express и UnionPay.

Данные на карте могут быть как рельефными, так и плоскими. Рельефные карты бывают двух типов: с выдавленной и вдавленной информацией (номер карты, срок действий, имя владельца) на поверхности карты. Эти способы нанесения данных называются эмбоссированием и индент-печатью соответственно. Как правило, рельефные символы дополнительно окрашиваются в серебристый или золотой для эмбоссированной печати и в черный или белый для индент-печати. Реквизиты на поверхности карты могут располагаться по-разному. Обычно номер и срок действия карты расположены горизонтально на лицевой стороне карты, но сейчас выпускаются и нестандартные карты, на которых, например, номер и срок действия размещаются на обратной стороне [3, 4].

Для сравнительного анализа результатов рассматриваемых методов распознавания банковских был проведен ряд экспериментов на устройстве iPhone 13Pro с процессором Apple A15 Bionic.

В качестве объекта распознавания выбраны 43 банковские карты разных платежных систем с различным расположением номера карты и разным количеством цифр в номере. В результате были извлечены не только номера карт, но и срок действия, имя владельца и CVC-код.

Для проверки качества распознавания эксперименты были проведены при различных внешних факторах. При этом фиксировались значения следующих метрик:

- время распознавания символов на изображении;
- время обработки изображений символов с помощью модели машинного обучения;
- время вынесения предсказания;
- время синтаксического анализа;
- время одного цикла распознавания;
- общее время распознавания информации на карте;
- количество необходимых для распознавания циклов.

На рис. 1 графически представлено расположение вышеперечисленных этапов работы алгоритма относительно друг друга.

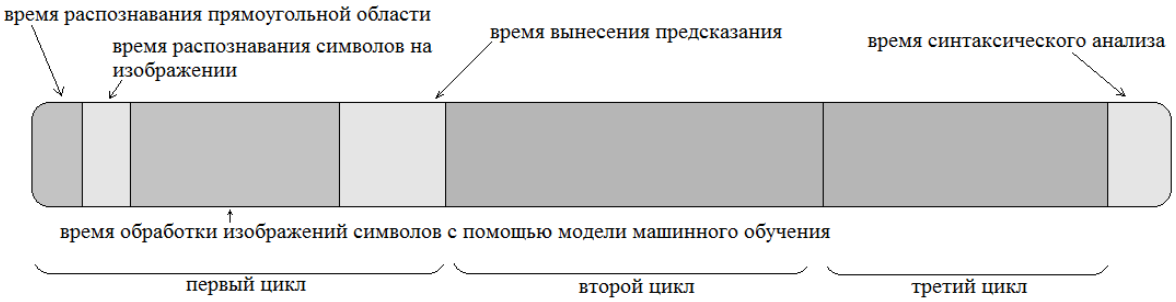


Рис. 1. Порядок следования этапов работы алгоритма

Фрагмент полученных результатов экспериментов приведен в табл. 1 и 2. В рамках эксперимента каждая карта была распознана тремя способами: средствами операционной системы iOS, с помощью сторонней модели машинного обучения, с помощью ручной настройки границ изображений. Количество циклов в каждом случае варьировалось в зависимости от результатов работы реализованного в мобильном приложении компонента, ответственного за вынесение предсказания.

На основании полученных результатов был проведен анализ и получены следующие выводы. Методы для определения информации на карте, где границы определяются

средствами платформы iOS с помощью фреймворка Vision и с помощью сторонней модели машинного обучения дают отличные результаты для эмбоссированных карт. Метод определения данных карты, где пользователь может сам настроить границы, дает наиболее быстрые и точные результаты для карт с индент-печатью и темным фоном, что обусловлено более точным определением границ.

Метод, использующий средства системы iOS, дает лучшие результаты распознавания для карт с индент-печатью светлых тонов без рисунков. Данный факт можно объяснить тем, что границу информации на карте при таких условиях определить легче, что приводит к дальнейшему сокращению необходимой к выполнению работы и уменьшению как общего времени выполнения, так и числа необходимых циклов.

У «ручного» метода время распознавания эмбоссированных карт светлых тонов, наоборот, возрастает, поскольку при таких условиях в качестве символов текста распознаются различные «ложные» области на изображении. Это приводит к дальнейшему возрастанию вычислительной нагрузки и увеличению необходимой к обработке информации.

Заключение

Все рассмотренные в работе методы могут быть применимы для решения задачи распознавания банковских карт. Проведен ряд экспериментов с разными картами. Точность распознавания символов зависит от точности распознавания границ прямоугольника, в котором они располагаются, от цвета фона карты и рисунка, а также от способа нанесения информации на карту. В дальнейшем планируется продолжить работу над получением и статистической обработкой результатов экспериментов.

Литература

1. Мащенко, А. Е. Сравнение результатов решений задачи распознавания текста с помощью мобильного устройства / А. Е. Мащенко, С. Ю. Болотова // Математика, информационные технологии, приложения : Сборник трудов Межвузовской научной конференции молодых ученых и студентов, Воронеж, 26 апреля 2023 года. – Воронеж: Издательско-полиграфический центр "Научная книга", 2023. – С. 274-277. – EDN LDTOKG.
2. Мащенко, А. Е. Решение задачи распознавания текста с помощью мобильного приложения / А. Е. Мащенко, С. Ю. Болотова // Актуальные проблемы прикладной математики, информатики и механики : сборник трудов Международной научной конференции, Воронеж, 13-15 декабря 2021 г. Воронеж, 2022. С. 200-202. ISBN 978-5-6045486-6-0.
3. Как выглядит банковская карта: внешний вид. – Режим доступа: https://www.banki.ru/wikibank/vneshniy_vid_bankovskoy_kartyi/?ysclid=lva35rp18s136275716. – (Дата обращения: 11.02.2024).
4. ВНЕШНИЙ ВИД И ОСОБЕННОСТИ КАРТ. – Режим доступа: https://storage.yandexcloud.net/bucket-cms-prod-7ff1c333-51d7-4a2d-9e7d-9e26f8ee3b6e/bank_card_about_bb2a540e4c.pdf. – (Дата обращения: 11.02.2024).

УДК 519.6

ОБ ОДНОЙ ИНТЕРЕСНОЙ ОСОБЕННОСТИ ЭКСТРАПОЛЯЦИИ КУБИЧЕСКИМИ СПЛАЙНАМИ

И. А. Меркулов

Воронежский государственный университет

Введение

Как правило, кубические сплайны применяются для интерполяции дискретных значений, но в последнее время появляются статьи, в которых рассматривается экстраполяция данных сплайнами, в частности, [1]-[2].

Постановка задачи и основные результаты

В данной статье будут рассмотрены результаты экстраполяции статистических данных кубическими сплайнами с шагом, равным 1.

Кратко опишем построение кубического сплайна.

Пусть на отрезке $[a,b]$ задана таблица значений

$x_0 = a$	x_1	x_2	$\cdots$	$x_n = b$
y_0	y_1	y_2	$\cdots$	y_n

с равномерной сеткой $\Delta_n : a = x_0 < x_1 < \ldots < x_n = b$, $h = x_i - x_{i-1}$ $(i = 1,\ldots,n)$ – шаг сетки.

Рассмотрим частичный отрезок $[x_{i-1}, x_i]$ $(i = 1,\ldots,n)$ и построим на нем кубический сплайн $S_i(x)$ с краевыми условиями $S''(a) = S''(b) = 0$. Запишем его в виде

$$S_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3 \quad (i = 1,\ldots,n),$$

где

$$a_i = y_i,\ b_i = \frac{c_i}{2}h - \frac{d_i}{6}h^2 + \frac{y_i - y_{i-1}}{h},\ d_i = \frac{c_i - c_{i-1}}{h}\ (c_0 = 0).$$

Коэффициенты c_i находятся как решение системы

$$\begin{cases} c_0 = 0 \\ c_{i-1}h + 4c_i h + c_{i+1}h = F_i \quad (i = 1,\ldots,n-1), \\ c_n = 0 \end{cases}$$

где $F_i = 6 \cdot \dfrac{y_{i+1} - 2y_i + y_{i-1}}{h}$ $(i = 1,\ldots,n-1)$.

Будем использовать кубические сплайны для экстраполяции статистических данных по ДТП (в тыс.) в России, которые приведены в табл. 1-3.

Таблица 1

Количество ДТП и погибших в них с 2000 по 2022 год

Год	Количество ДТП	Количество погибших
2000	157.5	29.6
2001	164.4	30.9
2002	184.4	33.2
2003	204.3	35.6
2004	208.6	34.5
2005	223.3	33.9
2006	229.1	32.7
2007	233.8	33.3
2008	218.3	29.9
2009	203.6	26.1
2010	199.4	26.6
2011	199.9	27.9
2012	203.6	27.9
2013	199.4	27.0
2014	199.7	26.9
2015	184.0	23.1
2016	173.7	20.3
2017	169.4	19.1
2018	168.1	18.2
2019	164.4	16.9
2020	145.1	16.2
2021	133.3	14.9
2022	126.7	14.1
2023	132.5	14.5

Таблица 2

Количество ДТП с водителями, находящимися в состоянии опьянения, и погибших в них, с 2015 по 2022 год

Год	Количество ДТП	Количество погибших
2015	15.3	3.8
2016	15.7	4.6
2017	15.0	4.3
2018	15.2	4.3
2019	14.7	4.1
2020	14.5	4.1
2021	11.8	3.4
2022	10.8	3.2
2023	10.6	3.0

Таблица 3

Количество ДТП, произошедших в темное время суток, и погибших в них, с 2015 по 2022 год

Год	Количество ДТП	Количество погибших

2015	67.7	11.8
2016	60.4	9.8
2017	60.2	9.5
2018	57.3	8.8
2019	56.9	8.1
2020	50.9	7.9
2021	44.7	6.8
2022	41.4	6.3
2023	42.9	6.3

Экстраполяцию будем проводить на один шаг вперед, постепенно отбрасывая по одному значению слева, то есть для таблицы 1 рассматривая сначала 2003-2022 годы, затем 2004-2022, 2005-2022, … и 2020-2022, для таблиц 2-3 – 2015-02022, 2016-2022, …, заканчивая 2020-2022. Для всех случаев прогнозируемые значения в 2023 году равны одному и тому же числу для каждого случая:

1) для таблицы 1 количество ДТП – 120.1 и число погибших – 13.3;
2) для таблицы 2 количество ДТП – 9.9 и число погибших – 2.9;
3) для таблицы 3 количество ДТП – 38.2 и число погибших – 5.9.

Для исключения ошибок в программе был рассмотрен пример (табл. 4), в котором для разного количества точек на последнем частичном отрезке $[x_3, x_4]$ (в нашем случае это отрезок $[3,4]$) были построены кубические сплайны S_4, коэффициенты для которых были посчитаны вручную.

Таблица 4

Пример

i	0	1	2	3	4
x_i	0	1	2	3	4
y_i	3	1	5	4	2

Результаты вычислений приведены в табл. 5.

Таблица 5

Значения коэффициентов

номера точек	a_4	b_i	c_i	d_i	$\frac{d_i}{6}$
2-3-4	$a_4 = 2$	$b_3 = -1.5$	$c_3 = -1.5$	$d_3 = -1.5$	
		$b_4 = -2.25$	$c_2 = c_4 = 0$	$d_4 = 1.5$	$\frac{d_4}{6} = 0.25$
1-2-3-4	$a_4 = 2$	$b_2 = 1.4(6)$	$c_2 = -1.76$	$d_2 = -7.6$	
		$b_3 = -2.1(3)$	$c_3 = 0.4$	$d_3 = 8$	
		$b_4 = -1.9(3)$	$c_1 = c_4 = 0$	$d_4 = -0.4$	$\frac{d_4}{6} = -0.0(6)$
0-1-2-3-4	$a_4 = 2$	$b_1 = \frac{109}{28}$	$c_1 = \frac{327}{28}$	$d_1 = \frac{327}{28}$	

		$b_2 = \frac{533}{56}$	$c_2 = \frac{75}{7} = \frac{300}{28}$	$d_2 = -\frac{27}{28}$	
		$b_3 = 0$	$c_3 = \frac{33}{28}$	$d_3 = -\frac{267}{28}$	
		$b_4 = -\frac{101}{56}$	$c_0 = c_4 = 0$	$d_4 = -\frac{33}{28}$	$\frac{d_4}{6} = -\frac{11}{56}$

Значения всех трех сплайнов в точке $x_5 = 5$ равны 0. А это возможно лишь в том случае, когда для шага $h = 1$ для любого числа частичных отрезков в последней точке $x_n = b$ значение $b_n + \frac{d_n}{6}$ – одно и то же число, что подтверждается результатами, приведенными в таблице.

Заключение

Этот интересный факт требует дальнейшего изучения и доказательства (если это возможно).

Научный руководитель – доцент, канд. физ.-мат. наук, доцент кафедры вычислительной математики и прикладных информационных технологий ВГУ, Глушакова Татьяна Николаевна.

Литература

1. Глушакова, Т. Н. Применение сплайнов для экстраполяции скачкообразных кривых / Т. Н. Глушакова, К. П. Лазарев, О. А. Будко // Научный альманах. – 2022, № 2-2(88). – С. 37-46.

2. Сплайн-экстраполяция MathCAD 12 руководство. – Режим доступа: http://radiomaster.ru/ cad/mc12/glava_13/index06.php

УДК 51-74

РАЗРАБОТКА КОЭФФИЦИЕНТА НЕРАЗЛИЧИМОСТИ ДЛЯ НЕЧЕТКОЙ МЕТРИКИ

Т. А. Моисеева

Воронежский государственный университет

Введение

На практике часто встречается ситуация, когда точное измерение расстояния невозможно или представляет интерес его приближенное значение. Для учета подобных ситуаций встречается несколько подходов, например, вероятностный и статистический. В [1] приводится новый подход – нечеткий, и вводится определение нечеткой метрики. В [2] предложенный подход был модифицирован. Будем использовать понятие нечеткой метрики из [2].

Пусть U – произвольное множество, T – непрерывная треугольная норма, M – нечеткое множество на $U \times U \times (0,\infty)$ с непрерывной функцией принадлежности $\mu_M : (0,\infty) \to (0,1]$, удовлетворяющей следующим свойствам для $\forall x, y, z \in U$ и $\forall u, v > 0$:

a) $\mu_M(x,y,u) > 0$;

b) $\mu_M(x,y,u) = 1$ тогда и только тогда, когда $x = y$;

c) $\mu_M(x,y,u) = \mu_M(y,x,u)$;

d) $\mu_M(x,y,u) * \mu_M(y,z,v) \le \mu_M(x,z,u+v)$,

тогда μ_M есть нечеткая метрика, а тройка $fms = (U, M, T)$ называется *нечетким метрическим пространством*.

В качестве примеров можно привести такие широко известные нечеткие метрики, как экспоненциальная и стандартная [3].

Экспоненциальная нечеткая метрика имеет вид: $fms_e = (U, M_e, *_M)$, где функция принадлежности задается формулой:

$$\mu_{M_e}(x,y,u) = e^{-\frac{d(x,y)}{u}},$$

где $*_M$ - минимальная t-норма, или $x *_M y = \min\{x, y\}$, $d(x,y)$ - произвольная метрика.

Стандартная нечеткая метрика $fms_s = (U, M_s, *_M)$ имеет функцию принадлежности $\mu_{M_s}(x,y,u) = \dfrac{u}{u + d(x,y)}$.

В [3] предлагается несколько подходов к конструированию нечетких метрик на основе аддитивных генераторов треугольных норм. В статье [4] рассматривается генерация нечетких метрик на основе аддитивных генераторов треугольных норм из класса рациональных функций, позволяя получить нечеткие метрики, зависящие от параметров. Различные параметры влияют на вид нечеткой метрики, что можно заметить визуально, анализируя поверхность, порождаемую нечеткой метрикой. Тем не менее, желательно иметь возможность

рассчитать количественные коэффициенты, позволяющие определить характеристики конкретной нечеткой метрики в зависимости от параметров.

В [4] была сгенерирована нечеткая метрика $\mu_\rho(x,y,u)=\frac{1-\rho}{e^{\frac{(1-\rho)d(x,y)}{u}}-\rho}$ при $\rho<1$, $\rho\neq 0$ на основе аддитивного генератора $t_\rho(x)=\frac{1}{1-\rho}\ln\frac{\rho(x-1)+1}{x}$, где $\rho<1$, $\rho\neq 0$. В качестве функции $d(x,y)$ будем рассматривать евклидово расстояние.

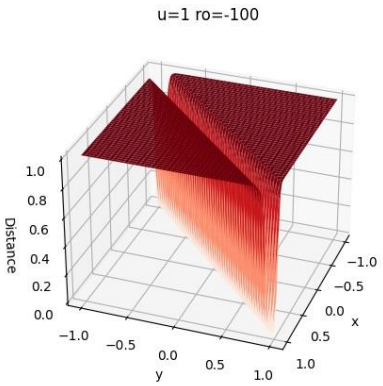
Заметим, что если $d(x,y)=0$ (это означает, что $x=y$), то $\mu(x,y,u)=1$ для любого u. Следовательно, при заданной функции u функция принадлежности метрики μ определяет нечеткое бинарное отношение сходства. Согласно [5], дополнение этого отношения является несходством и задает некоторую функцию расстояния – новую метрику. Для рассмотренной нечеткой метрики новая метрика выражается формулой:

$$r_\rho(d,u)=\frac{e^{(1-\rho)\frac{d}{u}}-1}{e^{(1-\rho)\frac{d}{u}}-\rho}.$$

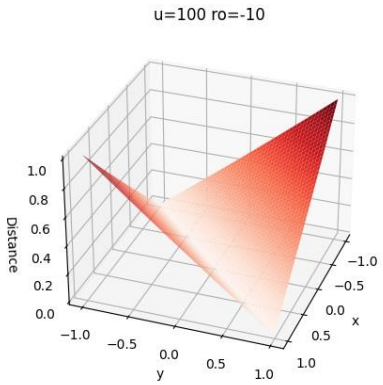
Цель настоящего исследования – представить количественный показатель, характеризующий нечеткую метрику при заданных параметрах – *коэффициент неразличимости*. В статье приводится разработанная формула показателя в общем виде для всех типов нечетких метрик, основанных на метрике Евклида, и выводится частный случай для одной из метрик, полученных на основе аддитивного генератора в форме логарифма от дробно-линейной функции.

1. Введение метрики «коэффициент неразличимости»

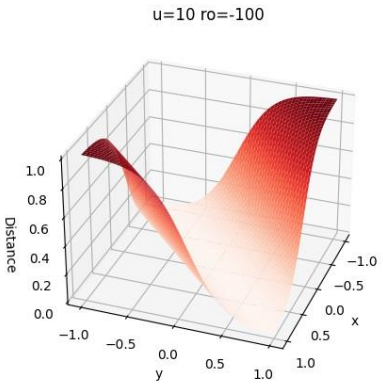
Визуализация поверхностей предложенных метрик [2-5] позволила выделить несколько характерных поверхностей функций расстояния (рис. 1). При этом поверхность П1 – частный случай поверхностей П3 и П4, а поверхность П2 – частный случай поверхностей П4 и П5.



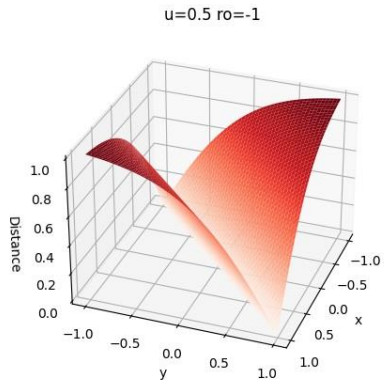
Поверхность П1



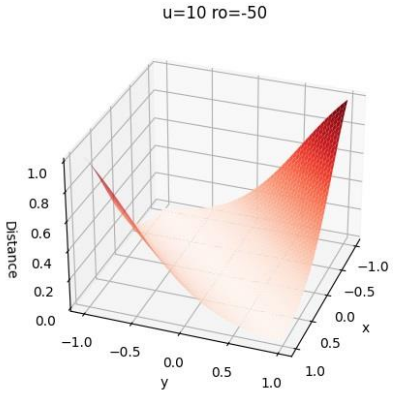
Поверхность П2



Поверхность П3



Поверхность П4



Поверхность П5

Рис. 1. Различные типы поверхностей функции расстояния

Каждой поверхности можно поставить в соответствие числовую характеристику – уровень различимости/неразличимости по отношению к некоторой базовой функции расстояния. Если поверхность, соответствующую расстоянию Евклида, взять как пример поверхности с наибольшим уровнем различимости оценок объектов, то в качестве уровня различимости можно взять расстояние между поверхностью, образованной расстоянием Евклида, и поверхностью, образованной заданной функцией расстояния. По сути, коэффициент неразличимости количественно оценивает долю тех пар объектов заданного множества, которые имеют одинаковое значение функции расстояния.

Коэффициентом неразличимости $Indist\left(r\left(d\right)\right)$ функции расстояния $r\left(d\right)$ относительно евклидовой метрики d назовем величину

$$Indist\left(r\left(d\right)\right)=\int_0^1\left|\left(r\left(d\right)-d\right)\right|\mathrm{d}d\,,$$

где значения $r(d)$, а также d нормированы к $\left[0,1\right]$.

Т. к. в вычислениях используется нормированное расстояние, вклад в метрику вносят только значения функции на отрезке [0,1]. Для данной метрики существует всего четыре случая размещения точки перегиба: точка перегиба находится левее нуля (поверхность П4), правее единицы (поверхность П5), в отрезке [0,1] (поверхность П3) и не существует

(поверхность П4). Данные варианты позволяют получить различные поверхности метрики.

Для поверхностей типа П1, П2 и П4 $\int_0^1 \left|\left(r\left(d\right)-d\right)\right|\mathrm{d}d = \int_0^1 \left(r\left(d\right)-d\right)\mathrm{d}d$, т. к. $r\left(d\right)$ лежит выше прямой $y=d$. Для поверхностей типа П3 модуль может раскрываться следующим образом. $\int_0^1 \left|\left(r\left(d\right)-d\right)\right|\mathrm{d}d = \int_0^a \left(d-r\left(d\right)\right)\mathrm{d}d + \int_a^1 \left(r\left(d\right)-d\right)\mathrm{d}d$, где a - точка перегиба. Таким образом, свой вклад в оценку неразличимости вкладывают участки как меньшие $y=d$, так и большие. Другой случай для поверхности П3 – когда точка перегиба находится правее 1. Тогда мы получаем вогнутую функцию, которая находится под прямой $y=d$. В таком случае модуль раскрывается следующим образом: $\int_0^1 \left|\left(r\left(d\right)-d\right)\right|\mathrm{d}d = \int_0^1 \left(d-r\left(d\right)\right)\mathrm{d}d$.

Будем рассматривать только один случай: метрику $r_\rho\left(d,u\right)=\frac{e^{(1-\rho)\frac{d}{u}}-1}{e^{(1-\rho)\frac{d}{u}}-\rho}$. Определим значения параметров, при которых порождаются поверхности различных типов, и рассчитаем коэффициент неразличимости для каждого случая.

2. Анализ значений параметров, определяющих разные типы поверхностей

Точками, подозрительными на точки перегиба, являются решения уравнения $d=\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)$. Найдем значения параметров, при которых получаются поверхности разного типа.

Первый случай. Найдем значения параметров, при которых точка перегиба будет находиться правее 1. Решим систему неравенств

$$\begin{cases} d=\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)>1 \\ \rho<1 \\ u>0 \end{cases}.$$

Всего существует 2 решения системы:

$$\begin{cases} u>\frac{\rho-1}{\ln\left(-\frac{1}{\rho}\right)} \\ \rho<-1 \end{cases}, \begin{cases} u<\frac{\rho-1}{\ln\left(-\frac{1}{\rho}\right)} \\ -1<\rho<0 \end{cases}.$$

Второй случай. Найдем значения параметров, при которых точка перегиба будет находиться в отрезке [0, 1]. Решим систему неравенств

$$\begin{cases} d = \dfrac{u}{\rho - 1}\ln\left(-\dfrac{1}{\rho}\right) < 1 \\ d = \dfrac{u}{\rho - 1}\ln\left(-\dfrac{1}{\rho}\right) > 0 \\ \rho < 1 \\ u > 0 \end{cases}.$$

Существует два решения системы без учета ограничений:

$$\begin{cases} 0 < u < \dfrac{\rho - 1}{\ln\left(-\dfrac{1}{\rho}\right)}, \\ \rho < -1 \end{cases} \quad \begin{cases} \dfrac{\rho - 1}{\ln\left(-\dfrac{1}{\rho}\right)} < u < 0 \\ -1 < \rho < 0 \end{cases}.$$

При заданных ограничениях на параметры $\rho < 1, u > 0$ подходит только первое решение.

Третий случай. Найдем параметры, при которых точка перегиба будет находиться левее 0. Решим систему неравенств

$$\begin{cases} d = \dfrac{u}{\rho - 1}\ln\left(-\dfrac{1}{\rho}\right) < 0 \\ \rho < 1 \\ u > 0 \end{cases}.$$

Решениями системы будут:

$$\begin{cases} u > 0 \\ -1 < \rho < 0 \end{cases}, \begin{cases} u < 0 \\ \rho < -1 \end{cases}.$$

При заданных ограничениях на параметры подходит только первое решение.

Четвертый случай. Точек перегиба не существует при значениях $0 < \rho < 1$.

3. Формирование коэффициентов неразличимости для различных типов функций

На практике, помимо нормировки базового расстояния, на котором основана нечеткая метрика, также нормируется и значение нечеткой метрики. В итоге результирующее значение нечеткой метрики считается по следующей формуле:

$$r(d) = \frac{\dfrac{e^{(1-\rho)\frac{d}{u}} - 1}{e^{(1-\rho)\frac{d}{u}} - \rho} - r_{\min}}{r_{\max} - r_{\min}},$$

где $r_{\min}$ и $r_{\max}$ - соответственно минимальное и максимальное значения рассматриваемой метрики.

Т. к. функция метрики неубывающая, то, в общем случае, $r_{\max} = r(1) = \dfrac{e^{\frac{1-\rho}{u}} - 1}{e^{\frac{1-\rho}{u}} - \rho}$, а $r_{\min} = 0$.

В итоге получим формулу для метрики:

$$r=\frac{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-1\right)\left(e^{\frac{1-\rho}{u}}-\rho\right)}{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-\rho\right)\left(e^{\frac{1-\rho}{u}}-1\right)}.$$

Коэффициент неразличимости для 1 случая.

$$Indist\left(r\left(d\right)\right)=\int_0^1\left(d-\frac{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-1\right)\left(e^{\frac{1-\rho}{u}}-\rho\right)}{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-\rho\right)\left(e^{\frac{1-\rho}{u}}-1\right)}\right)dd=$$

$$=\left.\frac{\rho d\left(e^{\frac{\rho}{u}}\left(2\rho-d\right)+e^{\frac{1}{u}}(\mathrm{d}-2)\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(1-\rho e^{\frac{(\rho-1)\mathrm{d}}{u}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}\right|_0^1=$$

$$=\frac{\rho\left(e^{\frac{\rho}{u}}\left(2\rho-1\right)-e^{\frac{1}{u}}\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(\frac{1-\rho e^{\frac{(\rho-1)}{u}}}{1-\rho}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}.$$

Коэффициент неразличимости для 3 случая и 4 случая.

$$Indist\left(r\left(d\right)\right)=\int_0^1\left(\frac{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-1\right)\left(e^{\frac{1-\rho}{u}}-\rho\right)}{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-\rho\right)\left(e^{\frac{1-\rho}{u}}-1\right)}-d\right)dd=$$

$$=-\left.\frac{\rho d\left(e^{\frac{\rho}{u}}\left(2\rho-d\right)+e^{\frac{1}{u}}(\mathrm{d}-2)\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(1-\rho e^{\frac{(\rho-1)\mathrm{d}}{u}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}\right|_0^1=$$

$$=\frac{-\rho\left(e^{\frac{\rho}{u}}\left(2\rho-1\right)-e^{\frac{1}{u}}\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(\frac{1-\rho}{1-\rho e^{\frac{(\rho-1)}{u}}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}.$$

Коэффициент неразличимости для 2 случая будет считаться по формуле:

$$Indist\left(r\left(d\right)\right)=\int_{0}^{\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)}\left(d-\frac{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-1\right)\left(e^{\frac{1-\rho}{u}}-\rho\right)}{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-\rho\right)\left(e^{\frac{1-\rho}{u}}-1\right)}\right)dd+\int_{\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)}^{1}\left(\frac{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-1\right)\left(e^{\frac{1-\rho}{u}}-\rho\right)}{\left(e^{\frac{(1-\rho)\mathrm{d}}{u}}-\rho\right)\left(e^{\frac{1-\rho}{u}}-1\right)}-d\right)dd\ .$$

Если R(d) - первообразная d$-$r(d), то

$$Indist\left(r\left(d\right)\right)=\mathrm{R}(d)\Big|_{0}^{\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)}-\mathrm{R}(d)\Big|_{\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)}^{1}=2R\left(\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)\right)-R(0)-R(1)=$$

$$=\frac{\rho d\left(e^{\frac{\rho}{u}}\left(2\rho-d\right)+e^{\frac{1}{u}}(\mathrm{d}-2)\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(1-\rho e^{\frac{(\rho-1)\mathrm{d}}{u}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}\Bigg|_{0}^{\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)}-$$

$$-\frac{\rho d\left(e^{\frac{\rho}{u}}\left(2\rho-d\right)+e^{\frac{1}{u}}(\mathrm{d}-2)\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(1-\rho e^{\frac{(\rho-1)\mathrm{d}}{u}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}\Bigg|_{\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)}^{1}=$$

$$=2\frac{\rho\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)\left(e^{\frac{\rho}{u}}\left(2\rho-\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)\right)+e^{\frac{1}{u}}\left(\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)-2\right)\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln(2)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}-$$

$$-\frac{\rho\left(e^{\frac{\rho}{u}}\left(2\rho-1\right)-e^{\frac{1}{u}}\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(1-\rho e^{\frac{(\rho-1)}{u}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}-\frac{2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(1-\rho\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}=$$

$$=2\frac{\rho\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)\left(2\left(\rho e^{\frac{\rho}{u}}-e^{\frac{1}{u}}\right)+\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)\right)+2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln(2)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}-$$

$$-\frac{\rho\left(e^{\frac{\rho}{u}}\left(2\rho-1\right)-e^{\frac{1}{u}}\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}-\frac{2u\left(e^{\frac{1}{u}}-\rho e^{\frac{\rho}{u}}\right)\ln\left(\left(1-\rho e^{\frac{(\rho-1)}{u}}\right)\left(1-\rho\right)\right)}{2\rho\left(e^{\frac{1}{u}}-e^{\frac{\rho}{u}}\right)}.$$

Если заменить $e^{\frac{1}{u}}-e^{\frac{\rho}{u}}=s$ и $\rho e^{\frac{\rho}{u}}-e^{\frac{1}{u}}=t$, получится:

$$Indist\left(r\left(d\right)\right)=2\frac{\rho\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)\left(2t+\frac{u}{\rho-1}\ln\left(-\frac{1}{\rho}\right)s\right)-2ut\ln(2)}{2\rho s}-$$
$$-\frac{\rho\left(e^{\frac{\rho}{u}}\left(2\rho-1\right)-e^{\frac{1}{u}}\right)-2ut\ln\left(\left(1-\rho e^{\frac{(\rho-1)}{u}}\right)\left(1-\rho\right)\right)}{2\rho s}.$$

Таким образом, были выведены формулы для коэффициента неразличимости с учетом различных характерных поверхностей функций расстояния.

Заключение

В статье рассмотрены нечеткие метрики, в частности, нечеткая метрика, основанная на логарифме дробно-линейной функции. Была приведена классификация поверхностей, визуализирующих расстояния, и рассмотрены значения параметров нечеткой метрики, порождающих поверхности каждого типа. Также была введена новая количественная характеристика нечетких метрик, основанных на расстоянии Евклида – коэффициент неразличимости, и выведены формулы его вычисления для рассмотренной нечеткой метрики. Таким образом, для каждой нечеткой метрики с заданными параметрами по значениям параметров возможно определить ее тип и показатель, количественно характеризующий ее отличие от метрики с наибольшей различимостью.

Научный руководитель: доктор технических наук, профессор, зав. кафедрой ВМиПИТ ВГУ, Леденева Татьяна Михайловна.

Литература

1. Kramosil I. Fuzzy metrics and statistical metric spaces / I. Kramosil, J. Michálek // Kybernetika. – 1975. – Vol. 11. – Pp. 336-344.
2. George, A. On Some Results in Fuzzy Metric Spaces / A. George, P. Veeramani. // Fuzzy Sets and System. – 1994. – Vol.64. - Issue 64. – Pp. 395-399.
3. Grigorenko, O. Two new methods to construct fuzzy metrics from metrics / O. Grigorenko, J.-J. Miñana, O. Valero // Fuzzy Sets and Systems. – 2023. – Vol. 467. – Pp. 108483.
4. Ledeneva, T. On One Approach to Constructing a Fuzzy Metric / T. Ledeneva, T. Moiseeva // 2023 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Lipetsk, Russian Federation, 2023. – Pp. 175-179.
5. Кофман А. Введение в теорию нечетких множеств / А. Кофман. – М. : Радио и связь, 1982. - 432 с.

УДК 004

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ РЕКОМЕНДАЦИИ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ

С. А. Мунтяну

Воронежский государственный университет

Введение

Одним из наиболее важных направлений в области информационных технологий является разработка алгоритмов рекомендации, которые помогают пользователям находить контент, соответствующий их предпочтениям и интересам.

Для оценки эффективности алгоритмов рекомендации используются нейронные сети. Цель работы – определить, какие модели нейронных сетей наилучшим образом подходят для задач рекомендации, а также выявить их особенности и преимущества в сравнении с традиционными методами.

Рассматриваются несколько ключевых аспектов, связанных с использованием нейронных сетей.

1. Выбор и подготовка данных. Рассматриваются наборы данных, используемые для задач рекомендации, и методы их предобработки для использования в обучении нейронных сетей.

2. Архитектуры нейронных сетей. Рассматриваются архитектуры нейронных сетей, применяемые в задачах рекомендации, включая сверточные нейронные сети, рекуррентные нейронные сети, а также их комбинации.

3. Обучение и оценка моделей. На основе экспериментов по обучению нейронных сетей на выбранных наборах данных проводится оценка их производительности с помощью различных метрик качества.

4. Сравнение с другими методами. Делается оценка эффективности нейронных сетей в задачах рекомендации в сравнении с другими традиционными методами, такими как коллаборативная фильтрация, методы содержания и гибридные подходы.

1. Выбор и подготовка данных

Набор данных должен быть достаточно большим, чтобы модель могла извлечь общие закономерности из данных и обучиться на них. Однако важно также учитывать, что данные должны быть репрезентативными для целевой аудитории или предметной области. После выбора набора данных необходимо проводится его предобработка перед использованием в обучении нейронной сети: удаление дубликатов записей и обработка пропущенных значений, нормализация и масштабирование данных для ускорения процесса обучения и улучшения стабильности модели.

2. Архитектуры нейронных сетей

Рассматриваются несколько основных типов нейронных сетей, которые применяются в этой области: сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN) и их комбинации.

2.1. Сверточные нейронные сети (CNN)

Сверточные нейронные сети часто применяются в задачах анализа изображений, но они также могут быть эффективны в обработке последовательных данных, таких как тексты или временные ряды. В контексте рекомендательных систем CNN могут использоваться для анализа содержательных характеристик объектов (например, описаний фильмов или товаров) или для извлечения признаков из изображений (например, постеров фильмов).

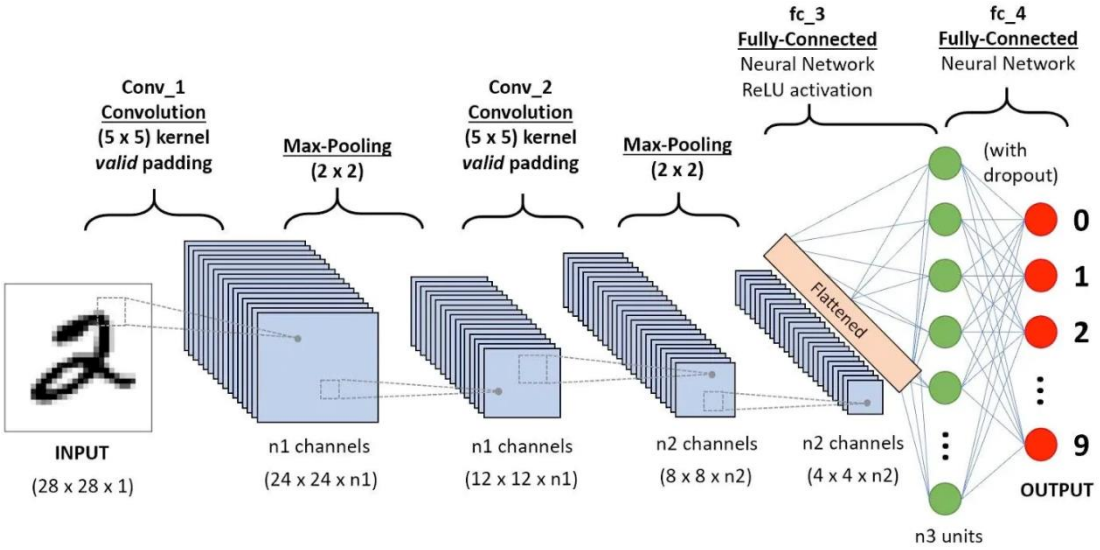


Рис. 1. Сверточные нейронные сети (CNN)

2.2. Рекуррентные нейронные сети (RNN)

Рекуррентные нейронные сети хорошо подходят для работы с последовательными данными, такими как тексты или временные ряды. Они могут учитывать контекст информации и обрабатывать последовательности переменной длины. В задачах рекомендации RNN могут использоваться для анализа истории взаимодействий пользователя с платформой или для моделирования последовательностей данных, связанных с объектами рекомендации.

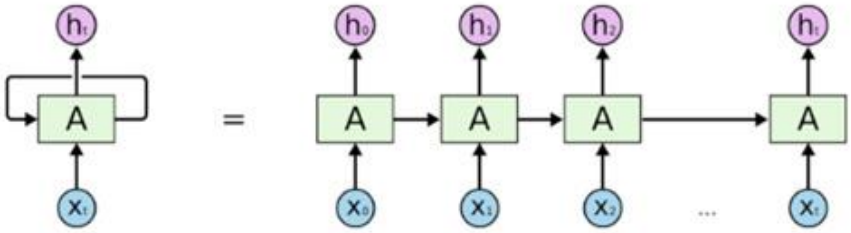


Рис. 2. Рекуррентные нейронные сети (RNN)

2.3. Комбинации архитектур

Часто эффективность рекомендательных систем можно увеличить путем комбинирования различных архитектур нейронных сетей. Например, можно использовать

сверточные слои для извлечения признаков из входных данных, а затем передать эти признаки в рекуррентные слои для моделирования последовательности или контекста.

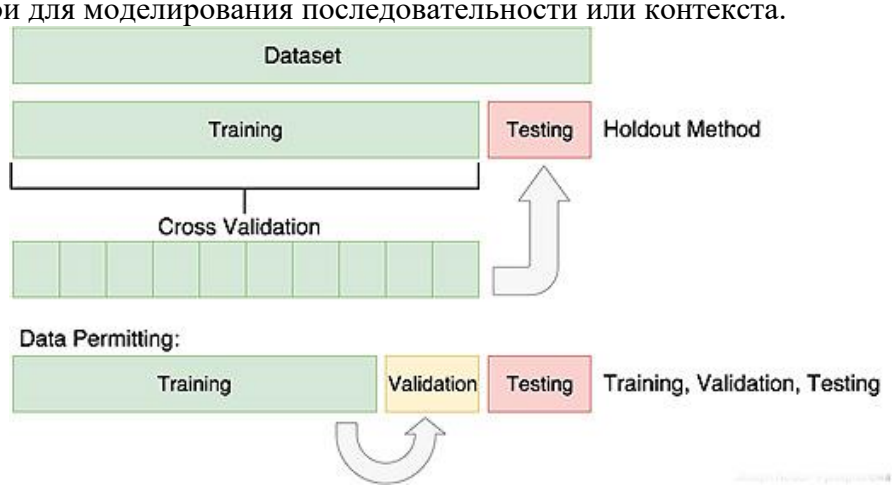


Рис. 3. Комбинации архитектур

Важно отметить, что настройка и выбор конкретной архитектуры зависят от характеристик данных, задачи рекомендации и доступных ресурсов (например, вычислительная мощность). Подбор оптимальной архитектуры часто требует проведения серии экспериментов с различными конфигурациями и анализа результатов.

3. Обучение и оценка моделей

При обучении нейронных сетей для задач рекомендации необходимо провести серию экспериментов, чтобы определить оптимальные параметры модели и оценить ее производительность. Для этого используются различные метрики качества, которые позволяют оценить точность и релевантность рекомендаций модели.

Одной из ключевых метрик является среднеквадратичная ошибка (Mean Squared Error, MSE). Эта метрика используется для оценки точности модели путем сравнения реальных значений с предсказанными. Формула для расчета MSE выглядит следующим образом:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

где - y_i реальное значение, - $\hat{y}_i$ предсказанное значение, а n - количество примеров в тестовом наборе данных.

Другой важной метрикой является точность (Accuracy). Эта метрика измеряет долю правильных предсказаний модели относительно общего числа предсказаний. Формула для расчета точности:

$$Accuracy = \frac{\text{Количество правильных предсказаний}}{\text{Общее количество предсказаний}}$$

Также в задачах рекомендации широко используются метрики ранжирования, такие как Precision@k и Recall@k, где - k количество рекомендаций. Precision@k измеряет долю правильно предсказанных элементов среди первых рекомендаций, а Recall@k измеряет долю правильно предсказанных элементов среди всех релевантных элементов.

Для оценки производительности модели могут использоваться метрики, такие как F1-мера (F1-score), которая является средним гармоническим между точностью и полнотой:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Проведение экспериментов с обучением нейронных сетей на выбранных наборах данных включает в себя обучение моделей на обучающей выборке, а затем оценку их производительности на тестовой выборке с использованием указанных метрик. Сравнение результатов различных экспериментов позволяет выбрать оптимальную архитектуру и параметры модели для конкретной задачи рекомендации.

4. Сравнение с другими методами

Сравнение эффективности нейронных сетей с традиционными методами в задачах рекомендации является важным аспектом исследований в области машинного обучения и рекомендательных систем.

1. Коллаборативная фильтрация: метод основан на анализе схожести пользователей или предметов. Для каждого пользователя или предмета строятся профили, а затем используются для рекомендации. Коллаборативная фильтрация может быть основана на памяти (memory-based) или модельной (model-based). При сравнении с нейронными сетями, коллаборативная фильтрация часто имеет проблемы с холодным стартом (cold start problem), когда данных о новых пользователях или предметах недостаточно.

2. Методы содержания: используют информацию о характеристиках предметов или пользователей для создания рекомендаций. Например, если пользователь покупал книги про фантастику, то рекомендациями будут другие книги в этом жанре. Эти методы часто более устойчивы к холодному старту, но могут страдать от проблемы ограниченности контента.

При сравнении нейронных сетей с вышеперечисленными методами обычно оцениваются такие аспекты, как точность рекомендаций, скорость обучения и прогнозирования, способность обрабатывать большие объемы данных и обучаться на разреженных данных, а также устойчивость к проблеме холодного старта. Рассмотрим несколько графиков сравнения между нейронными сетями и традиционными методами, взяв за основу вышеупомянутые критерии.

Точность рекомендаций: для оценки точности рекомендаций сравниваются фактические рекомендации с предсказанными моделью. Это может включать в себя расчет таких метрик, как средняя квадратичная ошибка (MSE), точность (precision), полнота (recall), F1-мера и другие.

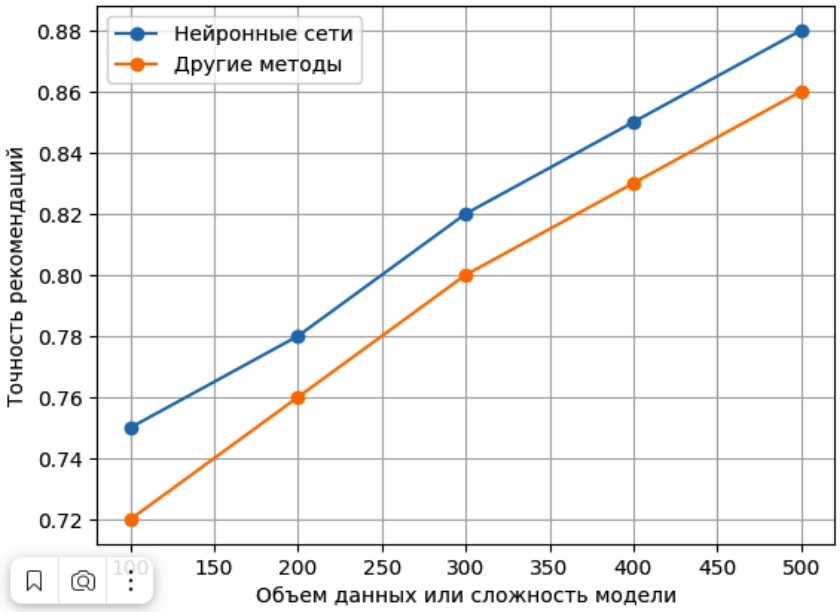


Рис. 4. Точность рекомендаций по мере увеличения объема данных

Скорость обучения и прогнозирования: скорость обучения измеряется как время, затраченное на обучение модели на тренировочных данных, а скорость прогнозирования измеряется как время, затраченное на генерацию рекомендаций для новых пользователей или предметов. Оба значения времени могут зависеть от размера набора данных, сложности модели и вычислительных ресурсов.

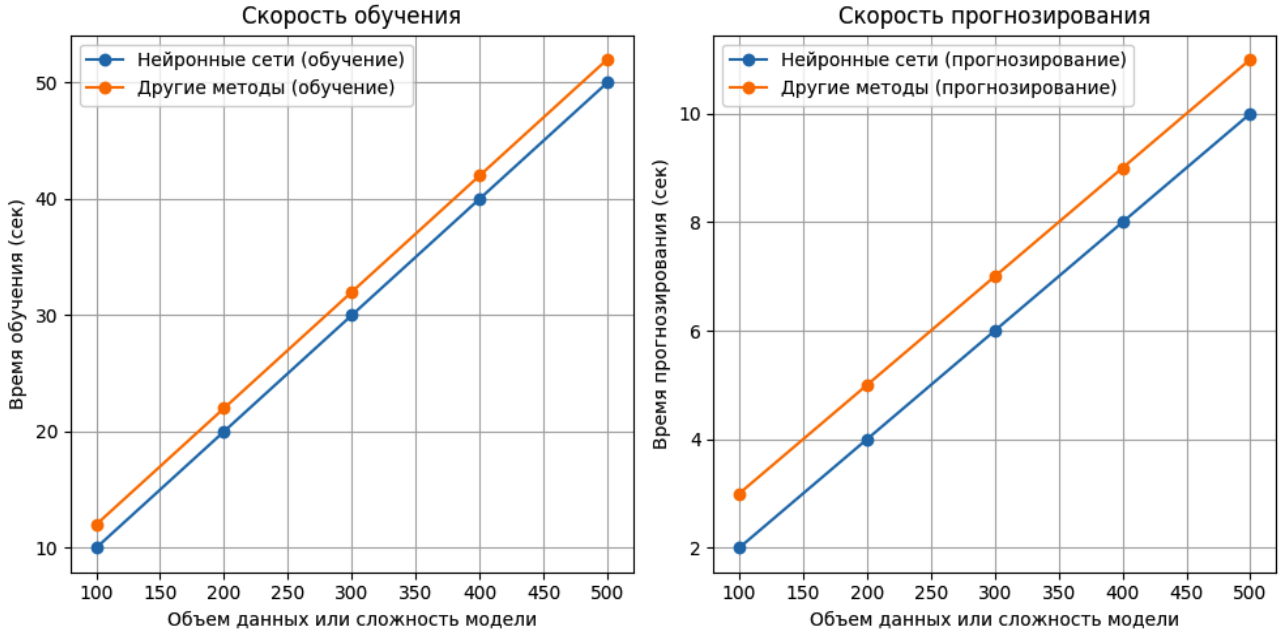


Рис. 5. Скорость обучения и скорость прогнозирования

Обработка разреженных данных: разреженные данные характеризуются большим количеством пропущенных значений. Для оценки способности модели обрабатывать разреженные данные можно построить график зависимости точности рекомендаций от уровня разреженности данных.

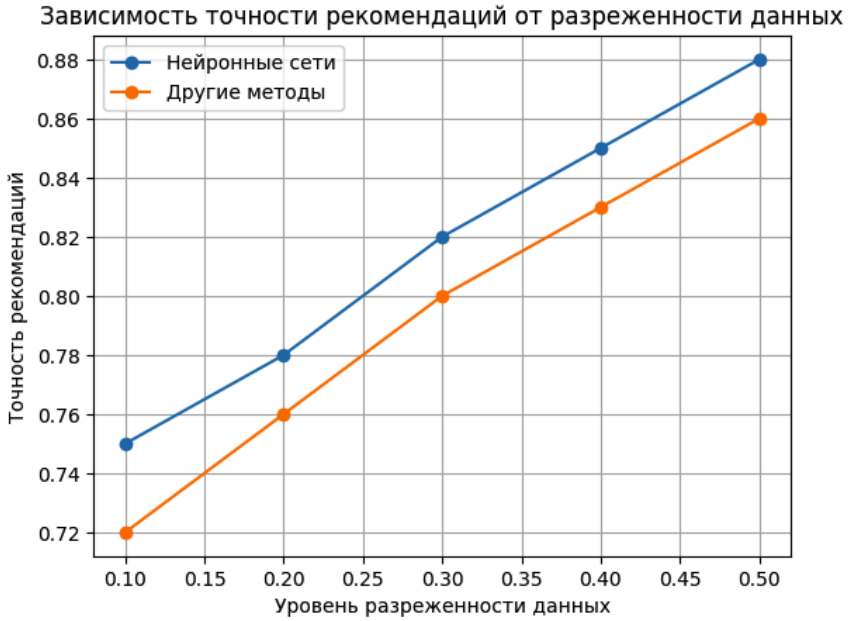


Рис. 6. Зависимость точности рекомендаций от разреженности данных

Устойчивость к холодному старту: это способность модели делать рекомендации для новых пользователей или предметов, для которых у нас ограниченные или отсутствующие данные. Оценка устойчивости к холодному старту может проводиться путем включения новых данных с различными уровнями "новизны" и анализа изменения точности рекомендаций.

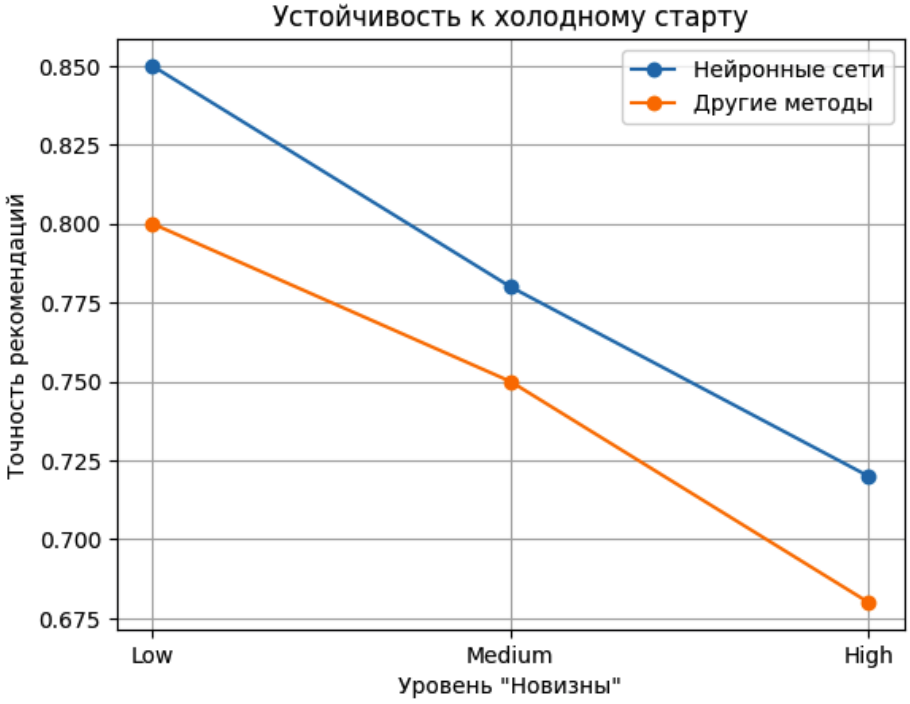


Рис. 7. Устойчивость к холодному старту

Использование ресурсов: включает в себя объем вычислительных ресурсов, таких как память и процессорное время, необходимых для обучения и прогнозирования с использованием модели. Можно оценить использование ресурсов с помощью профилирования процесса обучения и прогнозирования, а также сравнить объем ресурсов, необходимых для различных методов.

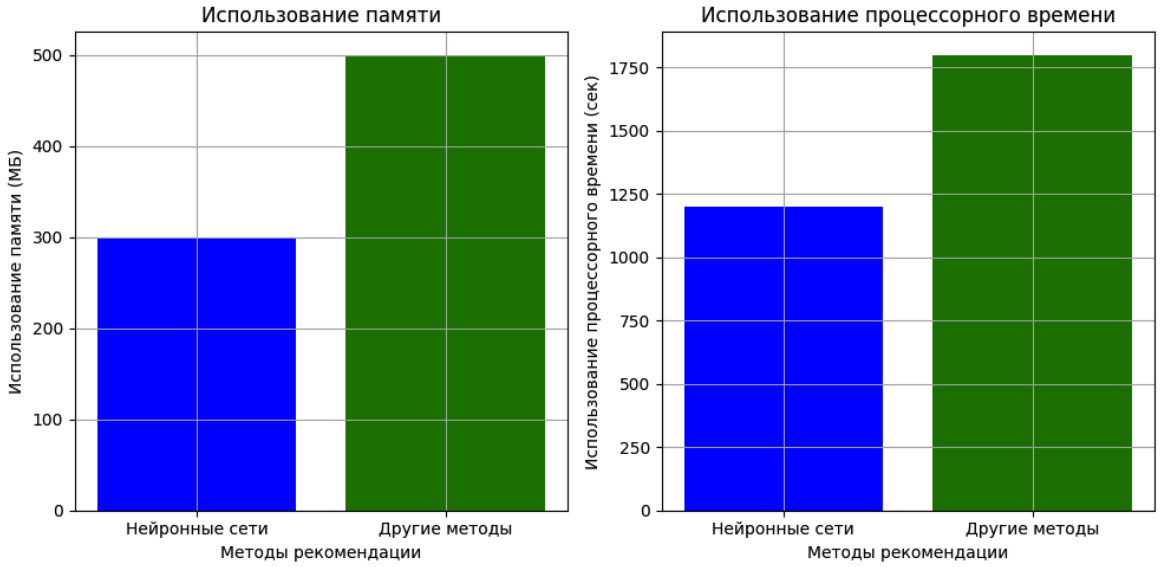


Рис. 8. Использование ресурсов для обучения

Заключение

Использование нейронных сетей для более персонализированных рекомендаций контента позволяет создавать более точные и адаптивные системы рекомендаций..

В исследованиях эффективности алгоритмов рекомендации с использованием нейронной сети учитывается разнообразие архитектур и подходов, таких как сверточные и рекуррентные нейронные сети, что позволяет подбирать наиболее подходящий метод для конкретной задачи.

Список литературы

1. Конопацкая И. И. "Применение нейронных сетей в алгоритмах рекомендации". – М.: Научный мир, 2018.

2. Арефьев Н. Н., Власов М. А., Жукова А. В. "Исследование эффективности алгоритмов рекомендации на основе нейронных сетей". – Информационные технологии, 2019, № 2.

3. Гончаров А. "Применение нейронных сетей для улучшения эффективности алгоритмов рекомендации в онлайн-магазинах". – Московский журнал вычислительной математики и кибернетики, 2017, № 3

4. Михальченко А. В., Шамшураева И. В. "Современные методы анализа данных и прогнозирования с использованием нейронных сетей". – Минск: БГУ, 2020.

5. Петров Д. И. "Нейронные сети в алгоритмах рекомендации контента". – Санкт-Петербург: Питер, 2019.

УДК 004.8

ПРИЛОЖЕНИЕ ДЛЯ ГЕНЕРАЦИИ АУДИОЗАПИСЕЙ ПО ЗАДАННЫМ КРИТЕРИЯМ

Т. В. Назаренко

Воронежский государственный университет

Аннотация. Рассматривается задача разработки приложения для генерации аудио, приводится функциональность существующих приложений, структура разрабатываемого программного продукта и описание критериев для генерации аудио.
Ключевые слова: генерация аудио, создание, музыка, аудио, звук, критерии генерации.

Введение

Исследования в области генерации аудио применяют инновационные методы и инструменты для автоматического композирования (составления) музыки, а также улучшения и расширения творческих возможностей музыкантов и продюсеров. Например, создание оригинальной музыки и автоматическое приведения трека к соответствию техническим стандартам сервисов.

Тем не менее, пользователям не хватает бесплатного и «юзер-френдли» программного обеспечения, с помощью которого они бы могли создавать аудио для собственных интересов. От видеороликов до игр – звуковое сопровождение играет важную роль. Генерация аудио по заданным параметрам позволяет создавать уникальные треки, адаптированные под конкретного слушателя и ситуацию. Создание приложения для генерации аудио – это также возможность для исследования новых методов и алгоритмов в области обработки звука и искусственного интеллекта.

Генерация музыки является популярным направлением в области машинного обучения. Некоторые алгоритмы могут создавать музыку в разных жанрах, а также анализируя существующие композиции, выявлять общие закономерности и использовать их для генерации новых музыкальных произведений. Некоторые из этих экспериментов порождают результаты, демонстрируя, насколько разнообразной и неожиданной может быть музыка.

В табл. 1 приведены результаты сравнения функциональности программных продуктов.

1. Общий анализ функциональности существующих приложений

Рассмотрим известные приложения и сайты, а также проанализируем их функциональность.

Amper Music [1] – это онлайн-платформа, использующая искусственный интеллект для автоматического создания музыки по предоставить текстовому описанию проекта, жанру или настроению. Конкретные детали об использующихся на платформе нейронных сетях и архитектуры, не являются общедоступными. Однако можно предположить, что Amper Music включает в себя: анализ исходного текста (для того, чтобы понять настроение, тему и другие характеристики, которые могут влиять на создаваемую мелодию), генерацию музыкальных фрагментов (для создания отдельных элементов композиции и в дальнейшем комбинировать их в единое целое), модели машинного обучения, такие как рекуррентные нейронные сети (RNN), которые можно применять для моделирования последовательностей музыкальных событий.

MusicLM [2] – это модель, способная создавать оригинальную высококачественную музыку на основе текстовых описаний. Основной нейронной сетью для непосредственной генерации аудио является Transformer. Для обеспечения высокого качества звучания в MusicLM используется WaveGAN (генеративная адверсариальная сеть, специализированная на синтезе аудио, которая помогает создавать музыку с четкими и приятными звуками). Для создания музыкальных фрагментов, которые соответствуют текстовым описаниям, используются генеративные автокодировщики (generative autoencoders) способные выявлять скрытые признаки в данных и генерировать новые, согласованные с заданными условиями.

Jukedeck [3] – это онлайн-платформа для генерации персонализированной музыки с использованием искусственного интеллекта. Пользователь выбирает параметры композиции, такие как жанр, настроение и длительность, и Jukedeck создает соответствующую музыкальную композицию. На платформе используется комбинация различных алгоритмов и методов машинного обучения: GAN, RNN и DNN, которые позволяют создавать реалистичные и оригинальные музыкальные композиции, путём извлечения признаков из музыкальных данных и создания новых музыкальных образцов.

VEED [4] – это веб-редактор видео, который использует искусственный интеллект для создания профессионального контента. Также VEED предоставляет инструмент «текст в музыку», который позволяет создавать уникальные саундтреки для видеороликов. Генератор использует такие алгоритмы искусственного интеллекта, как трансформеры (transformer), генеративные автокодировщики (generative autoencoders), WaveGAN и многослойные рекуррентные сети (multi-layer recurrent networks), которые используются для учета долгосрочных зависимостей в музыке, что помогает ей сохранять согласованность с текстовыми описаниями на протяжении длительных мелодий.

AIVA (Artificial Intelligence Virtual Artist) [5] – это система искусственного интеллекта, способная создавать оригинальные музыкальные композиции в различных жанрах. Хотя AIVA не работает напрямую с текстовыми описаниями, он может быть интегрирован в процесс создания музыки с учетом некоторых параметров, указанных пользователем. Она использует технологии глубокого обучения и нейронные сети (например, генеративно-состязательные сети, глубокие нейронные сети (DNN)) для генерации музыки, которая соответствует заданным критериям и случайности.

OpenAI MuseNet [6] – это глубокая нейронная сеть, способная генерировать оригинальную музыку в различных стилях и инструментах. Её обучали на сотнях тысяч MIDI-файлов, в которых она сама обнаружила паттерны гармонии, ритма и стиля, предсказывая следующий токен в музыкальных последовательностях. Также в OpenAI MuseNet были использованы трансформеры (transformer) для моделирования структуры и гармонии музыкальных композиций.

Таблица 1

Сравнительная таблица приложений

	Amper Music	**MusicLM**	**Jukedeck**	**VEED**	**AIVA**	**OpenAI MuseNet**
Генерация мелодии по тексту	да	да	нет	да	да	да
Применение жанров	да	да	да	нет	да	да
Мобильное приложение	нет	нет	нет	да	нет	нет
Скорость генерации	средняя	медленная	быстрая	быстрая	медленная	быстрая
Интерактивность	да	нет	нет	нет	нет	да
Коллекция жанров	средняя	широкая	ограниченная	нет	широкая	широкая
Качество	высоко	высокое	среднее	высоко	высокая	высокое

генерируемых аудио	е			е		
Наличие настраиваемых параметров	да	да	да	нет	да	нет
Дополнительные функции	редактирование настроения мело-дии	редактирование целей мелодии	редактирование настрое-ния мелодии	редактирование атмос-феры	редактирование типа генерации	смешение разных стилей

2. Анализ функциональности приложения и постановка задачи

Для создания собственного приложения, которое генерирует аудио по заданным пользователем критериям, необходимо обозначить ограничения, которые будут использоваться для получения необходимого пользователю результата. Были взяты параметры описания человека или персонажа, которые легко определяемы для пользователя и которые в дальнейшем трансформируются в более конкретные характеристики для белее точного контроля над генерируемым выводом. Например, при выборе пользователя в критерии хобби «плавание», программа будет преобразовывать это в характеристики «звук воды» и «энергично», а для хобби «чтение» – «расслаблено» и «фортепиано».

Параметры для генерируемого аудио:

- профессия;
- характер;
- хобби;
- взаимоотношение (например, мать, муж, дочь);
- тип личности;
- внешность;
- возраст;
- ощущения;
- должность;
- мировоззрение;
- культура (национальная принадлежность);
- раса;
- преобладающий навык;
- опыт;
- преобладающая способность.

Результаты работы приложения должны сохраняться в галерее приложения на время сессии пользователя. В дальнейшем возможно сохранить сгенерированные файлы на устройство, либо поделиться ими в социальных сетях.

На основании проанализированных ранее приложений был сделан вывод о том, что для написания программного обеспечения понадобится библиотека AudioCraft. В ней содержится такой основной компонент как MusicGen – нейронная сеть для генерации музыки (не звуков).

Преимущества работы с MusicGen:

1. MusicGen – одноэтапная модель. Они работают быстро, так как нет необходимости в нескольких проходах по данным, в отличие от многоэтапных моделей, которые могут иметь несколько последовательных шагов (например, предварительная обработка данных, извлечение признаков, классификация).

2. Используется авторегрессивная архитектуру Transformer. Т.е. модель обрабатывает предыдущие элементы последовательности и использует их для предсказания следующего элемента. Transformer использует механизм внимания для установления связей между разными частями входных данных. Это позволяет модели фокусироваться на наиболее важных элементах последовательности при генерации следующего элемента.
3. MusicGen обучена на данных с частотой дискретизации 32 кГц, что позволяет получить более высокое качество генерации и обеспечить меньшие потери информации. Чем выше частота дискретизации, тем более детализированными будут семплы (отсчеты) аудиосигнала. Это позволяет улучшить качество модели и уменьшить потерю информации при преобразовании аналогового сигнала в цифровой формат.
4. Используется 4 кодовые книги (мелодическая, гармоническая, ритмическая и инструментальная). Каждая кодовая книга вносит свой вклад в общий музыкальный результат, что позволяет создавать разнообразные и интересные музыкальные композиции, учитывая различные аспекты музыки

Таким образом, необходимо разработать приложение для генерации аудио, позволяющие приложению:

- по запросу пользователя собирать данные критериев и ранжировать их;
- на основе полученных данных формировать запрос к нейронной сети;
- генерировать аудио;
- по запросу пользователя сохранять сгенерированное аудио в отдельный файл.

Инструментальные программные средства должны обеспечивать поддержку следующих основных функций:

1. Выбор сценария генерации.
2. Выбор приоритета критериев.
3. Ввод данных критериев.
4. Выбор длительности аудио.
5. Сохранение сгенерированного аудио в файл.

4. Структура программы

В структуре программных средств можно выделить несколько блоков, каждый из которых обеспечивает выполнение определённых групп действий (рис. 1).

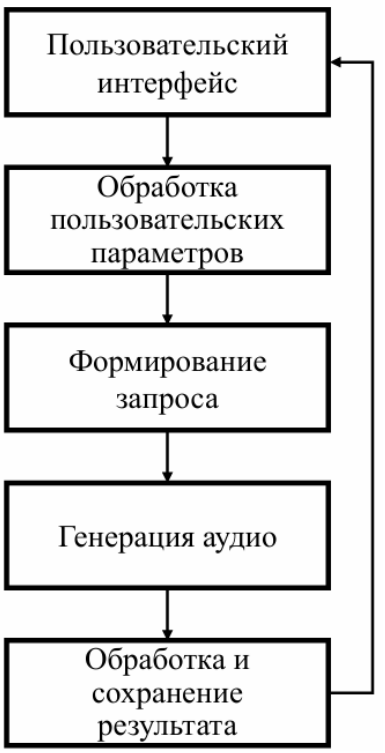


Рис. 1. Схема взаимодействия блоков программы

Пользовательский интерфейс обеспечивает взаимодействие пользователя с основными функциями программы и реакцию программы на действия, совершённые пользователем.

Блок обработки пользовательских параметров собирает введённые пользователем параметры и проверяет их корректность.

Блок формирования запроса необходим для преобразования введенных пользователем критериев в заранее заданные характеристики и помещение их в тело формируемого запроса для блока генерации.

Блок генерации аудио является алгоритмом генерации – нейронная сеть.

Блок обработки и сохранения результата позволяет приводить полученный аудиофайл к техническим стандартам сервисов, а также сохранять результат в файл.

Заключение

Разработанное приложение является интересным инструментом для генерации аудио с индивидуальной функциональностью. В начале работы с программным продуктом пользователь заполняет поля критериев и ранжирует их. Затем программа трансформирует полученные данные в более конкретные характеристики, чтобы сформировать запрос к нейронной сети. С помощью предобученной нейронной сети MusicGen модель генерирует аудио. По завершению генерации пользователь может сохранить аудио в отдельный файл.

В дальнейшем функционал приложения может быть расширен. Например, можно предоставить пользователям возможность редактировать сгенерированное аудио для дальнейшего удобства пользования.

Параметры для редактирования:

- вырезание;
- обрезка;
- объединение нескольких файлов;
- изменение тональности;

- изменение скорости;
- добавление эффектов.

Литература

1. Amper Music – URL: https://www.shutterstock.com/music (дата обращения 13.04.2024).
2. MusicLM – URL: https://musiclm.com/ (дата обращения 13.04.2024).
3. Jukedeck – ULR: https://www.jukedeck.com/ (дата обращения 13.04.2024).
4. VEED – URL: https://www.veed.io/tools (дата обращения 13.04.2024).
5. AIVA – URL: https://creators.aiva.ai/ (дата обращения 13.04.2024).
6. OpenAI MuseNet – URL: https://openai.com/research/musenet (дата обращения 13.04.2024).

УДК 004.42

Особенности разработки web-платформы кардиологического центра

Е. В. Назарьева

Воронежский государственный университет

Введение

Изучение и разработка инновационной платформы, направленной на контроль и мониторинг здоровья у людей с сердечно-сосудистыми заболеваниями, является чрезвычайно актуальной и значимой задачей. Сайт для любого медицинского центра является ключевым элементом формирования его имиджа и взаимодействия с пациентами. Критически важно обеспечить сайт структурированным, удобным для использования и содержательным, чтобы вызвать доверие и заинтересовать посетителей.

ФЗ №323 от 21.11.2011 г. обязывает медицинские учреждения, предоставляющие платные услуги, иметь собственный веб-ресурс. Однако, важность сайта для кардиологического центра превышает лишь юридические обязательства.

Рассмотрим ключевые функциональные возможности сайта, отражающие его важность. Сайт должен четко и полно описывать услуги, предоставляемые клиникой, их стоимость, квалификацию врачей и преимущества, чтобы посетители могли принять информированное решение.

Статистические данные свидетельствуют о том, что этапность лечения и своевременная медицинская помощь в случае ухудшения состояния играют определяющую роль в предотвращении осложнений и смертности у этой категории пациентов.[1] Для лиц, страдающих сердечно-сосудистыми заболеваниями, важно регулярно контролировать физиологические параметры, такие как 2 артериальное давление. Они находятся в постоянной угрозе возникновения острых состояний, требующих быстрой реакции и квалифицированной помощи. Именно в таких ситуациях эффективная и оперативная система мониторинга состояния здоровья становится жизненно важной.

1. Актуальность проблемы

Сердечно-сосудистые заболевания (ССЗ) остаются одной из основных причин смерти во всем мире. По данным Всемирной организации здравоохранения (ВОЗ), ежегодно от ССЗ умирает около 17,9 млн человек, что составляет около 31%всех смертей в мире. К ним относятся Инфаркты, инсульты, аритмии, гипертония и другие сердечно-сосудистые заболевания [1].

Несмотря на значительные достижения в области медицины, доступ к качественному медицинскому обслуживанию остается сложным для многих людей. Такие факторы, как удаленность от медицинских учреждений, ограниченность транспорта, высокая стоимость медицинских услуг и длительное время ожидания приема, делают своевременный доступ к медицинской помощи весьма затруднительным. В контексте сердечно-сосудистых заболеваний особенно важно разработать веб-платформы для кардиологических центров. Такие платформы могут предоставить пациентам возможность отслеживать показатели здоровья, такие как артериальное давление и частота пульса, а также получать прогнозы и рекомендации по профилактике сердечно-сосудистых заболеваний.

Таким образом, разработка веб-платформ для кардиоцентров позволит не только повысить доступность медицинских услуг, но и поможет пациентам более эффективно следить за своим здоровьем и предотвратить возникновение сердечно-сосудистых заболеваний.

2. Постановка задачи

Разработка web-платформы для кардиологического центра представляет собой многоэтапный процесс, включающий в себя не только создание пользовательского интерфейса, но и реализацию функционала.

Необходимо выбрать инструменты и определить основные функции веб-платформы кардиологического центра. Это включает в себя выбор языков программирования, а также определение основных функциональных возможностей платформы, таких как визуализация медицинских данных, аутентификация пользователей и прогнозирование риска заболеваний.

Также нужно проанализировать каким образом мы можем прогнозировать риск сердечно-сосудистых заболеваний. Это включает в себя изучение метода машинного обучения алгоритма, а конкретно алгоритма Random forest, его преимуществ и ограничений, а также проведение практического анализа на основе имеющихся данных о пациентах.

Выполнение указанных задач позволит разработать эффективную веб-платформу, которая будет способствовать повышению уровня медицинского обслуживания и улучшению здоровья пациентов.

3. Выбор инструментов

Для разработки веб-платформы кардиологического центра проанализируем HTML и CSS в качестве основных языков разметки и стилей соответственно.

HTML (HyperText Markup Language) является основой веб-страниц и позволяет создавать структуру контента, определяя различные элементы и их взаимосвязи. Использование HTML обеспечивает ясность и удобство в организации содержания веб-страниц, что важно для представления информации о кардиологических услугах и пациентах.

CSS (Cascading Style Sheets) используется для оформления и стилизации веб-страниц, что позволяет создавать привлекательный и современный дизайн интерфейса. Этот язык обеспечивает гибкость в управлении внешним видом элементов, адаптивность для различных устройств и браузеров, а также возможность создания анимаций и эффектов.

Выбор HTML и CSS для разработки веб-платформы кардиологического центра обеспечивает надежную основу для создания удобного и привлекательного пользовательского интерфейса, что соответствует целям проекта по предоставлению информации о медицинских услугах и обеспечению комфортного взаимодействия с платформой

Для обеспечения более углубленной интерактивности и улучшенного пользовательского опыта web-платформа кардиологического центра будет реализовывать функционал авторизации пользователей с возможностью динамического отображения графиков артериального давления. Это достигается путем интеграции JavaScript, который обеспечивает интерактивное взаимодействие пользователя с веб-интерфейсом и динамическое обновление данных без необходимости перезагрузки страницы.[4]

JavaScript будет использоваться для создания графического интерфейса, который позволит пользователям визуально отслеживать динамику своего артериального давления в разные временные точки.

Бэкэнд часть платформы будет осуществлена с использованием языка программирования PHP. Язык обеспечивает высокую гибкость и функциональность для реализации различных бизнес-логик и алгоритмов, необходимых для работы платформы

4. Интеллектуальный анализ данных как способ решения задачи прогнозирования риска сердечно-сосудистых заболеваний

Основная идея заключается в использовании методов машинного обучения для анализа исторических данных и прогнозирования будущих событий, в данном случае риска сердечно-сосудистых заболеваний.

Задачи классификации разделяют пациентов на различные классы на основе их медицинских характеристик. Эти алгоритмы могут выявлять скрытые зависимости и обнаруживать закономерности, которые полезны для прогнозирования состояния пациента.

Примером метода классификации являются случайные леса, которые представляют собой алгоритм машинного обучения, основанный на комитетах деревьев решений. К преимуществам этого метода можно отнести его способность эффективно обрабатывать данные с множеством классов и признаков, а также нечувствительность к монотонным преобразованиям значений признаков. Однако следует отметить, что у этого метода есть и недостатки, такие как большая размерность модели и потребность в большом количестве ресурсов.

Во многих случаях к моменту возникновения необходимости в прогнозировании имеется некоторая накопленная историческая информация. В медицинских учреждениях обычно хранятся значительные объемы данных о пациентах в системах управления здравоохранением. Методы интеллектуального анализа данных помогают извлечь ценные сведения из исторических данных и выявить скрытые зависимости, которые затем могут быть использованы для составления прогнозов на будущее.

При этом предполагается, что рассмотренные прошлые взаимосвязи и характеристики данных не потеряют своей актуальности в будущем. Методы классификации являются наиболее широко используемыми алгоритмами в здравоохранении, поскольку они классифицируют пациентов, распределяя их по классам на основе значений атрибутов, и помогают предсказать их состояние. Классификация обычно относится к методам машинного обучения под наблюдением, которые требуют, чтобы исходные данные были сначала разделены на несколько классов. При работе с этими данными алгоритм обнаруживает скрытые взаимосвязи между атрибутами, которые влияют на результаты прогнозирования. На выходе алгоритм представляет собой классификатор, построенный на основе обучающего набора данных, состоящего из набора данных и связанного с ним кортежа меток классов. (рисунок 1).

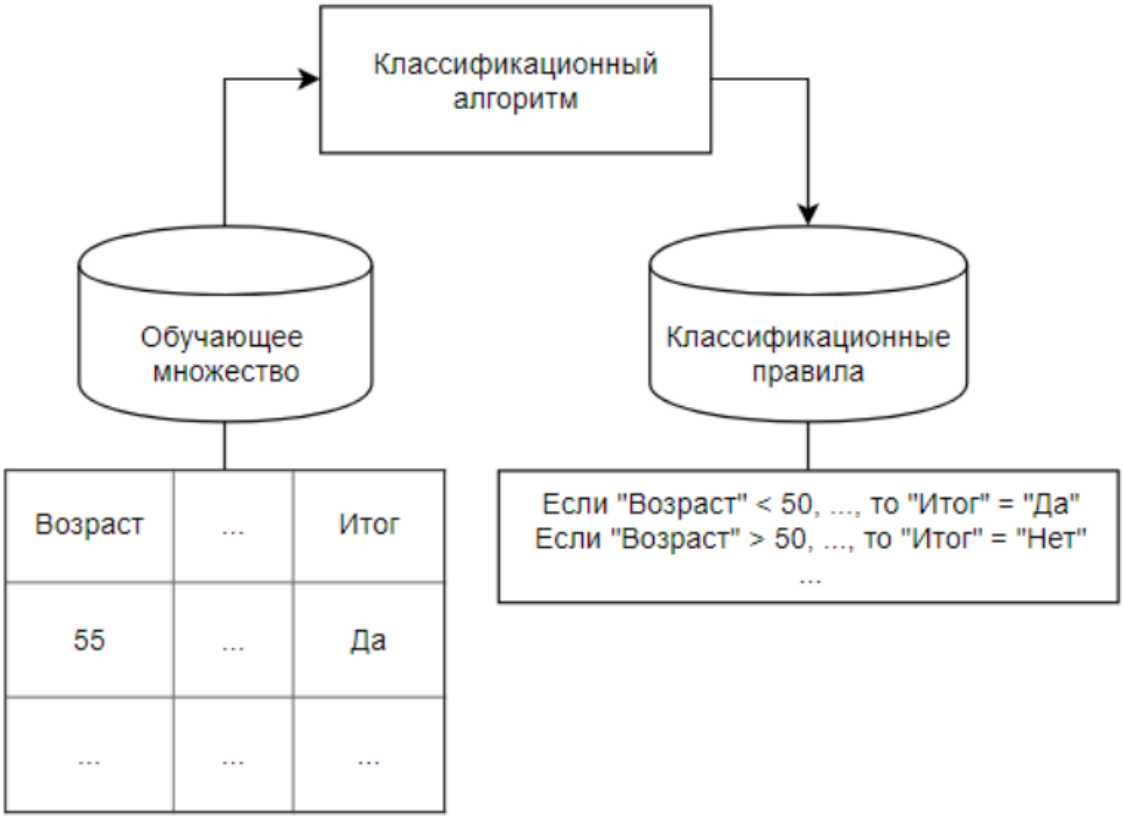


Рис. 1. Построение классификационной модели

Когда на вход поступает новый случай, построенная модель классификации относит его к одному из предопределенных классов, как показано на рисунке 2.

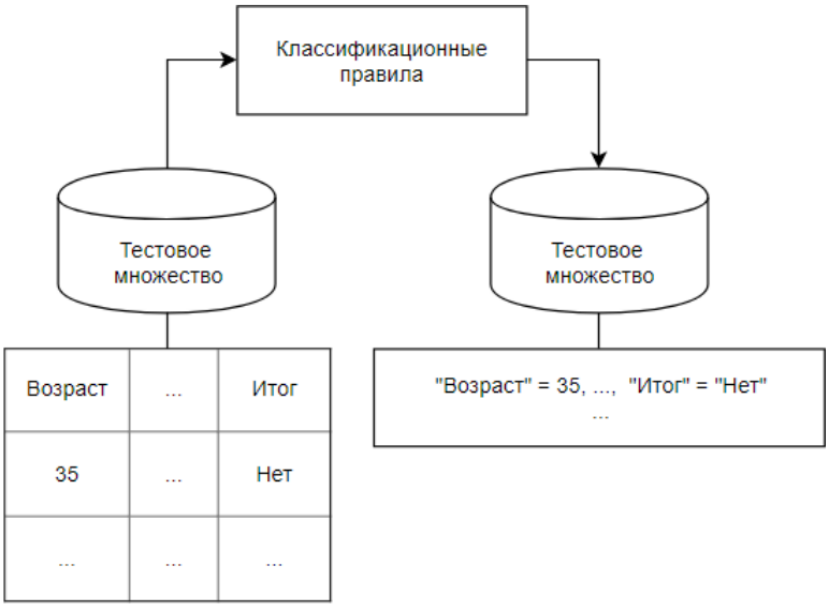


Рис. 2. Использование классификационной модели

Существует множество алгоритмов классификации, которые возможно использовать для решения поставленной задачи. Мы рассмотрим метод случайного леса.

5. Метод случайного леса

Random forest - алгоритм машинного обучения, предложенный Лео Брейманом и Адель Катлер, который использует комитеты деревьев решений. Алгоритм применяется для решения задач классификации, регрессии и кластеризации.

Предположим, что обучающая выборка состоит из N образцов, размерность пространства признаков равна M, а параметры заданы как неполное число признаков для обучения [5].

5.1. Алгоритм обучения

Из обучающей выборки генерируется случайная подвыборка с итерациями размера N;

Для классификации этой подвыборки строится дерево решений, при создании очередного узла дерева выбирается набор признаков и на их основе производится разбиение; выбор лучшего из этих m признаков осуществляется по критерию Джини. Чтобы рассчитать критерий Джини для набора признаков с классом, предположим, что □ ∈ {1,2, … ,□} и пусть □□ будет долей элементов, помеченных классом i в наборе, тогда критерий Джини вычисляется по формуле (1):

$$I_G(p) = 1 - \sum_{i=1}^{j} p_i^2 \quad (1)$$

дерево строится до тех пор, пока в нем не останется подвыборки, и не подвергается процедуре обрезки ветвей. Классификация объектов осуществляется путем голосования: каждое дерево комитета относит объекты, подлежащие классификации, к одному из классов, и побеждает класс, за который проголосовало наибольшее количество деревьев.

Преимущества:

- Эффективная обработка данных с множеством классов и признаков;
- Нечувствительность к монотонному преобразованию значений признаков;
- Эквивалентная обработка категориальных и непрерывных значений;
- Есть возможность оценить значимость признаков в модели по отдельности; тесты out-of-bag на невыбранных выборках;
- Высокая параллельность.

Недостатки - очень высокая размерность модели; O(K), где K - количество деревьев; высокий параллелизм. Случайные леса очень ресурсоемкие (поскольку глубокие деревья

строятся независимо), а точность страдает из-за ограничений по глубине (для решения сложных задач необходимо построить много глубоких деревьев). Время обучения деревьев увеличивается почти линейно с их количеством [5].

6. Прогнозирование ССЗ на примере

6.1 Метод случайного леса

У нас есть набор данных, который включает информацию о 1000 пациентах.

Каждый пациент описан по следующим признакам:

- Возраст (в годах)
- Артериальное давление (в мм рт. ст.)
- Уровень холестерина (в мг/дл)
- Уровень глюкозы (в мг/дл)
- Индекс массы тела (ИМТ)
- Наличие курения (0 - не курит, 1 - курит)

Наличие сердечно-сосудистого заболевания (0 - отсутствует, 1 - присутствует). Данные разделяются на обучающий и тестовый наборы в пропорции 70% к 30%.

Обучение модели Random Forest:

- Создается ансамбль из 100 деревьев решений.
- Каждое дерево обучается на случайной выборке из 700 пациентов (70% обучающего набора).
- Для каждого узла дерева выбирается случайное подмножество из 3 признаков (из 6 доступных).
- Голосование или среднее значение результатов всех деревьев используется для принятия решения.

Прогнозирование:

- Каждое дерево предсказывает вероятность наличия сердечно-сосудистого заболевания для каждого пациента в тестовом наборе.
- Результаты всех деревьев усредняются для получения окончательного прогноза.

Модель оценивается на тестовом наборе данных с использованием метрик, таких как точность, чувствительность, специфичность и площадь под ROC-кривой. Например, модель может показать точность 80%, что означает, что она правильно классифицировала 80% пациентов.

Пример прогнозирования:

Допустим, у нас есть пациент в тестовом наборе, который имеет следующие характеристики:

- Возраст: 55 лет
- Артериальное давление: 140 мм рт. ст.
- Уровень холестерина: 200 мг/дл
- Уровень глюкозы: 120 мг/дл
- ИМТ: 28
- Не курит

Модель Random Forest предсказывает, что вероятность наличия сердечно-сосудистого заболевания для этого пациента составляет 0.75 (75%).

На данном примере показано, как алгоритм Random Forest используется для прогнозирования риска сердечно-сосудистых заболеваний на основе медицинских данных

Заключение

На основе поставленных задач удалось определить ключевые аспекты функциональности и выбрать необходимые инструменты, что является важным шагом в создании качественной и эффективной веб-платформы.

Были выбраны основные языки разметки и стилей HTML и CSS, благодаря своей эффективности, гибкости и широкой поддержке веб-браузерами. Эти инструменты обеспечивают возможность создания привлекательного и интуитивно понятного интерфейса, что является ключевым элементом успешного взаимодействия пользователей с платформой. Для обеспечения функционала авторизации пользователей и динамического отображения графиков артериального давления был выбран язык программирования JavaScript. Для обработки запросов пользователя, валидации данных и взаимодействия с базой данных был выбран язык программирования PHP.

Особое внимание было уделено анализу метода Random Forest для прогнозирования риска сердечно-сосудистых заболеваний. Этот анализ позволил получить более глубокое понимание применимости алгоритма в контексте медицинской диагностики и выявить его преимущества и ограничения. Такой подход не только расширил наши знания о методах машинного обучения, но и предоставил ценные инсайты для выбора оптимального метода прогнозирования риска заболеваний.

Таким образом, создание веб-платформы для кардиологического центра представляет собой перспективное направление, которое имеет потенциал значительно улучшить доступ к медицинским услугам и повысить эффективность диагностики и профилактики сердечно-сосудистых заболеваний.

Научный руководитель: доктор технических наук, заведующий кафедрой математического обеспечения ЭВМ ВГУ, Абрамов Геннадий Владимирович.

Литература

1. Корягина Н. А., Рямзина И. Н., Шапошникова А. И. и др. Основные факторы риска сердечно-сосудистых заболеваний у молодого работающего населения. КВТиП. 2013;3:40-42. doi.org/10.15829/1728-8800-2013-3-40-42
2. Об актуальных проблемах борьбы с сердечно-сосудистыми заболеваниями // Совет Федерации Федерального Собрания РФ. – М., 2015 – 108 с
3. Документация по библиотеке scikit-learn для машинного обучения с Python [Электронный ресурс] — Режим доступа: http://scikit-learn.org/stable/ (Дата обращения: 14.04.2024).
4. JavaScript // MDN web docs [Электронный ресурс] – Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript
5. Random forest // Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Random_forest.
6. Печников В. Н. Создание Web-страниц и Web-сайтов. - М.: Триумф: 12 2010. - 370 с.

УДК 004.051

АНАЛИЗ ETL-ИНСТРУМЕНТА APACHE NIFI В РАЗРАБОТКЕ СИСТЕМЫ МАРКЕТИНГОВЫХ КАМПАНИЙ

В. С. Наркевич

Воронежский государственный университет

Введение

В современном мире сбор, обработка и анализ данных играют ключевую роль в развитии бизнеса. Одним из важных компонентов этого процесса является ETL-инструмент Apache NiFi, предназначенный для автоматизации процессов извлечения, трансформации и загрузки данных. в данной статье мы рассмотрим анализ применения Apache NiFi в разработке системы маркетинговых кампаний. Благодаря своей гибкой конфигурации, интеграции с различными источниками данных и возможностям обработки больших объемов информации, Apache NiFi становится неотъемлемым инструментом для эффективного управления данными в маркетинге. Рассмотрим основные преимущества и недостатки, примеры использования в процессе разработки и анализа маркетинговых кампаний с использованием Apache NiFi.

1. Архитектура Apache NiFi

Apache NiFi представляет собой платформу для автоматизации потоков данных, которая включает в себя следующие основные компоненты [1]:

1. FlowFile: базовая единица данных в Apache NiFi, представляющая собой объект, который перемещается через поток данных и содержит сами данные, а также атрибуты и метаданные.
2. Processor: компонент, который выполняет обработку данных, как правило, принимая FlowFile, выполняя над ним определенные операции (например, преобразование, фильтрацию, агрегацию) и отправляя результат в следующий узел в потоке данных.
3. Connection: связь между компонентами.
4. Flow Controller: управляющий модуль, который обеспечивает координацию между компонентами, управляет потоком данных, обеспечивает масштабируемость и отслеживает статус выполнения операций.
5. Flow: Совокупность всех компонентов (Processor, Connection и т. д.), образующих последовательность операций над данными.
6. Controller Services: Общие службы, такие как базы данных, аутентификация, шифрование и другие, которые могут быть использованы в различных компонентах Flow.
7. Reporting Task: Компонент, который собирает и передает метрики и статистику о выполнении процессов в Apache NiFi для мониторинга и анализа.

2. Преимущества Apache NiFi в маркетинговых системах

Apache NiFi предоставляет широкие возможности для автоматизации процессов обработки данных, интеграции с различными источниками, а также мониторинга и управления потоками информации. в контексте маркетинга это позволяет собирать данные о поведении

потребителей, проводить сегментацию аудитории, анализировать результаты кампаний и оптимизировать стратегии продвижения.

Преимущества использования Apache NiFi в маркетинговых системах [2]:

1. Гибкость и масштабируемость. Apache NiFi позволяет легко создавать и изменять потоки данных, что делает его гибким инструментом для маркетинговых систем. Кроме того, NiFi может масштабироваться для обработки больших объёмов данных, что важно для маркетинговых систем, которые обрабатывают большие объёмы данных о клиентах, продуктах и кампаниях.
2. Интеграция с различными источниками данных. Apache NiFi может интегрироваться с различными источниками данных, такими как базы данных, файлы, веб-сервисы и т. д. Это позволяет маркетинговым системам получать данные из различных источников и обрабатывать их в одном месте.
3. Обработка данных в реальном времени. Apache NiFi позволяет обрабатывать данные в режиме реального времени, что важно для маркетинговых систем, которые должны быстро реагировать на изменения в поведении потребителей.
4. Мониторинг и анализ данных. Apache NiFi предоставляет инструменты для мониторинга и анализа данных, что позволяет маркетинговым системам отслеживать эффективность кампаний и принимать обоснованные решения.
5. Автоматизация процессов. Apache NiFi позволяет автоматизировать процессы обработки данных, что снижает вероятность ошибок и повышает эффективность маркетинговых систем.
6. Высокая производительность. Apache NiFi оптимизирован для обработки больших объёмов данных и может обеспечивать высокую производительность при обработке данных.

3. Недостатки Apache NiFi в маркетинговых системах

Недостатки использования Apache NiFi при разработке маркетинговых систем [3]:

1. Ограничения производительности. Apache NiFi может обрабатывать большие объёмы данных, но его производительность может быть ограничена при обработке очень больших объёмов данных. Это может привести к снижению эффективности маркетинговых систем.
2. Зависимость от инфраструктуры. Apache NiFi требует наличия надёжной инфраструктуры, такой как серверы, сети и хранилища данных. Если инфраструктура не соответствует требованиям NiFi, это может привести к сбоям в работе маркетинговых систем.
3. Отсутствие готовых решений. Apache NiFi предоставляет разработчикам гибкость и контроль над потоками данных, но это также означает, что разработчики должны создавать решения для конкретных маркетинговых задач. Это может потребовать времени и усилий.
4. Необходимость тестирования. Разработчики должны тестировать потоки данных, созданные с помощью Apache NiFi, чтобы убедиться, что они работают правильно. Это может добавить дополнительный этап в процесс разработки.
5. Ограниченная поддержка. Хотя Apache NiFi имеет большое сообщество разработчиков и пользователей, его поддержка может быть ограниченной. Разработчики могут столкнуться с трудностями при поиске ответов на свои вопросы и решении проблем.
6. Риск безопасности. Apache NiFi обеспечивает безопасность данных, но разработчики должны принимать меры для защиты своих потоков данных от несанкционированного доступа. Это может потребовать дополнительных усилий и ресурсов.

4. Примеры использования Apache NiFi в маркетинге

Примеры использования Apache NiFi в маркетинговых системах:

1. Сбор данных из различных источников. NiFi может использоваться для сбора данных из различных маркетинговых источников, таких как веб-сайты, социальные сети, рекламные платформы и т. д. Собранные данные могут быть обработаны и проанализированы для получения ценной информации о поведении потребителей.
2. Обработка данных о клиентах. NiFi может использоваться для обработки данных о клиентах, таких как демографические данные, данные о поведении клиентов, данные о покупках и т. д. Обработанные данные могут быть использованы для персонализации маркетинговых кампаний и повышения их эффективности.
3. Анализ данных о маркетинговых кампаниях. NiFi может использоваться для анализа данных о маркетинговых кампаниях, таких как данные о показах рекламы, данные о кликах по рекламе, данные о конверсиях и т. д. Анализ этих данных может помочь оптимизировать маркетинговые кампании и повысить их эффективность.
4. Создание отчётов о маркетинговых кампаниях. NiFi может использоваться для создания отчётов о маркетинговых кампаниях, которые могут быть использованы для анализа результатов кампаний и принятия решений о будущих кампаниях.
5. Интеграция с другими маркетинговыми системами. NiFi может быть интегрирован с другими маркетинговыми системами, такими как системы управления контентом, системы управления взаимоотношениями с клиентами и т. д. Интеграция с этими системами может обеспечить более эффективное управление маркетинговыми кампаниями.
6. Мониторинг маркетинговых кампаний. NiFi может использоваться для мониторинга маркетинговых кампаний в режиме реального времени. Мониторинг кампаний позволяет оперативно реагировать на изменения в поведении потребителей и оптимизировать кампании.
7. Управление данными о маркетинговых кампаниях. NiFi может использоваться для управления данными о маркетинговых кампаниях, включая данные о бюджетах кампаний, данные о результатах кампаний и т. д. Управление данными позволяет обеспечить прозрачность и контроль над маркетинговыми кампаниями.

4.1. Реализация NiFi-загрузчика для обработки данных о клиентах

Применение NiFi в обработке данных о клиентах позволяет организациям эффективно собирать, обрабатывать и направлять информацию, улучшая взаимодействие с клиентами. На рис. 1 представлен пример NiFi-загрузчика, задача которого состоит в том, чтобы собрать информацию по комфортному времени информирования клиента в зависимости от его таймзоны.

В начале создается поток данных в NiFi, который будет отвечать за загрузку информации о клиентах. в данном случае в качестве источника информации используется база данных. Далее полученные данные преобразуются в нужный формат и направляются в хранилище.

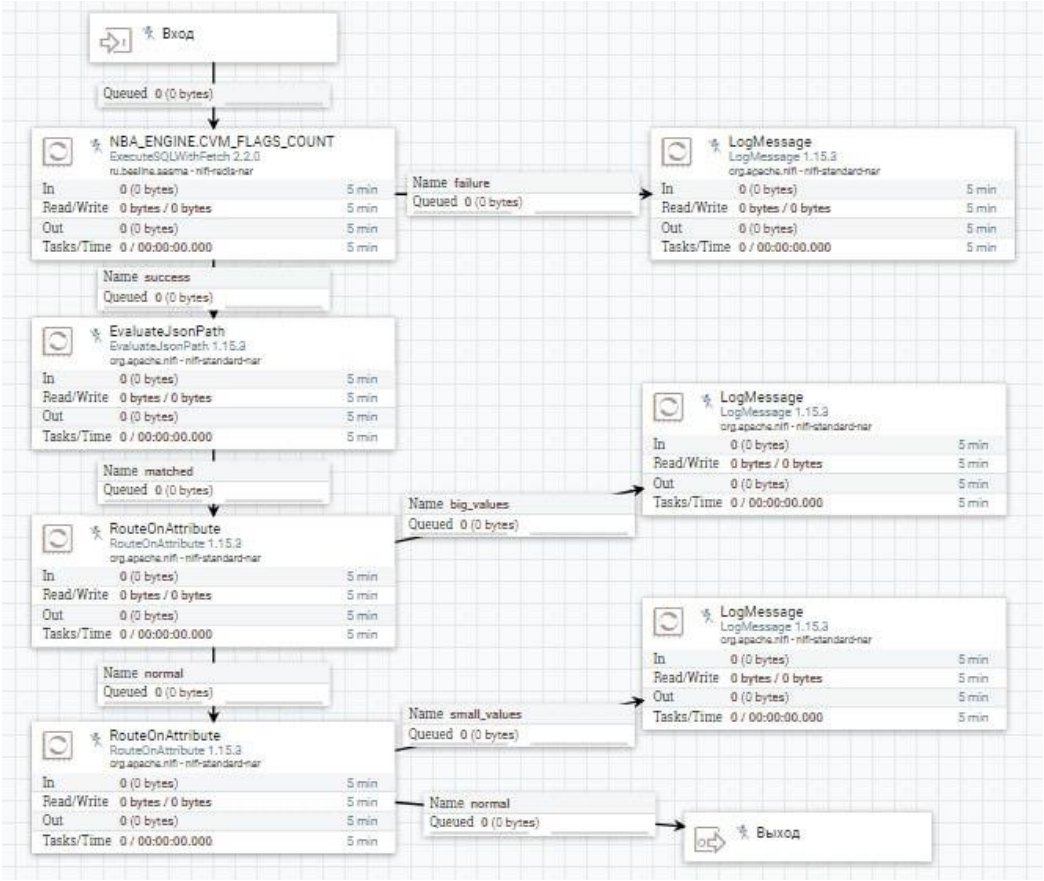


Рис. 1. NiFi загрузчик для обработки данных

Заключение

Apache NiFi представляет собой мощный инструмент для разработки систем маркетинговых кампаний, обеспечивая автоматизацию и оптимизацию процесса работы с данными. Его использование позволяет компаниям улучшить эффективность маркетинговых стратегий, повысить вовлеченность аудитории и увеличить конверсию кампаний. Разработчики и маркетологи, применяющие Apache NiFi, могут значительно ускорить работу с данными и повысить эффективность своих маркетинговых исследований.

Реализация NiFi-загрузчика для обработки данных о клиентах позволяет организациям создать надежный процесс сбора, обработки и направления информации, повышая уровень сервиса и удовлетворенность клиентов.

Информация о научном руководителе: д-р техн. наук, проф. кафедры программного обеспечения и администрирования информационных систем ВГУ Воронина И. Е.

Литература

1. NiFi Documentation – Режим доступа: https://nifi.apache.org/docs/nifi-docs/. – (Дата обращения: 10.04.2024).

2. Учебное пособие по Apache NiFi Tutorial – Режим досупа: https://datafinder.ru/products/uchebnoe-posobie-po-apache-nifi-tutorial-guide-instrukciya/. – (Дата обращения: 10.04.2024).

3. Apache NIFI – Краткий обзор возможностей на практике – Режим доступа: https://habr.com/ru/articles/465299/. – (Дата обращения: 10.04.2024).

УДК 519.816

О МЕТОДАХ ФОРМИРОВАНИЯ ВЕСОВ ПРИ ПОРЯДКОВОМ ВЗВЕШЕННОМ АГРЕГИРОВАНИИ

А. А. Несмеянова
Воронежский государственный университет

Введение

В задачах системного анализа, мониторинга и принятия решений часто требуется решить задачи ранжирования или разделения объектов на классы, при этом каждому объекту соответствует некоторый набор (вектор) признаков, компоненты которого рассматриваются как частные оценки, полученные из различных источников. Для решения таких задач часто используется подход, заключающийся в построении обобщенных оценок на основе операторов агрегирования и включающий следующие шаги: построение оценочной модели [1], выбор наиболее подходящего оператора агрегирования, вычислении обобщенных оценок для объектов заданного множества и формирование решения задачи (ранжирования, классификации, выбора лидера и др.).

Рассмотрим следующую известную в теории принятия решений задачу: пусть имеется некоторое множество объектов, обладающих определенным набором свойств, каждое из которых оценивается некоторым показателем (признаком, критерием) в соответствующей ему шкале. Требуется построить обобщенную оценку каждого объекта для их распределения по классам, ранжирования или выбора лучшего.

Цель статьи заключается в представлении подходов к конструированию процедуры агрегирования, которая включает важнейший этап – настройку весовых коэффициентов, при этом их интерпретация может быть различной. Если операция агрегирования принадлежит классу классических средних, то весовой коэффициент отражает значимость, важность показателя, его вклад в обобщенную оценку. Для операций, реализующих порядковое взвешенное агрегирование, набор весов определяется на основе некоторого принципа «нечеткого большинства» [2-3].

1. Теоретическая база исследования

Согласно [2], существует три стратегии агрегирования: *конъюнктивная*, в соответствие с которой обобщенная оценка есть минимальная $(\min)$ из компонентов векторной оценки; *дизъюнктивная*, когда в качестве обобщенной выступает максимальная $(\max)$ компонента векторной оценки; *компромиссная*, в соответствие с которой обобщенная оценка находится между $\min$ и $\max$. В общем случае порядковое взвешенное агрегирование реализует все перечисленные стратегии и формально определяется понятием OWA-оператора.

Пусть $\mathbf{a}=(a_1,a_2,\ldots,a_n)$ – векторная оценка объекта; $\mathbf{b}=(b_1,...,b_n)$ – вектор $\mathbf{a}$, упорядоченный по невозрастанию; $\mathbf{w}=(w_1,w_2,\ldots.w_n)$ – вектор весовых коэффициентов, причем $\sum_{i=1}^{n} w_i = 1$. OWA-оператор A, ассоциированный с вектором весов $\mathbf{w}$ – это отображение $\mathrm{A}:[0,1]^n \to [0,1]$, такое что

$$\mathrm{A}(\mathbf{w},\mathbf{b}) = \sum_{i=1}^{n} w_i b_i \,. \tag{1}$$

Особенностью OWA-оператора является наличие числовых характеристик, настройка которых позволяет реализовать процедуру агрегирования с заданными свойствами. К таким характеристикам относятся следующие [2]:

– характеристики *orness* и *andness*, позволяющие оценить дизъюнктивные и конъюнктивные свойства оператора A,

$$orness(\mathbf{w}) = \sum_{i=1}^{n} \frac{n-i}{n-1} w_i \in [0,1], \tag{2}$$

$$andness(\mathbf{w}) = 1 - orness(\mathbf{w}); \tag{3}$$

– мера энтропии, характеризующая насколько равномерно учитываются аргументы при агрегировании,

$$\mathrm{H}(\mathbf{w}) = -\sum_{i} w_i \log_2 w_i \in [0, \ln n]; \tag{4}$$

– мера вариабельности, отражающая степень различия (или изменчивости) весовых коэффициентов,

$$\mathrm{D}^2(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(w_i - \frac{1}{n}\sum\nolimits_{j=1}^{n} w_i \right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(w_i - \frac{1}{n} \right)^2; \tag{5}$$

– *s*-ый момент вектора весов $\mathbf{w}$, $s \in N$, используемый для оценки того, какими частными оценками можно пренебречь, чтобы подчеркнуть другие при заданном показателе компенсации $tradeoff(\mathbf{w})$,

$$\mu_s(\mathbf{w}) = \frac{1}{n}\sum_{i} w_i^s; \tag{6}$$

– показатель компенсационных свойств, позволяющий определить, насколько малые значения частных оценок могут быть скомпенсированы большими значениями оценок,

$$tradeoff(\mathbf{w}) = 1 - \sqrt{n \cdot \sum_{i} \frac{\left(w_i - 1/n \right)^2}{n-1}} \in [0,1]. \tag{7}$$

Данные числовые характеристики позволяют оценить структуру вектора весов, используемого при агрегировании вектора аргументов.

Характеристики $orness(\mathbf{w})$, $andness(\mathbf{w})$ являются одними из самых важных, поскольку с помощью них можно оценить позицию лица, принимающего решение (ЛПР) по отношению к риску. В [2] утверждается, что оптимистическая позиция, когда ЛПР склонно к риску, предполагает дизъюнктивное агрегирование, а пессимистическая, при которой ЛПР не склонно к риску) – наоборот, конъюнктивное. В нашем случае, если $orness(\mathbf{w}) \to 1$ $(andness(\mathbf{w}) \to 0)$, то позиция ЛПР оптимистическая, в противном случае, если $orness(\mathbf{w}) \to 0$ $(andness(\mathbf{w}) \to 1)$, то позиция ЛПР пессимистическая.

В [2] отмечено, что наличие компенсационных свойств также можно оценить с помощью $orness(\mathbf{w})$, так как max соответствует показателю полной компенсации (здесь $orness(\mathbf{w}) = 1$). В [2] приведено несколько важных правил:

– $orness(\mathbf{w}) > 0.5 \to \mathrm{A}(\mathbf{w},\mathbf{b})$ называется квазидизъюнкцией;

– $orness(\mathbf{w}) < 0.5$ $(andness(\mathbf{w}) > 0.5) \to \mathrm{A}(\mathbf{w},\mathbf{b})$ называется квазиконъюнкцией.

OWA-оператор относится к классу взвешенных средних. Его главной проблемой является *определение вектора весов*. Обзоры существующих подходов к решению этой

проблемы представлены в [2-4]. Один из известных и широко распространенных подходов к определению весов заключается в использовании функций квантификации, которые формализуют понятие «нечеткого большинства».

Функцией квантификации называется отображение $\mathrm{Q}:[0,1]\to[0,1]$, удовлетворяющее следующим свойствам: Q – непрерывная неубывающая функция, такая что $\mathrm{Q}(0)=0$, причем существует x', для которого $\mathrm{Q}(x')=1$.

С помощью функции квантификации Q весовые коэффициенты определяются следующей формулой [2-4]:

$$w_i=\mathrm{Q}\left(\frac{i}{n}\right)-\mathrm{Q}\left(\frac{i-1}{n}\right),\quad i=\overline{1,n}. \tag{8}$$

Способы задания весовых коэффициентов определяют соответствующие классы OWA-операторов. Например, функция квантификации

$$\mathrm{Q}_{a,b}(x)=\begin{cases}0, & 0\le x<a,\\ \dfrac{x-a}{b-a}, & a\le x\le b,\\ 1, & x>b\end{cases}$$

определяет весовые коэффициенты следующего вида:

$$w_i=\begin{cases}0, & 1\le i\le an,\\ \dfrac{i-an}{n(b-a)}, & an<i\le an+1,\\ \dfrac{1}{n(b-a)}, & an+1<i<bn,\\ \dfrac{bn+1-i}{n(b-a)}, & bn\le i<bn+1,\\ 0, & bn+1\le i\le n,\end{cases}$$

которые соответствуют EVROWA-операторам (OWA based on Extreme Values Reductions), суть которых заключается в том, что игнорируются первые an и последние $n-(bn+1)$ значений аргументов, т.е. не учитываются максимальные и минимальные частные оценки [2-3].

В качестве функций квантификаций, формализующих кванторы *большинство*, *много*, *как можно больше* и подобные, используются степенные функции вида: $\mathrm{Q}_a=x^a,\quad a>1$. В [2] показано, что при $a=2$ $andness(\mathrm{Q}_a)=1-orness(\mathrm{Q}_a)=1-\dfrac{1}{1+a}\approx 0.67>0.5$, поэтому соответствующий OWA-оператор является квазиконъюнкцией.

Рассмотрим подход, основанный на решении оптимизационных задач, суть которого заключается в нахождении такого набора весов, который максимизирует или минимизирует некоторую числовую характеристику, при этом ограничения имеют следующий вид:

$$\begin{cases}\sum_{i=1}^{n}\dfrac{n-i}{n-1}w_i=const,\\ \sum_{i=1}^{n}w_i=1,\quad w_i\in[0,1]\quad (i=\overline{1,n}).\end{cases}$$

В качестве целевых функций рассматриваются следующие [2-3]:

$$\mathrm{H}(\mathbf{w}) = -\sum_i w_i \ln w_i \to \max,$$

$$\mathrm{H}(\mathbf{w}) = 1 - \underset{i}{Max}\, w_i \to \max \Leftrightarrow \underset{i}{Max}\, w_i \to \min,$$

$$\mathrm{D}^2(\mathbf{w}) = \frac{1}{n}\sum_i (w_i - \frac{1}{n})^2 \to \min.$$

Эти функции реализуют принцип максимальной энтропии, порождающий семейство MEOWA (Maximum Enthropy OWA) операторов. Также в [2] предлагается использовать принцип моментов, по которому можно пренебречь одними оценками, чтобы подчеркнуть другие. В данном случае используется функция вида:

$$\mu_s(\mathbf{w}) = \frac{1}{n}\sum_i w_i^s \to \min.$$

При решении оптимизационной задачи с данной целевой функцией могут использоваться суммарные моменты, как натуральной, так и дробной или отрицательной степени.

2. Новый подход к определению весовых коэффициентов

В [3] был предложен подход определения вектора весов на основе нормального вероятностного распределения. В рамках этого подхода весовые коэффициенты определяются так:

$$w_i = \frac{e^{-\frac{(i-\mu_n)^2}{2\sigma_n^2}}}{\sum_{j=1}^{n}\left(e^{-(j-\mu_n)^2 \big/ 2\sigma_n^2}\right)}, i = \overline{1,n}, \tag{9}$$

где

$$\mu_n = \frac{1}{n}\frac{n(n+1)}{2} = \frac{n+1}{2}, \quad \sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(i-\mu_n)^2 = \left(\frac{n^2-1}{12}\right)^2. \tag{10}$$

В [4] была развита идея метода определения вектора весов на основе эллиптических вероятностных распределений. В этом подходе обобщенная формула для нахождения весовых коэффициентов имеет вид:

$$w_i = \frac{\mathrm{g}\left[\left(\frac{i-\mu_n}{\sigma_n}\right)^2\right]}{\sum_{j=1}^{n}\mathrm{g}\left[\left(\frac{j-\mu_n}{\sigma_n}\right)^2\right]}, \tag{11}$$

где g — генератор плотности эллиптического вероятностного распределения.

В табл. 1 приведены несколько генераторов из [4].

В [4] были доказаны следующие свойства вектора весов, найденного с помощью этого подхода:

– $w_i = w_{n+1-i}, \quad i = \overline{1,n}$ (симметричность);

– $orness(\mathbf{w}) = 0.5$;

Таблица 1

Эллиптические распределения и их генераторы плотности

Распределение	Генератор плотности g(x)
Нормальное	$g(x) = e^{-x/2}$
Показательное	$g(x) = e^{-rx^s}, \quad r,s > 0$
Лапласа	$g(x) = e^{-\lvert x\rvert}$
Стьюдента	$g(x) = \left(1+\frac{x}{m}\right)^{1+m/2}, \quad m \in N \setminus \{0\}$

$\begin{cases} w_i \le w_{i+1}, & i = \overline{1, \left[n+1/2\right]}, \\ w_i \ge w_{i+1}, & i = \overline{\left[n+1/2\right]+1, n}, \end{cases}$ причём g – невозрастающий генератор.

В [3-4] сделан акцент на том, что данный подход позволяет уменьшить влияние несправедливо высоких и низких частных оценок и увеличить влияние частных оценок, близким к среднему значению.

3. Вычислительный эксперимент

Рассмотрим иллюстративный пример, демонстрирующий возможности OWA-агрегирования. Предположим, что в курортной зоне решено построить санаторий, при этом в качестве критериев для выбора проекта застройки выбраны следующие: *климатические условия* (КУ), *близость к морю* (М), *наличие инфраструктуры* (И), *возможность создания экскурсионных маршрутов и доступных достопримечательностей* (Э), *стоимость проекта* (П). В результате проведения тендера были заявлены 5 проектов, каждый из которых оценивался по перечисленным критериям в 100-балльной шкале. Необходимо выбрать оптимальный проект с учетом всех принятых для оценки критериев. Частные оценки проектов представлены в следующей таблице.

Таблица 2

Частные оценки проектов

Проект	**КУ**	**М**	**И**	**Э**	**П**
$\mathbf{p}_1$	85	70	33	95	98
$\mathbf{p}_2$	100	65	80	43	95
$\mathbf{p}_3$	80	61	83	71	88
$\mathbf{p}_4$	75	39	81	92	96
$\mathbf{p}_5$	88	59	41	95	100

Для формирования обобщенной оценки будем использовать OWA-агрегирование, при этом для определения весовых коэффициентов воспользуемся двумя подходами. Первым рассмотрим подход к определению весов на основе эллиптического нормального

распределения, а вторым – подход к определению весов с помощью решения оптимизационной задачи по минимизации вариабельности $\mathrm{D}(\mathbf{w})$ весовых коэффициентов.

Выберем в качестве генератора функцию $\mathrm{g}(x) = e^{-\frac{x}{2}}$, которая соответствует нормальному распределению. Тогда по формулам (10) получим

$$\mu_5 = \frac{n+1}{2} = \frac{6}{2} = 3, \quad \sigma_5^2 = \left(\frac{n^2-1}{12}\right)^2 = \left(\frac{24}{12}\right)^2 = 4.$$

Согласно формуле (9), получим весовые коэффициенты

$$w_1 = \frac{e^{-\frac{1}{2}}}{3.9780} \approx 0.1525, \quad w_2 = \frac{e^{-\frac{1}{8}}}{3.9780} \approx 0.2218, \quad w_2 = \frac{1}{3.9780} \approx 0.2514.$$

По свойству симметричности весов имеем

$$w_4 = w_2 \approx 0.2218, \quad w_5 = w_1 \approx 0.1525.$$

Таким образом, вектор весовых коэффициентов имеет вид

$$\mathbf{w} = (0.1525, 0.2218, 0.2514, 0.2218, 0.1525).$$

Переупорядочивая компоненты векторных оценок $\mathbf{p_i}$ и используя вектор $\mathbf{w}$, найдём обобщенные оценки проектов

$$A_1 = \mathrm{OWA}_{\mathrm{g}(x)}(\mathbf{p_1}) = 98 \cdot 0.1525 + 95 \cdot 0.2218 + 85 \cdot 0.2514 +$$
$$+70 \cdot 0.1525 + 33 \cdot 0.2218 \approx 77.94;$$
$$A_2 = \mathrm{OWA}_{\mathrm{g}(x)}(\mathbf{p_2}) = 100 \cdot 0.1525 + 95 \cdot 0.2218 + 80 \cdot 0.2514 +$$
$$+65 \cdot 0.1525 + 43 \cdot 0.2218 \approx 77.41;$$
$$A_3 = \mathrm{OWA}_{\mathrm{g}(x)}(\mathbf{p_3}) = 88 \cdot 0.1525 + 83 \cdot 0.2218 + 80 \cdot 0.2514 +$$
$$+71 \cdot 0.1525 + 61 \cdot 0.2218 \approx 76.99;$$
$$A_4 = \mathrm{OWA}_{\mathrm{g}(x)}(\mathbf{p_4}) = 96 \cdot 0.1525 + 92 \cdot 0.2218 + 81 \cdot 0.2514 +$$
$$+75 \cdot 0.1525 + 39 \cdot 0.2218 \approx 77.92;$$
$$A_5 = \mathrm{OWA}_{\mathrm{g}(x)}(\mathbf{p_5}) = 100 \cdot 0.1525 + 95 \cdot 0.2218 + 88 \cdot 0.2514 +$$
$$+59 \cdot 0.1525 + 41 \cdot 0.2218 \approx 77.78.$$

Заметим, что наилучшим проектом является проект $\mathbf{p_1}$.

Теперь получим вектор весов, решая оптимизационную задачу следующего вида:

$$\begin{cases} \mathrm{D}(\mathbf{w}) = \dfrac{1}{5}\displaystyle\sum_{i=1}^{5}\left(w_i - \frac{1}{5}\right)^2 \to \min, \\ \mathrm{orness}(w) = \displaystyle\sum_{i=1}^{5}\frac{n-i}{n-1}w_i = 0.5, \\ \displaystyle\sum_{i=1}^{5} w_i = 1. \end{cases}$$

Для нахождения экстремума составим функцию Лагранжа

$$\mathrm{L}\left(w,\lambda_0,\lambda_1,\lambda_2\right)=\lambda_0 D\left(w\right)+\lambda_1\left(orness\left(w\right)-0.5\right)+\lambda_2\left(\sum_{i=1}^{5}w_i-1\right)=$$

$$=\lambda_0\frac{1}{5}\left(\left(w_1-\frac{1}{5}\right)^2+\left(w_2-\frac{1}{5}\right)^2+\left(w_3-\frac{1}{5}\right)^2+\left(w_4-\frac{1}{5}\right)^2+\left(w_5-\frac{1}{5}\right)^2\right)+$$

$$+\lambda_1\left(w_1+\frac{3}{4}w_2+\frac{2}{4}w_3+\frac{1}{4}w_4-0.5\right)+\lambda_2\left(w_1+w_2+w_3+w_4+w_5-1\right).$$

Пусть $\lambda_0=1$. Выпишем достаточные условия экстремума

$$\begin{cases}\dfrac{\partial \mathrm{L}}{\partial w_i}=\dfrac{2}{5}\left(w_i-\dfrac{1}{5}\right)+\dfrac{4-i+1}{4}\lambda_1+\lambda_2=0,\quad i=\overline{1,5}, & (12)\\ w_1+\dfrac{3}{4}w_2+\dfrac{2}{4}w_3+\dfrac{1}{4}w_4-0.5=0, & (13)\\ w_1+w_2+w_3+w_4+w_5-1=0. & (14)\end{cases}$$

Из (12) получим подсистему

$$\begin{cases}w_1-w_2=-\dfrac{5}{8}\lambda_1,\\ w_1-w_3=-\dfrac{5}{4}\lambda_1,\\ w_1-w_4=-\dfrac{15}{8}\lambda_1,\\ w_1-w_5=-\dfrac{5}{2}\lambda_1.\end{cases}$$

Выразив w_2,w_3,w_4,w_5 через w_1 и λ_1 и подставив полученное выражение в (14), получим $w_1=\frac{1}{5}-\frac{5}{4}\lambda_1$. Аналогично выразив остальные коэффициенты через подобные системы и затем подставив w_1,w_2,w_3,w_4,w_5 в (13), получим, что $\lambda_1=0,\quad \lambda_2=0$ и

$$w_1=w_2=w_3=w_4=w_5=\frac{1}{5}.$$

Таким образом, $\mathbf{w}^*=\left(1,0,0,\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5}\right)$ — подозрительная на экстремум точка. Проверяя для найденного решения условия теоремы Куна-Таккера [5], получим, что $\mathbf{w}^*$ является точкой минимума функции $\mathrm{L}\left(\mathbf{w},\lambda_0,\lambda_1,\lambda_2\right)$.

Найдём обобщенные оценки для проектов в соответствии с рассматриваемым подходом

$$\mathrm{OWA}_{optim}\left(\mathbf{p_i}\right)=\frac{1}{5}\left(\sum_{j=1}^{5}p_j^{(i)}\right),\quad i=\overline{1,5}.$$

В нашем случае $\sum_{j=1}^{5}p_j^{(1)}=\sum_{j=1}^{5}p_j^{(2)}=\ldots=\sum_{j=1}^{5}p_j^{(5)}=383.$

Тогда $\mathrm{OWA}_{optim}\left(\mathbf{p_1}\right)=\mathrm{OWA}_{optim}\left(\mathbf{p_2}\right)=\ldots=\mathrm{OWA}_{optim}\left(\mathbf{p_5}\right)=76.6.$

В этом случае все обобщенные оценки оказались равными, а, следовательно, выбор лучшего объекта оказался не возможен. Таким образом, моделирование процедуры агрегирования является важнейшим этапом решения задач принятия решений, в которой в качестве основы используется агрегирование.

Заключение

1) Обобщенные оценки, полученные на основе подхода, основанном на решении оптимизационной задачи по минимизации вариабельности весов, не позволяют сделать однозначный выбор наилучшего проекта строительства.

2) Обобщенные оценки, полученные с помощью OWA-операторов, ассоциированных с вектором весов, который был получен первым методом, основанном на нормальном вероятностном распределении, позволяют сделать однозначный выбор в пользу проекта с вектором оценок $\mathbf{p_1}$.

3) В первом методе оценки, близкие к среднему значению, получили наибольший вес, в то время как, вероятно, несправедливо завышенные/заниженные оценки приобрели меньший вес. При решении данной задачи этот момент сыграл решающую роль в поиске обобщенных оценок, различных по значениям.

4) Операторы, ассоциированные с векторами весов, полученными с помощью этих методов, реализуют компромиссную стратегию агрегирования.

Литература

1. Леденева, Т. М. Агрегирование информации в оценочных системах / Т. М. Леденева, С. Л. Подвальный // Вестник Воронеж. гос. ун-та. Сер. Системный анализ и информационные технологии. – 2016. – № 4. – С. 155–164.

2. Леденева, Т. М. Обзор основных классов операторов порядкового взвешенного агрегирования / Т. М. Леденева, И. Н. Левкина // Вестник Воронеж. гос. ун-та. Сер. Системный анализ и информационные технологии. – 2022. – № 1. – С. 5–31.

3. Xu, Z. S. An Overview of Methods for Determining OWA Weights. / Z. S. Xu // International Journal of Intelligent Systems. – 2005. – vol. 20. – С. 843–865.

4. Sha, X. Y. Elliptical Distributions-Based Weights-Determining Method for OWA Operators / X. Y. Sha, Z. S. Xu, C. Yin // International Journal of Intelligent Systems. – 2019. – vol. 34, no. 5. – С. 858–877.

5. Фиакко, А. Нелинейное программирование / А. Фиакко, Г. Мак-Кормик. – Пер. с англ. – Москва : Изд-во Мир, 1972. – 240 с.

УДК 004.415.532.2, 004.043

Разработка методов и инструментов обработки больших данных по результатам оптимизации процессов тестирования

А. Д. Обыдённый

Воронежский государственный университет

Введение

При разработке программного обеспечения, тестирование играет ключевую роль в обеспечении надежности и функциональности продукта. С ростом сложности современных систем возрастает потребность в методах тестирования, способных обнаруживать и предотвращать широкий спектр потенциальных дефектов.

Один из эффективных и гибких методов тестирования — это тестирование на основе свойств (Property-based testing). Этот подход основан на создании тестовых случаев, ориентированных на свойства программы, что позволяет проводить более глубокие и комплексные проверки, чем традиционные методы.

Однако, с увеличением объемов данных, генерируемых и анализируемых в процессе тестирования на основе свойств, возникает необходимость в разработке эффективных методов и инструментов обработки больших данных (Big Data) для дальнейшего их анализа.

Среди существующих методов решения можно выделить традиционные алгоритмы обработки данных, такие как сортировка, фильтрация и агрегация, которые могут быть адаптированы для работы с большими объемами данных, но эти методы могут столкнуться с проблемами производительности и масштабируемости при работе с огромными наборами данных, характерными для тестирования на основе свойств.

Другой подход заключается в использовании специализированных инструментов и технологий, таких как распределенные системы обработки данных: Apache Hadoop, Apache Spark; базы данных для аналитики больших данных: Apache Cassandra, MongoDB; техники параллельной обработки данных. Эти методы могут обеспечить более высокую производительность и масштабируемость при обработке больших объемов данных, что особенно важно в контексте тестирования на основе свойств.

В данной статье рассматриваются подходы к обработке больших данных, полученных в результате тестирования методом тестирования на основе свойств, и представляются разработанные методы и инструменты.

1. Тестирование на основе свойств

Тестирование на основе свойств (Property-based testing) представляет собой методологию тестирования программного обеспечения, где тесты создаются на основе математических свойств, которые должны выполняться в различных условиях работы программы. Вместо того чтобы создавать ряд фиксированных тестовых случаев, тестирование на основе свойств позволяет описать общие характеристики программы и проверить их для различных входных данных, генерируемых автоматически.

Основная концепция этого метода заключается в том, чтобы выразить ожидаемые свойства программы в виде математических выражений или предикатов, таких как инварианты или соотношения, которые должны оставаться верными в течение всего времени выполнения

программы. Затем генерируются случайные входные данные и проверяется выполнение этих свойств для каждого набора данных.

Таким образом, можно выделить следующие преимущества тестирования на основе свойств:

- Широкое покрытие.
- Автоматизация тестирования.
- Обнаружение скрытых дефектов.

Однако, этот метод также имеет свои ограничения, включая необходимость тщательного определения свойств программы и сложность обработки ошибок, выявленных в процессе тестирования.

2. Анализ больших данных

Основным подходом к обработке больших данных является распределенная обработка, где данные разбиваются на части и обрабатываются параллельно на нескольких вычислительных узлах. Это позволяет ускорить обработку данных и обеспечить масштабируемость системы. Наиболее популярными инструментами для распределенной обработки данных являются Apache Hadoop и Apache Spark.

Один из первых и наиболее известных методов обработки больших данных - это MapReduce (рис. 1). Этот подход был разработан компанией Google для эффективной обработки и анализа данных в распределенной среде. MapReduce разбивает обработку данных на два основных шага: Map и Reduce.

1. Map: Входные данные разбиваются на небольшие фрагменты, которые обрабатываются параллельно на нескольких узлах кластера. Каждый узел выполняет операцию Map, которая преобразует входные данные в набор пар ключ-значение.
2. Shuffle and Sort: Пары ключ-значение собираются вместе, сортируются и перемешиваются по ключу, чтобы гарантировать, что все значения с одним и тем же ключом попадают на один узел.
3. Reduce: Объединение всех значений с одинаковыми ключами и последующая агрегация данных.

Помимо MapReduce, существуют и другие методы обработки больших данных: Spark, Hadoop, Storm и Flink.

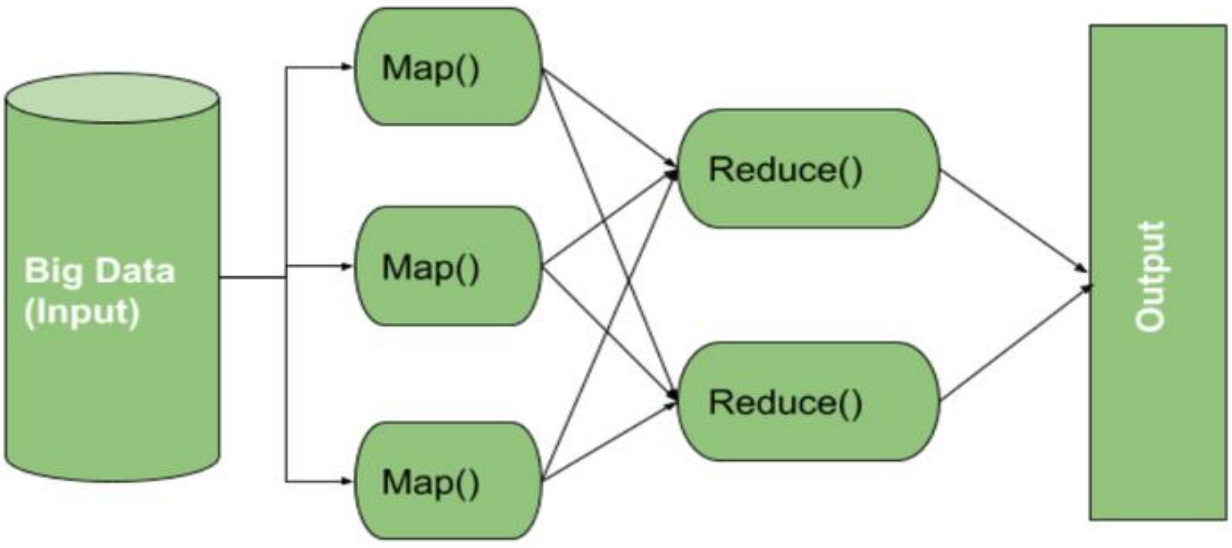


Рис. 11. Модель вычислений MapReduce

3. Архитекрута тестирования на основе свойств и обработки больших данных

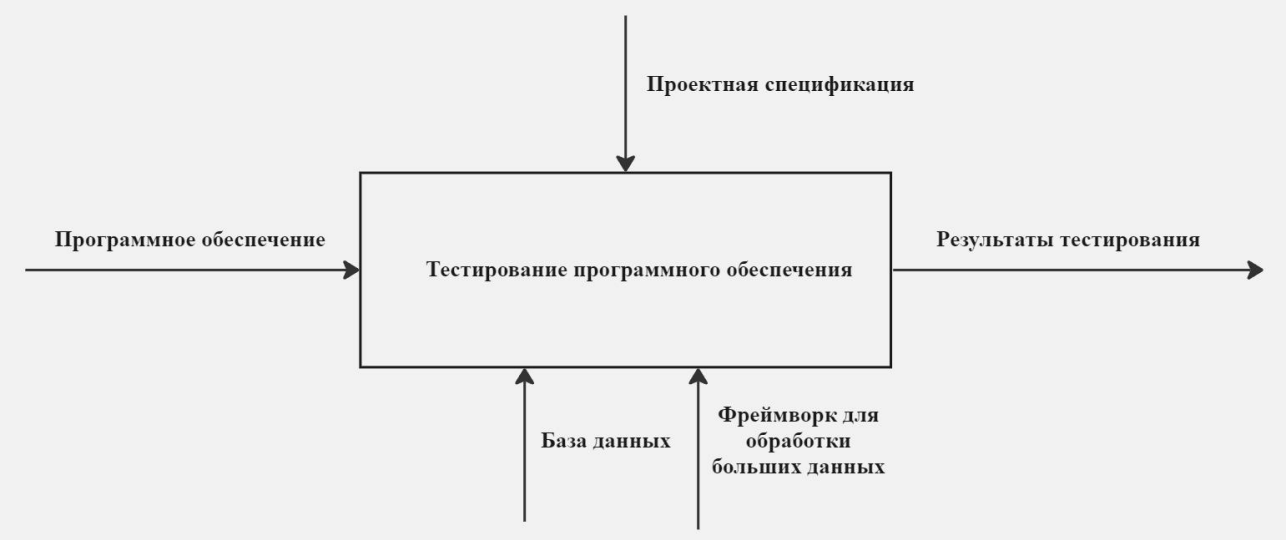


Рис. 2. Диаграмма IDEF0 проведения тестирования

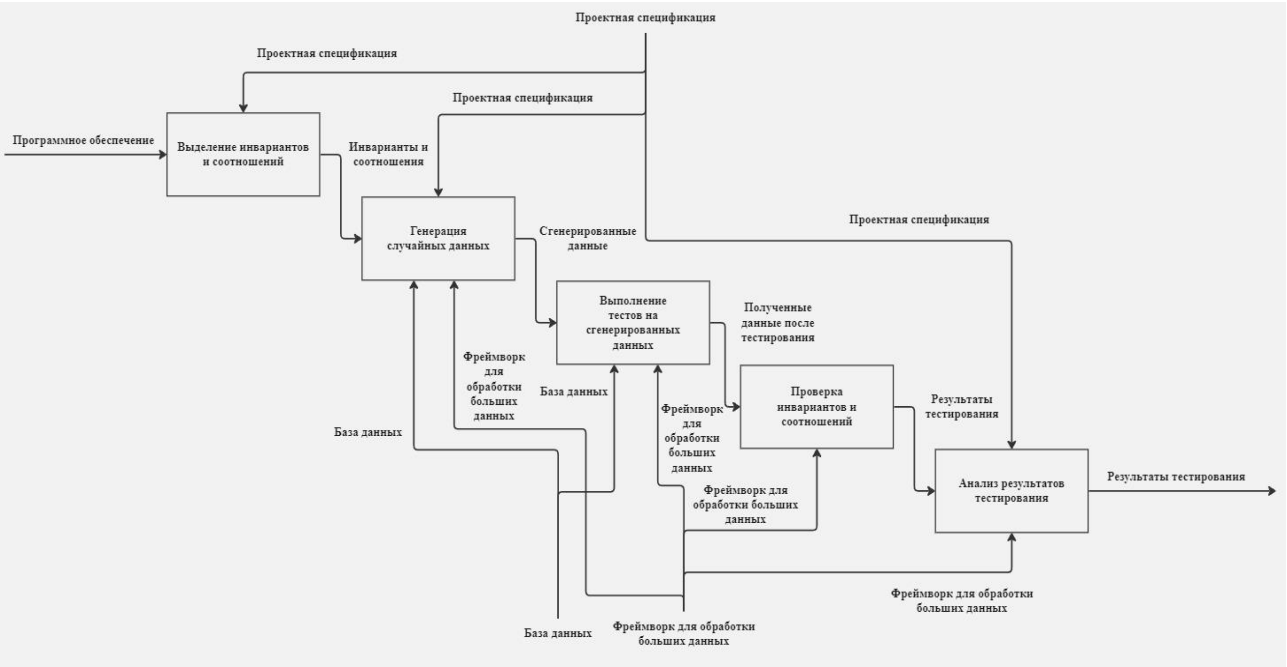


Рис. 3. Декомпозиция диаграммы IDEF0 проведения тестирования

Заключение

В данной статье были рассмотрены методы и инструменты обработки больших данных по результатам тестирования методом тестирования на основе свойств. Этот метод тестирования представляет собой иной подход к тестированию программного обеспечения, который позволяет автоматизировать процесс создания тестов, основанных на математических свойствах программы. Кроме того, этот метод позволяет проводить более обширные и систематические проверки, что существенно повышает качество и надежность

программного продукта. Успешная реализация этого метода требует эффективной обработки и анализа больших объемов данных, сгенерированных в процессе тестирования.

Литература

1. Дастин, Э. Тестирование программного обеспечения. Внедрение, управление и автоматизация / Э. Дастин, Д. Рэшка, Д. Пол; Пер. с англ. М. Павлов. - М.: Лори, 2013. - 567 с.
2. Джефф Рэшка, Элфрид Дастин, Джон Пол. Автоматизированное тестирование программного обеспечения. Внедрение, управление, эксплуатация. Издательство Лори, 2012
3. Фрэнкс, Б. Укрощение больших данных: как извлекать знания из массивов информации с помощью глубокой аналитики / Б. Фрэнкс ; перевод с английского А. Баранов. — Москва : Манн, Иванов и Фербер, 2014. — 352 с. — ISBN 978-5-00057-146-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/62154 (дата обращения: 18.04.2024). — Режим доступа: для авториз. пользователей.
4. Gantz, J. The digital universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East - United States / J. Gantz, D. Rainsel // IDC Country brief, 2013.
5. Myers, G. J., Badgett, T., Sandler, C. (2012). The Art of Software Testing (Vol. 3rd ed). Hoboken, N.J.: Wiley. Retrieved from http://search.ebscohost.com/login.aspx?direct=truesite=edslivedb=edsebkAN=396276

УДК 004

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОБРАБОТКИ ДАННЫХ ОБ УРОВНЕ САХАРА В КРОВИ

С. В. Овсянников, Т. В. Курченкова

Воронежский государственный университет

Введение

Информационные технологии оказывают значительное влияние на сферу здравоохранения и помогают пациентам с диабетом. Системы непрерывного мониторинга глюкозы, мобильные приложения и устройства для самоконтроля позволяют пациентам с диабетом отслеживать уровень сахара в крови, физическую активность и другие показатели.

Диабет — это хроническое заболевание, которое оказывает существенное воздействие на способность организма использовать глюкозу (сахар) в крови. В последние десятилетия заболеваемость диабетом стремительно растет, и это становится острой проблемой для общества. Согласно данным, с 1980 по 2024 год количество людей, страдающих диабетом, выросло с 108 миллионов до 422 миллионов человек [1]. Особенно остро проблема диабета влияет на страны с низким и средним уровнем дохода.

Для контроля уровня сахара в крови может использоваться глюкометр непрерывного действия (ГНД) — это небольшое носимое устройство, которое измеряет уровень глюкозы (сахара) каждые несколько минут.

1. Постановка задачи

Разработать мобильное приложение мониторинга уровня сахара в крови. Приложение должно содержать следующую функциональность:

1. Подключение пользователя к API поставщика данных.
2. Настройка индивидуальных предпочтений для визуализации данных.
3. Построение графиков: процента времени в установленном диапазоне уровня сахара; уровня глюкозы; коэффициента вариации значений глюкозы; информации об отображении изменчивости значений глюкозы.

2. Подготовка и выгрузка данных

Данные об уровне сахара будут выгружаться из API производителей ГНД: *Nightscout* и *Dexcom*. Производители предоставляют структуру набора данных и сами данные для выгрузки.

После успешного прохождения стартового окна пользователь попадает на главный экран, где выполняется начальная загрузка данных. Стандартным периодом загрузки является неделя. После успешной загрузки запускается фоновое обновление данных с интервалом в 5 минут, при котором выполняется три попытки загрузки новых данных из сервиса. Независимо от успешности загрузки, таймер продолжает свою работу и с периодичностью 5 минут происходит попытка обновления данных. При выборе графика за 30 дней выполняется проверка данных: за какой промежуток имеются и за какой была попытка их запросить. Исходя из этих данных запрашиваются оставшиеся промежутки. При последующих переключениях

между периодами, предварительно выполняется проверка о наличии данных в базе данных (используется пакет *drift*) [2]. Если данные имеются, то они отображаются сразу (без необходимости загрузки), иначе выполняется загрузка.

3. Функциональность и используемые сервисы

Главный экран приложения состоит из следующих блоков:

1. *Appbar*. Содержит в себе приветственную строку и динамическую строку, которая меняет текст в зависимости от показателей уровня сахара в крови у пользователя.
2. Панель переключателей с периодами для отображения графиков: 7 дней, 30 и 90 дней.
3. Графики: процент времени в установленном диапазоне уровня сахара; уровень глюкозы; коэффициент вариации значений глюкозы; информация об отображении изменчивости значений глюкозы.

Графики строятся при помощи библиотек: fl_chart, syncfusion_flutter_charts, charts_flutter.

Стартовое окно приложения и сервисы, которые в нем используются:

1. *ICredentialsService*: отвечает за хранение и обработку учетных данных для *Dexcom*, *Nightscout* [3, 4].
2. *IStorageRepository*: репозиторий для хранения/получения данных.
3. Учетные данные для подключения к провайдеру данных (*IServiceCredentials*).
4. Состояния стартового окна приложения (*OnboardingStateSnapshot*).
5. Общие настройки приложения (*AppSettings*).
6. Профиль пользователя (*UserProfile*).
7. Ранее запрошенный с сервиса пользователем диапазон глюкозы (`*Map<String, String>*`).
8. *IAppSettingsService*: сервис для получения/сохранения текущих настроек приложения.
9. *IProfileService*: сервис для получения/сохранения состояния профиля пользователя данных.

Информационные *push*-уведомления.

Основное взаимодействие системы уведомлений с приложением происходит через *PersonalizedNotificationsService* [5]. Есть два типа *push*-уведомлений. Первый тип *push*-уведомлений регистрируется в зависимости от активности пользователя в приложении (например: по субботам всегда будет приходить *push* по теме «Прошла рабочая неделя, пора посмотреть как ваше здоровье»). Второй тип *push*-уведомлений отправляется при достижении показателей по графикам определённых значений (очень высокая вариация показаний глюкозы).

4. Интерфейс пользователя

В стартовом окне приложения (рис. 1) находятся инструкции по подключению к поставщикам данных. Стартовое окно состоит из нескольких экранов, которые отображаются только во время первого открытия приложения на устройстве.

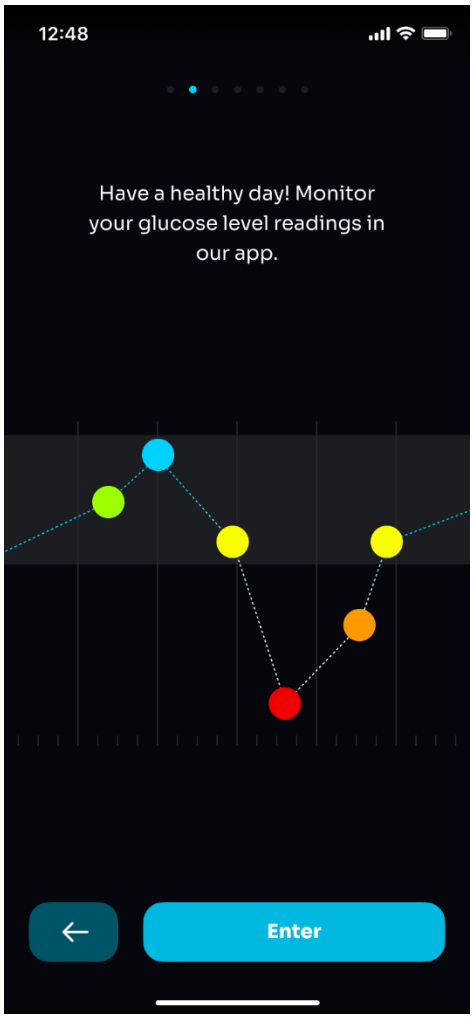


Рис. 1. Стартовое окно

Главный экран — первый раздел, который доступен пользователю после входа в приложение. На экране (рис. 2) сверху изображены: аватар пользователя, временные интервалы. В разделе посередине находятся графики времени в установленном диапазоне глюкозы, уровня глюкозы, коэффициента вариации уровня глюкозы. В нижней части экрана находится график изменчивости значений глюкозы (рис. 3).

Рис. 2. Главный экран. Отображение времени в установленном диапазоне глюкозы

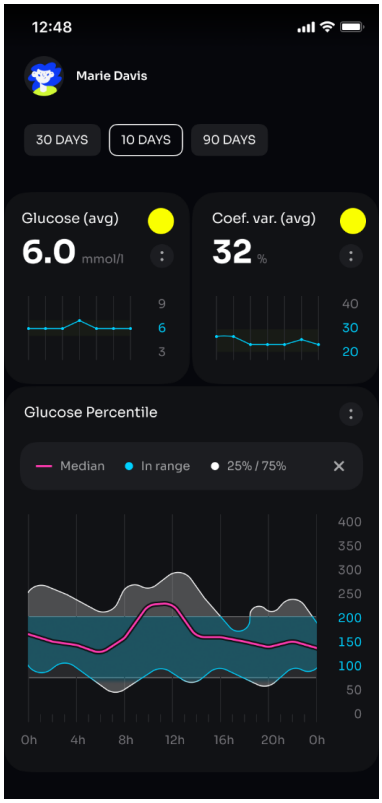


Рис. 3. Главный экран. Отображение изменчивости значений глюкозы

Пользователь, меняя настройки стандартных значений в легенде графика, может выставить удобный и привычный формат отображения данных. Если пользователю необходимо изменить источник данных, это можно сделать через настройки.

Заключение

Было разработано Flutter приложение для мониторинга уровня сахара в крови. Мобильное решение включается в себя минимально необходимый набор функциональности для его комфортного использования. Приложение будет полезно людям с диабетом, использующим глюкометры беспрерывного действия и желающим отслеживать свой уровень глюкозы на долгосрочном промежутке времени.

Литература

1. Материалы по теме диабет ВОЗ – URL: https://www.who.int/ru/news-room/fact-sheets/detail/diabetes (Дата обращения: 15.03.2024)

2. Flutter: структура проекта. – URL: https://education.yandex.ru/handbook/flutter/article/flutter-struktura-proekta (Дата обращения: 15.03.2024)

3. API Dexcom. – URL: https://developer.dexcom.com/home (Дата обращения: 15.03.2024)

4. Документация по API Nightscout. – URL: https://nightscout-test.readthedocs.io/en/latest/Nightscout/EN/Technical%20info/api.html (Дата обращения: 15.03.2024)

5. Гайд по сборке компонентов. – URL: https://www.figma.com/community/file/1220795227825907928 (Дата обра- щения: 15.03.2024)

УДК 519.81, 519.86

ОСОБЕННОСТИ СТРАТЕГИЧЕСКОГО ПЛАНИРОВАНИЯ В СТРОИТЕЛЬНОЙ ОТРАСЛИ

Е.В. Орлова

Воронежский государственный университет

Введение

В современном мире, строительная отрасль играет важную роль в экономическом развитии общества. Однако, чтобы успешно действовать на этом конкурентном рынке, строительным и инженерным компаниям необходимо иметь четкий план, который бы обеспечивал им конкурентное преимущество и помогал достичь целей. Стратегическое планирование в строительстве является неотъемлемой частью успешного бизнеса. Это процесс определения долгосрочных целей компании, разработки стратегий и тактик для их достижения, а также контроля и корректировки планов в соответствии с изменяющейся средой. Основными целями стратегического планирования в строительстве являются оптимизация процессов, повышение эффективности и конкурентоспособности компании, улучшение качества предоставляемых услуг и минимизация рисков.

1. Особенности проектов в строительной отрасли

Одними из типовых проектов, для которых применяется сценарный анализ, являются проекты в газовой сфере гражданского строительства. Проектирование магистральных газонефтепроводов представляет собой сложный комплекс работ, включающий предварительный сбор, изучение и обобщение различной информации, проведение большого объема полевых работ, принятие и согласование решений, связанных с выбором конфигурации линейной части трассы, переходов через естественные и искусственные препятствия, площадок для сооружения компрессорных и насосных станций, установлением закономерностей природо-грунтовых факторов и их взаимодействия с проектируемыми объектами.

Действующими методическими документами предусмотрено, что главным предпроектным документом, в котором обосновывается перечень, экономическая эффективность и техническая возможность проектирования и строительства объектов трубопроводного транспорта, являются стратегия развития и размещения газовой, газоперерабатывающей и химической промышленности. На базе которых, составляется наиболее рациональная схема потоков газа, по которой в дальнейшем осуществляется проектирование объектов транспорта.

Помимо стратегии развития на предпроектной стадии разрабатывается технико-экономическое обоснование (ТЭО). В составе этих документов разрабатываются материалы с необходимыми расчетами, которые обосновывают целесообразность проектирования, строительства, расширения или реконструкции трубопроводов и имеющихся на нем объектов,

а также основные технико-экономические показатели, характеризующие эти работы. Они должны предусматривать максимальное использование действующих магистральных газонефтепроводов с учетом их расширения, имея в виду, что данные направления - наиболее эффективный путь увеличения производственных мощностей. В процессе их разработки определяется потребность в нефти и газе по экономическим районам, тяготеющим к трассе, оцениваются направления их использования путем анализа существующих и перспективных топливно-энергетических балансов и прогрессивных норм расхода различных видов топлива для производства продукции по наиболее топливоемким отраслям промышленности.

2. Характеристики проектов в газовой сфере

В проекте закладываются и разрабатываются главные технологические и конструктивные решения, определяется окончательная стоимость объекта. Поэтому для оценки проектов при выборе для реализации, рассматривается технико-экономическое обоснование (ТЭО), Оно представляет собой пояснительную записку, содержащую характеристику сырьевой базы (месторождения или их группы), возможные сроки ее освоения и поэтапной эксплуатации, перспективы использования продукта первоочередными и потенциальными потребителями с учетом неравномерности спроса на газ и газопродукты, параметры трубопроводов, оптимальные направления трасс (по предварительным данным), данные о необходимых материальных и финансовых ресурсах, показатели режима эксплуатации, преимущества перед другими источниками топливоснабжения и промышленной переработки.

Одной из важнейших задач ТЭО является определение основных технико-экономических показателей, к числу которых относятся объемы капитальных вложений с учетом сопряженных затрат в смежные отрасли промышленности, численность работников, производительность труда, ориентировочный размер себестоимости транспорта газа, удельные расходы топлива, электроэнергии и других ресурсов, а также показатели, характеризующие экономическую эффективность капитальных вложений.

При разработке ТЭО проводится принципиальное согласование общего направления и альтернативных вариантов с Госгортехнадзором, Минсельхозом (по вопросам пересечения территорий мелиорации и орошения) и другими заинтересованными организациями. ТЭО проходит экспертизу и после этого заказчик, основываясь на ТЭО, выдает одному из специализированных проектных институтов задание на проектирование магистрального трубопровода. Институт в этом случае становится генеральным проектировщиком.

В соответствии с действующим порядком проектирование осуществляется в одну или две стадии. Одностадийное проектирование используется в том случае, когда предполагается сооружение объектов, строительство которых может быть осуществлено по типовым и ранее использовавшимся индивидуальным проектам, а также для технически несложных строек. В этом случае разрабатывается рабочий проект. В остальных случаях проектирование осуществляется в две стадии, предусматривающие разработку проекта и рабочей документации.

Алгоритм отбора и оценка проектов для реализации в газовой сфере представлен на рис. 1.

Рис. 1. Алгоритм отбора и оценка проектов для реализации в газовой сфере

3. Сценарный анализ как основа стратегического управления

С целью повышения эффективности функционирования компаний, в том числе и газовой сферы), в основе стратегического управления должно лежать стратегическое планирование. Основными инструментами аналитического стратегического планирования являются ситуационный и сценарный анализ. Ситуационный анализ позволяет определить текущее положение, маркетинговые возможности и проблемы, с которыми предприятие может столкнуться в будущем.

Сценарный анализ применяется для оценки картины будущего компаний. Однако отрасли не хватает инструментария, позволяющее оценивать стратегию развития не только в целом, но и отдельные проекты. В отличие от прогнозов, в сценариях учитываются не только количественные показатели, полученные расчетным путем, но и качественные, полученные в результате обработки мнений экспертов.

Процесс формирования сценария при сценарном подходе включает в себя пять основных этапов. Первым этапом является определение главных целей и задач анализа, установление критериев успеха и выбор методов и инструментов, необходимых для его

проведения. Затем следует сбор и обработка данных, касающихся исследуемой области или ситуации. Он может включать в себя анализ статистических данных, проведение интервью, исследование литературы. Далее необходимо проанализировать полученные данные и выявить основные тренды, закономерности и проблемы. На основе этого анализа формулируются различные сценарии развития событий, а также возможные решения и стратегии действий. Для каждого сценария проводится оценка рисков и возможных последствий. Заключительным этапом является составления отчета о проведенном анализе. В качестве сценариев в строительной сфере выступают те или иные проекты, которые планируются для реализации. Выбор того или иного проекта зависит от множества факторов, которые на первый взгляд трудно разделяемы, в том числе по приоритетам. К тому же, на выбор проекта влияет мнение множества заинтересованных сторон.

Постановка задачи оценки проектов в газовой сфере представляет собой задачу многокритериального выбора. Ставится задача, например: имеется объект, в рамках которого предстоит выбор одного из нескольких проектов газификации; заказчик в отношении проекта может иметь интересы в отношении проекта как административного, так и отраслевого характера; выбор необходимо осуществить по ряду критериев. Сложность задачи связана с выбором по большому числу критериев, критерии носят как количественный, так и качественный характер. В такой постановке это задача многокритериального выбора на ограниченном множестве, т.к. количество альтернатив всегда ограничено представленными проектами.

Сложность такой задачи связана с оценкой эффективности проектов для реализации, в частности, как в выборе критериев, так и оценке по этим критериям. Выбор критериев связан с формированием модели оценки, позволяющей учесть специфику строительной сферы и проектов газовой сферы. При этом нужно предусмотреть, чтобы сами критерии, могли оцениваться как количественными показателями, так и качественными. Для такой задачи применим метод анализа иерархии. Иерархическая модель оценки сценариев наиболее полно отображает систему планирования, характерную для предприятий строительной отрасли и газовой сферы.

Модель оценки проектов газовой сферы, планируемых для реализации в рамках программы развития газового хозяйства региона, представлена на рис. 2. Она представляет собой декомпозицию проблемы выбора, что с позиции заказчика упрощает процесс выбора. Нижний уровень иерархии представляет собой критерии, по которым осуществляется выбор, т.е. характеристики проектов согласно ТЭО. Исходными данными являются: множество проектов, предлагаемых для реализации, критерии оценки. На выходе - ранжированное множество проектов, предлагаемых для реализации. В ходе решения задачи предполагается сбор экспертного мнения, которое будет положено в основу расчета оценок. Решение задачи сводится к получению весов важности каждого уровня иерархии, свертке по ним всей иерархии и нахождению весов важности каждой из альтернатив (проектов).

Заключение

Модель оценки, разработанная на базе метода анализа иерархий, может использоваться в алгоритме отбора проектов и стать частью системы поддержки, как инструмент, упрощающий принятие решения и позволяющий рассмотреть большее количество критериев выбора. В свою

очередь, это делает возможность принятия заинтересованными лицами более объективного решения и в кратчайшие сроки.

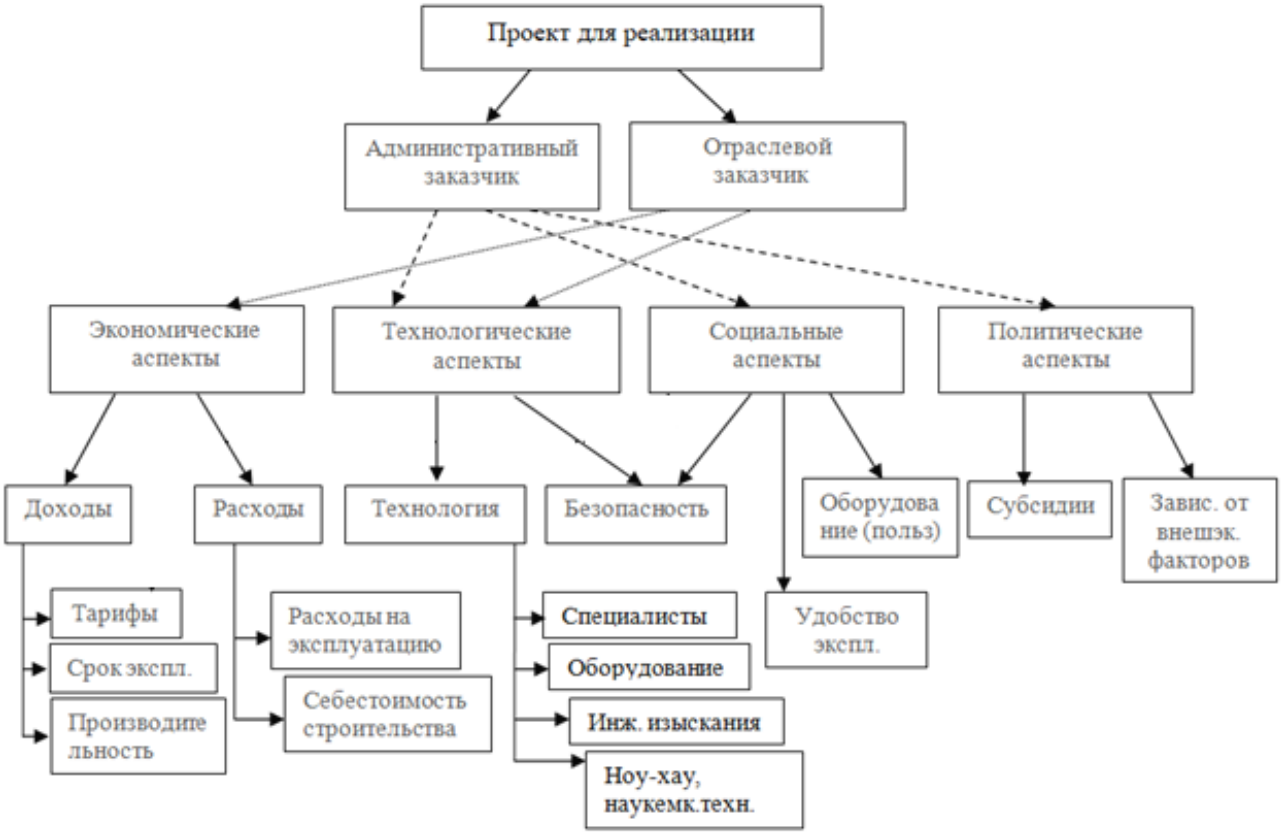


Рис. 2. Модель оценки проектов газовой сферы

Литература

1. Асаул, А.Н. Стратегическое планирование развития строительной организации / А.Н. Асаул. – Санкт – Петербург : СПбГАСУ, 2010. – 163 с.

2. Макаров, И.М. Теория выбора и принятия решений / И.М. Макаров и др. – Москва : Наука. 1982. – 330 с.

3. Петров, А.Н. Стратегическое планирование развития предприятия / А.Н. Петров – Санкт-Петербург : Изд-во СПбУЭФ. – 2009. – 496 с.

4. Ухлова, В. В. Метод анализа иерархий как средство поддержки принятия решений в стратегическом аналитическом планировании / В.В. Ухлова, Т.В. Азарнова, О.Ю. Пономарева // Экономическое прогнозирование: модели и методы: материалы IX международной научно-практической конференции: под общ. ред. В.В. Давниса, В.И. Тиняковой. – Воронеж. – 2013. – С.9-12.

5. Ухлова, В. В. Адаптация метода анализа иерархий для возможности проведения сценарного анализа проектов развития предприятий газовой сферы / Г.Н. Мартыненко, В.И. Лукьяненко, В. В. Ухлова // Системы управления и информационные технологии, - Воронеж, 2021. – №1(83), – С. 43-48.

УДК 004.8

Большие языковые модели и их оценка

И. Р. Осипов

Воронежский государственный университет

Введение

На сегодняшний день широкое распространение получили большие языковые модели, способные решать широкий спектр задач по обработке естественного языка. Подобному прогресс стал возможен благодаря появлению больших языковых моделей – моделей, обладающих огромным количеством параметров и обученных на большом объеме данных с целью решения различных задач. Они используются для анализа запросов с целью выдачи корректного результата, для получения корректного перевода с одного язык на другой. Все это помогает людям на повседневной основе, в связи с этим стоит вопрос о том, как понимать, насколько хорошо модель справляется с той или иной задачей, ведь ее правильная работа зависит от процесса тонкой настройки модели, которую тяжело проводить без точных измерений ее производительности. На сегодняшний день существует множество подходов к оценке больших языковых моделей, однако они не лишены недостатков. Так, оценки могут не учитывать те или иные факторы, например, понимания контекста запроса.

Таким образом, целью данной работы является усовершенствование существующих подходов к оценке языковых моделей, что включает в себя изучения современных больших языковых моделей, их оценок, а также влияния различных параметров настройки моделей на оценку.

1. Большие языковые модели

1.1. Трансформеры

Большая языковая модель — это языковая модель, состоящая из нейронной сети со множеством параметров (обычно миллиарды весовых коэффициентов и более), обученной на большом количестве неразмеченного текста с использованием обучения без учителя. Особенностью таких моделей является то, что, обучаясь на больших объемах данных, модели изучая статистические взаимосвязи в тексте, а также выучивать общие знания о мире, благодаря чему они способны решать широкий спектр задач, будь то генерация или классификация.

История больших языковых трансформеров начинается с появления архитектуры «трансформер», представленной в статье «Attention Is All You Need». Изначально предложенные в качестве решения для улучшения задачи машинного переводы, в дальнейшем модифицированные варианты трансформеров стали применятся для решения множества задач. В своей основе трансформер имеет архитектуру кодировщик-декодировщик, которые используют слои внимания. Архитектура трансформера продемонстрирована на рисунке 1.

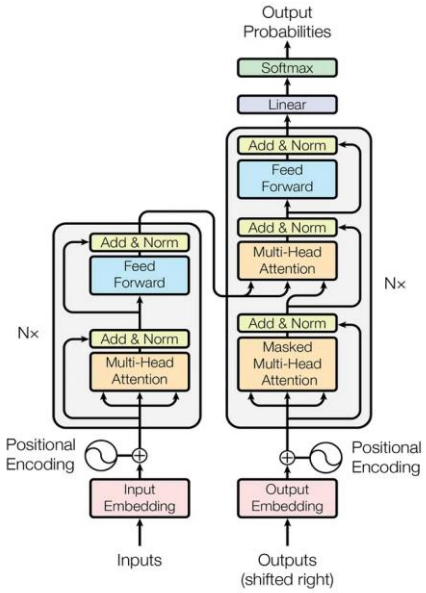


Рис. 1 Архитектура модели трансформера

Как было упомянуто выше, трансформер состоит из кодировщика и декодировщика, которые в свою очередь имеют содержат слои энкодеров и декодеров. Каждый из энкодеров содержит несколько вложенных слоев. Первым таким слоем является слой внутреннего многоголового внимания, задача которого заключается в том, чтобы узнать, на какие позиции во входных данных со значениями следует обратить внимание для каждой позиции в запросе. Затем полученные результаты передаются в слой прямого распространения (полносвязанный), применяемый отдельно к каждой позиции, после чего результаты передаются следующему кодировщику. Декодеры имеют схожую структуру, но между слоем внутреннего внимания и слоем прямого распространения имеют слой внимания, благодаря которому декодер имеет возможность фокусироваться на важных частях входящего предложения, а также каждый последующий слой декодировщика после начального на вход получает ключи/значения из кодировщика, а запросы из предыдущего слоя декодировщика.

Благодаря механизму внимания, модель способна понимать отношения между словами. Слой внутреннего внимания по мере того как модель обрабатывает каждое слово, внутреннее внимание позволяет модели взглянуть на другие позиции входной последовательности и найти подсказку, помогающую лучше закодировать данное слово. На текущий момент в трансформерах используется вариация слоев внимания под названием слои многоголового внимания, которые улучшают производительность модели за счет повышения ее способности фокусироваться на различных позициях и дополнительных подпространств представления.

Все модели на базе трансформеров используют подход переноса обучения, благодаря которому модель может быть использована для решения задач, для которых ее изначально не обучали. Так, изначально модель обучается на больших массивах данных с целью получить комплексное понимание языка, после чего модель может быть тонко настроена для решения конкретной задачи путем дополнительного обучения на необходимых данных.

Одной из самых популярных разновидностей трансформеров является GPT.

1.2. GPT

Основной шрифт Times New Roman, размер 12 пт. Допускается только автоматическая расстановка переносов (рис. 1).

GPT (Generative pre-trained transformer) – модель на основе транфсормеров, имеющая в своей основе только декодировщик. Особенностью данной модели является то, что она использует в качестве слоя внутреннего внимания модификацию, названную «маскированным внутренним вниманием» — такой слой внимания, в котором последующие слова маскируются с помощью изменения процесса подсчета внутреннего внимания и блокирования ин-формации от токенов, находящихся справа от той позиции, которая высчитывается в данный момент, из-за чего модель не способна знать, что идет после текущего обрабатываемого токена. После обучения генеративной модели на немаркированных данных, модель можно настроить на решение определенной задачи с использованием небольшого количества маркированных данных. Благодаря такому подходу, GPT, созданная изначально для генерации текста, способна решать ряд задач по обработке естественного языка, например, проводить классификацию текста.

Пример архитектуры GPT представлен на рисунке 2.

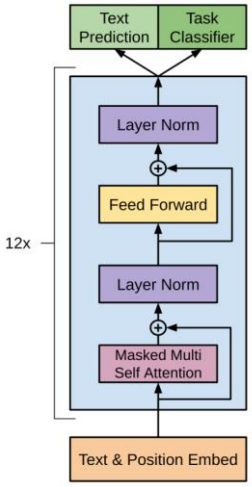


Рис. 2 Архитектура модели GPT

На текущий момент GPT имеет следующие версии:

1) GPT-1. Первая версия этой модели состояла из 12 слоёв со 117 миллионами параметров, которая была обучена на корпусе BookCorpus, состоящим из 7000 книг размером 4.5 ГБ. Данная модель имела проблему с генерацией длинных текстов, но на отдельных задачах с использованием тонкой настройки могла добиться хороших результатов. Максимальный размер контекста у GPT-1 – 512 токенов.
2) GPT-2. Улучшенная версия GPT-1, которая выросла в размерах и получившая 48 слоёв с 1.5 миллиардами параметров, обученная на корпусе WebText, состоящим из 40 ГБ текста, взятого с различных интернет-ресурсов. В отличие от своей предшественницы, GPT-2 способна успешно работать с текстом большой длины, а также успешно решать некоторые языковые задачи без предварительной обработки. Максимальный размер контекста у GPT-2 – 1024 токенов.
3) GPT-3. Доработанная версия GPT-2, получившая 175 миллиардов параметров, которая была обучена на огромном корпусе данных общим объемом в 570 ГБ текста, состоящем из дата-сетов CommonCrawl, WebText, двух книжных корпусов и английской Википедии. Из архитектурных изменений — только немного оптимизировали attention. Максимальный размер контекста у GPT-3 – 2048 токенов. Имеет вариацию GPT-3.5, которая обладая той же архитектурой, что и GPT-3, за счет измененного метода обучения смогла добиться лучших результатов.
4) GPT-4. Новейшая версия модели GPT, получившая возможность работы с изображениями в качестве входных данных. Также модель способна читать,

анализировать или генерировать до 25 000 слов текста, что является огромным скачком в сравнении с предыдущими версиями. Максимальный размер контекста – до 32 768 токенов.

2. Оценки языковых моделей

На текущий момент существуют множество оценок языковых моделей. К одним из самых популярных оценок относят кросс-энтропию и перплексию, которые используют для общей оценки языковой модели без учета задачи.

Кросс-энтропия, представляет собой метрику, определяющую собой степень расхождения между вероятностным распределением полученной языковой модели и распределением моделируемого естественного языка. Чем больше значение кросс-энтропии, тем больше разница между двумя распределениями. Данную метрику можно интерпретировать следующим образом: чем меньше значение кросс-энтропии, тем меньше неопределенность модели в предсказании следующего элемента последовательности.

Формула данной метрики имеет следующий вид:

$$H(p,q) = -\sum_i p(x_i)\log_2 q(x_i),$$

где p – реальное распределение языка, q – распределение языка, смоделированное на данных.

Перплексия представляет собой статистическая мера, которая показывает степень неопределенности модели при предсказании следующего элемента последовательности. Существует два подхода для определения перплексии: определение ее через кросс-энтропию

$$PPL(p,q) = 2^{H(p,q)}$$

или как обратную вероятность набора $W = w_1, \ldots, w_n$, нормализованного по количеству элементов

$$PPL(W) = \sqrt[n]{\frac{1}{\prod_i p(w_i)}}.$$

Также перплексию можно рассматривать как взвешенный коэффициент ветвления, который представляет собой среднее количество слов, возможных после данного слова. Интерпретировать это можно следующим образом: значение перплексии модели *PPL(W)* аналогично тому, что при предугадывании следующего значения модели необходимо выбрать из *PPL(W)* значений.

По аналогии с кросс-энтропией, чем меньше значение перплексии, тем лучше модель предсказывает следующее значение последовательности. На практике зачастую используется перплексия ввиду того, что она лучше отображает изменения в степени неопределенности модели.

3. Оценка моделей на практике

В качестве оцениваемых моделей возьмем обученную на русском языке вариацию GPT-3 от Сбербанка под названием ruGPT-3 в версиях small, который включают в себя 125 миллионов параметров, а в качестве наборов данных для обучения возьмем датасет EMERCOM, который включает в себя обращения в службу психологической поддержки и ответы от психологов. Пример данных в датасете изображен на рисунке 3.

```
{
     "id":100,977
     "question":{"author": "Додо", "title": "Когда я избавлюсь от этих &quot;подростковых депрессий&quot; ?", "description": "Мне 17. Когда я избавлюсь этой \"подростковой депрессии\"? Могу я избавиться от неё ещё в подростковом возрасте? Что делать? Я хочу быть нормальным. Когда пройдёт это состояние?"},
     "answer":{"author": "Консультант службы", "content": "Здравствуйте!Вы пытаетесь понять, есть ли возможность справиться с подростковой депрессией не в далёкой перспективе, а в ближайшей, и что для этого необходимо сделать. В первую очередь хочется поддержать Ваше желание найти помощь в этом вопросе. К сожалению, Вы не написали о том, диагностировал врач это расстройство или это Ваше предположение. Прежде всего, чтобы начать борьбу с депрессией, необходимо понимать, какая причина её вызвала. В соответствии с этим уже даются рекомендации и при необходимости назначается лечение. Я рекомендую Вам обратиться на телефон доверия, чтобы детально обсудить с психологом Вашу проблему и рассмотреть возможные варианты помощи: 8-800-200-01-22 (круглосуточно, бесплатно).Берегите себя!С уважением, Ольга"}
   }
```


Рис. 3 Данные в датасете

Выделив вопросы и ответы, преобразуем его в подходящий для модели формат, пример данных которого изображен на рисунке 4.


```
<s>Вопрос: Когда я избавлюсь от этих &quot;подростковых депрессий&quot; ?", "description": "Мне 17. Когда я избавлюсь этой \"подростковой депрессии\"? Могу я избавиться от неё ещё в подростковом возрасте? Что делать? Я хочу быть нормальным. Когда пройдёт это состояние?
   Ответ: Вы пытаетесь понять, есть ли возможность справиться с подростковой депрессией не в далёкой перспективе, а в ближайшей, и что для этого необходимо сделать. В первую очередь хочется поддержать Ваше желание найти помощь в этом вопросе. К сожалению, Вы не написали о том, диагностировал врач это ра-стройство или это Ваше предположение. Прежде всего, чтобы начать борьбу с депрессией, необходимо понимать, какая причина её вызвала. В соответствии с этим уже даются рекомендации и при необходимости назначается лечение. Я рекомендую Вам обратиться на телефон доверия, чтобы детально обсудить с психологом Вашу проблему и рассмотреть возможные варианты помощи: 8-800-200-01-22 (круглосуточно, бесплатно).Берегите себя!С уважением, Ольга</s>
```


Рис. 4 Преобразованные данные

После преобразования данных, преобразовываем полученный набор данных в 2 набора данных для обучения с количеством примеров, равным 5 тысячам и 5 сотням.

При обучении будем настраивать следующие параметры:

1) Функция оптимизации: Adam или Adafactor;
2) Количество эпох: 5 или 10;
3) Объем набора данных для обучения: 5 тысяч и 5 сотен примеров.

Перед обучением модель на наборе данных имеет значение перплексии, равное 26.7916. В зависимости от параметров, которые были настроены при обучении, значения перплексии у обученных моделей получили следующие результаты:

1) Функция обучения: Adam, количество эпох: 5, объем набора данных для обучения: 5 сотен примеров. Перплексия равна 22.92726. Сгенерированный текст: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Вы пишите о том, что Вам сейчас непросто, Вы испытываете трудности в поиске себя, в';
2) Функция обучения: Adam, количество эпох: 10 эпох, объем набора данных для обучения: 5 сотен примеров. Перплексия равна 27.99728. Сгенерированный текст: 'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что Вам сложно справляться со своими эмоциями, Вам сложно';
1) Функция обучения: Adafactor, количество эпох: 5 эпох, объем набора данных для обучения: 5 сотен примеров. Перплексия равна 22.59419. Сгенерированный текст: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Вы пишете о том, что Вам сейчас непросто. Вы переживаете утрату близкого человека,';
3) Функция обучения: Adafactor, количество эпох: 10 эпох, объем набора данных для обучения: 5 сотен примеров. Перплексия равна 27.4007. Сгенерированный текст:

'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что Вам сложно жить, Вы испытываете чувство одиночества';

4) Функция обучения: Adam, количество эпох: 5, объем набора данных для обучения: 5 тысяч примеров. Перплексия равна 17.82776; количество эпох: 5, объем набора данных для обучения: 5 тысяч примеров. Сгенерированный текст: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Вы обратились в службу экстренной психологической помощи МЧС'.

Исходя из полученных результатов, можно утверждать, что дополнительное обучение действительно помогает модели генерировать текст, соответствующий задаче, причем можно заметить, что самый лучший результат достигается модель, которая дообучалась на большом наборе данных. Исходя из определения перплексии, можно сделать вывод, что благодаря большему количеству текстовых данных, неопределенность при генерации последующего слова в ответе уменьшилась, и если изначальная модель выбирала из приблизительно 27 слов при генерации последующего токена, то дообученная версия уже выбирает из 18. При этом также можно заметить, что увеличение количества эпох сказывается негативно, и выбор следующего токена модели становится делать чуть сложнее, чем предобученной версии. Однако можно заметить, что низкий показатель перплексии не значит, что модель выдаст самый лучший ответ: так на запрос «Что сделать чтобы захотелось жить», обученная версия с самой лучшей оценкой выдает ответ «Вы обратились в службу экстренной психологической помощи МЧС», что хоть и лучше, чем ответ от модели без обучения, но явно не является релевантным ответом. Можно заметить, что более-менее соответствующий вопросу вывод дает модель из пункта 4), которая имеет достаточно большое значение перплексии.

Попробуем настроить различные параметры для генерации текста и посмотрим, как это повлияет на результат. Настраивать будем модель из пункта 5), поскольку она была самой близкой к релевантному ответу. Будем использовать различные методы генерации текста: Greedy Search (жадный поиск, всегда берется элемент с наибольшей вероятностью), Beam Search (лучевой поиск, создание нескольких последовательностей на основе наиболее вероятных токенов с выбором последовательности, которая имеет наибольшую правдоподобность), Top-K Sampling (отбор следующего токена из K наиболее вероятных токенов) и Top-P Sampling. В случае с Beam Search мы будем настраивать количество лучей, а в случае с методами сэмплирования будем настраивать параметры K и P, а также значение температуры, которое отвечает за случайность выбора токена из подборки. Регулируя данные параметры, получим следующие значения для вывода модели:

1) BeamSearch: При количестве лучей, равном 2, получаем перплексию, равную 2.0996. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что испытываете трудности в получении внимания со стороны близких людей. В связи с этим возникает вопрос: что делать, если хочется жить? Ответ на этот вопрос'.

   При количестве лучей, равном 3, получаем перплексию, равную 2.0648. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что испытываете трудности в получении внимания со стороны близких людей. В связи с этим возникает вопрос: что делать, если хочется жить? Что делать, если'.

   При количестве лучей, равном 4, получаем перплексию, равную 1.1595. Вывод следующий: Вопрос: Что сделать чтобы захотелось жить Ответ: '.

2) Top-K Sampling: При K, равном 3, и температуре, равной 0.5, получаем перплексию, равную 1.7788. Вывод следующий: 'Что сделать чтобы захотелось жить Ответ: Уважаемая anna! Вы пишете, что в вашей жизни произошла неприятная ситуация,

которая повлияла на вашу жизнь и здоровье. Вы потеряли близкого человека, который вам очень дорог.'.

При K, равном 3, и температуре, равной 1, получаем перплексию, равную 1.7561. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Вы задаетесь вопросом, что делать, чтобы желание жить возникло? Ответ:Вы задаетесь вопросом, что сделать, чтобы захотеть жить? Ответ:Вы задаетесь вопросом, что делать, чтобы желание'.

При K, равном 5, и температуре, равной 0.5, получаем перплексию, равную 1.9505. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Вы находитесь в трудной жизненной ситуации, которая влияет на ваше эмоциональное состояние, на ваше самочувствие. Вы потеряли близкого человека, который вас поддерживал, но в то же время, вы чувствуете себя подавленным'.

При K, равном 5, и температуре, равной 1, получаем перплексию, равную 3.0662. Вывод следующий: 'Что сделать чтобы захотелось жить Ответ: Вы хотите понять, что делать, что изменить в своем поведении. Вы хотите найти решение, которое поможет вам изменить ситуацию., в своем обращении Вы пишете о том, что Вам сложно проживать свои чувства'.

При K, равном 7, и температуре, равной 0.5, получаем перплексию, равную 1.6775. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Ощущение, что Вы испытываете чувство одиночества, которое не связано с какими-то проблемами или проблемами окружающих. Вы испытываете чувство одиночества, которое не связано с какими-то проблемами или'.

При K, равном 7, и температуре, равной 1, получаем перплексию, равную 3.9317. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Вы спрашиваете, как пережить развод. И в данном вопросе мне очень хочется задать Вам несколько уточняющих вопросов. В чём причины такого длительного расстройства? Что можно сделать, чтобы помочь вам пережить'.

3) Top-P Sampling: При P, равном 0.1, и температуре, равной 0.5, получаем перплексию, равную 1.0. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что Вам сложно жить, Вы испытываете чувство одиночества, которое мешает Вам жить. , в своём обращении Вы пишете о том, что Вам сложно'.

   При P, равном 0.1, и температуре, равной 1, получаем перплексию, равную 1.7561. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что Вам сложно жить, Вы испытываете чувство одиночества, которое мешает Вам жить. , в своём обращении Вы пишете о том, что Вам сложно'.

   При P, равном 0.3, и температуре, равной 0.5, получаем перплексию, равную 1.0401. Вывод следующий: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что Вам сейчас непросто, Вы испытываете трудности в получении поддержки и принятии себя. , я понимаю, что Вы сейчас переживаете непростой период в жизни'.

   При P, равном 0.3, и температуре, равной 1, получаем перплексию, равную 1.2656. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: Уважаемая anna!! Вы пишите, что хотите жить, но не знаете, как это сделать. В своем обращении Вы пишете, что хотите умереть, но не знаете,'.

   При P, равном 0.5, и температуре, равной 0.5, получаем перплексию, равную 1.081. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: В своём обращении Вы пишете о том, что Вам сложно жить, Вы испытываете чувство

одиночества, которое мешает Вам жить. , в своём обращении Вы пишете о том, что Вам сложно.'

При P, равном 0.5, и температуре, равной 1, получаем перплексию, равную 2.1616. Вывод следующий: 'Вопрос: Что сделать чтобы захотелось жить Ответ: ! , Вы пишите, что у Вас очень активный образ жизни, и Вы часто ходите в спортзал. Но есть одно НО. Вы не пишете, как давно Вы занимаетесь фитнесом, '.

Исходя из результатов настройки генерации, можно сделать вывод, что настройка генерации действительно может помочь улучшить оценку и сделать вывод более релевантным запросу: так, самая низкая перплексия имеется у того вывода, в котором несколько раз ссылаются на проблему с жизнью. Однако заметно, что в нем начинает проявляется проблема с повторением частей текста, что и может объяснить низкое значение перплексии. При этом можно также заметить, что одну из низких оценок имеет результат, где в качестве ответов генерируются пробелы.

Полученные результаты не должны удивлять: исходя из определения перплексии, эта метрика измеряет степень уверенности в генерации последующего токена, а не правильности, данная метрика не учитывает релеватность вывода, не показывает понимания контекста запроса модели. Отсюда можно сделать вывод, что для комплексной оценки качества модели необходимы методы, которые будут учитывать эти моменты.

Заключение

В рамках данной статьи были рассмотрены некоторые существующие на текущий момент большие языковые модели и подходы к их оценке, а также произведена оценка модели ruGPT-3 small и ее дообученных версий на корпусе EMERCOM. Было выявлено, что существующие популярные оценки языковых моделей не подходят для измерения качества работы, поскольку они не учитывают необходимые при генерации качеств, например, релеватности или понимания контекста.

Литература

1. Осваиваем архитектуру Transformer. Разработка современных моделей с помощью передовых методов обработки естественного языка / пер. с анг. В. С. Яценкова. – М.: ДМК Пресс, 2022. – 320 с.

2. Russian GPT-3 models. – Режим доступа: https://github.com/ai-forever/ru-gpts. – (Дата обращения: 19.04.2024).

3. Transformers. – Режим доступа: https://git.lnyan.com/docs/transformers/main/en/index. – (Дата обращения: 19.04.2024)

4. Evaluation Metrics for Language Modeling. – Режим доступа: URL: https://thegradient.pub/understanding-evaluation-metrics-for-language-models (Дата обращения: 19.04.2024)

004.855.6

Интеграция нейронных сетей в мобильное приложение для решения задачи детектирования

А. В. Палагутин, С. Ю. Болотова

Воронежский государственный университет

Введение

В настоящее время современные технологии информатики и телекоммуникаций стремительно проникают в различные сферы жизни, включая экономику, экологию и науку. Одним из активных предложений является применение искусственного интеллекта (ИИ) и нейронных сетей в мобильных разработках, которые открывают новые перспективы для процессов автоматизации и улучшения качества жизни. Данная работа ориентирована на подход к изучению нейронных сетей в мобильной разработке с использованием приложений для обнаружения растений в том же духе. Растения играют основополагающую роль в биосфере, оказывая влияние на экологию, продовольственную безопасность и общее благополучие человека. Поэтому разработка специальных приложений, позволяющих автоматически распознавать и классифицировать растения, является актуальной.

1. Обучение сети для детектирования

Для начала необходимо решить задачу детектирования растений с применением компьютерного зрения. В качестве сети для детектирования следует выбрать Yolov5. Yolov5 — более легкая и широкая версия Yolo, предназначенная для ускорения обнаружения объектов. Эта архитектура использует метод You Only Look Once (YOLO) для обнаружения объектов в реальном времени. Yolov5 может работать на мобильных устройствах с низким энергопотреблением и обеспечивать точность обнаружения, что делает данную архитектуру (рис. 1) отличным выбором для мобильных приложений.[1]

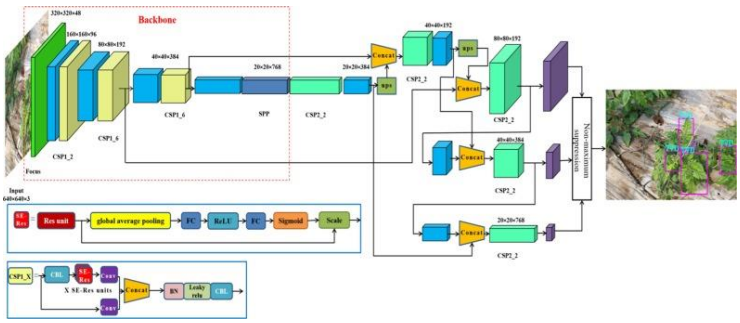


Рис. 1. Архитектура Yolov5

Сбор данных для обучения моделей YOLOv5 по изображениям растений включает в себя несколько шагов, каждый из которых требует особого подхода и использования соответствующих инструментов и технологий.

Сбор изображений: первый шаг - сбор изображений растений, которые будут использоваться для обучения моделей. Важно собрать достаточное количество изображений, чтобы модель могла обучаться на различных типах растений и условиях освещения. При этом необходимо учесть, что изображения должны быть достаточно качественными и четкими, чтобы обеспечить точность обнаружения объектов.

Тэггинг изображений: после сбора изображений необходимо уменьшить их тэггинг для обучения модели. Это можно сделано с помощью инструмента для аннотирования изображений LabelImg. При переносе изображений необходимо указать координаты границ объектов (ограничивающих рамок) вокруг каждого растения и присвоить их соответствующему классу. Важно обеспечить точность и качество тэггинга, чтобы модель могла эффективно обучаться на них.

Конвертация данных: после подготовки изображений и тэгов необходимо выполнить их конвертацию в формат, который можно использовать для обучения YOLOv5. Это можно сделать с помощью такого инструмента, как Python-библиотека OpenCV для чтения изображений и преобразования их в формат YOLOv5. В процессе обучения модели, изображения загружаются в память и преобразуются в формат, подходящий для ввода в модель. Затем выполняется предсказание и применяется немаксимальное подавление для обнаружения объектов.

После подготовки изображений выполнено обучение с помощью скрип-та train.py, который содержится в репозитории YOLOv5. Скрипт принимает множество параметров, которые настраиваются в процессе обучения. Это такие параметры как размер входного изображения (ширина и высота), размер пакета (batch size), количество эпох (циклов обучения), путь к файлу конфигурации данных путь к файлу конфигурации модели (yolov5s.yaml), путь к предобученной модели и имя сохраненной модели. После обучения модель тестируется скриптом detect.py.

2. Обучение сверточной нейронной сети

Следующая задача, которую необходимо решить, это классификация полученного растения. В качестве сети для классификации растений применима архитектура MobileNet (рис. 2), которая хорошо показывает себя в решении схожих задач.

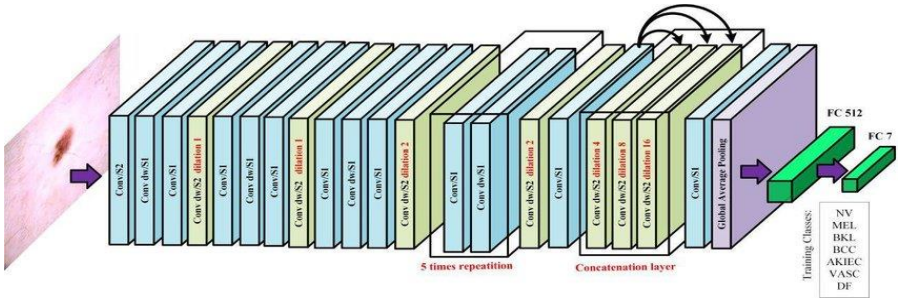


Рис. 2. Архитектура MobileNet

Сбор данных для обучения схож со сбором данных для модели детектирования, существенным различием является выделение классов обучающей выборки, в данном случае будем считать, что каждое изображение в папке принадлежит одному классу, следовательно обучающий набор будет состоять из n папок с изображениями, где n – число различных классов растений.

После настройки гиперпараметров необходимо выполнить обучение сети на готовом наборе данных. Обучение проводится с использованием методики стохастического наклона (SGD) и ее модификаций, таких как Adam, RMSProp plus и т.д. На протяжении всего обучения сети требуется следить за высочайшим качеством обучения и корректировать гиперпараметры на основе результатов. Также стоит использовать различные подходы к регуляризации.[2]

В данном случае при обучении сети MobileNet применимы следующие методы регуляризации:

- Регуляризация L2 с коэффициентом 0,0005 для предотвращения переобучения сети.
- Dropout с коэффициентом 0,5 для предотвращения переобучения сети.
- Раннее прекращение для предотвращения переобучения посредством непрерывного обучения, когда точность на валидационном датасете перестает улучшаться.

Кроме того, при обучении сети MobileNet следует применить метод переноса обучения, который позволяет использовать предварительно обученную модель для решения новых задач. Для данной модели учитывается только вес сверточного слоя. Полносвязные слои были инициализируются случайными значениями.[3]

После завершения обучения сети выполняется ее тестирование на независимом наборе данных. На основе результатов тестирования сделаны выводы о качестве обучения сети и ее эффективности для решения поставленной задачи. Получена точность классификации в районе 80%.

3. Интеграция с мобильным приложением

Интеграция модели YOLOv5 с TensorFlow Lite была выполнена следующим образом:

- Модель YOLOv5 конвертируется в формат TensorFlow Lite с использованием инструментов, предоставленных TensorFlow Lite. Этот процесс включает в себя изменение внутренней структуры модели и преобразование параметров модели для ее совместимости с TensorFlow Lite. Конвертированная модель оптимизируется для более эффективного выполнения на мобильных устройствах, что включает в себя уменьшение размера модели и оптимизацию вычислительных операций.[4]
- Для интеграции модели TensorFlow Lite в мобильное приложение добавлены необходимые библиотеки и файлы модели к проекту приложения.
- Мобильное приложение предоставляет входные данные, т.е изображения, полученные с камеры устройства. Эти данные были предварительно обработаны и преобразованы в формат, который соответствует ожидаемому входу модели TensorFlow Lite. Подготовка входных данных включает в себя масштабирование и нормализацию, необходимые для правильной работы модели.
- Модель TensorFlow Lite возвращает результаты детектирования, которые включают обнаруженные объекты, их координаты, классы и оценки уверенности. Эти результаты использованы для ведения справочника растений (рис. 3).
- Для обеспечения высокой производительности на мобильных устройствах применены различные техники оптимизации. Это такие техники как себя квантизация модели (понижение точности вычислений) и оптимизация вычислительных операций. [5] Эти шаги помогли уменьшить нагрузку на

процессор устройства, что положительно влияет общую производительность приложения, работающего с моделью TensorFlow Lite.

Рис. 3. Пример детектирования растения

Заключение

В результате исследования собраны и проаннотированы различные данные для обучения модели. После изучения различным моделей, которые могут быть применены для решения задач детектирования в мобильной разработке и обучения выбранной архитектуры YOLOv5, проведена ее оптимизация для мобильных устройств, что позволило получить эффективное средство, способное точно и быстро детектировать растения. Продолжение исследований в этой области и развитие технологий детектирования позволит сделать управление природными ресурсами более надежным и автоматизированным в будущем.

Литература

1. Evolution of Yolo algorithm and Yolov5: The state-of-the-art object detection algorithm (URL: https://www.theseus.fi/bitstream/handle/10024/452552/Do_Thuan.pdf?isAllowed=y&sequence=2) (дата обращения: 12.01.24)
2. Umberto Michelucci. Advanced Applied Deep Learning / Um-berto Michelucci – New York: Apress Berkeley, CA, 2019. – 285p
3. Antonio Gulli. Deep Learning with TensorFlow 2 and Keras / Antonio Gulli, Amita Kapoor, Sujit Pal –Birmingham: Packt Publishing, 2019. – 646p
4. Mohammad Jani. Model Compression Methods for YOLOv5: A Review/Mohammad Jani, Jamil Fayyad, Younes Al-Younes, Homayoun Naj-jaran – Canada: University of Victoria, 2023 – 18p
5. Себастьян Рашка. Python и машинное обучение / Себастьян Рашка, Мирджали Вахид – Бирмингем: Packt Publishing, 2019. – 770p

УДК 519.6; 658.5

ОСОБЕННОСТИ ФОРМИРОВАНИЯ КОМАНД МУЛЬТИПРОЕКТОВ В IT-СФЕРЕ

С.А. Палкина

Воронежский государственный университет

Введение

Превышение предложений над спросом на рынке труда в IT-сфере позволяет работодателям отбирать не только хороших специалистов, но и максимально эффективных для компании. В понятие эффективности специалиста стала вкладываться способность сотрудника вести качественно, в том числе одновременно и несколько, проектов. Однако это требует пересмотра существующих методик подбора персонала, которые учитывали бы предстоящую загрузку кандидатов.

В работе поставлена цель: разработать модель оценки кандидата для отбора в команды мультипроектов в IT-сфере, которая позволит учитывать профессиональные, личностные и социальные характеристики, определяющие потенциал работы в мультипроектах. Это позволит формировать команды из специалистов, которые могут реализовывать мультипроекты в IT-сфере на качественно высоком уровне.

1. Формирование команд проектов

1.1. Постановка задачи

Изучение работ по подходамк формированию команд проектов позволяет сделать вывод о значимости личностных качеств, социальных характеристик и профессиональных компетенций соискателей. При этом многие исследователи отмечают, что определяющими факторами в успешности проектов являются не столько профессиональный уровень и опыт или все характеристики в совокупности, сколько личностные качества сотрудников. Это связано с тем, что реализация мультипроектов, это, в первую очередь, командная работа с большой вовлеченностью. При этом единой методики подбора специалистов, учитывающей и профессиональные компетенции, и личностные качества, и социальные характеристики нет. В зависимости от направленности проектов применяются подходы, определяющие порядок и критерии отбора кандидатов как сотрудников в штат, или работ на конкретном проекте. В данном исследовании акцент сделан на мультипроектах в IT-сфере, так как их доля в отрасли информационных технологий постоянно растет. Проекты в IT-сфере обладают рядом специфичных характеристик, что выделяет их среди других проектов.

Мультипроект рассматривается как совокупность нескольких монопроектов, которые выполняются одной или несколькими организациями или командами в рамках одной программы или портфеля. В данной работе, рассматривается ситуация, когда подбирается кандидат для работ по проектам, реализуемым одной организацией. При этом эффективность мультипроекта рассматривается аналогично эффективности обычного (моно) проекта. Под эффективностью понимается достижение высоких результатов с наименьшими затратами, в том числе, и по времени. Ставится задача: сформировать модель оценки кандидата в мультипроект, которая позволит учесть личностные и социальные характеристики, а также

профессиональные компетенции.

1.2. Процесс формирования команды

Процесс формирования команды проекта начинается с подбора руководителя на проект. Для этого формируется экспертная комиссия, которая отбирает кандидатов по имеющейся базе претендентов. Кандидатура руководителя согласовывается в большинстве случаев с заказчиком и утверждается организацией-исполнителем проекта. Далее руководитель определяет процедуру формирования команды, в том числе, критерии отбора кандидатов и порядок их утверждения в проект. Согласно определенным ранее критериям формируется база кандидатов, по которой осуществляется отбор участников проекта. При отсутствии необходимых кандидатов база расширяется. Для отбора кандидатов может использоваться информация из анкет соискателей, проводиться тестирование или собеседование. Отобранные кандидаты утверждаются в проект. После этого, команда считается сформированной. В ходе реализации проекта, состав команды может быть пересмотрен. Поиск и отбор кандидатов аналогичен подбору участников для новой команды.

1.3. Модель оценки кандидата на проект

Один из подходов формирования команд для проектов основан на применении метода многокритериальной оптимизации – методе анализа иерархий. Использование метода позволяет сформировать видение идеального кандидата в проект и упростить процесс оценки потенциального кандидата. Метод МАИ основан на декомпозиции проблемы (цели) на составляющие, упрощающие выбор. В данном случае целью является нахождение (определение среди соискателей) идеального кандидата. Составляющими декомпозиции являются критерии отбора (оценки) кандидата в проект. Модель оценки кандидата в проект на базе метода МАИ представлена на рис. 1.

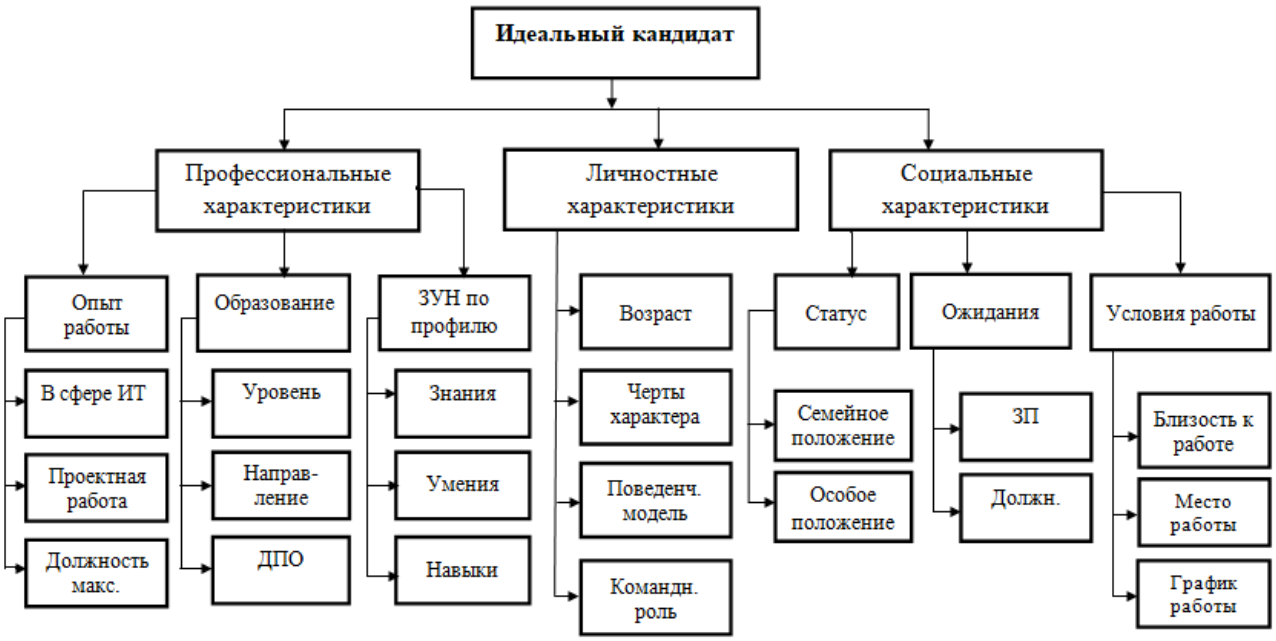


Рис.1. Модель оценки соответствия кандидата в проект

Согласно модели, отбор и оценка каждого потенциального кандидата производится по

таким показателям как профессиональные, личностные и социальные характеристики. В профессиональных характеристиках учитываются образование, опыт работы, знания, умения и навыки (ЗУН) по специфике проекта. Предполагается, что ЗУН оцениваются отдельно с использованием, например, тестов. При этом в критерии для дальнейшей оценке используется итоговый балл, полученный, например, по результатам тестирования. В личностных характеристиках учитываются возраст, черты характера, поведенческая модель и возможная командная роль. Личностные характеристики могут быть получены как результат самооценки кандидата, так и по результатам отдельного тестирования.

Компоненты «Статусы», «Ожидания», «Условия работы» блока социальных характеристик позволяют сделать выбор между кандидатом более или менее «удобным» для компании, т.е. готовым к работе в офисе или согласному только на частичную занятость, потенциально загруженным в периоды сессий, если является студентом, ожидающим большей заработной платы или желающий долгое время работать на одной и той же позиции. Это связано с тем, что для хорошей работоспособности важны и удаленность работы от дома, и образ жизни, который зависит от того, соискатель студент или молодой несемейный человек, и даже от ожиданий по заработной плате. Характеристики нижнего уровня модели считаются заданными на момент оценки кандидатов. По ним определяется рейтинговый балл для отбора кандидата в проект.

2. Формирование команд мультипроектов

2.1 Модель оценки кандидата в мультипроект

К недостаткам приведенной выше модели, которые делают ее не совсем корректной для использования в формировании команд мультипроектов, следует отнести недостаточно корректную декомпозицию блока профессиональных характеристик. В IT-сфере принято оценивать не столько знания, умения и навыки, сколько их связки (skills). Принято выделять Hard skills, Soft skills, Meta skills и Self skills. Hard skills — «жесткие» или профессиональные навыки. Soft skills — «мягкие» навыки, личностные характеристики, которые не закрепляются за одной профессиональной областью. Meta skills — базовые качества личности, которые считаются врожденными, на основе которых складываются остальные навыки. Self skills или self management – это навыки самоуправления, саморегулирования, организации себя, планирования, контроля и мотивация, навыки ориентированные на умение управлять собой. Self skills в ряде случаев могут стать решающими при отборе кандидатов. Именно они позволяют отобрать кандидата, который сможет работать на нескольких проектах одновременно, без снижения качества работы. В связи с этим, модель оценки кандидата в мультипроект следует скорректировать. Итоговая модель представлена на рис. 2. Блок «ЗУН по профилю» заменен на блок «Компетенции (Skills)». Блок «Черты характера» удален, т.к. он учитывается в блоке «Компетенции (Skills)». Блок «Meta skills» в модели не сформирован по двум причинам: качества трудно выявляемы тестами, базовые качества личности проявляются в других навыках, более легко выявляемых.

2.2. Процесс формирования команды мультипроекта

Формирование команды мультипроекта аналогично процедуре, приведенной в пункте 1.2. Подбор тестов для блока «Компетенции (Skills)» может осуществляется на основе существующих методик, которые используются как для подбора кандидатов в проект (любой), так и при отборе на вакансию в штат компании (не на проектную работу).

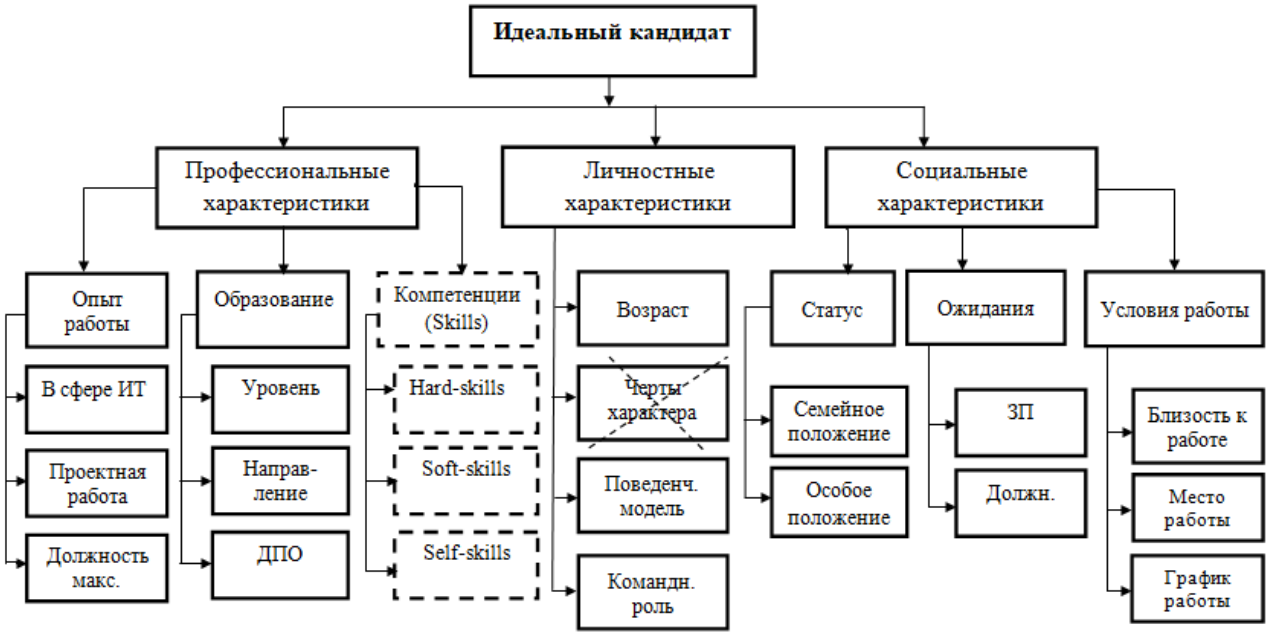


Рис.2. Модель оценки соответствия кандидата в мультипроект

Заключение

Модель может использоваться для формирования команд мультипроектов, а также монопроектов и стартап-проектов. В ней учитываются профессиональные, личностные и социальные характеристики. При этом профессиональные характеристики заложены отдельным блоком и, в зависимости от функциональных обязанностей участника проекта, могут быть любыми. Универсальность модели заключается и в том, что при формировании команды мультипроекта, руководителю не обязательно формировать видение для каждой позиции проекта. Достаточно сформировать единожды и далее изменять оценку только по уровню элементов, которые разняться для разных позиций, например, по блоку «Hard-skills».

Литература

1. Азарнова Т.В. Модели и методы принятия решений / Составители: Т.В. Азарнова, Ю.В. Бондаренко, Н.Б. Баева, Е.С. Дашкова, В.В. Ухлова; Воронежский государственный университет. – Воронеж: Издательский дом ВГУ, 2021. – 310 с.

2. Бондаренко Ю.В., Горошко И.В., Васильчикова Е.В. Экспертно-тестовый механизм комплексной оценки кандидатов при подборе персонала // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2020. Т. 20. № 1. – С. 100-110.

3. Джабраилова, З.Г. Моделирование процесса выбора кандидатов на вакантные должности с применением нечеткой логики / З.Г. Джабраилова, С.Р. Нобари // Искусственный интеллект: сб. статей. – Баку, 2009. – С. 254–259.

4. Жуков, Ю. М., Журавлёв А. В., Павлова Е. Н. Технологии командообразования: учеб. пособие для студ. вузов Москва : Аспект Пресс, 2008. – 320 с.

5. Кибанов, А.Я. Управление персоналом организации: стратегия, маркетинг, интернационализация: учеб. пособие / А.Я. Кибанов, И.Б. Дуракова. – Москва : Инфра-М, 2009. – 301 с.

6. Магура, М.И. Поиск и отбор персонала. – Москва, 2003. – 312 с.

7. Ногин, В.Д. Принятие решений при многих критериях / В.Д. Ногин. – Санкт-Петербург : ЮТАС, 2007. – 104 с.
8. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати., – пер. с англ. – Москва : Радио и связь, 1993. – 278 с.
9. Ухлова, В. В. Применение системного подхода в процедурах формирования команд стартапов ИТ-проектов / Актуальные проблемы прикладной математики, информатики и механики / В. В. Ухлова, С. А. Палкина // Сборник трудов Международной научной конференции, Воронеж, 13-15 декабря 2021 г. — Воронеж, 2022. — С. 791-796.
10. Палкина С.А. Один из подходов формирования эффективных команд стартап-проектов / С.А. Палкина, А.Э. Потемкина // Математика, информационные технологии, приложения : сборник трудов Межвузовской научной конференции молодых ученых и студентов, Воронеж, 26 апреля 2023 г. – Воронеж : Научная книга, 2023. — С. 334-337.

УДК 004.6, 004.9

ПРОБЛЕМЫ ПРИМЕНЕНИЯ ТЕХНОЛОГИЙ BIG DATA В ОБРАЗОВАНИИ

С.А. Палкина, Д.В. Шерстюк

Воронежский государственный университет

Введение

Использование технологий big data открывает широкие возможности для анализа данных из разных источников. Принятие управленческих решений становится более объективным, а значит, и их результативность выше. Цифровизация, на путь которой, встало и образование, теперь также нуждается в технологиях big data. Считается, что это улучшит качество образования, повысит прозрачность предоставления образовательных услуг, откроет новые возможности для анализа образовательной деятельности. Однако, чтобы действительно, использовать технологии big data следует не только перевести информацию в цифровой вид, но изменить ряд бизнес-процессов в деятельности образования, которые касаются не только регламентов сбора и хранения данных, но и связаны с изменением программного обеспечения образовательных учреждений.

В работе поставлена цель: оценить трудности использования технологий big data для анализа образовательной деятельности вузов.

1. Задачи для big data в образовании

Под задачами big data понимаются задачи, которые используют данные в формате «VVV», т.е. многоформатные, в большом объеме и имеющие либо высокую скорость накопления, либо требующие высокой скорости обработки. К таким задачам можно отнести задачи, связанные с оцениванием посещаемости студентами занятий, оцениванием успешности освоения дисциплин, анализом загрузки учебных аудиторий, задачи по анализу проведения приемных кампаний, формирования образовательных траекторий, оценке конкурентоспособности образовательных программ. Каждая из этих задач предполагает, что информация по «бизнес-процессу» собирается в цифровом виде и хранится в учебном заведении, т.е. доступ к ней открыт. Рассмотрим проблемы внедрения технологий big data в образование на примере задачи, по оценке конкурентоспособности образовательных программ.

Согласно федеральному закону N 273-ФЗ «Об образовании в Российской Федерации» от 29 декабря 2012 г. организация и осуществление образовательной деятельности по образовательным программам высшего образования - программам бакалавриата, специалитета и магистратуры регламентируется федеральным государственным образовательным стандартом (ФГОС). В ВГУ большинство направлений подготовки реализуются согласно стандарту ФГОС3++, который базируется на профессиональных стандартах. Соответственно, образовательная программы, основана на формировании профессиональных компетенций, и именуется как основная профессиональная образовательная программа (ОПОП).

Уровень конкурентоспособности образовательных программ можно оценивать на основе востребованности выпускников на рынке труда. Под рынком труда понимается механизм взаимодействия между работодателями и наемными работниками, выражающий социально-экономические, трудовые и правовые отношения между ними. Роль вузов в формировании рынка труда заключается в том, что вузы предоставляют образовательные

услуги потенциальным работникам. В рамках образовательных программ формируются профессиональные компетенции, которые будут (или не будут) востребованы работодателем. Будем считать что, чем больше спрос на выпускника, тем выше конкурентоспособность ОПОП. Соответственно, будем считать, что конкурентоспособность ОПОП определяется соотношением количества выпускников вуза с подготовкой по соответствующему профессиональному стандарту с количеством вакансий на рынке (потребностью рынка труда). Взаимосвязь ОПОП, профессиональных стандартов и потребностей работодателя на рынке труда представлена на рис. 1. Ставится задача: проанализировать уровень конкурентоспособности образовательных программ ВГУ на региональном рынке труда.

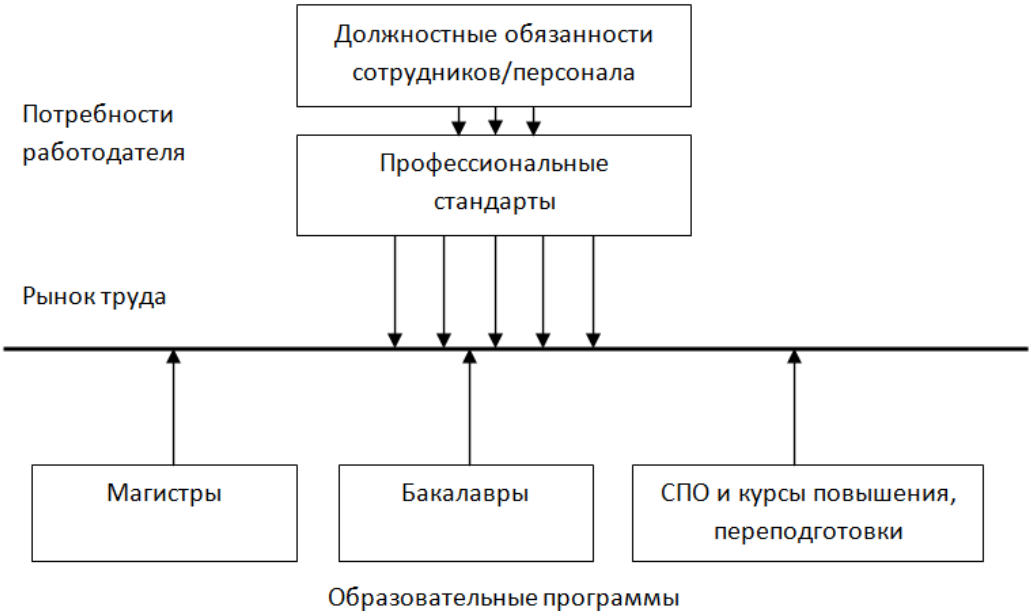


Рис. 1. Взаимосвязь ОПОП, профессиональных стандартов и потребностей работодателя на рынке труда

2. Источники данных

Для задачи, сформулированной выше, источниками данных будут источники со стороны государства, вуза и со стороны ресурсов по поиску/предложению работы. Со стороны государства – перечень профессиональных стандартов, перечень уровней и направлений подготовки специалистов. Со стороны вуза (ВГУ) источниками для предоставления перечня реализуемых ОПОП – реестр ОПОП, информация по профессиональным стандартам, реализованным в ОПОП – учебный план или описание ОПОП, информация по контингенту обучающихся – выгрузки из информационной системы вуза. Со стороны ресурсов по поиску/предложению работы – выгрузки о текущих вакансиях и резюме специалистов, ищущих работу.

Согласно методологии CRISP-DM, для проведения анализа данных, необходимо произвести сбор (выгрузку) данных, оценить качество данных, привести их к виду, удобному для обработки и только затем исследовать.

Информация в открытых источниках со стороны государства размещается в pdf-файлах. Со стороны вуза: учебный план может быть получен в формате plx, pdf или xls, реестр ОПОП – в формате pdf или xls, описание ОПОП – в формате pdf, выгрузки о контингенте предоставляются в текстовом формате doc. Со стороны ресурсов по поиску/предложению

работы: информация выгружается в форматах txt, json или xls.

3. Проблемы обработки и использования данных

Согласно описаниям источников, приведенным в пункте 2, форматы исходных файлов различны. В связи с этим, при выгрузке данных потребуется либо переформатирование данных, либо использование ETL-платформ. При этом последняя не решает всех проблем, связанных с конвертацией форматов и требует повышения качества данных. После выгрузки данные будут иметь один формат, но не всегда удобный для обработки. В данном случае, единый и легко получаемый формат – текстовый, pdf-формат. Однако для анализа это самый сложный формат. Более предпочтительным является формат электронных таблиц xls. В этом случае, учебные планы, хранящиеся в первоисточниках как plx-файлы, следует конвертировать в xls. Реестры ОПОП также следует обрабатывать по первоисточнику в xls-формате. Сложность использования этих форматов ограничена уровнем доступа к ним, т.к. файлов в этих форматах нет в открытом доступе. Та же проблема наблюдается и для данных стороны государства, в открытых источниках они размещены в формате pdf.

Информация, которая будет использоваться для оценки конкурентоспособности ОПОП: шифр и наименование ОПОП, уровень подготовки, период обучения, наименование профессионального стандарта, трудовые функции, уровень квалификации, количество человек, обучающихся по программе (контингент). Несмотря на то, что информация официальная, в документы она не всегда вносится из справочников. В связи с этим, качество получаемой информации может быть низким, что делает невозможной ее дальнейшую обработку. Ошибки, встречающиеся при выгрузке данных, приведены в табл. 1.

Таблица 1

Возможные ошибки выбранных атрибутов

Выбранные атрибуты	Ошибки
Шифр и название ОПОП, направления подготовки, уровень подготовки, шифр и название направления профессионального стандарта	отсутствие единого формата неполные значения недопустимые символы, в том числе лишние пробелы
Численность контингента, уровень подготовки, уровень квалификации, период обучения	пустые значения отсутствие единого формата недопустимые символы несогласованные данные
Трудовые функции	отсутствие единого формата недопустимые символы, в том числе лишние пробелы

Одним из решений обозначенных проблем является создание внутренней для вуза базы данных, позволяющей консолидировать необходимые данные. При этом данные будут регулярно выгружаться с использованием API-загрузчиков, созданных с учетом каждого из источников.

Заключение

Обозначенные проблемы применения технологий big data не должны останавливать вузы в использовании современных инструментов анализа данных. Они показывают необходимость адаптации существующего ПО, например, систем Business Intelligence, под задачи вуза и показывают важность разработки специального алгоритмического обеспечения для подготовки данных для решения задач аналитики. Более того, решение данных проблем позволит

осуществлять визуализацию данных по ОПОП, что существенно упростит аналитику имеющихся данных.

Литература

1. Арзамасцева, Л.П. Роль профессионального образования в формировании современной рабочей силы высокого качества / Л.П. Арзамасцева, О.А. Колесникова, Ю.В. Хицкова // Вестник ВГУ. Сер. Экономика и управление. 2018. № 2. – Режим доступа: http://www.vestnik.vsu.ru/pdf/econ/2018/ 02/2018-02-07.pdf.
2. Бондаренко, Ю.В. Экспертно-тестовый механизм комплексной оценки кандидатов при подборе персонала / Ю.В. Бондаренко, И.В. Горошко, Е.В. Васильчикова // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2020. – Т. 20. № 1. – С. 100-110.
3. Владимирова, Н. В. Актуальные вопросы взаимодействия системы профессионального образования и рынка труда / Н. В. Владимирова. – Режим доступа: https://урок.рф/library/aktualnie_voprosi_vzaimodejstviya_sistemi_professi_022930.html.
4. Кибанов, А.Я. Управление персоналом организации: стратегия, маркетинг, интернационализация: учеб. пособие / А.Я. Кибанов, И.Б. Дуракова. – Москва : Инфра-М, 2009. – 301 с.
5. Пухова, А.Г. Предпосылки и факторы, влияющие на формирование и функционирование регионального рынка труда // Вестник Минского университета. – 2014. № 2(6). – С. 7.
6. Ухлова, В.В. Практические аспекты интеграции информационных систем с целью решения задач аналитики / В.В. Ухлова, Ю.А. Максименко // Сборник трудов Международной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 7–9 декабря 2020 г.) – Воронеж : Издательство «Научно-исследовательские публикации». – 2021. – С. 407-411.

УДК 004.49

Атаки на цепочки поставок.

Д. И. Перелыгин

Воронежский государственный университет

Введение.

Атаки на цепочку поставок - это угрозы, нацеленные на разработчиков программного обеспечения и поставщиков. Их основная цель - это получить доступ к исходным кодам, процессам сборки или механизмам обновления путем заражения допустимых приложений для распространения вредоносных программ. Для этого существует два направления атак: на поставщика и на потребителя программного обеспечения. Для злоумышленника нет необходимости пытаться атаковать цель напрямую - вскрыть защиту компании и обходить её системы безопасности, если можно атаковать, например, библиотеку, которая используется в разработке и внедрить в нее уязвимость, открывающую доступ к инфраструктуре.

1. Типы атак.

Чаще всего встречаются атаки следующих типов:

Программные — хакеры целятся в исходный код ПО поставщика. Они внедряют зловредные элементы в доверенное приложение или компрометируют сервер с обновлениями. Такие атаки трудно отследить, потому что злоумышленники часто подписывают код украденными легитимными сертификатами.

Аппаратные — злоумышленники атакуют устройства связанные со всей цепочкой поставок, например клавиатуры или веб-камеры. Часто они используют для этого бэкдоры.

Микропрограммные — киберпреступники внедряют вредоносные программы в загрузочный код компьютера. Атака запускается после включения устройства, в результате опасности подвергается вся система. Подобные инциденты происходят быстро, и если у компаний нет специальной защиты, вторжение с высокой вероятностью останется незамеченным.

2. Причины атак на цепочки поставок.

Почему компании становятся жертвами атак на цепочки поставок?

Есть три главные причины: 1) Продукты организаций имеют низкий уровень защищённости. ИТ-системы 96 % крупных российских компаний содержат уязвимости, с помощью которых хакеры могут проникнуть во внутреннюю сеть и реализовать 89 % недопустимых событий — их определяют сами организации.

2) Компании внедряют небезопасные компоненты от сторонних производителей и ПО с открытым кодом (Open Source Software, OSS). С одной стороны, OSS позволяет сократить затраты на разработку и сроки создания продуктов, с другой — может внести критические уязвимости, которые откроют злоумышленникам двери во внутреннюю инфраструктуру компаний. Как минимум 33 % российского ПО на базе открытого кода содержит проблемы безопасности. В OSS и платформах с репозиториями в том числе бреши обнаруживаются весьма часто.

Кроме этого, в OSS нередко используются сторонние компоненты, которые в свою очередь могут содержать другие элементы и так далее. Когда разработчик внедряет такую «матрёшку», он косвенно связывает свой проект с её внутренними частями.

Компании доверяют партнёрам и забывают о безопасности. Меньше 10 % компаний оценивают уровень безопасности поставщиков услуг, причём в основном они используют для этого опросные листы или проверяют только юридическую чистоту — аудит носит формальный характер и не позволяет объективно оценить зрелость ИБ. Реальные проверки проводит лишь 1 %, и зачастую предпосылкой к этому служит уже произошедшая атака.

Также, 94 % организаций сегодня используют облачные сервисы. Эксперты утверждают, что безопасность таких продуктов пока находится на низком уровне, в них много опасных уязвимостей, которые в 2023 году выступают одной из главных ИБ-угроз [1].

3. Примеры атак на цепочки поставок.

В 2020 году киберпреступники заразили вирусом обновление ПО Orion от SolarWinds. В результате они получили доступ ко всем его пользователям — а это сотни государственных организаций США и 80% компаний из списка Fortune 500. В 2021 году хакеры взломали онлайн-платформу для тестирования ПО Codecov и заразили один из скриптов вирусом, который перехватывал загрузки и собирал конфиденциальную информацию, в том числе учётные данные, токены и ключи. Codecov применяют более 29 тысяч клиентов, и хакеры использовали их данные для проникновения в сети производителей ПО. Из последнего - были обнаружены киберпреступники, умело манипулирующие функциями поиска на GitHub. Злоумышленники создают вредоносные репозитории с популярными названиями и темами, различными методами добиваются повышения рейтинга в поиске и скрывают свое вредоносное ПО в основном в виде запутанного кода глубоко в файлах репозитория [1].

4. Структура и методы атаки на цепочки поставок.

Злоумышленник может воспользоваться любой из данных точек и внедрить свой код или ПО. Для компрометации цепочки поставок могут быть использованы различные методы:

Методы атак, используемые для компрометации цепочки поставок	
Заражение вредоносным программным обеспечением	Шпионское ПО, используемое для кражи учетных данных у сотрудников
Социальная инженерия	Фишинг, поддельные приложения, опечатки и т.д.
Атака грубой силой	Угадывание пароля SSH
Использование уязвимости программного обеспечения	SQL-инъекция или эксплойт переполнения буфера в приложении
Использование уязвимости конфигурации	Проблемы с конфигурацией
Физическая атака или модификация	Физическое вторжение
Разведка по открытым источникам (OSINT)	Поиск учетных данных в Интернете

Среди методов атак, используемых для компрометации клиента через его поставщика

Методы атак, используемые для компрометации клиента	
Доверительные отношения	Доверять сертификату, доверять автоматическому обновлению, резервному копированию
Компромисс путем управления	Вредоносные скрипты на веб-сайте для заражения пользователей вредоносными программами
Фишинг	Поддельные уведомления об обновлении, сообщения
Заражение вредоносным ПО	Троян удаленного доступа, бэкдор

можно выделить:

Активы поставщика, на которые нацелились злоумышленники, относятся к тому, что было целью атаки. По этому можно определить намерения злоумышленника, проанализировав список затронутых активов.

Защита от атак на цепочки поставок.

Гарантированно обезопасить себя от атаки цепочки поставок невозможно, а поэтому главное целью и задачей защищающейся стороны должна стать остановка атаки на ранней стадии, прежде чем злоумышленник смог закрепиться внутри инфраструктуры и нанести урон.

Для того, чтобы вовремя обнаружить и остановить атаку на цепочку поставок, необходимо [2]:

- Использовать актуальные средства как для конечных устройств, так и для сети;
- Настроить мониторинг сети на предмет подозрительной активности;
- Организовать процесс резервного копирования, на случай уничтожения или шифрования данных;
- Внедрить политику безопасности, например, разрешать запуск только авторизированных приложений;
- Внедрить безопасный жизненный цикл разработки программного обеспечения (SDLC);
- Поддержание программного обеспечения в актуальном состоянии;

Заключение.

По прогнозам экспертов, тенденция применять атаки на цепочки поставок сохранится в 2024 году и атаки будут становиться все более таргетированными, поэтому организациям следует уделять приоритетное внимание безопасности цепочки поставок в своей общей стратегии безопасности, чтобы снизить риски, связанные с этими атаками.

Литература.

1. Чем опасна атака на цепочку поставок ПО и как с ней бороться. – Режим доступа:

http://www.sberbank.ru/ru/person/kibrary/articles/chem-opasna-ataka-na-cepochku-postavok-po-i-kak-s-nej-borotsya - (Дата обращения: 18.04.2024)

3. Атаки на цепочки поставок: какие существуют риски и как от них защититься. – Режим доступа: https://www.anti-malware.ru/analytics/Threats_Analysis/Supply-Chain-Attack - (Дата обращения: 18.04.2024)

УДК 519.85

Алгоритмы обучения параметров когнитивных карт

А. П. Першина

Воронежский государственный университет

Введение

Для решения задач в различных сферах эксперт зачастую составляет модель, задача которой объяснить взаимосвязь между различными факторами, влияющими на решение поставленной проблемы. Одним из подходов построения подобных моделей является когнитивный подход. Данный подход ориентирован на выявление структуры знаний о рассматриваемой системе и на моделирование слабоструктурированных связей между основными процессами, протекающих в ней. Подход заключается в построении когнитивной карты проблемной ситуации. Когнитивные карты широко используются в различных областях, например: в психологии, в образовании, в археологии.

В данной статье рассмотрены основные аспекты моделирования с помощью когнитивных карт и приведен краткий обзор методов обучения параметров когнитивных карт.

1. Основные аспекты моделирования с помощью когнитивных карт

Когнитивная карта представляет собой ориентированный граф, узлы которого представляют собой некоторые объекты (концепты), а дуги – связи между ними, характеризующие причинно-следственные отношения. Когнитивные карты могут быть использованы для качественной оценки влияния отдельных концептов друг на друга и на устойчивость системы в целом, для моделирования и оценки эффективности применения различных управленческих стратегий и прогноза принимаемых решений [1].

Среди когнитивных моделей выделяют традиционные и нечеткие. Традиционные карты задаются в виде ориентированных графов и представляют моделируемую систему в виде множества концептов, отображающих существенные для проблемной ситуации объекты или атрибуты, и множества дуг, характеризующих причинно-следственные связи между объектами. Традиционные когнитивные карты используются для качественной оценки влияния отдельных концептов на устойчивость системы. На рисунке 1 представлена пример традиционной когнитивной карты.

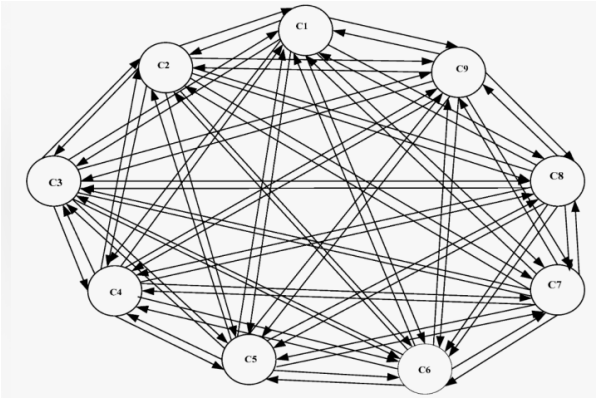


Рис.1. Традиционная когнитивная карта

Нечеткие когнитивные карты (fuzzy *cognitive* maps) были предложены Б. Коско в 1986 г. и используются для моделирования нечетких причинных взаимосвязей между концептами некоторой предметной области. В отличие от простых когнитивных карт, нечёткие когнитивные карты представляют собой нечёткий ориентированный граф. Направленные рёбра графа не только отражают причинно-следственные связи между концептами, но и определяют степень влияния (вес) связываемых концептов. Активное использование нечётких когнитивных карт в качестве средства моделирования систем обусловлено возможностью наглядного представления анализируемой системы и хорошей интерпретируемостью причинно-следственных связей между концептами [3].

На рисунке 2 изображена взвешенная когнитивная карта.

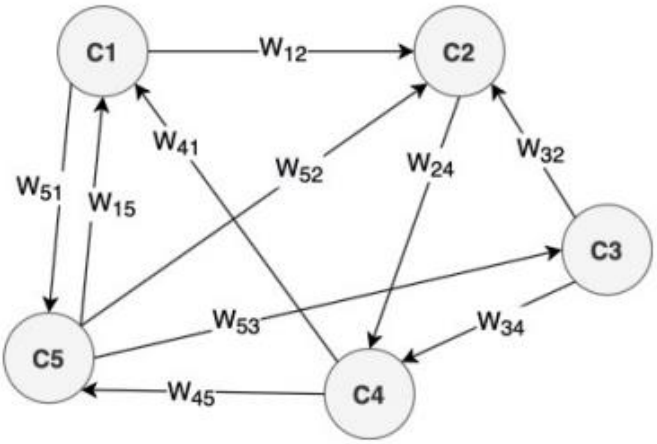


Рис.2. Нечёткая когнитивная карта

С математической точки зрения взвешенные когнитивные карты описываются кортежем из 3-х элементов:

$\langle \mathrm{C}, \mathrm{E}, W \rangle$ (1)

где $C = \{C_1, C_2, C_3, \ldots, C_N\}$ – множество факторов (концептов) рассматриваемой задачи, которые являются вершинами ориентированного графа, $\mathrm{E} = \{\mathrm{e}_{11}, \mathrm{e}_{12}, \ldots, \mathrm{e}_{ij}, \ldots, \mathrm{e}_{NN}\}$ – множество дуг ориентированного графа (взаимосвязей между концептами C_i и C_j), $W = \{\omega_{11}, \omega_{12}, \ldots, \omega_{ij}, \ldots, \omega_{NN}\}$, $\omega_{ij} \in [-1, +1]$ – множество весов взаимосвязей между концептами C_i и C_j.

Интерпретация причинно-следственной связи ω_{ij} между двумя концептами C_i и C_j, следующая:

если $\omega_{ij} > 0$, то усиление (ослабление) концепта C_i приведет к усилению (ослаблению) концепта C_j с интенсивностью $|\omega_{ij}|$;

если $\omega_{ij} < 0$, то усиление (ослабление) концепта C_i, приведет к ослаблению (усилению) концепта Cj с интенсивностью $|\omega_{ij}|$;

если $\omega_{ij} = 0$ (или очень близок к 0), это говорит об отсутствии причинно-следственной связи между C_i и C_j поэтому в графе нет ребра с причинно-следственной связью.

По приведённой математической модели можно выделить несколько видов взвешенных когнитивных карт:

1. знаковая КК – когнитивная карта, для которой $\omega_{ij} \in \{-1,0,+1\}$;
2. простая взвешенная КК – когнитивная карта, для которой $\omega_{ij} \in [-1,+1]$;
3. функциональная взвешенная КК – когнитивная карта, в которой степень влияния оценивается как отдельная функция взаимосвязи двух факторов $\omega_{ij} = f(C_i, C_j, \omega_{ij}), \omega_{ij} \in [-1,+1]$;

Нечеткая КК – когнитивная карта, математическая модель которой описывается следующим кортежем из 3-х элементов:

$$\langle C, E, W, A, f \rangle \quad (2)$$

где C, E и W – имеют тоже значение, что и для формы описанной кортежем (1), только W принимает нечеткие значения $A{:}X_i(t) \rightarrow A_i(t)$ – функция, ставящая в соответствие каждому значению фактора C_i в момент времени t его активационное значение, f – функция трансформации, необходимая для преобразования значений, полученных в ходе моделирования [4].

2. Методы обучения

Задача обучения КК состоит в построении матрицы весов W(M×M), которая базируется на оценках экспертов или имеющихся данных. В основном, в классических методах обучения считается, что набор концептов всегда формируется экспертом и обучение касается только матрицы весов. Алгоритмы обучения делятся на три типа: алгоритм обучения Хебба, Силова и алгоритмы, которые зависят от ошибки.

Алгоритм обучения Хебба

Хеббовские обучающие алгоритмы - это методы обучения без учителя, которым не требуется обучающая выборка, то есть показатели целевых концептов, для которых будет идти расчет, заранее неизвестны. Цель обучения нечётких когнитивных карт при использовании алгоритмов Хебба - получение матриц весов на основе знаний экспертов и повышение точности ранее определенных весов. Этот алгоритм был предложен Бартом Коско и Джулией Дикерсон в 1993. Суть обучения весов в их итеративном перерасчете по формуле (3) до тех пор, пока не будет найдена нужная структура. Веса исходящих ребер для каждого концепта в матрице весов изменяются только при изменении соответствующего значения концепта:

$$\omega_{ij}^{(t+1)} = \begin{cases} \omega_{ij}^{(t)} + \eta_t\left(\Delta A_i^{(t)} \Delta A_j^{(t)} - \omega_{ij}^{(t)}\right), & \text{если } \Delta A_i^{(t)} \neq 0 \\ \omega_{ij}^{(t)}, & \text{если } \Delta A_i^{(t)} = 0' \end{cases} \quad (3)$$

где $\omega_{ij}^{(t)}$, $\omega^{(t+1)}_{ij}$ – веса связи между факторами C_i и C_j, в момент времени t и $(t+1)$, $t \in [1,\dots,T]$, η_t – убывающий коэффициент обучения для постепенного замещения старых весов новыми, $\Delta A_i^{(t)}$ – приращение значения концепта C_i в момент времени t:

$$\Delta A_i^{(t)} = A_i^{(t)} - A^{(t-1)}_i. \quad (4)$$

Главный недостаток данного подхода заключается в отсутствии информации о системе в целом. Дифференциальный метод обучения обновляет веса между каждой парой концептов, он принимает в расчет только эти два концепта и игнорирует влияние других концептов. На сегодняшний день существует множество модификаций алгоритма обучения Хебба.

Алгоритм обучения Силова

В модели Силова отношения между концептами КК рассматриваются как элементы нечеткой матрицы смежности для графа карты. Здесь появляется проблема вычисления отрицательных влияний концептов, поскольку веса могут оказаться отрицательными, а операции над нечеткими множествами определены для функции принадлежности на отрезке [0, 1]. В данном алгоритме проблема разрешается увеличением мощности множества концептов вдвое. Чтобы увеличить мощность множества концептов вдвое, производится переход от первичной КК с положительно-отрицательными нечеткими связями $\mathrm{W} = |w_{ij}|$ к нечеткой матрице положительных связей $\mathrm{r} = |r_{ij}|$ размерностью 2n x 2n (n – число концептов) по правилу:

$$\begin{cases} w_{ij} > 0, mo\ r_{2i-1,2j-1} = w_{ij}, r_{2i,2j} = w_{ij} \\ w_{ij} < 0, mo\ r_{2i-1,2j} = w_{ij}, r_{2i,j-1} = -w_{ij} \\ w_{ij} = 0, mo\ r_{2i-1,2j-1} = 0, r_{2i,j-1} = 0 \end{cases} \quad (5)$$

Затем строится транзитивное замыкание матрицы R, чтобы обнаружить скрытое влияния концептов в КК: $\overline{R} = R \vee R^2 \vee \dots R^2$ по формуле: $R^2 = R[\,]R$, где о – выбранная S-конорма. При вычислении транзитивного замыкания получаются нечеткие значения выходного концепта с применением характерных для нечеткой логики операций t-норм над нечеткими значениями приращений входных концептов и весов причинно-следственной связи. При применении связки min передача непосредственного влияния концепта A_i на концепт A_j производится так:

$$A^{(t+1)}_j = \min\,(A_i^t, w_{ij}) \quad (6)$$

После определения скрытых положительных и отрицательных влияний в КК можно получить матрицу $V = (v_{ij}, v_{ij})$, которая дает возможность оценить прямое и обратное влияние компонентов на систему посредством данного преобразования:

$$\begin{cases} v_{ij} = max(r_{2i-1,2j-1}, r_{2i,2j}) \\ v_{ij} = -max(r_{2i-1,2j-1}, r_{2i,2j}) \end{cases} \quad (7)$$

где v_{ij} обозначает силу положительного влияния, а $\overline{v}_{ij}$ - силу отрицательного влияния [2].

Алгоритмы, которые зависят от ошибки

Цель алгоритма обучения, зависящего от ошибки, состоит в формировании матриц, которые минимизируют функцию ошибки в соответствии с разницей между целевыми значениями выходных концептов и найденными выходными значениями на текущем шаге. Эти алгоритмы более затратные, поскольку пытаются подогнать модель под набор ретроспективных данных. Для данного алгоритма необходимо определение целевой функции, требующей оптимизации, что представляет собой основную задачу процесса обучения [2]. Некоторые исследования показали, что данные методы расширяют функциональность КК и повышают их способность к обобщению.

3. Заключение

Таким образом, когнитивные карты представляют собой простой и универсальный инструмент для моделирования и исследования процессов в различных системах и задачах. Применение когнитивных карт позволяет описать структуру системы, оценить её устойчивость, проанализировать переходные процессы в системе.

Подводя итоги, отметим, что методы, основанные на алгоритмах Хебба и Силова, имеют невысокую способность к обобщению. С другой стороны, основной недостаток методов обучения, зависящих от ошибки, в том, что генерируемые ими решения с трудом поддаются интерпретации.

4. Литература

1. Емельяненко А.С. Процесс построения когнитивных карт / А.С. Емельяненко, Д.В. Колесник // Вопросы студенческой науки. – 2019. – №12(40). – С. 309-316.
2. Коваленко А.В. Обзор динамических свойств и алгоритмов обучения нечетких когнитивных карт / А.В. Коваленко, А.В. Петухова, Д.М. Теунаев // Научный журнал КубГАУ. – 2021. – №167(03). – С. 1-20.
3. Нечёткие когнитивные карты. – Режим доступа: https://intuit.ru/studies/courses/3735/977/lecture/. – (Дата обращения: 08.04.2024).
4. Обзор алгоритма обучения нечёткой когнитивной карты на основе нелинейного правила Хэбба. – https://www.vectoreconomy.ru. – (Дата обращения: 08.04.2024).

УДК 004.852

ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ С ПОМОЩЬЮ АЛГОРИТМА ПРЕСНОВОДНЫХ ГИДР

А.А. Петина

Воронежский государственный университет

Введение

В современном обществе нейросети становятся все более актуальными и доступными. Нейронные сети (НС) — это математические модели, а также их программные реализации, построенные по принципу организации и функционирования биологических нейронных сетей. Они способны обрабатывать огромные объемы информации, выявлять закономерности и делать предсказания. Одной из главных причин актуальности нейросетей является их способность анализировать и обрабатывать большие объемы данных с высокой точностью.

Существует множество различных методов обучения НС: метод градиентного спуска, импульсные методы, адаптивные методы, квазиньютоновские методы [1], но ориентация нейронных сетей на различные типы задач обуславливает необходимость разработки и поиска новых подходов обучения, с помощью которых возможно повышение качества работы нейросети.

1. Обучение НС (материалы и методы)

1.1. Теоретические сведения о нейронных сетях

Основной структурной и функциональной частью нейросети является нейрон, модель которого включает три основных элемента:

1) набор синапсов или связей: осуществляют связь между нейронами и умножают входной сигнал на число, характеризующее силу связи, — вес синапса;

2) сумматор: складывает входные сигналы, взвешенные относительно соответствующих синапсов нейрона;

3) функция активации: ограничивает амплитуду выходного сигнала нейрона [2].

Математическая модель нейрона имеет следующий вид:

$$S = \sum_{i=1}^{n} w_i x_i + w_0 ,$$

$$y = f(S),$$

здесь w_i — вес связи, w_0 — значение нейронного смещения, S — результат суммирования, x_i — компонент входного вектора (входной сигнал), y — выходной сигнал нейрона, n — число входов нейрона, f — функция активации [3].

В данной работе использовалась функция активации SoftMax, которая обычно используется для мультиклассовой классификации.

Формально стандартная (единичная) функция SoftMax $\sigma : \mathbb{R}^K \mapsto (0,1)^K$ определяется как $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, где $K \geq 1$, $i = 1,...,K$ и $\mathbf{z} = (z_1,...,z_K) \in \mathbb{R}^K$ [4].

В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа: входные нейроны, на которые подаются входные сигналы; выходные нейроны, значения которых представляют выходы всей нейронной сети; скрытые нейроны, которые не имеют связей с входными сигналами, и их выходные значения не являются выходами сети.

С точки зрения топологии можно выделить три основных типа НС [3]: полносвязные; многослойные; слабосвязные (с локальными связями).

В данной работе каждый нейрон одного слоя связан с каждым нейроном соседнего слоя, то есть реализована многослойная сеть.

1.2. Постановка задачи обучения

Обучение нейронной сети происходит путем настройки весов связей между нейронами на основе обучающих данных. Это достигается с помощью минимизации ошибки между выходом сети и ожидаемым результатом. Процесс обучения – это задача условной оптимизации, которая в общем случае сводится к задаче поиска экстремума (минимума или максимума) целевой функции с заданными ограничениями и имеет следующий вид:

$$\mathbf{x}^{opt} = \arg\left(\min_{\mathbf{x} \in D} f(\mathbf{x})\right).$$

Её математическая постановка выглядит следующим образом: необходимо найти такой вектор переменных $\mathbf{x}$, при котором функция $f(\mathbf{x})$ достигает глобального экстремума в некоторой замкнутой n-мерной области D. При этом функция $f(\mathbf{x})$ может иметь множество локальных экстремумов и не выражаться аналитически.

Для решения поставленной задачи успешно применяются популяционные алгоритмы оптимизации, основанные на механизмах, заимствованных из живой природы. Популяционные алгоритмы работают сразу с несколькими допустимыми векторами из области D, которые называются особями, агентами или частицами. Совокупность особей образует популяцию. Особь с наилучшим значением целевой функции называется лидером [5].

На каждом шаге решения особи популяции перемещаются в соответствии с конкретным алгоритмом. В зависимости от конкретного популяционного алгоритма у особи есть стратегия поведения. Это формирует сложное поведение популяции, которое называется роевым интеллектом [6]. Одним из таких алгоритмов является алгоритм пресноводных гидр (АПГ) [5]. В отличие от классических популяционных алгоритмов в АПГ координаты частиц определяются непосредственно с некоторой вероятностью, что позволяет увеличить охват пространства поиска, а также повысить устойчивость к попаданию в локальный экстремум и преждевременной сходимости [5], именно поэтому далее будет рассмотрен именно алгоритм пресноводных гидр.

После выбора архитектуры сети следует «обучить» выбранную нейросеть, то есть подобрать такие значения ее весовых коэффициентов, чтобы полученный набор весов минимизировал ошибку аппроксимации. В начале работы алгоритма нет абсолютно никакой информации о направлении движения в плане настройки весовых коэффициентов матрицы. В условиях неопределенности генетические алгоритмы имеют высокие шансы для достижения требуемых результатов.

Рассмотрим математическую модель поставленной задачи. Дано множество векторов $\mathbf{X}$ — входов НС, множество векторов $\mathbf{Y}$ — требуемые результаты распознавания. Задача заключается в том, чтобы найти весовую матрицу $\mathbf{W}$, элементами которой являются вещественные числа на интервале $[0,1]$, чтобы выполнялось равенство для всех векторов $\mathbf{X}$ и $\mathbf{Y}$ соответственно

$$\mathbf{X} \times \mathbf{W} = \mathbf{Y}.$$

Необходимо осуществить настройку весовой матрицы $\mathbf{W}$ с помощью генетического алгоритма [7].

Генетические алгоритмы для обучения нейросети применяются как альтернатива методу обратного распространения ошибки. Целью обучения является минимизация логарифмической функции потерь (кросс-энтропия)

$$H(P,Q) = -\sum_{x} P(x) \log Q(x),$$

где P — распределение истинных ответов, а Q — распределение вероятностей прогнозов модели.

Обучение методом обратного распространения ошибки сводится к подбору значений весов прямонаправленной нейронной сети, основываясь на принципах наискорейшего спуска. Один из основных недостатков этого классического алгоритма состоит в возможном попадании в локальные минимумы функции стоимости. Применение генетического алгоритма позволяет избежать этой проблемы. На первом этапе происходит генерация начальной популяции весовых коэффициентов (ВК). Каждая особь популяции представляет собой набор весовых коэффициентов НС. Затем происходит оценка значения приспособленности каждой особи. Вычисление фитнес-функции основано на результате работы нейронной сети и состоит из следующих этапов [8]: инициализация ВК; запуск НС; оценка правильных ответов сети. На основании значения пригодности каждой особи, происходит выполнение операторов переноса особей в различных направлениях (к лидеру, к среднему положению особей, в случайном направлении). В результате операций появляется новая популяция ВК, более пригодная для дальнейшего развития [8]. Таким образом происходит настройка весовых коэффициентов. Критерием остановки является достижение заданного количества эпох.

1.3. Описание алгоритма пресноводных гидр

В алгоритм заложены механизмы, подобные поведению пресноводных гидр. Так, при наступлении неблагоприятных условий среды, гидра погибает, выпуская в воду яйцо, переносимое с током воды, из которого вырастает новая особь гидры. Этот факт формализован в алгоритме с помощью оператора переноса. Особь гидры может перемещаться к средней лучшей позиции всех особей популяции, к особи с наилучшим значением целевой функции за все время поиска и в случайном направлении. При этом выбор направления выполняется по-разному. В первом варианте алгоритма (QH-AHP-алгоритм) для этого применяется метод анализа иерархий (AHP), а во втором (QH-B-алгоритм) — байесовский подход [5].

Модификация на основе метода анализа иерархий (QH-AHP-алгоритм)

Направление движения особи определяется с помощью метода анализа иерархий (Analytic Hierarchy Process, AHP). Метод позволяет выбирать решение среди p возможных

вариантов. Предпочтения в пользу того или иного варианта выражаются матрицей парных сравнений $\mathbf{A}$. Ее элементы $a_{ij} \in \left\{\frac{1}{9},...,1,2,...,9\right\}$ показывают, во сколько раз i-й вариант лучше j-го ($i, j = \overline{1, p}$), причем $a_{ij} = \frac{1}{a_{ji}}$ и $a_{ii} = 1$. Нормируя элементы матрицы $\mathbf{A}$ по правилу $b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{p} a_{ij}}$, сформируем матрицу $\mathbf{B}$. Начальные значения элементов матрицы $\mathbf{A}$ представлены в табл. 1.

Таблица 1

Начальные значения элементов матриц парных сравнений

Направление	К средней лучшей позиции	К лидеру	Случайное
К средней лучшей позиции	1	$\frac{1}{3}$	$\frac{1}{7}$
К лидеру	3	1	$\frac{1}{5}$
Случайное	7	5	1

Алгоритм [5]:

Шаг I. Создать начальную популяцию. Для каждой j-й особи задать матрицу парных сравнений $\mathbf{A}_j$ (см. табл. 1), определить значение целевой функции, положить число итераций τ_j, в течение которых целевая функция не улучшалась, равным нулю. Задать номер итерации $k = 1$. Найти лидера.

Шаг II. Определить средний вектор $\mathbf{C}$ лучших положений особей популяции и найти значение параметра λ.

Шаг III. Для каждой j-й особи выполнить следующее:

1. Найти вектор приоритетов $\mathbf{w}$.
2. Выполнить шаг в направлении с наибольшим приоритетом $p_{\max} = \arg\max_p w_p$.

Если он одинаков у нескольких направлений, то выбрать любое из них.

3. Вычислить значение целевой функции. Если оно улучшилось, то в строке $i = p_{\max}$ матрицы $\mathbf{A}$ увеличить элементы a_{ij} (например, на 1, если $a_{ij} > 1$, и на 0,1, если $0 < a_{ij} < 1$). Если при этом значение a_{ij} оказывается вне диапазона $[0{,}1; 9]$, то положить его равным ближайшему граничному значению. Положить τ_j равным нулю. Обновить лучшее положение особи $\mathbf{x}_j^*$.

4. Если значение целевой функции ухудшилось, то уменьшить соответствующие элементы матрицы $\mathbf{A}$. Увеличить τ_j на единицу.

5. Пересчитать элементы, симметричные тем, которые изменились: $a_{ji} = \frac{1}{a_{ij}}$.

6. Если $\tau_j = \tau_{\max}$, то применить оператор переноса, рассчитать значение целевой функции в новой точке и вернуть элементам матрицы $\mathbf{A}_j$ исходные значения (см. табл. 1).

Шаг IV. Найти лидера и проверить выполнение критерия останова:

$$\chi = \max_q \left| \frac{f^k - f^{k-q}}{f^k} \right| < 10^{-4},$$

где $q = 1, ..., \min(k, \omega)$; значение параметра ω принимается равным 100 итерациям.

В случае выполнения завершить поиск. Положить $k = k + 1$ и перейти к шагу II.

Модификация на основе байесовского подхода (QH-B-алгоритм)

Для выбора направления движения особи используется байесовский подход. При этом рассматриваются следующие случайные события (гипотезы): H_1 — особь перемещается к вектору $\mathbf{C}$; H_2 — особь движется к лидеру; H_3 — перемещение особи случайным образом.

В результате перемещения особи появляются следующие свидетельства: A — значение целевой функции особи улучшилось; $\bar{A}$ — значение ухудшилось.

На основе события A делается вывод о правильности той или иной гипотезы. Предполагается, что события H_1, H_2, H_3 образуют полную группу событий. При использовании байесовского подхода исходные априорные вероятности гипотез $P(H_1), P(H_2), P(H_3)$ последовательно пересчитываются на основе поступающих свидетельств A или $\bar{A}$:

1) Если значение целевой функции улучшилось (наступило событие A), то вероятность $P(H_r)$ равна апостериорным вероятностям формулы:

$$P(H_r \mid A) = \frac{P(H_r) \cdot P(A \mid H_r)}{\sum_{s=1}^{3} P(H_s) \cdot P(A \mid H_s)},$$

где $r \in \{1, 2, 3\}$.

2) Если значение целевой функции ухудшилось, то вероятность $P(H_r)$ вычисляется по формуле:

$$P(H_r \mid \bar{A}) = \frac{P(H_r) \cdot P(\bar{A} \mid H_r)}{\sum_{s=1}^{3} P(H_s) \cdot P(\bar{A} \mid H_s)},$$

где $r \in \{1, 2, 3\}$.

2. Обсуждение результатов

2.1. Программная реализация алгоритма обучения

Разработанная программа вычисляет ошибки выхода нейронных сетей, обученных с помощью алгоритма пресноводных гидр и метода обратного распространения ошибки, с целью последующего сравнительного анализа качества работы НС.

Для программной реализации алгоритмов обучения был выбран объектно-ориентированный язык программирования C#. В следующей таблице представлены основные классы и их методы.

Таблица 2

Классы и методы

Класс Population	
`void FindLeaderMin()`	поиск лидера в задаче на минимум
`void FindC()`	ищет средний вектор лучших положений
`void FindThePriorityVector()`	поиск вектора приоритетов
`void StepInTheHighestPriorityDirection()`	для каждой особи определяет шаг в более приоритетном для нее направлении
`bool IfStop()`	проверка критерия останова
Класс Individual	
`void MoveToC()`	движение к среднему вектору лучших положений
`void MoveToLeader()`	движение в направлении к лидеру
`void MoveToRandomDirection()`	движение особи в случайном направлении
`void TransferOperator()`	реализация оператора переноса
Класс NeuralNetwork	
`void Activation()`	реализация функции активации нейрона
`void DirectDistribution()`	считает выход нейронной сети на заданных входных параметрах

2.2. Организация вычислительного эксперимента

Целью вычислительного эксперимента является исследование поведения нейронной сети на определенном наборе данных при заданных оптимальных параметрах популяционного алгоритма [10]. Сравнение ошибок выхода разработанных нейронных сетей, реализованных с помощью QH-AHP-алгоритма, QH-B-алгоритма и метода обратного распространения ошибки, выполнялось на примере классификации входных данных по двум классам. В качестве диапазона изменения всех весовых коэффициентов x_i был выбран промежуток $[0,1]$. Рассматривались функции от $n=1$ переменной. Результаты работы нейронных сетей исследовались в зависимости от количества скрытых слоев нейронной сети и количества эпох.

2.3. Анализ результатов и выводы

На рис. 1 и рис. 2 показаны графики зависимости ошибки выхода нейронной сети от количества эпох и скрытых слоев в архитектуре НС.

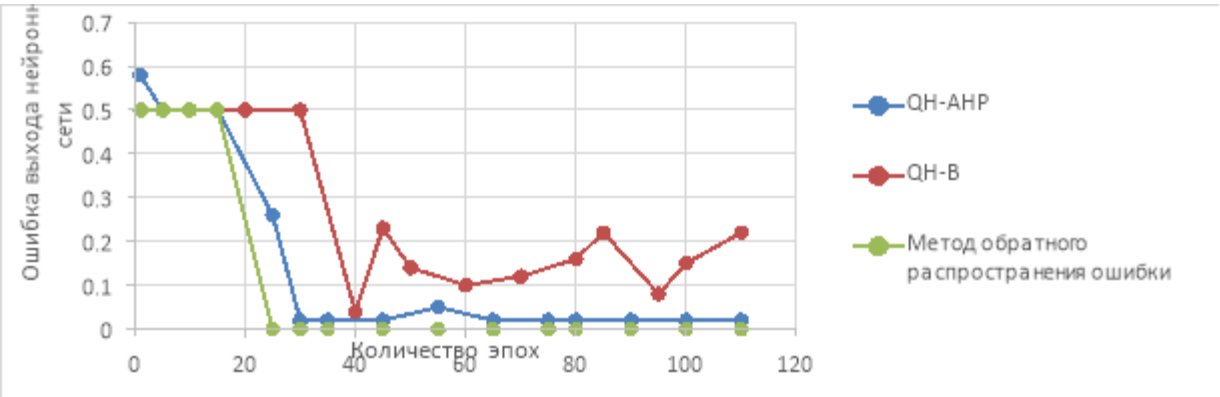


Рис. 1. График зависимости ошибки выхода НС от количества эпох

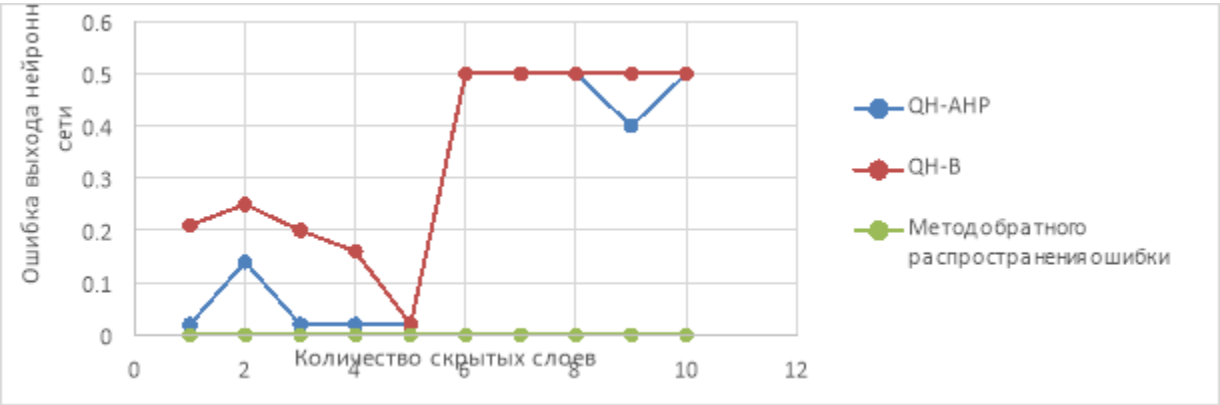


Рис. 2. График зависимости ошибки выхода НС от количества скрытых слоев

В проведенном эксперименте АПГ, основанный на методе анализа иерархий, показал более точные результаты, чем алгоритм, использующий байесовский подход, который имеет более резкие и частые изменения значения выхода нейронной сети.

Не трудно заметить, что разработанные НС, использующие рассмотренный АПГ, ухудшают свои результаты при увеличении количества скрытых слоев. Можно предположить, что это происходит вследствие возрастания вычислительной сложности алгоритмов при поиске оптимального решения.

Также следует отметить, что поведение различных модификаций алгоритма пресноводных гидр схоже между собой, так как в реализацию заложен одинаковый принцип поиска оптимального решения – движение в некотором направлении, в алгоритмах различается лишь выбор самого перемещения.

В табл. 3 представлены исследуемые методы обучения и их минимальная ошибка выхода НС в порядке эффективности работы алгоритмов на данном наборе данных.

Таблица 3

Методы обучения в порядке эффективности работы алгоритмов

Название метода	**Минимальная ошибка выхода НС**
Метод обратного распространения ошибки	0
QH-AHP-алгоритм	0,02
QH-B-алгоритм	0,02

Заключение

Таким образом, в результате работы была разработана программа обучения нейронных сетей с использованием алгоритма пресноводных гидр и были достигнуты следующие цели:

- сформулированы понятия АПГ и нейронных сетей;
- была разработана программа обучения НС;
- проведен сравнительный анализ влияния параметров нейронной сети на ее эффективность.

Литература

1. Каширина, И.Л. Исследование и сравнительный анализ методов оптимизации, используемых при обучении нейронных сетей/ И. Л. Каширина, М. В. Демченко // Вестник ВГУ, серия: системный анализ и информационные технологии. – 2018. – №4. – С. 123–132.
2. Полякова А. С. О настройке нейронных сетей при помощи генетического алгоритма / А. С. Полякова // Актуальные проблемы авиации и космонавтики, серия: информационные технологии. – 2014. – С. 297–298.
3. Цой Ю. Р. Разработка генетического алгоритма настройки искусственной нейронной сети: дис. … канд. техн. наук. – Томск: ТПУ, 2004. – 62 с.
4. Функция Softmax. – Режим доступа: https://translated.turbopages.org. – (Дата обращения: 14.01.2024).
5. Королев, А.С. Квантовая модификация алгоритма пресноводных гидр для решения задачи оптимизации / А. С. Королев, Д. В. Майков // Вестник ВГУ, серия: системный анализ и информационные технологии. – 2020. – №2. – С. 37–48.
6. Матренин П. В. Разработка адаптивных алгоритмов роевого интеллекта а проектировании и управлении техническими системами: дис. … степени канд. техн. наук. – Новосибирск: НГТУ, 2018. – 197 с.
7. Мищенко В.А., Коробкин А.А. Использование генетических алгоритмов в обучении нейронных сетей // Современные проблемы науки и образования. 2011. № 6. URL: http://www.science-education.ru/ru/article/view?id=5138 (дата обращения: 01.04.2015).
8. Аталов Е. В. Подсистема обучения нейронных сетей с использованием генетических алгоритмов: дис. … степени магистра техники и технологии. – Пенза: ПГУ, 2017. – 116 с.
9. Разработка программного комплекса для синтеза популяционных алгоритмов. – Режим доступа: https://referat.yabotanik.ru. – (Дата обращения: 15.03.2015).
10. Петина, А. А. Программная реализация алгоритма пресноводных гидр / А.А. Петина // Математика, информационные технологии, приложения: сб. тр. науч. конф. молодых ученых и студентов – Воронеж: ВГУ, 2023. – С. 364–371.

УДК 517.926+51.76

МОДЕЛЬ ЗАГРЯЗНЕНИЯ ВОДОЁМОВ ОРГАНИЧЕСКИМИ ОТХОДАМИ

А.И. Петренко

Воронежский государственный университет

Введение

Загрязнение водоёмов органическими отходами, такими как токсические вещества, стоки городов и предприятий хозяйственной, химической, текстильной и пищевой промышленности, представляет собой одну из основных экологических проблем современного общества. Моделирование загрязнения водоёмов является важной задачей экологии, поскольку качество воды напрямую влияет на здоровье людей и экосистему.

В данной работе рассматривается математическая модель, описывающая экологическую систему, включающую в себя воду и растворённые в ней кислород и органические отходы. Взаимосвязь между концентрацией растворённого кислорода и органическими отходами играет ключевую роль в понимании процессов, происходящих в данной системе. Модель такой системы была получена Стритером и Фелпсом.

1. Исследование классической модели Стритера-Фелпса

1.1. Решение системы

Классическая модель Стритера-Фелпса описывается системой линейных дифференциальных уравнений (1), с начальными условиями (2)

$$\begin{cases} \dfrac{dL}{dt} = -k_1 L \\ \dfrac{dD}{dt} = k_1 L - k_2 D \end{cases} \tag{1}$$

$$\begin{cases} L(0) = L_0 \\ D(0) = D_0 \end{cases} \tag{2}$$

где k_1 – коэффициент отбора кислорода (1/день); k_2 – коэффициент реаэрации (1/день); L – концентрация отходов (мг/л); D – дефицит кислорода (мг/л).

Найдем характеристические числа и соответствующие им собственные вектора системы (1)

$$A = \begin{pmatrix} -k_1 & 0 \\ k_1 & -k_2 \end{pmatrix} \qquad \begin{vmatrix} -k_1 - \lambda & 0 \\ k_1 & -k_2 - \lambda \end{vmatrix} = 0$$

$$\lambda^2 + k_1\lambda + k_2\lambda + k_1 k_2 = 0$$

$$\lambda_1 = -k_1, \lambda_2 = -k_2$$

$$1)\lambda_1 = -k_1 : \begin{pmatrix} 0 & 0 \\ k_1 & k_1 - k_2 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},\ \alpha = \frac{(k_1 - k_2)\beta}{k_1} = \left(\frac{k_2}{k_1} - 1\right)\beta, \forall \beta \Rightarrow h_1 = \begin{pmatrix} k_2 - k_1 \\ k_1 \end{pmatrix}$$

$$2)\lambda_2 = -k_2 : \begin{pmatrix} k_2 - k_1 & 0 \\ k_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{cases} (k_2 - k_1)\gamma = 0 \\ k_1\gamma = 0 \end{cases} \begin{cases} \gamma = 0 \\ \forall \delta \end{cases} \Rightarrow h_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Тогда общее решение системы (1) имеет вид

$$\begin{cases} L(t) = c_1(k_2 - k_1)e^{-k_1 t} \\ D(t) = c_1 k_1 e^{-k_1 t} + c_2 e^{-k_2 t} \end{cases} \quad (3)$$

Воспользуемся начальными условиями (2), получаем

$$\begin{cases} L(0) = c_1(k_2 - k_1) = L_0 \\ D(0) = c_1 k_1 + c_2 = D_0 \end{cases} \begin{cases} c_1 = \dfrac{L_0}{k_2 - k_1} \\ c_2 = D_0 - \dfrac{k_1}{k_2 - k_1} L_0 \end{cases}$$

Тогда решение задачи Коши (1), (2) описывают зависимости

$$\begin{cases} L(t) = L_0 e^{-k_1 t} \\ D(t) = \dfrac{k_1}{k_2 - k_1} L_0 (e^{-k_1 t} - e^{-k_2 t}) + D_0 e^{-k_2 t} \end{cases} \quad (4)$$

Здесь $L(t)$ описывает изменение концентрации органических отходов, а $D(t)$ – изменение дефицита кислорода с течением времени. Для полученного решения (4) был проведён анализ зависимости $L(t)$ и $D(t)$ от различных факторов.

1.2. Зависимость от временного интервала

На примере больших и средних рек с температурой воды $t = 20^\circ C$ и медленным течением, по берегам которых расположены промышленные предприятия со стоками, содержащими токсические вещества $(k_1 = 0.0103; k_2 = 0.198)$, было исследовано изменение концентрации органических отходов и дефицита кислорода в зависимости от увеличения временного интервала при начальных значениях $L_0 = 5$ мг/л и $D_0 = 2$ мг/л.

Анализ полученных конечных значений концентрации отходов и дефицита кислорода при временных интервалах равных 30, 40 и 60 суткам позволяет сделать вывод, что при более длительном временном интервале происходит естественное самоочищение водоёма, приводящее к снижению содержания органических отходов и увеличению уровня кислорода.

1.3. Зависимость от характеристики водного объекта

В данном пункте рассматривается зависимость $L(t)$ и $D(t)$ от вида водоёма и типа загрязнения. Для всех приведённых ниже случаев начальные значения выбраны $L_0 = 5$, $D_0 = 2$. Для более удобного анализа конечных значений дефицита кислорода и концентрации органических отходов в зависимости от характеристики водного объекта составлена табл.1.

По полученным данным можно сделать вывод о том, что для всех видов водоёмов наивысший уровень содержания кислорода (наименьший дефицит) наблюдается при загрязнении смешанными стоками городов и предприятий пищевой промышленности. Водные объекты, находящиеся под воздействием хозяйственных стоков, имеют более низкий уровень кислорода и большее количество отходов. Два других типа загрязнений приводят к различным

конечным значениям параметров, в зависимости от вида водного объекта. Так, в больших реках с медленным течением и слабопроточных водоёмах, последующими по убыванию уровня кислорода идут стоки предприятий, выбрасывающих токсические вещества, а самый низкий уровень кислорода наблюдается при загрязнении стоками химической и текстильной промышленности. Стоит отметить, что в данном случае содержание органических отходов ниже, чем в случае загрязнения токсическими веществами. А вот в малых реках с быстрым течением стоки, содержащие токсические вещества, оказывают более сильное негативное влияние, чем стоки химической и текстильной промышленности.

Таблица 1

Значения $L(t)$ и $D(t)$ для различных водоёмов

	Слабопроточные или стоячие водоёмы ($k_2 = 0.146$)	Большие и средние реки с медленным течением ($k_2 = 0.198$)	Малые реки с быстрым течением ($k_2 = 0.802$)
Смешанные стоки городов и предприятий пищевой промышленности ($k_1 = 0.3017$)	$L(t) = 0.000586$ $D(t) = 0.145266$	$L(t) = 0.000586$ $D(t) = 0.0418456$	$L(t) = 0.000586$ $D(t) = 0.000353603$
Хозяйственные стоки ($k_1 = 0.1983$)	$L(t) = 0.0130422$ $D(t) = 0.213055$	$L(t) = 0.0130422$ $D(t) = 0.0832025$	$L(t) = 0.0130422$ $D(t) = 0.00428404$
Стоки предприятий химической и текстильной промышленности ($k_1 = 0.1034$)	$L(t) = 0.224796$ $D(t) = 0.418673$	$L(t) = 0.224796$ $D(t) = 0.236587$	$L(t) = 0.224796$ $D(t) = 0.0332721$
Стоки предприятий, содержащие токсические вещества ($k_1 = 0.0103$)	$L(t) = 3{,}6709$ $D(t) = 0.298929$	$L(t) = 3{,}6709$ $D(t) = 0.205982$	$L(t) = 3{,}6709$ $D(t) = 0.0477584$

Таким образом, можно сделать вывод, что уровень содержания кислорода и отходов в реках зависит от характеристики стоков, поступающих в водоёмы. Также следует отметить, что вне зависимости от типа реки, наиболее интенсивное загрязнение водных потоков происходит за счёт токсических веществ и стоков химической и текстильной промышленности.

Если рассматривать изменение дефицита кислорода в зависимости от вида водоёма, то значение дефицита кислорода будет наименьшим в малых реках с быстрым течением, а наибольшим в слабопроточных или стоячих водоёмах, вне зависимости от вида стоков. При этом, концентрация органических отходов будет одинакова для разных водоёмов при совпадении типа загрязнения.

2. Исследование модели Стритера-Фелпса с управляющим фактором

2.1. Решение системы

Предположим, что за процессом загрязнения воды осуществляется контроль, например, с помощью очистных сооружений. В этом случае, для более точного описания процессов

загрязнения, в систему (1) добавляется управляющий фактор u_0 $(0 < u_0 \le 1)$, который представляет собой количество отходов, удалённых в единицу времени. Получается следующая линейная неоднородная система (ЛНС) дифференциальных уравнений

$$\begin{cases} \dfrac{dL}{dt} = -k_1 L - u_0 \\ \dfrac{dD}{dt} = k_1 L - k_2 D \end{cases} \tag{5}$$

с начальными условиями (2).

Общее решение соответствующей однородной системы было получено ранее и записано в виде системы (3).

Решение ЛНС (5) ищется следующим образом

$$f(t) = \begin{pmatrix} u_0 \\ 0 \end{pmatrix} \quad \begin{cases} L = A \\ D = B \end{cases} \quad \begin{cases} -k_1 A - u_0 = 0 \\ k_1 A - k_2 B = 0 \end{cases} \quad \begin{cases} A = -\dfrac{u_0}{k_1} \\ B = -\dfrac{u_0}{k_2} \end{cases} \quad \begin{cases} L = -\dfrac{u_0}{k_1} \\ D = -\dfrac{u_0}{k_2} \end{cases}$$

Тогда общее решение системы (5) имеет вид

$$\begin{cases} L_{он}(t) = c_1(k_2 - k_1)e^{-k_1 t} - \dfrac{u_0}{k_1} \\ D_{он}(t) = c_1 k_1 e^{-k_1 t} + c_2 e^{-k_2 t} - \dfrac{u_0}{k_2} \end{cases}$$

Потребуем, чтобы полученное решение удовлетворяло начальным условиям (2)

$$\begin{cases} L(0) = c_1(k_2 - k_1) - \dfrac{u_0}{k_1} = L_0 \\ D(0) = c_1 k_1 + c_2 - \dfrac{u_0}{k_2} = D_0 \end{cases} \quad \begin{cases} c_1 = \dfrac{k_1 L_0 + u_0}{k_1 k_2 - k_1^2} \\ c_2 = D_0 - \dfrac{k_1 L_0 + u_0}{k_2 - k_1} + \dfrac{u_0}{k_2} \end{cases}$$

Тогда решение задачи (5), (2) принимает вид

$$\begin{cases} L_{он}(t) = \dfrac{k_1 L_0 + u_0 - u_0 e^{k_1 t}}{k_1 e^{k_1 t}} \\ D_{он}(t) = \dfrac{-k_1 e^{k_1 t} u_0 + k_2 e^{k_2 t} u_0 + k_1 e^{k_1 t + k_2 t} u_0 - k_2 e^{k_1 t + k_2 t} u_0 - k_1 k_2 e^{k_1 t} L_0 + k_1 k_2 e^{k_2 t} L_0 - k_1 k_2 e^{k_1 t} D_0 + k_1 k_2 e^{k_2 t} D_0}{(k_1 - k_2) k_2 e^{k_1 t + k_2 t}} \end{cases}$$

Для полученного решения проведён анализ зависимости $L(t)$ и $D(t)$ от различных факторов.

2.2. Зависимость от изменения u_0

Для больших и средних рек с температурой воды $t = 20^{\circ}C$ и медленным течением ($k_2 = 0.198$), которые подвержены загрязнению стоками, содержащими токсические вещества

Таблица 2

Значения $L(t)$ и $D(t)$ при различных значениях u_0

u_0	Концентрация органических отходов	Дефицит кислорода
0.01	*3.41283*	*0.194584*
0.05	2.38052	0.148993
0.07	*1.86437*	*0.126198*
0.1	*1.09013*	*0.0920044*
0.15	*-0.20025*	*0.0350156*

($k_1 = 0.0103$), проведен анализ изменения количества органических отходов и дефицита кислорода при увеличении значения количества отходов, удалённых в единицу времени u_0 . Для более удобного сравнения конечных значений $L(t)$ и $D(t)$ составлена табл.2.

Исходя из данных табл. 2, можно сделать вывод, что при увеличении количества отходов, удаляемых в единицу времени, концентрация органических отходов и дефицит кислорода уменьшаются, следовательно, установка очистных сооружений оказывает положительное влияние на состояние водоёмов. Однако, если взять слишком большие значения u_0, то задача теряет содержательный смысл. Это можно увидеть в последней строке табл.2 (концентрация органических отходов при $u_0 = 0.15$ принимает отрицательное значение).

2.3. Зависимость от характеристики водного объекта при фиксированном u_0

Исследование изменения концентрации отходов и дефицита кислорода при постоянном значении $u_0 = 0.05$ проведено для тех же видов водоёмов и загрязнений, которые были рассмотрены ранее. В ходе анализа выявлено, что для каждого типа водного объекта задача теряет содержательный смысл при всех видах загрязнений, за исключением ситуации, когда по берегам расположены промышленные предприятия со стоками, содержащими токсические вещества.

3. Сравнение неуправляемой водной системы и управляемой извне

Проанализируем эффективность установки очистных сооружений, сравнив уровни концентрации органических отходов и дефицита кислорода до и после установки очистных сооружений в различных водоёмах, прилегающих к промышленным предприятиям, стоки которых содержат токсические вещества (рис.1, рис.2, рис.3).

Из представленных данных видно, что управляемая система с $u_0 = 0.05$ показывает более низкую концентрацию органических отходов и меньший дефицит кислорода по сравнению с неуправляемой системой. В особенности, в слабопроточных или стоячих водоёмах и в больших и средних реках с медленным течением происходит более значительное улучшение показателей дефицита кислорода, чем в малых реках с быстрым течением. Таким образом, можно заключить, что установка очистных сооружений эффективна и способствует улучшению экологического состояния водоёмов.

Конечные значения до установки очистных сооружений: **L(t) = 3.6709, D(t) = 0.205982**

Конечные значения после установки очистных сооружений: **L(t) = 2.38052, D(t) = 0.148993**

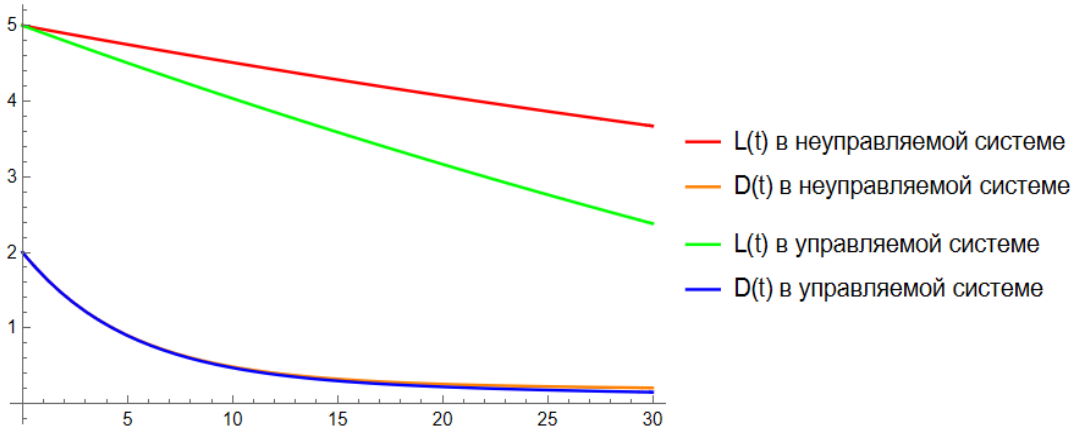


Рис. 1. Графики для больших и средних рек с медленным течением

Конечные значения до установки очистных сооружений: **L(t) = 3.6709, D(t) = 0.298929**

Конечные значения после установки очистных сооружений: **L(t) = 2.38052, D(t) = 0.226654**

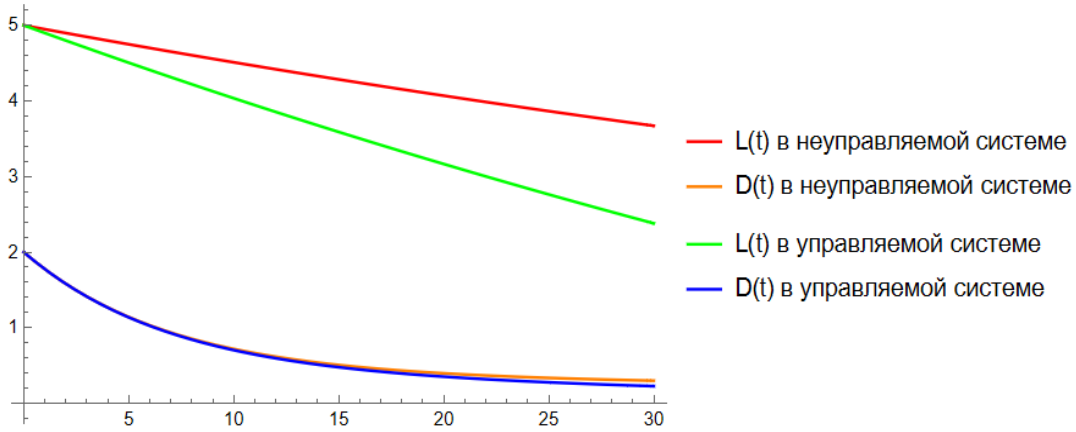


Рис. 2. Графики для слабопроточных или стоячих водоёмов

Конечные значения до установки очистных сооружений: **L(t) = 3.6709, D(t) = 0.0477584**

Конечные значения после установки очистных сооружений: **L(t) = 2.38052, D(t) = 0.0317816**

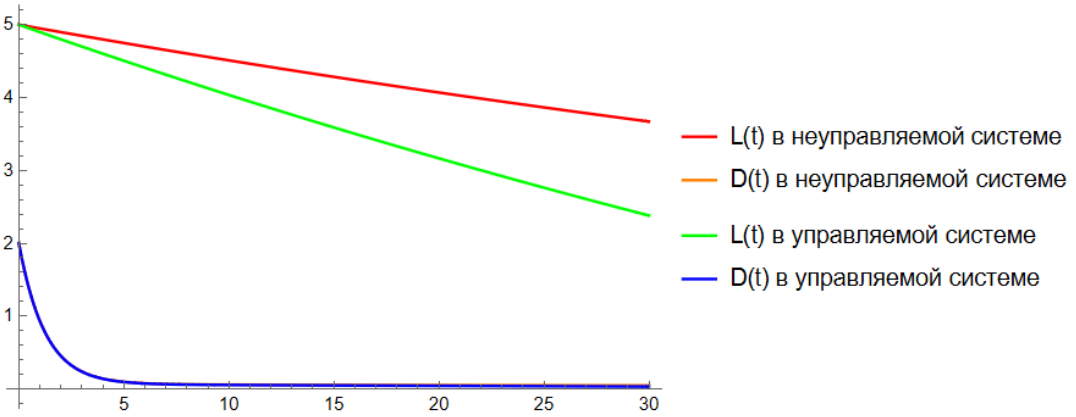


Рис. 3. Графики для малых рек с быстрым течением

Заключение

В данной работе была рассмотрена модель загрязнения водоёмов органическими веществами, охватывающая различные типы водных объектов: большие и средние реки с низкой скоростью течения, стоячие водоёмы и малые реки с быстрым течением.

Каждый из этих объектов проявляет различную чувствительность к загрязнению. В рамках исследования данной проблемы, была решена система, учитывающая изменение коэффициентов в зависимости от типа водоёма и загрязнения. По полученному решению были построены графики и проведён анализ. Исследования показали, что концентрация дефицита кислорода и органических отходов в водоёме снижается при увеличении временного интервала. Реки с быстрым течением обладают более высоким конечным содержанием кислорода по сравнению с медленно текущими или стоячими водоёмами.

Кроме того, установлено, что уровень кислорода и содержание органических веществ в воде зависят от типа сточных вод. В результате этого было выявлено, что стоки городов и предприятий пищевой промышленности оказывают наименьшее воздействие на водные объекты, в то время как водоёмы, загрязняемые токсическими веществами и стоками предприятий химической и текстильной промышленности, имеют наибольшее содержание органических отходов и высокий дефицит кислорода.

Наряду с классической системой Стритера-Фелпса, описывающей водную систему, рассмотрена водная система, управляемая извне. На основе полученных решений построены графики, отражающие динамику изменений параметров в зависимости от воздействия очистных сооружений. Анализ этих графиков позволил выявить положительное влияние таких сооружений на изменение уровня кислорода и концентрации органических отходов в водоёмах. Исходя из этого, можно сделать вывод о том, что наличие очистных сооружений играет важную роль в улучшении состояния водных объектов.

Список литературы

1. Местецкий, Л.М. Математические модели в экологии: учеб. пособие / Л.М. Местецкий. – Тверь: Твер. гос. ун-т, 1997. – 40 с.
2. Пэнтл, Р. Методы системного анализа окружающей среды / Р. Пэнтл. – Москва: Мир, 1979. – 214 с.
3. Андреева, Е.А. Вариационное исчисление и методы оптимизации: учеб. пособие / Е.А. Андреева, В.М. Цирулёва. – Тверь: Твер. гос. ун-т, 2001. – 576 с.
4. Справочник по очистке природных и сточных вод / Л.Л. Пааль [и др.] ; отв. ред. Л.Л. Пааль. – Москва: Наука, 1994. – 333 с.
5. Матвеев, Н.М. Сборник задач и упражнений по обыкновенным дифференциальным уравнениям / Н.М. Матвеев. – Ленинград: Ленинградский университет, 1960. – 286 с.

УДК 004.65

Сравнительный анализ производительности различных СУБД на мобильных устройствах

А. В. Петрова

Воронежский государственный университет

Введение

В сфере разработки мобильных приложений скорость работы приложения играет ключевую роль, непосредственно влияя на его конкурентоспособность. Одним из основных факторов, определяющих скорость работы мобильного приложения, является производительность используемой приложением СУБД, на которую в свою очередь влияют объем хранимых в СУБД данных, качество оптимизации запросов и производительность оборудования.

Для оценки скорости работы мобильных приложений будут использованы такие метрики, как время отклика и пропускная способность, отражающие время выполнения запроса от момента его создания до получения результата и объем данных, который система способна обрабатывать эффективно и без сбоев. В качестве инструментов для работы с данными в исследовании выбраны SolarWinds и SQL Sentry. У этих программ удобный интерфейс, подходящий для решения поставленных задач, и глубокие методики анализа времени отклика системы.

Далее рассмотрены методики проведения тестов, с помощью которых оценивается производительность СУБД, и сравнительного анализа полученных результатов.

1. Проведение тестов

1.1. Описание серии тестов

Опишем выбранный алгоритм проведения серии тестов для каждой из сравниваемых СУБД. Тесты проводятся по описанной далее методике. Сначала выбирается тип объекта. Всего используется пять различных типов объектов для их проверки в различных условиях.

1. Song – простой объект (типы данных полей – строки, числа, логические значения);

2. Note – более сложный объект, который содержит также поля типа даты и перечислимого типа. Большинство СУБД требуются преобразования таких объектов для хранения информации;

3. Video – объект, который содержит другие объекты;

4. Chat + ChatMember – связь один-ко-многим;

5. Film + Actor – связь многие-ко-многим.

Для каждой СУБД с объектами конкретного типа выполняются тесты для определения характеристик скорости выполнения команды. Данные тесты производятся несколько раз с

одним и тем же типом объектов, после чего находится среднее значение времени выполнения операции.

Набор тестов, по которым проводится анализ, имеет следующий состав:

- Создание объекта;
- Обновление объектов;
- Удаление объектов;
- Считывание всех объектов;
- Считывание по id;
- Считывание по фильтрам.

1.2. Масштабирование тестов

Тесты проводятся с постоянно увеличивающимся количеством сущностей – сначала используются 10 сущностей, затем 100, затем 1000 и так далее. Данная мера необходима для выявления зависимости скорости выполнения операции от количества сущностей, а также установления характера этой зависимости.

Первоначально тесты проводятся с минимальным количеством сущностей, чтобы определить базовую скорость выполнения заданных операций. Затем количество сущностей увеличивается, чтобы определить зависимость скорости выполнения запросов от нагрузки на СУБД. Данная мера позволяет выявить зависимость между количеством сущностей и временем выполнения операции. Тесты проводятся для нескольких записей одновременно для вычисления среднего времени выполнения операции с установленным количеством сущностей.

Анализ результатов тестов позволяет определить, какова зависимость скорости работы от количества сущностей – линейная, экспоненциальная или какая-то другая. Такой подход позволяет оптимизировать производительность приложения, выявляя «узкие места» и проблемные участки.

2. Анализ результатов

Проводилось тестирование следующих СУБД для мобильной платформы IOS: Core Data, Object Box, Realm, YapDatabase. В результате проведения вышеописанных тестов были получены данные, отображенные на графиках. По горизонтали отображено количество проведения тестов, по вертикали – их среднее время выполнения. Амплитуда линий графиков обусловлена влиянием изменения типа объекта, над которым проводится тестирование. Ниже представлены результаты (рис. 1-6).

При создании объектов создается новая запись в таблице данных. Такая запись содержит информацию, связанную с создаваемым объектом, его атрибуты или характеристики.

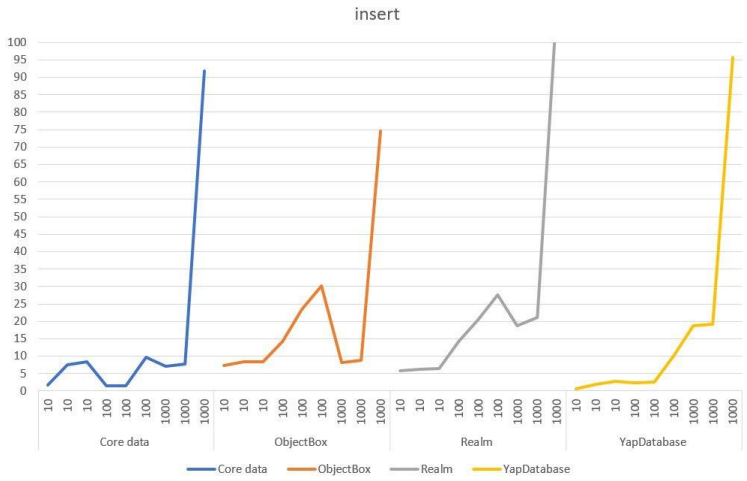


Рис. 1. Результаты теста создания объектов

В процессе обновления объектов в базе данных система определяет соответствующие записи или объекты, подлежащие изменениям, и осуществляет эти изменения. Данный механизм включает в себя идентификацию целевых записей для обновления в таблице базы данных и применение изменений.

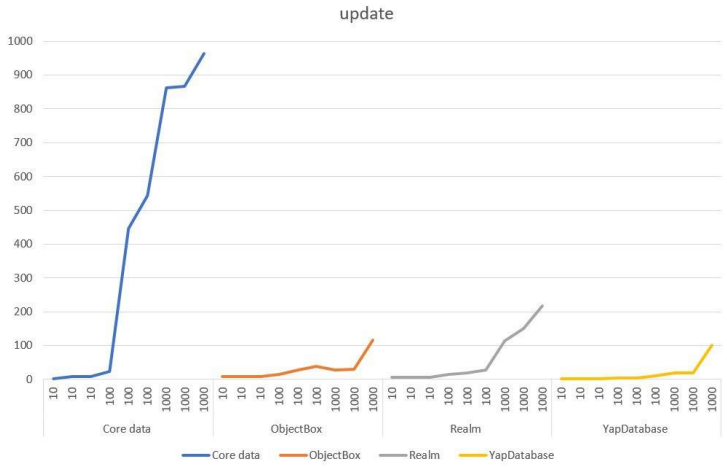


Рис. 2. Результаты теста обновления объектов

В процессе удаления объектов из базы данных выбранные записи или объекты убираются из соответствующей таблицы. Данный процесс состоит из идентификации целевых записей и их последующего удаления.

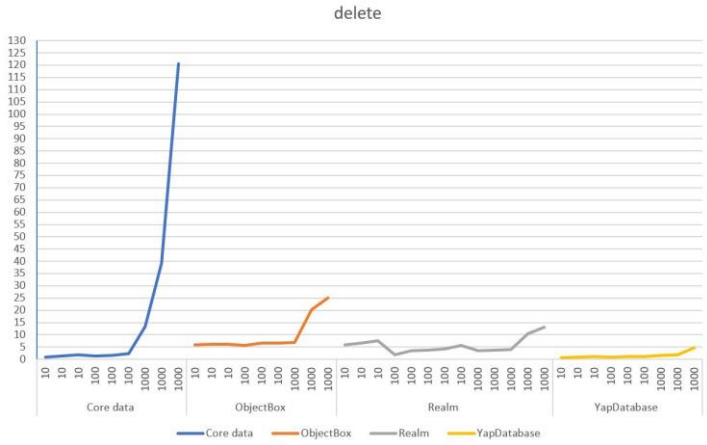


Рис. 3. Результаты теста удаления объектов

При считывании объектов происходит извлечение данных из таблицы или нескольких таблиц базы данных. В работе используется три вида считывания – считывание всех данных (selectAll), считывание по id (selectById) и считывание по фильтрам (selectWithFilters).

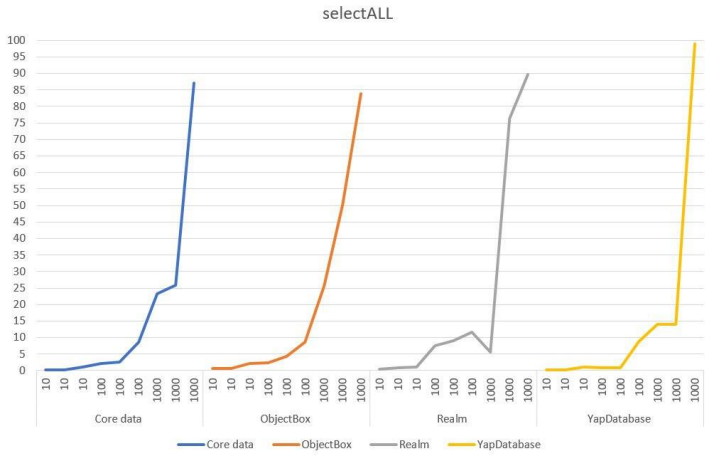


Рис. 4. Результаты теста считывания всех объектов

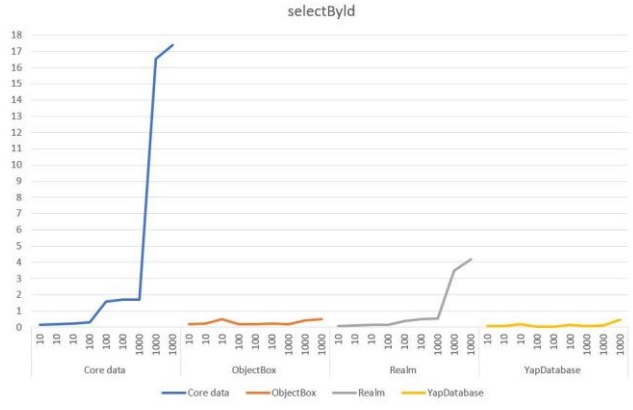


Рис. 5. Результаты теста считывания объектов по id

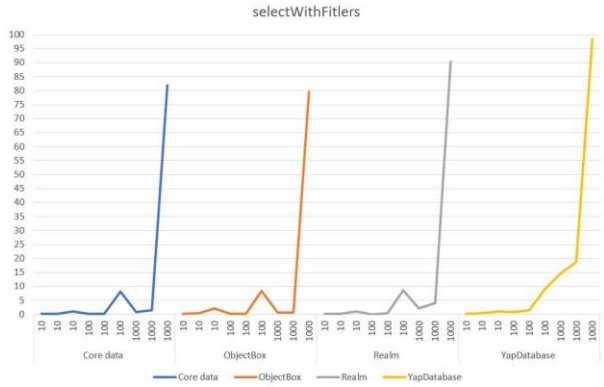


Рис. 6. Результаты теста считывания объектов по фильтрам

Заключение

В данной статье рассмотрены методики проведения тестов, с помощью которых была произведена сравнительная оценка производительности мобильных СУБД. На основании выполненного анализа можно сделать вывод о том, что Realm выполняет поставленные задачи сравнительно медленнее остальных систем, так как у неё максимальные показатели времени выполнения тестов, далее по скорости работы – зависимости времени среднего выполнения тестов от количества сущностей – следует СУБД Core data, за ней – YapDatabase, наиболее производительной среди исследуемых систем является ObjectBox.

Литература

1. Упом, М. SQL для анализа данных / М. Упом, М. Гольдвассер, Б. Джонстон. — Бирмингем, Великобритания : Packt Publishing, 2019.
2. Линов, Г. Анализ данных с использованием SQL и Excel, 2-е издание / Г. Линов. — Хобокен : Wiley, 2015.
3. Винанд, М. SQL Производительность объяснена/ М. Винанд. — Оттакринг : Markus Winand, 2012.
4. Танимура, С. SQL для анализа данных: Продвинутые техники преобразования данных в инсайты / С. Танимура. — Себастополь : O'Reilly Media, 2021.
5. Фейли, К. Программирование баз данных SQL (Пятая редакция) / К. Фейли. — Кармел, Калифорния : Questing Vole Press, 2020.

УДК 57.087.01+004.04

Недостатки использования биометрических данных при безналичной оплате

Э. В. Пешкова

Воронежский государственный университет

Введение

В современном мире безналичная оплата становится все более популярной, а вместе с ней и методы аутентификации пользователей посредством биометрических данных. Биометрические данные представляют собой уникальные биологические характеристики каждого человека, такие как отпечатки пальцев, распознавание лица, голосовые особенности и другие. Эти данные могут быть использованы для идентификации и аутентификации пользователей без необходимости запоминания паролей или использования физических объектов, таких как карты или ключи.

Одним из основных преимуществ использования биометрических данных в безналичной оплате является высокий уровень удобства для пользователей. Вместо того чтобы запоминать сложные пароли или нести с собой физические объекты, пользователи могут просто использовать свои уникальные биометрические данные для проведения транзакций. Это особенно удобно в случае мобильных платежных систем, где достаточно просто приложить палец или лицо к устройству для подтверждения оплаты.

Кроме того, использование биометрических данных может повысить уровень безопасности безналичных платежей. Пароли и PIN-коды могут быть украдены или подобраны, в то время как биометрические данные уникальны для каждого человека и труднее подделать. Это делает биометрическую аутентификацию более надежной и защищенной от мошенничества.

Однако, несмотря на все свои преимущества, использование биометрических данных в безналичной оплате также сопряжено с рядом недостатков и рисков, которые необходимо учитывать. Об этих недостатках и рисках, а также об актуальности проблемы будет подробно рассказано в следующих разделах статьи.

1. Обзор методов сбора и использования биометрических данных

1.1. Описание различных типов биометрических данных

Биометрические данные представляют собой разнообразные биологические параметры, которые могут быть использованы для идентификации и аутентификации личности. Среди наиболее распространённых видов биометрических данных можно выделить следующие:

Отпечатки пальцев: Это один из самых распространенных типов биометрических данных. Уникальные узоры и характеристики на поверхности пальцев используются для идентификации человека. Системы сканирования отпечатков пальцев используются в различных устройствах, от мобильных телефонов до банкоматов.

Распознавание лица: Этот тип биометрических данных использует уникальные черты лица человека, такие как форма глаз, носа и рта, для идентификации. Системы распознавания лица могут быть использованы в системах видеонаблюдения, а также в смартфонах и компьютерах для разблокировки устройств.

Голосовые характеристики: Биометрические данные голоса используются для аутентификации пользователя на основе уникальных особенностей и характеристик их голоса, таких как тон, интонация и скорость речи. Такие данные могут быть использованы в системах голосового управления или аутентификации при телефонных звонках.

Сканирование радужки глаза: Этот метод биометрической аутентификации использует уникальные особенности радужки глаза для идентификации человека. Системы сканирования радужки глаза могут быть более точными и надежными, чем другие методы, но требуют специализированных оборудований для сбора данных.

Геометрия ладони: Этот тип биометрических данных использует уникальные особенности геометрии ладони человека для идентификации. Это может включать в себя длину и ширину пальцев, расстояния между ними и другие характеристики. Системы сканирования ладони могут использоваться в устройствах доступа и контроля доступа.

Электрокардиограмма (ЭКГ): Уникальные параметры сердечного ритма и электрической активности сердца, которые могут быть использованы для идентификации личности.

Это только некоторые из основных типов биометрических данных, и существуют и другие методы, такие как сканирование сетчатки глаза и многие другие. Каждый из них имеет свои уникальные особенности и применения, и выбор конкретного метода зависит от требований конкретной системы аутентификации или идентификации.

1.2. Технологии сбора и обработки биометрических данных

С развитием технологий биометрической аутентификации становится все более важным понимание основных компонентов и процессов, которые обеспечивают функционирование таких систем. Сенсоры и сканеры, алгоритмы обработки, шаблоны и хранилища данных, устройства аутентификации, а также шифрование и защита данных - все они играют ключевую роль в обеспечении безопасности и эффективности биометрических систем.

Сенсоры и сканеры: Это устройства, способные считывать и регистрировать биометрические данные. Например, сенсоры отпечатков пальцев могут сканировать поверхность пальца для получения его уникальных характеристик. Сенсоры для распознавания лица могут использовать камеры и специальные алгоритмы для захвата и анализа изображений лица.

Алгоритмы обработки: Это программные инструменты, используемые для анализа и обработки собранных биометрических данных. Алгоритмы могут включать в себя методы сопоставления шаблонов, машинное обучение и нейронные сети для сравнения биометрических данных с ранее сохраненными шаблонами и определения их соответствия.

Шаблоны и хранилища данных: Полученные биометрические данные могут быть преобразованы в шаблоны или уникальные коды, которые могут быть хранены в базе данных для дальнейшего сравнения. Защита и безопасность таких хранилищ данных играют ключевую роль в предотвращении несанкционированного доступа и утечек информации.

Устройства аутентификации: Это устройства, используемые для сопоставления биометрических данных пользователя с ранее сохраненными шаблонами для подтверждения его личности. Например, мобильные устройства с датчиками отпечатков пальцев или камерами для распознавания лица могут использоваться для аутентификации пользователей перед совершением транзакций или получением доступа к данным.

Шифрование и защита данных: Поскольку биометрические данные являются чувствительной информацией, важно обеспечить их безопасность во время передачи и хранения. Технологии шифрования и методы защиты данных помогают предотвратить несанкционированный доступ и использование биометрических данных.

2. Недостатки и риски использования биометрических данных

2.1. Конфиденциальность и безопасность данных

Одним из ключевых аспектов использования биометрических данных является обеспечение их конфиденциальности и безопасности. Поскольку биометрические данные являются уникальными идентификаторами каждого человека, их утечка или несанкционированное использование может иметь серьезные последствия для личной жизни и безопасности пользователей.

Защита конфиденциальности биометрических данных включает в себя ряд мероприятий, направленных на предотвращение несанкционированного доступа к этим данным и их использования без согласия пользователя. Это включает в себя принципы минимальной необходимости, при которых данные собираются и хранятся только в необходимом объеме для выполнения определенных задач. Кроме того, используются технологии шифрования для защиты передачи и хранения биометрических данных, а также механизмы аутентификации и авторизации для контроля доступа к ним.

В свою очередь, обеспечение безопасности биометрических данных включает в себя защиту от несанкционированного доступа и использования, а также предотвращение возможных атак на системы, содержащие эти данные. Это может включать в себя использование физических и логических мер безопасности, таких как биометрическая аутентификация для доступа к данным, мониторинг защищенных зон и систем обнаружения вторжений для раннего обнаружения возможных угроз.

Важно также учитывать вопросы прозрачности и согласия пользователей относительно сбора и использования их биометрических данных. Пользователи должны быть осведомлены о том, как именно и для каких целей будут использоваться их данные, и должны иметь возможность дать согласие на их сбор и обработку.

Обеспечение конфиденциальности и безопасности биометрических данных является ключевым аспектом разработки и внедрения систем и приложений, использующих такие данные. Это позволяет не только защитить личные данные пользователей, но и поддерживать их доверие к системам и сервисам, которые они используют.

2.2. Возможности кражи личности и мошенничества

Хотя биометрические данные представляют собой уникальные характеристики каждого человека, они также могут стать объектом мошенничества и кражи личности. Возможности мошенников и хакеров постоянно усиливаются, и важно осознавать риски, связанные с использованием биометрических данных.

В случае утечки биометрических данных, таких как отпечатки пальцев или распознавание лица, злоумышленники могут использовать эти данные для подделки или взлома систем аутентификации. Например, скомпрометированные отпечатки пальцев могут быть использованы для взлома систем доступа или мобильных устройств.

Кроме того, биометрические данные могут быть подвержены мошенничеству, когда злоумышленники используют украденные или поддельные данные для аутентификации в системах или совершения мошеннических действий. Например, мошенники могут попытаться подделать отпечаток пальца или создать маску для обхода систем распознавания лица.

Для защиты от таких угроз необходимо принимать меры предосторожности и реализовывать технические и организационные меры безопасности. Это может включать в себя использование многоуровневых систем аутентификации, биометрических алгоритмов с высоким уровнем безопасности, регулярное обновление программного обеспечения для

предотвращения уязвимостей, а также обучение пользователей о методах предотвращения мошенничества и защите их личных данных.

Осведомленность о рисках и активное принятие мер по их минимизации помогут уменьшить вероятность возникновения кражи личности и мошенничества в контексте использования биометрических данных.

2.3. Этические и правовые аспекты

Использование биометрических данных вызывает важные этические и правовые вопросы, связанные с приватностью, индивидуальными правами и безопасностью. Важно соблюдать законодательство и этические стандарты для обеспечения справедливого и этичного использования биометрических данных.

Один из основных этических вопросов связан с правом на приватность и конфиденциальность персональных данных, включая биометрические данные. Люди имеют право на контроль над собственными данными и их использованием, и важно учитывать этот аспект при сборе, хранении и обработке биометрических данных.

При использовании биометрических данных важно обеспечить справедливость и равноправие для всех пользователей. Нельзя допустить дискриминацию или неравное обращение на основе биометрических характеристик, и необходимо применять эти данные с учетом принципов равенства и беспристрастности.

Важно обеспечить согласие и информированность пользователей относительно сбора и использования их биометрических данных. Пользователи должны быть осведомлены о том, как именно и для каких целей будут использоваться их данные, и должны иметь возможность дать свое согласие или отказаться от него.

Существует ряд законов и нормативных актов, регулирующих использование биометрических данных, таких как GDPR в Европейском союзе и другие местные законы о защите данных. Организации и компании, собирающие и обрабатывающие биометрические данные, должны соблюдать эти законы и нормы, чтобы обеспечить права и защиту пользователей.

Обеспечение этического и законного использования биометрических данных является важным аспектом разработки и внедрения систем, использующих такие данные. Учитывая их чувствительность и потенциальные риски, необходимо строго соблюдать принципы этики и законодательства для защиты прав и интересов пользователей.

3. Альтернативные подходы и решения

3.1. Использование множественных аутентификационных методов

Множественная аутентификация - это процесс использования двух или более методов для подтверждения личности пользователя. Этот подход повышает уровень безопасности и защиты данных, так как он усложняет задачу злоумышленникам в случае попытки несанкционированного доступа.

Существует несколько типов множественной аутентификации, включая:

Что-то, что вы знаете: Этот метод основан на знании уникальной информации, такой как пароль, пин-код или ответ на секретный вопрос.

Что-то, что у вас есть: Этот метод включает использование физического объекта в качестве подтверждения личности, например, магнитной карточки, USB-ключа или мобильного устройства.

Что-то, что вы являетесь: Этот метод использует биометрические данные, такие как отпечаток пальца, сканирование лица или голосовая аутентификация.

Комбинируя эти методы, организации могут создать более надежные системы аутентификации, которые сложнее поддаются взлому или несанкционированному доступу. Например, множественная аутентификация может включать запрос пароля в сочетании с отправкой одноразового кода на заранее зарегистрированный мобильный телефон пользователя.

Этот подход к аутентификации широко используется во многих сферах, включая финансовые учреждения, корпоративные сети, онлайн-сервисы и мобильные приложения. Он играет ключевую роль в обеспечении безопасности и защиты данных от несанкционированного доступа и мошенничества.

3.2. Развитие технологий шифрования и защиты

С развитием информационных технологий и увеличением объема цифровых данных возрастает и необходимость в более надежных методах защиты данных. Технологии шифрования играют ключевую роль в обеспечении конфиденциальности и целостности информации, а также в защите от несанкционированного доступа.

Среди основных тенденций в развитии технологий шифрования и защиты данных можно выделить следующие:

Усиление методов шифрования: С развитием компьютерных мощностей и квантовых вычислений старые методы шифрования могут становиться уязвимыми, поэтому разрабатываются более совершенные алгоритмы и протоколы шифрования для защиты данных.

Развитие квантового шифрования: Квантовое шифрование представляет собой новую область в области криптографии, которая основана на принципах квантовой механики. Оно обеспечивает более высокий уровень безопасности и защиты данных за счет использования квантовых свойств частиц.

Использование многофакторной аутентификации: Для усиления защиты данных широко применяется многофакторная аутентификация, которая комбинирует несколько методов аутентификации, таких как пароль, биометрические данные и физические токены.

Развитие методов обнаружения угроз: Вместе с усилением методов защиты данных разрабатываются и совершенствуются методы обнаружения и предотвращения угроз. Это включает в себя использование машинного обучения и искусственного интеллекта для анализа поведения пользователей и обнаружения аномалий в сети.

Фокус на защите конечных точек: С увеличением числа кибератак, направленных на конечные точки, такие как компьютеры, мобильные устройства и интернет-подключенные устройства, все больше внимания уделяется защите этих устройств и обеспечению безопасности данных на них.

Эти тенденции в развитии технологий шифрования и защиты данных позволяют организациям и пользователям обеспечивать более высокий уровень безопасности и защиты данных в цифровой среде.

3.3. Роль регулирования и законодательства в области безопасности биометрических данных

Регулирование и законодательство играют ключевую роль в обеспечении безопасности и защиты биометрических данных, устанавливая стандарты и требования к их сбору, хранению и использованию. Законы и нормативные акты в области защиты данных направлены на защиту приватности и прав пользователей, а также на предотвращение злоупотреблений и утечек информации.

Среди основных аспектов регулирования и законодательства в области безопасности биометрических данных можно выделить следующие:

Законы о защите данных, такие как GDPR в Европейском союзе и HIPAA в Соединенных Штатах, устанавливают правила и требования по сбору, хранению и обработке биометрических данных. Они защищают права пользователей на контроль над своими данными и требуют согласия на их использование.

Законы и нормативные акты также устанавливают требования к безопасности и защите биометрических данных, включая меры шифрования, управления доступом и предотвращения утечек данных. Они обязывают организации и компании соблюдать определенные стандарты и меры безопасности для защиты данных.

Законы о защите данных предусматривают ответственность и наказание за нарушения правил и требований по обработке и защите биометрических данных. Организации могут быть подвергнуты штрафам и санкциям за несоблюдение законодательства в этой области.

Законы о защите данных также требуют от организаций обеспечивать прозрачность и отчетность в отношении сбора, использования и обработки биометрических данных. Это включает предоставление информации пользователям о целях использования их данных и обеспечение доступа к этой информации по запросу.

Регулирование и законодательство в области безопасности биометрических данных играют важную роль в обеспечении приватности и защиты данных пользователей. Они устанавливают стандарты и требования, которые должны соблюдаться организациями и компаниями для защиты личной информации и предотвращения злоупотреблений с биометрическими данными.

Заключение

В данный момент мы рассмотрели ключевые аспекты использования биометрических данных в современных системах безналичной оплаты. Биометрические технологии представляют собой мощный инструмент для обеспечения безопасности и удобства пользователей, однако они также сопряжены с рядом вызовов и рисков.

Мы обсудили различные типы биометрических данных, сенсоры и сканеры, алгоритмы обработки, шаблоны и хранилища данных, устройства аутентификации, а также методы шифрования и защиты данных, которые составляют основу систем биометрической аутентификации.

Важно понимать, что хотя биометрические технологии предлагают значительные преимущества, они также вносят определенные риски, связанные с конфиденциальностью и безопасностью данных. Поэтому необходимо тщательно обдумывать и реализовывать меры по защите и безопасности при использовании биометрических систем.

В дальнейшем исследовании и разработке необходимо уделить особое внимание улучшению технологий аутентификации, разработке более надежных и эффективных методов защиты данных, а также анализу этических и правовых аспектов использования биометрических технологий.

Следуя принципам прозрачности, безопасности и эффективности, мы можем создать современные системы безналичной оплаты, которые обеспечат высокий уровень защиты данных и удовлетворят потребности пользователей в безопасных и удобных методах оплаты.

Литература

1. Идентификация личности с использованием биометрии / Е.С. Пимкина, П.А.

Тушевский / Безопасность информационных технологий. 2006. №1. С. 36-38
2. Биометрия как способ идентификации в банковском секторе / Черных Я.В., Зубарев Р.О. / В сборнике: ПРОБЛЕМЫ РАЗВИТИЯ СОВРЕМЕННОГО ОБЩЕСТВА. сборник научных статей 5-й Всероссийской научно-практической конференции. Юго-Западный государственный университет. 2020. С. 282-284.
3. О применении биометрии для идентификации клиентов банка / Левашов М.В. Бухгалтерия и банки. 2020. № 3. С. 52-53.
4. Методы биометрии при обеспечении информационной безопасности / Корнев Л.В. Молодой ученый. 2022. № 17 (412). С. 358-361.

УДК 519.622

Построение метода Рунге–Кутты–Чебышева

М. А. Писарцов

Воронежский государственный университет

Введение

Методы Рунге–Кутты [2, 4, 5, 6] — наиболее часто используемые методы численного решения начальной задачи для обыкновенных дифференциальных уравнений. Наиболее известен явный 4-х стадийный метод Рунге.

Каждый метод Рунге–Кутты (РК) строится на основе некоторой квадратурной формулы. Например, явный 4-х стадийный метод Рунге построен на основе квадратурной формулы Симпсона.

Настоящая работа посвящена построению методов Рунге–Кутты на основе квадратурной формулы П. Л. Чебышёва. Ее особенность — простой вид коэффициентов b_j. Метод Рунге–Кутты–Чебышёва (РКЧ) может иметь любое число стадий и, как следствие, любой порядок аппроксимации; иными словами, можно построить метод Рунге–Кутты–Чебышёва, обеспечивающий любую наперед заданную точность.

Интересная особенность методов Рунге–Кутты–Чебышёва — в ходе вычислений они могут выходить в комплексную область; этот случай и рассматривается в данной работе.

1. Определение метода Рунге–Кутты

Рассмотрим начальную задачу

$$\begin{aligned} y'(x) &= f(x,y), \\ y(x_0) &= y_0, \end{aligned}$$

где f дифференцируема столько раз, сколько потребуется.

Пусть s — натуральное число. Пусть заданы числа c_i, b_j и a_{ij}, где $i,j = 1,\ldots,s$. *(Неявным)* s*-стадийным методом Рунге-Кутты* (сокращённо РК) называют [2, 4, 5, 6] следующее правило вычисления приближения y_1 к точному значению $y(x_0 + h)$ решения y начальной задачи в точке $x_0 + h$:

$$k_i = f\left(x_0 + hc_i, y_0 + h\sum_{j=1}^{s} a_{ij}k_j\right),\ i = 1,\ldots,s, \#(1)$$

$$y_1 = y_0 + h\sum_{j=1}^{s} b_j k_j. \qquad \# \quad (2)$$

В общем случае набор равенств (1) представляет собой систему уравнений относительно k_i. Обычно эта система может быть решена при малых h методом последовательных приближений, поскольку в правой части формулы (1) перед k_j стоит малый множитель h.

Если $a_{ij} = 0$ при $i \le j$, метод называют *явным*. В этом случае числа k_i можно найти из уравнении (1) последовательно (без решения уравнений). Но при этом возникает узкая область устойчивости. Если $a_{ij} = 0$ только при $i < j$, то метод называют *диагональным неявным*. В этом случае числа k_i можно снова находить последовательно, решая при этом нелинейные уравнения, содержащие только одну неизвестную. Нас интересуют *неявные* методы, когда все или почти все числа a_{ij} могут оказаться ненулевыми.

Параметры c_i, b_j и a_{ij} метода РК принято располагать в виде табл.1, называемой *таблицей Бутчера*.

Таблица 1

Таблица Бутчера метода РК

c_1	a_{11}	a_{12}	...	$a_{1,s-1}$	$a_{1,s}$
c_2	a_{21}	a_{22}	...	$a_{2,s-1}$	$a_{2,s}$
...	...	...	...	...	...
c_s	$a_{s,1}$	$a_{s,2}$	...	$a_{s,s-1}$	$a_{s,s}$
	b_1	b_2	...	b_{s-1}	b_s

Очевидно, таблица Бутчера содержит в себе всю информацию о методе Рунге–Кутты.

Одним из основных критериев качества метода РК считают его порядок аппроксимации. Говорят, что метод РК имеет *порядок аппроксимации* m, если для любых достаточно гладких дифференциальных уравнений имеем

$$y_1 - y(x_0 + h) = O(h^{m+1}) \qquad \text{при } h \to 0,$$

где $y(x_0 + h)$ — значение точного решения начальной задачи. Эквивалентное требование — ряды Тейлора по h приближенного y_1 и точного y решений совпадают до членов порядка h^m.

Найти порядок аппроксимации конкретного метода в принципе несложно, но это требует громоздких вычислений. Надо разложить по формуле Тейлора выражение y_1, то есть результат подстановки (1) в (2) через h, и сравнить полученное с разложением по формуле Тейлора точного решения начальной задачи в точке x_0. Число совпавших членов минус один и есть порядок аппроксимации метода.

Как построить метод РК нужного порядка аппроксимации? Очевидно, надо подобрать параметры так, чтобы соответствующее число слагаемых рядов Тейлора совпало.

Известно [2, 4, 5], что уравнения порядка аппроксимации не являются независимыми, и во многих случаях коэффициенты a_{ij} можно найти, решая только линейные относительно a_{ij} уравнения.

Эти линейные относительно a_{ij} уравнения можно описать следующим образом: для того, чтобы метод РК имел порядок аппроксимации m, достаточно [6], [2, теорема 24], чтобы выполнялись условия:

$$\sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q}, \qquad q = 1,\dots,m, \qquad (B(m))$$

$$\sum_{i=1}^{s} a_{ij} c_j^{q-1} = \frac{c_i^q}{q}, \qquad i = 1,\dots,s;\ q = 1,\dots,\eta, \qquad (C(\eta))$$

$$\sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} = \frac{b_j}{q}\left(1 - c_j^q\right), \quad j = 1,\dots,s;\ q = 1,\dots,\zeta, \qquad (D(\zeta))$$

для некоторых целых η и ζ, удовлетворяющих неравенствам $m \le \eta + \zeta + 1$ и $m \le 2\eta + 2$.

2. Квадратурная формула Чебышёва

Квадратурная формула Чебышёва — это квадратурная формула численного интегрирования, которую изобрёл Пафнутий Львович Чебышёв. Особенность квадратурной формулы Чебышёва заключается в том, что весовые коэффициенты b_j берутся равными друг другу (и тем самым легко вычисляются с любой точностью), а узлы подбираются так, чтобы обеспечить максимально возможный порядок аппроксимации.

Эта формула позволяет получить приближённое значение интеграла с помощью минимального количества вычислений, что делает её эффективной для численных расчётов, особенно в тех случаях, когда требуется небольшой объём вычислений.

Квадратурная формула Чебышёва на отрезке $[-1,1]$ имеет вид

$$\int_{-1}^{1} f(x)dx \approx \frac{2}{n}\sum_{k=1}^{n} f(c_k), \#(3)$$

где c_k — узлы.

Опишем алгоритм нахождения узлов c_k, приведённый в книге [1, с. 375, п. 2]. Пусть $\mathcal{P}_s(x) = x^s + a_1 x^{n-1} + \dots + a_s$ — многочлен, корнями которого являются узлы $c_1,\dots,c_s$. В книге [1, с.376] выписана система линейных уравнений, из которых находятся коэффициенты многочлена $\mathcal{P}_s$. После этого, решая уравнение $\mathcal{P}_s(x) = 0$, находим узлы $c_1,\dots,c_s$.

По условию известно, что коэффициенты многочлена $\mathcal{P}_s$ с нечётными номерами равны нулю: $a_1 = a_3 = \ldots = a_{2s+1} = 0$. Таким образом, узлы c_j симметричны относительно точки 0, и при нечётном n один из корней многочлена $\mathcal{P}_s(x)$ (узлов c_k) равен нулю.

Также П. Л. Чебышёв установил, что при $n = 1,2,\ldots,7,9$ узлы c_j квадратурной формулы — вещественные. А при $n = 8$ узлы c_j квадратурной формулы комплексные. Чуть позже С. Н. Бернштейн доказал теорему [1, теорема 2, с. 377], в которой говорится, что при $n \geq 10$ среди узлов c_j квадратурной формулы обязательно имеются комплексные.

3. Построение метода Рунге–Кутты–Чебышёва

Опишем один из возможных алгоритмов построения метода Рунге–Кутты на основе квадратурной формулы Чебышёва; будем называть такие методы методами Рунге–Кутты–Чебышёва. Проведем построение на примере 8-ми стадийного ($s = 8$) метода Рунге–Кутты–Чебышёва, что приводит к комплексным значениям узлов c_j.

Порождающий многочлен квадратурной формулы Чебышёва имеет вид

$$\mathcal{P}_8(x) = -\frac{43}{42525} - \frac{148x^2}{2835} + \frac{22x^4}{45} - \frac{4x^6}{3} + x^8.$$

Узлы c_i находим, решая уравнение $\mathcal{P}_8(x) = 0$:

$$\begin{aligned}
c_1 &= 0.5 - 0.064523i,\\
c_2 &= 0.5 + 0.064523i,\\
c_3 &= 0.049253,\\
c_4 &= 0.950747,\\
c_5 &= 0.239715 + 0.0242212i,\\
c_6 &= 0.760285 - 0.0242212i,\\
c_7 &= 0.239715 - 0.0242212i,\\
c_8 &= 0.760285 + 0.0242212i.
\end{aligned}$$

Как видно, предположения Чебышева оказались верны: получены комплексные корни. В силу основной идеи квадратурной формулы Чебышева все b_j равны $\frac{1}{s} = \frac{1}{8}$.

Осталось найти матрицу A из коэффициентов a_{ij}, которая в случае $s = 8$ имеет размер 8×8. Так как имеем 16 неизвестных a_{ij}, то и уравнений должно быть столько же. Для составления уравнений используем системы равенств $(C(\eta))$ и $(D(\zeta))$ с конкретными $\eta,\zeta = 0,1,2,\ldots$, удовлетворяющими неравенствам $m \leq \eta + \zeta + 1$ и $m \leq 2\eta + 2$, где m — порядок аппроксимации (напомним, что эти неравенства обеспечивают максимальный возможный порядок аппроксимации метода). После перебора всех возможных случаев выяснилось, что наиболее удобными являются значения $\eta = 8$ и $\zeta = 0$. Воспользуемся ими и выпишем соответствующую таблицу Бутчера (табл.2):

Таблица 2

Таблица Бутчера метода РКЧ

$0.5 - 0.064523i$	$0.0625 - 0.114575i$	$0.0625 + 0.055595i$	…
$0.5 + 0.064523i$	$0.0625 - 0.055595i$	$0.0625 + 0.114575i$	…
0.049253	$-0.0146691 + 0.110314i$	$-0.0146691 - 0.110314i$	…
0.950747	$0.139669 + 0.110314i$	$0.139669 - 0.110314i$	…
$0.239715 + 0.0242212i$	$0.00423593 + 0.00367685i$	$0.00294102 - 0.00335585i$	…
$0.760285 - 0.0242212i$	$0.122059 + 0.00335585i$	$0.120764 - 0.00367685i$	…
$0.239715 - 0.0242212i$	$0.00294102 + 0.00335585i$	$0.00423593 - 0.00367685i$	…
$0.760285 + 0.0242212i$	$0.120764 + 0.00367685i$	$0.122059 - 0.00335585i$	…
	0.125	0.125	…

…	$0.124437 - 0.00011071i$	$0.000563062 - 0.00011071i$	$0.113073 + 0.0247398i$	
…	$0.124437 + 0.00011071i$	$0.000563062 + 0.00011071i$	$0.122844 + 0.0274007i$	
…	$0.0782477 - 5.4737 * 10^{-16}i$	$-0.00234283 + 3.18858 * 10^{-16}i$	$0.0175569 - 0.258688i$	
…	$0.127343 - 4.7804 * 10^{-15}i$	$0.0467523 + 2.75332 * 10^{-15}i$	$0.141214 - 0.0769025i$	
…	$0.122821 + 0.0000374515i$	$-0.000319928 + 0.0000101192i$	$0.0550213 + 0.149856i$	
…	$0.12532 - 0.0000101192i$	$0.00217862 - 0.0000374515i$	$0.12603 - 0.0084778i$	.
…	$0.122821 - 0.0000374515i$	$-0.000319928 - 0.0000101192i$	$0.0577957 + 0.126166i$	
…	$0.12532 + 0.0000101192i$	$0.00217862 + 0.0000374515i$	$0.126751 - 0.00831532i$	
…	0.125	0.125	0.125	…

…	$0.00215603 - 0.0274007i$	$0.122844 - 0.0274007i$	$0.011927 + 0.0247398i$
…	$0.011927 - 0.0247398i$	$0.113073 - 0.0247398i$	$0.00215603 + 0.0274007i$
…	$-0.0162137 + 0.0769025i$	$0.0175569 + 0.258688i$	$-0.0162137 - 0.0769025i$
…	$0.107443 + 0.258688i$	$0.141214 + 0.0769025i$	$0.107443 - 0.258688i$
…	$-0.00103026 + 0.0084778i$	$0.0577957 - 0.126166i$	$-0.00175059 - 0.00831532i$
…	$0.0699787 - 0.149856i$	$0.126751 + 0.00831532i$	$0.0672043 + 0.126166i$
…	$-0.00175059 + 0.00831532i$	$0.0550213 - 0.149856i$	$-0.00103026 - 0.0084778i$
…	$0.0672043 - 0.126166i$	$0.12603 + 0.0084778i$	$0.0699787 + 0.149856i$
…	0.125	0.125	0.125

С целью проверки качества полученного метода РК возьмём дифференциальное уравнение на отрезке $[0,2]$

$$y'(x) = 1 + x * y(x) \#(4)$$

с начальным условием:

$$y(0) = 0.5.$$

Известно аналитическое решение этой начальной задачи:

$$y(x) = \left\{0.5e^{\frac{x^2}{2}}\left(\sqrt{2\pi}\,\mathrm{erf}\left(\frac{x}{\sqrt{2}}\right) + 1\right)\right\},$$

где $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}\,dt$. Изобразим аналитическое решение на графике (рис.1) в виде кривой, а полученные приближённые решения укажем поточечно с шагом $h = 0.2$ (т.е. применим метод РКЧ 10 раз на каждом из отрезков длины 0.2):

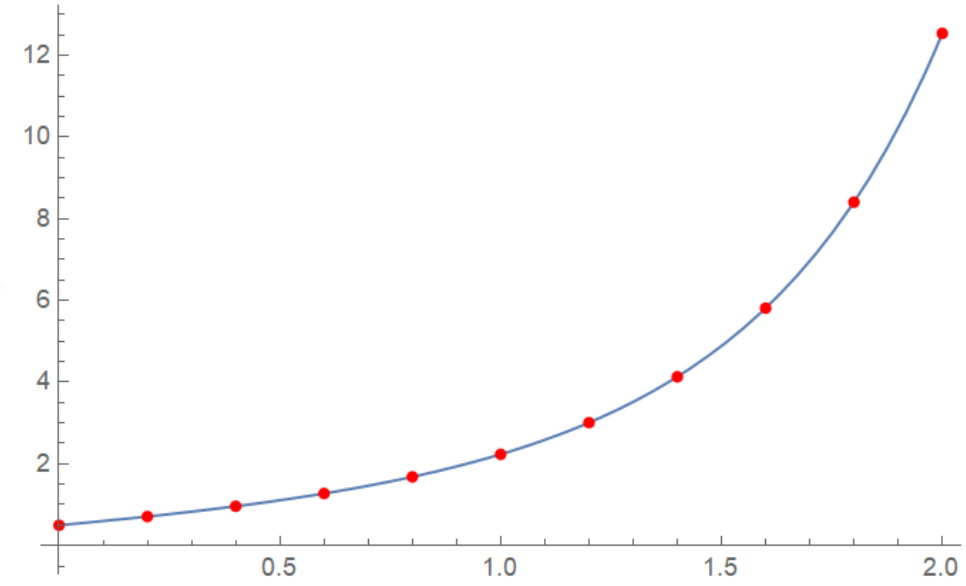


Рисунок 1. График решения

Визуально нельзя точно определить качество проделанной работы, поэтому составим сравнительную таблицу (табл.3) погрешности:

Таблица 3

Сравнительная таблица погрешности решения.

Значения в узлах	0	0.2	0.4	0.6	0.8	…
Решения:						
Аналитически	0.5	0.712789	0.963975	1.27607	1.68322	…
Метод РК	0.5	0.712789	0.963975	1.27607	1.68322	…
Погрешность	0	$2.22045 * 10^{-16}$	$4.44089 * 10^{-16}$	$1.55431 * 10^{-15}$	$2.88658 * 10^{-15}$	…

…	1.0	1.2	1.4	1.6	1.8	2.0
…	2.23505	3.00949	4.13227	5.812	8.40455	12.534

…	1.0	1.2	1.4	1.6	1.8	2.0
…	2.23505	3.00949	4.13227	5.812	8.40455	12.534
⋯	$5.32907 * 10^{-1}$	$9.32587 * 10^{-1}$	$1.24345 * 10^{-1}$	$2.66454 * 10^{-1}$	$5.50671 * 10^{-1}$	$3.01981 * 10^{-13}$

Замечание: в табл. 3 не указывалась мнимая часть приближений, полученных рассматриваемым методом Рунге–Кутты-Чебышёва, так как она не превышала 10^{-17}, что находится в пределах точности вычислений.

Все вычисления проводились в пакете «Математика» [3, 7].

Заключение

Таким образом, численный эксперимент показывает, что построенный метод Рунге–Кутты–Чебышёва даёт достаточно высокую точность приближенного решения дифференциальных уравнений при сравнительно большом шаге h.

Литература

1. Бабенко, К. И. Основы численного анализа / К. И. Бабенко. – Второе изд. – М.–Ижевск: Регулярная и хаотическая динамика, 2002. – 848 с. – ISBN: 5-93972-162-1.
2. Курбатов, В. Г. Конструирование методов Рунге–Кутты: учебное пособие / В. Г. Курбатов. – Второе изд. – Воронеж: Издательский дом ВГУ, 2024. – 98 с. – ISBN: 978-5-9273-3908-2.
3. Курбатов, В. Г. Пакет “Математика” в прикладных научных исследованиях: учебное пособие / В. Г. Курбатов, В. Е. Чернов. – Воронеж: Издательский дом ВГУ, 2016. –240 с. – ISBN: 978-5-9273-2297-8.
4. Хайрер, Э. Решение обыкновенных дифференциальных уравнений. Жесткие и дифференциально-алгебраические задачи / Э. Хайрер, Г. Ваннер. – М.: Мир, 1999. – 685 с. – ISBN: 5-03-003117-0, 9783540604525.
5. Хайрер, Э. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи / Э. Хайрер, С. Нёрсетт, Г. Ваннер. – М.: Мир, 1990. – 685 с. – ISBN: 5-03-001179-X, 9783540171454.
6. Butcher, J. C. Implicit Runga–Kutta processes / John Charles Butcher // Math. Comp. – 1964. – Vol. 18. – P. 50–64.
7. Wolfram, S. The Mathematica book / S. Wolfram. – Fifth edition. – New York: Wolfram Media, 2003. – 1488 p. – ISBN: 1-57955-022-3.

УДК 004.852

Исследование основных подходов к реализации искусственного интеллекта для игры «Нарды»

Н. В. Покатаев

Воронежский государственный университет

Введение

В последние десятилетия развитие искусственного интеллекта привнесло новые аспекты в изучение игры «Нарды». По мере того, как исследователи и энтузиасты стремятся разработать системы искусственного интеллекта, способные конкурировать с человеческим опытом, появляются различные теории и подходы в настоящей области.

1. Постановка задачи

Нарды — это игра для двух игроков, в которой каждый игрок стремится перемещать свои шашки по доске и в конечном итоге выбить их. В игре присутствуют элементы случайности из-за броска костей и стратегических решений о том, как передвигать шашки. Сложность возникает из-за множества возможных ходов и неопределенности, вносимой бросками игральных костей. Существует множество разновидностей игры «Нарды». Наиболее известными считаются короткие и длинные нарды. В рамках настоящей работы будет рассмотрена первая разновидность.

Целью данного исследования является изучение основных методологий, используемых при внедрении искусственного интеллекта в игру «Нарды». В настоящей работе будут рассмотрены классические эвристические подходы, алгоритмы по обработке и обходу деревьев решений, а также методологии последних лет, основанные на нейронных сетях.

2. Эвристические алгоритмы

Эвристический алгоритм — это последовательность действий, основанная на некоторой эвристике. Такие алгоритмы называют приближенными, так как эвристика — некоторое не относящееся к области исследования правило, позволяющее достичь определенных результатов при определённых обстоятельствах.

Алгоритмы данного типа направлены на достижение баланса между скоростью и результативностью принятия решений, предоставляют практические и прагматичные стратегии для преодоления сложностей игрового процесса в игре «Нарды». Многие подходы подробно описаны в литературе [1]. В рамках настоящей работы будут изложены лишь некоторые из них:

1. **Удаление с доски**. Эвристика направлена на эффективное удаление шашек с доски на заключительном этапе игры. В ней приоритет отдается удалению шашек с пунктов дома игрока, где количество шашек максимально, чтобы сократить количество очков, необходимых для выигрыша. Такой подход позволяет не оставлять уязвимых мест и сохранять сбалансированную позицию на доске.
2. **Позиционная эвристика**. Данная эвристика оценивает общее положение на доске для принятия решений. Она определяет приоритетность ходов, которые улучшают позицию игрока, одновременно препятствуя прогрессу противника. Учитываются

такие факторы, как распределение шашек, контроль очков и потенциал для наращивания очков.

3. **Эвристика контакта**. Эвристика направлена на усиление взаимодействия с шашками противника. Она определяет последовательность ходов, которые создают возможность поразить или заманить в ловушку шашки противника.
4. **Гоночная эвристика**. Данная эвристика нацелена на быстрое перемещение шашек и удаление их с доски, чтобы свести к минимуму количество очков и помешать продвижению противника.
5. **Эвристика «Куб удвоения».** Эвристика позволяет оценить, следует ли предлагать или принимать удвоение во время игры, основываясь на таких факторах, как позиция игрока и количество занятых пунктов.
6. **Безопасная эвристика**. Данная эвристика направлена на рассредоточение шашек и поддержание гибкой позиции, чтобы не оставлять пункты, занятые одной шашкой (блоты), уязвимыми для атаки.

3. Деревья решений

Подходы, основанные на деревьях решений, представляют набор алгоритмов и методологий, используемых для изучения и анализа обширного пространства решений, присущего игре. Эти методы используют древовидные структуры, называемые деревьями игры, для представления возможных последовательностей ходов и их результатов, что позволяет агентам искусственного интеллекта (некоторый алгоритм принятия решений, автономная система) принимать обоснованные решения на основе стратегических соображений.

На рис. 1. изображен пример узла некоторого дерева решений абстрактной игры.

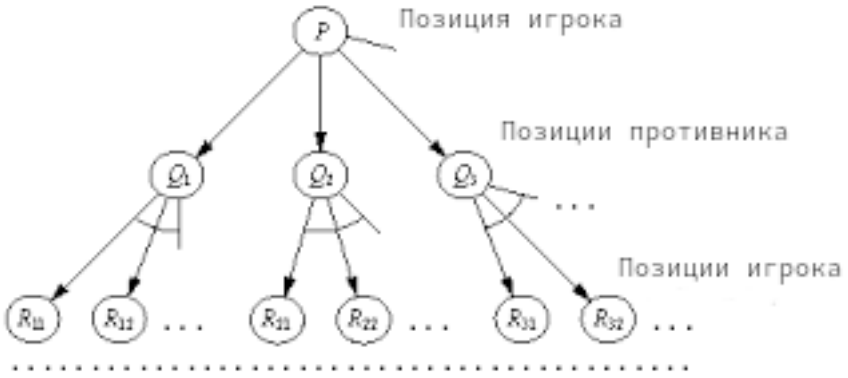


Рис 1. Схематичное представление некоторого узла дерева решений

Каждый узел в дереве представляет определенное игровое состояние, а ребра представляют возможные ходы, переводящие игру из одного состояния в другое.

В конечных игровых состояниях (узлах дерева) применяется оценочная функция для оценки желательности результата для игрока. Эта функция обычно учитывает такие факторы, как положение шашек, количество пунктов, контроль доски и другие стратегические соображения.

В рамках научного пособия [2] рассматривается широкий спектр тем, связанных с искусственным интеллектом в разнообразных играх, включая алгоритмы поиска, которые изложены ниже. В нем содержатся подробные объяснения, примеры и обсуждения этих методов в контексте абстрактных игровых агентов.

3.1. Алгоритм «минимакс» и «максимин»

Алгоритмы «Минимакс» и «Максимин» — это алгоритмы принятия решений, используемые в теории игр и искусственном интеллекте. Они предназначены для определения оптимального хода игрока в игре с нулевой суммой, где выигрыш одного игрока означает проигрыш другого.

Оценка v_i для i-го игрока может быть представлена в общем виде (1).

$$v_i = \max_{a_i} \min_{a_{-i}} v_i(a_i, a_{-i}) \quad (1)$$

где $(-i)$ представляет индекс, отличный от i-го игрока, a_i обозначает действие, предпринятое i-ым игроком, a_{-i} действие, предпринятое другим игроком соответственно.

Алгоритм «Максимин» в формульной нотации похож на «Минимакс» за исключением, что стремится максимизировать минимально возможный выигрыш для игрока, предполагая, что противник будет действовать таким образом, чтобы свести его к минимуму, в то время как «Минимакс» стремится минимизировать максимальный потенциальный проигрыш для игрока, предполагая, что противник будет действовать таким образом, чтобы максимизировать его.

3.2. Алгоритм «Альфа-бета-обрезка»

Алгоритм «Альфа-бета-обрезка» является модифицированной версией алгоритма «Минимакс». Основной целью модификации является производительность. Тогда как анализ «Минимакс» подхода требует обхода всех конечных состояний, число которых растет экспоненциально сложности игры (количеству достижимых состояний), данный алгоритм позволяет двухкратно уменьшить сложность за счет специальных параметров α и β. Эти параметры сохраняются в качестве пороговых значений, определяя направление поиска и отсекая ненужные ветви игрового дерева, тем самым значительно повышая эффективность процесса поиска. Результат работы алгоритма соответствует алгоритму «Минимакс».

3.3. Алгоритм «Поиск по дереву Монте-Карло»

Алгоритм «Поиск по дереву Монте-Карло» является одним из самых популярных алгоритмов в области поиска решений на дереве. Отличительной особенностью данного алгоритма от других является обучаемость. Такое поведение достигается за счет введения специальных параметров в дерево решений, представляющих собой количество сыгранных и выигранных партий, соответственно. Схематично шаги данного алгоритма проиллюстрированы на рис. 2.

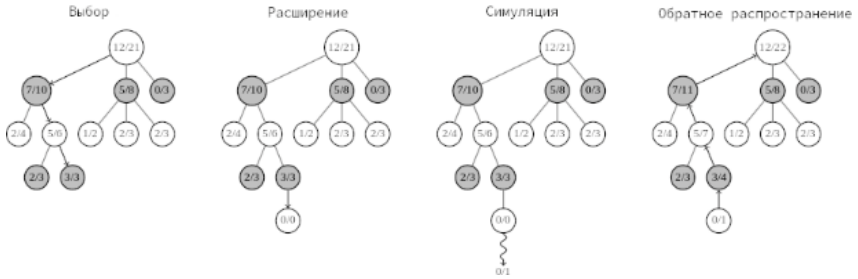


Рис. 2. Схематичное представление алгоритма «Поиск по дереву Монте-Карло»

Алгоритм состоит из 4 шагов:

1. **Выбор**. Выбирается ход соперника в случае, если ход не существует — он будет добавлен.
2. **Расширение**. К ходу противника прибавляется узел, содержащих ход агента, а также «нулевой результат».
3. **Симуляция**. Вычисляется результат партии от текущего состояния до окончания игры.
4. **Обратное распространение**. Результат симуляции распространяется от текущего узла до корневого.

4. Нейросетевые алгоритмы

Нейронные сети обладают преимуществами перед эвристическими алгоритмами в игре «Нарды» и других областях благодаря своей адаптивности, способности изучать сложные паттерны, гибкости, оптимизации с помощью обучения и устойчивости к изменчивости. Нейронные сети могут постоянно повышать свою производительность за счет обучения на основе новых данных, распознавать сложные закономерности, адаптироваться к различным сценариям решения проблем и демонстрировать устойчивость к неопределенности и шуму в данных.

4.1. TD-Gammon

Разработанный Джеральдом Тесауро в начале 1990-х годов, TD-Gammon произвел революцию в искусственном интеллекте для игры «Нарды», используя архитектуру нейронной сети, обученную с помощью метода «Обучение с подкреплением». Благодаря самостоятельной игре и миллионам имитационных игр TD-Gammon научился оценивать позиции на доске, предсказывать исходы и принимать стратегические решения, устанавливая новый стандарт адаптивного и интеллектуального игрового процесса.

В основе строения нейронных сети данного типа, лежит процесс принятия решений по Маркову, а именно математическое моделирование принятия решений с использованием дискретных временных шагов. На каждом i-ом шаге обучения осуществляется выбор некоторого хода $a_i \in A$, где A — является множеством всех доступных ходов для некоторого состояния $s_i \in S$, где S — определяет множество всех доступных состояний.

Каждый ход приводит к изменениям некоторой окружающей среды (адаптивное проблемное пространство с такими атрибутами, как переменные, граничные значения, правила и допустимые действия). В свою очередь, окружающая среда выполняет оценку a_i-го хода учитывая s_i-е состояние. Таким образом, находясь в состоянии s_i алгоритм выполняет поиск a_i-го хода, согласно некоторой оценке r_i и получает от окружающей среды новую оценку r_{i+1}, переходя в состояние s_{i+1} (рис. 3.).

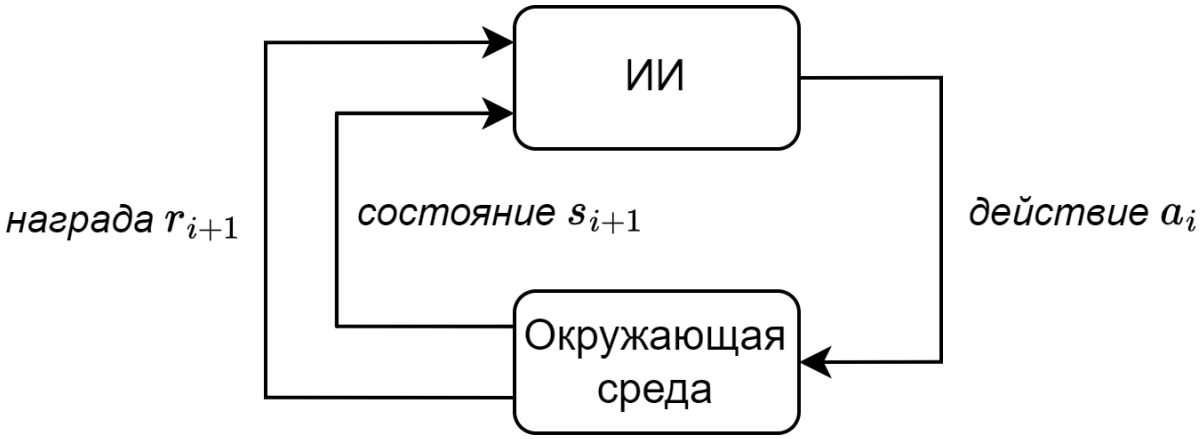


Рис. 3. Схематичное представление процесса обучения с подкреплением

Подробнее с результатами исследования можно ознакомиться в статье [3].

Заключение

В настоящей работе были рассмотрены классические эвристические подходы, алгоритмы по обработке и обходу деревьев решений, а также методологии последних лет, основанные на нейронных сетях. Каждая из этих подходов обладает уникальными преимуществами и возможностями, что делает данные методологии — ценными инструментами в области искусственного интеллекта и принятия решений, в частности.

Литература

[1] Eshaghian Alex. Backgammon Backgame Strategies. — Hardcover, 2023.

[2] Russell Stuart J, Norvig Peter. Artificial intelligence: a modern approach. — Pearson, 2016.

[3] Tesauro Gerald и др. Temporal difference learning and TD-Gammon // Communications of the ACM. — 1995. — Т. 38, № 3. — С. 58–68.

УДК 004.021, 004.051

ОБЗОР АЛГОРИТМА СОГЛАСОВАНИЯ REACT FIBER

Ю.С. Полякова, К.Г. Резников

Воронежский государственный университет

Введение

В настоящее время использование библиотек и фреймворков стало неотъемлемой частью веб-разработки [1]. Стандарты веб-технологий постоянно улучшаются, а сложность растет. Одним из самых популярных готовых инструментов для разработки является React. Инстурмент представляет собой JavaScript библиотеку для создания пользовательских интерфейсов. В ее основе лежит алгоритм, который отслеживает изменения в состоянии компонента и обновляет состояние страницы в браузере. Данный процесс называется согласование (reconciliation).

В ранних версиях библиотеки алгоритм обновления состояния был синхронным и работали по принципу стек структуры. Добавление новых элементов и удаляение существующих возможно только с вершины, что вызывало ряд проблем. Например, работа не могла быть прервана после ее начала или начата до опустошения имеющегося стека.

Fiber - фундаментальное изменение, впервые появившееся в 16 версии React, представляющее собой масштабную переработку старого алгоритма согласования. Данный алгоритм согласования от React называется Fiber Reconciler. Название происходит от fiber, которое используется для представления узла дерева DOM (Document Object Model).

Основными целями средства согласования Fiber являются инкрементный рендеринг, представляющий собой процесс получения изображения и позволяющий разделять и распределять работу по отображению, улучшение или более плавный рендеринг анимации и жестов пользовательского интерфейса, а также оперативность взаимодействия с пользователем. Кроме того, новое согласование позволяет разделить работу на несколько частей и разбить работу рендеринга на несколько кадров. Оно также добавляет возможность определять приоритет для каждой задачи, останавливать работу и возвращаться к ней позже, пере использовать уже завершенные работы или приостанавливать работу в случае, если она больше не требуется.

В настоящей статье подробно рассматривается алгоритм, на котором основаны средства согласования Fiber библиотеки React [2].

1. Структура fiber-узлов

Как самостоятельная единица fiber (строчная буква «f») — это простой объект языка JavaScript. Он представляет собой элемент React или узел дерева DOM.

Во время согласования данные каждого React элемента, возвращенные из метода *render*, объединяются в дерево fiber-узлов. Каждый React элемент имеет соответствующий fiber-узел. В отличие от React элементов, fiber-узлы не создаются заново при каждом рендере, а работает за счет мутабельных структур данных, которые хранят состояние компонентов и DOM.

Когда элемент React впервые преобразуется в fiber-узел, React использует данные из элемента для создания fiber в функции *createFiberFromTypeAndProps*. При последующих

обновлениях React повторно использует fiber-узел и просто обновляет необходимые свойства, используя данные из соответствующего React-элемента. React также может потребоваться переместить узел в иерархии или удалить его, если соответствующий элемент React больше не возвращается из метода render.

Все fiber-узлы связаны между собой через список, используя следующие свойства fiber-узлов: *child*, *sibling* и *return* [3].

2. Работа React Fiber

React выполняет работу в двух основных фазах: render и commit. Во время первой фазы render React применяет обновления к компонентам, запланированные через *setState* или React.render, и выясняет, что требуется обновить в пользовательском интерфейсе. Если это первоначальный рендеринг, React создает новый fiber-узел для каждого элемента, возвращенного из метода *render*. При последующих обновлениях fiber-узлы для существующих React элементов используются повторно и обновляются. Результатом фазы является дерево fiber-узлов, помеченных побочными эффектами. Эффекты описывают работу, которая должна быть выполнена во время следующей фазы commit. Во время этой фазы React берет fiber-дерево, помеченное эффектами, и применяет их к экземплярам. Он просматривает список эффектов и выполняет обновления дерева DOM и другие изменения, видимые пользователю.

Важно понимать, что работа во время первой render фазы может выполняться асинхронно. React может обработать один или несколько fiber-узлов в зависимости от доступного времени, затем остановиться, чтобы сохранить проделанную работу и уступить какому-либо событию, затем продолжает работу с того места, где остановился. Иногда, однако, может потребоваться отбросить проделанную работу и начать все сначала. Такие паузы возможны благодаря тому, что работа, выполняемая в этой фазе, не приводит к каким-либо видимым пользователю изменениям, таким как обновление DOM. В отличие от этого, следующая commit фаза всегда синхронна. Это происходит потому, что работа, выполняемая на этом этапе, приводит к изменениям, видимым пользователю, например, к обновлению дерева DOM. Поэтому React должен выполнять её за один проход.

Отдельно стоит остановиться на том, как React Fiber создает дерево связанных списков и что делает при наличии обновлений. Перед следует объяснить, что такое текущее (current) дерево и выполняемое (workInProgress) дерево и как происходит обход этих деревьев.

Дерево, которое в данный момент очищается для визуализации пользовательского интерфейса, называется текущим. Именно оно использовалось для рендеринга текущего пользовательского интерфейса. При каждом обновлении Fiber создает выполняемое дерево, которое создается на основе обновленных данных из элементов React. React выполняет работу над этим деревом и, после его обновления, использует для следующего рендеринга. Как только это выполняемое дерево отображается в пользовательском интерфейсе, оно становится текущим деревом. Диаграмма описанных деревьев представлена на рисунке 1.

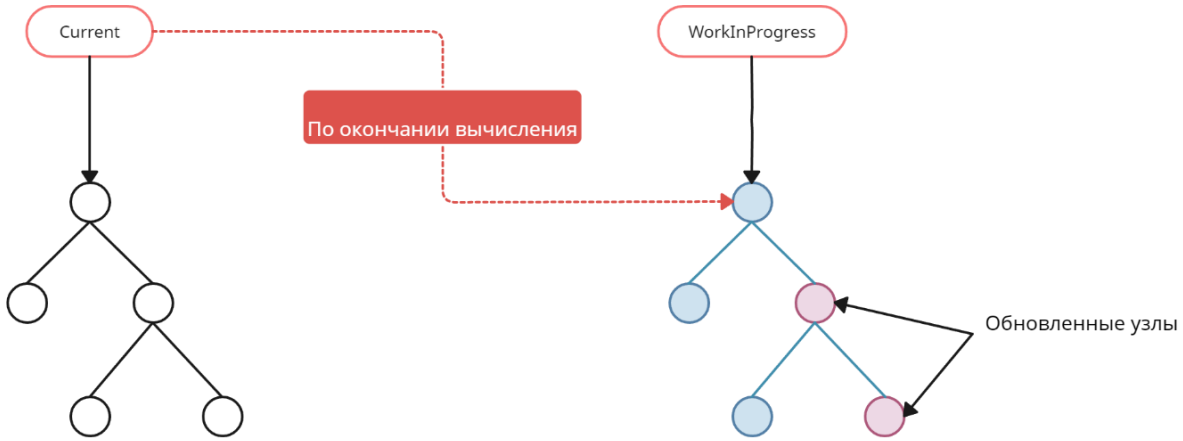


Рис. 1. Текущие и выполняемые деревья

3. Render фаза

Алгоритм согласования всегда начинается с самого верхнего fiber-узла HostRoot с помощью функции *renderRoot*. Например, если вы вызвать метод *setState* глубоко в дереве компонентов, то React начнет с вершины, но быстро пропустит родительские узлы, пока не доберется до компонента, у которого и был вызван этот метод.

Существует 4 основные функции, которые используются для обхода дерева и инициирования или завершения работы:

- *performUnitOfWork*;
- *beginWork*;
- *completeUnitOfWork*;
- *completeWork*.

Рассмотрим первые две функции *performUnitOfWork* и *beginWork*:

Функция *performUnitOfWork* получает fiber-узел из выполняемого дерева и начинает работу, вызывая функцию *beginWork*. Это функция, которая запускает все действия, которые должны быть выполнены для fiber-узла. Функция *beginWork* всегда возвращает указатель на следующего ребенка (child) для обработки в цикле или *null*.

Если есть следующий ребенок, происходит переход к нему. Однако если дочернего элемента нет, React знает, что достиг конца ветви, и поэтому может завершить текущий узел. После того, как узел будет завершен, нужно будет выполнить работу для сиблингов (sibling) и вернуться к родителю после этого**.** Эти действия производятся в функции *completeUnitOfWork*.

Весь код функции заключается в большом цикле *while*. React попадает в эту функцию, когда у узла workInProgress нет дочерних элементов. После завершения работы для текущего fiber-узла, он проверяет, есть ли у него дочерний узел. Если он найден, React выходит из функции и возвращает указатель на сиблинга. Он будет присвоен переменной *nextUnitOfWork*, и React выполнит работу для ветви, начинающейся с этого сиблинга с помощью *compliteWork*. Важно понимать, что на данный момент React выполнил работу только для предшествующих сиблингов. Он не выполнил работу для родительского узла. Только когда все ветви, начинающиеся с дочерних узлов, завершены, React завершает работу для родительского узла и возвращается назад [4].

Как видно из реализации, обе функции *performUnitOfWork* и *completeUnitOfWork* используются в основном для целей итерации, в то время как основная деятельность

происходит в функциях *beginWork* и *completeWork*. Упрощенный вариант обхода дерева при работе render фазы можно увидеть на рисунке 2.

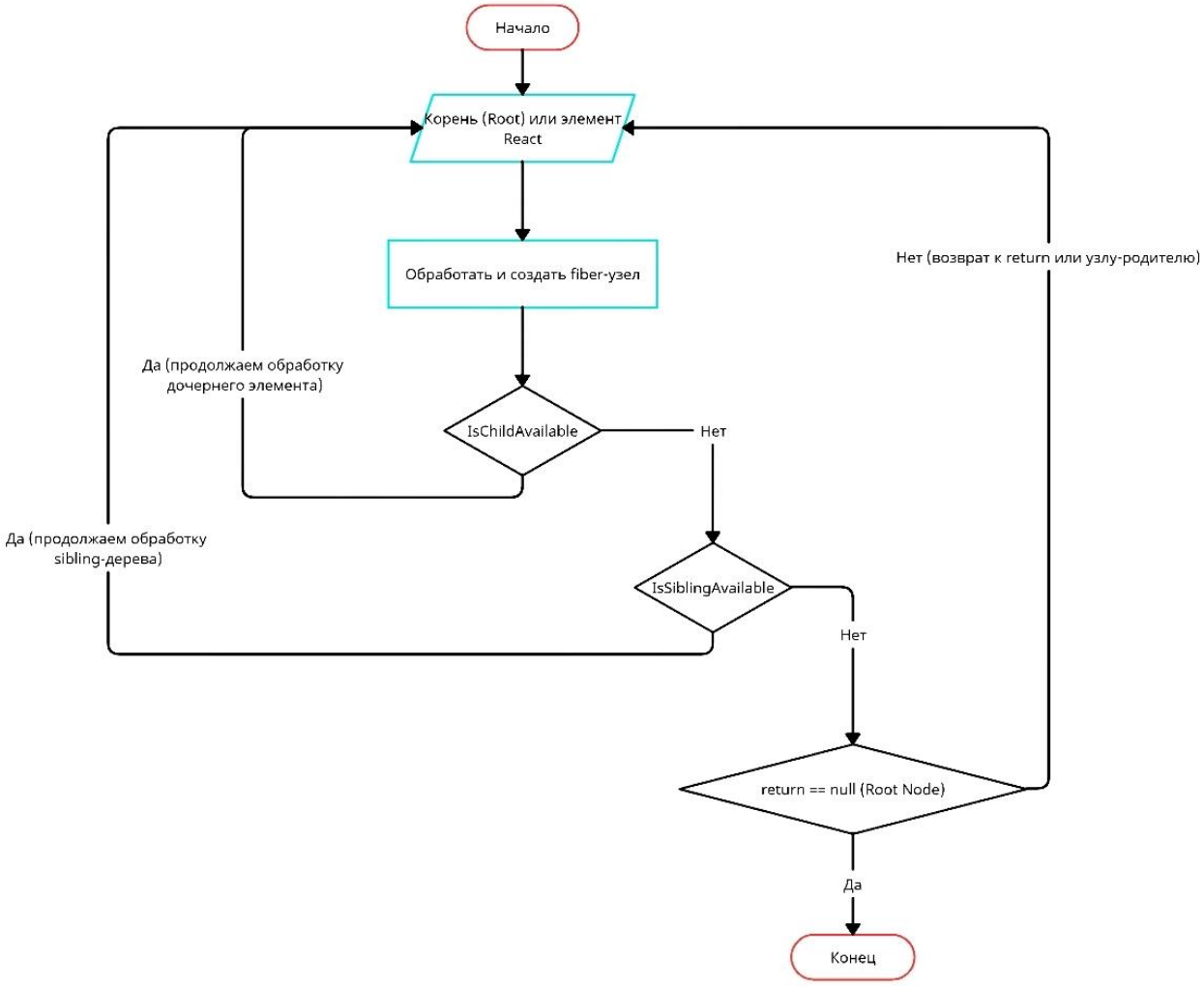


Рис. 2. Обход дерева в React Fiber

4. Commit фаза

Данная фаза начинается с функции *completeRoot*. Именно здесь React обновляет дерево DOM и вызывает методы жизненного цикла до и после мутации.

Когда React доходит до этой фазы, у него есть два дерева и список эффектов. Первое дерево представляет состояние, которое в настоящее время отображается на экране. Затем есть альтернативное дерево, построенное во время фазы render. Именно оно называется выполняемым деревом и представляет состояние, которое должно быть отражено на экране. Это альтернативное дерево связано с текущим деревом через указатели *child* и *sibling*.

Также присутствует список эффектов - подмножество узлов из выполняемого дерева, связанное через указатель *nextEffect*. Следует помнить, что список эффектов — это результат выполнения фазы render. Основанная задача рендеринга заключается в том, чтобы определить, какие узлы должны быть вставлены, обновлены или удалены, и какие компоненты должны вызвать свои методы жизненного цикла. И именно об этом нам говорит список эффектов. И это именно тот набор узлов, который итерируется во время фазы commit.

В целях отладки доступ к текущему дереву можно получить через свойство *current* fiber-корня. Доступ к выполняемому дереву можно получить через свойство *alternate* узла HostFiber в текущем дереве.

Основной функцией, выполняемой на этапе commit, является *commitRoot*. В основном, эта функция делает следующее:

- вызывает метод жизненного цикла *getSnapshotBeforeUpdate* на узлах, помеченных эффектом Snapshot;
- вызывает метод *componentWillUnmount* жизненного цикла для узлов, помеченных эффектом Deletion;
- выполняет все вставки, обновления и удаления в DOM;
- устанавливает дерево finishedWork в качестве текущего;
- вызывает метод жизненного цикла *componentDidMount* на узлах, помеченных эффектом Placement;
- вызывает метод *componentDidUpdate* жизненного цикла для узлов, помеченных эффектом Update.

После вызова метода предварительной мутации *getSnapshotBeforeUpdate,* React фиксирует все побочные эффекты внутри дерева. Он делает это в два прохода. Первый проход выполняет все вставки, обновления, удаления и размонтирования DOM (хоста). Затем React назначает дерево finishedWork на FiberRoot, помечая дерево workInProgress как текущее. Это делается после первого прохода фазы commit, чтобы предыдущее дерево было актуальным во время выполнения *componentWillUnmount*, но до второго прохода, чтобы законченная работа была актуальной во время *componentDidMount*/*Update*. Во время второго прохода React вызывает все остальные методы жизненного цикла и обратные вызовы. Эти методы выполняются как отдельный проход, так что все размещения, обновления и удаления во всем дереве уже были вызваны [5, 6].

Заключение

В данной статье была рассмотрена работа алгоритма согласования React Fiber. Было подробно разобрано как происходит преобразование React элементов к fiber-узлам, как строятся и обновляются DOM деревья. Отдельно были рассмотрены основные фазы алгоритма – render и commit.

Таким образом React Fiber является фундаментальным изменением для улучшения производительности благодаря использованию инкрементного рендеринга, что позволяет эффективно работать с веб-приложениями, переполненными анимацией и сложными элементами. Это дает Fiber значительное преимущество над старым алгоритмом обновления состояния React, работающего по принципу стек структуры.

Литература

1. Резников К. Г. Разработка программного обеспечения для визуализации трехмерных поверхностей в веб-браузере / К. Г. Резников, С. Н. Медведев // Вестник ВГТУ. Том 17, №6. – Воронеж: ВГТУ, 2021 - С. 13-19.
2. Яровая, Е. В. Архитектурные решения в библиотеке Реакт / Е. В. Яровая // Столыпинский вестник. – 2022. – № 5. – С. 2679-2684.
3. An Introduction to React Fiber - The Algorithm Behind React // Velotio.com : сайт. – URL: https://www.velotio.com/engineering-blog/react-fiber-algorithm (дата обращения: 06.03.2024)
4. Virtual DOM and Internals– Режим доступа: https://legacy.reactjs.org/docs/faq-internals.html#gatsby-focus-wrapper (Дата обращения: 06.03.2024).

5. Reconciliation. – Режим доступа: https://legacy.reactjs.org/docs/reconciliation.html (Дата обращения: 06.03.2024).
6. ReactFiberScheduler.js – Режим доступа: https://github.com/facebook/react/blob/95a313ec0b957f71798a69d8e83408f40e76765b/packages/react-reconciler/src/ReactFiberScheduler.js (Дата обращения: 06.03.2024).

УДК 004.4

РАЗРАБОТКА FRONTEND ПРИЛОЖЕНИЯ «СУРДОПЕРЕВОД» С ИСПОЛЬЗОВАНИЕМ ФРЕМЙМВОРКА REACT

Ю.С. Полякова, Е.В. Трофименко

Воронежский государственный университет

Введение

Глухота и слабослышание остаются серьезными проблемами в современном обществе. В России насчитывается значительное количество лиц, имеющих нарушения слуха, как среди взрослых, так и среди детей. Так, статистика показывает, что количество взрослых глухих и слабослышащих в России на 2023 год составляет около 5 миллионов человек. Среди них около 2 миллионов человек (40%) являются полностью глухими, то есть неспособны услышать звуки на одном или обоих ушах. Остальные 3 миллиона человек (60%) имеют проблемы со слухом, но могут использовать слуховые аппараты или другие средства для улучшения слуха. Что касается детей, то статистика сильно различается в зависимости от возрастной группы. По оценкам специалистов, около 1-2% новорожденных имеют слуховые проблемы. Это означает, что в год в России появляется около 10-20 тысяч глухих или слабослышащих детей. Уже в возрасте 3 лет количество детей со слуховыми нарушениями увеличивается и составляет около 3-5% от общего числа [1].

Люди с нарушениями слуха в своем общении наиболее часто используют жестомимический язык, дактильную (пальцевую) азбуку и чтение с губ. Все эти способы относятся к невербальным средствам общения. Обычные же люди используют преимущественно вербальный способ общения и зачастую не имеют достаточных знаний для перехода к невербальному общению с теми, кто не способен общаться словесно, поэтому задача разработки веб-приложения «Сурдоперевод» является весьма актуальной. В статье рассматривается разработка frontend приложения «Сурдоперевод» с использованием фреймворка React.

1. Средства и методы разработки

В основе приложения лежит SPA или Single Page Application подход. SPA представляет собой одностраничное веб-приложение, которое загружается на одну HTML-страницу. Благодаря динамическому обновлению с помощью JavaScript, во время использования не нужно перезагружать или подгружать дополнительные страницы. Это означает, что пользователь видит в браузере весь основной контент, а при прокрутке или переходах на другие страницы, вместо полной перезагрузки нужные элементы просто подгружаются [2].

Для разработки приложения был выбран React — JavaScript-библиотека для создания пользовательских интерфейсов. В React.js реализован декларативный подход. Такой метод носит название «UI — функция данных». Он предусматривает, что разработчик описывает поведение интерфейса в зависимости от событий и данных. У React традиционно выделяют ряд преимуществ, которые выделяют ее других библиотек, предназначенных для разработки интерфейсов. В первую очередь это высокая гибкость библиотеки, позволяющая выбрать другие библиотеки, которые следует использовать для решения узких задач, например,

установления контроля за хранением данных и переходами между экранами. Если позволяет функциональность, то всё приложение можно сделать целиком на React. Еще одним преимуществом является универсальность. Будучи JavaScript-библиотекой React может работать в самом разном окружении и на самых разных устройствах [3].

Еще одной отличительной чертой React является использование *компонентного подхода*. Компонентный React позволяет повторно использовать существующий код и создавать сложные пользовательские интерфейсы из простых строительных блоков, называемых компонентами. React поощряет повторное использование компонентов, позволяя разработчику создать библиотеку многократно используемых компонентов, которые можно размещать на всех частях разрабатываемого проекта. Это помогает сократить время разработки и улучшить организацию кода, а также позволяет упростить обслуживание и обновление программных систем за счет изоляции и модульности конкретных функциональных возможностей [4].

Также следует отметить, что React использует *виртуальный DOM* вместо реального DOM для отображения обновлений. Это позволяет React вносить. изменения, не требуя перезагрузки страницы. Использование виртуального DOM позволяет эффективно обновлять интерфейс, отображая только измененные компоненты, что приводит к улучшению масштабируемости и производительности веб-приложений.

Виртуальный DOM в React позволяет эффективно обновлять и отображать страницы, что приводит к повышению производительности по сравнению с традиционными методами. Это делает React популярным выбором для создания высокопроизводительных веб-приложений. Использование виртуального DOM также предотвращает межсайтовые скриптовые атаки, что делает его безопасным вариантом для веб-разработки. Более того, односторонняя привязка данных в React снижает риск манипулирования данными и повышает безопасность [5].

2. Разработка приложения

Приложение «Сурдоперевод» разрабатывается для помощи в общении с людьми, обладающими нарушениями слуха. Так как целевой группой пользователей являются граждане России, основными языками для перевода в приложении будут являться русский и русский жестовый язык (РЖЯ). Приложение, в свою очередь, должно предоставлять пользователю возможность как переводить РЖЯ в текст на русском языке, так и производить обратное преобразование. Поскольку программный продукт является первым приложением с указанным функционалом, необходимым условием для своевременного улучшения продукта является добавление возможности использования обратной связи, позволяющей оставлять отзывы о работе приложения, а также предложения по его улучшению. Возможности пользователя при взаимодействии с приложением отображены на рисунке 1.

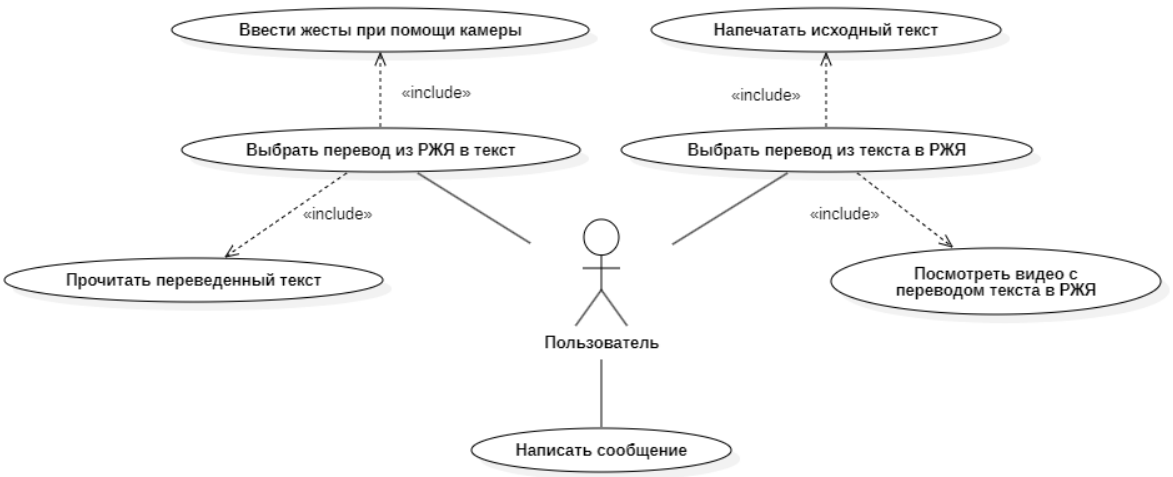


Рис. 1. Диаграмма вариантов использования приложения

Интерфейс приложения должен быть максимально прост и понятен для пользователя и включать в себя только основные функции. Составленная структура главного окна интерфейса приложения представлена на рисунке 2 и включает в себя ряд элементов.

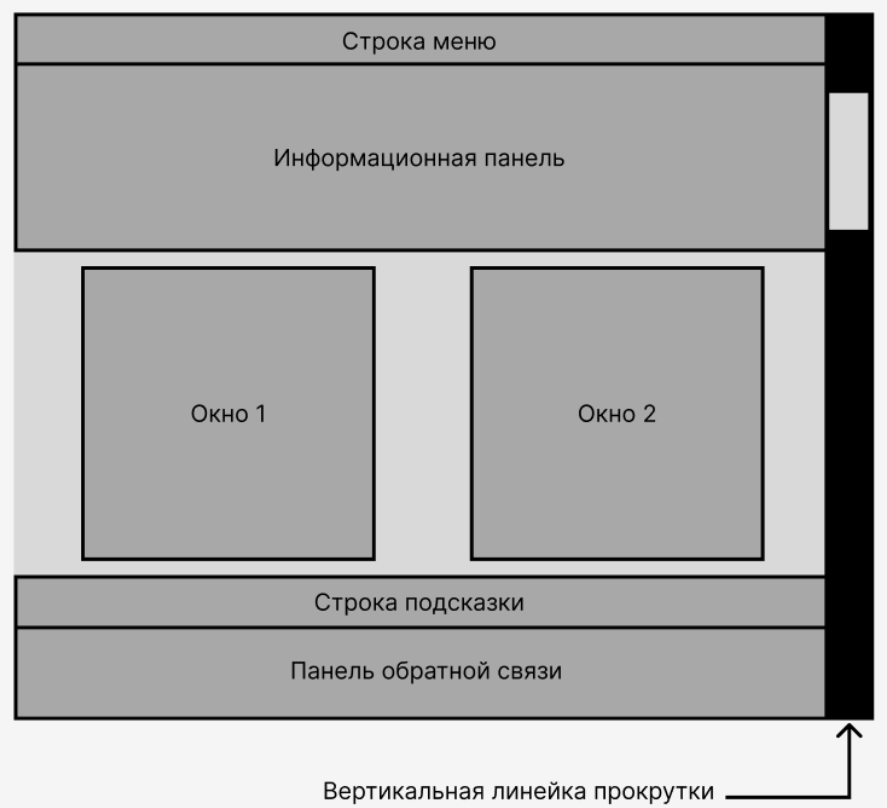


Рис. 2. Структура главной страницы приложения

Строка меню содержит имена групп команд, объединенных по функциональному признаку. Строка меню находится в верхней части экрана. Выбор режима из строки меню открывает соответствующее подменю, а выбор определенной опции в нем обеспечивает доступ к меню более низкого уровня. Такая система вложенных меню составляет основу интерфейса и присутствует на каждом из реализованных окон приложения.

Информационная панель отображает основную информацию о программном продукте и кратко описывает предоставляемые для пользователя возможности, в то время *как строка подсказки* содержит информацию о возможных действиях пользователя в текущий момент.

Панель обратной связи содержит контактные данные разработчиков, а также позволяет пользователям напрямую отправить отзыв о работе приложения или свои предложения по улучшению. Сама возможность обратной связи реализована в виде формы, в которую пользователь должен ввести свою электронную почту и непосредственно сам текст обращения.

Отдельно стоит остановиться на элементах, обозначенных как *Окно 1* и *Окно 2*. Они представляют собой кнопки, при нажатии которых осуществляется переход в окна, соответствующие выбранному типу перевода. Структура этих окон отображена на рисунке 3.

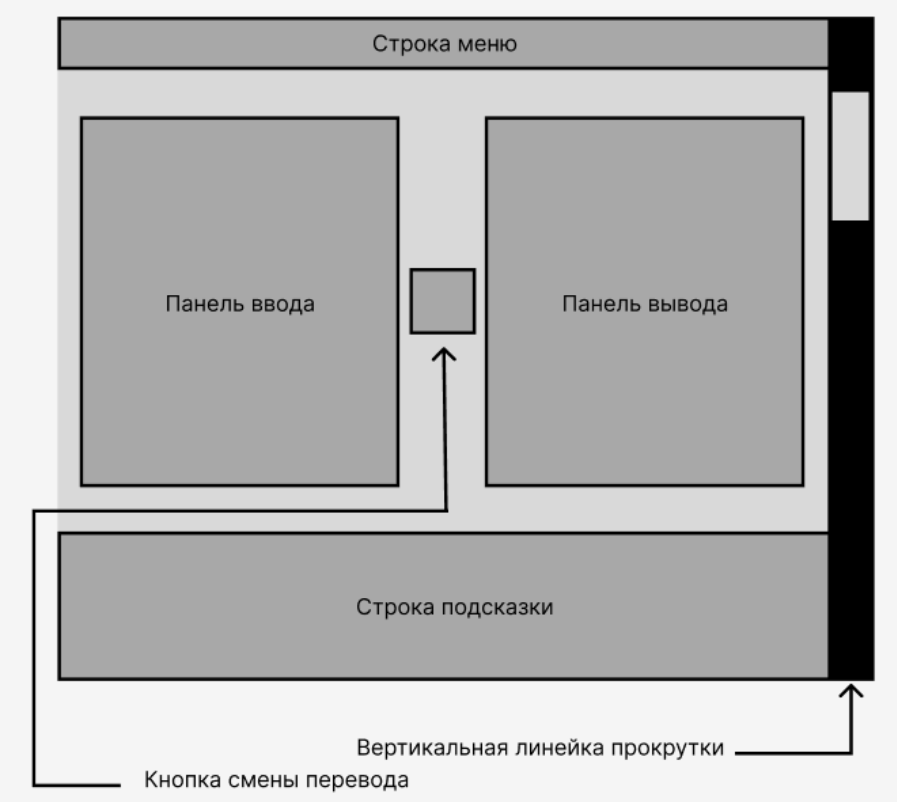


Рис. 3. Структура окна перевода

Новыми для интерфейса приложения в этом окне являются элементы *Панель ввода* и *Панель вывода.* В случае перевода, подразумевающего преобразование РЖЯ в текст, через панель ввода приложение получает доступ к камере устройства пользователя, считывает показанные жесты и переводит их в текстовый формат, поочередно распознавая каждый из показанных жестов, после чего отображает в окне панели вывода. Если же был выбран перевод из текста в РЖЯ, пользователь вручную вводит текст в область панели ввода и получает видеоряд из жестов, демонстрируемым виртуальным переводчиком (рис. 4).

Рис. 4. Виртуальный переводчик

Преобразование происходит путем отображения в нужном порядке видеофайлов демонстрации жестов виртуальным переводчиком, URL которых при обращении к ним совпадает с самим словом, которое требуется перевести. Таким образом видеоряд воспроизводится без задержек, обусловленных преобразованием и детальной обработкой исходного теста.

Заключение

В данной статье была рассмотрена разработка frontend приложения «Сурдоперевод» с использованием фреймворка React, позволяющее облегчить коммуникацию с людьми, обладающими нарушениями слуха, путем перевода как РЖЯ в текст, так и обратного перевода. Приложение является удобным и интуитивно понятным в использовании и не требует от пользователя специальных навыков и умений.

Литература

1. Статистика: количество глухих и слабослышащих взрослых и детей в России на 2023 год – Режим доступа: https://aktosr.ru/chitayut/statistika-kolichestvo-gluhih-i-slaboslyshaschih-vzroslyh-i-detey-v-rossii-na-2023-god?utm_referrer=https://www.google.com/ – (Дата обращения: 24.03.2024)
2. Виды веб-приложений: выбираем подходящий вариант для вашего бизнеса – Режим доступа: https://leantech.ai/vebprilozhenie-osnovnye-vidy-i-ih-osobennosti – (Дата обращения: 09.04.2024).
3. Берьянов, М. С. Исследование современного стека React разработки в 2022 году / М. С. Берьянов, И. Р. Салахов, М. Д. Иванов // Столыпинский вестник. – 2022. – № 8. – С. 4766-4784.
4. Компонентный подход – Режим доступа: https://habr.com/ru/sandbox/182978/ – (Дата обращения: 09.04.2024).
5. Горбачев, А. А. Сравнение классического процесса реализации веб-приложений и подхода с использованием библиотеки React / А. А. Горбачев, Е. С. Горбачева // Молодой исследователь Дона. – 2020. – Т. 22, № 1. – С. 28-31.

УДК *004.021*

ИССЛЕДОВАНИЕ МЕТОДОВ МНОГОКРИТЕРИАЛЬНОГО АНАЛИЗА В ЗАДАЧЕ ПОСТРОЕНИЯ МАРШРУТОВ

Д. В. Попов

Воронежский государственный университет

Введение

Построение маршрутов является фундаментальной задачей в различных областях, таких как логистика, транспорт, торговля и даже наука о данных. Эта проблема становится особенно актуальной в условиях растущей сложности современных систем, где требуется эффективное распределение ресурсов для минимизации временных и финансовых затрат.

Одним из основных аспектов при решении задачи построения маршрутов является многокритериальный анализ. В отличие от классических методов, которые стремятся оптимизировать единственный критерий, многокритериальный подход учитывает несколько конфликтующих целей одновременно. Это важно, поскольку реальные системы могут иметь множество целей, включая минимизацию времени, расходов на топливо, уменьшение загруженности дорог и многие другие.

Данное исследование рассматривает различные методы многокритериального анализа в контексте построения маршрутов. Цель состоит в изучении эффективности и применимости различных методов в различных сценариях, а также исследовании их преимуществ и недостатков.

1. Актуальность задачи

Современный мир стал свидетелем стремительного роста городской плотности населения и объемов грузоперевозок, что создает огромные вызовы для транспортной инфраструктуры и логистических систем. В контексте такого быстрого развития становится критически важным эффективное управление маршрутами и транспортными потоками. Анализ и построение маршрутов перевозок необходимы для минимизации временных и финансовых затрат, улучшения обслуживания клиентов и снижения негативного воздействия на окружающую среду. Адаптивные алгоритмы, способные учитывать изменчивые условия дорожного движения и предсказывать возможные проблемы, играют ключевую роль в обеспечении эффективности и устойчивости логистических систем.

Исследование методов многокритериального анализа в построение маршрутов представляет собой одну из актуальных задач в современном мире, особенно в контексте быстрого развития городов, роста объемов грузоперевозок и увеличения мобильности населения. Сложность современных транспортных сетей и разнообразие факторов, влияющих на выбор маршрутов, таких как временные ограничения, транспортные пробки, экологические ограничения и требования безопасности, требуют разработки инновационных подходов к построения маршрутов. Эффективное управление транспортными потоками становится ключевым для снижения затрат, улучшения обслуживания и уменьшения негативного воздействия на окружающую среду. Таким образом, изучение методов многокритериального анализа в построение маршрутов не только имеет теоретическую значимость, но и непосредственное приложение в современной логистической практике, что делает эту задачу

важной и востребованной.

2. Сбор данных

В процессе исследования методов многокритериального анализа в построение маршрутов одним из ключевых этапов является сбор данных, который представляет собой систематическое и целенаправленное получение информации о различных аспектах маршрутов.

В основном, сбор данных для исследований многокритериального анализа в построение маршрутов часто используются уже готовые ГИС (геоинформационные системы) решения. Эти системы предоставляют доступ к обширным базам геопространственных данных, включая цифровые карты, сведения о дорожной инфраструктуре, географические признаки и данные о транспортных потоках. При помощи ГИС можно осуществлять анализ маршрутов с учетом различных критериев, таких как расстояние, время в пути, транспортные пересадки, а также учитывать различные ограничения, например, дорожные работы или пробки.

Использование уже готовых ГИС решений обладает рядом преимуществ, включая доступность обширных баз данных, удобство в использовании, а также возможность автоматизации анализа и построение маршрутов. Такие системы часто предоставляют широкий набор инструментов для визуализации данных, проведения анализа и принятия решений на основе полученных результатов. В то же время, важно учитывать особенности каждого конкретного ГИС решения и его способность адаптироваться к уникальным требованиям и задачам исследования.

3. Обработка данных

3.1. Метод взвешенных критериев (MCDM)

Метод взвешенных критериев (MCDM) — это аналитический подход, используемый для принятия решений в условиях множества критериев. Основная идея заключается в том, чтобы преобразовать различные, часто несопоставимые, критерии в общую шкалу, которая позволит сравнивать различные альтернативы. Для этого каждый критерий оценивается по своей значимости с помощью заданных весов. Веса могут быть определены экспертно или на основе анализа данных [1].

Одним из ключевых преимуществ MCDM в построение маршрутов является его простота применения. Этот метод легко адаптируется к различным ситуациям и требованиям, что делает его удобным инструментом для принятия решений в логистическом и транспортном секторе. Гибкость MCDM позволяет учитывать предпочтения пользователей, что позволяет создавать персонализированные маршруты.

Тем не менее, MCDM имеет свои недостатки. Один из них – субъективность в определении весов критериев. Это может привести к искажению результатов, особенно если веса задаются экспертами на основе их субъективного мнения. Кроме того, метод может игнорировать взаимосвязь между критериями, что может привести к неправильным выводам. Еще одним недостатком является чувствительность к изменениям в весах критериев, что может потребовать повторного анализа и корректировки решений.

3.2. Метод Парето-оптимальности (Pareto Optimality)

Метод Парето-оптимальности основан на концепции множества Парето, которое в контексте многокритериальной оптимизации представляет собой множество альтернатив, для

которых невозможно улучшить хотя бы один критерий без ухудшения других. Иными словами, решение считается Парето-оптимальным, если не существует другого решения, которое было бы лучше по всем критериям одновременно. В построение маршрутов это означает, что множество Парето-оптимальных маршрутов представляет собой набор альтернативных маршрутов, не доминируемых другими, то есть маршрутов, для которых невозможно найти альтернативу, обеспечивающую лучший результат по всем критериям одновременно.

Плюсы метода Парето-оптимальности для задачи многокритериального анализа в построение маршрутов включают его способность предоставлять полное представление о множестве альтернативных маршрутов, учитывая их различные компромиссы между различными критериями, а также обеспечивать прозрачность и наглядность в выборе оптимальных маршрутов для принятия решений. Однако метод Парето-оптимальности может столкнуться с проблемой большого количества альтернативных маршрутов в сложных сценариях, что может затруднить анализ и выбор оптимального маршрута. Также не всегда возможно однозначно определить предпочтения между альтернативами в множестве Парето.

3.3. Метод ограниченного порядка предпочтения (Lexicographic Ordering)

Метод ограниченного порядка предпочтения, или лексикографическое упорядочение, предполагает упорядочивание критериев по важности, а затем сравнение альтернативных маршрутов в соответствии с этим порядком. То есть сначала сравниваются маршруты по самому важному критерию, а затем, если значения равны, по следующему по важности критерию, и так далее. Если два маршрута различаются по первому критерию, то выбирается тот, который лучше по этому критерию. В случае равенства по первому критерию, они сравниваются по второму критерию, и так далее.

Плюсы метода ограниченного порядка предпочтения для задачи многокритериального анализа в построение маршрутов включают его простоту и прямолинейность в принятии решений, а также возможность явного учета важности каждого критерия. Кроме того, этот метод позволяет легко сравнивать альтернативные маршруты и выбирать оптимальный в соответствии с предпочтениями принимающей стороны. Однако метод ограниченного порядка предпочтения может привести к упрощенным или неполным решениям, поскольку он не учитывает возможные компромиссы между критериями и может игнорировать информацию о более сложных отношениях между альтернативами. Также важно точно определить порядок важности критериев, что может быть сложно в реальных ситуациях.

3.4. Метод независимого выбора

Метод независимого выбора (PROMETHEE, Preference Ranking Organization METHod for Enrichment of Evaluations), является алгоритмическим методом многокритериальной оптимизации, который используется для ранжирования альтернативных маршрутов на основе их относительной привлекательности по отношению к заданным критериям. Основная идея заключается в том, чтобы сравнивать каждую пару альтернатив по каждому критерию и вычислять их относительные преимущества. Затем эти относительные преимущества суммируются, учитывая веса критериев, и полученные значения используются для ранжирования маршрутов. PROMETHEE также позволяет учитывать предпочтения принимающей стороны, вводя функцию предпочтения, которая определяет степень удовлетворенности каждой альтернативы [2].

Плюсы метода независимого выбора для задачи многокритериального анализа в построение маршрутов включают его способность учитывать как относительные преимущества альтернатив по каждому критерию, так и предпочтения принимающей стороны, что делает его

более гибким и адаптивным к различным контекстам. Кроме того, PROMETHEE обеспечивает прозрачность в процессе принятия решений, позволяя анализировать вклад каждого критерия и влияние предпочтений на итоговые результаты. Однако метод может столкнуться с проблемой чувствительности к выбору параметров, таких как веса критериев и функции предпочтения, а также с необходимостью в достаточно полной и точной информации о каждой альтернативе и ее характеристиках.

3.5. Метод анализа иерархий

Метод анализа иерархий (AHP) — это структурированный метод принятия решений, разработанный Томасом Саати, который позволяет исследователям и принимающим решениям учитывать различные критерии и предпочтения при принятии сложных решений. AHP основан на идее иерархической структуры, где сложная проблема разбивается на более простые уровни, что облегчает анализ. В рамках AHP критерии и альтернативы представлены в виде иерархической структуры, где на верхнем уровне находится цель, а на последующих уровнях – критерии и подкритерии [3].

Одним из ключевых плюсов AHP для многокритериального анализа в построение маршрутов является его способность учитывать различные аспекты принятия решений, такие как время, стоимость, безопасность и т. д., и преобразовывать их в общий числовой показатель для сравнения альтернатив. Кроме того, AHP позволяет учитывать предпочтения пользователей и веса отдельных критериев, что делает его полезным инструментом для создания оптимальных маршрутов, учитывая различные потребности и предпочтения. Однако, недостатками могут быть сложность определения иерархии критериев, а также субъективность в определении и установлении весовых коэффициентов, что может привести к искажению результатов анализа.

4. Проблемы и ограничения

При исследовании методов многокритериального анализа в построение маршрутов возникают различные проблемы и ограничения, которые могут затруднить процесс принятия решений и получение оптимальных результатов. Одной из основных проблем является сложность учета всех аспектов и ограничений, связанных с логистическими операциями, такими как время в пути, стоимость перевозки, условия дорог и другие факторы. Важно учитывать не только различные критерии, но и их взаимосвязь, что может быть сложной задачей.

Другой проблемой является большое количество вариантов и альтернативных маршрутов. Обработка и анализ больших объемов данных может потребовать значительных вычислительных ресурсов и времени, особенно при использовании сложных моделей многокритериального анализа. Кроме того, возникают ограничения, связанные с точностью и достоверностью данных, используемых для анализа, а также субъективностью при определении весов критериев и предпочтений пользователей. Такие проблемы и ограничения требуют тщательного анализа.

Заключение

Хотя методы взвешенных критериев (MCDM) и ограниченного порядка предпочтения (Lexicographic Ordering) обладают своими преимуществами, они не являются наилучшим выбором для построения маршрутов в данном контексте. MCDM, хоть и прост в применении и гибок, страдает от субъективности в определении весов критериев, что может привести к

искажению результатов анализа. Ограниченный порядок предпочтения, в свою очередь, упрощает решения, но не учитывает компромиссов между критериями и может игнорировать более сложные отношения между альтернативами.

Метод Парето-оптимальности (Pareto Optimality) и метод независимого выбора (PROMETHEE) обладают своими преимуществами, но они также не являются наилучшим выбором для построения маршрутов. Парето-оптимальность может столкнуться с проблемой большого количества альтернативных маршрутов в сложных сценариях, а PROMETHEE требует точной информации о каждой альтернативе и ее характеристиках, что может быть затруднительно в реальных условиях.

В сравнении с вышеперечисленными методами, AHP обеспечивает баланс между гибкостью и надежностью, учитывая различные аспекты принятия решений и предпочтения пользователей. Во-первых, AHP учитывает различные аспекты принятия решений, преобразовывая их в общий числовой показатель для сравнения альтернатив. Это позволяет учесть разнообразные потребности и предпочтения при выборе оптимальных маршрутов. Во-вторых, AHP позволяет учитывать предпочтения пользователей и веса отдельных критериев, что делает его более гибким и адаптивным к различным контекстам.

Литература

1. Т. Рафгарден. Совершенный алгоритм. Графовые алгоритмы и структуры данных : учеб. пособие / Т. Рафгарден. – Москва : Прогресс книга, 2019. – 256 с.

2. Т. Кормен. Алгоритмы. Построение и анализ : учеб. пособие / Т. Кормен. – 2-е изд. – Москва : Вильямс, 2006. – 1296 с.

3. В. В. Подиновский. Многокритериальные задачи принятия решений: теория и методы анализа : учебник для вузов / В. В. Подиновский. – Москва : Юрайт, 2024. — 486 с.

УДК 004.93

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ ТИПА ОВАЛА ЛИЦА С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ

Е. В. Попова, Е. В. Трофименко

Воронежский государственный университет

Введение

На сегодняшний день определение лица на фотографии и видео является важной задачей в области компьютерного зрения и обработки изображений. Функция определение лица человека используется в различных сферах и приложениях, например, для аутентификации пользователей на мобильных устройствах или для приложений, где можно примерить образ к своей фотографии. Стилисты и визажисты в своих исследованиях были бы рады иметь инструмент, способный быстро и эффективно классифицировать людей по овалу лица для подбора образа. Это, безусловно, было бы огромной экономией не только времени, но еще и человеческих и финансовых ресурсов.

Эти различные, выше представленные причины подтверждают необходимость существования приложения, способного эффективно и быстро классифицировать различные овалы лица по фотографии.

Основной целью данной статьи является создание и обучение нейронной сети, способной определять тип овала лица по фотографии и описание приложения, которое позволит пользователям примерять прическу с своей фотографии.

1. Обучение нейронной сети

1.1. Библиотеки для обучения нейронной сети

Для обучения нейронной сети для предсказания типа овала лица по изображению, будут использованы язык программирования Python и среда разработки Google Colab, а также следующие библиотеки:

1. TensorFlow / Keras — для построения и обучения нейронной сети [1];
2. OpenCV — для работы с изображениями, их предварительной обработки и загрузки в нейронную сеть [2];
3. NumPy — для работы с массивами данных, которые будут использоваться в обучении и тестировании модели.

1.2. Этапы обучения нейронной сети

Для реализации обучения нейронной сети для определения типа овала лица по фотографии необходимо выполнить следующие шаги.

1. Загрузить набора данных, который будет использоваться для обучения и тестирования модели.
2. Загрузить изображения из папок загруженного набора данных с использованием библиотеки OpenCV. Также потребуется изменить размеры изображений до одного общего размера, что значительно упростит работу с данными.

3. Разделить данные на обучающий и тестовый наборы. Самый оптимальный вариант — это использовать 75% изображений для обучения и 25% для тестирования.
4. Создать нейронную сеть с помощью TensorFlow и Keras. Модель должна состоять из сверточных слоев для извлечения важных признаков из изображений, а затем полностью связанных слоев для классификации.
5. После создания модели, необходимо ее обучить на обучающем наборе, используя изображения и соответствующие им метки (названия типов овала лица).
6. Наконец, проверить точность модели на тестовом наборе, используя неразмеченные данные для проверки предсказаний и определения точности классификации.

Таким образом, данный алгоритм позволит обучить нейронную сеть на базе изображений различных типов овала лица. Загрузка, предварительная обработка и разделение данных, создание и обучение модели нейронной сети с использованием TensorFlow и Keras, а также проверка ее точности на тестовом наборе позволят разработать систему, способную автоматически определять тип овала лица по предоставленной фотографии. Данная система может иметь широкий спектр применения, начиная от решений в области красоты и моды и заканчивая медицинскими и исследовательскими целями [3].

1.3. Реализация алгоритма обучения нейронной сети

Для обучения нейронной сети распознаванию форм лица был загружен набор данных с сайта Kaggle. В данном наборе хранится 5 папок с названиями типа овала лица: Heart (в форме сердца), Oblong (вытянутое), Oval (овальное), Round (круглое), Square (квадратное). Итоговое количество - 5000 изображений с лицами людей.

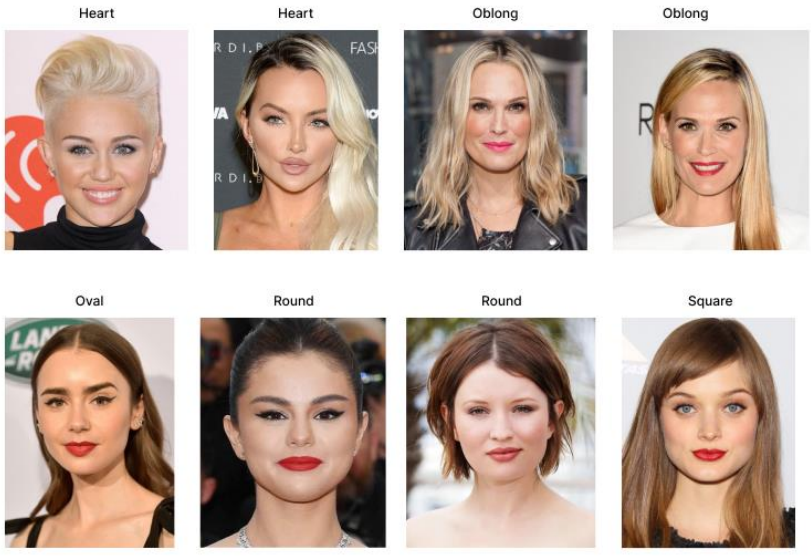


Рис. 12. Примеры изображений в наборе данных

На рисунке 2 изображен код, который загружает изображения из папок различных типов лица (Square, Oblong, Oval, Heart, Round) и их метки (face_types) из заданной директории (dataset_path), предварительно обработанные и добавляет их в списки data и labels.

Каждое изображение преобразуется с помощью OpenCV функции cv2.imread() и cv2.resize() для изменения размера изображения до 224x224 пикселей. Затем изображения добавляются в список data, а их соответствующие метки добавляются в список labels. Списки data и labels преобразуются в массивы numpy с помощью np.array(). Изображения также

нормализуются путем деления на 255.0, чтобы получить значения в диапазоне от 0 до 1.0. Это будет полезно для обучения нейронной сети.

```
# создадим пустые списки для изображений и их меток
data = []
labels = []
```

```
for face_type in face_types:
    face_folder = os.path.join(dataset_path, face_type)
    for img_name in os.listdir(face_folder):
        if not img_name.startswith('.'):
            img_path = os.path.join(face_folder, img_name)
            img = cv2.imread(img_path)
            if img is not None:
                img = cv2.resize(img, (224, 224))
                data.append(img)
                labels.append(face_type)
```

```
# преобразуем списки в массивы numpy
data = np.array(data) / 255.0
labels = np.array(labels)
```

Рис. 13. Подготовка изображений для дальнейшего использования

Затем используется LabelBinarizer из библиотеки scikit-learn для преобразования категориальных меток в бинарное представление [4]. Исходные метки имеют значения «Square», «Oblong», «Oval», «Heart», «Round», но после бинаризации каждая метка будет представлена в виде бинарного вектора. Например, «Square» станет 1, 0, 0, 0, 0, «Oblong» станет 0, 1, 0, 0, 0, и так далее.

```
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
```

Рис. 14. Преобразование меток в бинарное представление

Для классификации изображений будет использована известная глубокая нейронная сеть VGG16. Она состоит из 16 слоев, включая сверточные слои и полносвязанные слои. С помощью библиотеки Keras загружается предварительно обученная модель VGG16 с весами, обученными на наборе данных ImageNet.

На рисунке 4 показано создание модели нейронной сети. Сначала создается экземпляр модели Sequential. В качестве первого слоя модели добавляется предварительно обученная модель VGG16. Затем добавляется слой Flatten, который преобразует выход предыдущего слоя в одномерный вектор, чтобы можно было добавить полностью связанные слои. Далее добавляется полносвязный слой с 64 нейронами и функцией активации ReLU и наконец, добавляется выходной полносвязный слой с 5 нейронами (соответствующими количеству классов) и функцией активации softmax, которая используется для многоклассовой классификации.

```
# Создаем нейронную сеть с использованием предобученной модели VGG16
vgg_model = VGG16(include_top=False, input_tensor=Input(shape=(224, 224, 3)))

model = Sequential()
model.add(vgg_model)
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(5, activation='softmax'))
```

Рис. 15. Создание модели нейронной сети

Компиляция и обучение модели нейронной сети показано на рисунке 5. В данном случае, функция потерь установлена как «categorical_crossentropy», что предполагает, что модель выполняет многоклассовую классификацию. Оптимизатор задан как «adam». Кроме этого, указана метрика «accuracy», которая будет использоваться для оценки точности модели во время обучения. С помощью метода fit() обучается модель на предоставленных данных. Количество эпох равно 10. Batch_size, то есть размер пакета, указывающий, сколько образцов будет обрабатываться перед обновлением весов модели, равен 32.

```
# скомпилируем модель
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# обучим модель на обучающих данных
model.fit(trainX, trainY, validation_data=(testX, testY), epochs=10, batch_size=32)
```

```
Epoch 1/10
10/10 [==============================] - 273s 28s/step - loss: 2.6521 - accuracy: 0.2778 - val_loss: 1.6430 - val_accuracy:
Epoch 2/10
10/10 [==============================] - 272s 28s/step - loss: 1.2565 - accuracy: 0.4444 - val_loss: 1.4862 - val_accuracy:
Epoch 3/10
10/10 [==============================] - 275s 28s/step - loss: 0.8508 - accuracy: 0.7026 - val_loss: 1.4087 - val_accuracy:
Epoch 4/10
10/10 [==============================] - 278s 29s/step - loss: 0.5804 - accuracy: 0.8431 - val_loss: 1.5391 - val_accuracy:
Epoch 5/10
10/10 [==============================] - 270s 28s/step - loss: 0.4357 - accuracy: 0.9248 - val_loss: 1.7343 - val_accuracy:
Epoch 6/10
10/10 [==============================] - 288s 28s/step - loss: 0.3204 - accuracy: 0.9477 - val_loss: 1.5672 - val_accuracy:
Epoch 7/10
10/10 [==============================] - 250s 26s/step - loss: 0.2226 - accuracy: 0.9967 - val_loss: 1.4316 - val_accuracy:
Epoch 8/10
10/10 [==============================] - 268s 28s/step - loss: 0.1936 - accuracy: 0.9935 - val_loss: 1.6765 - val_accuracy:
Epoch 9/10
10/10 [==============================] - 269s 28s/step - loss: 0.1321 - accuracy: 0.9967 - val_loss: 1.6150 - val_accuracy:
Epoch 10/10
10/10 [==============================] - 247s 25s/step - loss: 0.0950 - accuracy: 1.0000 - val_loss: 1.6777 - val_accuracy:
<keras.src.callbacks.History at 0x788dabf1e380>
```

Рис. 16. Компиляция и обучение модели нейронной сети

На рисунке 6 показана оценка точности модели на тестовой выборке. С помощью метода evaluate вычитается оценка производительности модели на тестовых данных. testX представляет тестовые данные (изображения) и testY - соответствующие им метки (классы). Точность (accuracy) модели на тестовой выборке составляет 83%. Точность — это метрика, которая измеряет долю правильных прогнозов модели относительно всех прогнозов на тестовой выборке. В данном случае точность в 83% означает, что модель правильно классифицировала 83% изображений из тестовой выборки в соответствии с их метками. Это показатель хорошей производительности модели, особенно при классификации изображений.

```
# оценим точность модели
_, accuracy = model.evaluate(testX, testY)
print('Точность на тестовой выборке: %.2f' % (accuracy * 100))
```

```
4/4 [==============================] - 68s 15s/step - loss: 1.6777 - accuracy: 0.8325
Точность на тестовой выборке: 83.25
```

Рис. 17. Оценка точности модели

2. Разработка интерфейса

Для мобильного приложения был разработан интерфейс самого приложения для платформы Android [5]. Данное мобильное приложение имеет множество возможностей для пользователя. Они изображены на Use Case диаграмме на рисунке 7 [6]. Целью данного мобильного приложения является предоставление возможности пользователю добавлять фото с лицом, подбирать прическу, окрас и просмотр рекомендуемых причесок по типу овала лица.

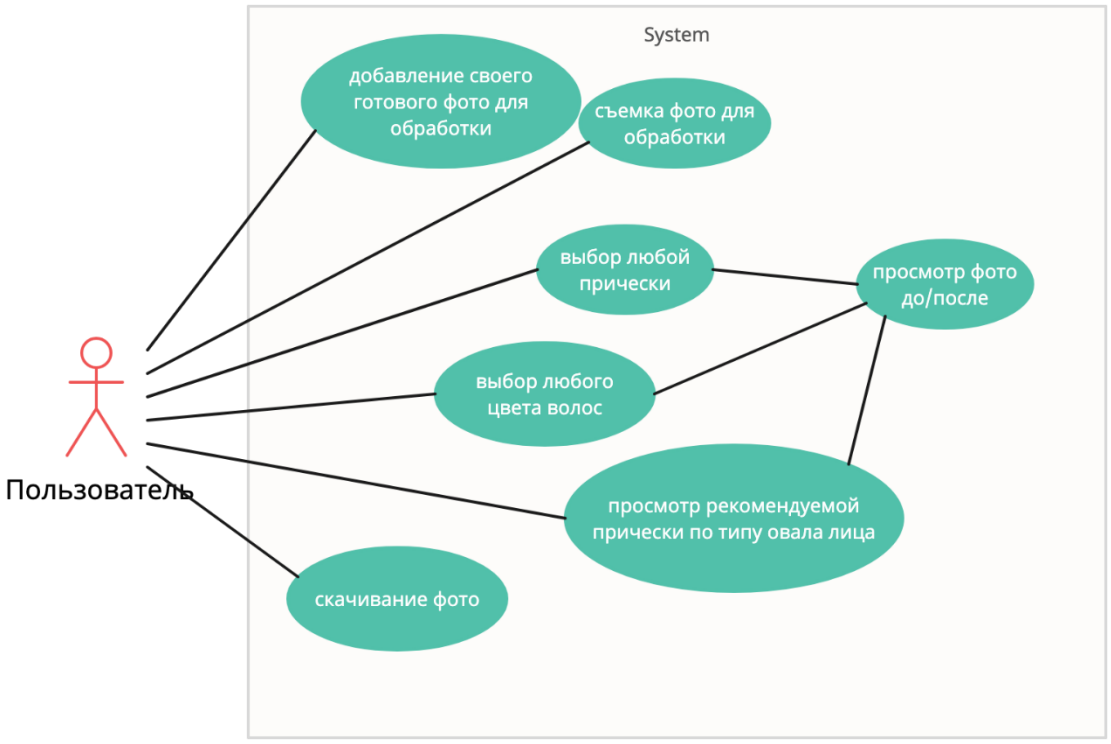


Рис. 18. Use Case мобильного приложения

Главная страница показана на рисунке 8. Здесь имеется две кнопки, дающие возможность пользователю добавить готовое фото или сделать новое.

Рис. 19. Главная страница мобильного приложения

После добавления фото пользователь переходит на страницу, которая показана на рисунке 9, где может выбрать любую прическу и окрас, а также, просмотреть рекомендуемые прически, подобранные приложением по типу овала лица. К тому же, пользователь имеет возможность посмотреть фото до/после с помощью ползунка, и понять, идет ли ему та или иная прическа и окрас.

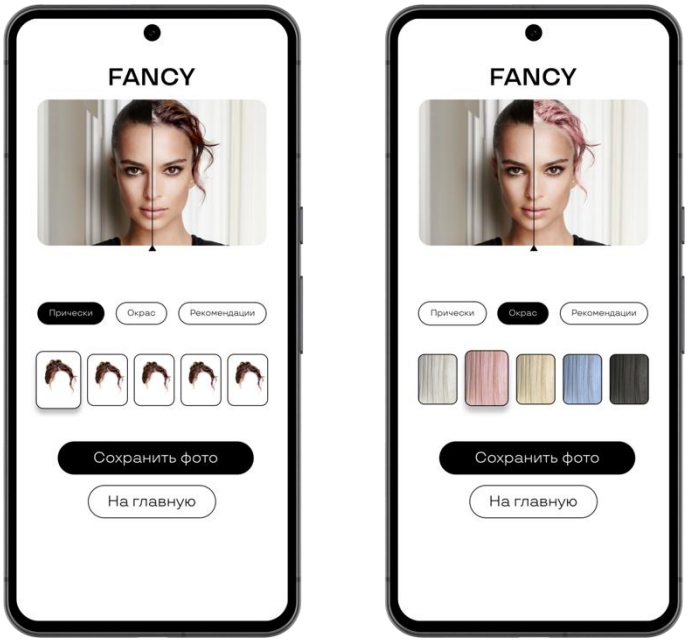


Рис. 20. Страница с выбором прически и окраса

Заключение

В результате проделанной работы была создана и обучена нейронная сеть для определения типа овала лица на большом наборе данных. Реализация данного метода может

быть использована в мобильном приложении для рекомендательной системы по подбору прически. Более того, в процессе был разработан пользовательский интерфейс для Android платформы. Стоит отметить, что реализованный интерфейс для мобильного приложения довольно простой, поэтому любой пользователь может воспользоваться им.

Литература

1. Сверточная нейронная сеть. – Режим доступа: https://habr.com/ru/articles/348000/
2. OpenCV в Python. – Режим доступа: https://habr.com/ru/articles/519454/
3. Face Recognition A Convolutional Neural Network Approach [Электронный ресурс] – Режим доступа: https://clgiles.ist.psu.edu/papers/IEEE.TNN.face.recognition.hybrid.nn.pdf
4. Sklearn.preprocessing.LabelBinarizer. – Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelBinarizer.html
5. Разработка мобильного приложения в Android Studio – Режим доступа: https://nationalteam.worldskills.ru/skills/razrabotka-mobilnogo-prilozheniya-v-android-studio/
6. Использование диаграммы вариантов использования UML при проектировании программного обеспечения – Режим доступа: https://habr.com/ru/articles/566218/

УДК 004.75

РАЗРАБОТКА ЧАТ–БОТА ДЛЯ СОЦИАЛЬНЫХ СЕТЕЙ НА ОСНОВЕ АГЕНТ–ОРИЕНТИРОВАННОГО ПОДХОДА

М. А. Принев

Воронежский государственный университет

Введение

Агент–ориентированный подход к разработке цифровых продуктов является перспективным инструментом для организации динамических взаимодействий и информационных интеграций в интеллектуальных системах. При таком подходе обеспечивается гибкая декомпозиция общей задачи, при которой происходит разбиение на подзадачи, каждая из которых выполняется отдельным агентом–модулем. Агенты находятся во взаимодействии друг с другом, обмениваясь сообщениями в асинхронном режиме. Таким образом, обеспечивается распределение нагрузки и многопоточность работы всей системы, что повышает надежность и быстродействие.

Многоагентное моделирование относится к современным информационным технологиям, которые используются для реализации свойства интеллектуальности информационных систем [1–3]. Агенты–модули способны решать некоторые задачи на уровне человека, то есть система становится адаптированной к выполнению целей, которые требуют наличия встроенного интеллекта. В этом случае многоагентная система (МАС) выполняет функции распределенного интеллекта. В аналитическом прогнозе Gartner на 2020 год среди десяти главных тенденций в области обработки данных и аналитики на первом месте указан тренд «Искусственный интеллект станет распределенным и «ответственным» [4], следовательно, разработки, основанные на агент–ориентированном подходе, будут являться перспективными и востребованными в обществе.

Один из известных примеров агент–ориентированного подхода — это интернет–боты, имитирующие активность человека в Интернете. В общем случае агенты–модули способны получить задачу, вступить в коммуникацию с другими агентами, получить от них недостающую дополнительную информацию и с ее помощью выполнить поставленную пользователем задачу [5-7]. Агенты должны обладать такими свойствами, как автономность (способность действовать независимо от пользователя), адаптивность (способность гибко реагировать на возникшие обстоятельства) и коммуникативность (способность к коммуникациям с пользователями и другими агентами). В настоящее время одним из актуальных инструментов взаимодействия с пользователями являются чат–боты в социальных сетях. Их использование позволяет вести активный диалог с пользователями и решать поставленные задачи без привлечения людских ресурсов компании, что приводит к уменьшению трудозатрат.

На основе агент–ориентированного подхода автором разработан концепт Универсальной Медицинской Информационной Системы (УМИС) удаленного мониторинга с возможностью обработки телеметрических показателей для использования в социальной сети ВКОНТАКТЕ [7–10].

1. Техническое описание УМИС

Скриншот группы УМИС в социальной сети ВКОНТАКТЕ для тестирования разработанного интеллектуального чат–бота представлен на рис. 1.

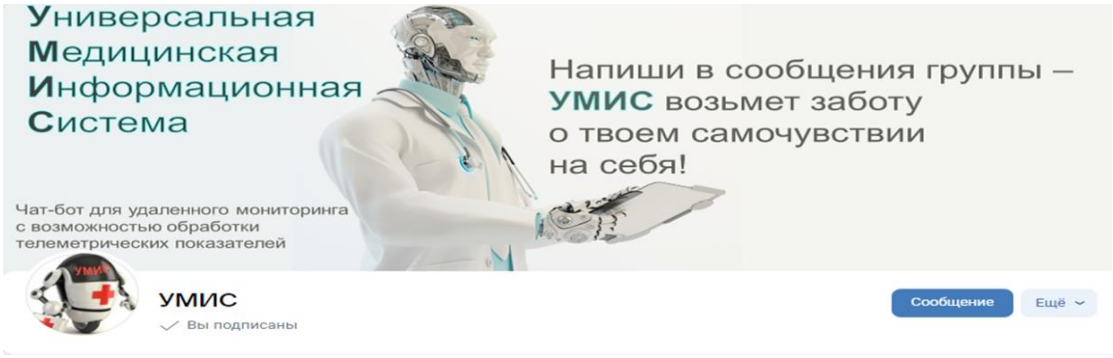


Рис. 1. Скриншот группы УМИС

УМИС предназначена для реализации следующих функций:

- сбор показателей самочувствия пациента посредством удаленного мониторинга, в том числе телеметрических показателей, полученных при помощи различных датчиков;
- автоматическое получение пациентом рекомендаций в соответствии с показателями удаленного мониторинга;
- автоматическое оповещение врача при возникновении критических ситуаций;
- возможность прямого диалога с лечащим врачом;
- возможность редактирования контента оператором при помощи специализированного интерфейса;
- возможность сохранения данных удаленного мониторинга каждого пациента;
- возможность редактирования, загрузки и хранения профиля пользователя;
- возможность сохранения и загрузки токенов подключения;
- резервный способ хранения данных в формате системы файлов.

Принцип работы УМИС заключается в осуществлении оперативного и продуктивного взаимодействия пациента с лечащим врачом в дистанционном формате посредством чат–бота в социальной сети ВКОНТАКТЕ, оснащенного специализированными функциями, а именно:

- функция получения и сохранения данных удаленного мониторинга;
- функция гибкой настройки контента, содержащегося в теле бота;
- функция вариабельности использования телеметрических данных, полученных при помощи датчиков;
- функция оптимизации и утилизации ресурсов.

Универсальность системы заключается в возможности использовать чат–бот для удаленного мониторинга заболеваний различного профиля. Это становится возможным благодаря наличию в УМИС специализированного интерфейса, позволяющего оператору осуществлять гибкую настройку и редактировать контент тела бота в соответствии с профилем заболевания. Например, при использовании УМИС для послеоперационного телемониторинга при паллиативной ЧЧХС (Чрескожной чреспеченочной холангиостомии) [11] оператор может в соответствии с характеристиками заболевания настроить рекомендации, количество и содержание контролируемых параметров самочувствия пациента, а также телеметрических показателей, полученных при помощи датчиков. А в случае мониторинга пациентов с сахарным диабетом содержание контента в теле бота будет уже настраиваться в соответствии с характеристиками этого заболевания. Все данные, полученные в процессе работы чат–бота сохраняются в базе данных, что позволяет осуществлять их последующую статистическую обработку.

Алгоритм работы чат-бота представлен на рис. 2.

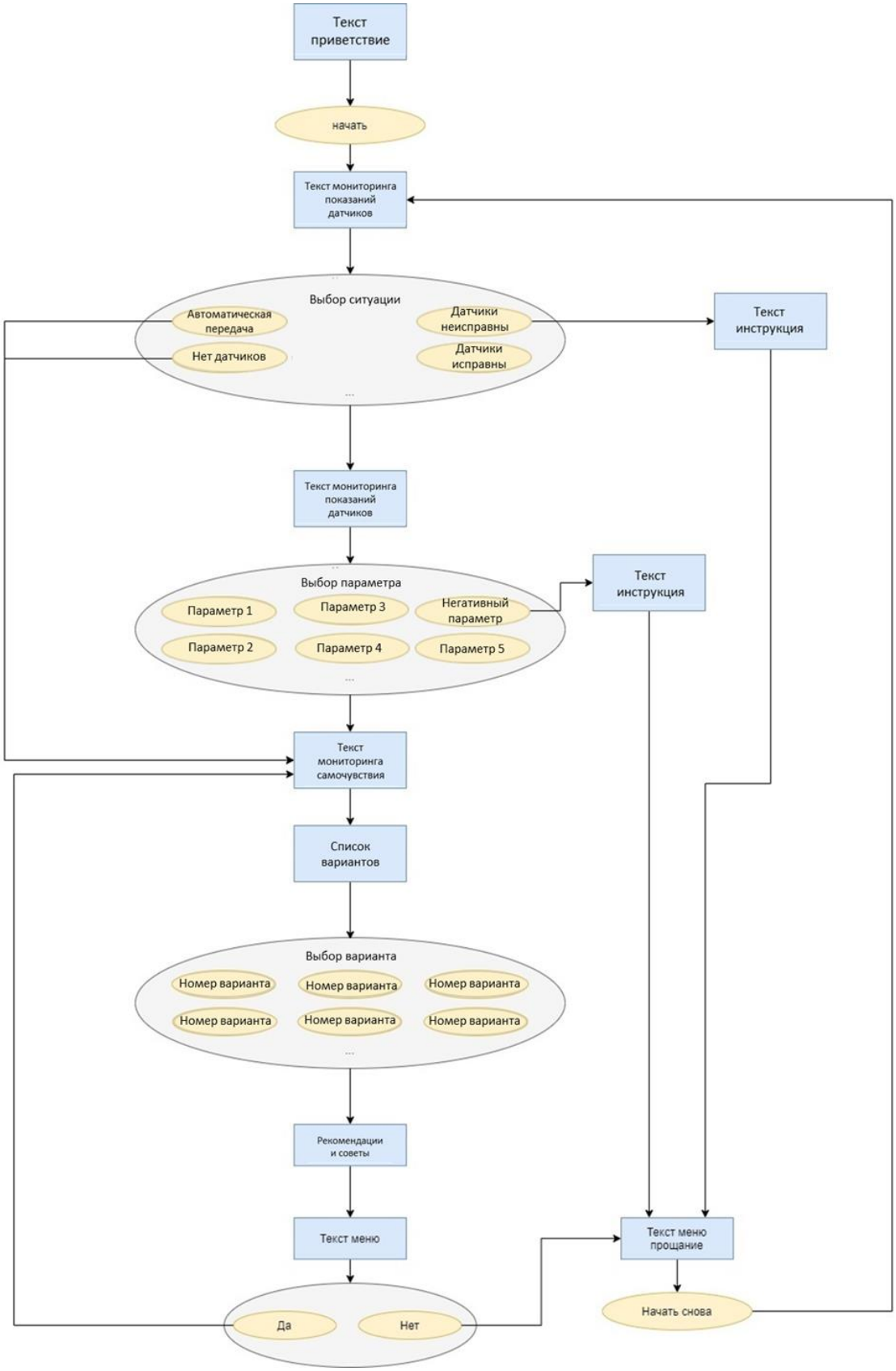


Рис. 2. Алгоритм работы чат–бота

Структура алгоритма позволяет за счет выбора нужной ситуации по наличию, уровню автоматизации и работоспособности телеметрических датчиков использовать УМИС для различных категорий пациентов, как имеющих датчики с автоматической отправкой данных, так и вынужденных передавать показания датчиков вручную или вовсе не пользующихся датчиками. В случае поломки датчика, чат-бот отправляет оповещение лечащему врачу для оперативного решения проблемы. Также врач получает оповещение, если показатели самочувствия пациента требуют срочного принятия решения. В алгоритме предусмотрена возможность пройти процедуру телемониторинга повторно в случае, если пациент допустил неправильный выбор параметра. Для сокращения времени, которое пациент тратит на процедуру удаленного мониторинга, ему не нужно самому выбирать параметры по каждому пункту опросника, достаточно выбрать вариант, соответствующий его самочувствию.

Чат-бот оснащен функцией оптимизации и утилизации ресурсов, что обеспечивает его надежную работу в условиях пиковых нагрузок и позволяет обрабатывать большое количество одновременно поступающих данных.

Инструментальные средства и технологии реализации УМИС включают: язык программирования Python, библиотеку vk_api, библиотеку PIT, командный интерпретатор для создания скриптов Unix shell («sh»), операционная система – Windows или Linux. База данных УМИС реализована на СУБД MYSQL.

УМИС разработан на основе агент-ориентированного подхода и имеет модульную структуру. Структура бота состоит из следующих агентов–модулей:

- *FileQ.py* — модуль отвечает за работу с файлами вопросов и ответов;
- *DbSevice.py* — модуль, ответственный за работу с БД;
- *AdminPanel.py* — модуль для отображения пользовательского интерфейса и настройки работы бота;
- *InstallDB.py* — модуль для первоначальной настройки БД при первом запуске;
- *KeyBoard.py* — модуль для отображения клавиатуры в ВК;
- *ListActiveTH.py* — модуль менеджера потоков;
- *RunServer.py* — модуль запуска диалога с пользователем в отдельном потоке;
- *Server.py* — модуль управления диалогами с пользователем;
- *ServerTHW.py* — модуль диалога;
- *TextBot.py* — модуль для хранения стандартных фраз для бота.

Панель администратора позволяет производить настройку работы и подключения чат–бота, а также редактировать текст сообщений в специальном текстовом редакторе.

Было проведено тестирование работы УМИС, в ходе которого производилась проверка работоспособности бота. В ходе тестирования было проведено несколько нагрузочных сессий с максимальной нагрузкой на чат–бота, причем в каждой сессии происходило увеличение ее продолжительности, количества пользователей, общающихся с ботом, а также количества одновременных диалогов. В каждой сессии оценивалось количество диалогов с задержкой ответа чат–бота и количество вылетов. В результате тестирования задержка ответа однократно появилась во время двадцатичетырехчасовой сессии, при количестве пользователей превышающем триста человек, при более двадцати одновременных диалогах с чат–ботом. Анализ происхождения задержки показал, что причина задержки обусловлена недостаточной мощностью серверного оборудования при тестировании, алгоритмических ошибок, вызывающих задержку ответа, обнаружено не было. В первой двадцатичетырехчасовой сессии тестирования произошел один вылет, причины которого были проанализированы. По результатам сессии была проведена коррекция и оптимизация кода, а также повторная тестовая двадцатичетырехчасовая сессия, результаты которой подтвердили, что проблема вылета устранена.

Участники тестирования оставили отзывы, о впечатлениях после общения с чат–ботом. По результатам отзывов, положительно оценили результат общения 83% участников, 57% оставивших отзывы воспользовались бы услугами чат–бота в случае реального заболевания, 8% респондентов оставили пожелания по коррекции контента и интерфейса чат–бота.

Результаты тестирования работы чат-бота в рабочих условиях представлены на рис. 3.

Рис. 3. Результаты тестирования УМИС

Заключение

Использование агент-ориентированного подхода при разработке интеллектуального чат-бота позволило обеспечить стрессоустойчивость бота при пиковых нагрузках за счет декомпозиции задач, многопоточности и распределенной нагрузки. Разработанный чат–бот является универсальным цифровым продуктом, благодаря имеющемуся настроечному интерфейсу, позволяющему редактировать контентное наполнение и функции бота.

Литература

1. Городецкий В. И. Современное состояние и перспективы индустриальных применений многоагентных систем / В. И. Городецкий, О. Л. Бухвалов, П. О. Скобелев // Управление большими системами: сборник трудов. – 2017. – № 66. – С. 94–157. – EDN YZAKOV.
2. Гладков, Л. А. Эволюционное проектирование многоагентных систем / Л. А. Гладков, Н. В. Гладкова // Известия ЮФУ. Технические науки. – 2021. – № 4(221). – С. 51-61. – DOI 10.18522/2311-3103-2021-4-51-61. – EDN ZBIIIK.

3. Julian V, Botti V. Multi-Agent Systems // Applied Sciences. – 2019. – 9(7). – 1402. – Режим доступа: https://doi.org/10.3390/app9071402 (дата обращения 20.03 2024).
4. Gartner Top 10 Strategic Technology Trends For 2020. – Режим доступа: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020 (дата обращения: 25.03.2024).
5. Принев, М. А. Проблемы выбора и исследования эффективности способа интеграции распределенных систем / М. А. Принев // Актуальные проблемы прикладной математики, информатики и механики : сборник трудов Международной научной конференции, Воронеж, 12–14 декабря 2022 года / Воронежский государственный университет. – Воронеж: Научно-исследовательские публикации, 2023. – С. 889-893. – EDN UXKLWG.
6. Умная обработка процессов. – Режим доступа: https://www.tadviser.ru/index.php/Статья:Умная_обработка_процессов (дата обращения: 27.03.2024).
7. Хоп Г., Вульф Б. Шаблоны интеграции корпоративных приложений.: Пер. с англ. - М.: ООО "И.Д. Вильямс", 2007. - 672 с. ISBN: 978-5-8459-1146-9 EDN: QMQSTX.
8. VK Разработчикам. – Режим доступа: https://vk.com/dev (дата обращения: 25.03.2024).
9. Документация по Python. – Режим доступа: https://www.python.org/ (дата обращения: 27.03.2024).
10. Федеральный закон «О персональных данных» от 27.07.2006 N 152-ФЗ (последняя редакция). – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_61801/ (дата обращения: 25.03.2024).
11. Боровский, С. П. Чрескожная чреспеченочная холецистостомия в лечении острого холецистита у больных с высоким операционным риском / С. П. Боровский, П. К. Собянина, А. А. Сергеева // Аллея науки. – 2018. – Т. 6, № 10(26). – С. 520-525. – EDN VRFNEY.

УДК 004.8

РАСПОЗНАВАНИЕ ЯЗЫКА ЖЕСТОВ С ПОМОЩЬЮ МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ

В. В. Приходько

Воронежский государственный университет

Введение

Язык жестов – это самостоятельный язык, состоящий из жестов, каждый из которых производится в основном движениями рук в сочетании с мимикой и положением других частей тела.

Язык жестов применим во многих сферах жизни, но в основном он применяется тогда, когда невозможно общение с использованием обычной человеческой речи. Самым частым случаем применения языка жестов является общение с глухим или слабослышащим человеком. Однако в этом случае может возникнуть множество сложностей, так как далеко не каждый человек знает жестовый язык, да и сами жестовые языки бывают разные и могут отличаться друг от друга.

В настоящее время проблему непонимания жестового языка может решить сурдопереводчик, однако его услуги являются платными, кроме того, это может нарушать конфиденциальность общения с человеком. Помимо этого, существует также переписка с помощью текста, но такой способ является довольно медленным и дискомфортным.

Проблему комфортной и быстрой коммуникации с глухим или слабослышащим человеком может решить разработка и реализация алгоритма, способного быстро переводить информацию с языка жестов в текст или в стандартную человеческую речь в реальном времени. Преимуществом такого подхода является скорость и комфорт в общении, а также не затрагивается конфиденциальность диалога. Реализовать такой алгоритм можно при помощи методов глубокого машинного обучения путём разработки и обучения модели, которая принимает на вход информацию на языке жестов, а затем преобразует её в результат, который будет понятен человеку, который не знает жестового языка.

Целью данной работы является выбор средств для разработки и программной реализации алгоритма распознавания языка жестов с использованием методов глубокого машинного обучения.

1. Особенности глубокого обучения

Глубокое обучение – это один из разделов машинного обучения, делающий упор на изучение последовательных слоев все более значимых представлений. В основе глубокого обучения лежит идея о многослойном представлении модели данных. Глубиной модели называют количество слоёв, которые она содержит. Глубокое обучение вовлекает в процесс огромное количество слоев, в отличии от других подходов в машинном обучении, которые зачастую подразумевают собой изучение одного – двух слоёв представления данных.

В глубоком обучении многослойные представления данных чаще всего изучаются с помощью нейронных сетей, выступающих в роли моделей. Нейронная сеть структурирована в виде слоёв, которые наложены друг на друга. Чем больше слоёв содержит модель, тем более точную и глубокую обработку информации возможно произвести. Практически каждая многослойная модель имеет следующие основные слои:

- Входной слой – слой, отвечающий за приём входного массива данных;
- Скрытый слой – основной слой модели. Зачастую состоит из множества слоёв. Именно в скрытом слое происходит обработка и преобразование входных данных;
- Выходной слой – слой, отвечающий за формирование и вывод результата, полученного в ходе обработки входных данных.

Пример получения результатов использованием алгоритма глубокого обучения, представлен на рисунке 1. В данном примере, нейронная сеть, имеющая несколько слоёв, преобразует изображение цифры для её распознавания.

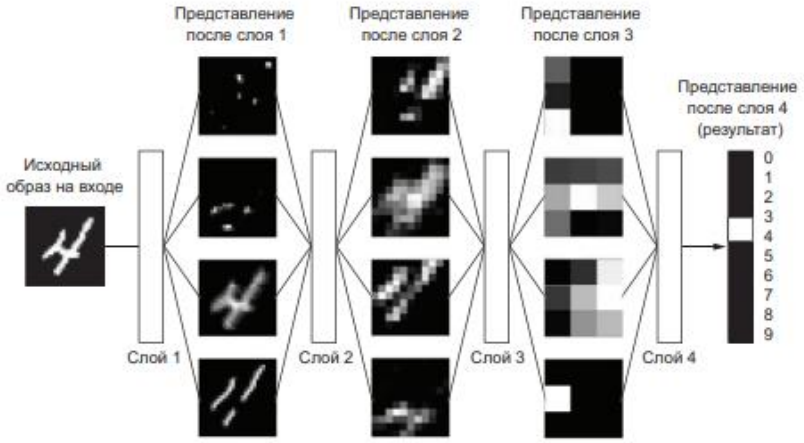


Рис 1. Пример преобразования цифры с помощью глубокого обучения

2. Этапы распознавания жестов

Существует много алгоритмов для распознавания языка жестов. Но в целом, алгоритм состоит из методов, представленных на рисунке 2.

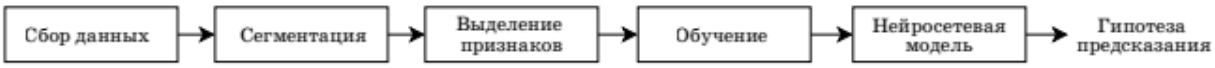


Рис 2. Этапы распознавания жестов

- На этапе сбора данных создаётся жестовой корпус, необходимый для обучения нейронной сети. Также, параллельно с этим может происходить предварительная обработка входных данных, при которой к ним применяется широкий круг цифровых преобразований (масштабирование, изменение яркости, контрастности, фильтрация и т. д.).
- Сегментация – разбиение входных данных на определённые области по ряду признаков. Сегментация может проводиться как вручную, так и автоматически.
- Выделение информативных признаков. На данном этапе происходит выделение признаков, которые в будущем будут использоваться для обучения модели. Данные признаки прямо влияют на обучение модели, так как именно из них будут формироваться входные данные для обучения нейронной сети. Выделение нужных для обучения признаков происходит с использованием разных методов, которые базируются на разных типах данных.

- Далее происходит выбор модели и её последующее обучение с помощью методов глубокого обучения.

3. Использование нейронных сетей для определения жестов

Для определения жестов лучше всего подойдут следующие типы нейронных сетей: свёрточная нейронная сеть и нейронная сеть с долгой краткосрочной памятью. Свёрточные нейронные сети специализируются на обработке изображений и видео. Для обучения таких сетей используется метод обратного распространения ошибки для корректировки значений параметров сети в процессе обучения. В свёрточной нейронной сети присутствует локальное распределение весов, что снижает сложность модели. Чем выше уровень свёртки, тем качественнее извлекаемые в процессе обучения признаки. В отличии от полносвязной нейронной сети, в сверточной сети каждый нейрон текущего слоя не связан со всеми нейронами предыдущего слоя, а связан лишь с конкретным набором нейронов, что позволяет уменьшить сложность модели. Свёрточная нейронная сеть состоит из следующих основных слоёв:

- Слой свёртки – основной слой свёрточной нейронной сети. В каждом слое свёрточной нейронной сети каждый нейрон связан лишь с определённым набором нейронов из предыдущего слоя. В процессе свёртки область изображения уменьшается, и процесс свертки применяется уже к новой полученной области, затем данная операция повторяется определённое количество раз, в зависимости от количества слоёв. Таким образом, сеть способна классифицировать изображение до малейших деталей. При каждой последующей свёртке можно получать из изображений всё более точные признаки. В процессе свёртки нейронная сеть удаляет ненужную и оставляет полезную для анализа изображения информацию.
- Слой пуллинга. В этом слое происходит выборка наиболее важных признаков, которые были отобраны слоем свёртки. Для результатов пуллинга также может быть применена свёртка, причём многократно. Данные процессы нужны для построения иерархии признаков: от самых примитивных и простых до самых сложных.
- Полносвязный слой. В данном слое происходит классификация всех полученных признаков полносвязной нейронной сетью, на основе которой происходит формирование и вывод конечного результата.

Схема работы свёрточной нейронной сети представлена на рисунке 4.

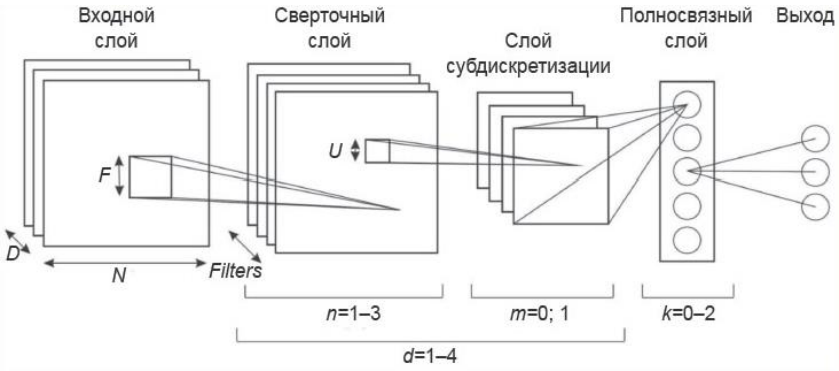


Рис 3. Схема работы свёрточной нейронной сети

Для работы с языком жестов свёрточные нейронные сети можно использовать для анализа и обработки 2D и 3D изображений жестов.

Нейронная сеть с долгой краткосрочной памятью – тип рекуррентных нейронных сетей, которые могут обучаться долгосрочным зависимостям.

Рекуррентная нейронная сеть – нейронная сеть, которая способна накапливать информацию на предыдущих примерах в процессе обучения. Такие нейронные сети хорошо подходят для обработки последовательных данных. В отличии от полносвязных и свёрточных нейронных сетей, рекуррентные сети имеют зависимые друг от друга входы. Таким образом, обучение сети зависит не только от состояния обучения на текущем примере, но и учитывает состояния на предыдущих примерах. Другими словами, рекуррентная нейронная сеть может обладать памятью и способна решать текущие задачи, основываясь на результатах предыдущих задач.

Проблема рекуррентных нейронных сетей заключается в том, что имеется разрыв между предыдущей информацией и точкой, в которой она нужна. Например, когда требуется вспомнить только последнюю информацию (к примеру, назвать следующее слово, основываясь на предыдущих), сеть справляется с этой задачей. Но со временем разрыв времени всё больше увеличивается, и сеть начинает терять связь между информацией, на которой было произведено обучение.

Проблему долгосрочных зависимостей решает нейронная сеть с долгой краткосрочной памятью – нейронная сеть, которая способна обучаться долгосрочным зависимостям. Данные сети так же являются рекуррентными, и способны решать текущие задачи, основываясь на предыдущих, но в отличии от обычных рекуррентных сетей, они способны запоминать информацию в течение длительного периода времени. Схема работы сети представлена на рисунке 5.

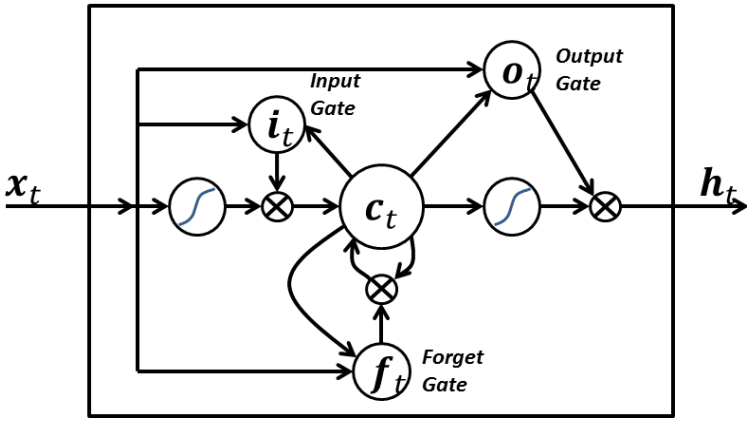


Рис. 4. Схема работы нейронной сети с долгой краткосрочной памятью

На рисунке представлена схема работы блока нейронной сети с долгой краткосрочной памятью, где x_t- входной вектор, h_t – выходной вектор, c_t - вектор состояний, которые меняются в зависимости от входных данных. В целом, данная нейронная сеть работает по следующему алгоритму:

1. Сначала определяется, какая информация будет удалена из состояния ячейки (f_t);
2. На следующем этапе нейронная сеть определяет, какая новая информация будет записана в ячейку (i_t);
3. Затем ячейка обновляется и переходит в новое состояние;
4. На последнем этапе происходит формирование выходного результата (o_t).

При работе с языком жестов можно использовать нейронные сети с долгой краткосрочной памятью для выявления пространственно-временных признаков. Также их можно применять в совокупности со свёрточными нейронными сетями.

4. Алгоритм распознавания языка жестов

Процесс разработки алгоритма для определения языка жестов показан на рисунках 6, 7, 8 и 9 в нотациях описания бизнес-процессов (IDIEF0) и диаграммы потоков данных (DFD).

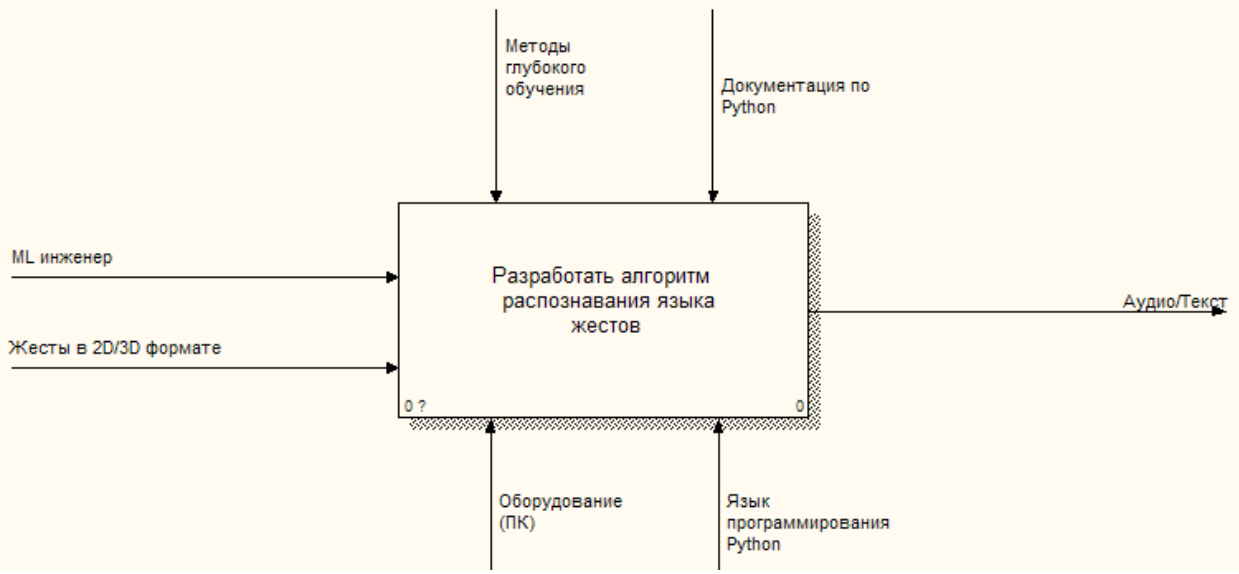


Рис 5. Схема разработки алгоритма для распознавания языка жестов в нотации IDIEF0

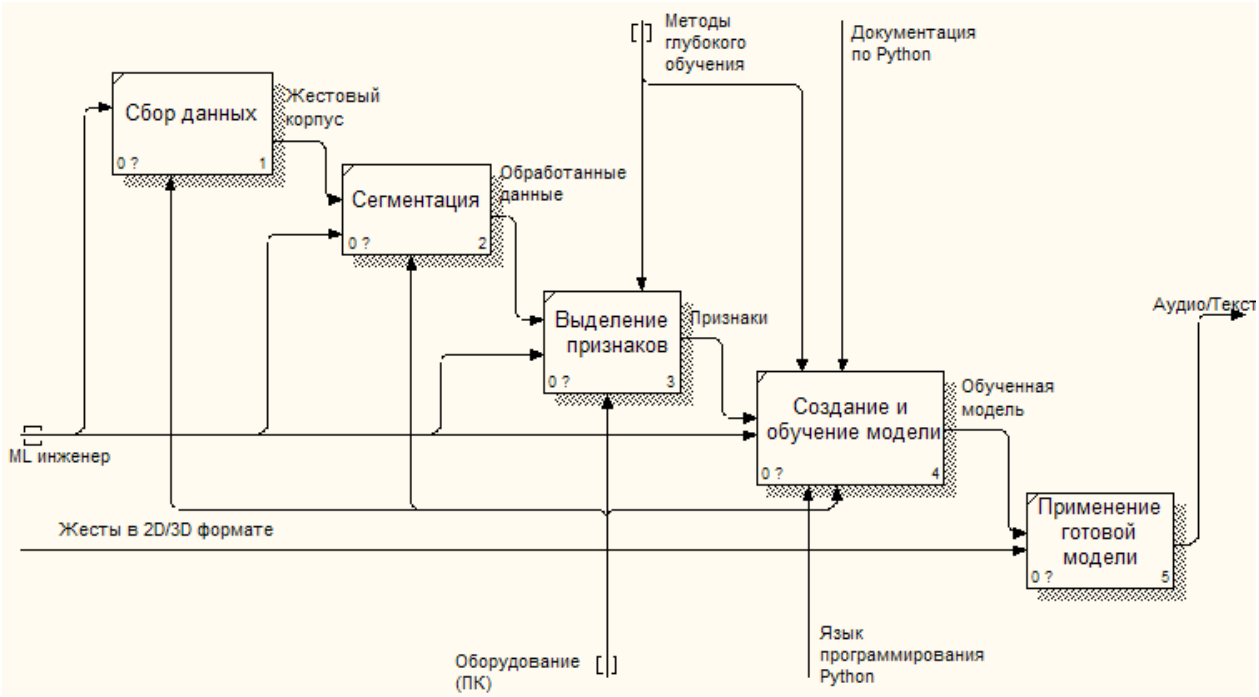


Рис 6. Схема разработки алгоритма для распознавания языка жестов в нотации IDIEF0

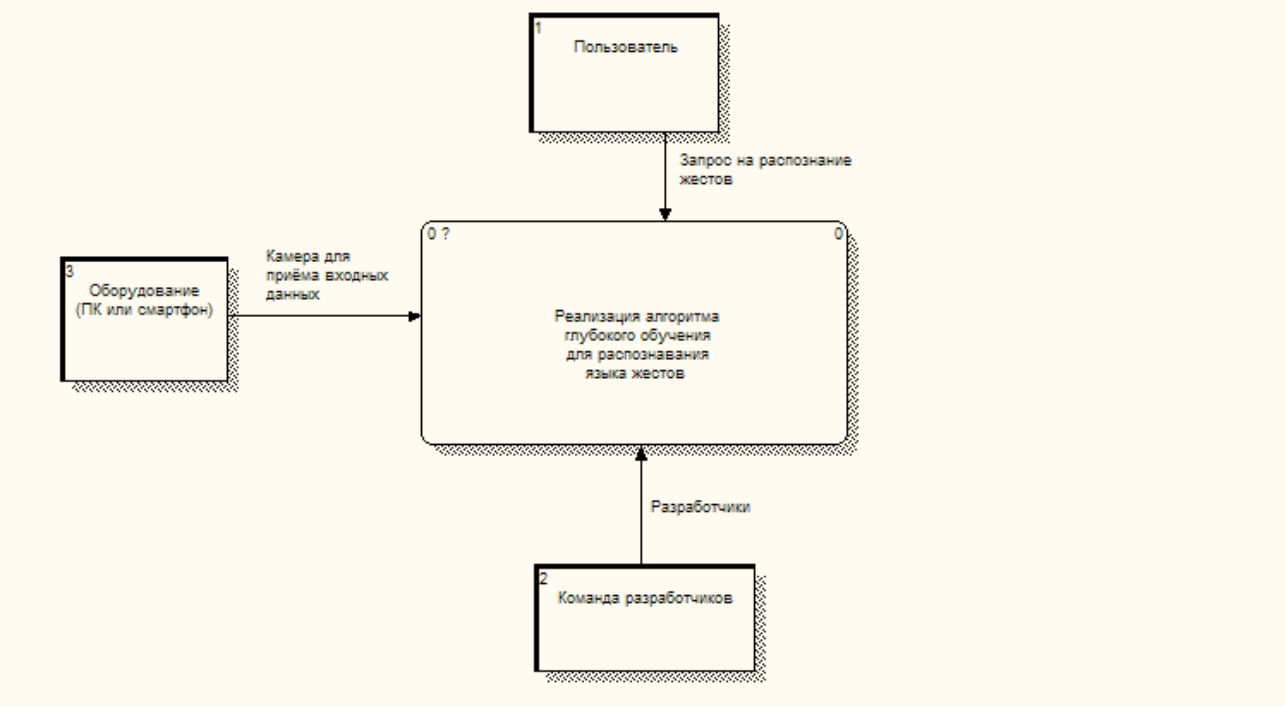


Рис 7. Схема разработки алгоритма для распознавания языка жестов в нотации DFD

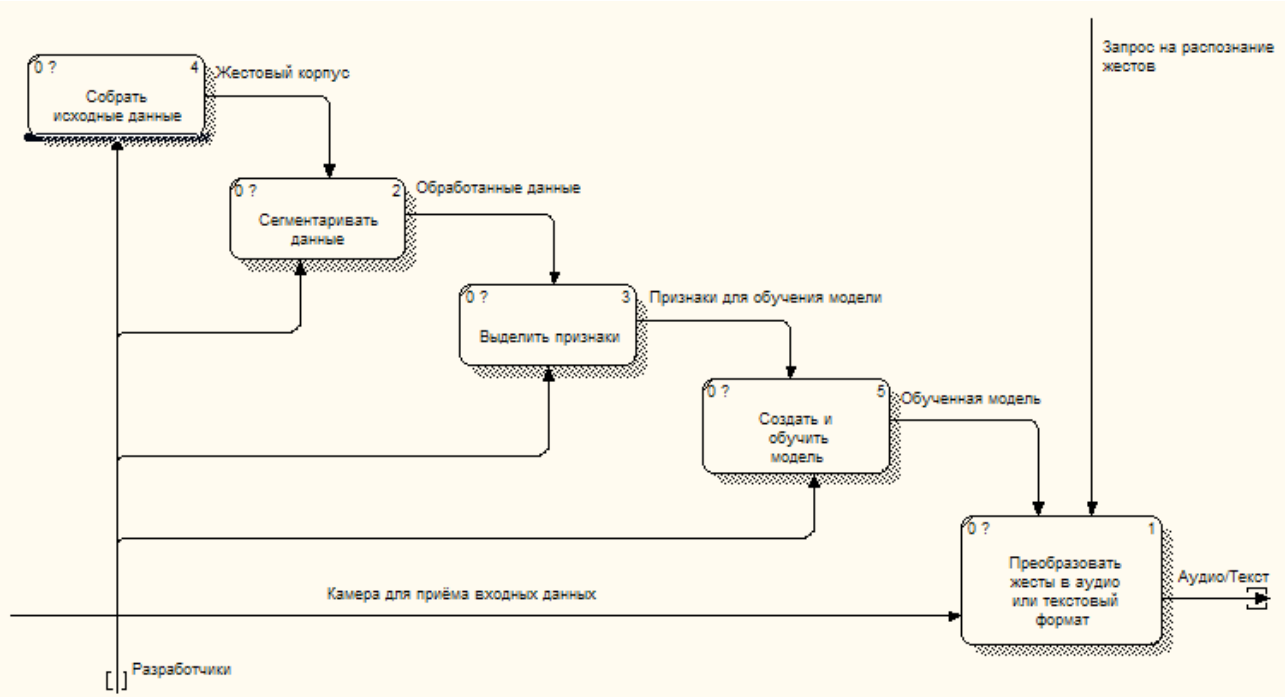


Рис 8. Схема разработки алгоритма для распознавания языка жестов в нотации DFD

Python в машинном обучении

Python является одним из самых популярных и эффективных языков программирования для использования в машинном обучении. Данный язык программирования имеет простой

синтаксис и высокую скорость исполнения кода, что позволяет направить все усилия разработчика непосредственно на машинное обучение.

Простой синтаксис языка программирования Python позволяет быстро подготовить базу для начала обучения нейронной сети. Также в Python имеется большое количество библиотек и фреймворков, которые ориентированы на машинное обучение, за счёт которых данный язык является наиболее эффективным для решения задач, связанных с машинным обучением.

Заключение

В работе был рассмотрена актуальность применения языка жестов, а также актуальность реализации алгоритма, способного преобразовывать язык жестов в аудио или текстовый формат. Также были рассмотрены этапы реализации алгоритма для распознавания языка жестов с помощью методов глубокого машинного обучения, обоснован выбор нейронный сетей для реализации алгоритма, показана схема реализации алгоритма для распознавания языка жестов.

В будущих работах планируется реализация и применение на практике собственного алгоритма распознавания языка жестов с использованием методов глубокого машинного обучения.

Литература

1. Элбон К. Машинное обучение с использованием Python. Сборник рецептов – СПб.: «БХВ - Петербург», 2019, 363 с.

2. Шолле Ф. Глубокое обучение на Python. — СПб.: Питер, 2018. — 400 с.

3. Рюмин Д. А., Кагиров И. А., Аксёнов А. А., Карпов А. А. Аналитический обзор моделей и методов автоматического распознавания жестов и жестовых языков. Информационно-управляющие системы, 2021, № 6, с. 10–20. doi:10.31799/1684- 8853-2021-6-10-20.

4. Базовый курс. – URL: https://neurohive.io/ru/osnovy-data-science/ (Дата обращения 13.04.2024).

УДК 004.94

ИСПОЛЬЗОВАНИЕ ШУМА ПЕРЛИНА ДЛЯ ГЕНЕРАЦИИ ИГРОВОГО ПОЛЯ В КОМПЬЮТЕРНЫХ ИГРАХ

А. А. Протопопов, Е. В. Трофименко

Воронежский государственный университет

Введение

Игровая индустрия не стоит на месте. Одним из актуальных направлений в её развитии является процедурная генерация миров. Данный метод принципиально превосходит генерацию, выполняемую вручную, поскольку на разработку требуется меньше времени и расходов. Размеры загружаемых файлов в этом случае обычно удается существенно уменьшить, т. к. при поставке в игре отсутствуют огромные, заранее построенные карты. Данная техника широко используется в таких популярных играх, как «Minecraft», «Terraria» или «No Man’s Sky» в основе которых лежат открытые миры.

В данной статье мы представляем прототип обучающей игры в жанре пошаговой стратегии, в которой для генерации игрового поля использовали шум Перлина.

1. Описание игры

Игроку предстоит управлять воинами, перемещая их по сгенерированному клетчатому полю с целью уничтожить всех вражеских воинов и захватить вражеские замки. Каждый ход делиться на 3 этапа:

1. Действие отрядов: перемещение по игровому полю, атака вражеских воинов, захват замков.
2. Получение ресурсов. Каждый из замков, подчиненных игроку, приносят определенное количество ресурсов, которые тратятся на следующем этапе.
3. Строительство. Игрок может потратить ресурсы покупку и усиление воинов.

Игрок побеждает, если все вражеские замки захвачены, а вражеские воины уничтожены. Если у игрока не осталось воинов и замков, игра заканчивается поражением.

Для реализации проекта использовалась среда разработки Unity. Unity— это программная среда, на основе которой конструируются и создаются игры. Разработана американской компанией Unity Technologies. Данная среда позволяет разрабатывать мобильные игры, проекты для ПК (Windows, iOS, Linux) и консолей, а также интернет приложения [1]. Основными преимуществами являются:

- доступность. Начать разработку и выпуск первых проектов можно бесплатно с тарифом Personal, он предназначен для частных лиц и небольших организаций с доходом менее $100 тыс. за 12 месяцев;
- низкий порог вхождения в разработку. В библиотеке Asset Store есть бесплатные шаблоны персонажей, звуков и фонов, которые можно использовать в первых проектах. На официальном сайте движка есть статья «How to make a game with no coding in Unity», в которой подробно описываются инструменты, необходимые новичкам для создания игр.

2. Обзор алгоритмов для генерации игрового поля

2.1. Двумерный алгоритм midpoint displacement

Рассмотрим применение данного алгоритма на одномерном отрезке (с его помощью можно, создать линию горизонта, кроме этого он используется и для двумерного случая).

Изначально произвольным образом задается высота на концах отрезка и разбивается точкой посередине на два подотрезка. Эта точка смещается на случайную величину, после чего повторяется разбиение и смещение для каждого полученного подотрезка. Случайные смещения должны быть пропорциональны длинам отрезков, на которых производятся разбиения.

Алгоритм для двумерного случая начинается с присвоения случайных высот четырем углам всей карты целиком (для удобства рассматривается квадратная карта). После чего её разбивают на 4 квадрата. В каждом из них известно значение в одном из углов. Для отыскания значений в оставшихся углах используется та же интерполяция, что и для одномерного midpoint displacement: точка в центре получается усреднением высот всех 4 угловых точек, а каждая серединная точка на стороне большого квадрата — усреднением пары точек, лежащих на концах соответствующей стороны [2]. После чего центральная точка случайным образом сдвигается вверх или вниз (в пределах, пропорциональных стороне квадрата) Процесс рекурсивно продолжается для получившихся квадратов. На рис. 1 представлена схема работы данного алгоритма.

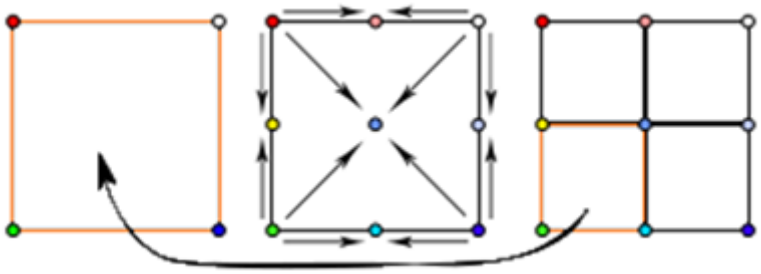


Рис. 1. Схема работы двумерного алгоритма midpoint displacement

2.2. Алгоритм diamond-square

Алгоритм diamond-square отличается от двумерного midpoint displacement тем, что состоит из двух шагов. Первый— т. н. «square» — определяет центральную точку в квадрате путем усреднения угловых и добавлением собственно случайного отклонения. Второй же шаг — «diamond» — призван определить высоту точек, лежащих на серединах сторон. Здесь(если говорить о точках на вертикальной стороне)усредняются не только две точки «сверху» и «снизу», но и пара точек «слева» и «справа», которые являются центральными и получены на шаге «square» [3]. Важно заметить, что эти две высоты, полученные на предыдущем шаге, должны быть уже посчитаны — поэтому обсчет нужно вести «слоями», сначала для всех квадратов выполнить шаг «square» — затем для всех ромбов выполнить шаг «diamond» — и перейти к меньшим квадратам. На рис. 2 представлена схема работы данного алгоритма.

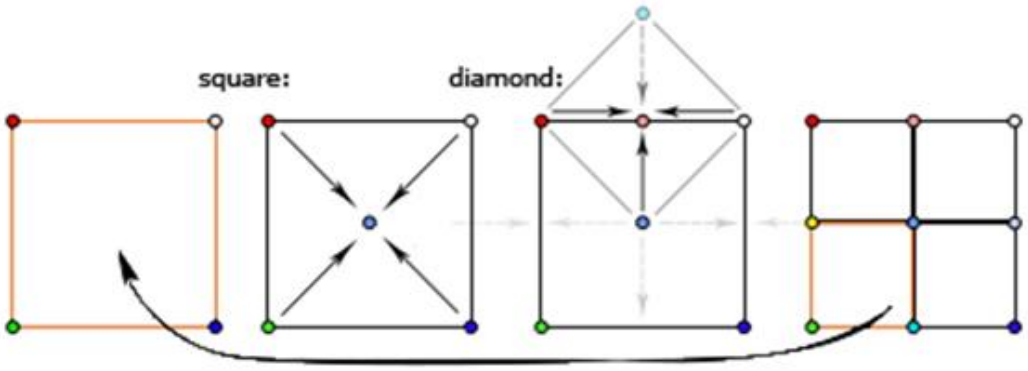


Рис. 2. Схема работы алгоритма diamond-square

На этапе «diamond» при проходе точек, лежащих на границе начальной карты, алгоритм использует высоту точек, которые находятся за пределами всей карты. В качестве решения данной проблемы можно предложить следующие варианты:

1. Считать эти высоты равными 0.
2. Представить, что плоскость свернута в тор. При попытке узнать высоту точки, лежащей на l левее левой границы карты, берётся высота точки, отстоящей на l от правой границы. Реализация: взятие координат по модулю, равному размеру карты.

2.3. Шум Перлина

Идея обычного сглаженного шума в том, что есть дискретная сетка псевдослучайных значений, и для запрашиваемой точки происходит интерполяция между узлами сетки (чем ближе точка к какому-нибудь узлу сетки, тем больше его значение соответствует значению узла).

Шум Перлина — это тип градиентного шума, разработанный Кеном Перлином. Он генерирует плавный, непрерывный шаблон значений, имеющий последовательную структуру. Данный шум широко используется для визуальных эффектов в компьютерной графике, создания естественно выглядящих ландшафтов, облаков, текстур и других органических форм.

Главная идея шума Перлина и отличие от предыдущих алгоритмов[4]:

1. В узлах сетки находятся псевдослучайные вектора (двухмерные для двухмерного шума, трехмерные для трехмерного и так далее), а не псевдослучайные числа.
2. Интерполирование между скалярными произведениями векторов от вершин квадрата до точки внутри квадрата (куба в трехмерном варианте) и псевдослучайных векторов (при описании шума Перлина их называют градиентными векторами).

В улучшенном варианте шума Кен Перлин использует всего 12 градиентных векторов. Для двухмерного варианта требуется всего 4 — по количеству граней квадрата. Вектора направлены (условно из центра куба/квадрата) в сторону каждой из граней и не нормализованы.Взятие градиентного вектора должно быть одинаковым для одинаковых целочисленных координат дискретной сетки.

3. Реализация

В качестве используемого алгоритма был выбран алгоритм шум Перлина. Основная причина выбора данного алгоритма заключается в том, что для нахождения высоты для каждой точки двумерного массива (игровая карта представляет собой поле размера n на n клеток)

достаточно одни раз пройти по каждому элементу массива (используя один вложенный цикл) в отличие от первых двух алгоритмов[5].

Основные этапы генерации карты:

1. Расстановка клеток. Каждая клетка представляет собой шестиугольную призму, размеры которой известны заранее, что позволяет расставить их в правильном порядке, не нарушая целостность карты. Размер игрового поля игрок может выбрать заранее в главном меню игры.
2. Создание карты высот. Все данные о созданных клетках хранятся в двумерном массиве, поэтому для изменения их высот необходимо сделать проход по данному массиву, применяя для каждой клетки функцию шума Перлина, которая принимает на вход 2 координаты и возвращает одно значение в диапазоне от 0 до 1. В качестве этих координат можно использовать индексы двумерного массива, однако в этом случае карта высот будет хранить одинаковые значения для каждой клетки. Чтобы это избежать каждый из индексов конкретной клетки приводится в диапазон от 0 до 1. После приведения к индексу добавляется случайное действительное число из заранее известного диапазона. Данная операция выполняется для того, чтобы избежать генерации одинаковых карт высот для одного размера карты.
3. Применение карты высот к клеткам. Полученная карта высот представляет собой двумерный массив действительных чисел в диапазоне от 0 до 1. Для удобства восприятия клетке присваивается один из трех цветов в зависимости от значения высоты. Для большей наглядности ландшафта каждой клетке задаётся значение высоты из полученной карты, умноженное на коэффициент увеличения. На рис. 3 представлено сгенерированное игровое поле размера 30 на 30 клеток с коэффициентом увеличения равным 3.5.

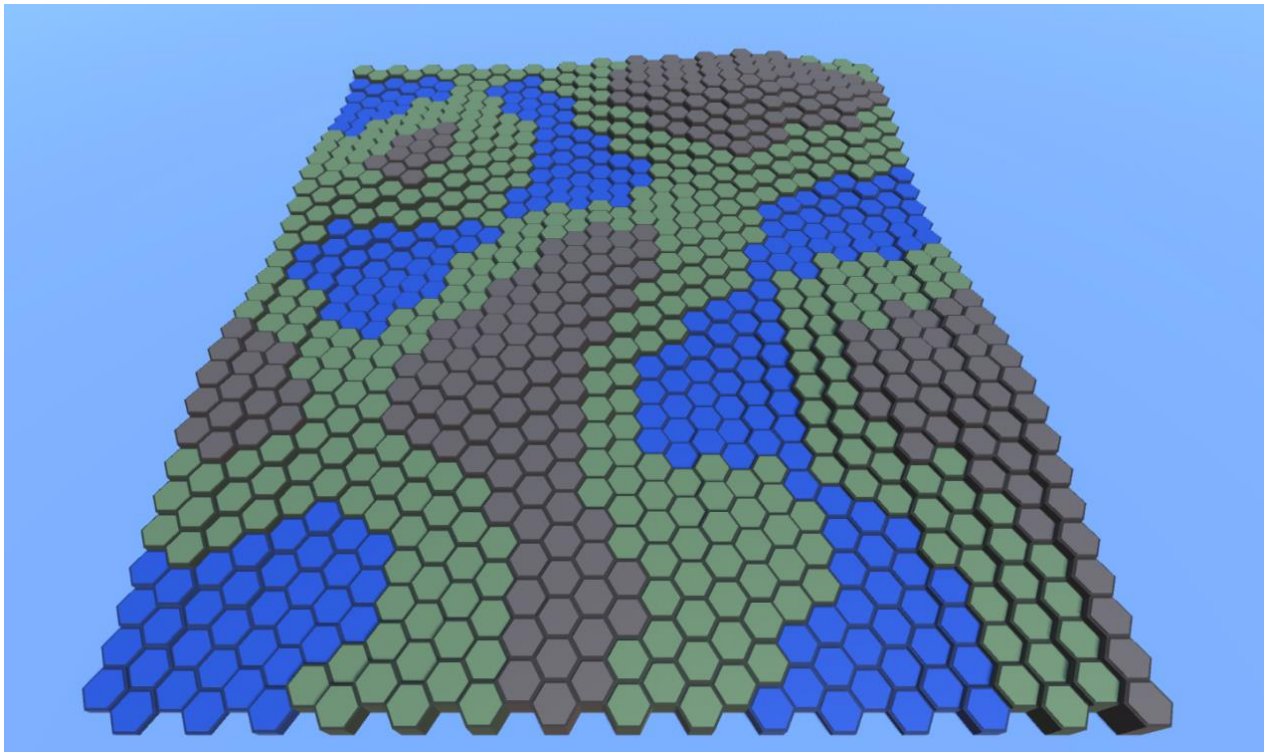


Рис. 3. сгенерированное игровое поле размера 30 на 30 клеток с коэффициентом увеличения равным 3.5

Важной особенностью игры является решение школьных математических задач разной сложности. Каждый ход на экране будет появляться окно с выбором задачи, за решение которой будет начисляться игровые ресурсы, помимо этого игрок может по своему желанию решить дополнительные задачи для усиления своих воинов.

В игре будет представлено 2 вида задач:

- тестовые, которые будут храниться в базе данных, куда они будут загружены с сайтов. Их решением будет выбор правильного ответа из предложенных вариантов;
- генерируемые. Представляют собой линейные и квадратные уравнения, линейные системы, а также геометрические задачи, которые будут генерироваться по мере необходимости.

В случае неправильного ответа, игроку будет предложено ознакомиться с соответствующей теорией и решить другую задачу текущего или более низкого уровня сложности.

У игрока будет возможность просмотреть статистку решенных и нерешенных задач. Это поможет игроку сориентироваться в своем уровне знаний.

Заключение

В данной статье был представлен прототип обучающей пошаговой стратегии с использованием шума Перлина для генерации игрового пространства. Рассмотрены концепт игры и реализация шума Перлина для данной игры, так же рассмотрены другие алгоритмы, используемые для генерации пространства. Обучающие игры могут стать важным инструментом обучения и развития школьников, а также будут полезны старшему поколению для поддержания знаний на должном уровне.

Литература

1. Первые шаги в Unity / Хабр. – Режим доступа: https://habr.com/ru/companies/otus/articles/576434/ – (Дата обращения: 20.03.2024).
2. Барсуков, А. В. Применение алгоритма midpoint-displacement в процедурной генерации ландшафтов / А. В. Барсуков // StudNet. – 2020. – № 2. – С. 469–478.
3. Воронкина, А. Ю. Моделирование процедурных ландшафтов / А. Ю. Воронкина, В. С. Шляга // Вестник Московского государственного университета печати. – 2015. – № 4. – С. 34-42.
4. Колесников, С. В. О решении задачи генерации ландшафтов / С. В. Колесников // Решетневские чтения. – 2013. – № 17. – С. 214–215.
5. Смирнова, Т. М. Моделирование игровой сцены при разработке компьютерной игры на платформе Unity3D / Т. М. Смирнова, С. Ю. Михайлов // Сборник материалов X Международной научно-практической конференции. – Москва: Общество с ограниченной ответственностью "ИРОК", 2022. – С. 260–263.

УДК 004.932

ПОСТРОЕНИЕ ЦИФРОВОГО ДВОЙНИКА РОБОТА ROIN-RTS-100

Н. С. Пугачев

Воронежский государственный университет

Введение

Возможность реализации современных алгоритмов используемых в сфере робототехники требует соответствующего уровня программного обеспечения, цель которого обеспечить отказоустойчивость, точность и воспроизводимость действий. С этими целями помогает справится открытая операционная система для роботов ROS 2[1]. Помимо алгоритмов, ROS2 представляет определенный шаблон разработки и тестирования программного обеспечения для роботов, а также предоставляет возможность создания цифрового двойника для удобной и не затратной отладки алгоритмов и получения синтетических данных, для применения систем машинного обучения

В этой статье содержится информация о проделанной работе по построению цифрового двойника робота ROIN RTS 100.

1. ROS 2 как архитектурное решение

1.1 *Структура ROS 2*

При подготовке программного обеспечения роботизированных систем, встает вопрос о том как обеспечить безопасность, отказоустойчивость и слаженную работу всех компонентов данной системы. Подходящим решением является метод организации программного кода предложенной в системе ROS 2. В ней предложено выделять различные компоненты системы в отдельные компоненты, называемые нодами. Нода может содержать в себе контроллер мотора, алгоритм, программное обеспечение необходимое для датчика или камеры и т. д.. Для обмена данными существует структура которая называется топик. Каждая нода может как писать сообщения в топик, так и подписываться на него чтобы получать сообщения которые приходят в него. Помимо нод и топиков, существуют также сервисы, позволяющие нодам общаться по принципу request response. Данный механизм позволяет существенно снизить нагрузку на обработку ненужного потока сообщений.

Есть также механизм разделения программного кода функций робота в отдельные пакеты, пакет содержит в себе описание нод. Таким образом для разных задач можно использовать комбинации разных пакетов. Причем вести разработку пакетов и их тестирование как независимо так и в совокупности с другими пакетами. Что тоже вносит определенную гибкость и ясность при добавлении новых функций или отключения неиспользуемых.

Такой образ представления роботизированной системы как мультиагентной является достаточно отказоустойчивой поскольку программист получается возможность отдельно тестировать и разрабатывать каждую ноду. Кроме того, сообщения из топиков можно хранить, чтобы в дальнейшем анализировать и корректность работу системы.

1.2 *Описание модели робота*

Для того чтобы создать цифрового двойника, в первую очередь необходимо задать его модель, учитывающую физические и геометрические характеристики робота. Для геометрической составляющей была разработана 3D модель робота, посредством конвертации CAD[3] модели в CGI[2] формат (рис. 1), с последующей ретопологией, для оптимизации количества вершин и исправления некоторых ошибок конвертации.

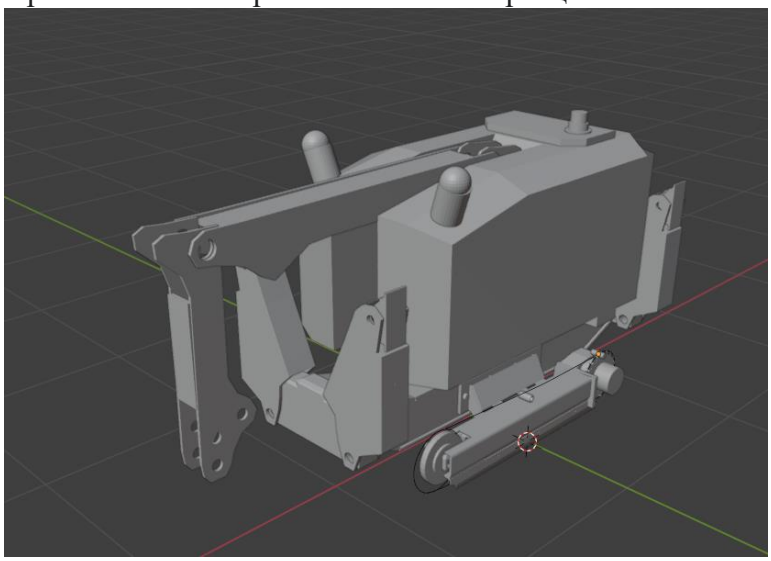


Рис. 1. Визуализация CGI модели робота в программе Blender

Далее робот был описан в формате URDF[4]. URDF (рис. 2) позволяет описывать структуру робота в виде дерева, в котором каждая составная часть обозначается как Link, а каждое сочленение как Joint. Link несет в себе информацию о геометрических, визуальных и физических данных составной части. Joint же описывает вид сочленения, его ограничения и прочность.

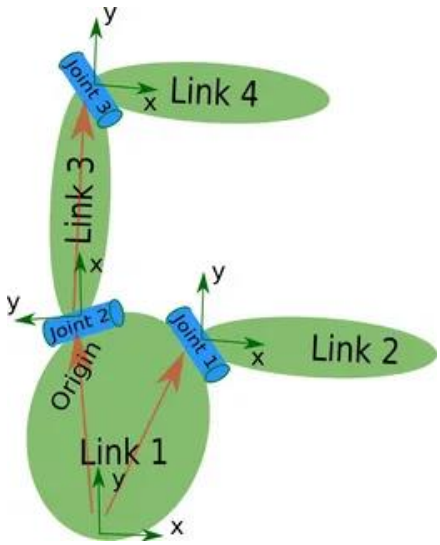


Рис. 2. Схема URDF формата

После экспорта составных частей был написан алгоритм дополнительного преобразования получившихся 3д моделей включающий в себя:

1. запоминание изначального положения центральной точки модели
2. перемещение модели в центр координат

3. расчет новых координат каждой составной части путем вычитания из текущей центральной точки конкретной центральной точки родительской сущности(будь то Joint или Link)

Результат описания структуры робота в виде дерева формата urdf представлен ниже (рис. 3).

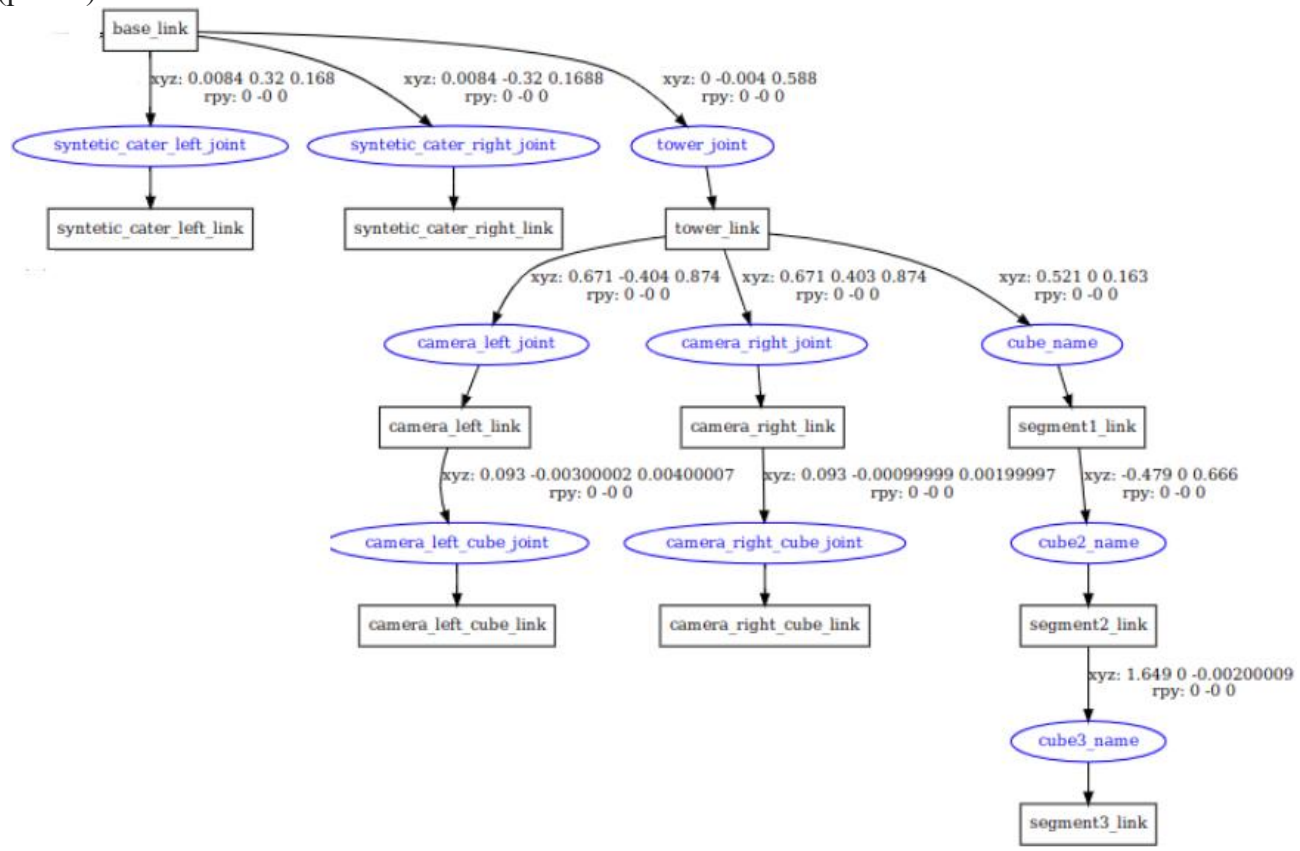


Рис. 3. Схема представления модели робота как дерева

Следует учитывать, что древовидная система не позволяет реализовывать моделирование систем закрытой кинематики. Этот аспект стоит учесть на этапе проектирования модели.

Для более удобного, наглядного и читаемого кода описания модели, был использован язык xacro. Xacro - набор макросов для работы с xml файлами, позволяющий разносить информацию по разным файлам а затем собирать их в один.

1.3 *Rviz 2*

Как только модель робота описана, появляется возможность ее визуализировать. Для этого в системе ROS 2 есть утилита под названием Rviz2[5] (рис. 4). Rviz2 универсален, так как он, по сути, подписывается на топики и подходит не только для визуализации модели робота, но также и для визуализации одометрии, цели назначения, положений составных частей и сочленений, изображений с камер и показателей с датчиков. Результаты работы Rviz 2 показаны на рис. 4.

Как только работа над моделью робота заключена, встал вопрос о выборе симуляции. Поскольку Ros 2 имеет официальную поддержку Gazebo[6], было принято решение использовать именно ее. Gazebo позволяет симулировать не только физическое но и визуальное окружение агента. Необходимо задать окружение работа в формате sdf, а затем

выполнить подгрузку робота в это окружение. Важно то, что описание робота в urdf формате подходит и для Gazebo, то есть все параметры и значения остаются неизменны.

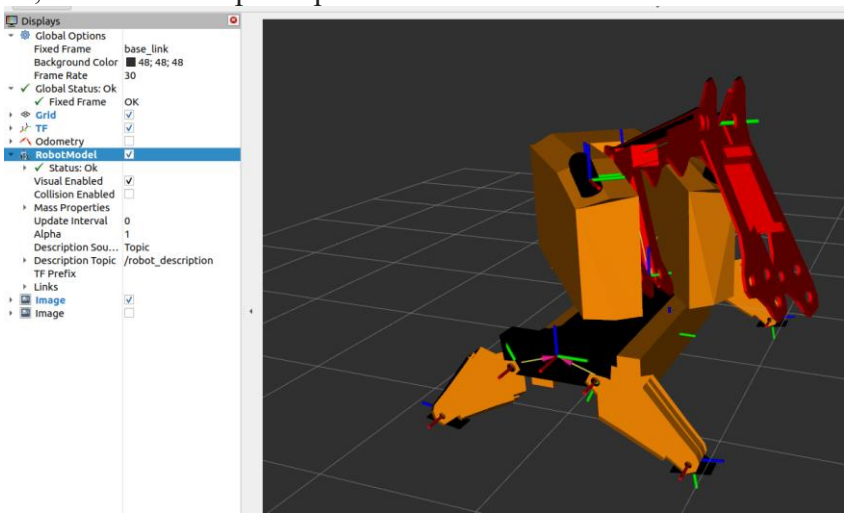


Рис. 4. Визуализация робота с помощью утилиты RVIZ 2

2. Подключение симуляции

Для описания интерфейсов контроля и сбора информации, в Gazebo представлен довольно большой набором плагинов. В данной работе были использованы:

1. TrackController;
2. Imu;
3. camerasensor;

Все они инициализированы в отдельных xacro файлах и объединяются с изначальным файлом описания робота при запуске симуляции.

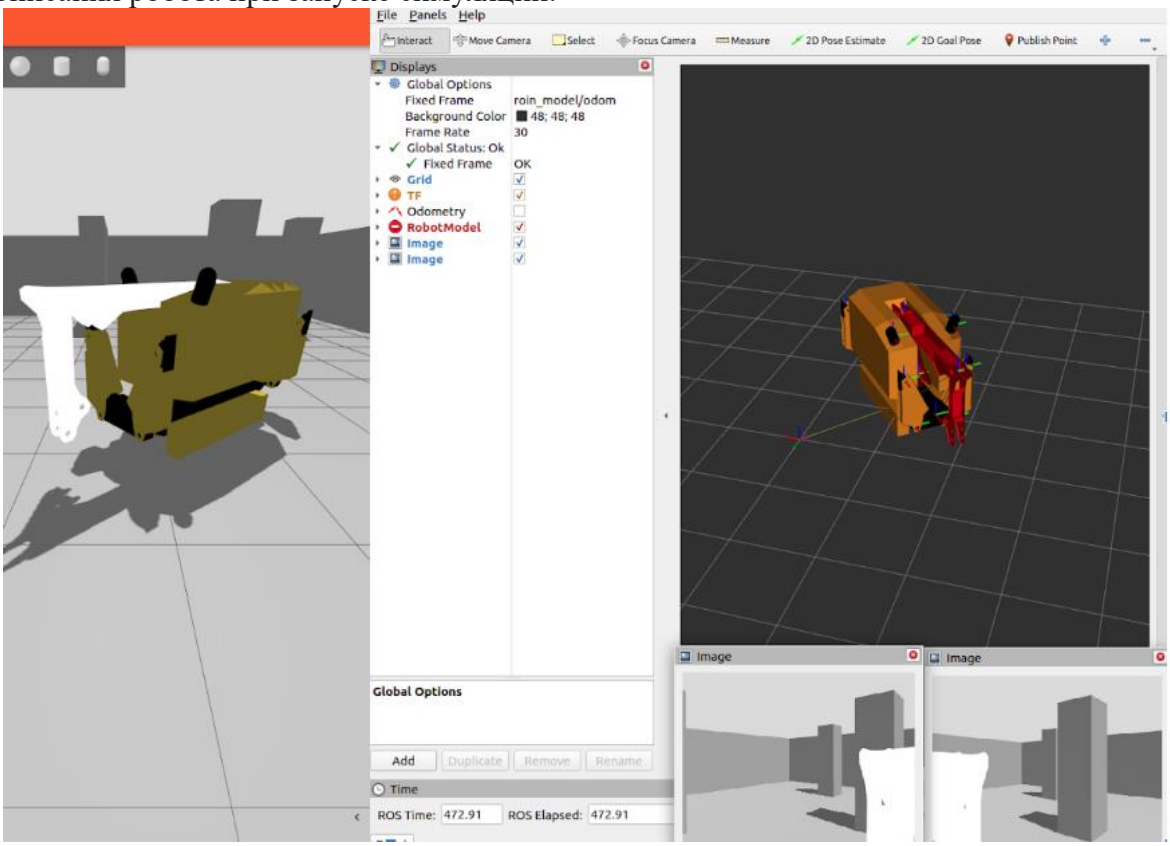


Рис. 5. Работа симуляции Gazebo и синхронной визуализацией в RVIZ 2

На рис. 5. показана совместная работа Gazebo и ROS 2. Общение между Gazebo и Ros2 осуществляется с помощью утилиты ros_gz_bridge[7], которая осуществляет конвертацию форматов сообщений топиков и обеспечивает их совместимость. Реализация такого сообщения показана на рис. 6.

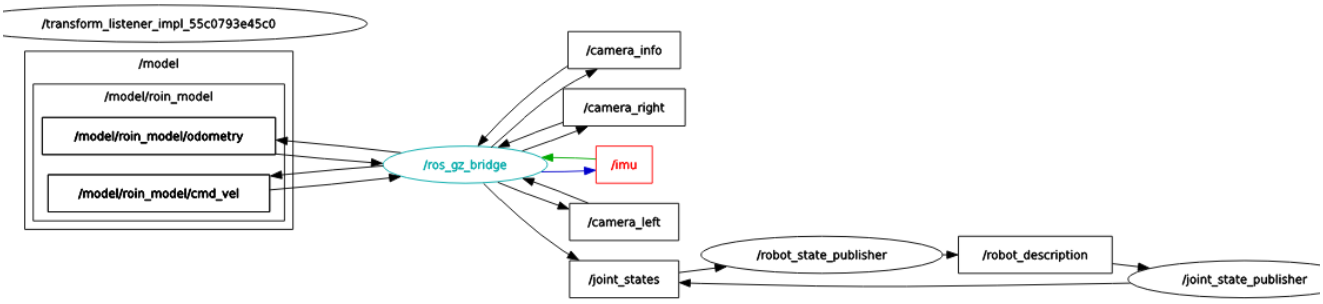


Рис. 6. Схема работы симуляции Gazebo в паре с ROS 2 через использование ros_gz_bridge

Gazebo передает в ROS 2 информацию о скоростях каждой составной части, положениях сочленений, данные с камер, данные с IMU датчика и информацию об одометрии.

Заключение

Разработанная в ходе этой работы система содержит в себе модель робота ROIN RTS 100, содержащую в себе как геометрические, так и физические свойства оригинала. Затем эта модель была добавлена в симуляцию GAZEBO. После чего был налажен мост коммуникаций между ROS 2 и симуляцией. С помощью системы плагинов Gazebo, были смоделированы датчики, камеры и двигатели, что позволяет получать данные с них и через мост коммуникаций передавать их в ROS 2. Таким образом был сформирован интерфейс симуляции модели робота принимающий на вход управляющие воздействия, а на выходе, возвращающий показания приборов. Симуляция позволяет применять и отлаживать алгоритмы навигации и собирать данные для систем машинного обучения, проводить их развертывание и тестирование. Кроме того, Ros 2 практически бесшовно подменять симулированные компоненты робота на реальные.

Литература

1. Оффициальная документация ROS 2: https://docs.ros.org/en/iron/index.html. – (Дата обращения: 01.10.2023).
2. Документация Blender: https://docs.blender.org/manual/en/latest/contribute/. – (Дата обращения: 09.2.2024).
3. Документация FreeCAD: https://wiki.freecad.org/Getting_started. – (Дата обращения: 15.11.2023).
4. Документация URDF: https://wiki.ros.org/urdf. – (Дата обращения: 05.10.2024).
5. Код Rviz2: https://github.com/ros2/rviz. – (Дата обращения: 01.01.2024).
6. Оффициальный сайт Gazebo: https://gazebosim.org/home. – (Дата обращения: 15.01.2024).
7. Код ros_gz_bridge https://github.com/gazebosim/ros_gz. – (Дата обращения: 12.03.2024).

УДК *004.05*

ОПТИМИЗАЦИЯ СБОРКИ С ИСПОЛЬЗОВАНИЕМ VITE: РАЗДЕЛЕНИЕ КОДА НА ЧАНКИ

Д.А. Путнов, С. В. Сорокин, Е.М. Аристова, И.С. Коровченко

Воронежский государственный университет

Введение

В современном мире оптимизация скорости загрузки web-страниц и приложений становится критически важной. Пользователи ожидают мгновенного отклика и непрерывной работы, и даже небольшие задержки могут негативно повлиять на пользовательский опыт и уменьшить конверсию. Кроме того, поисковые системы учитывают скорость загрузки при ранжировании web-страниц, что делает быструю загрузку важным конкурентным фактором и напрямую влияет на потенциально возможное количество посетителей web-ресурса.

Целью работы является анализ методов разделения кода и оценка их эффективности при разработке web-приложений с использованием Vite, как наиболее быстрого и современного сборщика на нынешнем рынке, в качестве основного инструмента сборки приложения.

1. Описание проблемы

Результатом сборки web-приложения с использованием Vite является следующий набор файлов:

1) index.html — основной HTML-файл, который содержит точку входа в приложение и ссылки на все необходимые стили CSS и скрипты JavaScript;
2) script.js — содержит весь JavaScript код приложения, включая код используемых в приложении зависимостей;
3) style.css — содержит все стили приложения, описанные на языке CSS.

Файл index.html содержит базовую вёрстку приложения и специальные теги для подключения CSS и/или JavaScript файлов, а также корневой элемент, в который будет монтироваться приложение посредством исполнения JavaScript кода на стороне клиента сразу после загрузки всех необходимых файлов.

Файл index.html, как правило, является неизменным по мере роста приложения. В среднем, его вес достигает не более 100 Кб. Однако, с развитием и ростом количества возможностей в приложении, файлы script.js и style.css могут увеличиваться в размере прямо пропорционально написанному в ходе процесса разработки коду. Как правило, такие файлы не имеют верхнего предела по размеру, из-за чего в крупных приложениях могут достигать веса свыше 1 Мб, что является критическим значением, так как может привести к избыточной загрузке и длительному времени начальной загрузки приложения, особенно в больших проектах.

Без разделения кода web-приложение загружает весь свой код целиком при каждой загрузке страницы.

Для решения проблемы длительной загрузки больших приложений могут использоваться два следующих подхода [1]:

1) *«ленивая» загрузка* — это техника оптимизации загрузки ресурсов в web-приложениях, при которой ресурсы загружаются только в момент их фактического

необходимости. Это означает, что ресурсы (такие как изображения, скрипты, стили и другие файлы) не загружаются при первоначальной загрузке страницы, а загружаются только тогда, когда пользователь действительно взаимодействует с элементами, требующими эти ресурсы. Для реализации «ленивой» загрузки в приложениях с использованием библиотеки React, применяется встроенная функция — React.lazy [2];

2) *разделение на чанки* — разбиение кода на отдельные файлы (чанки) в зависимости от логической структуры приложения или его функциональных блоков. Это позволяет загружать только те части приложения, которые действительно нужны пользователю в данный момент, а также организовать процесс параллельной загрузки нескольких JavaScript и CSS файлов.

При выборе способа оптимизации загрузки, обычно предпочтение отдается «ленивой» загрузке, в то время как разделение на чанки остается менее распространенным методом, хотя и является логическим продолжением первой стратегии. Сочетание обеих стратегий позволяет создать оптимальные условия сборки приложения и достичь наилучших показателей скорости его загрузки для конечных пользователей.

2. Сравнение и анализ возможных вариантов сборки

В этом разделе будет проведен сравнительный анализ использования такого способа оптимизации, как разделение на чанки, а именно, два возможных случая: сборка приложения без дополнительных настроек и сборка с разметкой чанков и организацией «ленивой» загрузки модулей.

Для примера возьмём web-приложение «Менеджмент системы для организации бизнеса с моделью интервальной аренды», разработанное с применением простой модульной архитектуры и состоящее из 5 страниц: «Страница аутентификации», «Личный кабинет пользователя», «Обзорная панель администратора системы», «Страница аудита инвентарных запасов предприятия» и «Страница контроля пользовательских сессий». Каждая страница представляет собой независимый модуль интерфейса, который может быть загружен на клиентское устройство только в случае необходимости его отображения и/или использования. Вес всего кода web-приложения без учёта библиотек составляет 631 Кб.

2.1. Без разделения

В случае без разметки чанков, мы получаем итоговый результат сборки, состоящий из трёх файлов: index.html, style.css, script.js. Таким образом, пользователь будет загружать весь JavaScript код приложения одним файлом (рис. 1):

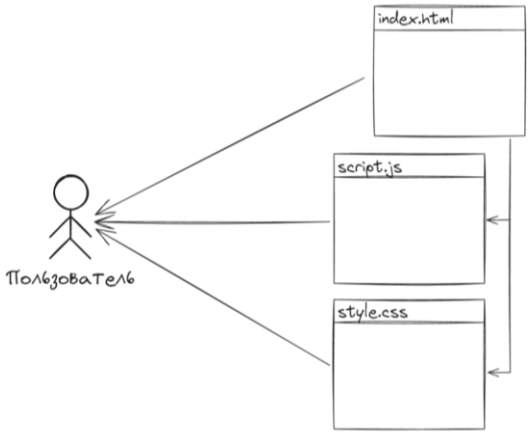


Рис. 1. Асинхронная загрузка файлов для случая без разделения кода

Общий вес собранного web-приложения — 1.2 Мб, из которых script.js — 1.01 Мб. Вес общего файла JavaScript кода увеличился относительно исходного кода, поскольку script.js также содержит все необходимые зависимости, указанные в приложении, которые ранее не учитывались в общих подсчетах. Для вычисления времени загрузки файла в работе используется формула:

$$t = \frac{S}{V}, \tag{1}$$

где t — искомое время загрузки в секундах, S — общий размер файла, V — пропускная способность канала связи за секунду времени.

Используя исходные данные, по формуле (1) получаем, что время загрузки файла script.js при скорости интернета в 512 Кбит/сек, равно 16 секундам.

2.2. С разметкой чанков

Разметка чанков в сборщике Vite осуществляется через конфигурацию rollup. В свою очередь, rollup — сборщик, который Vite использует для сборки приложения в режиме production (режиме работы, оптимизированном для реального использования, когда приложение готово к развертыванию и использованию конечными пользователями) в среде выполнения JavaScript Node.js [3].

Разбиение на чанки вручную конфигурируется свойством manualChunks [4] конфигурации rollup в Vite (рис. 2):

```
export default defineConfig(() => {
 return {
   ...
   build: {
     rollupOptions: {
       output: {
         manualChunks: {
           ...
         },
       },
     },
   },
 };
});
```

Рис. 2. Структура конфигурации в общем виде

Конфигурация представляет собой пары ключей и значений, где в качестве ключа указывается название результирующего пакета JavaScript кода, а в качестве значений — названия используемых библиотек, используемых в проекте.

Для корректного разбиения на чанки, можно руководствоваться следующими рекомендациями:

1) *логическая группировка* — разбиение зависимостей на логически связанные группы. Например, в случае с web-приложениями, разработанными с помощью библиотеки React, стоит объединить в один чанк такие зависимости как react, react-dom, react-router-dom, поскольку эти библиотеки могут работать в браузере только совместно. В том числе, крупные библиотеки или фреймворки, такие как React или Vue, которые используются в разных частях приложения, вы можете разместить их в отдельных чанках, чтобы они не загружались повторно для каждой страницы. Напротив, в отдельный чанк стоит вынести, например, библиотеку для построения графиков, поскольку она отвечает за кардинально другой функционал приложения и может быть загружена независимо отдельным пакетом, не блокируя загрузку и работу других;
2) *частотная группировка* — если некоторые части приложения используются чаще, чем другие, это стоит учитывать при разбиении на чанки. Чаще используемые части могут быть объединены в более крупные чанки для улучшения производительности и скорости загрузки, в то время как менее часто используемый функционал лучше вынести в отдельный пакет. Связано это с тем, что большая часть пользователей, вероятно, не будет использовать представленный функционал достаточно часто, чтобы загружать код для его реализации при каждом открытии страницы. Гораздо более выгодно, с точки зрения скорости загрузки и оптимизации сетевой нагрузки, загружать такие библиотеки и модули динамически по мере необходимости;
3) *группировка по размеру исходного кода* — большие библиотеки имеет смысл вынести в отдельный чанк, в то время как несколько легковесных библиотек — объединить в один. Таким образом, удастся добиться наиболее высокой скорости загрузки приложения, минимизировав затраты на создание отдельного канала связи для загрузки большого количества легковесных файлов.

Учитывая пункты, описанные выше, можем привести пример конфигурации разбиения библиотек на чанки для приложения (рис. 3):

```
export default defineConfig(() => {
  return {
    ...
    build: {
      rollupOptions: {
        output: {
          manualChunks: {
            core: ['react', 'react-dom', 'react-router-dom'],
            network: ['axios', '@tanstack/react-query', '@loadable/component'],
            utils: ['classnames', 'dayjs', 'escape-html', 'tailwind-merge', 'zustand'],
            table: ['@tanstack/react-table', '@tanstack/react-virtual'],
            chart: ['reactflow', '@dagrejs/dagre'],
            icons: ['@ant-design/icons'],
            ui: [
              '@chakra-ui/react',
              'chakra-react-select',
              'framer-motion',
              'prismjs',
              'react-simple-code-editor',
            ],
          },
        },
      },
    },
  };
});
```

Рис. 3. Пример конфигурации для тестового приложения

В результате имеем разбиение на 7 независимых чанков:

1) core — пакет библиотек, который содержит ключевой набор кода, без которого невозможна работа приложения на стороне клиента;
2) network — набор библиотек, который содержит весь необходимый код для организации работы с сетью;
3) utils — набор утилит приложения, который включает в себя как легковесные функции, легковесный пакет для работы с данными и пакет zustand для менеджмента состояний приложения;
4) table — набор библиотек, реализующих табличное представление в приложении;
5) chart — набор библиотек, реализующих представление в виде графов;
6) icons — набор иконок, используемых в приложении;
7) ui — набор компонентных библиотек, составляющих основу интерфейса.

После сборки приложения получаем следующий результат (рис. 4):

```
dist/assets/utils-rGMhL8iC.js                32.42 kB │ gzip:  11.86 kB
dist/assets/icons-XFbavIkF.js                37.42 kB │ gzip:  13.21 kB
dist/assets/table-rAZtVjmH.js                68.37 kB │ gzip:  17.68 kB
dist/assets/network-ro8zswka.js              75.65 kB │ gzip:  25.84 kB
dist/assets/chart-bGkevXzl.js               183.73 kB │ gzip:  60.84 kB
dist/assets/core-462KA6WU.js                197.80 kB │ gzip:  64.44 kB
dist/assets/ui-TefDgWFw.js                  458.21 kB │ gzip: 152.63 kB
```

Рис. 4. Результаты сборки приложения

Заметим, что каждый чанк имеет размер не более чем 500 Кб. Рассчитаем скорость загрузки приложения при открытии стартовой страницы приложения. Для отрисовки первоначальной страницы пользователю требуется загрузить чанки core и network. Итоговый размер загружаемых библиотек: 273.45 Кб.

Используя формулу (1), рассчитаем скорость загрузки необходимого объёма кода с учётом разбиения на чанки. Получим результирующую скорость загрузки, равную 4 секундам, что в 4 раза быстрее относительно замеров скорости до проведения оптимизации.

Заключение

В результате применения комбинации таких методов оптимизации загрузки, как разделение кода на чанки и «ленивая» загрузка модулей, удалось значительно сократить время загрузки приложения, а именно — начальная загрузка приложения ускорилась в 4 раза. Разбиение кода и библиотек на независимые чанки, каждый из которых содержит определенный функционал, позволило эффективно управлять загрузкой ресурсов.

Разделение кода на чанки является эффективным методом оптимизации скорости загрузки web-приложений, позволяя обеспечить максимальную сетевую доступность приложения и наилучший пользовательский опыт.

Литература

1. Макконнелл С. Совершенный код : учебное пособие / С. Макконнелл. — 2-е изд. — Москва : Русская редакция, 2010. — 889 с.
2. Тиленс Т.М. React в действии : учебное пособие / Т.М. Тиленс. — 1-е изд. — Санкт-Петербург : Питер, 2019. — 368 с.
3. Node.js в действии : учебное пособие / К. Симпсон [и др.] ; под ред. Н. Райлих. — 2-е изд. — Санкт-Петербург.: Питер, 2018. — 432 с.
4. Build Options | Vite. — Режим доступа : https://vitejs.dev/config/build-options#build-rollupoptions. – (Дата обращения: 08.04.2024).

УДК *004.5; 65.011.56*

Разработка backend сервиса, которое использует метод срединного квадратичного отклонения для рекомендации друзей в социальной сети на основе общих интересов и предпочтений.

Н. А. Пушкин, Е. П. Белоусова

Воронежский государственный университет

Введение

В современном мире с развитием цифровых коммуникаций и социальных сетей мессенджеры становятся неотъемлемой частью повседневной жизни, предоставляя обмен сообщениями и рекомендации для расширения социального круга. Важным аспектом современных мессенджеров является разработка эффективных API-сервисов для улучшения функциональности и удобства взаимодействия пользователей.

1. Цель исследования

Цель данной работы заключается в создании RESTful API-сервиса для мессенджера, основанного на рекомендациях в друзья на основе среднеквадратичного отклонения информации профиля пользователя. Данный подход предполагает использование статистического метода для оценки сходства между пользователями и определения потенциальных контактов на основе предыдущих взаимодействий, что призвано улучшить функциональность мессенджеров и предоставить пользователям персонализированный опыт и точные рекомендации.

2. Роль мессенджеров и рекомендательных систем

Мессенджеры играют важную роль, предоставляя платформы для мгновенного обмена сообщениями и общения. Рекомендательные системы становятся неотъемлемой частью функционала мессенджеров, предлагая пользователям рекомендации по добавлению новых контактов на основе общих интересов и взаимодействий. Эффективность этих рекомендаций напрямую влияет на привлекательность мессенджера для пользователей.

3. Применение среднеквадратичного отклонения в рекомендательных системах

Среднеквадратичное отклонение (RMSE) отклонения информации профиля пользователя является инструментом для оценки точности предсказаний в рекомендательных системах, позволяя измерить разницу между фактическими и прогнозируемыми значениями. В разработке рекомендательных систем для мессенджеров, использование RMSE помогает оценить точность рекомендаций на основе взаимодействий пользователей.

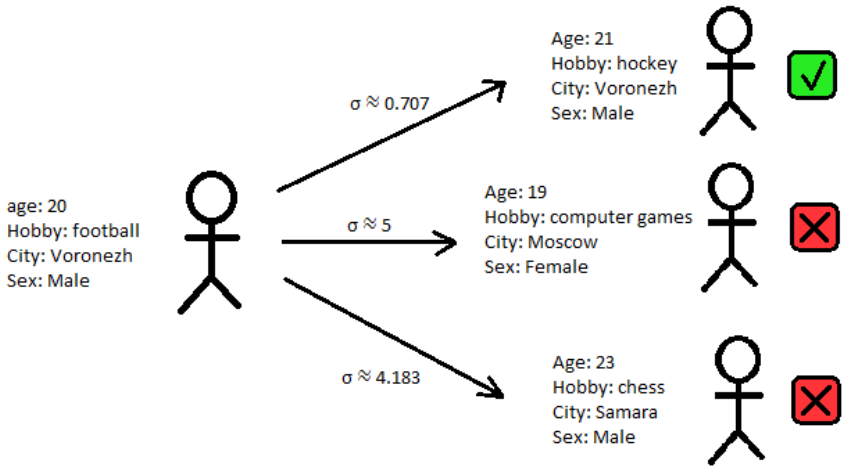


Рис 1. Иллюстрация работы приложения

4. Существующие решения и вызовы

Существует множество решений и вызовов в разработке рекомендательных систем для мессенджеров. Основные вызовы включают точность предсказаний при ограниченных данных, учет новых пользователей и изменение их предпочтений. Разработка API-сервиса на основе RMSE для мессенджеров является актуальным исследовательским направлением.

5. Оптимизация и масштабирование

Выявление потенциальных точек оптимизации алгоритма и работа над их улучшением для обеспечения эффективности и оптимальной работы сервиса при увеличении нагрузки.

6. Методология исследования

В данном исследовании фокус сосредоточен на создании API-сервиса для мессенджера, сфокусированного на рекомендациях контактов на основе метода среднеквадратичного отклонения. Цель заключается не только в функциональности, но и в оптимизации процессов для обеспечения эффективной работы сервиса в реальном времени.

7. Разработка RESTful API-сервиса

7.1 ***Архитектурное проектирование***

- Идентификация основных компонентов системы: пользователи, сообщения, взаимодействия.

- Выбор наиболее подходящей архитектуры API для обеспечения эффективной свя-зи между клиентами и сервером.

7.2 ***Реализация алгоритма рекомендаций на основе среднеквадратичного отклонения***

- Сбор данных о взаимодействиях пользователей в мессенджере.

- Подготовка данных для оценки сходства между пользователями с применением метода среднеквадратичного отклонения.

- Разработка алгоритма, учитывающего предыдущие взаимодействия для предоставления рекомендаций.

8. Результаты разработки API-сервиса

В результате создан API-сервис, обеспечивающий взаимодействие между клиентами мессенджера и сервером. Реализован алгоритм рекомендаций на основе среднеквадратичного отклонения, который определяет сходство между пользователями и предлагает новые контакты на основе их предыдущих взаимодействий.

9. Эффективность и точность рекомендаций

Проведенный анализ показал высокую точность метода рекомендаций, обеспечивая пользователям персонализированный опыт с учетом их предыдущих действий и взаимодействий.

10. Обсуждение результатов

Несмотря на достигнутые успехи, важно учитывать, что точность и эффективность рекомендаций зависят от количества и качества данных. Дальнейшее улучшение алгоритма и расширение набора данных могут повысить качество рекомендаций.

Заключение

Разработанный API-сервис представляет собой важный шаг в улучшении функциональности мессенджеров и предоставлении пользователям персонализированных рекомендаций. Для повышения точности и масштабируемости рекомендаций целесообразно продолжать исследования по оптимизации алгоритма и расширению объема данных.

Литература

1. Курс по программированию - GeekBrains ВТБ (Java для молодых специалистов Банка ВТБ (ПАО)), Neil Alishev Spring course
2. Habr Java Spring Framework - https://habr.com/ru/articles/490586/
3. Habr Spring Boot - https://habr.com/ru/articles/674858/
4. Habr REST AIP - https://habr.com/ru/articles/435144/
5. Habr Микросервисы Spring Boot - https://habr.com/ru/articles/485094/comments/
6. Руководство по вкатыванию в backend-разработку на Java для почти начинающих и сочувствующих - https://github.com/EightM/JavaBackendStartGuide
7. Шилдт Г. Полный курс Java SE 9
8. JSON Web Token (JWT) — пример Java реализации на Spring Boot OAuth2 Resource Server 6.0 - https://habr.com/ru/articles/684270/
9. Аутентификация с использованием Spring Security и JWT-токенов - https://habr.com/ru/articles/278411/
10. Базы данных на Java – первые шаги - https://java-course.ru/begin/database01/

УДК 004.05

Анализ стабильности реактивной и проактивной Java системы

Д. И. Пятайкин, С.Ю. Болотова

Воронежский государственный университет

Введение

В современном мире разработки программного обеспечения растет необходимость в создании систем, способных эффективно справляться с разнообразными нагрузками и условиями эксплуатации. Одним из ключевых аспектов при проектировании и разработке таких систем является их стабильность – способность системы сохранять работоспособность и продолжать обслуживать запросы пользователей даже в условиях сбоев или непредвиденных обстоятельств.

В этой статье проанализированы два популярных подхода к построению веб-приложений на платформе Java: реактивной архитектуре с использованием Spring WebFlux и проактивной архитектуре с применением Spring MVC. Рассмотрены различные подходы к обработке запросов, управлению ресурсами, обработке ошибок, а также возможности масштабирования и адаптации к изменяющимся нагрузкам.

Целью данного анализа является помощь разработчикам и архитекторам в принятии обоснованных решений при выборе подхода к разработке стабильных и отзывчивых веб-приложений на платформе Java.

1. Обзор подходов

Реактивное программирование представляет собой парадигму разработки ПО, ориентированную на создание отзывчивых, масштабируемых и устойчивых систем. Основные принципы реактивной архитектуры включают асинхронное и неблокирующее выполнение операций, использование потоковой обработки информации [1-3].

Spring WebFlux – это реактивный веб-фреймворк, предоставляющий возможность создания реактивных приложений на платформе Java [4]. Он основан на реактивном стеке проекта Reactor, который предлагает асинхронные и неблокирующие операции с использованием реактивных типов данных, таких как Flux и Mono.

В реактивной архитектуре с Spring WebFlux активно используется механизм backpressure, который является ключевым компонентом обеспечения стабильности и устойчивости системы. Backpressure – это механизм управления потоком данных, который позволяет получателю сообщений (например, клиенту или другому сервису) контролировать скорость и объем данных, поступающих от отправителя [5], как демонстрируется на рис. 1.

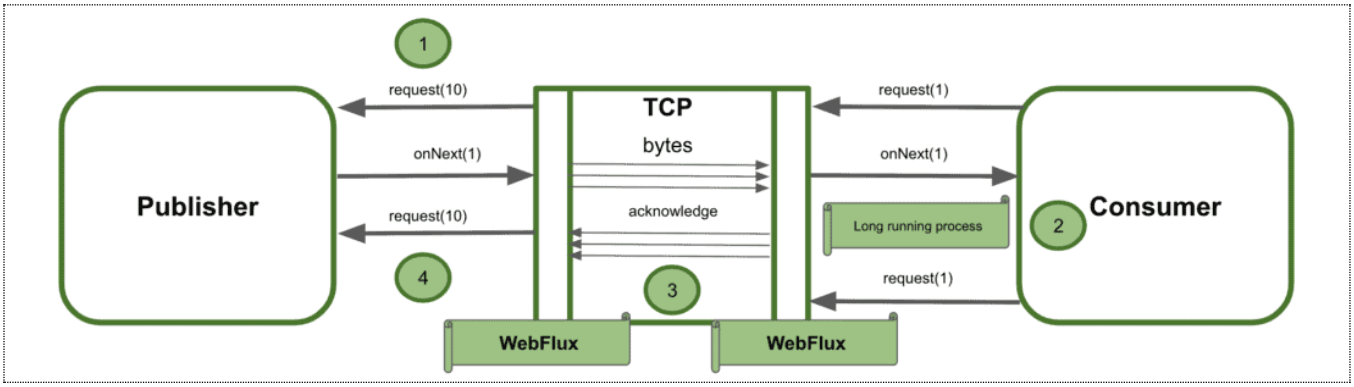


Рис. 21 Схема backpressure механизма

Этот механизм становится критически важным при работе с асинхронными потоками данных, такими как Flux и Mono в Spring WebFlux, где скорость производства данных может быть намного выше, чем скорость их потребления. Если получатель не успевает обрабатывать данные, это может привести к перегрузке и сбоям в системе, как представлено на рис. 2.

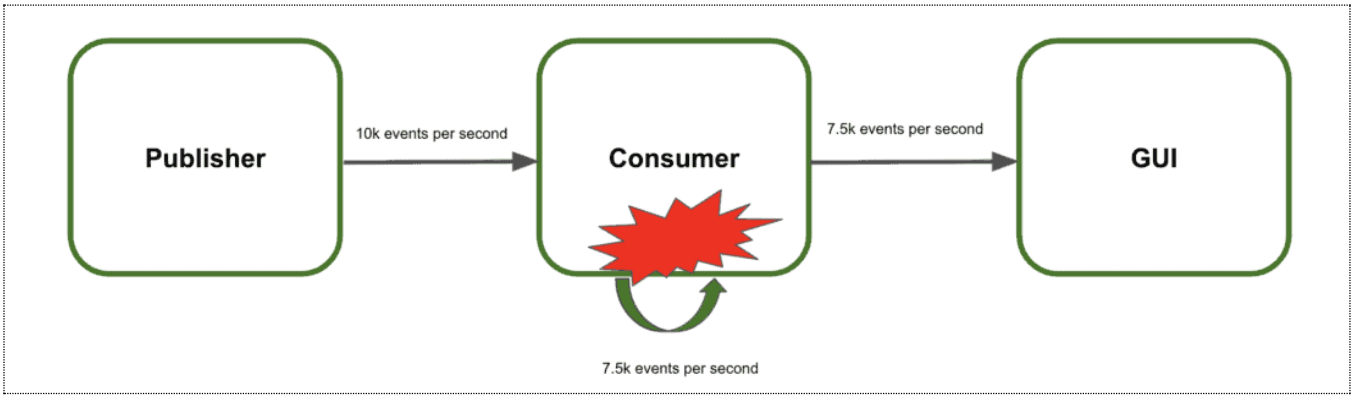


Рис. 22 Перегрузка сервера

Проактивная архитектура, представленная в Spring Framework с использованием модуля Spring MVC (Model-View-Controller), является классическим подходом к построению веб-приложений на Java. Она основана на синхронных операциях и блокирующем вводе-выводе.

В контексте Spring MVC, механизм backpressure не является стандартной частью архитектуры, поскольку Spring MVC работает синхронно и блокирует поток выполнения до завершения обработки запроса. Однако, можно рассмотреть аналогичные концепции, которые помогают улучшить стабильность системы.

1. Пулы потоков: Spring MVC позволяет настраивать и использовать пулы потоков для параллельной обработки запросов.

2. Circuit Breaker: Circuit Breaker – это паттерн, который широко используется для обеспечения стабильности системы во взаимодействии с внешними сервисами или ресурсами [6]. Хотя Spring MVC не предоставляет встроенной реализации Circuit Breaker, существуют сторонние библиотеки, такие как Hystrix или Resilience4j, которые можно интегрировать с приложением Spring MVC [7-8].

3. Распределенный кэш и кэширование: использование распределенного кэша и кэширования может помочь снизить нагрузку на базу данных и улучшить общую производительность системы. Spring Framework предоставляет возможности для интеграции с различными кэш-решениями, такими как Ehcache, Redis или Hazelcast [9-10].

Хотя архитектурные концепции и инструменты, доступные в Spring MVC, могут отличаться от реактивных подходов, эти методы все же позволяют разработчикам повышать устойчивость и стабильность системы путем эффективного управления ресурсами, обработки ошибок и предотвращения перегрузок.

2. Анализ

Для проверки надежности и эффективности двух различных подходов в обеспечении стабильности было проведено стресс-тестирование двух приложений. В сценарии стресс-теста 600 пользователей одновременно создавали 2000 запросов за одну секунду. Далее, эмулировалось снижение пропускной способности для демонстрации реакции систем на экстремальные условия. Приложения запускались со следующими параметрами: -Xmx1G -Xms256m -XX:MaxDirectMemorySize=1g.

На рис.3 изображен график стресс-тестирования Spring MVC с применением паттерна Circuit Breaker. В точке 1 происходит эмуляция снижения пропускной способности. После этого наблюдается значительный рост потребления памяти. В точке 2 сервер падает с ошибкой OutOfMemoryError.

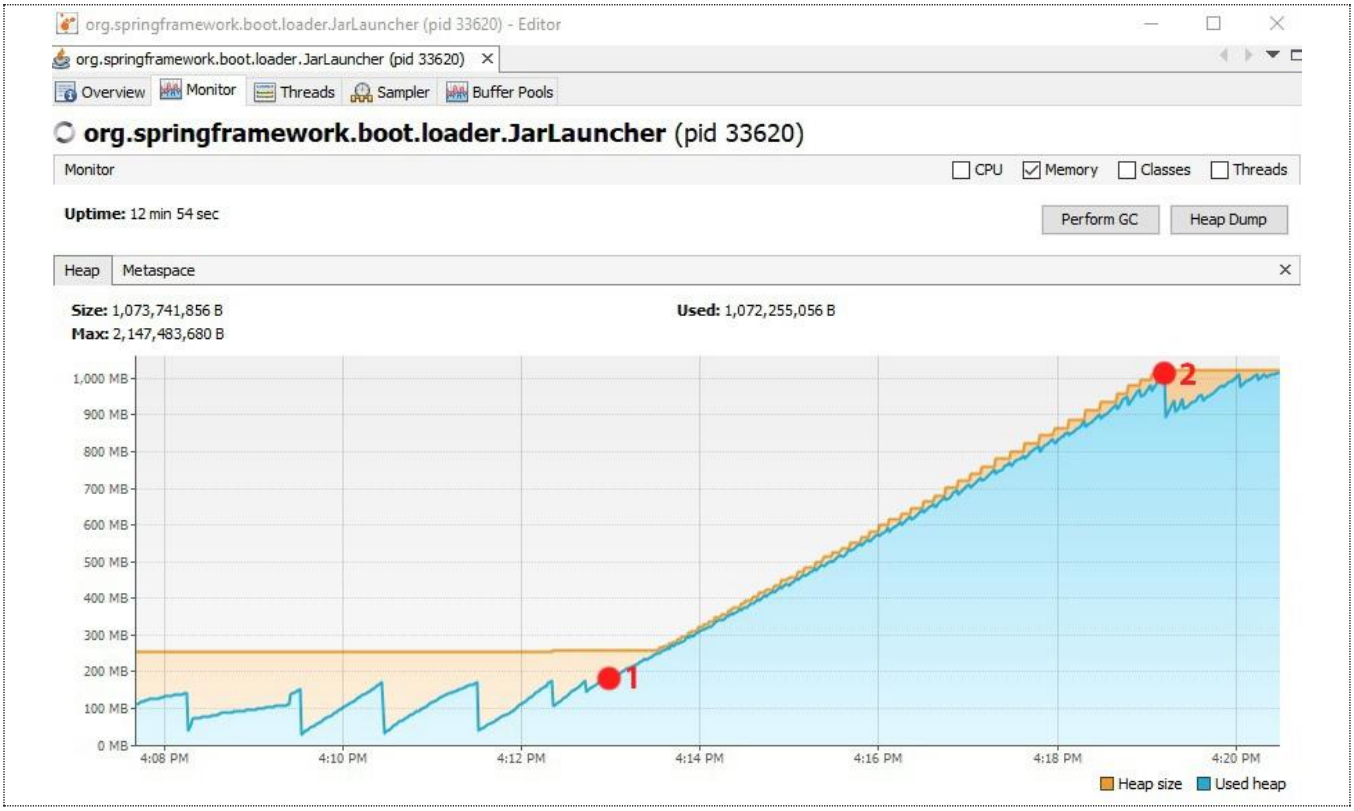


Рис. 3 График стресс-тестирования Spring MVC

На рис. 4 представлена динамика стресс-тестирования Spring WebFlux с применением механизма Backpressure. В момент времени 1 симулируется уменьшение пропускной способности сервера. Это приводит к незначительному увеличению использования оперативной памяти. В точке 2 сервер адаптируется к изменениям, сбалансировав количество генерируемых запросов и продолжив их обработку.

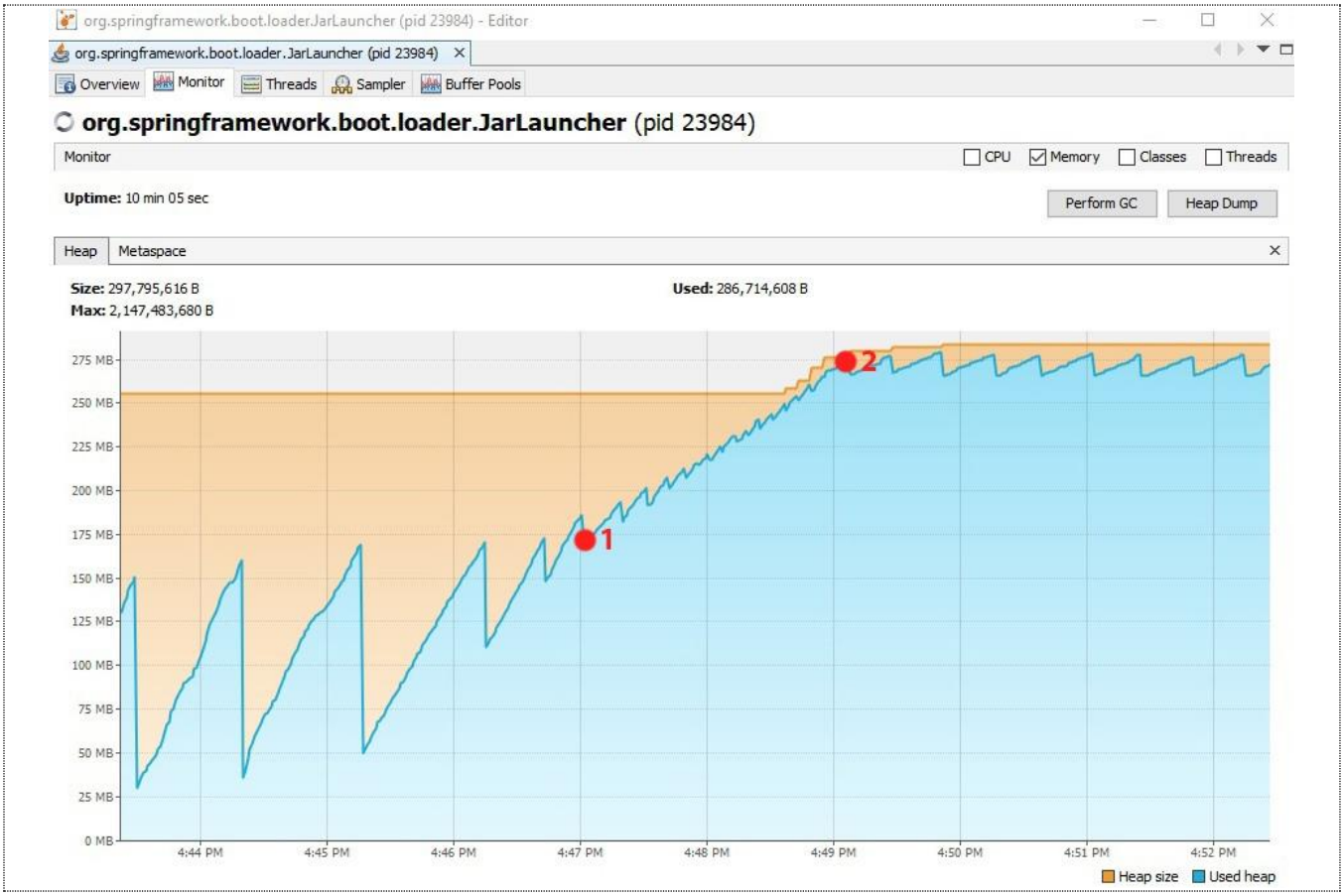


Рис. 4 График стресс-тестирования Spring WebFlux

В контексте снижения пропускной способности сервера подход с Circuit Breaker привел к значительному росту потребления памяти и к аварийному завершению работы. Применение механизма Backpressure в Spring WebFlux позволило системе адаптироваться к изменяющимся условиям, обеспечивая надежную обработку запросов при минимальном увеличении использования оперативной памяти.

Заключение

Проведенный анализ подчеркнул важность выбора подходящей технологии для обеспечения стабильности и отзывчивости системы. Реактивное программирование с применением Spring WebFlux, основанное на асинхронном и неблокирующем выполнении операций, демонстрирует преимущества в управлении высокими нагрузками и взаимодействии с асинхронными потоками данных. С другой стороны, проактивная архитектура с использованием Spring MVC, хотя и не обладает встроенной поддержкой асинхронности,

предоставляет ряд инструментов, которые могут быть эффективно использованы для обеспечения стабильности и отзывчивости приложений в определенных сценариях.

Литература

1. Nurkiewicz, T. Reactive Programming with RxJava: Creating Asynchronous, Event-Based Applications / T. Nurkiewicz, E. Meijer, B. Christensen. – Sebastopol : O'Reilly Media, 2016. – 370 с. – ISBN 978-1491931653.
2. Кун, Р. Реактивные шаблоны проектирования / Р. Кун, Б. Ханафи, Дж. Аллен. – Санкт-Петербург : Питер, 2018. – 416 с. – ISBN 978-5-4461-0474-1.
3. Reactive Systems in Java : сайт. – URL: https://www.baeldung.com/java-reactive-systems (дата обращения: 06.04.2024)
4. Spring Reactive : сайт. – URL: https://spring.io/reactive (дата обращения: 06.04.2024)
5. Spring Webflux Backpressure : сайт. – URL: https://www.baeldung.com/spring-webflux-backpressure (дата обращения: 06.04.2024)
6. Newman, S. Building Microservices. Designing Fine-Grained Systems / S. Newman. – California : Oreilly & Associates Inc, 2015. – 259 с. – ISBN 978-1491950357.
7. Guide to Resilience4j With Spring Boot : сайт. – URL: https://www.baeldung.com/spring-boot-resilience4j (дата обращения: 08.04.2024)
8. A Guide to Spring Cloud Netflix – Hystrix : сайт. – URL: https://www.baeldung.com/spring-cloud-netflix-hystrix (дата обращения: 08.04.2024)
9. Spring Data Redis : сайт. – URL: https://spring.io/projects/spring-data-redis (дата обращения: 08.04.2024)
10. Java Hazelcast : сайт. – URL: https://www.baeldung.com/java-hazelcast (дата обращения: 08.04.2024)

УДК *004*

WEB-СЕРВИСЫ 1С

У. А. Ревина

Воронежский государственный университет

Введение

«1С: Предприятие» является широко используемым в России программным обеспечением для автоматизации управленческого учёта и управления предприятием. Система 1С позволяет компаниям организовать бухгалтерский и управленческий учёт, управление продажами, складом, производством, ресурсами и другими бизнес-процессами в единой информационной среде.

1С — необходимый инструмент для многих компаний, который помогает им справляться с задачами различной сложности. В современном мире бизнес требует эффективного управления и оперативного принятия решений. Программный продукт «1С: Предприятие» предоставляет компаниям возможность автоматизировать процессы и улучшить контроль над бизнесом.

Web-сервисы в системе 1С предоставляют ещё больше гибкости и масштабируемости бизнес-процессов. Они обеспечивают доступ к информации и функциональности программы 1С из любого места, где есть интернет. Это значительно упрощает взаимодействие сотрудников и клиентов, увеличивает скорость принятия решений и повышает эффективность работы компании.

Web-сервис 1С — это программное средство, позволяющее интегрировать различные приложения и информационные системы с платформой «1С:Предприятие». Он состоит из набора стандартизированных интерфейсов, доступных через web-протоколы, облегчающих обмен данными и бизнес-логикой между 1С и другими системами.

Web-сервисы 1С играют важную роль в современном бизнесе, позволяя:

- автоматизировать различные бизнес-процессы, такие как обмен данными с партнёрами, синхронизация с интернет-магазинами, управление складом и т.д.
- облегчить интеграцию «1С:Предприятия» с другими информационными системами, такими как CRM, ERP, системы управления складом, бухгалтерские системы и т. д.
- расширить функциональность «1С:Предприятия» за счёт подключения дополнительных модулей и сервисов.
- повысить эффективность работы компании за счёт автоматизации рутинных задач, ускорения обработки данных и улучшения межведомственного взаимодействия.

Web-сервисы 1С основаны на открытых стандартах SOAP и XML, что обеспечивает совместимость с различными платформами и языками программирования. Они могут работать в режиме реального времени и асинхронно.

Интеграция с внутренними системами — одна из возможностей Web-сервиса 1С. Она возможна с внутренними системами, внешними системами и мобильными приложениями. Например, к внутренним системам относятся CRM, ERP, системы управления складом и бухгалтерские системы, а к внешним системам: системы электронного документооборота, платёжные системы или государственные сайты.

Web-сервисы 1С являются мощными инструментами для оптимизации рутинных задач в бизнесе. Их использование позволяет достичь нескольких важных результатов:

Во-первых, web-сервисы сокращают время выполнения задач. Благодаря автоматизации и автоматическому обмену данными между различными системами, процессы становятся более эффективными и быстрыми. Например, web-сервис может автоматически формировать и отправлять отчёты, позволяя сотрудникам не тратить время на ручное создание и отправку документов.

Во-вторых, использование web-сервисов снижает количество ошибок. Все операции и обмен данными осуществляются автоматически, что исключает возможность человеческого фактора и ошибок, связанных с неправильным вводом данных. Это важно в финансовых и бухгалтерских процессах, где даже небольшая ошибка может иметь серьёзные последствия.

Наконец, web-сервисы также повышают производительность труда. Благодаря автоматизации рутинных задач, сотрудники освобождаются от монотонных и повторяющихся действий, что позволяет им фокусироваться на более важных и творческих задачах. Это способствует росту эффективности работы и общей производительности бизнеса.

Web-сервисы 1С являются легко масштабируемыми инструментами, что делает их доступными и полезными для компаний любого размера.

Существует два варианта масштабирования: горизонтальное и вертикальное. Горизонтальное масштабирование предусматривает добавление новых серверов для равномерного распределения нагрузки и обработки запросов от большего числа пользователей. Это позволяет повысить производительность и отзывчивость системы. Вертикальное масштабирование касается увеличения мощности существующих серверов, например, путём добавления процессоров или памяти. Это позволяет улучшить производительность web-сервиса при обработке большого объёма данных.

Благодаря своей масштабируемости, web-сервисы 1С позволяют компаниям удовлетворить изменяющиеся потребности и требования бизнеса. Они могут быть успешно применены как в небольших компаниях, так и в крупных организациях, обеспечивая эффективный обмен данными и оптимизацию бизнес-процессов.

Web-сервисы 1С, как и любой инструмент, имеют свои ограничения, которые важно учитывать при их использовании:

1. Неполный охват функциональности: Web-сервисы 1С не могут полностью заме-нить все функции, которые предоставляет «1С:Предприятие». Доступ к некоторым

объектам и функциям программы может быть ограничен, а также некоторые сложные операции и бизнес-логику невозможно реализовать через web-сервисы.
2. Ограниченная производительность: Web-сервисы могут работать медленнее, чем локальные приложения, особенно при низкой пропускной способности сети или при большой нагрузке на сервер.
3. Сложность разработки: Для разработки и внедрения web-сервисов 1С требуются опытные и квалифицированные специалисты, а также знание языка программирования «1С:Предприятие».
4. Возможные проблемы с безопасностью: Web-сервисы могут быть уязвимы для хакерских атак, поэтому необходимо обеспечить надёжность и безопасность передачи данных.
5. Ограниченная совместимость: Web-сервисы 1С не всегда совместимы с другими платформами и языками программирования, поэтому перед использованием их необходимо проверить их совместимость с другими системами.
6. Необходимость постоянной поддержки: Web-сервисы 1С требуют постоянного обновления и сопровождения, что может потребовать дополнительных ресурсов и усилий.
7. Дополнительные расходы: Разработка, внедрение и поддержка web-сервисов 1С могут привести к дополнительным расходам для организации.

Примеры кейсов:

- «1С:Бухгалтерия» использует web-сервисы для интеграции с системами электронного документооборота.
- Интернет-магазин Ozon использует web-сервисы для синхронизации информации с «1С:Предприятие».
- Российские железные дороги используют web-сервисы для автоматизации системы управления.

Литература

1. Хрусталёва Е. Ю. Технологии интеграции «1С: Предприятия 8.3» / Хрусталёва Е. Ю. — Москва: 1С-Паблишинг, 2020. — 503 с.

2. Хрусталева Е. Ю. Знакомство с разработкой мобильных приложений на плат-форме "1С:Предприятие 8" / Хрусталева Е. Ю. — 3-е изд. — Москва: 1С-Паблишинг, 2022. — 274 с.

3. Радченко, М. Г., Хрусталева, Е. Ю. 1С:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы. / М. Г. Радченко, Е. Ю. Хрусталева — 3-е изд. — Москва: 1С-Паблишинг, 2023. — 984 с.

4. Интенсивное обучение программированию в 1С. — URL: https://uc1.1c.ru/course/intensivnoe-obuchenie-programmirovaniyu-v-1s-onlajn-video/ https://wiki. merionet. ru/articles/apache-ili-iis/

5. Официальный форум 1с. – URL: https://forum—1c.ru/

УДК 003.26+004.056

ЗАЩИЩЁННЫЕ ПРОМЫШЛЕННЫЕ КРИПТОПРОТОКОЛЫ.

П.С. Ретунский

Воронежский государственный университет

Введение

Тема моего выступления защищённые промышленные криптопротоколы. Особое внимание уделим протоколу CRISP — неинтерактивному методу защищенной передачи данных, разработанному специально для промышленных систем в тесной кооперации с российскими производителями программируемых логических контроллеров.

Этот протокол, ставший результатом усилий 2017-2018 годов, представляет собой уникальное решение, не основанное на подходах и разработках западных вендоров. CRISP был создан для обеспечения передачи компактных блоков данных с минимальными требованиями к вычислительным мощностям и каналам связи, что делает его идеально подходящим для использования в условиях современной промышленности.

Особенностью протокола является его способность защищать данные, передаваемые как в TCP/IP-сетях, так и в альтернативных сетях, например, использующих технологии LPWAN. Протокол CRISP реализует защиту исходных сообщений через опциональное шифрование и вычисление имитовставки, что обеспечивает аутентификацию и защиту от повторного навязывания сообщений.

1. CRISP

1.1. Что такое и для чего нужен

CRISP (CRyptographic Industrial Security Protocol) – неинтерактивный протокол защищенной передачи данных, разработанный для применения в индустриальных системах. Протокол CRISP может быть использован для обеспечения конфиденциальности и имитозащиты сообщений и для защиты от навязывания повторных сообщений.

Протокол CRISP реализует защиту исходных сообщений путем их опционального шифрования, а также вычисления имитовставки, в частности, для аутентификации сообщений и для защиты от навязывания повторных сообщений с использованием криптографических методов. Протокол CRISP представляет собой совокупность набора полей, правил их формирования и обработки и может использоваться с любым протоколом передачи данных, способным доставить сформированные данные адресатам. Он не требует установления соединений, обладает минимальными накладными расходами и низкой ресурсозатратностью. Применяемые в протоколе криптографические наборы на основе блочного шифра «Магма» обеспечивают эффективную работу, в том числе, на маломощных устройствах.

Свойства протокола CRISP позволяют с его помощью защищать данные, передаваемые

как в TCP\IP-сетях, так и в сетях, построенных не на основе стека протоколов TCP/IP, например, при использовании технологий узкополосной передачи LPWAN. Применять новый национальный стандарт можно в качестве слоя защиты для протокола LoRaWAN RU (ГОСТ Р 71168-2023), NB-IoT, ZigBee, XNB, а также для ряда промышленных протоколов. Высокая энергоэффективность позволяет использовать криптографический протокол не только в автоматизированных системах управления технологическим процессом (АСУ ТП), но и в IIoT-системах, где устройства традиционно имеют питание от батарейки или получают его из среды функционирования.

Например, компания «ИнфоТеКС» использует криптографический протокол CRISP в продуктах решения ViPNet SIES – встраиваемых средствах криптографической защиты информации (СКЗИ) для защиты системы IIoT и АСУ.

1.2. Структура CRISP

В своей курсовой работе я реализую данный протокол согласно этому набору (табл. 1.):

Таблица 1

Набор MAGMA-CTR-CMAC8: CS=3

Параметр	**Значение**
EncryptionAlg	Блочный шифр «Магма» согласно ГОСТ Р 34.12-2015 в режиме гаммирования согласно ГОСТ Р 34.13-2015
MACAlg	Блочный шифр «Магма» согласно ГОСТ Р 34.12-2015 в режиме выработки имитовставки согласно ГОСТ Р 34.13-2015
MACLength	8 байтов
DeriveIV	См. описание далее
DeriveKey	См. описание далее

Алгоритм шифрования

Для шифрования данных используется блочный шифр «Магма» согласно ГОСТ Р 34.12-2015 в режиме гаммирования согласно ГОСТ Р 34.13-2015. В качестве ключа используется ключ шифрования сообщения KENC. Значение входного параметра режима гаммирования s = 64. В качестве синхропосылки используется значение, определенное в п. 7.3.3. Криптографическое дополнение данных для режима гаммирования не предусмотрено.

Имитовставка

Для вычисления имитовставки ICV, содержащейся в поле ICV CRISP-сообщения, используется блочный шифр «Магма» согласно ГОСТ Р 34.12-2015 в режиме выработки имитовставки согласно ГОСТ Р 34.13-2015. В качестве ключа используется ключ имитозащиты сообщения KMAC. Значение входного параметра режима выработки имитовставки s = 64. Криптографическое дополнение данных для режима выработки имитовставки выполняется согласно ГОСТ Р 34.13-2015.

Синхропосылка

Для формирования из 48-битного порядкового номера сообщения SeqNum, содержащегося в поле SeqNum CRISP-сообщения, 32-битной синхропосылки IV используются 32 младших бита SeqNum

Ключи

Для выработки производных ключей шифрования и имитозащиты сообщения

используется функция CMAC(Key,Data), которая реализуется с помощью блочного шифра «Магма» согласно ГОСТ Р 34.12-2015 в режиме выработки имитовставки согласно ГОСТ Р 34.13-2015 для данных Data на ключе Key. Значение входного параметра режима выработки имитовставки s = 64. Криптографическое дополнение для режима выработки имитовставки выполняется согласно ГОСТ Р 34.13-2015.

Заключение

Подводя итог, хочется сказать о значительном прогрессе в разработке и внедрении криптографических решений, специально адаптированных под нужды промышленности. Протокол CRISP, который мы рассмотрели, является ярким примером инновационного подхода к защите данных, который сочетает в себе высокую энергоэффективность и надёжность, необходимые для современных индустриальных систем.

Мы видим, что криптографическая безопасность в индустриальных системах и IoT продолжает оставаться одним из приоритетных направлений научных исследований и разработок. Особое внимание заслуживает внедрение национальных стандартов и разработка криптографических решений, которые помогают защищать критически важные системы от современных угроз.

Литература

1. Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров: ГОСТ Р 34.13-2015. -Введ. 19.06.2015.- Москва : Стандартинформ, 2015. – 38 с.
2. Информационная технология. Криптографическая защита информации. Блочные шифры: ГОСТ Р 34.12-2015. -Введ. 19.06.2015.- Москва : Стандартинформ, 2015. – 21 с.

УДК: 04.415.53

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

А.В. Русанов

Воронежский государственный университет

Введение

В современном цифровом мире, где программное обеспечение становится основой функционирования многих отраслей, от финансов до здравоохранения, вопросы качества, безопасности и надежности становятся более актуальными, чем когда-либо. Разработка, тестирование и обслуживание требуют значительных ресурсов. Когда проект включает в себя миллионы строк кода, потребление ресурсов может быть огромным, и, даже в таких случаях, выпуск продукта без дефектов и его последующее обеспечение высоким уровнем качества становятся сложными задачами. Важно отметить, что стоимость выявления и исправления ошибок с течением времени значительно возрастает, и не выявленный дефект может привести к серьезным финансовым убыткам.

1. Актуальность проблемы

В частности, элементы графического интерфейса (GUI) играют ключевую роль во взаимодействии пользователей с программами, придавая функционалу интуитивное и понятное управление. Однако, именно GUI-элементы, будучи визуальным и пользовательским слоем, могут содержать уязвимости, которые остаются незамеченными традиционными методами тестирования.

Тестирование графического интерфейса, как эффективный метод обнаружения дефектов и багов, приходит на помощь в решении данной проблемы. Этот подход позволяет автоматизировано генерировать разнообразные и нестандартные действия, проверяя программное обеспечение.

Актуальность данной темы также подчеркивается ростом числа веб-приложений, мобильных приложений и программного обеспечения в целом. Успешные атаки на веб-ресурсы и утечки чувствительной информации свидетельствуют о необходимости системного тестирования, чтобы гарантировать безопасность и стабильность в условиях быстрого темпа цифрового развития.

2. Постановка задачи

Для начала необходимо определиться, что такое функциональное тестирование графического интерфейса (GUI). Функциональное тестирование — это такое тестирование, когда программное обеспечение проверяют на соответствие функциональным требованиям и спецификациям.

В данной работе будет продемонстрирован один из видов такого тестирования графического интерфейса. В качестве объекта был выбран смартфон на базе ОС Android.

Эмуляция устройства будет создана с помощью Android Studio – интегрированной среды разработки (IDE) от Google для создания Android-приложений. В качестве основного инструмента для работы был выбран Appium. Благодаря открытому коду данного проекта, его можно интегрировать в различные ПО и использовать для автоматизации процессов.

Сама программа для тестирования интерфейса написана на языке Python версии 3.11(testing) в интегрированной среда для разработки приложений PyCharm.

Для демонстрации процесса программа в автоматическом режиме проверит работоспособность нескольких элементов интерфейса: она перейдёт в приложение Settings, затем среди доступного списка опций смартфона откроет пункт Battery и в конце вернётся обратно на главный экран устройства.

3. Реализация задачи

Как было указано ранее, для реализации теста был написан скрипт на языке Python, в котором указаны выполняемые действия. Основой являются две функции:

1. *def driver():*
   app_driver = webdriver.Remote(appium_server_url, options = capabilities_options)
   yield app_driver
   if driver:
   driver.quit()

В начале указывается фикстура pytest с именем driver. Далее инициализируется WebDriver с использованием определенных ранее возможностей. После возвращается экземпляр драйвера для функции тестирования. В конце выполнения теста завершается работу драйвера, чтобы освободить ресурсы.

2. *def test_find_battery(driver) -> None:*
   el = driver.find_element(by=AppiumBy.XPATH, value='//[@text="Battery"]')*
   el.click()
   sleep(3)
   driver.press_keycode(3)

Функция тестирования: данная функция является тестовым случаем pytest. Она находит элемент с текстом "Battery" с использованием XPath и кликает на него. Затем ждет 3 секунды. В комментарии указано "Вернуться на главный экран". Следующая строка использует driver.press_keycode(3) для имитации нажатия кнопки "Home" на устройстве Android.

Запуск происходит в среде PyCharm. Для того, чтобы отслеживать работу среды Appium, параллельно открыта командная строка, пример интерфейса на рис. 1. Вначале происходит непосредственное соединение между Appium и средой смартфона.

Стоит отметить, что при первом запуске теста на устройство устанавливается приложение Appium для дальнейшей работы с ним. Далее происходит непосредственное тестирование интерфейса: программа находится необходимое приложение и пытается перейти в него. Данный процесс продемонстрирован на рис. 2.

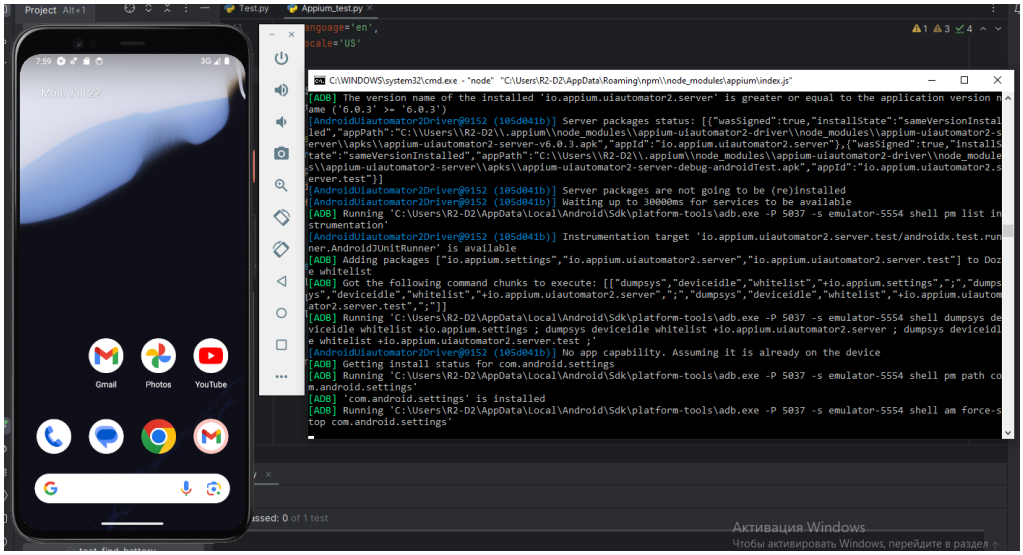


Рис. 1. Отслеживание работы программы Appium во время теста

После успешного перехода в Settings остаётся лишь проверить пункт Battery, с чем программа так же успешно справляется. Данный шаг продемонстрирован на рис. 3.

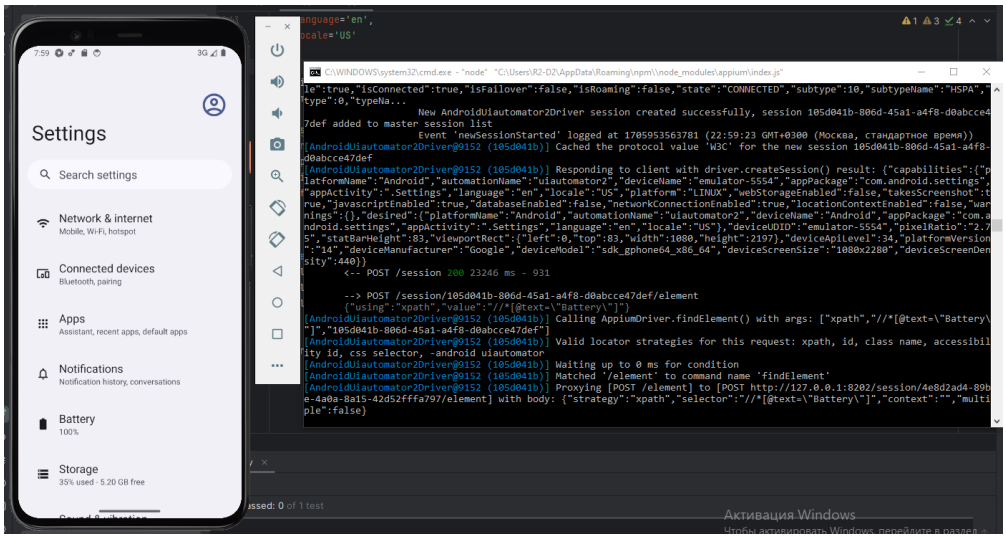


Рис. 2. Открытие приложения Settings

В качестве итога имеется успешное выполнение теста, о чём свидетельствует отчёт в командной строке в соответствии с рис. 4.

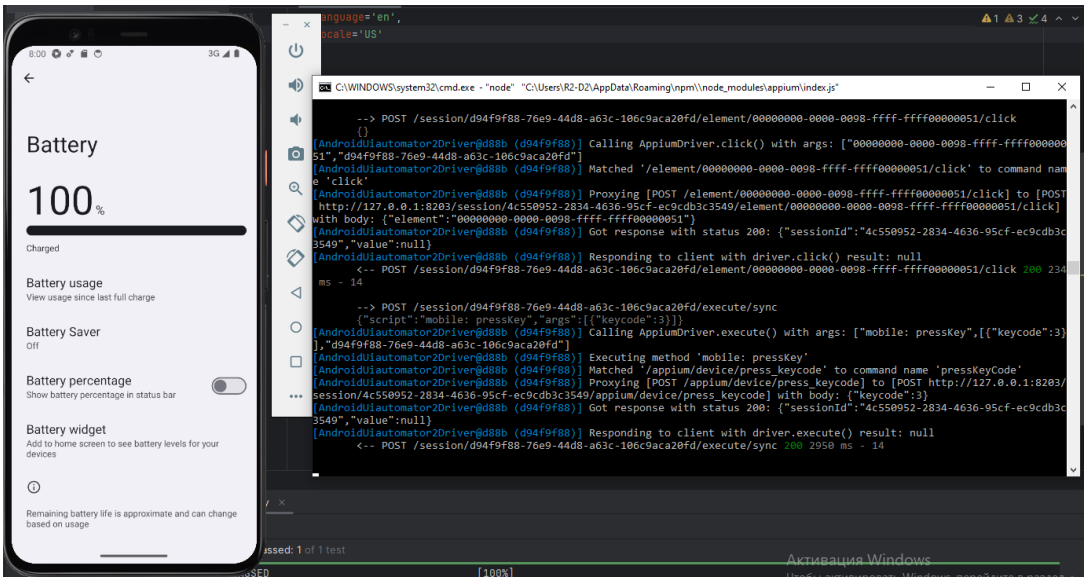


Рис. 3. Открытие пункта Battery в смартфоне

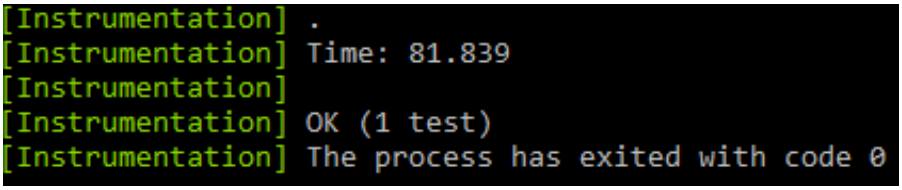


Рис. 4. Результаты работы теста

Заключение

В рамках данной работы было проведено исследование функционального тестирования графического интерфейса. Данный метод тестирования представляет собой мощный инструмент для выявления деффектов в программном обеспечении. В контексте графического интерфейса, это оказывается особенно актуальным, учитывая сложность взаимодействия пользователей с приложениями. Полученные результаты имеют практическую значимость и могут быть использованы при разработке и тестировании графических интерфейсов мобильных и веб-приложений. Эффективное использование функционального тестирования помогает повысить безопасность и стабильность программного обеспечения.

Литература

1. Томилов И.О. [и др.] Разработка методики применения фаззинга для анализа уязвимостей программного обеспечения / И.О. Томилов // Системы управления, связи и безопасности. – 2018. – №4. – С. 48-53.

2. Саттон М. FUZZING исследование уязвимостей методом грубой силы/ М. Саттон, Грин А., Амини П. – Санкт-Петербург – Москва: Символ - Плюс, 2009. – С. 35-39.

3. Тестирование графического интерфейса – Режим доступа: https://vladislaveremeev.gitbook.io/qa_bible/vidy-metody-urovni-testirovaniya/testirovanie-graficheskogo-interfeisa-vizualnoe-testirovanie-gui-graphical-user-interface-testing (Дата обращения: 09.04.2024).

4. Appium – гайд. – Режим доступа: https://testengineer.ru/appium-gaid/ (Дата обращения: 11.04.2024).

5. Мобильные автотесты с нуля. – Режим доступа: https://www.youtube.com/watch?v=I62TIT_Nhho/ (Дата обращения: 11.04.2024).

УДК 004.9

Эффективность использования параллельных вычислений в веб-браузерах при переходе от веб-приложения к мобильному

И. Р. Рябых

Воронежский государственный университет

Введение

В современном мире мобильные устройства становятся все более популярными и востребованными. Переход от использования веб-приложений к мобильным версиям приложений становится неизбежным для многих компаний и разработчиков. Однако, этот процесс может столкнуться с проблемами эффективности и производительности, особенно в контексте параллельных вычислений в веб-браузерах.

Использование параллельных вычислений в веб-браузерах может значительно улучшить производительность мобильных приложений, ускоряя обработку данных и улучшая пользовательский опыт. Однако, эффективность применения параллельных вычислений требует тщательного анализа и оптимизации для каждого конкретного случая.

В данной статье будут рассмотрены возможности и ограничения использования параллельных вычислений в веб-браузерах при переходе от веб-приложения к мобильному, а также представим рекомендации по оптимизации процесса для достижения максимальной эффективности и производительности.

1. Оценка текущего состояния параллельных вычислений в веб-браузерах.

1.1. Обзор существующих методов и технологий параллельных вычислений в веб-разработке.

Существует несколько методов и технологий для реализации параллельных вычислений в веб-браузерах. Ниже приведен обзор основных из них:

1. **Web Workers**: Web Workers позволяют выполнять скрипты в фоновом потоке, не блокируя основной поток браузера. Это позволяет выполнять вычислительно интенсивные операции параллельно с отрисовкой интерфейса. Web Workers поддерживаются всеми современными браузерами и являются одним из наиболее распространенных способов реализации параллельных вычислений.

2. SharedArrayBuffer и Atomics: SharedArrayBuffer предоставляет способ совместного доступа к буферам памяти между различными потоками. Совместно с API Atomics, это позволяет безопасно обмениваться данными между потоками и синхронизировать их выполнение.

3. Service Workers: Service Workers – это скрипты, которые работают в фоновом режиме и могут использоваться для кэширования данных, предварительной загрузки ресурсов и других задач. Хотя Service Workers не предназначены для выполнения вычислений, они могут помочь улучшить производительность приложения, освобождая основной поток от некоторых задач.

4. WebAssembly: WebAssembly – это низкоуровневая бинарная инструкция, которая может быть исполнена в веб-браузере. WebAssembly обеспечивает более высокую производительность по сравнению с JavaScript, что делает его хорошим выбором для выполнения вычислительно сложных операций параллельно.

5. GPU Computing: Некоторые браузеры поддерживают использование графического процессора (GPU) для выполнения параллельных вычислений через API, такие как WebGL или WebGPU. Это позволяет использовать мощности GPU для ускорения обработки данных и графики.

Эти методы и технологии предоставляют разработчикам широкие возможности для реализации параллельных вычислений в веб-браузерах и улучшения производительности мобильных приложений. Однако, необходимо тщательно оценить требования приложения и выбрать наиболее подходящий подход для достижения оптимальных результатов.

1.2. Аналих проблем и ограничений при использовании параллельных вычислений в веб-приложениях

При использовании параллельных вычислений в веб-приложениях существуют определенные проблемы и ограничения, которые могут повлиять на производительность и надежность приложения. Ниже приведены некоторые из них:

1. **Ограничения Web Workers**:

– Web Workers имеют ограничения в доступе к DOM элементам, что может затруднить синхронизацию данных между основным потоком и фоновыми потоками.

– Каждый Web Worker имеет собственный контекст выполнения, что требует передачи данных между потоками через сериализацию и десериализацию, что может быть затратным по ресурсам.

2. **Безопасность SharedArrayBuffer**:

– SharedArrayBuffer был временно отключен во многих браузерах из-за уязвимостей, связанных с спектральными атаками (Spectre).

– Для безопасного использования SharedArrayBuffer необходимо правильно обрабатывать синхронизацию доступа к данным между потоками.

3. **Сложность разработки**:

– Разработка параллельных вычислений в веб-приложениях может быть сложной из-за необходимости управления множеством потоков, синхронизации данных и обработки ошибок.

– Неправильное использование параллельных вычислений может привести к ошибкам и нестабильности приложения.

4. **Производительность**:

– Не всегда параллельные вычисления приводят к улучшению производительности из-за накладных расходов на создание и управление потоками.

– Некорректное использование параллельных вычислений может привести к блокировке основного потока браузера и ухудшению пользовательского опыта.

5. **Совместимость и поддержка**:

– Некоторые методы параллельных вычислений, такие как WebAssembly или GPU Computing, могут быть не поддерживаться во всех браузерах или требовать специфических настроек.

– Не все браузеры одинаково хорошо поддерживают параллельные вычисления, что может затруднить разработку кросс-браузерных приложений.

В целом, параллельные вычисления в веб-приложениях предоставляют широкие возможности для улучшения производительности и функциональности приложений, однако требуют внимательного подхода к разработке и учета ограничений каждого метода для достижения оптимальных результатов.

2. Преимущества параллельных вычислений в мобильной разработке

2.1. Рассмотрение потенциальных выгод от применения параллельных вычислений в мобильных веб-приложениях.

Применение параллельных вычислений в мобильной разработке может принести ряд значительных преимуществ, улучшающих производительность, отзывчивость и эффективность приложений. Ниже перечислены потенциальные выгоды от использования параллельных вычислений в мобильных веб-приложениях:

1. **Улучшенная производительность**:

– Параллельные вычисления позволяют распределить нагрузку на несколько ядер процессора мобильного устройства, что может значительно увеличить скорость выполнения вычислений.

– Многопоточность позволяет эффективно использовать ресурсы устройства и ускорить выполнение операций, таких как обработка данных, вычисления и рендеринг.

2. **Улучшенная отзывчивость**:

– Использование параллельных вычислений позволяет выполнять тяжелые вычисления в фоновом режиме, не блокируя основной поток пользоватательского интерфейса.

– Это позволяет приложению оставаться отзывчивым и плавным даже при выполнении сложных операций.

3. **Энергоэффективность**:

– Параллельные вычисления могут помочь оптимизировать использование ресурсов устройства, что может привести к уменьшению потребления энергии и увеличению времени автономной работы устройства.

4. **Увеличение возможностей приложения**:

– Параллельные вычисления позволяют обрабатывать большие объемы данных, выполнять сложные алгоритмы и реализовывать функциональность, которая требует высокой производительности.

– Это открывает новые возможности для создания более мощных и функциональных мобильных приложений.

5. **Оптимизация работы с сетью**:

– Параллельные вычисления могут быть использованы для оптимизации работы с сетью, например, для параллельной загрузки и обработки данных с сервера, ускоряя обновление контента в приложении.

Использование параллельных вычислений в мобильной разработке может значительно повысить производительность и эффективность приложений, обеспечивая лучший пользовательский опыт и расширяя возможности функциональности приложений.

2.2. Изучение способов оптимизации производительности и улучшения пользовательского опыта через параллельные вычисления.

Изучение способов оптимизации производительности и улучшения пользовательского опыта через параллельные вычисления в мобильной разработке может принести ценные результаты. Вот некоторые способы, как параллельные вычисления могут помочь оптимизировать производительность и улучшить пользовательский опыт в мобильных приложениях:

1. Распределение вычислений:

– Использование параллельных вычислений позволяет распределить нагрузку на несколько ядер процессора, что ускоряет выполнение операций и снижает время отклика приложения.

2. **Многопоточность**:

– Разделение задач на отдельные потоки позволяет выполнять одновременно несколько операций, таких как загрузка данных, обработка изображений или выполнение сложных вычислений, что повышает эффективность приложения.

3. **Фоновые задачи**:

– Параллельные вычисления позволяют выполнять тяжелые вычисления и обработку данных в фоновом режиме, не блокируя пользовательский интерфейс, что способствует плавности работы приложения.

4. **Оптимизация работы с сетью**:

– Использование параллельных вычислений позволяет оптимизировать работу с сетью, например, параллельно загружать данные с сервера или кэшировать данные для быстрого доступа, улучшая скорость загрузки и отображения контента.

5. **Управление ресурсами устройства**:

– Параллельные вычисления помогают эффективно использовать ресурсы устройства, такие как процессор, память и сеть, что позволяет оптимизировать энергопотребление и увеличить производительность приложения.

Изучение способов применения параллельных вычислений в мобильной разработке и оптимизации производительности может значительно улучшить пользовательский опыт, сделать приложение более отзывчивым и эффективным, а также расширить его функциональность.

3. Практическое применение параллельных вычислений в мобильной веб-разработке

3.1. Представление конкретных примеров использования параллельных вычислений для ускорения загрузки страниц, обработки данных и других задач в мобильных веб-приложениях

В мобильной веб-разработке параллельные вычисления могут быть использованы для улучшения производительности и оптимизации пользовательского опыта. Вот несколько конкретных примеров применения параллельных вычислений в мобильных веб-приложениях:

1. **Параллельная загрузка ресурсов**:

– Использование параллельных запросов к серверу для загрузки статических ресурсов, таких как изображения, стили и скрипты, позволяет ускорить загрузку страницы и улучшить время отклика приложения.

2. **Многопоточная обработка данных**:

– Разделение обработки данных на несколько потоков позволяет эффективно обрабатывать большие объемы информации, например, при работе с большими наборами данных или при выполнении сложных вычислений на стороне клиента.

3. **Фоновые вычисления**:

– Выполнение тяжелых вычислений и обработки данных в фоновом режиме с помощью Web Workers позволяет не блокировать пользовательский интерфейс и предоставлять плавный пользовательский опыт.

4. **Параллельная обработка событий**:

– Распределение обработки событий на разные потоки позволяет улучшить отзывчивость интерфейса и предотвратить задержки при выполнении различных действий пользователем.

5. **Кэширование данных**:

– Использование параллельных вычислений для кэширования данных на клиентской стороне позволяет быстро получать доступ к ранее загруженным данным без необходимости повторной загрузки с сервера.

6. **Оптимизация работы с API**:

– Параллельная обработка запросов к API позволяет эффективно загружать и обрабатывать данные с различных источников, ускоряя работу приложения и улучшая пользовательский опыт.

Использование параллельных вычислений в мобильной веб-разработке может значительно повысить производительность приложения, ускорить загрузку страниц, оптимизировать обработку данных и создать более отзывчивый пользовательский интерфейс.

3.2. Обсуждение методов интеграции параллельных вычислений с существующими технологиями и инструментами разработки.

Интеграция параллельных вычислений с существующими технологиями и инструментами разработки в мобильной веб-разработке может быть достигнута через использование следующих методов:

1. **Web Workers**:

– Web Workers позволяют выполнять скрипты в фоновых потоках, не блокируя основной поток исполнения JavaScript. Это позволяет выполнять тяжелые вычисления, обработку данных и другие операции параллельно, улучшая производительность и отзывчивость приложения.

2. **Service Workers**:

– Service Workers используются для обработки событий сети в фоновом режиме, что позволяет кэшировать ресурсы, обрабатывать запросы к серверу и выполнять другие задачи без прямого взаимодействия с пользовательским интерфейсом. Это также способствует оптимизации производительности приложения.

3. **Распределенные вычисления на стороне сервера**:

– Для выполнения сложных вычислений или обработки больших объемов данных можно использовать распределенные вычисления на стороне сервера с помощью технологий, таких как Node.js и фреймворки для параллельных вычислений.

4. **Использование библиотек и фреймворков**:

– Существуют специализированные библиотеки и фреймворки, такие как TensorFlow.js для машинного обучения в браузере или D3.js для визуализации данных, которые могут использовать параллельные вычисления для улучшения производительности и функциональности приложения.

5. **Асинхронные запросы к API**:

– Использование асинхронных запросов к API позволяет загружать данные параллельно и эффективно обрабатывать ответы от сервера без блокировки пользовательского интерфейса.

6. **Оптимизация работы с базами данных**:

– При работе с базами данных на клиентской стороне или с использованием IndexedDB можно использовать параллельные вычисления для эффективного доступа к данным, выполнения запросов и обработки результатов.

Интеграция параллельных вычислений с существующими технологиями и инструментами разработки в мобильной веб-разработке позволяет улучшить производительность, оптимизировать обработку данных и создать более отзывчивый пользовательский опыт.

4. Выбор подходящих инструментов и технологий для параллельных вычислений в мобильных веб-приложениях

4.1. Обзор современных инструментов и технологий, которые могут быть использованы для реализации параллельных вычислений в мобильной веб-разработке.

При разработке мобильных веб-приложений с поддержкой параллельных вычислений важно выбирать подходящие инструменты и технологии, которые обеспечат эффективную работу приложения. Вот некоторые из современных инструментов и технологий, которые могут быть использованы для реализации параллельных вычислений в мобильной веб-разработке:

1. Web Workers позволяют выполнять скрипты в фоновых потоках, что освобождает основной поток для других задач и улучшает отзывчивость приложения. Они могут быть использованы для выполнения тяжелых вычислений параллельно с основным потоком JavaScript.

2. Service Workers используются для обработки событий сети в фоновом режиме, что позволяет кэшировать ресурсы, обрабатывать запросы к серверу и выполнять другие задачи параллельно с основным потоком JavaScript. Они могут значительно улучшить производительность и надежность мобильного веб-приложения.

3. WebAssembly представляет собой бинарный формат исполняемого кода, который может быть запущен в веб-браузере. Он обеспечивает высокую производительность и позволяет использовать другие языки программирования, такие как C++ или Rust, для реализации вычислений, которые могут быть выполнены параллельно с JavaScript.

4. Некоторые браузеры предоставляют специальные расширения, такие как WebGPU API, которые позволяют использовать графический процессор для параллельных вычислений. Это особенно полезно для задач, требующих интенсивной обработки графики или видео.

5. Существуют специализированные фреймворки, такие как TensorFlow.js или Parallel.js, которые облегчают реализацию параллельных вычислений в мобильных веб-приложениях. Они предоставляют удобные API и инструменты для работы с параллельными задачами.

При выборе инструментов для параллельных вычислений в мобильных веб-приложениях необходимо учитывать требования проекта, особенности целевой аудитории, доступные ресурсы и опыт разработчиков. Каждый из перечисленных инструментов имеет свои преимущества и ограничения, поэтому важно выбирать тот, который наилучшим образом подходит для конкретной задачи.

4.2. Сравнение различных подходов и рекомендации по выбору наиболее подходящих инструментов для конкретных задач.

При выборе подходящих инструментов и технологий для параллельных вычислений в мобильных веб-приложениях следует учитывать несколько ключевых факторов, таких как тип задачи, требования к производительности, сложность реализации, доступность ресурсов и опыт разработчиков. Рассмотрим сравнение различных подходов и рекомендации по выбору наиболее подходящих инструментов для конкретных задач:

1. Web Workers:

– **Подходит для**: Выполнения тяжелых вычислений в фоновом режиме, обработки данных, распределенных вычислений.

– **Рекомендации**: Web Workers являются отличным выбором для простых параллельных задач, требующих минимальной синхронизации с основным потоком. Они легко внедряются во многие мобильные веб-приложения и обеспечивают хорошую производительность.

2. Service Workers:

– **Подходит для**: Обработки событий сети, кэширования ресурсов, выполнения задач в фоновом режиме.

– **Рекомендации**: Service Workers полезны для задач, связанных с сетью и обработкой запросов к серверу. Они могут значительно улучшить производительность мобильного веб-приложения за счет выполнения задач в фоновом режиме.

3. WebAssembly:

– **Подходит для**: Высокопроизводительных вычислений, использования других языков программирования.

– **Рекомендации**: WebAssembly обеспечивает высокую производительность и позволяет использовать другие языки программирования для реализации сложных вычислений. Он подходит для приложений, требующих интенсивной обработки данных.

4. Расширения браузера:

– **Подходит для**: Графических вычислений, обработки видео, интенсивной обработки графики.

– **Рекомендации**: Расширения браузера, такие как WebGPU API, могут быть использованы для параллельных вычислений, связанных с графикой и видео. Они обеспечивают доступ к графическому процессору для ускорения выполнения задач.

5. Фреймворки для параллельных вычислений:

– **Подходит для**: Сложных параллельных задач, машинного обучения, обработки данных.

– **Рекомендации**: Специализированные фреймворки, такие как TensorFlow.js или Parallel.js, предоставляют удобные инструменты для реализации сложных параллельных вычислений. Они подходят для приложений, требующих высокой производительности и специализированных вычислений.

При выборе подходящих инструментов для параллельных вычислений в мобильных веб-приложениях необходимо учитывать конкретные требования проекта и особенности задачи. Рекомендуется провести анализ производительности, сложности реализации, доступности ресурсов и опыта разработчиков для выбора наиболее подходящего инструмента или комбинации инструментов.

Заключение

Параллельные вычисления играют ключевую роль в оптимизации производительности мобильных веб-приложений, позволяя распределять вычислительные задачи между несколькими потоками или устройствами для улучшения скорости выполнения и отзывчивости приложения. Развитие технологий параллельных вычислений в мобильной веб-разработке открывает новые возможности для создания более эффективных и мощных приложений.

Для разработчиков, стремящихся оптимизировать свои веб-приложения при переходе на мобильные устройства через использование параллельных вычислений, важно учитывать следующие ключевые рекомендации:

1. Анализ производительности: Перед внедрением параллельных вычислений в приложение необходимо провести тщательный анализ производительности, чтобы определить узкие места и потенциальные задачи, которые могут быть оптимизированы с помощью параллельных вычислений.

2. Выбор подходящих инструментов: Выбор подходящих инструментов и технологий для параллельных вычислений зависит от конкретных требований приложения и задач, которые требуется выполнить. Необходимо оценить преимущества и ограничения каждого инструмента перед принятием решения.

3. Оптимизация кода: Разработчики должны стремиться к оптимизации кода при реализации параллельных вычислений, чтобы избежать избыточной нагрузки на систему и обеспечить эффективное использование ресурсов устройства.

4. Тестирование и отладка: После внедрения параллельных вычислений необходимо провести тщательное тестирование и отладку приложения, чтобы убедиться в его стабильной

работе и отсутствии ошибок.

5. Обучение и саморазвитие: Постоянное обучение и изучение новых технологий параллельных вычислений поможет разработчикам быть в курсе последних тенденций и эффективно применять их в своей работе.

В целом, эффективное применение параллельных вычислений в мобильной веб-разработке требует комплексного подхода, сочетающего техническое понимание, оптимизацию производительности и умение выбирать подходящие инструменты для конкретных задач. С постоянным развитием технологий параллельных вычислений и повышением доступности высокопроизводительных устройств, перспективы дальнейшего развития эффективности использования параллельных вычислений в мобильной веб-разработке остаются перспективными и обещают новые возможности для создания инновационных приложений.

Литература

1. Федотов, И. Е. Модели параллельного программирования / И.Е. Федотов. – М.: Солон-Пресс, 2019. – 848 с.

2. Шеметова, А. Д. Методические приемы обучения параллельному программированию / А.Д. Шеметова. – М.: Синергия, 2022. – 770 с.

3. Богачёв К.Ю. Основы параллельного программирования. – М.: БИНОМ. Лаборатория знаний, 2003. – 342 с.

4. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. : Пер. с англ. – М.: Изд-во "Вильямс", 2003. – 512 с.

УДК 004.85

Использование нейронной сети для игры в шахматы

А. В. Рягузов, Д. В. Борисенков

Воронежский государственный университет

Введение

Данная статья посвящена применению сверточной нейронной сети для решения традиционной задачи искусственного интеллекта — автоматического выбора хода при игре в шахматы. Рассмотрено влияние специфики решаемой задачи на особенности применения выбранного метода и альтернативных методов ее решения. Описаны архитектура предлагаемой нейронной сети, процессы предварительной обработки данных, обучения и тестирования нейронной сети.

1. Использование сверточных нейронных сетей для прогнозирования ходов

1.1 Использование свероточных неронных сетей в других играх.

Сверточные нейронные сети доказали, что могут успешно решать различные традиционные задачи искусственного интеллекта, которые сводятся к задачам классификации. Так, в декабре 2014 года ученые Эдинбургского университета Кристофер Кларк и Амос Сторки опубликовали статью о работе сверточной нейросети для игры го, в которой сообщили о точности в 44,4 % в прогнозировании ходов профессиональных игроков. Поскольку го является игрой, требующей наличия высокого уровня в построении стратегии, навыков абстрактного логического мышления и возможности полагаться на интуицию, данный результат можно назвать очень успешным и указывающим на то, что сверточные нейронные сети, обученные © использованием правильной архитектуры и на подходящем и исчерпывающем наборе данных, могут играть на уровне профессиональных игроков, решая комплексные лотические задачи.

Успех срерточных нейронных сетей для игры в го можно объяснить достаточно плавным расположением позиций в игре, остающимися непрерывными в течении игр. Кроме того, поскольку каждый ход в го добавляет на доску одну фигуру, по сути, меняя значение пикселя, разница в представлении доски до и после хода плавная и почти всегда связана с важными закономерностями, понимаемыми сетью, которая способствует согласованности с алгоритмами классификации го.

Го — не единственная игра, в которой сверточные нейронные сети показали свою состоятельность. Так, в 1999 году пионер эволюционных вычислений Дэвид Фогель опубликовал статью, где было показано, как нейросеть может научиться играть в шашки с помощью коэволюции. В процессе обучения не использовались предшествующие человеческие профессиональные игры, а, скорее, основное внимание уделялось минимальному набору информации, содержащейся в шашечной доске: расположению и типу фигур, а также разнице в количестве фигур на двух сторонах.

1.2. Проблемы подхода сверточных нейронных сетей к шахматам

В отличие от го, шахматы в большей степени построены на эвристике нескольких видов фигур в разнообразных краткосрочных стратегиях, перерастающих в долгосрочные стратегии. Это имеет важное значение, поскольку преимущество позиции в шахматах зависит не только от количества фигур, но и от каждой фигуры по отдельности и их особенностей, включая вид фигуры, ее расположение в конкретной позиции, количество возможных ходов и т.д. Это делает выявление закономерностей в шахматах более зависимым от понимания того, как специфическое расположение фигур увеличивает или уменьшает их ценность.

Еще одно отличие шахмат от го заключается в том, что шахматная позиция после каждого хода не меняется плавно, так как каждый ход вызывает сдвиг на два пикселя на доске 8×8, в 1/32 раза, что гораздо более существенно, чем изменение на 1 пиксель доски 19×19 в го (изменение в 1/361 раза).

Эти причины заставляют задуматься о целесообразности описания логических закономерностей в шахматах в активационных слоях. Такие важные концепции, как пешечные цепи и их защита, наличие проходных пешек, возможность рокировки, аванпосты,вилки, закрытая или открытая позиция, в которых могут быть сильны или слабы разные фигуры, лучше всего описываются эвристическими методами и информационными системами, описывающие определенные логики, такие как «если перед пешкой впереди по диагонали стоит фигура», или «если слон на главной диагонали», или «сейчас дебют, миттельшпиль или эндшпиль» и т. д.

Понимание шахмат в большей степени характеризуется знанием предметной области. Таким образом, сверточная нейронная сеть должна поддерживаться и сочетаться с другими методами и подходами шахматного интеллекта для достижения наилучших результатов с возможностью просчитывать на много ходов вперед и связью с функцией оценки.

1.3. Шахматное мышление как распознавание образов

Прямой подход классификации ходов – это нестандартный подход, в котором ходы совершаются без понимания того, почему эти ходы делаются. Вместо этого используется понимание того, какие модели побуждают совершать действия в определенной ситуации. Однако для прогнозирования шахматных ходов за очень короткое время с высокой точностью используется предопределенная модель.

Традиционные подходы к разработке шахматного интеллекта делятся на две категории: с использованием функции оценки и с использованием функции поиска. Функция оценки оценивает позицию по относительному значению того, насколько высока вероятность победы, а функция поиска включает в себя расчет на несколько ходов вперед с использованием минимакса и функции оценки. Поскольку шахматы – это конечная разрешимая игра, этот подход в первую очередь ограничен вычислительными возможностями и эффективностью оценочной функции. Скачки в шахматном интеллекте устраняют любое из этих ограничений за счет эффективной навигации по пространству поиска или включения шахматных концепций в оценочную функцию. Неудивительно, что подходы машинного обучения к шахматам извлекли выгоду из задачи создания эффективной оценочной функции, пытаясь распознать образы на точки данных, помеченные цифрой 1 – если белые являются выигрывающей стороной, и 0, если белые являются проигрывающей стороной. В этом случае данные рассматриваются просто как образцы позиций, увиденных в реальных играх, которые были доведены до конечного результата с надеждой, чтооптимальные ходы были сыграни вперед и что преимущество в данной позиции проявляется в правильном исходе игры (то есть игрок продолжает играть

оптимально). Несмотря на того, что данный подход правильный, он серьезно скомпрометирован из-за слабой маркировки набора данных, и мало что можно сделать, чтобы преодолеть эту систему.

1.4. Сверточные нейронные сети в шахматах

Критики сверточных нейросетей утверждают, что нейросети не могут адекватно объяснить такие тактические преимущества, поскольку формы этих условий слишком общие в рамках шахматной доски и подвержены влиянию внешних переменных. Однако можно утверждать, что эти недостатки в основном являются результатом плохо сформулированной задачи обучения бинарным обозначением выигрыша и проигрыша. Такой алгоритм позволяет развить интуицию предсказания относительно того, соответствуют ли локальные закономерности выигрышному или проигрышному состоянию, ассоциация которых, вероятно, слаба в большинстве шахматных позиций. Однако существует другая и более подходящая для сверточной нейросети задача – использование небольших локальных функций для создания шахматных ходов. Такие функции активируются в механизмах, которые служат эвристикой и интуитивными шаблонами, созданными реальными игроками. По этой причине стоит отказаться от односторонней маркировки шахматной доски и смоделировать инкрементальные тактические решения, помечая каждое состояние доски ходом, сделанной на ней. Такая философия лучше отражает сущность шахматных ходов в профессиональной шахматной игре: почти каждый ход, сделанный игроком с высоким рейтингом ЭЛО, является разумным ходом, особенно при усреднении по всему обучающему набору.

При этом подходе в исходном образе имеют значение те закономерности, которые побуждают ходить определенным образом. Цена увеличения информативности пометок заключается в том, что пространство классов значительно выросло. Интересно также отметить, что классификация на следующем лучшем ходу действует как предварительный расчет для дальнейших состояний доски, включенных в функцию поиска, так как потребности для поиска соответствуют пониманию того, что ход был сыгран для данного представления на доске. Прогноз в этой модели теперь важен для составления последовательных стратегических планов, а не для более строгих оценок доски с использованием минимакса.

1.5. Основной подход

Основная задача данного подхода к обучению заключается в том, что количество возможных ходов очень велико. Пространство классов для следующего хода в го всегда представляет собой некоторое подмножество 19×19=361 возможных позиций, в шахматах, несмотря на в среднем 50 возможных ходов в любой позиции, существует $(8\times8)^2$ =4096 возможных классов, для которых сверточная нейросеть должна получить оценку. В связи с этим, есть смысл разделить задачу классификации на две части, используя более нестандартный подход. В первый части происходит обучение нейросети для координат, с которых фигура должна уйти. Это отражает идею побега, когда фигура находится под боем, или король должен сделать ход. Нейросеть принимает в качестве входных данных представление доски, а затем выводит распределение вероятностей по сетке для того, насколько желательно для фигуры покинуть поле, в случае если на других полях нет фигур (фигуры соперника помечены нулем). Вторая часть – это обучение 6 отдельных нейросетей для того, чтобы выяснить, на какую координату было бы выгодно поставить каждую из 6фигур. Например, есть нейросеть для слона, которая принимает на вход шахматную доску

и выводит распределение вероятностей по всей сетке, насколько желательно поставить слона на каждое поле. Все поля, на которые слон не может пойти, помечаются 0.

Получаем оптимальный ход, сопоставив нейросеть выбора фигуры и нейросеть выбора хода путем умножения значений в нейросети фигур на самое высокое значение в соответствующей нейросети ходов, приняв максимальный аргумент по всей композиции для получения двух координат: поле с которого перемещается фигура, и поле, на которое она перемещается. В нейросети фигур нулевыми вероятностями помечаются позиции, на которых нет фигур того же цвета, в нейросети ходов нулевыми вероятностями помечаются позиции, на которые невозможно совершить ход.

Стоит отметить, что каждая нейросеть теперь имеет размер, равный 64 (квадратный корень от первоначального) ценой удвоения времени обучения, поскольку теперь нужно принимать решение только по одной позиции. Однако этот подход отражает большую часть человеческой интуиции, лежащей в основе шахматного мышления: иногда ход делается в духе защиты фигуры под атакой, и в других случаях, чтобы увидеть позиционное преимущество. Недостатком этого подхода является то, что весьма специфические комбинации ходов между обеими нейросетями не изучены, хотя и считается, что существует достаточно представлений для изучения в каждой сети.

Поскольку тип образа в данном проекте уникален для задачи классификации изображений, у нас было мало исходных данных о том, как различные архитектуры соответствовали ситуации.

2. Технический подход

2.1. Структура данных

В данной модели шахматная доска представлена как образ размером 8×8×6, два измерения представляют собой шахматную доску, 6 каналов соответствуют каждому типу фигур. Используется разное представление для каждого типа фигур, поскольку их ценности

– дискретные значения, разность между двумя типами фигур не является непрерывной, поэтому не может быть посчитана на одном канале. Также в данной модели используется один слой для обоих цветов, свои фигуры имеют значение +1, фигуры противника – Значение -1.

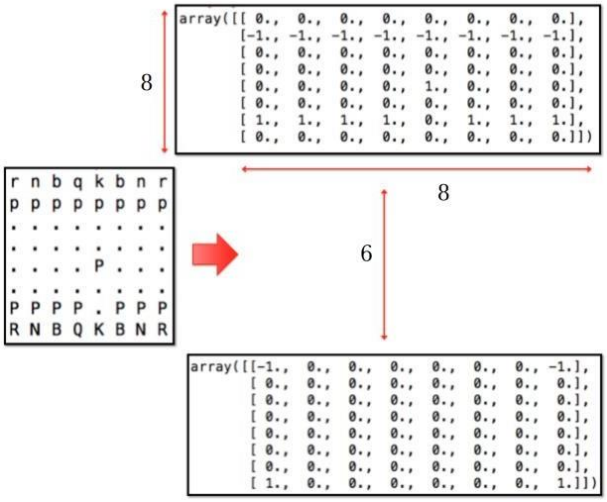


Рис. 1. Представление доски. Верхний слой соответствует пешкам, нижний – ладьям

Нейронные сети должны будут обучаться на 7 наборах данных. Первый – набор данных выбора хода с координатами, с которых фигура делает свой ход. Остальные 6 – наборы выбора фигур с координатами полей, на которые может пойти класс каждой фигуры.

Несмотря на то, что не имеет значения, на каком цвете будет обучаться нейросеть, нужно убедиться, что расположение доски со стороны нейросети, будет совпадать с цветом. По этой причине можно расширить данные так, чтобы алгоритм также мог обучиться и на черных, и на белых фигурах: при обучении на черных фигурах доска отражается горизонтально и вертикально. Таким образом, используя это увеличение, нейросеть также может играть на стороне черных в режиме реального времени.

2.2. Архитектура сверточной неронной сети

Все 7 нейросетей принимают в качестве входных данных описанные выше представления, в качестве выходных данных – распределение вероятностей размерностью 8×8, представляющее оценки каждой позиции. Предпочтительная архитектура – нейросеть, состоящая из 3 слоев: сверточный слой с функцией активации relu, 2 слоя аффинных преобразований с использованием функции потерь softmax, с 32 и 128 функциями соответственно. Обучение нейросетей с более комплексной архитектурой, например с дополнительными сверточными слоями, становится менее возможным, поскольку поиск по сетке на нейросети из 5 слоев приводит к приводящий к отсутствию обучения по всем гиперпараметрам, в то время как обучение сети из 3 слоев приводит к минимальнымразличиям. Возможно это связано с тем, что слишком много параметров используется для небольших и разреженных данных, которые становится трудно обучать на более высоких уровнях. Подчеркивается необходимость наличия двух аффинных слоев вверху, чтобы функции низкого уровня могли сопровождаться более сильными глобальными слоями.

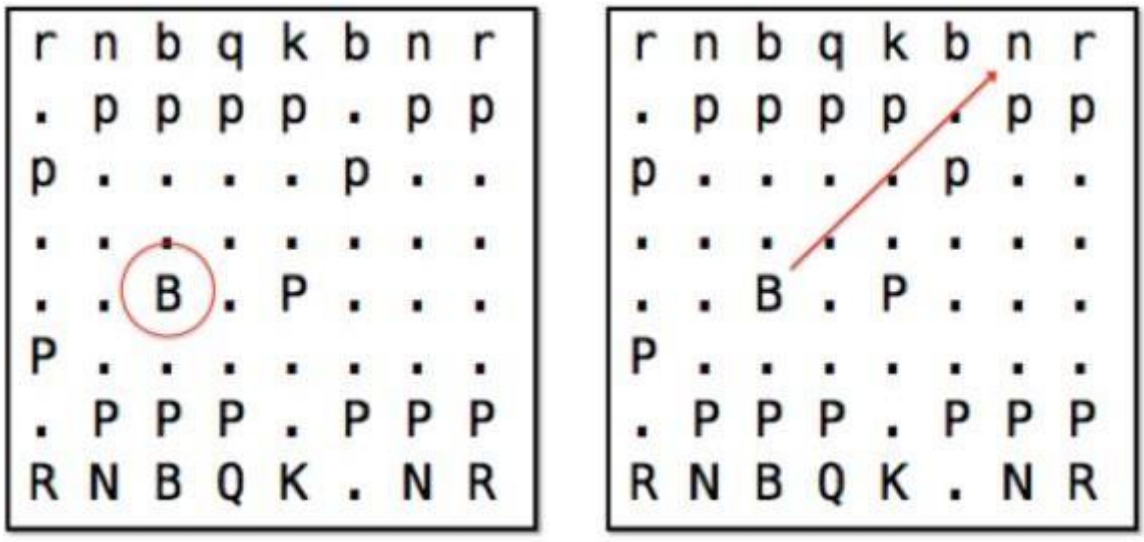


Рис. 2. Представление наборов данных по выбору фигуры и по выбору хода соответственно

2.3. Предобработка

Набор данных FICS состоит из шахматных партий в формате PGN, в котором записаны все ходы, сыгранные обоими игроками, в алгебраической шахматной нотации (SAN). Чтобы этот набор данных соответствовал нашему проекту, нотацию SAN необходимо было преобразовать в две координаты, представляющие позицию, из которой была перемещена фигура, и позицию, в которую она была перемещена. Затем эти ходы были разыграны на шахматной доске, чтобы получить представление доски в каждой позиции, закодированное в структуре данных, описанной выше. Затем используются два набора меток: координата, из которой была перемещена фигура, и координата, в которую она была перемещена.

2.4. План обучения

2.4.1 Выборка

Обучение сети происходит для более 200 000 ходов в более чем 20 000 играх, при этом нейросеть выбора фигур обучается на каждой фигуре, а сеть выбора каждого хода обучается на каждом ходе, сделанном соответствующей частью сети. Предполагается, что тренировочные данные отбирают широкий спектр шахматных ситуаций и стратегий.

2.4.2. Пулинг

Мы не используем слои пулинга, чтобы сохранить как можно больше данных во время обучения. Объединение в пул для изучения преобразований также не имеет значения, поскольку любое преобразование шахматного изображения оказывает огромное влияние на результат доски.

2.4.3. Инициализация весов

Важно отметить, что веса необходимо инициализировать очень низкими значениями, чтобы они соответствовали небольшим значениям данных, состоящим из -1, 0 и 1 во входном слое. При начальном обучении с высокой инициализацией входные данные не имели никакого влияния на окончательные оценки класса, а их общее влияние на прямое распространение подавлялось высокими весами. Перекрестно проверили порядок величины инициализаций, удалось определить, что 10^{-7} является оптимальной инициализацией для параметров в первом слое, используя более крупные инициализации в более глубоких слоях значением 10^{-6}, когда данные менее разрежены и чувствительны к плохим начальным прямым распространениям.

2.4.4. Регуляризация

Мы используем минимальное количество регуляризации. Поощрение сглаживания параметров, не сразу применимо к этой задаче, поскольку шахматы демонстрируют большую энтропию, чем распознавание изображений; однако было обнаружено, что некоторая регуляризация изначально повышает производительность.

2.4.5. Исключение (дропаут)

Как и в случае с регуляризацией, дропаут не соответствует этой задаче. Образ достаточно мал, поэтому все функции должны взаимодействовать друг с другом на каком-то уровне, так что исключение некоторых активаций неизбежно приведет к устранению важных закономерностей в прямом распространении. Кроме того, поскольку данных уже мало, отсев систематически удаляет при обучении фрагменты данных, которые очень необходимы в этойзадаче.

2.4.6 Функция потерь

Как уже было отмечено выше, была использована функция потерь softmax, чтобы ходы можно было интерпретировать как сделанные с вероятностью, а не как оценку по произвольной шкале. Это особенно важно, когда сочетание селектора фигур с селектором хода для двух произвольных гамм не может быть составлено вместе каким-либо осмысленным образом. Вероятности как выходные данные также полезны при интерпретации второго и третьего лучших ходов для наблюдения за другими предполагаемыми стратегиями алгоритма.

2.4.7. Обновление параметров

Было использовали обновление параметров RMSProp, чтобы подчеркнуть концепцию «уверенности» в обучении. Поскольку на силу обновления RMSProp влияет скользящее среднее величин недавних градиентов, это позволяет повторяющимся изменениям оказывать более сильное влияние на модель. Это также способствует тому, что окончательное распределение очков будет иметь более высокое стандартное отклонение, что отражает большую уверенность в нескольких ходах.

2.5. Тестирование

2.5.1. Валидация

В ходе проверки нужно сравнивать точность прогнозирования ходов с реальными ходами. Правильный прогноз показывает, что модель хорошо понимает, как игроки делают ходы. Если он не предсказывает реальные ходы правильно, это не обязательно означает, что он делает неидеальные или плохие ходы. Во-первых, точность проверки не измеряет, насколько «плохими» были неправильные прогнозы. Это похоже на иерархический классификатор, который ошибочно предсказывает «конкретный» класс, но правильно предсказывает класс более высокого уровня (например, «кошка», не указывая, какой тип кошки). Такой «иерархический» подход к измерению того, насколько шаг близок к обозначенному результату, невозможен, поскольку не существует показателя того, «насколько шаг близок к намеченному результату».

Во-вторых, при сравнении прогнозов с ходами других игроков не учитываются различия в стратегиях игроков. То есть, вычисляя средние значения игроков, нейросеть сама изучает «смешанный и усредненный» стиль игры игроков, у которых она обучалась. Это представляет собой одну из более широких проблем этого алгоритма: ходы реальных игроков иногда выполняются с расчетом на несколько ходов, но сеть обучается только на

одноуровневом просмотре вперед. Таким образом, сеть лучше всего работает при проверке в ситуациях с однозначными стратегиями. Более продвинутая реализация модели могла бы реализовать модель «биграммы», в которой она обучается для двух (или более) ходов одновременно для лучшего прогнозирования.

2.5.2. Тестирование против компьютера

Другой режим тестирования — это сопоставление алгоритма с компьютерным интеллектом, например с шахматным движком Sunfish. Чтобы сделать алгоритм играбельным, распределение вероятностей на доске 8 × 8, выдаваемое сетями, необходимо обрезать, чтобы представить действительный выбор фигур в сети выбора фигур и допустимые ходы в селекторе ходов. Сеть выбора частей отсекается путем поиска всех белых фигур на образе, и допустимые ходы фильтруются с использованием встроенных алгоритмов шахматной логики в модуле Python-Chess.

Заключение

На основе предложенной в статье архитектуры создан прототип нейронной сети для выбора хода при игре в шахматы, написаны процедуры предобработки данных для нее. В настоящее время проводятся эксперименты с обучением нейронной сети и тестированием результатов обучения. При тестировании, в частности, используется программа для решения той же задачи, созданная ранее автором на основе другого подхода (минимаксный алгоритм).

Литература

1. Erik Bernhardsson: Deep learning for chess. – Режим доступа:https://erikbern.com/2014/11/29/deep-learning-for-chess.html
2. Christopher Clark, Amos Storkey: Teaching Deep Convolutional Neural Networks to PlayGo – Режим доступа: https://arxiv.org/abs/1412.3409
3. Johannes Furnkranz: Machine learning in computer chess: The next generation – Режимдоступа: https://ofai.at/papers/oefai-tr-96-11.pdf
4. Sebastian Thrun: Learning to play the game of chess – Режим доступа: https://www.ri.cmu.edu/pub_files/pub1/thrun_sebastian_1995_8/thrun_sebastian_1995_8.pdf
5. A.L. Samuel: Some studies in machine learning using the game of checkers – Режимдоступа: https://gwern.net/doc/reinforcement-learning/model-free/1959-samuel.pdf
6. Felipe Calderan: Minimax algorithm applied to chess engines – Режим доступа: https://fvcalderan.github.io/myworks/articles/minimax_chess_engine.pdf

УДК 004.852

Исследование возможностей нейросетевой архитектуры YOLO-NAS для детекции людей на изображениях.

С. А. Рягузов

Воронежский государственный университет

Введение

По мере того, как мир становится все более цифровым, потребность в передовых методах компьютерного зрения, таких как обнаружение объектов, стала более насущной, чем когда-либо. Обнаружение объектов – это задача компьютерного зрения, которая включает в себя идентификацию и определение местоположения объектов на изображении или видео. Оно является важной частью многих приложений, таких как беспилотные автомобили, робототехника и видеонаблюдение.

Одна из самых популярных нейронных сетей для этой задачи является YOLO, новейшим релизом которой является YOLO-NAS, которая и рассматривается в этой статье.

Цель данного исследования заключается в изучении возможностей нейросетевой архитектуры YOLO-NAS в детектировании людей на изображениях.

1. Постановка задачи

Ставится задача по исследованию возможностей YOLO-NAS по детектированию людей на изображениях.

На входе система должна получать изображение, на котором могут находится люди, а на выходе давать информацию о том есть ли люди на изображении.

2. Исходные данные

Исходные данные представляют из себя видео для обучения, валидации и тестирования длительностью девять, четыре и одна минута (рис. 1).

Соответствующие видео были разбиты на кадры с помощью приложения FFmpeg. В итоге получаем две директории с тренировочными и валидационными данными, в тренировочном наборе 131 фото, в валидационном наборе 58 фото.

Далее происходит аннотирование изображений с помощью CVAT (Computer Vision Tool), инструмента для аннотации данных, который широко используется в области машинного обучения и компьютерного зрения (рис. 2).

Рис. 23. Видео для обучающей, валидационной выборки и тестирования

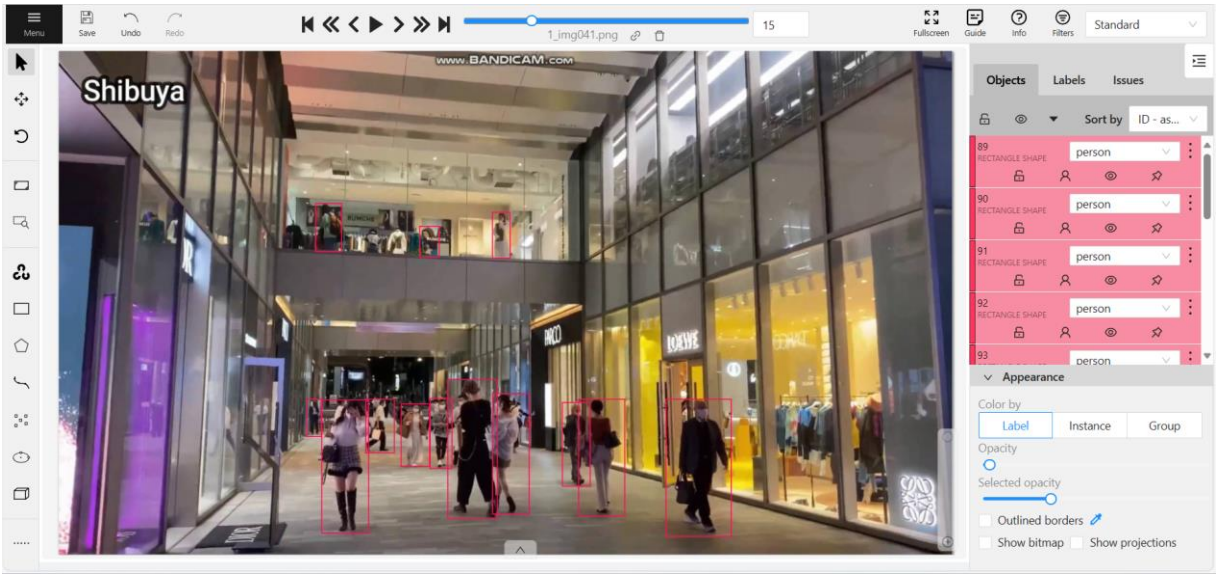


Рис. 2. Процесс аннотирования людей на фото

3. Обзор используемой модели

Для решения задачи по обнаружению людей на изображении была выбрана модель — YOLO-NAS-m (YOLO-NAS — название серии моделей; m — сокращение от medium, является указанием на размер модели). Данная модель предлагает сбалансированный подход, подходящий для обнаружения объектов с более высокой точностью, чем модели меньшего размера, но при этом сохраняя достаточную скорость работы для работы большенства приложений.

YOLO-NAS представляет собой революционную веху в эволюции моделей обнаружения объектов. Эта базовая модель, созданная с использованием передовой технологии поиска нейронной архитектуры (NAS), тщательно разработана, чтобы преодолеть ограничения своих предшественников YOLO. Уделяя основное внимание преодолению проблем и расширению границ производительности, YOLO-NAS представляет существенные улучшения, в частности, в поддержке квантования (уменьшения числа бит, используемых для представления весовых коэффициентов в модели) и улучшенном компромиссе между точностью и задержкой.

4. Архитектурные особенности YOLO-NAS

YOLO-NAS уникальным образом интегрирует блоки, поддерживающие квантование, и использует стратегию гибридного квантования, обеспечивая превосходную производительность, особенно в средах с ограниченными ресурсами. В этой архитектуре используются блоки QSP (Quantization-Specific Parameters) и QCI (Quantization-Centric Initialization), которые сочетают в себе преимущества повторной параметризации с 8-битным квантованием. Эти блоки, вдохновленные методологией, предложенной Chu et al., специально разработаны для минимизации потери точности во время квантования после обучения.

Ключевым аспектом подхода YOLO-NAS является селективное квантование, которое стратегически обрабатывает определенные части модели. Этот метод эффективно уравновешивает точность и задержку, одновременно снижая потери информации — распространенную проблему стандартных методов квантования, которые одинаково влияют на все слои, часто приводя к значительному снижению точности.

Метод гибридного квантования улучшает процесс квантования для сохранения точности. Это достигается путем выборочного квантования определенных слоев, в то время как другие слои остаются в исходном состоянии. Решение о том, какие слои квантовать, зависит от алгоритма выбора слоев. Этот алгоритм оценивает влияние каждого слоя на точность и задержку модели, а также тщательно рассматривает влияние переключения между 8-битным и 16-битным квантованием на общую задержку модели.

Отличительной особенностью YOLO-NAS является конструкция механизма обнаружения, которая предсказывает вероятность распределения для регрессии размера. Этот подход особенно полезен в сценариях, где размеры объектов значительно различаются. Предсказывая диапазон возможных размеров, а не один фиксированный размер, YOLO-NAS повышает точность обнаружения объектов различного масштаба. Это позволяет более тонко передавать данные от сложной, высокопроизводительной модели учителя к более простой и эффективной модели учащегося, что еще больше повышает практичность YOLO-NAS в различных приложениях.

YOLO-NAS имеет сложную пирамидальную систему, которая объединяет как нисходящие, так и восходящие информационные потоки. Этот конструктивный элемент расширяет возможности сети по сбору и эффективному использованию многомасштабной информации. В структуре «пирамида-внимание» нисходящий путь помогает фиксировать высокоуровневую семантическую информацию, в то время как восходящий путь фокусируется на более мелких деталях. Механизм внимания в этой пирамидальной структуре гарантирует, что сеть фокусируется на наиболее релевантных объектах в разных масштабах, тем самым улучшая свою способность обнаруживать объекты различных размеров и сложности.

Ключевой компонент в архитектуре YOLO-NAS, является продуктом передового процесса поиска сетевой архитектуры (NAS), в частности, с использованием запатентованной технологии Desi NAS AutoNAC, которая представляет собой движок оптимизации, разработанный Deci-AI. AutoNAC применяет Neural Architecture Search (NAS) для уточнения архитектуры модели глубокого обучения, чтобы повысить производительность модели при запуске на определенном оборудовании без ущерба, а иногда даже повышения точности модели (рис. 3).

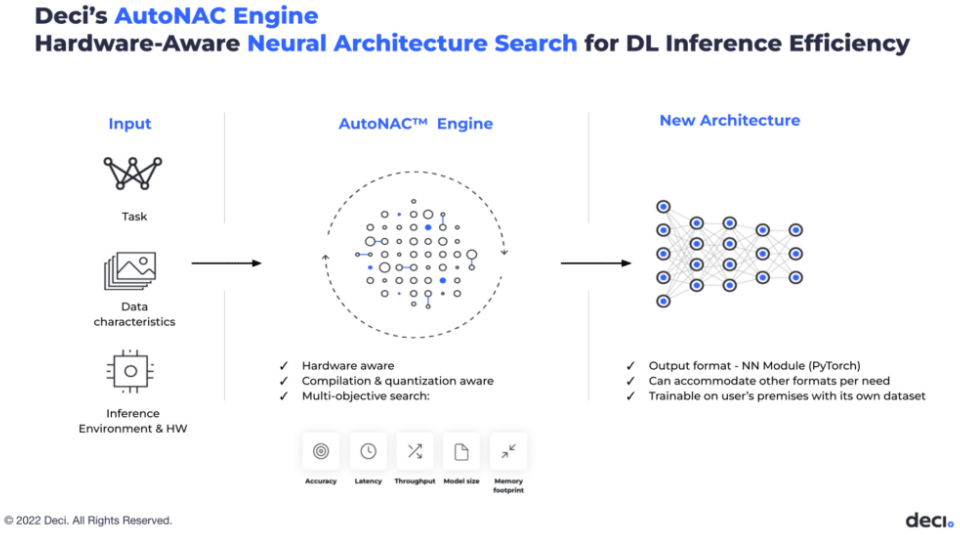


Рис.3. Архитектура AutoNAC

5. Обучение модели

Для обучения модели на собственных данных была выбрана модель YOLO-NAS-m.

Параметры для обучения модели приведены в табл. 1.

Таблица 1

Параметры для обучения модели

Параметр	Значение	Пояснение
`max_epochs`	50	Определяет количество эпох для обучения модели.
batch	16	Определяет количество изображений для обучения, которые будут обрабатываться перед обновлением внутренних параметров модели (рис. 4).
warmup_mode	linear_epoch_step	Определяет, как будет меняться скорость обучения, в данном случае, скорость обучения будет увеличиваться линейно с каждой эпохой.
optimizer	Adam	Адаптирует скорость обучения для каждого параметра в процессе обучения, в данном случае Adam используется с weght_decay равным 0.0001, что помогает предотвратить переобучение.
zero_weight_decay_on_bias_and_bn, ema	decay: 0.9 decay_type: threshold	Влияют на регуляризацию и улучшение обобщающей способности модели. Использование ema может помогает в улучшении качества обучения, особенно в условиях недостатка данных.
mixed_precision	True	Используется для ускорения процесса обучения и уменьшения потребления памяти за счет использования смешанной точности. Использование данного параметра обусловлено использованием графического процессора (GPU), который имеет специализированные аппаратные ресурсы для работы с числами различной точности.

Рис. 4. Структура и содержимое набора данных

6. Результат обучения модели

Метрики по обученной модели представлены на рис. 5.

Metric	Value
PPYoloELoss/loss_cls	0.7075137
PPYoloELoss/loss_iou	0.4490509
PPYoloELoss/loss_dfl	0.39254528
PPYoloELoss/loss	1.5491097
Precision@0.50	0.06286966055631638
Recall@0.50	0.9379084706306458
mAP@0.50	0.8527411222457886
F1@0.50	0.11784029006958008
Best_score_threshold	0.5900000333786011

Рис.5. Метрики по классам для обученной модели

Precision и Recall являются двумя ключевыми метриками, используемыми для оценки качества моделей классификации. Precision (точность) измеряет долю истинно положительных результатов среди всех положительных результатов, то есть, из всех предсказаний, сделанных моделью, сколько из них действительно верно. Recall (полнота) измеряет долю истинно положительных результатов среди всех реальных положительных результатов, то есть, из всех реальных положительных примеров, сколько из них было правильно идентифицировано моделью.

mAP (Mean Average Precision) является метрикой, используемой для оценки моделей обнаружения объектов. mAP вычисляется как среднее значение AP (Average Precision) для каждого класса, а затем усредняется по всем классам. AP рассчитывается как взвешенное среднее точностей на каждом пороге, где вес — это увеличение полноты от предыдущего порога. mAP учитывает компромисс между точностью и полнотой, а также учитывает как ложные положительные, так и ложные отрицательные результаты, что делает её подходящим показателем для большинства задач обнаружения.

F1-оценка является гармоническим средним между точностью и полнотой и используется для оценки баланса между этими двумя метриками. F1-оценка может быть использована для оценки качества модели, когда важно учитывать оба аспекта: как хорошо модель идентифицирует положительные примеры, так и как хорошо она избегает ложных срабатываний.

Precision@0.50 и Recall@0.50 на уровне 0.0628 и 0.9379 указывают на то, что модель имеет высокую полноту, но низкую точность.

mAP@0.50 равна 0.8527, что является довольно хорошим показателем.

F1@0.50 равна 0.1178, что указывает на низкую балансировку между точностью и полнотой.

Результат работы модели представлен на рис. 6.

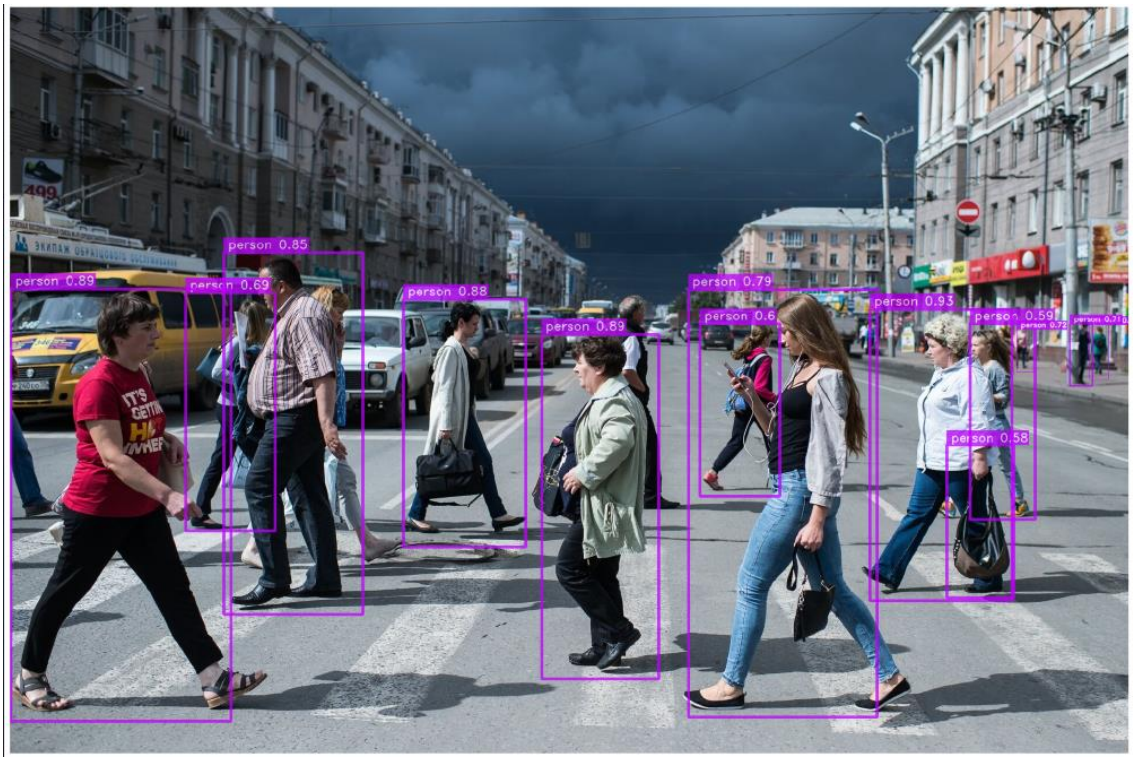


Рис.6. Результат работы модели

Заключение

В ходе проведенного исследования была рассмотрена и обучена модель YOLO-NAS-m, а также описаны её архитектурные особенности и параметры обучения. Обученная модель достаточно хорошо распознает людей на изображениях, на данном наборе данных.

Литература

1. Документация YOLO-NAS — URL: YOLO-NAS - Deci AI Documentation Hub (дата обращения 20.04.2024)

2. YOLOv8 против YOLO-NAS: Изучение усовершенствованного обнаружения объектов — URL: YOLOv8 vs. YOLO-NAS Showdown: Exploring Advanced Object Detection | Deci (дата обращения: 20.04.2024)

УДК 004.432.2

ПРОЦЕСС ОБРАБОТКИ И АНАЛИЗА ЭКСПЕРИМЕНТАЛЬНОГО НАБОРА ДАННЫХ. ОТБОР АТРИБУТОВ ДЛЯ КЛАССИФИКАЦИИ ПОДДЕЛЬНЫХ ПОДПИСЧИКОВ В СОЦИАЛЬНЫХ СЕТЯХ.

В. В. Савченко

Воронежский государственный университет

Введение

На сегодняшний день социальные сети являются неотъемлемой частью жизни как многих людей, так и многих компаний. Бизнес использует социальные сети для продвижения своих брендов и продуктов, а также для взаимодействия с клиентами. Социальные сети являются одним из наиболее распространенных и популярных средств коммуникации. При этом, важным показателем в социальных сетях является количество подписчиков, которое может быть использовано для определения влиятельности или популярности пользователя.

С ростом количества пользователей социальных сетей возрастает и интерес злоумышленников к данной сфере [1]:

Во-первых, поддельные аккаунты могут быть использованы для фишинговых атак, при которых злоумышленники пытаются получить доступ к конфиденциальной информации, например логину и паролю пользователя. Поддельные аккаунты могут использоваться для отправки фишинговых сообщений, которые могут включать вредоносные ссылки или приложения.

Во-вторых, поддельные аккаунты могут быть использованы для распространения ложной информации, которая может привести к панике, массовому истеризму или недоверию к организации, продукту или услуге. Это может быть особенно опасно для компаний, которые используют социальные сети для продвижения своих продуктов.

В-третьих, фейковые аккаунты могут быть использованы для спама, что может снизить эффективность социальной сети и ухудшить ее пользовательский опыт. Это может привести к оттоку пользователей и потере доверия к социальной сети.

В целом, поддельные аккаунты могут создавать серьезные угрозы для информационной безопасности, поэтому важно иметь инструменты, которые могут обнаружить и классифицировать такие аккаунты, а также разрабатывать стратегии для предотвращения их использования.

В данной работе будут рассмотрены некоторые из наиболее актуальных существующих функций и правил для обнаружения аномальных учетных записей в социальной сети, на основании базовых наборов данных реальных учетных записей и поддельных. Данный набор данных будет использоваться для обучения набора классификаторов машинного обучения, построенных на основе некоторых правил и функций.

1. Описание и первичная обработка экспериментального набора данных. Отбор атрибутов для классификации

В данной главе представлено описание и первичная обработка экспериментального набора данных, а также определяются атрибуты, которые будут использоваться для классификации поддельных подписчиков. Набор данных состоит из двух частей: набора

данных реальных аккаунтов и набора данных поддельных аккаунтов.

Описание и первичная обработка экспериментального набора данных — это процесс изучения и подготовки данных для дальнейшего анализа. Включает в себя проверку качества данных, обнаружение и устранение пропущенных значений, анализ выбросов, нормализацию данных и преобразование данных в формат, необходимый для использования в алгоритмах машинного обучения [2].

Отбор атрибутов для классификации — это процесс выбора наиболее важных атрибутов или функций из набора данных для использования в алгоритмах классификации [5]. Цель заключается в том, чтобы уменьшить количество атрибутов, которые нужно учитывать при построении модели, и увеличить точность и скорость ее работы.

1.1. Описание исходных данных для анализа поддельных подписчиков

В данной главе представлено описание исходных данных, которые были использованы для анализа поддельных подписчиков в социальных сетях. Наша цель заключается в разработке эффективных методов классификации и выявления поддельных аккаунтов среди реальных пользователей.

Для проведения исследования были использованы два набора данных: "genuine_accounts" и "fake_followers". Набор "genuine_accounts" содержит информацию о реальных аккаунтах, в то время как набор "fake_followers" содержит информацию о поддельных аккаунтах.

Оба набора данных представлены в формате CSV и содержат разнообразные атрибуты и характеристики, которые могут быть использованы для классификации аккаунтов. Набор "genuine_accounts" включает профильные данные, такие как имя пользователя, описание, фотографии профиля, а также данные о взаимодействии пользователей, например количество подписчиков, количество подписок и число публикаций. Набор "fake_followers" также содержит аналогичные атрибуты, но отличается от реальных аккаунтов в некоторых характеристиках, которые помогут разработать эффективные алгоритмы классификации.

Перед анализом данных оба набора были предварительно обработаны, включая удаление несущественных или повторяющихся атрибутов, а также приведение данных к единому формату и стандартизацию значений. Это позволит обеспечить удобство дальнейшей обработки и моделирования данных.

Далее будет рассмотрено содержание каждого набора данных.

1.2. Набор данных реальных аккаунтов

Набор данных "genuine_accounts" содержит информацию о реальных аккаунтах пользователей в социальной сети. В нем представлены 3474 записи (строки) и 40 столбцов, описывающих различные атрибуты.

Для расширения и улучшения набора данных, мы можем добавить новый атрибут с названием "is_fake", который будет иметь значение 0 для всех записей в наборе "genuine_accounts". Этот атрибут будет служить индикатором подлинности аккаунта: значение 0 будет означать, что аккаунт является реальным.

1.3. Набор данных поддельных аккаунтов

Набор данных “fake_followers” содержит информацию о поддельных аккаунтах пользователей в социальной сети. Он состоит из 3251 записей (строк) и 38 столбцов.

Для расширения и улучшения набора данных, мы можем добавить новый атрибут с

названием "is_fake", который будет иметь значение 1 для всех записей в наборе "fake_followers". Этот атрибут будет служить индикатором подлинности аккаунта: значение 1 будет означать, что аккаунт является поддельным.

2. Сравнительный анализ наборов данных

Цель данного анализа заключается в выявлении основных различий и характеристик между реальными и поддельными аккаунтами.

Количество записей: "genuine_accounts" содержит 3474 записи (строки), а "fake_followers" содержит 3351 запись (строк). Оба набора данных имеют значительное количество записей, что позволяет проводить статистические анализы и обучать модели машинного обучения на достаточно большом объеме данных.

На рис. 1 представлены графики распределения количества подписок. На рисунке слева – для реальных аккаунтов, справа – для поддельных.

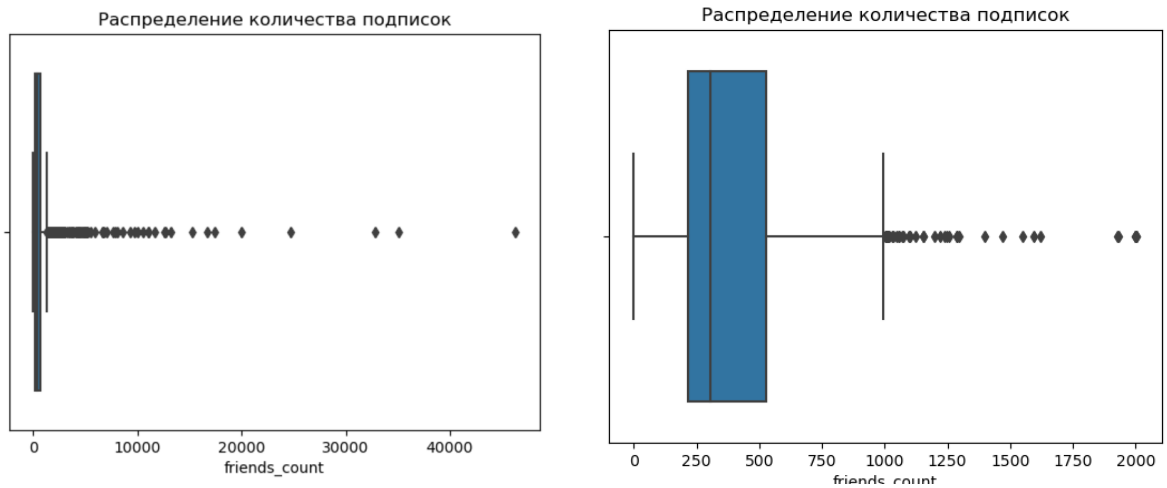


Рис. 1. Распределение количества подписок у пользователей

При сравнении количества подписок у реальных и поддельных аккаунтов обнаружено, что у поддельных аккаунтов количество подписок значительно больше по сравнению с реальными аккаунтами. Это наблюдение указывает на типичный паттерн поведения поддельных аккаунтов, которые часто стремятся установить максимальное количество подписок для привлечения внимания к своим аккаунтам или участия в массовых спам-действиях.

Анализ данных показывает, что медианное значение количества подписок у поддельных аккаунтов значительно выше, а также интерквантильный размах и верхний ус у поддельных аккаунтов шире по сравнению с реальными аккаунтами. Это свидетельствует о том, что большинство поддельных аккаунтов имеют очень большое количество подписок, превышающее обычное поведение реальных пользователей.

На рис. 2 приведены графики распределения подписчиков по временным зонам у реальных аккаунтов, а на рис. 3 у поддельных. На оси X отображаются временны зоны, а на оси Y – количество подписчиков, относящихся к каждой временной зоне. Каждому столбцу на графике соответствует определенная временная зона, а высота столбца указывает на количество подписчиков из этой временной зоны.

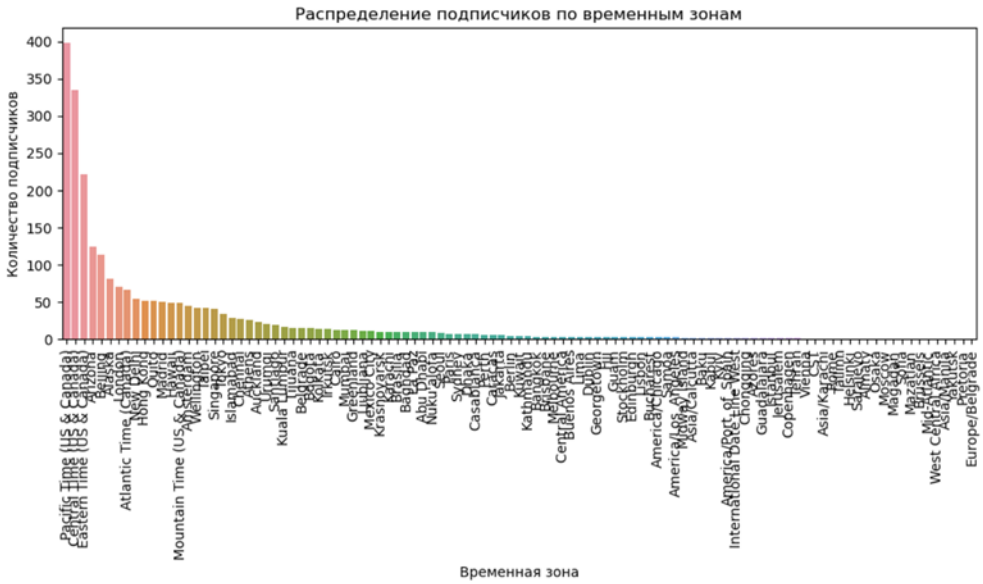


Рис. 2. Распределение подписчиков по временным зонам у реальных аккаунтов

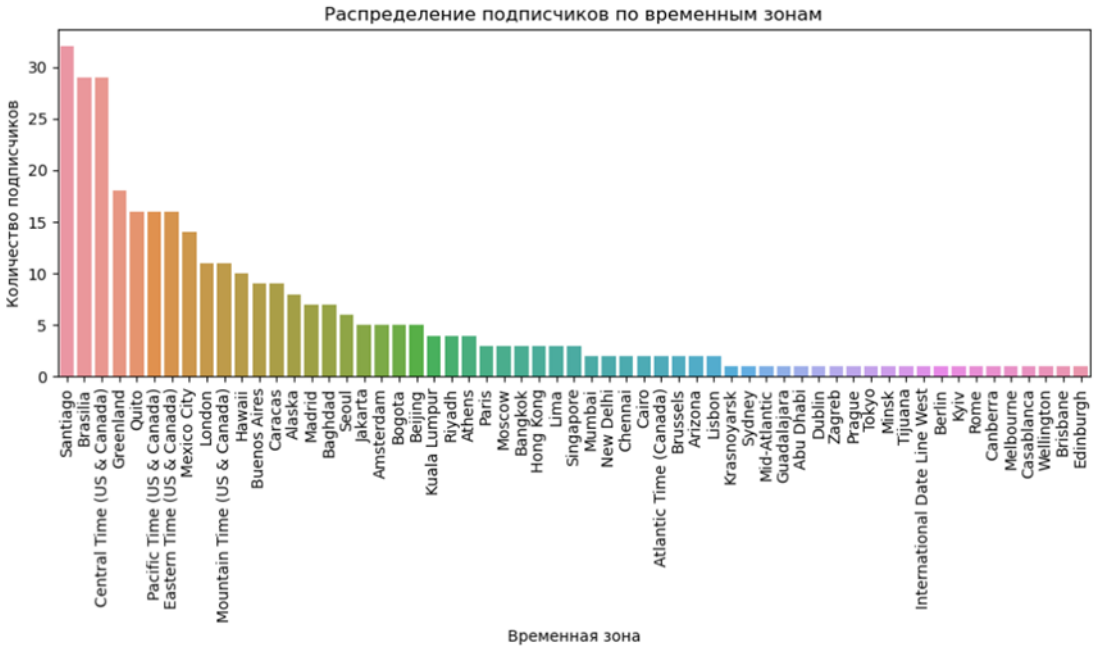


Рис. 3. Распределение подписчиков по временным зонам у поддельных аккаунтов

Графики позволяют наглядно определить распределение подписчиков в разных временных зонах. Более высокие значения некоторых столбцов на рис. 2 указывают на следующие аспекты:

- Реальные пользователи преобладают в некоторых регионах, так как там более распространена исследуемая социальная сеть, в то время как у поддельных аккаунтов распределение более плавное.
- Некоторые аккаунты реальных пользователей более популярны среди остальных пользователей, так как данные аккаунты привлекают большее внимание и

интерес среди других.

На рис. 4 представлен график сравнения общего числа количества подписчиков у реальных аккаунтов (слева) и поддельных (справа).

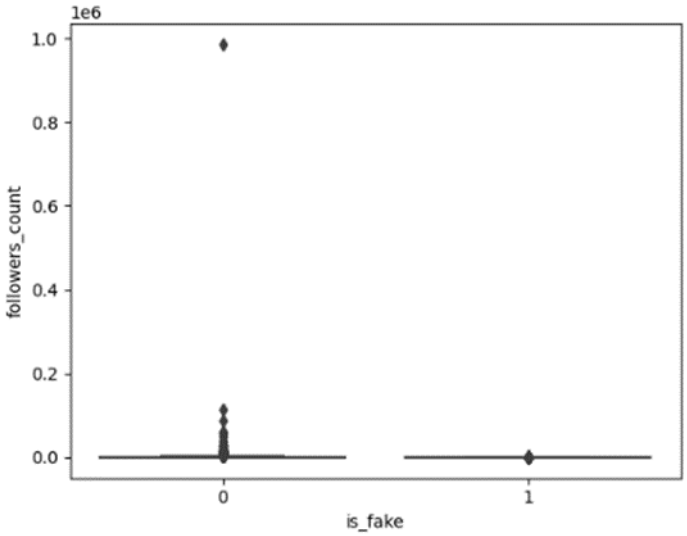


Рис. 4. Сравнение количества подписчиков

Данный график подтверждает, что аккаунты реальных пользователей являются более популярными, чем аккаунты поддельных пользователей.

По аналогии представлено сравнение количества публикаций у реальных (слева) и поддельных (справа) аккаунтов на рис. 5.

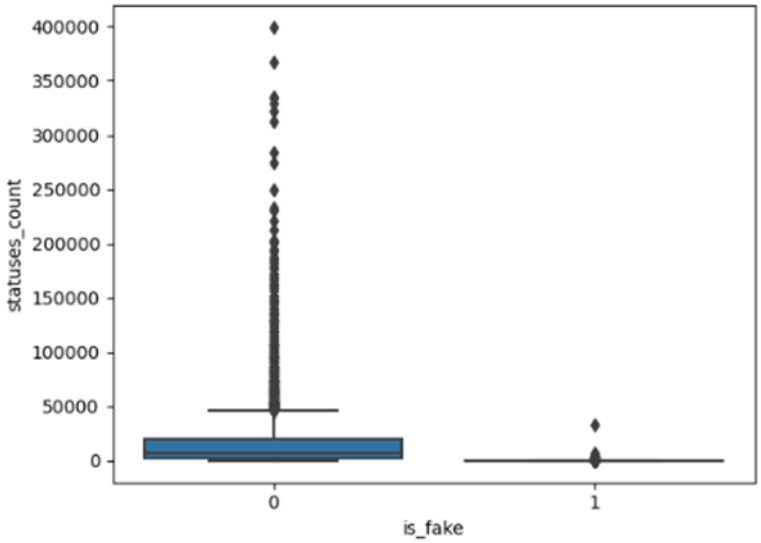


Рис. 5. Сравнение количества публикаций

Количество публикаций у реальных аккаунтов значительно выше количества публикаций у поддельных аккаунтов, что свидетельствует об активности реальных пользователей, их интересах, взаимодействии с аудиторией и маркетинговой стратегии.

Столбцы данных: Оба набора данных содержат информацию о различных атрибутах аккаунтов в социальных сетях. Эти атрибуты включают информацию о профиле пользователя

(имя, никнейм, описание, URL-адрес), статистике активности (количество публикаций, количество подписчиков и друзей), а также различные параметры и свойства профиля.

Типы данных: Оба набора данных содержат столбцы с разными типами данных, такими как целочисленные, числовые с плавающей запятой и строковые. Некоторые столбцы могут содержать булевы значения, указывающие на наличие или отсутствие определенных свойств у аккаунта.

Пропущенные значения: Оба набора данных содержат столбцы с пропущенными значениями. Например, некоторые столбцы, связанные с определенными свойствами профиля, могут быть заполнены только в некоторых случаях, что приводит к пропущенным значениям в остальных записях.

Для выполнения задач классификации данные наборы слиты в один, путем конкатенации, в итоговом наборе данных удалены атрибуты, в которых нет данных как в наборе genuine_accounts, так и в fake_followers. Итоговое количество записей в получившемся наборе данных 6825. Количество атрибутов – 37.

Сравнение количества реальных и поддельных аккаунтов в объединенном наборе данных представлено на рис. 6.

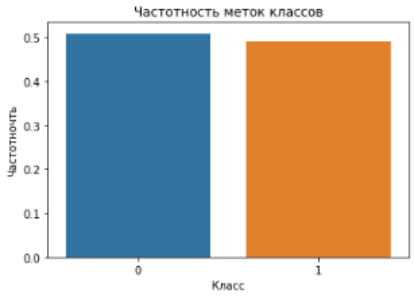


Рис. 6. Количество реальных и поддельных аккаунтов

3. Предварительная обработка данных. Отбор атрибутов для классификации

На рис. 7 представлена корреляционная матрица. Она показывает коэффициенты корреляции между парами атрибутов в наборе данных. Тепловая карта помогает визуально исследовать силу и направление связей между атрибутами [3]. Значения корреляции могут находиться в диапазоне от -1 до 1, где -1 указывает на полную отрицательную корреляцию, 1 – положительную корреляцию, 0 – отсутствие корреляции. Чем ярче цвет в ячейке, тем сильнее связь между соответствующими атрибутами.

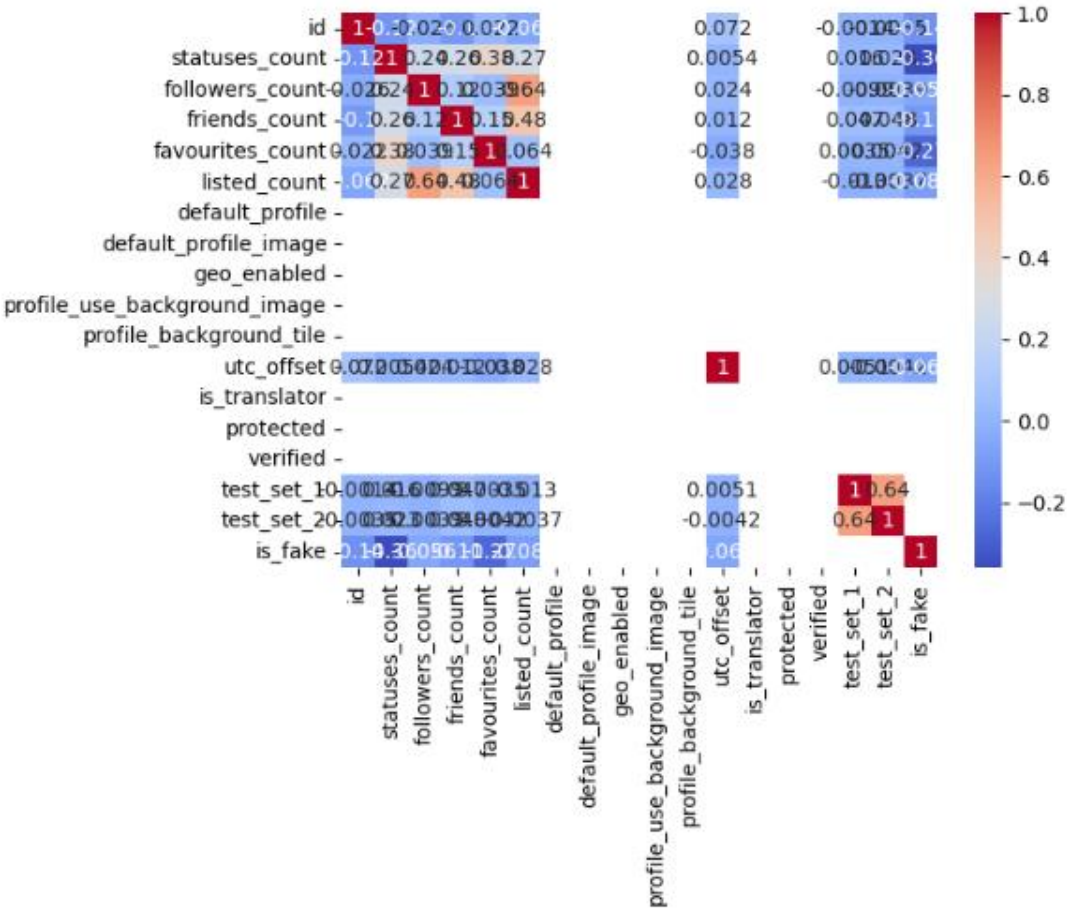


Рис. 7. Корреляционная матрица

На рис. 8 представлен график корреляции атрибутов с целевым атрибутом (“is_fake”) до проведения преобразований в наборе данных.

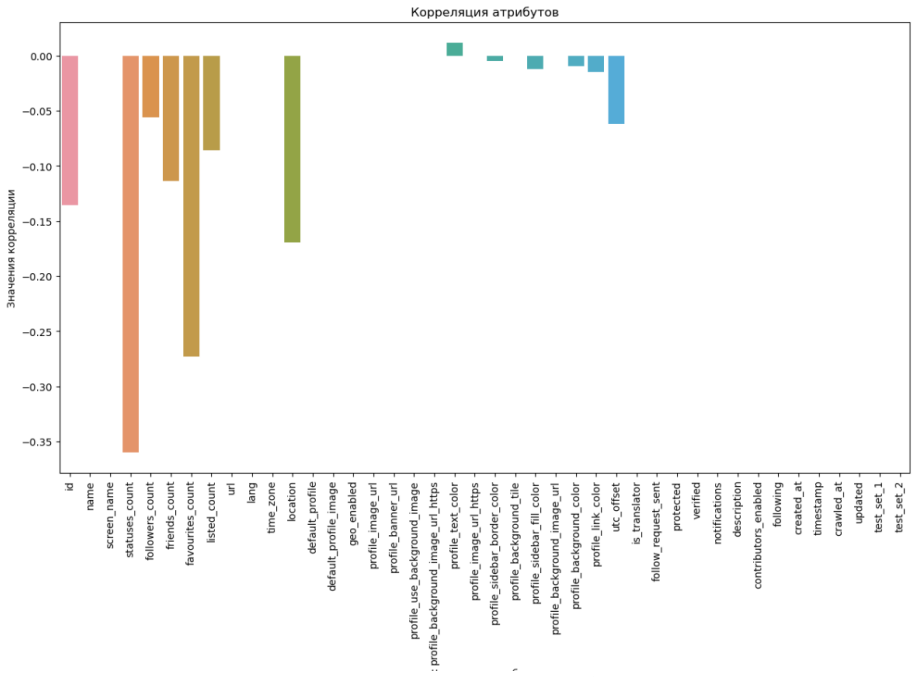


Рис. 8. Корреляция атрибутов

Как видно на рис. 8, большая часть атрибутов не коррелирует с целевым, по тем или иным причинам. Для дальнейшего анализа потребуется заполнить все пропущенные значения.

Пропущенные значения могут быть заполнены различными способами, включая удаление строк с пропущенными значениями, заполнение средними или медианными значениями, использование алгоритмов машинного обучения для предсказания значений и т.д. В данном случае для заполнения пропущенных значений был выбран метод заполнения средними значениями [4].

Для этого была проведена предварительная обработка данных, включающая удаление атрибутов, не имеющих значения для анализа, и преобразование категориальных атрибутов в числовые. Затем были заполнены пропущенные значения средними значениями по каждому атрибуту.

На рис. 9 представлен график корреляции атрибутов после проведения преобразований и заполнения пропущенных значений.

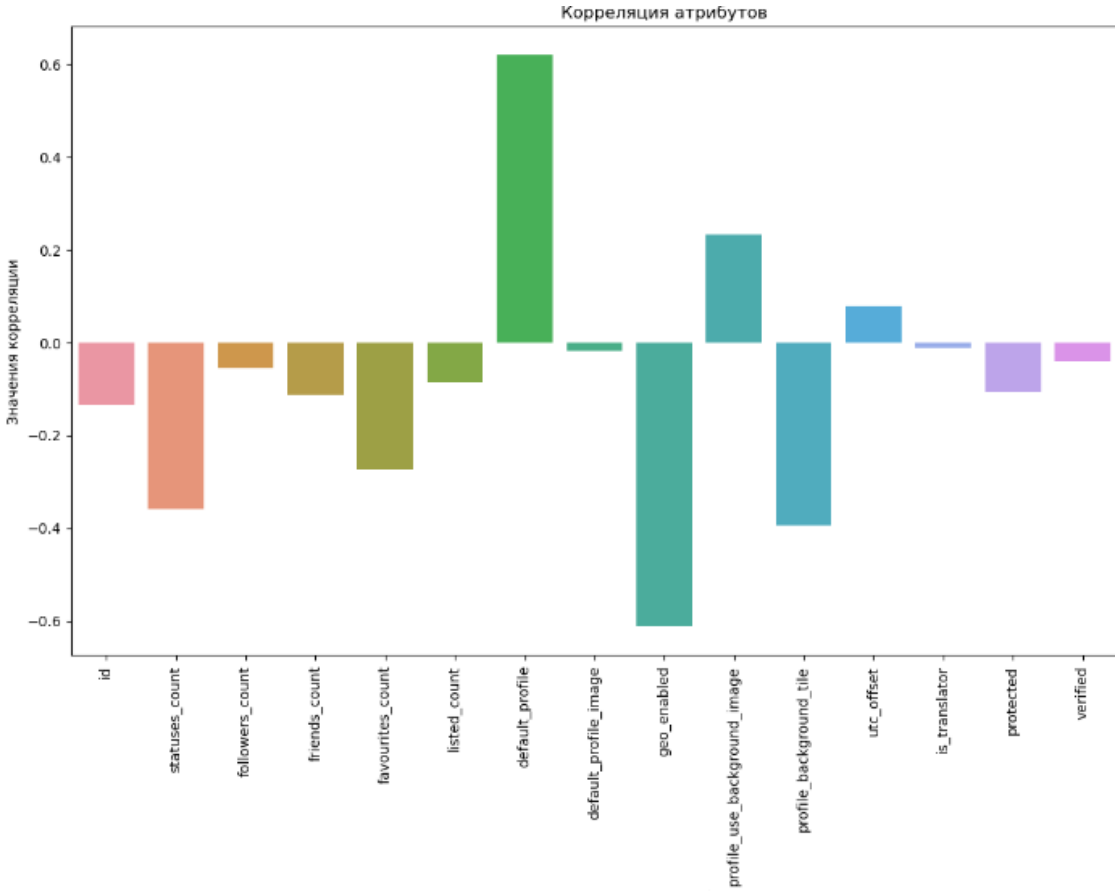


Рис. 9. Корреляция атрибутов после преобразований

Как видно на рис. 9, после проведения преобразований и заполнения пропущенных значений некоторые атрибуты стали коррелировать с целевым атрибутом (“is_fake”). Например, атрибуты “statuses_count”, “followers_count”, “friends_count” и “listed_count” имеют отрицательную корреляцию с целевым атрибутом, что может свидетельствовать о том, что поддельные аккаунты имеют меньшее количество статусов, подписчиков, друзей и списков, чем подлинные аккаунты. Атрибут “default_profile” имеет отрицательную корреляцию с целевым атрибутом, что может свидетельствовать о том, что поддельные не заполняют профиль по умолчанию.

Следовательно, проведение преобразований и заполнения пропущенных значений помогло выявить атрибуты, которые коррелируют с целевым атрибутом и могут быть использованы для предсказания подлинности аккаунта.

Таким образом для анализа данных важными являются:

- statuses_count – количество публикаций пользователя.
- followers_count – количество подписчиков.
- friends_count – количество аккаунтов, на которые подписан пользователь.
- favourites_count – количество сообщений, добавленных пользователем в избранное.
- listed_count – количество списков, в которых находится пользователь.
- time_zone – временная зона пользователя.
- default_profile– флаг, указывающий, имеет ли аккаунт стандартный профиль.
- profile_use_background_image – флаг, указывающий использует ли аккаунт фоновое изображение профиля (0 – нет, 1 да).
- geo_enabled – флаг, указывающий, включена ли геолокация у аккаунта.
- location – местоположение пользователя.
- utc_offset – смещение временной зоны пользователя в секундах.
- lang – язык аккаунта.
- is_fake – целевой признак, определяющий подлинность аккаунта.

На рис. 10 представлена информативность отобранных атрибутов.

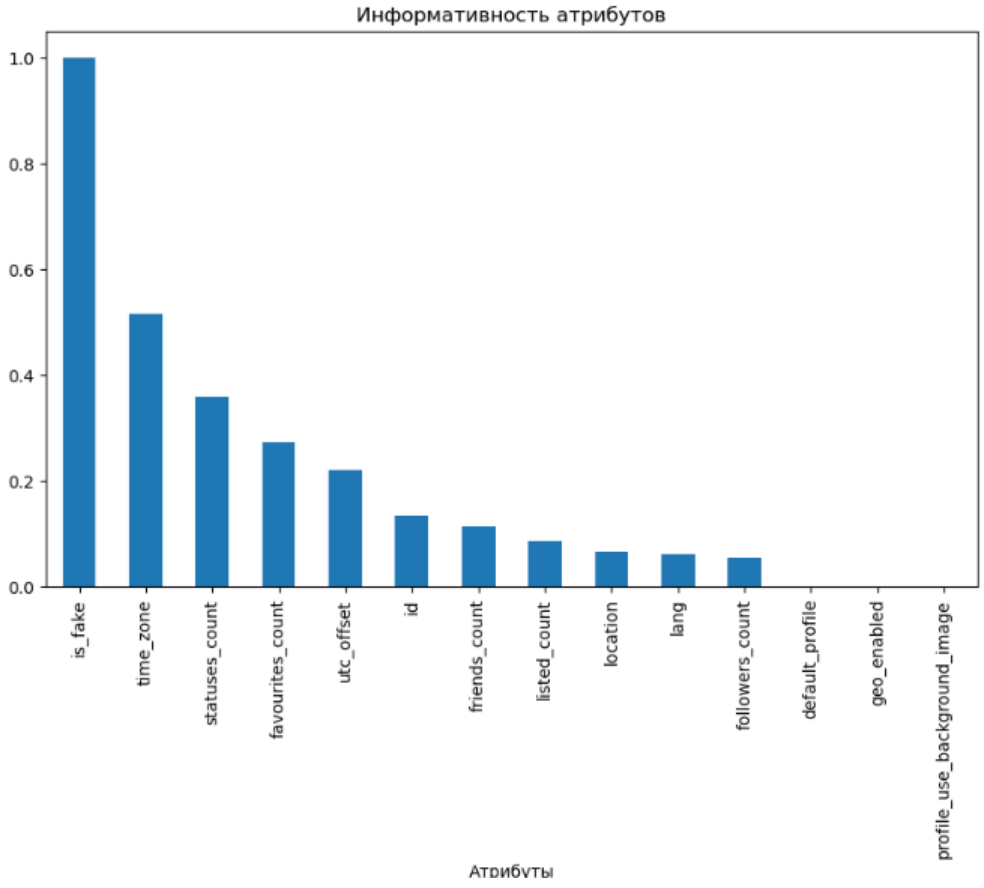


Рис. 10. Информативность атрибутов

Заключение

Проблема поддельных подписчиков является актуальной в социальных сетях и может иметь негативное влияние на доверие пользователей и безопасность социальной сети. Поэтому разработка эффективных алгоритмов классификации подделок имеет важное практическое значение.

В данной статье описывается процесс обработки и анализа экспериментального набора данных, а также отбора атрибутов для классификации поддельных подписчиков в социальных сетях. Для этого были использованы два набора данных: "genuine_accounts" и "fake_followers". Набор "genuine_accounts" содержит информацию о реальных аккаунтах, в то время как набор "fake_followers" содержит информацию о поддельных аккаунтах. После предварительной обработки данных, были определены основные атрибуты и характеристики, которые будут использоваться для классификации аккаунтов и выявления поддельных пользователей. В дальнейшем, на основе этих данных будет разработан эффективный метод классификации поддельных подписчиков в социальных сетях.

Литература

1. Wang, G., Mohanlal, M., Wilson, C., Wang, X., Metzger, M., Zheng, H., & Zhao, B. Y. (2013). Fame for Sale: Efficient Detection of Fake Twitter Followers. Proceedings of the 2013 IEEE International Conference on Communications (ICC), 1-6.
2. Wang, G., & Zeng, Q. (2019). The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race. Journal of Cybersecurity, 10(3), 457-473.
3. Pearson K. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling (англ.) // Philosophical Magazine, Series 5. — Vol. 50, no. 302. — P. 157—175.
4. MACHINE LEARNING: THE HAND BOOK // MACHINE LEARNING: THE HAND BOOK URL: https://mjlagattuta.github.io/Web_Class/2_WebPublication/Publication_v2/index.html (дата обращения: 04.03.2024).
5. Журавлёв, Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации. — Проблемы кибернетики: Вып.33. — 1978. — С. 5–68.

УДК 004.75

ПРЕДСТАВЛЕНИЕ САГИ В ВИДЕ КОНЕЧНОГО АВТОМАТА

А. И. Светашов

Воронежский государственный университет

Введение

В распределенных системах существуют такие системы, для которых критически важно поддержание согласованности данных и доступности системы. Например, в банков-ской сфере, электронной коммерции или в системах бронирования заказов. Одним из попу-лярных подходов для обеспечения этих свойств является использование шаблона Сага (Saga), который широко используется в архитектуре микросервисов (слабо связанных про-граммных модулей, расположенных на отдельных узлах или выполняющихся в отдельных виртуальных машинах).

Эта статья посвящена анализу возможности представления межсервисной транзакции в шаблоне Сага в виде конечного автомата и описанию необходимой информации о Саге в формате yaml.

1. Постановка задачи

Необходимо проанализировать возможность представления транзакции в шаблоне Сага в виде конечного автомата. Выделить свойства графа переходов состояний, определяющего Сагу.

Разработать правила описания Саги в yaml формате, которые однозначно определят состояния и переходы между состояниями системы для реализации транзакции. Учесть возможность валидации и интерпретации оркестратором описания Саги в заданном формате и последующего построения графа переходов состояний с гарантией выполнения необходимых свойств графа.

2. Анализ задачи

Сага — это шаблон для обработки транзакционных данных в распределенных системах. Он представляет собой последовательность транзакций, которые могут быть атомарно выполнены или компенсированы в отдельных сервисах.

Шаблон Сага может описать шаги процесса языком, близким к доменному. В основе Саги лежит конечный автомат. В случае оркестрированной Саги этот конечный автомат реализован в одном месте — оркестраторе. В случае хореографии Саги он распределен между компонентами системы. Для удобства описания Саги в едином формате и в одном файле будем рассматривать оркестрацию Саги для инкапсуляции логики транзакции в одном месте.

2.1. Детерминированный конечный автомат

Детерминированный конечный автомат состоит из следующих компонентов.

1. Конечное множество состояний Q.

2. Конечное множество входных символов $\sum$.
3. Функция переходов из одного состояния в другое $\delta(q,a)=p$, где q — текущее состояние автомата, a — входной символ, p — новое состояние, в которое автомат переходит из q при получении входного символа a.
4. Начальное состояние $s \in \Sigma$.
5. Множество конечных состояний $F \subset \Sigma$.

Нетрудно заметить, что в контексте конечного автомата Саги входными символами будут события — ответы от сервисов, а состояние будет определено в оркестраторе Саги. Функция переходов будет определена в оркестраторе Саги согласно заявленным правилам переходов, заданными изначально в файле определения Саги. Начальное состояние — это состояние Саги, при котором не поступало запросов на выполнение транзакции. В этом состоянии оркестратор ожидает запроса на инициирование транзакции. А множество конечных состояний определяет те состояния, при которых Сага успешно или неуспешно завершена.

Так как Сага в один момент времени может находиться только в одном состоянии, а переход между состояниями осуществляется при помощи событий, Сагу можно представить в виде графа. На рис. 1 представлен граф состояний Саги с двумя сервисами.

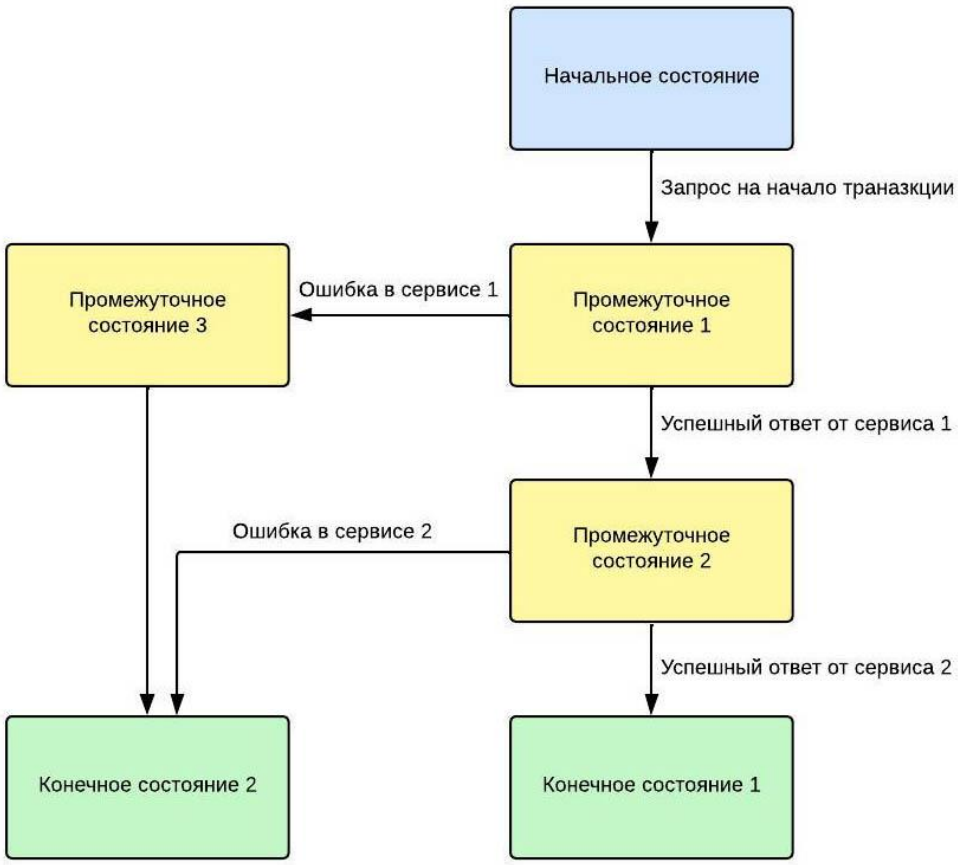


Рис. 1. Граф состояний Саги

Вершинами графа являются состояния Саги, а ребра представляют входные символы конечного автомата или события в Саге. Синим цветом отмечена вершина начального состояния. Зеленым цветом обозначены состояния, которые являются конечными, и при достижении которых Сага завершается. Желтым цветом обозначены состояния, которые не

являются конечными или начальными.

Рассмотрим свойства графа, определяющего Сагу. Граф является ориентированным и однонаправленным, так как Сага может переходить только в одно следующее состояние. Но так как Сага в конечном итоге должна прийти в одно из конечных состояний, то граф не должен содержать циклов. Переход из состояния p в состояние p недопустим, поэтому граф не должен содержать петель.

Сага должна быть завершена в конечном итоге, из этого следует, что все вершины, соответствующие конечным состояниям F должны быть достижимы из вершины начального состояния s.

Вершина графа v_s, соответствующая начальному состоянию s должна быть единственной, и ее полустепень захода должна быть равна нулю $\delta^-(v_s) = 0$.

Вершины графа V_F, соответствующие конечным состояниям F, имеют полустепень исхода равную нулю $\forall v \in V_F : \delta^+(v) = 0$, так как Сага, достигнув конечного состояния, завершается и более не может переходить в другие состояния.

Граф должен быть связным, так как любой путь начинается в начальной вершине, и заканчивается в конечной, а лишних состояний быть не должно.

2.2. Пример Саги процесса покупки товаров

Рассмотрим упрощенный пример Саги покупки товара для системы интернет-магазина. Система состоит из следующих сервисов.

1. Сервис заказов.
2. Сервис оплаты.
3. Сервис резерва.

Сервис заказов отвечает за регистрацию заказа пользователя в системе. Сервис оплаты предоставляет функции оплаты заказа и возврата денежных средств. Сервис резерва отвечает за резерв и выдачу товара.

Рассмотрим процесс покупки товара. В сервис заказов поступает запрос на создание заказа. Далее система должна вызвать метод для списания денег со счета покупателя в сервисе оплаты. Если оплата успешна, то система должна зарезервировать товар в сервисе резерва. Если оплата невозможна из-за недостатка средств, либо товара нет на складе, то заказ должен отмениться.

Для данной Саги детерминированный конечный автомат, описывающий процесс покупки товара с учетом возможных ошибок может выглядеть следующим образом:

1. Конечное множество состояний.
    1.1. Начальное состояние.
    1.2. Ожидание создания заказа.
    1.3. Ожидание оплаты.
    1.4. Ожидание резервирования.
    1.5. Ожидание возврата денег.
    1.6. Ожидание отмены заказа.
    1.7. Заказ зарезервирован.
2. Конечное множество входных символов.
    2.1. Покупка товаров.
    2.2. Заказ создан.
    2.3. Невозможно создать заказ.
    2.4. Заказ оплачен.

2.5. Ошибка при оплате.
2.6. Товар зарезервирован.
2.7. Ошибка резервирования.
2.8. Товар зарезервирован.
2.9. Возврат оформлен.
2.10. Заказ отменен.

3. Множество конечных состояний.
3.1. Заказ зарезервирован.
3.2. Начальное состояние.

Граф переходов состояний для данной Саги представлен на рис. 2.

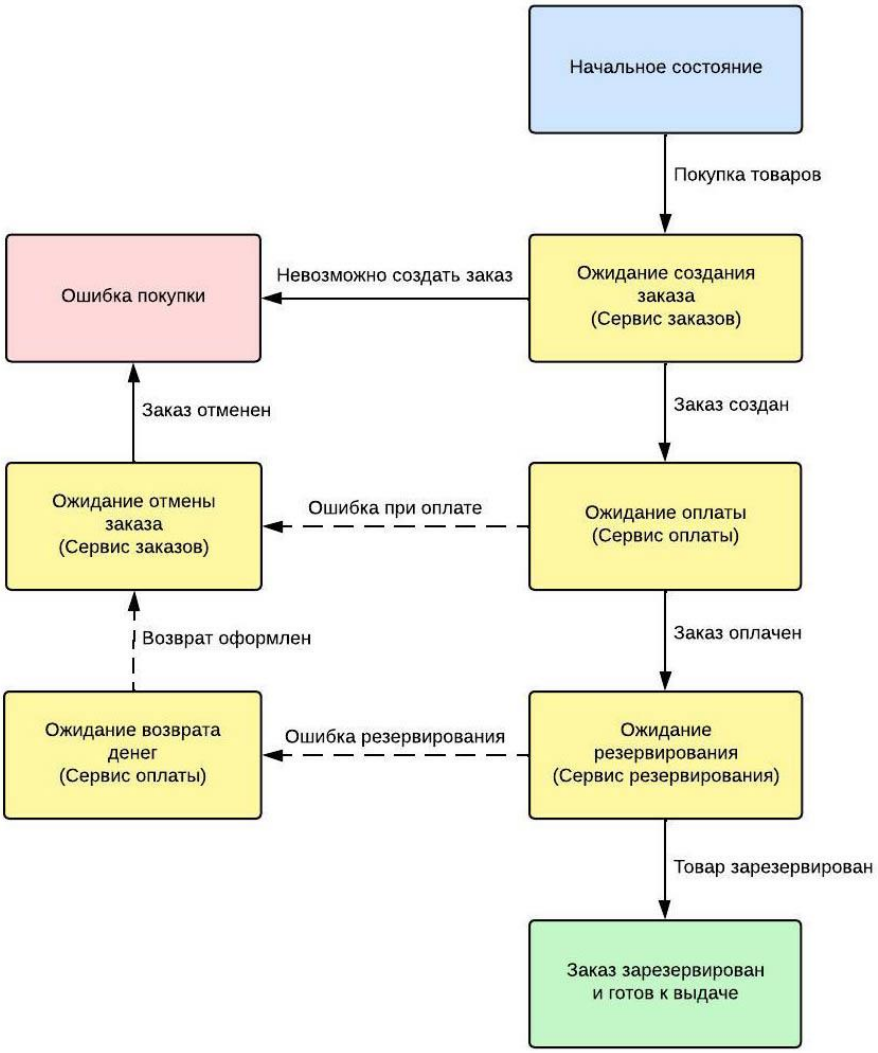


Рис. 2. Граф переходов состояний Саги процесса покупки товара

Пунктирными линиями обозначены переходы, при которых ожидается исполнение компенсационных действий.

3. Представление Саги в формате yaml

Так как любую сагу можно описать в виде конечного автомата, то опишем сагу, используя формат yaml.

1. *states* (состояния). Этот раздел содержит список состояний, которые может

принимать автомат. Каждое состояние представлено в формате:

- *name*, название состояния;
- *transitions*, список возможных переходов из этого состояния.

2. *transitions* (переходы). Этот раздел определяет возможные переходы между состояниями. Каждый переход представлен в формате:
   - *event*, событие, которое инициирует переход;
   - *target*, состояние, в которое происходит переход при возникновении события.
3. *initial_state* (начальное состояние). Этот элемент определяет начальное состояние автомата.
4. *final_states*. Этот элемент определяет конечные состояния автомата. После достижения любого из этих состояний автомат завершает свою работу.

Такая структура в формате yaml представляет собой структурированное описание конечного автомата, позволяющее легко понять его структуру и логику работы.

Рассмотрим пример описания Саги с такой структурой.

```
states:
  - name: Начальное состояние
    transitions:
      - event: Покупка товаров
        target: Ожидание создания заказа

  - name: Ожидание создания заказа
    transitions:
      - event: Заказ создан
        target: Ожидание оплаты
      - event: Невозможно создать заказ
        target: Ошибка покупки

  - name: Ожидание оплаты
    transitions:
      - event: Заказ оплачен
        target: Ожидание резервирования
      - event: Ошибка при оплате
        target: Ожидание отмены заказа

  - name: Ожидание резервирования
    transitions:
      - event: Товар зарезервирован
        target: Заказа зарезервирован
      - event: Ошибка резервирования
        target: Ожидание возврата денег

  - name: Ожидание возврата денег
    transitions:
      - event: Возврат оформлен
        target: Ожидание отмены

  - name: Ожидание отмены заказа
    transitions:
      - event: Заказ отменен
        target: Ошибка покупки

  - name: Заказ зарезервирован

  - name: Ошибка покупки

initial_state: Начальное состояние
final_states:
  - Заказ зарезервирован
  - Ошибка покупки
```

4. Проверка корректности yaml с описанием Саги

Получив описание Саги в yaml формате согласно заявленной структуре, оркестратор должен провалидировать его и построить на его основании граф переходов состояний.

4.1. Валидация входного yaml

В первую очередь файл с описанием должен соответствовать структуре, описанной в предыдущем разделе.

Необходимо проверить, что обязательно заполнено поле *initial_state*, в нем содержится только один элемент, и он содержится в разделе *states*.

Поле *final_states* должно содержать хотя бы один элемент. Все элементы из этого раздела должны содержаться в разделе *states*.

Все целевые состояния *target* в разделе *transitions* должны содержаться в разделе *states*. Проверив все эти утверждения, можно убедиться в корректности и перейти к формированию графа переходов состояний.

4.2. Формирование графа

Для формирования графа необходимо определить набор вершин и связать их между собой ребрами.

Для этого все состояния, описанные в разделе *states* входного yaml, вносятся в массив вершин с соответствующими метками из *name*. Далее, для каждой вершины находим раздел *transitions*, соответствующий метке вершины. Один элемент списка *transitions* является одним ребром, исходящим из вершины, и входящим в вершину, указанную в *target*. Меткой этого ребра является событие, указанное в *event*. Полученные ребра вносятся в массив ребер, а также у каждой вершины устанавливаются ссылки на входящие и выходящие состояния.

Вершины с метками, указанными в разделе *final_states*, помечаются булевым свойством как конечные для удобства. А вершина, соответствующая начальному состоянию *initial_state* помечается как начальная булевым свойством.

После построения графа необходимо проверить выполнение свойств графа, указанных в разделе 2.1.

Для проверки наличия циклов в графе можно использовать алгоритм обхода в глубину (DFS). Начиная с вершины, помеченной как начальной, начинаем обход графа в глубину. Каждую пройденную вершину помечаем как посещенную. Если при обходе находится вершина, уже помеченная как посещенная, то в графе есть цикл. Иначе, если таких вершин не было, граф ациклический. Наличие петель определить проще: если существует вершина, у которой есть связь с самой собой, это означает наличие петли.

Для проверки достижимости финальных вершин из начальной вершины используем алгоритм Дейкстры для поиска кратчайшего пути. Последовательно найдем пути из начальной вершины до каждой конечной. Если до какой-либо конечной вершины нет пути, то эта конечная вершина недостижима, и граф не может использоваться.

Полустепени захода и исхода начальной и конечных вершин можно определить по количеству исходящих и входящих в вершину. Количество входящих вершин в начальную должно быть равно нулю. А также количество исходящих вершин из конечных состояний должно быть равно нулю.

Проверить связность графа можно с помощью модифицированного алгоритма обхода в глубину, в котором будет возвращаться количество посещенных вершин. Запустив такой алгоритм от начальной вершины графа. Если результат равен количеству вершин графа, значит

все вершины графа достижимы из начальной. Это означает, что не существует других связных компонент кроме одной единственной.

Этот вид проверки позволяет не проверять отдельно достижимость всех конечных вершин из начальной, так как если они не достижимы, то число посещенных вершин графа в данном алгоритме будет меньше числа всех вершин графа. Также можно совместить проверку наличия цикла, на каждой вершине, проверяя, помечена ли она как посещенная. Если да, то найден цикл. Если нет, то увеличиваем число посещенных вершин и продолжаем алгоритм. Псевдокод алгоритма.

```
int dfs(u: int, visited: bool[]):
    int visitedVertices = 1
    visited[u] = true
    for v: (u, v) in E
        if not visited[v]
            visitedVertices += dfs(v, visited)
        else
            throw exception("Cycle found")
    return visitedVertices
```

Заключение

В статье была проанализирована возможность описания Саги в виде конечного автомата, а также описания переходов между состояниями как направленного ацикличного графа. Проанализированы свойства графа, которые обеспечивают формирование корректной Саги. Был предложен способ описания Саги в формате yaml, структура которого ссылается на определение детерминированного конечного автомата. Был рассмотрен процесс построения графа по содержимому yaml, а также представлены проверки выполнения необходимых свойств графа

Литература

1. Бёрнс Б. Распределенные системы. Паттерны проектирования / Б. Бёрнс – Санкт-Петербург : Питер, 2019. – 224 с.
2. Брауэр В. Введение в теорию конечных автоматов. / В. Брауэр – Москва : Радио и связь, 1987. – 392 с.
3. Ньюмен С. Создание микросервисов. / С. Ньюмен. – Санкт-Петербург : Питер, 2016. – 304 с.
4. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга / К. Ричардсон – Санкт-Петербург : Питер, 2022. – 544 с.
5. Хопкрофт Д. Введение в теорию автоматов, языков и вычислений / Д. Хопкрофт, Р. Мотвани, Д. Ульман – Москва : Вильямс, 2002. – 528 с.

УДК 004

Особенности реализации веб-приложения для поиска работы в IT-сфере

А. С. Свиридов

Воронежский государственный университет

Введение

На сегодняшний день многие фирмы для взаимодействия со своими клиентами «вышли в сеть». При помощи веб-приложений они смогли обрабатывать и выполнять их запросы. Веб-приложение может выступать посредником между компаниями и соискателями, например, при поиске работы в IT-сфере.

В настоящее время поиск работы соискателей в IT-сфере остается актуальным. IT-сфера динамически развивается, появляются новые специализации. Для того, чтобы оградить соискателя и компании от ненужных предложений, создаются узконаправленные сервисы.

Цель статьи – разобрать особенности реализации веб-приложения и используемые технологии, их взаимодействие друг с другом.

В статье рассматриваются особенности реализации веб-приложения для поиска работы в IT-сфере. В ходе работы будет выбран и обоснован архитектурный стиль проектирования приложения, перечислены ее особенности. Рассмотрены составляющие веб-приложения, современные технологии реализации: СУБД PostgreSQL, фреймворк Spring Boot, Spring Security, библиотека JavaScript React и HTTP-клиент Axios.

1. Структура работы веб-приложения

Серверная часть приложения реализована на Spring Boot и Spring Security [1]. Она представляет собой веб-службу и работает в стиле REST (Representation State Transfer), который предполагает применение различных типов HTTP-запросов для взаимодействия с клиентской частью приложения.

Клиентская часть приложения реализована на JavaScript-библиотеке с открытым исходным кодом React для создания пользовательских интерфейсов. В клиентской части страница отображается при помощи компонентов. Переход между страницами осуществляется при помощи стандартной библиотеки маршрутизации (Router). В сервисах вызываются методы библиотеки Axios, которые отправляют HTTP-запрос северной части.

Запросы в серверной части проходят через специальные фильтры, в которых происходит валидация данных, определяющих пользователя. В случаи успешной проверки запрос переходит в контроллеры, затем вызываются методы сервисов, в которых происходит логика работы приложения. Сервисы обращаются к репозиториям, имеющие доступ к данным.

На рис. 1 показана выбранная структура работы веб-приложения для поиска работы в IT-сфере.

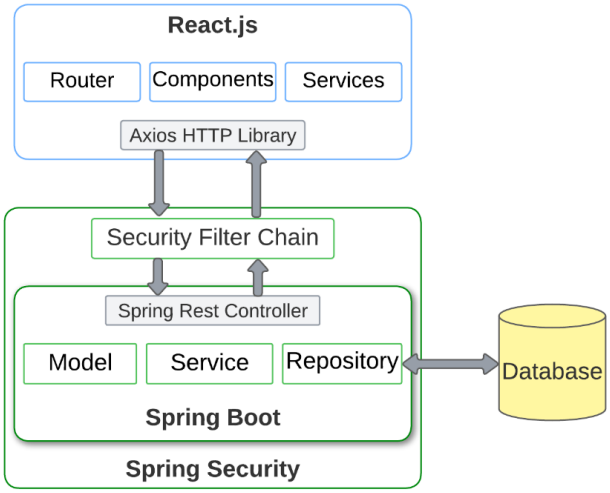


Рис. 1. Структура работы веб-приложения

Этот стиль может иметь следующие особенности:

- система должна быть разделена на серверную и клиентскую части;
- применяются различные типы HTTP-запросов для взаимодействия с клиентской частью веб-приложения;
- сервер не должен хранить какую-либо информацию о клиентах.

2. База данных

Для хранения данных на этапе реализации веб-приложения был выбран PostgreSQL – объектно-реляционная база данных [2]. Эта СУБД представляет все данные в виде объектов, имеет иерархию ролей. PostgreSQL имеет следующие преимущества:

- поддержка баз данных неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- лёгкая расширяемость.

Ориентируясь на перечисленные преимущества, PostgresSQL больше подходит для выбранной темы, чем другие СУБД. Например, MySQL, который недостаточно надёжен при работе с данными, и SQLite, у которого отсутствует система пользователей.

3. Серверная часть приложения

Spring Framework представляет собой контейнер внедрения зависимостей, с несколькими удобными слоями, которые позволяют быстрее и комфортнее создавать веб-приложения на языке программирования Java.

Авторы Spring предоставили некоторые утилиты, которые автоматизируют процедуру настройки и ускоряют процесс создания и развертывания Spring-приложений, под общим названием Spring Boot.

Spring Boot – это проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать веб-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода. Каждое Spring Boot веб-приложение включает встроенный веб-сервер.

Создание автономных веб-приложений со встроенными серверами не только удобно для разработки, но и является допустимым решением для приложений корпоративного уровня и становится все более полезно в мире микросервисов.

Для разделения пользователей по ролям, которые ограничивают его от обращений к не нужным для него ресурсам, используют Spring Security.

Spring Security – это фреймворк, обеспечивающий аутентификацию, авторизацию и защиту от распространенных атак. Благодаря первоклассной поддержке защиты императивных и реактивных приложений, он является стандартом де-факто для защиты приложений на базе Spring [3].

Запросы перед попаданием в контроллеры проходят через цепочку фильтров, один из которых отвечает за авторизацию пользователя. В этом классе происходит валидация данных, содержащие токен доступа и обновления в формате JSON. Для обработки ошибок во время обхода цепочки фильтров Spring Security предлагает реализовать интерфейс AuthenticationEntryPoint. Интерфейсы помогают достичь слабой связанности в программе [4]. Если валидация прошла успешна, то создаётся объект аутентификации, который сохраняется в контексте безопасности, вызывая специальный обработчик.

Объект аутентификации содержит данные, созданные при помощи реализации специального сервиса UserDetailsService. Сервис имеет единственный метод, который возвращает объект класса, реализующий интерфейс UserDetails. Этот объект основывается на данных пользователя, полученные из фильтра авторизации. Главной особенностью этого объекта является то, что он содержит роли.

Роли участвуют в разделении функциональности веб-приложения. В специальном конфигурационном файле прописываются адресы запросов и роли, которыми должен обладать авторизованный пользователь.

Основная логика работы приложения находится в сервисах, которые вызываются через специальные контроллеры. Сервисы, в свою очередь, взаимодействуют с репозиториями, путём вызова методов репозитория в методе сервиса. Репозиторий является интерфейсом, который расширяет интерфейс JpaRepository. JPA (Java Persistence API) – это стандартная технология, которая позволяет сопоставлять объекты с реляционными базами данных.

4. Клиентская часть приложения

Клиентская часть приложения реализована при помощи React. React – это JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. Важной особенностью библиотеки React является виртуальный DOM [5]. Это объект, в котором содержится информация о состоянии интерфейса. Например, после того как пользователь отправляет данные в форме на сервер, React определяет изменение состояния и обновляет внешний вид интерфейса.

React состоит из компонентов, которые являются самодостаточными элементами. Они могут быть использованы на странице неограниченное количество раз. Создавая небольшие фрагменты кода для определённых задач, которые можно использовать в разных частях приложения. Т.к. компоненты являются самодостаточными элементами узкого назначения, их можно использовать для разделения кода на логические структурные составляющие.

Заключение

В данной статье были определены технологии для реализации веб-приложений, разобраны их взаимодействие друг с другом, особенности реализации веб-приложения для поиска работы в IT-сфере. Например, разделение пользователей на роли, которая реализуется через Spring Security. Рассмотренные технологии помогут разработчику при создании других

узкопрофильных приложений.

Литература

1. Документация Spring. – URL: https://spring.io/guides/gs/securing-web/ (дата обращения: 13.04.2024).
2. PostgreSQL. – URL: https://www.postgresql.org/ (дата обращения: 16.04.2024).
3. Уоллс К. Spring в действии / К. Уоллс. – Москва : ДМК Пресс, 2022. – 544 с.
4. Шилдт Г. Java 8: руководство для начинающих / Г. Шилдт. – Москва : Вильямс, 2015. – 720 с.
5. Документация React. – URL: https://ru.reactjs.org/docs/ (дата обращения: 16.04.2024).

УДК 004.85

Исследование методов решения задачи интеллектуальной проверки орфографии в тексте

М. П. Свиридов

Воронежский государственный университет

Аннотация. Работа посвящена исследованию различных методов решения задачи интеллектуальной проверки орфографии в тексте, включая использование нейронных сетей. Результаты данной работы могут быть полезными для различных организаций и компаний, которые используют автоматическую проверку орфографии в своей работе.
Ключевые слова: искусственный интеллект, машинное обучение, нейронные сети, рекуррентные нейронные сети.

Введение

Во все времена правильное написание текстов было ключевым фактором для обеспечения качественной коммуникации между людьми, особенно в современном мире, где информация является ключевым фактором развития и успеха, точность и качество анализа текстовых данных при работе с документами и отчетами играют решающую роль. Однако, даже при наличии большого опыта и знаний в языке, ошибки в орфографии могут возникать у любого человека. В связи с этим, появляется необходимость использования специальных инструментов для проверки орфографии.

Существует множество методов решения задачи проверки орфографии в тексте. Например, правила проверки орфографии, словарные методы, алгоритмы на основе статистики и машинного обучения. Однако, с развитием технологий, нейронные сети стали популярным инструментом для решения данной задачи.

Нейронные сети (NN) – это компьютерные системы, которые имитируют работу человеческого мозга. Они используются для решения множества задач, таких как распознавание образов, классификация данных, прогнозирование и т.д. В настоящее время, нейронные сети используются во многих областях нашей жизни, таких как медицина, финансы, транспорт и т.д. Они помогают улучшить качество жизни людей и повысить эффективность работы различных организаций.

В контексте проверки орфографии, нейронные сети могут использоваться для моделей, которые могут распознавать и исправлять ошибки в тексте. Использование нейронных сетей для решения задачи проверки орфографии может значительно улучшить качество автоматической проверки и сделать ее более точной и эффективной. Методы решения задачи автоматизированной проверки орфографии широко используются в современных текстовых редакторах и других программах, где необходимо проверять правильность написания слов. Эти методы позволяют автоматически обнаруживать и исправлять орфографические ошибки, что значительно упрощает процесс редактирования текста и повышает его качество.

1. Методы решения задачи проверки орфографии

Методы решения задачи автоматизированной проверки орфографии широко используются в современных текстовых редакторах и других программах, где необходимо проверять правильность написания слов. Эти методы позволяют автоматически обнаруживать

и исправлять орфографические ошибки, что значительно упрощает процесс редактирования текста и повышает его качество.

Существует несколько методов решения задачи проверки орфографии:

1. Методы на основе статистических моделей: эти методы используются для определения вероятности правильности написания слова на основе его контекста. Они основаны на анализе большого количества текстов и вычислении частоты появления слов в различных контекстах. На основе этого анализа создается модель, которая может автоматически определять, является ли слово написанным правильно или нет. Примером такой системы является Hunspell, которая используется во многих текстовых редакторах и браузерах.
2. Методы на основе правил: эти методы используются для определения правильности написания слова на основе грамматических правил и правописания. Они могут быть использованы для проверки орфографии в текстах, написанных на языках с жесткими правилами правописания. Эти правила могут быть созданы экспертами-лингвистами или извлечены из словарей. Примером такой системы является языковой пакет LanguageTool, который использует правила для проверки орфографии, грамматики и пунктуации.
3. Методы на основе машинного обучения: эти методы используются для определения вероятности правильности написания слова на основе обучения нейронной сети на большом количестве текстов как с правильно написанными словами, так и с ошибочными. Нейронные сети и другие алгоритмы машинного обучения используются для создания модели, которая может автоматически обнаруживать и исправлять орфографические ошибки в тексте. Примером такой системы является система Grammarly, которая использует нейронные сети для проверки орфографии, грамматики и стиля письма.
4. Методы на основе комбинации различных подходов: эти методы используют комбинацию статистических моделей, правил и машинного обучения для достижения наилучшего результата при проверке орфографии. Принцип работы таких методов заключается в том, что они анализируют текст, используя несколько различных алгоритмов и моделей, а затем объединяют результаты для получения более точного результата. Например, система может использовать модель на основе статистических данных для определения часто встречающихся ошибок, модель на основе правил для проверки специфических правил грамматики и модель на основе машинного обучения для обнаружения новых ошибок.

2. Методы на основе машинного обучения

В рамках данной работы будут подробнее рассмотрены методы машинного обучения.

Методы на основе машинного обучения используются для проверки орфографии путем обучения нейронной сети на большом количестве текстов. Эти методы могут быть разделены на две категории: методы на основе моделей и методы на основе примеров.

1. Методы на основе моделей используются для создания модели, которая описывает правильное написание слова. Эта модель может использоваться для определения вероятности правильного написания слова на основе его контекста. Одним из примеров такой модели является скрытая марковская модель (СММ) [1], которая используется для описания последовательности событий, таких как последовательность букв в слове. СММ может быть обучена на большом количестве текстов, чтобы определить вероятность правильного написания слова на основе его контекста. Схематически работу данного примера можно представить следующим образом (рис. 1). Из рисунка видно, что на основе предыдущего шага выбирается часть речи, которая с грамматической точки зрения будет корректно подходить о контексту.

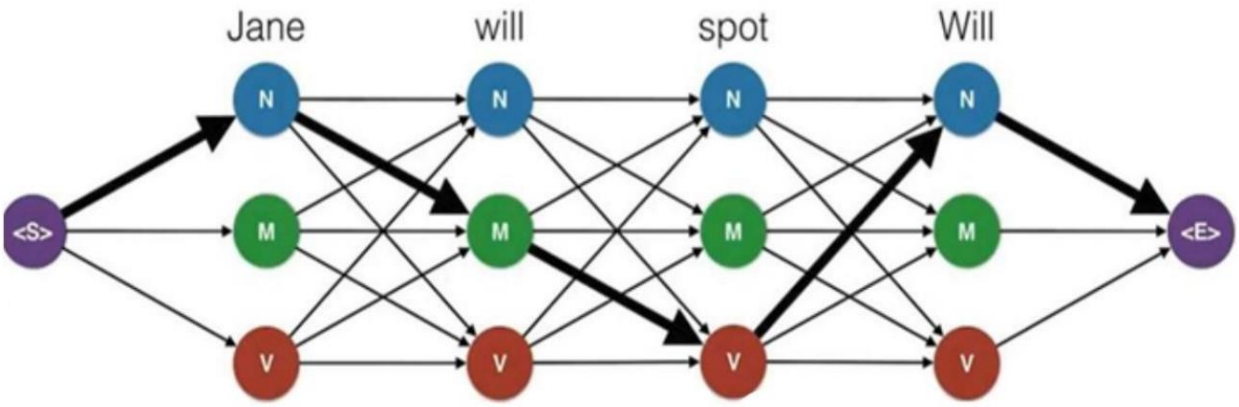


Рис. 1. Схема работы СММ

2. Методы на основе примеров используются для обучения нейронной сети на большом количестве примеров правильного и неправильного написания слов [2]. Нейронная сеть может использоваться для определения вероятности правильного написания слова на основе его контекста или в целом. Одним из примеров такой метод является метод опорных векторов (SVM), который используется для классификации данных на основе их признаков. Визуально его можно представить как две группы предметов (признаков), на основании расположения которых строится предположение об их возможном расположении в будущем (рис. 2)

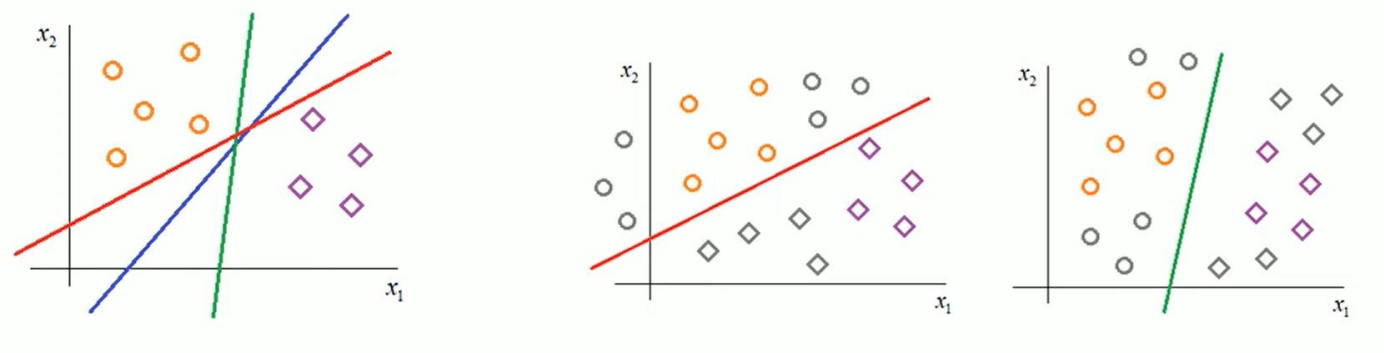


Рис. 2. Схематичное представление SVM

Заключение

В статье приведен обзор методов решения задачи автоматизированной проверки орфографии. Более подробно в данной работе освещены методы на основе машинного обучения, такие как: методы на основе моделей и методы на основе примеров, а также приведены иллюстрационные примеры их работы.

На основе описанных методов будет разработан программный комплекс и проведен вычислительный эксперимент, который позволит определить наиболее эффективный подход к автоматизированной проверке орфографии.

Литература

1. Электронный научно-практический журнал «Современные научные исследования и инновации» [Электронный ресурс] – URL: https://web.snauka.ru/issues/2017/05/72892
2. С. Николенко Скрытые марковские модели 2021 г. – 108 с.
3. А. Бурков Машинное обучение без лишних слов 2020 г. – 192 с.

УДК 004.946

АНАЛИЗ И СРАВНЕНИЕ ПЛАТФОРМ ДЛЯ СОЗДАНИЯ AR-ПРИЛОЖЕНИЙ VUFORIA С ARKIT, ARCORE

Д. К. Сидоренко, С.Ю. Болотова

Воронежский государственный университет

Введение

Развитие технологий дополненной реальности (AR) преобразовало множество сфер деятельности, от игр и развлечений до образования и промышленности. Прогресс в этой области открывает новые возможности для взаимодействия с информацией и окружающим миром, делая виртуальные объекты частью нашей повседневной жизни. Важной составляющей успеха AR-приложений является выбор подходящей платформы для их разработки. На текущий момент одной из самых популярных платформ для создания приложений AR является Unity.

Unity, являясь ведущей кроссплатформенной средой разработки игр и интерактивных приложений, позволяет разработчикам интегрировать различные AR-технологии, такие как Vuforia, ARKit и ARCore, предоставляя мощные инструменты для реализации сложных нефункциональных и функциональных требований к приложению дополненной реальности. Понимание технических характеристик, преимуществ и ограничений каждой платформы критично для выбора наиболее подходящего инструмента для конкретного проекта.

Vuforia Engine выделяется своей способностью к распознаванию и отслеживанию различных объектов и маркеров, что делает ее привлекательной для приложений, требующих взаимодействия с физическими объектами в реальном мире. ARKit и ARCore, напротив, сосредоточены на создании более реалистичных и плавных AR-взаимодействий, используя сложные алгоритмы отслеживания движений и понимания пространства. Ключевым вопросом становится: как выбрать между этими платформами для оптимальной реализации задуманного AR-проекта?

В данной работе проведен анализ и сравнение трех популярных платформ для разработки AR: Vuforia, ARKit и ARCore, с особым акцентом на использование Vuforia Engine. Сравнение осуществляется по ряду качественных и количественных характеристик. Это исследование поможет разработчикам сделать информированный выбор о том, какая технология лучше всего подходит для их специфических нужд в разработке AR-приложений.

1. Технический обзор платформ

1.1. Vuforia Engine

Vuforia Engine — это программное обеспечение для разработки приложений дополненной реальности, которое позволяет создавать AR-приложения на различных платформах, включая iOS, Android, Windows и устройства Microsoft HoloLens. Особенностью Vuforia является поддержка широкого спектра типов маркеров, таких как 2D-изображения, 3D-объекты и специальные VuMarks. Платформа была первоначально разработана компанией Qualcomm, а затем приобретена компанией PTC. Vuforia поддерживает разработку на таких языках программирования, как C#, что делает её особенно привлекательной для использования

в среде Unity, где этот язык используется нативно [1].

Данная платформа имеет преимущества:

— Кроссплатформенность: работает на большинстве мобильных устройств и операционных систем, что обеспечивает широкий охват аудитории;
— разнообразие типов отслеживания: поддержка Image Targets, Object Targets, VuMarks и Ground Plane дает гибкость в создании интерактивных и захватывающих AR-сцен;
— расширенные AR-возможности: включает поддержку расширенного отслеживания, которое позволяет приложениям продолжать функционировать даже при временной потере маркеров из виду.

Недостатки Vuforia:

— стоимость лицензирования: для коммерческого использования Vuforia может потребовать приобретения лицензии, что увеличивает стоимость проекта;
— зависимость от качества изображений маркеров: точность и стабильность отслеживания могут сильно зависеть от качества и условий освещения изображений, используемых как маркеры;
— высокое энергопотребление, в особенности, при использовании сложных и ресурсоемких функций распознавания и отслеживания.

1.2. ARKit

ARKit — это фреймворк, разработанный Apple специально для устройств iOS. Эта технология позволяет разработчикам создавать высококачественные AR-приложения, используя мощности устройств на базе iOS. ARKit поддерживает Objective-C и Swift, что позволяет разработчикам интегрировать AR функциональность непосредственно в приложения для iOS, используя эти популярные языки программирования [2].

Преимущества ARKit:

— интеграция с iOS: полная оптимизация под устройства Apple обеспечивает высокую производительность и стабильность;
— продвинутое отслеживание: технологии, такие как отслеживание лица и мощные алгоритмы визуального отслеживания, позволяют создавать очень реалистичные взаимодействия;
— платформа ARKit использует аппаратные возможности устройства, такие как специализированные сенсоры и процессоры, для оптимизации процесса отслеживания и обеспечения более плавного и точного взаимодействия с виртуальными объектами, что способствует лучшему энергопотреблению приложения.

Недостатки:

— ограниченность платформы: работает только на устройствах Apple, что ограничивает аудиторию приложений;
— высокие требования к аппаратному обеспечению: для полноценной работы требуются последние модели процессоров устройств, которые интегрированы преимущественно в последние модели устройств.

1.3. ARCore

ARCore — это платформа от Google, созданная для внедрения дополненной реальности в приложения на Android и iOS. ARCore позволяет разработчикам использовать Java, Kotlin для Android и Swift, Objective-C для iOS для создания AR-приложений. Эта технология

предоставляет разработчикам доступ к функциям, таким как отслеживание движения, понимание окружающего пространства и оценка освещённости [3].

Преимущества:

— доступность: поддерживает широкий спектр устройств Android, что обеспечивает значительную базу пользователей;
— продвинутое взаимодействие с пространством: платформа ARCore способна оценивать размеры и особенности окружения, что позволяет AR-объектам реалистично взаимодействовать с реальным миром;
— адаптация к освещению: имеет возможности адаптации к текущему уровню освещенности, что делает AR-элементы более естественными в различных условиях освещения.

Недостатки:

— ограниченная поддержка устройств: хотя и охватывает многие устройства, ARCore лучше всего работает на новейших и более мощных моделях;
— вариативность производительности: из-за разнообразия устройств Android может наблюдаться неоднородность в производительности и качестве AR.

2. Сравнительный анализ технологий Vuforia, ARKit и ARCore

В табл. 1 предоставлен обзор основных качественных характеристик платформ дополненной реальности: Vuforia, ARKit и ARCore. Vuforia Engine является наиболее универсальной в поддержке платформ, поддерживая iOS, Android, UWP и даже Unity Editor для разработки и тестирования. ARKit ограничен iOS, в то время как ARCore поддерживает Android и iOS, но с некоторыми ограничениями.

Все три платформы поддерживают отслеживание плоскостей, лиц и изображений. Однако Vuforia Engine выделяется поддержкой различных типов маркеров, включая Image Targets, Object Targets, VuMarks и Ground Plane, что обеспечивает большую гибкость при создании интерактивных AR-сцен.

ARKit предоставляет дополнительные возможности адаптации, такие как AR Light Estimation, что позволяет приложениям реагировать на изменения в освещении. ARCore также обеспечивает адаптацию к условиям освещения, хотя она менее развита, чем у ARKit. Vuforia Engine, однако, ограничен в этом аспекте, не обеспечивая высокой степени адаптации к изменениям в освещении.

В случае Vuforia Engine требуются маркеры или специфические объекты для распознавания и отслеживания, в то время как ARKit и ARCore могут работать без использования маркеров для базового отслеживания.

Таблица 1

Сравнение качественных характеристик.

Характеристика	**Vuforia**	**ARKit**	**ARCore**
Платформы	iOS, Android, UWP, Unity Editor, MacOS	iOS	Android и iOS
Отслеживание	Image Targets, Object Targets, VuMarks, Ground Plane, Model Targets	Плоскости, лица, изображения, 3D объекты	Плоскости, лица, изображения, 3D объекты
Освещение	Не очень адаптивно к условиям освещения	Очень адаптивно, включает AR Light Estimation	AR Light Estimation для адаптации под

			условия освещения
Маркеры	Требуют маркеры или специфические объекты	Не требует маркеров для базового отслеживания	Не требует маркеров для базового отслеживания
Глобальное позиционирование	Ограничено	Поддерживает ARWorldMap для масштабирования и сохранения AR-сцен	Поддерживает Cloud Anchors для совместного использования AR объектов
Доступность	Лицензирование для коммерческого использования	Бесплатно для разработчиков iOS	Бесплатно для разработчиков на Android и iOS

ARKit поддерживает ARWorldMap, что позволяет сохранять и восстанавливать AR-сцены между сессиями. ARCore также поддерживает сохранение и обмен AR-объектами с помощью Cloud Anchors. Vuforia Engine, однако, имеет ограниченные возможности в этом аспекте, не предоставляя поддержки для сохранения и восстановления AR-сцен между сессиями. Vuforia Engine требует лицензирования для коммерческого использования, в то время как ARKit и ARCore бесплатны для разработчиков.

На рис. 1 приведен столбчатый график количественных характеристик производительности AR-приложений при нормальном освещении 500 люмен, таких как средний показатель кадров в секунду (FPS), точность отслеживания (accuracy of tracking) для трёх платформ.

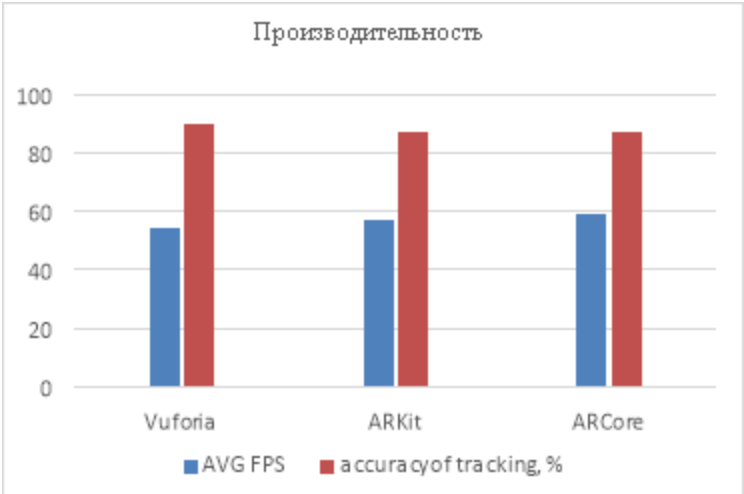


Рис. 5. Характеристики среднего значения FPS и точность отслеживания.

График показывает, что Vuforia обладает самой высокой точностью отслеживания за счёт использования маркеров, ARCore в свою очередь опережает двух оппонентов по среднему значению FPS.

3. Использование платформ в AR-приложениях

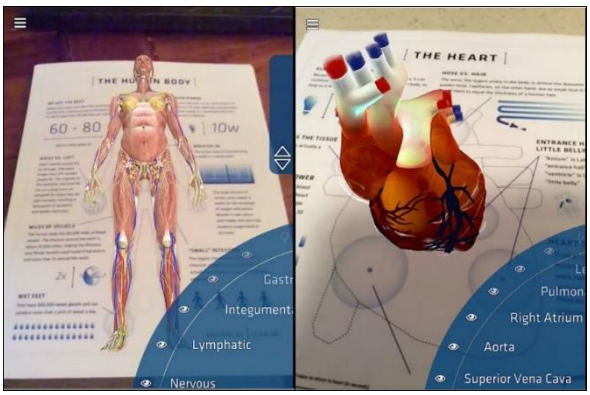


Рис. 6 Использование маркеров в AR-приложении [4]

Vuforia успешно используется для образовательных приложений, где требуется распознавание и интерактив с конкретными учебными пособиями. Например, Anatomy 4D (рис. 2) [4], — это приложение для изучения анатомии человеческого тела, которое использует распознавание маркеров на анатомических моделях и отображения дополнительной информации о различных органах и системах тела на экране мобильного устройства.

ARKit широко применяется в игровой индустрии и проектах, связанных с интерьерным дизайном, благодаря точной работе с плоскостями и оценке освещения. Например, IKEA Place, мобильном приложении, которое позволяет пользователям визуализировать мебель IKEA в их реальном окружении с помощью ARKit. Пользователи могут выбирать мебель из каталога и размещать ее в своей комнате, чтобы увидеть, как она выглядит в реальном масштабе; мобильная игра Pokemon GO.

ARCore используется в проектах по навигации и картографии, где важна точность позиционирования объектов в больших открытых пространствах, например, функция в Google Maps Google Maps AR Navigation использует ARCore для показа направлений на экране мобильного устройства, позволяя пользователям более точно ориентироваться в городе. Она отображает стрелки и указатели навигации на реальном виде с камеры устройства.

3. Принцип работы Vuforia

Vuforia использует технологию компьютерного зрения для распознавания и захвата плоских изображений или трехмерных объектов в реальном времени, позволяя разработчикам размещать виртуальные объекты через видоискатель камеры и корректировать их позицию на фоне. SDK Vuforia поддерживает различные типы 2D и 3D объектов, включая конфигурации множественных целей, изображения с меньшим количеством символов и маркеров. В SDK также добавлена функция, которая использует виртуальные кнопки для обнаружения локализованной заслоненности. Кроме того, оно может выбирать и переконфигурировать целевое изображение в реальном времени и создавать набор целей в соответствии с заданным сценарием. Диаграмма потока данных Vuforia показана на рис. 3 [5].

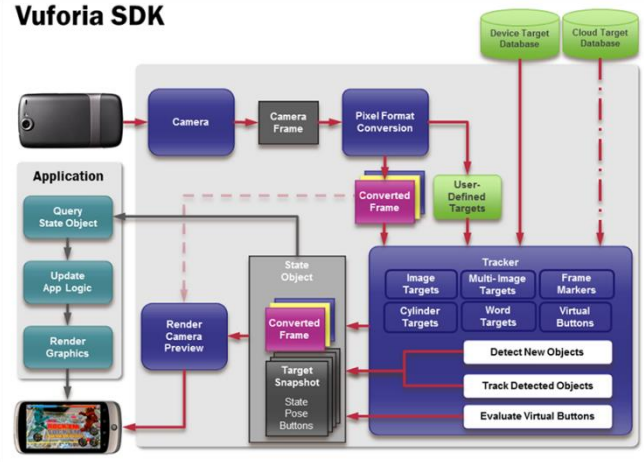


Рис. 3 Диаграмма потока данных [5]

Поток данных Vuforia разделен на четыре модуля: ввод, база данных, отслеживание и сопоставление и вывод рендера. Мобильные телефоны могут захватывать изображения каждого кадра в текущей реальной сцене через камеру, а затем своевременно сравнивать идентификационные объекты в базе данных в соответствии с преобразованием формата пикселей. После этого добавляются предварительно заданные виртуальные объекты, такие как 3D-модель, анимация или видео, в реальные сцены. Он также может взаимодействовать с этими

виртуальными объектами, рендерить и выводить информацию на мобильные устройства [5].

4. Выводы и результаты анализа

В результате использования и анализа платформ дополненной реальности можно сделать следующие выводы о преимуществах Vuforia перед ARKit и ARCore:

- более широкий спектр функций, таких как распознавание изображений и маркеров, что привлекательно для приложений, где важно точное определение позиции виртуальных объектов;
- лучший показатель точности отслеживания в нормальных условия за счёт распознавании маркеров;
- более простая настройка;
- возможность разрабатывать приложение под множество платформ;
- программная интеграция со средой разработки Unity за счёт нативного использования языка программирования C#.

Заключение

В заключение следует отметить, что применение платформ ARKit, ARCore и Vuforia в разработке приложений дополненной реальности предоставляет разработчикам мощные инструменты для создания увлекательных и инновационных проектов. Несмотря на схожие функциональные возможности ARKit и ARCore в отслеживании поверхностей и размещении виртуальных объектов, Vuforia выделяется более широким спектром функций, таких как распознавание изображений и маркеров, а также виртуальные кнопки для дополнительного взаимодействия. При этом все три платформы обладают хорошей производительностью, что позволяет создавать приложения с высоким качеством визуальной обработки и отзывчивым пользовательским интерфейсом. Однако, хочется отметить, что выбор конкретной платформы зависит от требований проекта и предпочтений разработчика. В конечном итоге, применение платформ ARKit, ARCore и Vuforia открывает новые возможности для разработки инновационных приложений дополненной реальности, которые могут изменить способ взаимодействия с реальным миром и обогатить пользовательский опыт.

Литература

1. Official Vuforia Engine Documentation. Режим доступа: https://developer.vuforia.com/

2. Apple Developer Documentation on ARKit. Режим доступа: https://developer.apple.com/augmented-reality/arkit/

3. ARCore Google Developers. Режим доступа: https://developers.google.com/ar

4. Сборник фотографий Flectone : база данных. – Режим доступа: https://flectone.ru/anatomiya-cheloveka-4-toma.html

5. Luo Dongyong, Zhang Shujun. An Automatic Navigation Method of Mobile AR Based on Unity 3D [J]. Computer and Digital Engineering 2015(11):2024-2028.

УДК 004.023

АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ В РАЗРАБОТКЕ НАВИГАЦИОННЫХ ПРИЛОЖЕНИЙ

О. С. Сидоров

Воронежский государственный университет

Введение

Навигатор - программное обеспечение, предназначенное для навигации и определения маршрута в пространстве. Существующие приложения-навигаторы предлагают широкий спектр функций, включая геолокацию, информацию о пробках и авариях, а также о точках интереса. Таким образом, разработка навигационного приложения включает в себя следующие этапы: составление карты, определение объектов и зданий на карте, определение кратчайшего маршрута для навигации, получение информации о загруженности дорог.

Цель работы: проанализировать существующие подходы в создании навигационных приложений.

Были поставлены задачи:

1. Изучить каким образом популярные навигационные приложения реализуют необходимую функциональность.
2. Проанализировать сложность найденных решений в условиях ограниченных ресурсов.

1. Создание карт

Для создания изображений на поверхности земли навигационные приложения применяют следующие методы.

1. Использование спутниковых сетейдля создания снимков местности с последующей обработкой и составлением карт.
2. Использование ГИС (географическая информационная система) - компьютерных систем, предназначенных для сбора, хранения, анализа, отображения и управления пространственными данными.

2. Определение объектов

Навигаторы используют различные методы и технологии для разметки объектов на картах.

1. Геокодирование - это процесс преобразования адресов или местоположений в географические координаты (широту и долготу). Навигаторы могут использовать геокодирование для определения точного местоположения объектов на карте. Данные могут быть получены от различных поставщиков, таких как Google Maps, OpenStreetMap и другие.
2. Информация от пользователей приложения. Некоторые навигационные приложения позволяют своим пользователям размещать объекты на картах и после проверки, их добавляют в базу данных карты приложения.
3. Навигаторы могут использовать фотографии и изображения объектов, сделанные пользователем или полученные из других источников, для разметки на карте, уточняя их расположение с помощью GPS данных.

Обычно навигаторы комбинируют несколько из перечисленных выше методов для создания точных и информативных карт.

3. Определение информации о загруженности дорог

Навигаторы получают информацию о загруженности дорог с помощью различных методов и источников данных:

1. Мобильные приложения и устройства могут собирать данные GPS о движении пользователей для отслеживания движения автомобилей и оценки скорости движения на различных участках дорог. Из-за неточности технологии GPS необходимо обрабатывать данные по сложным алгоритмам, чтобы отличить ошибку определение устройства от действительной ситуации на дороге.
2. Информацию можно получать от специализированных компаний, которые предоставляют информацию о трафике на основе данных с систем мониторинга трафика, например, с камер наблюдения, датчиков на дорогах и других источников.
3. Использование информации распространяемых существующими приложениями.

4. Нахождение кратчайших маршрутов

Навигаторы используют различные алгоритмы и технологии для поиска кратчайших маршрутов при прокладке путей. Примеры наиболее распространенных алгоритмов.

1. Алгоритм Дейкстры. Используется для поиска кратчайшего пути от одной вершины графа к остальным с учетом веса ребер.
2. Алгоритм A* (A-star): Этот алгоритм использует эвристику о расстоянии до цели, чтобы ускорить процесс поиска оптимального маршрута.
3. Генетические алгоритмы: Некоторые навигаторы используют методы, основанные на принципах эволюции и генетических алгоритмах для оптимизации маршрутов.

Все эти алгоритмы могут применяться для поиска не только кратчайшего, но и наиболее оптимального маршрута с учетом различных факторов, таких как время, расход топлива и т. д.

Анализ решений

Создание карт. Использование спутниковой съёмки является наиболее точным методом решения задачи, также позволяющим часто обновлять карту, но этот способ слишком дорогой. В то время как использование базы данных объектов усложнит разработку и создание методов конвертации данных в изображение, а также данный метод подвержен устареванию данных, но он более доступен.

Определение объектов на картах. Использование баз данных адресов и местоположений позволяет автоматизировать процесс, но создаёт зависимость от поставщиков данных. Возможность добавления объектов пользователями требует разработать удобный инструмент для редактирования, а также создать возможность проверять данные введённые пользователями. Задание объектов используя фотографии является не только не стабильным способом добавления данных, но также требует огромных временных затрат для поддержания актуальности.

Определение информации о загруженности дорог. Использование данных GPS требует большого количества пользователей, установивших приложение для работы, что не даёт гарантий правильной работы из-за неточности технологии GPS. Использование данных полученных от специализированных компаний является самым точным методом, но этот способ доступен лишь в крупных городах, и лишь на дорогах где установлено необходимое

оборудование. Использование информации, полученной от существующих приложений, является наиболее простым источником данных, но создаёт зависимость от этих приложений.

Нахождение кратчайших маршрутов. Алгоритм Дейкстры является наиболее популярным алгоритмом для нахождения кратчайших путей, но при увеличении размера графа, в связи с расширением покрытия карты, скорости работы этого алгоритма становится недостаточно. Алгоритм A* решает проблему алгоритма Дейкстры, но теряет в точности и быстродействии в случаях, когда маршрут требует двигаться от цели при построении. Использование нейронных сетей является очень сложным в реализации решением, требующих обширных знаний в создании и обучении сетей, также такие сети крайне требовательны к структурам хранения данных и входным значениям, что повышает сложность приложения.

Заключение

В результате исследования были проанализированы существующие решения задач, стоящих при разработке навигационных приложений. Поскольку наиболее точные методы не доступны в условиях ограниченных ресурсов, то при решении многих задачи приходится опираться на данные, распространяемые уже существующими приложениями и поставщиками данных.

Литература

1. Как устроены Яндекс.Карты. Лекция Владимира Зайцева в Яндексе – Режим доступа: https://habr.com/ru/companies/yandex/articles/219951/ (Дата обращения: 30.03.2024).

2. Как статистика помогает делать Яндекс.Пробки лучше – Режим доступа: https://habr.com/ru/companies/yandex/articles/210240/ (Дата обращения: 30.03.2024).

3. Как Яндекс создаёт карты – Режим доступа: https://yandex.ru/company/technologies/maps (Дата обращения: 30.03.2024).

4. Гугл Карты: как это работает? – Режим доступа: https://dzen.ru/a/YZf97bwuUyoKf5uY (Дата обращения: 30.03.2024).

5. Как работает карта Google? – https://www.geeksforgeeks.org/how-does-google-map-works/ (Дата обращения: 30.03.2024).

6. Откуда берутся данные в картах Google? – Режим доступа: https://dzen.ru/a/XeYo0_vm5wCwSmGy (Дата обращения: 30.03.2024).

7. GNSS smartphones positioning: advances, challenges, opportunities, and future perspectives – Режим доступа: https://www.researchgate.net/publication/356265982_GNSS_smartphones_positioning_advances_challenges_opportunities_and_future_perspectives (Дата обращения: 14.04.2024).

8. A Seamless Navigation System and Applications for Autonomous Vehicles Using a Tightly Coupled GNSS/UWB/INS/Map Integration Scheme – Режим доступа: https://www.researchgate.net/publication/357276196_A_Seamless_Navigation_System_and_Applications_for_Autonomous_Vehicles_Using_a_Tightly_Coupled_GNSSUWBINSMap_Integration_Scheme (Дата обращения: 14.04.2024).

УДК 004.42

РЕШЕНИЕ ЗАДАЧИ О ДВУМЕРНОМ РАСКРОЕ С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

А. Е. Симакова

Воронежский государственный университет

Введение

Настоящая статья посвящена применению генетических алгоритмов к решению задачи о двумерном прямоугольном раскрое. В задачах такого типа требуется разместить несколько заготовок на шаблоне с учетом ненаслоения заготовок друг на друга и параллельности сторон заготовок сторонам шаблона. Задача о раскрое относится к проблеме оптимизационного геометрического моделирования.

1. Постановка задачи

Рассмотрим постановку задачи о двумерном раскрое.

Дан шаблон размера M на N и набор прямоугольных заготовок $a_1, a_2, \ldots, a_n$ с размерами $(X_i,Y_i), i=\overline{1,n}$, где X_i отвечает за длину, а Y_i — за ширину заготовки a_i.

Вектор $b=(b_i),\ i=\overline{1,n},\ b_i \in \{1,2,\ldots,n\}$ представляет собой вектор приоритетов расположения заготовок в шаблоне. Предполагается, что значение $b_i=1$, если заготовка a_i является самой приоритетной для выбора и так далее по убыванию предпочтительности.

Также введем переменные $Z_i, i=\overline{1,n}$, где Z_i — количество заготовок i-го типа, попавшие в шаблон.

Требуется расположить заготовки на шаблоне с учетом ненаслоения заготовок друг на друга и параллельности сторон заготовок сторонам шаблона. При этом с целью максимального удовлетворения предпочтений по выбранным в шаблон заготовкам вводится следующая целевая функция:

$$F(Z)=\sum_{i=1}^{n} Z_i b_i \to min$$

В силу NP-трудности задачи двумерной прямоугольной упаковки для ее решения применяются эвристические алгоритмы полиномиальной сложности, что позволяет получить рациональное решение за приемлемое время. Для решения данной задачи будет использован генетический алгоритм.

2. Описание и применение генетического алгоритма

Генетический алгоритм — это алгоритм, который используется для решения задач оптимизации и моделирования путём случайного подбора популяции, индивидуумов в популяции, хромосом, где определение оптимального решения основано на комбинировании (скрещивании генов) и вариации параметров (мутации) с использованием механизмов, аналогичных естественному отбору в природе (рис. 1).

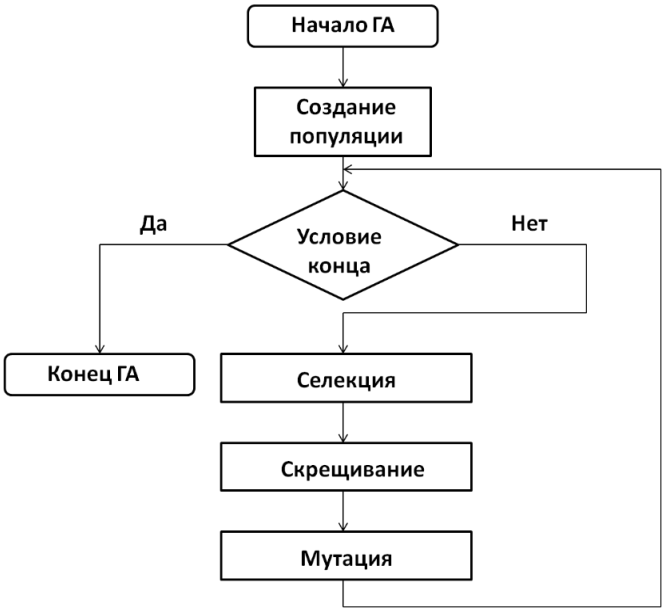


Рис.1. Схема генетического алгоритма

Рассмотрим подробнее этапы генетического алгоритма применительно к поставленной задаче.

Шаг 1. Инициализация популяции

Одной из основных задач при реализации генетического алгоритма является задача кодирования хромосомы. Для рассматриваемой задачи под хромосомой будет подразумеваться размещение заготовок на шаблоне, удовлетворяющее условию задачи.

Хромосома представляет собой матрицу вида

$$A = \begin{pmatrix} a_1 & a_2 & \dots & a_k \\ x_1 & x_2 & \dots & x_k \\ y_1 & y_2 & \dots & y_k \end{pmatrix},$$

где k – количество заготовок, попавших в шаблон. Каждый столбец соответствует заготовке, которую нужно разместить на шаблоне, и содержит два значения: тип заготовки и координаты её левого нижнего угла. Заметим, что заготовка за i-го типа может встречаться в хромосоме неограниченное число раз. На рис. 2 представлен пример особи.

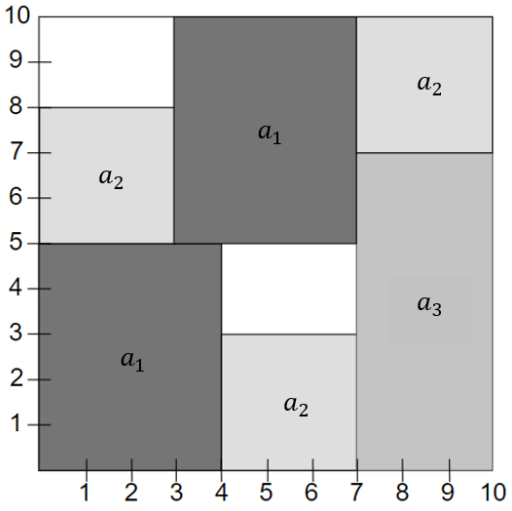


Рис.2. Пример особи

Хромосомой для данной особи будет являться следующая матрица:

$$A=\begin{pmatrix} a_1 & a_2 & a_3 & a_2 & a_1 & a_2 \\ 0 & 0 & 0 & 5 & 3 & 7 \\ 0 & 4 & 7 & 0 & 5 & 7 \end{pmatrix}.$$

Для кодирования заготовок на шаблоне применяется схема «Привязки к углу». В этой схеме рассматриваются все доступные углы на поле шаблона. Каждый ген хромосомы состоит из трех элементов: номера заготовки и координат левого нижнего угла, к которому она будет привязана. Заполнение шаблона происходит с левого нижнего угла поля шаблона. Если разместить заготовку не удается, попытка повторяется с соседней точки справа или сверху (если справа недостаточно места для размещения заготовки данного размера). Если текущая заготовка не помещается в шаблон, то выбирается следующая по приоритетности заготовка и проверяется возможность ее размещения. Этот процесс продолжается до тех пор, пока не будут просмотрены все заготовки.

На примере ниже (рис. 3) продемонстрировано возможное размещение заготовок. Заготовка a_1 привязана к углу с координатами (0,0) и имеет размеры $X = 4, Y = 7$. Это значит, что левый нижний угол следующий заготовки этого же типа будет иметь от координаты (4,0), а угол заготовка a_2 координаты (8,0), так как заготовка типа a_1 не поместится в свободное место.

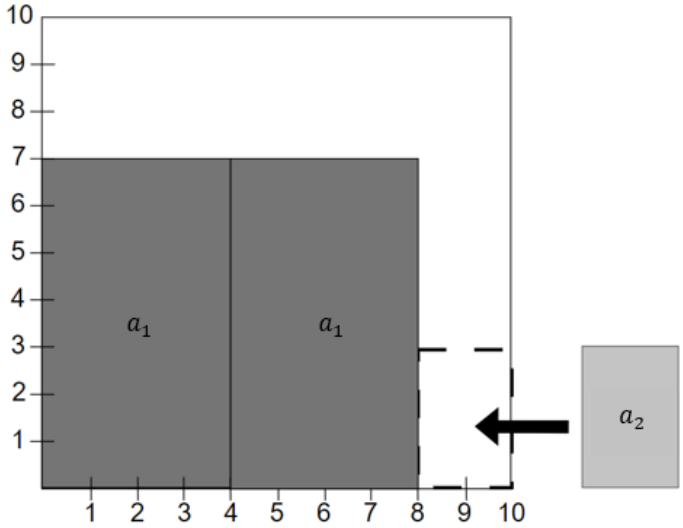


Рис.3. Возможное размещение заготовки a_2.

Шаг 2. Оценка популяции

На этом этапе каждое решение в популяции оценивается с помощью функции приспособленности (фитнес-функции). Функция приспособленности определяет, насколько хорошо решение соответствует целям задачи.

Фитнес функция в рассматриваемой задаче представляет собой сумму приоритетов помещаемых на шаблон заготовок:

$$F(Z) = \sum_{i=1}^{n} Z_i b_i \to min.$$

Так как наиболее приоритетная заготовка имеет минимальное значение приоритета, то представленная функция будет минимизироваться.

Шаг 3. Скрещивание

Выбираются два родителя с заданными хромосомами. Затем случайным образом выбираются точки скрещивания, которые определяют ту часть информации из хромосомы, которая будет передана потомку. Создание потомства происходит путем объединения частей хромосом родителей (рис. 4).

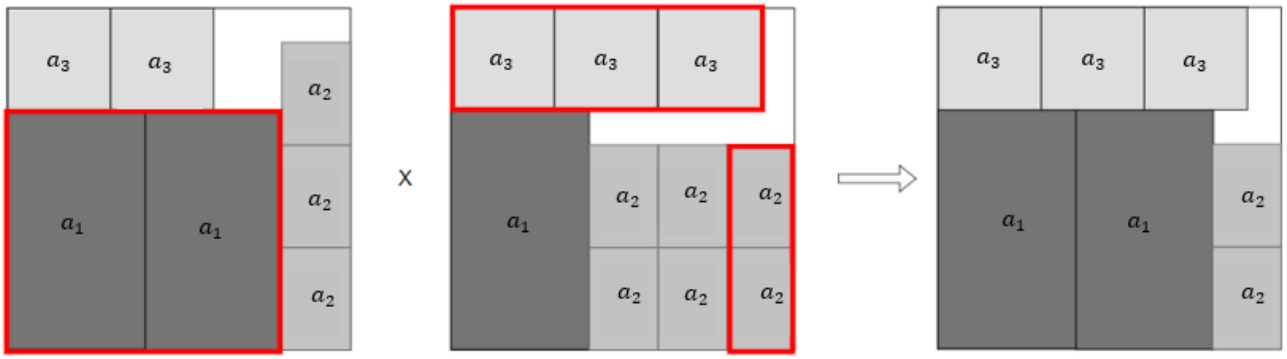


Рис.4. Получение потомка

В данном примере матрицы хромосом родителей выглядят следующим образом:

$$A_{род_1} = \begin{pmatrix} a_1 & a_1 & a_2 & a_2 & a_2 & a_3 & a_3 \\ 0 & 4 & 8 & 8 & 8 & 0 & 3 \\ 0 & 0 & 0 & 3 & 6 & 7 & 7 \end{pmatrix},$$

$$A_{род_2} = \begin{pmatrix} a_1 & a_2 & a_2 & a_2 & a_2 & a_2 & a_2 & a_3 & a_3 & a_3 \\ 0 & 4 & 6 & 8 & 4 & 6 & 8 & 0 & 3 & 6 \\ 0 & 0 & 0 & 0 & 3 & 3 & 3 & 7 & 7 & 7 \end{pmatrix}.$$

Выбраны следующие точки скрещивания:

$$A_{род_1} = \begin{pmatrix} a_1 & a_1 & - & - & - & - & - \\ 0 & 4 & - & - & - & - & - \\ 0 & 0 & - & - & - & - & - \end{pmatrix},$$

$$A_{род_2} = \begin{pmatrix} - & - & - & a_2 & - & - & a_2 & a_3 & a_3 & a_3 \\ - & - & - & 8 & - & - & 8 & 0 & 3 & 6 \\ - & - & - & 0 & - & - & 3 & 7 & 7 & 7 \end{pmatrix}.$$

Хромосома потомка в матричном виде представлена ниже:

$$A_{потомк} = \begin{pmatrix} a_1 & a_1 & a_2 & a_2 & a_3 & a_3 & a_3 \\ 0 & 4 & 8 & 8 & 0 & 3 & 6 \\ 0 & 0 & 0 & 3 & 7 & 7 & 7 \end{pmatrix}.$$

Если во время скрещивания находятся недопустимые значения (например, наслаивание частей хромосом), то одна из частей, выбранная случайным образом, исключается.

Затем родители меняются местами и создается еще один потомок. В результате число особей популяции возрастает в 2 раза.

После скрещивания потомство может содержать дубликаты или недопустимые значения (например, заготовки, размещенные вне шаблона или наслоенные друг на друга). На рис. 5 представлено наслоение заготовок родителей друг на друга.

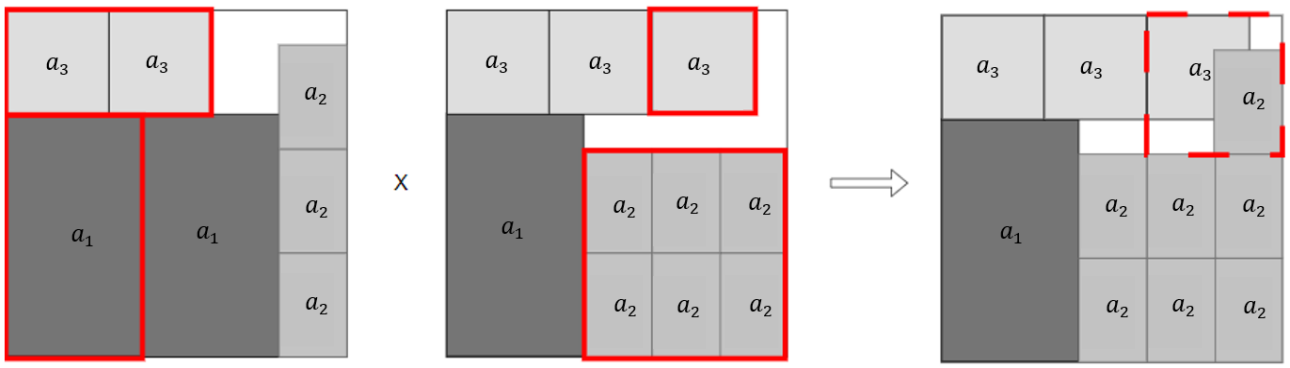


Рис.5. Недопустимые значения у потомка

Такие недопустимые значения заменяются на допустимые значения из родительских хромосом, то есть на заготовку меньшего размера. Это означает, что в представленном примере у потомка останется заготовка a_2, а заготовка a_3 будет удалена.

Шаг 4. Мутация

Мутация используется для внесения случайных изменений в хромосомы, что помогает алгоритму исследовать новые области пространства решений и избегать застревания в локальных минимумах.

Случайные изменения в генотипе затрагивают только потомков, созданных на текущей итерации: случайно выбранная заготовка будет заменена на заготовку другого размера, если это удовлетворяет условию ненаслоенности (рис. 6).

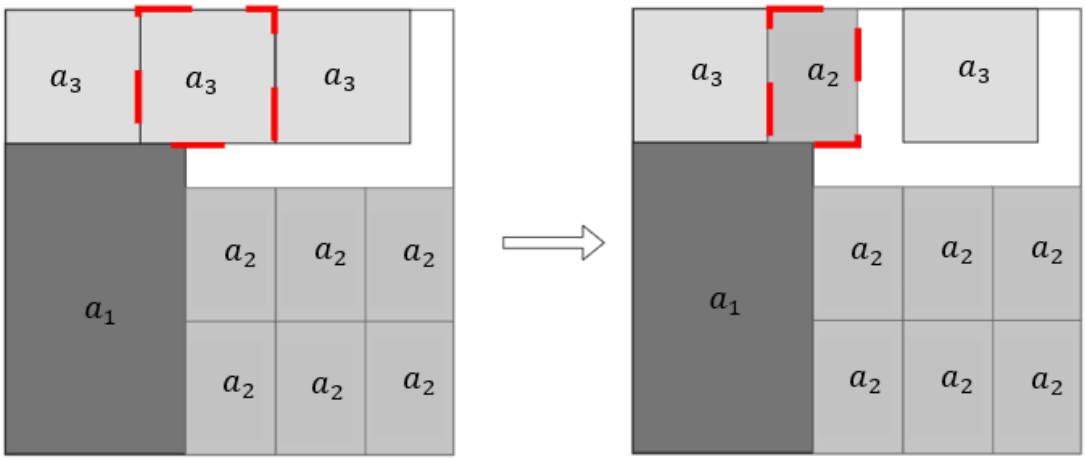


Рис.6. Пример мутации

В данном примере заготовка a_3 была заменена на заготовку меньшего размера, а именно на заготовку a_2.

Шаг 5. Селекция

После создания новых решений происходит процесс селекции, в ходе которого выбираются лучшие решения для создания следующего поколения.

Требуется отсечь потомков, которые показали худший результат по суммарному приоритету, описываемому фитнес-функцией.

На основе выбранных решений создается новое поколение, которое будет использоваться в следующем цикле алгоритма.

3. Результаты работы программы

Рассмотрим пример работы предложенного алгоритма формирования шаблона заготовок. Пусть заданы заготовки: a_1 размером 8×6 у.е., a_2 размером 3×3 у.е., a_3 размером 2×7 у.е. Вектор предпочтений имеет вид: $b=(1,2,3)$, то есть самым предпочтительным было использование заготовки a_1, наименее предпочтительным — a_3.

В результате применения предложенного алгоритма получено следующее расположение данных заготовок на шаблоне (рис. 7).

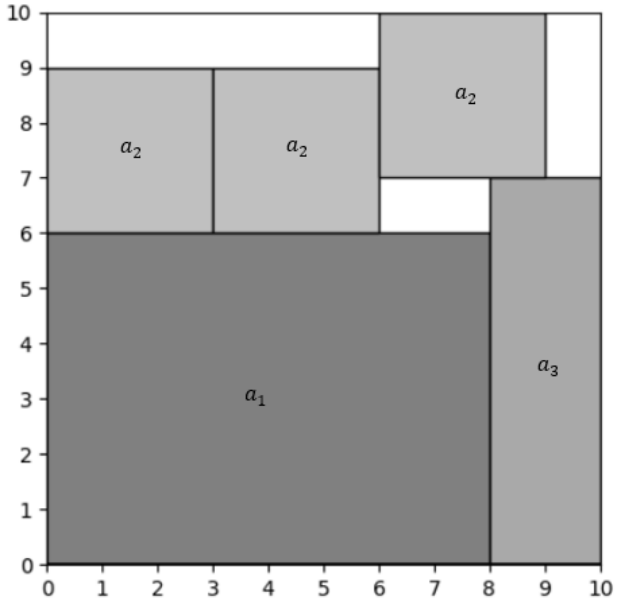


Рис.7. Новое расположение заготовок

Суммарное значение приоритетов заготовок для данной задачи:

$$F(Z)=\sum_{i=1}^{n} Z_i b_i = 1\cdot 1+3\cdot 2+1\cdot 3=10.$$

Заключение

В процессе исследования был рассмотрен и спроектирован жизненный цикл популяции, а также реализован генетический алгоритм для решения задачи двумерного прямоугольного раскроя.

Литература

1. Martello S., Toth P. Knapsack Problems. – Wiley, Chichester, 1990.
2. Голубцов Е. А. Методы оптимизации в задачах комбинаторного раскроя / Е А. Голубцов. – Москва : Физматлит, 2015. – 273 с.
3. Джон Х. Генетические алгоритмы в поисках оптимальности / Х. Джон. – Москва : Вильямс, 2001. – 149 с.
4. Кафаров В. В. Генетические алгоритмы: теория, применения, учебное пособие / В. В. Кафаров. – Москва : Лаборатория Базовых Знаний, 2019. – 220 с.
5. Каммерваер Д. Генетические алгоритмы: искусство интеллектуальной оптимизации / Д. Каммерваер. – Санкт-Петербург : Питер, 2003. – 146 с.

УДК 004.05

АНАЛИЗ ЗАВИСИМОСТЕЙ ЯЗЫКОВ ДЛЯ JVM

Д. Д. Скибин

Воронежский государственный университет

Аннотация. Работа посвящена описанию программных средств для создания наглядных графических представлений структуры внутренних зависимостей проектов для виртуальной машины java. Тема работы имеет важное значение для проектирования объектно-ориентированных систем с понятной архитектурой, также подобный инструментарий полезен для обратного проектирования и восстановления архитектуры. В статье описаны функциональные возможности инструмента, которые обеспечивают инфраструктуру для последующего выявления и устранения ошибок в проектировании программных систем.
Ключевые слова: автоматизация, граф, зависимости, ошибки, пакет, поиск, проектирование, разработка, визуализация, java.

Введение

Современные проекты могут содержать в себе сотни, а порой и тысячи классов, разделённых на множество пакетов и каталогов. С каждым новым файлом, в котором может содержаться несколько классов, разобраться в структуре проекта становится всё сложнее. В особенности это актуально для разработчиков, присоединившихся на поздних стадиях создания продукта, особенно это актуально для систем с открытым исходным кодом, которых с каждым днём становится всё больше

В процессе разработки программного обеспечения анализ, генерация и преобразование программ являются неотъемлемыми и полезными этапами. Анализ классов позволяет выявить потенциальные ошибки и неиспользуемый код. Кроме того, анализ классов служит основой для обратного проектирования кода, так как некоторые процессы повторной сборки могут зависеть только от соответствующих классов.

Генерация программ применяется в различных типах компиляторов, таких как традиционные компиляторы, компиляторы-заглушки и скелетные компиляторы, используемые в распределенном программировании [4].

Изучение проекта. может занять много времени, так же за этот период возрастает риск возникновения ошибок, что влечёт новые временные издержки. Чтобы подобные ошибки построения архитектуры приложения своевременно отследить, также необходимо понимать сложившуюся структуру проекта.

Структурирование проекта и отслеживание внутренних зависимостей являются ключевыми факторами, обеспечивающими успешное выполнение задач и преодоление возникающих препятствий. Это позволяет команде легко следить за прогрессом работы, а также быстро реагировать на возникающие проблемы и внести корректировки в планы и графики.

Актуальность данной работы заключается в том, что реализация идеи о визуализации структуры внутренних зависимостей проекта для виртуальной машины java, отвечает потребностям индустрии.

1. Теоретическая база для создания программных средств

Зависимость (связь) класса от другого класса в программировании означает, что один класс использует или зависит от другого класса. Это может проявляться в различных формах:

— методы и параметры: класс может вызывать методы другого класса или передавать ему параметры;

— объекты: класс может содержать в себе объекты другого класса;

— наследование: класс может наследовать другой класс для использования его функциональности.

Зависимость между классами может быть прямой или косвенной. Прямая зависимость означает, что класс A явно использует класс B. Косвенная зависимость возникает, когда класс A зависит от класса B через другие классы или интерфейсы.

Программы для виртуальной машины java в скомпилированном виде храниться в специальных java архивах, таких как:

— JAR-файл – это Java-архив. Представляет собой ZIP-архив, в котором содержится часть программы на языках JVM;

— Web Archive или Web Application Resource – формат файла, описывающий, как полное веб-приложение упаковывается в соответствии со спецификацией Java-сервлетов в файл в формате JAR или ZIP.

Исходный код на языке программирования сначала компилируется в байт-код с помощью компилятора, который входит в состав Java Development Kit (JDK). Байт-код сохраняется в бинарный файл, известный как class-файл – файл с расширением «.class» [1]. Затем они динамически загружаются в память с помощью загрузчика классов (ClassLoader), когда это необходимо.

Class-файл представляет собой бинарный файл, в котором информация обычно записывается без отступов между последовательными частями. Все значения размером 16 бит и 32 бита записываются с использованием двух или четырех последовательных 8-битных байтов

Class-файл содержит такую информацию как:

— интерфейсы, реализованные классом;

— поля класса, после количества полей следует таблица структур переменной длины, по одной для каждого поля с описанием типа поля и названия;

— методы. Для каждого метода содержится следующая информация: дескриптор метода (тип возвращаемого значения и список аргументов), количество слов, необходимых для локальных переменных метода, максимальное количество слов стека, необходимых для стека операндов метода, таблицу исключений, перехватываемых методом, байт-коды метода и таблица номеров строк.

В конечном итоге программа для JVM будет представлять собой архив, содержащий в каталогизированном виде все содержащиеся в проекте классы в виде class-файлов, которые в свою очередь содержать последовательности инструкций. Определив инструкцию, её можно чётко интерпретировать.

2. Постановка задачи

Необходимо разработать приложение для визуализации структуры связи (зависимостей) классов, позволяющие приложению:

— получить список class-файлов проекта;
— проанализировать содержимое class-файла, вычленив зависимости класса;
— построить на основе полученной информации визуализированный граф, с учётом таких деталей как частота упоминаний и тип связи.

Инструментальные программные средства должны обеспечивать поддержку следующих основных функций:

1. Чтение файлов из архива.
2. Чтение бинарных файлов.
3. Определение значений инструкций.
4. Получение имени связанного класса из инструкции.

3. Структура программы

В структуре программных средств можно выделить несколько блоков, каждый из которых будет обеспечивать выполнение определенных групп действий (рис. 1).

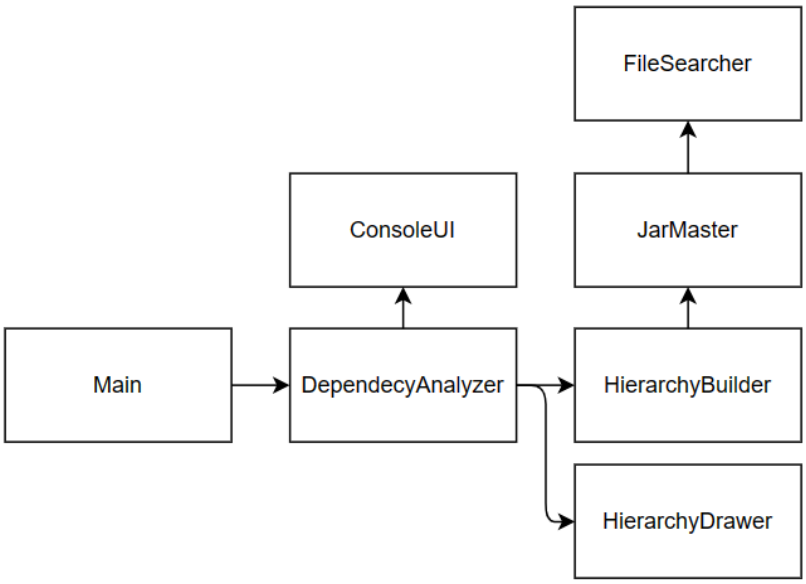


Рис. 1. Упрощённая структура программы

Main – класс, отвечающий за старт программы с учётом переданных аргументов.

ConsoleUI – класс обеспечивающий вывод информации в консоль и ввод дополнительных данных не переданных через аргументы при вызове консольного приложения, но необходимых для корректной работы программы.

DependenyAnalyzer – ключевой класс отвечающий за основную логику приложения.

HierarchyBuilder – класс анализирующий классы в виде массивов байтов, на основе чего строится структура, отражающая иерархию классов, так же фиксируются дополнительные сведения, которые могут быть полезны, для дальнейшего анализа.

JarMaster – класс отвечающий за чтение java архива и получение class-файлов в виде массива байтов.

FileSearcher – класс отвечающий за поиск jar-архива в директории проекта.

HierarchyDrawer – класс строящий графическое отображение иерархии классов.

2. Примеры работы программы

Рассмотрим следующую ситуацию, с которой можно применить приложение: существует проект на языке java, содержащий следующие классы: Main, AClass, BClass, FieldClass, CHelper_1, CHelper_2, CircleClass_1, CircleClass_2, CircleClass_3.

На рис. 2 представлен ввод данных при запуске приложения без аргумента указывающего путь к директории проекта.

```
Write absolute path to git repository:
>C:\Local_WorkSpace\Sandbox\Sandbox
Select jar files:
0) C:\Local_WorkSpace\Sandbox\Sandbox\target\Sandbox-1.0-SNAPSHOT-jar-with-dependencies.jar
1) C:\Local_WorkSpace\Sandbox\Sandbox\target\Sandbox-1.0-SNAPSHOT.jar
>1
```

Рис. 2. Ввод данных

В консоль вводится путь к директории, затем пользователю предлагается выбрать jar файл, так как в рамках одного проекта, может быть, несколько таких файлов.

После определения анализируемого архива ввод данных завершён.

На рис. 3 представлен текстовое представление зависимостей (связей) классов проекта.

```
Class: Main
Method(<init>);
   OPCODE: INVOKE method
    java/lang/Object()V
Method(main);
   > java.lang.String[]
   OPCODE: NEW
    AlphabetClass/AClass
    field/FieldClass
    AlphabetClass/BClass
    AlphabetClass/CClass
    field/CHelper_1
    field/CHelper_2
    circle/CircleClass_1
   OPCODE: INVOKE method
    field/FieldClass()V
    AlphabetClass/AClass(Lfield/FieldClass;)V
    AlphabetClass/BClass()V
    field/CHelper_1()V
    field/CHelper_2()V
    AlphabetClass/CClass(Lfield/CHelper_1;Lfield/CHelper_2;)V
    circle/CircleClass_1()V
    AlphabetClass/AClass()V
    AlphabetClass/BClass()V
    AlphabetClass/BClass(LAlphabetClass/AClass;)V
    AlphabetClass/CClass()V
Class: circle/CircleClass_1
Field: circleClass2; desc: Lcircle/CircleClass_2;
Method(<init>);
   OPCODE: NEW
```

Рис. 3. Частичное текстовое представление проекта

Так же создастся графическое отображение представленное на рис. 4.

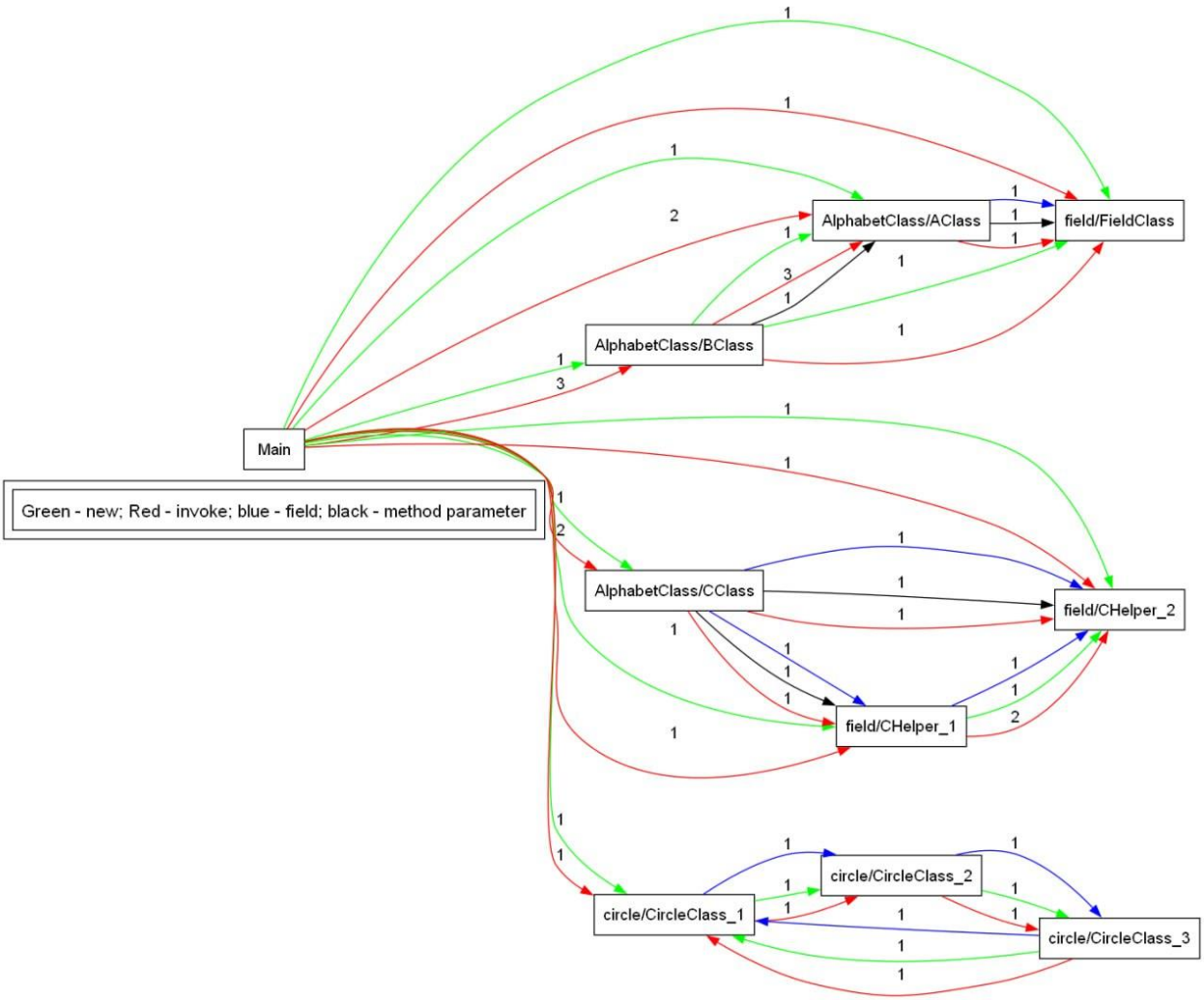


Рис. 4. Графическое отображение проекта

Заключение

Программный продукт выполняет поставленную задачу и может быть использован в процессе разработки программного обеспечения с использованием языков программирования для виртуальной машины Java. Разработаны программные средства, которые в перспективе могут применяться для выявления ошибок архитектуры проекта и ознакомления со структурой проекта. Это может упростить и ускорить разработку программный продуктов.

Существует перспектива развития приложения, так как в процессе разработки появились новые задачи. В будущем можно реализовать следующие возможности:

1. Строительство графа на основе различий веток git репозитория.
2. Группировка классов на графе по пакетам, в которых они находятся.
3. Улучшить читаемость графа.
4. Автоматическое выявление зависимостей.
5. Реализовать также отображение в виде матрицы зависимостей [2, 3].
6. Реализовать проект в виде плагина для Maven.

Литература

1. The Java® Virtual Machine Specification Java SE 14 Editi on ; Lindholm, T., Yellin, F., Bracha, G., Buckley – 14-е изд., обнов. ; Oracle America, Inc. 1997, 2020. – 620 с.
2. Identifying cycle causes with Enriched Dependency Structural Matrix / J. Laval, S. Denier, S. Ducasse and A. Bergel // 16th Working Conference on Reverse Engineering – 2009.
3. Using dependency models to manage complex software architecture : N. Sangal, E. Jordan, V. Sinha, and D. Jackson // In OOPSLA – 2005. – С. 167–176.
4. ASM: a code manipulation tool to implement adaptable systems / Bruneton E., Lenglet R., Coupaye T. // Adaptable and extensible component systems – 2002. – С. 4–5

УДК 004.4

ИССЛЕДОВАНИЕ СРЕДСТВ РЕАЛИЗАЦИИ ПЛАТФОРМЫ ДЛЯ ПУБЛИКАЦИИ СТАТЕЙ НА ОСНОВЕ ТЕХНОЛОГИИ NODE.JS

С. С. Сливкин

Воронежский государственный университет

Введение

В данной статье рассматривается процесс создания приложения для публикации статей с использованием Node.js — мощной и гибкой среды выполнения JavaScript.

Серверная часть приложения — это не только платформа для хранения и управления контентом, но и надежная основа для реализации функций аутентификации пользователей, обработки данных, обеспечения безопасности и создания удобного API для взаимодействия с клиентской стороной.

1. Описание приложения

Платформа обладает следующими функциями:

1. Регистрация и аутентификация: Пользователи смогут создавать учетные записи на платформе, аутентифицироваться и управлять своими профилями.
2. Публикация статей: Авторы смогут загружать свои научные статьи на платформу, прикреплять к ним метаданные (название, аннотация, ключевые слова) и выбирать соответствующие категории.
3. Оценка и комментирование статей: Зарегистрированные пользователи смогут оценивать статьи, оставлять комментарии и обсуждать научные работы с другими участниками сообщества.
4. Управление профилем: Авторы смогут управлять своими опубликованными статьями, отслеживать статистику просмотров и оценок, а также обновлять метаданные своих работ.
5. Уведомления: Пользователи смогут получать уведомления о новых комментариях, оценках и других событиях, связанных с их статьями или деятельностью на платформе.

2. Используемые технологии

Разберем подробнее каждую из использованных технологий:

1. Node.js: Среда выполнения JavaScript, позволяющая запускать код на стороне сервера. Node.js обеспечивает быстрое и масштабируемое выполнение кода благодаря асинхронной обработке ввода-вывода.
2. Express.js: Минималистичный и гибкий веб-фреймворк для Node.js, который упрощает создание маршрутов, обработку запросов и управление состоянием приложения.
3. Passport.js: Это популярная библиотека для аутентификации в Node.js, которая обеспечивает различные стратегии аутентификации, такие как аутентификация по логину и паролю, OAuth и другие. Passport.js также обеспечивает управление

сеансами пользователей и удобный интерфейс для создания кастомной логики аутентификации.

4. Socket.IO: Это библиотека для работы с веб-сокетами в Node.js, позволяющая устанавливать двустороннюю связь между клиентом и сервером в режиме реального времени. Socket.IO полезен для реализации функционала уведомлений и обновлений в реальном времени, таких как уведомления о новых комментариях или оценках статей.
5. JWT (JSON Web Tokens): Это открытый стандарт (RFC 7519) для создания токенов доступа, которые могут передаваться между двумя сторонами в виде JSON объекта. JWT широко используется для аутентификации и обмена данными между клиентом и сервером. В контексте данного приложения, JWT используется для создания и проверки токенов аутентификации, обеспечивая безопасную передачу данных между сервером и клиентом.
6. MongoDB: NoSQL база данных, основанная на документах, которая хранит данные в формате BSON (бинарное JSON). MongoDB отлично подходит для хранения и обработки больших объемов данных, таких как информация о пользователях, статьях, комментариях и оценках.

Использование этих технологий в совокупности позволяет создать эффективное и мощное веб-приложение для публикации и оценки научных статей, обеспечивая быстродействие, безопасность и удобство использования для пользователей.

3. Реализация проекта

3.1. Проектирование схемы базы данных

Была разработана схема базы данных, которая включала в себя следующие таблицы:

1. Коллекция "Users" (Пользователи): уникальный идентификатор пользователя, имя пользователя, email адрес пользователя, хэшированный пароль пользователя.
2. Коллекция "Articles" (Статьи): уникальный идентификатор статьи, идентификатор автора статьи, название статьи, ключевые слова, краткое описание статьи, текст статьи.
3. Коллекция "Comments" (Комментарии): уникальный идентификатор комментария, идентификатор статьи, идентификатор пользователя.
4. Коллекция "Ratings" (Оценки статей): идентификатор статьи, идентификатор пользователя, оценка статьи.

3.2. Аутентификация и Авторизация

Для реализации аутентификации и авторизации пользователей на платформе для публикации и оценки научных статей, следует следующий подход:

При регистрации нового пользователя, система запрашивает у него уникальное имя пользователя, адрес электронной почты и пароль. Пароль должен быть захеширован перед сохранением в базу данных. Затем, при попытке входа в систему пользователь должен предоставить свои данные. Система проверяет на соответствие данным в базе данных. Если данные верны, пользователь считается аутентифицированным.

После успешной аутентификации сервер создаёт JWT токен, содержащий идентификатор пользователя и другие необходимые данные. Токен отправляется клиенту и включается в заголовок Authorization для последующих запросов.

С помощью JWT токенов система проверяет права доступа пользователя к определенным функциям и ресурсам. Например, только автор может редактировать или удалять свои статьи, администратор может модерировать комментарии и т.д. Если пользователь имеет права доступа к определенному действию, система должна разрешить выполнение этого действия. Если пользователь не имеет необходимых прав доступа, система должна отклонить запрос и вернуть ошибку авторизации.

3.3. Управление контентом

Серверная (backend) часть приложения ответственна за управление контентом, предоставляя возможности клиентской части для создания, чтения, обновления и удаления контента согласно установленным правам доступа. Это включает в себя разработку API, который обрабатывает запросы от клиентской части приложения и взаимодействует с базой данных для выполнения операций с контентом.

При создании статьи или другого контента через API, данные о новом контенте отправляются на сервер, где происходит их валидация и сохранение в базу данных. Backend также отвечает за обработку запросов на чтение контента, возвращая клиенту запрошенную информацию из базы данных.

Помимо этого, backend реализует логику управления контентом, такую как категоризация и тегирование статей, а также управление авторством и правами доступа. Например, система аутентификации и авторизации обеспечивает контроль над тем, кто может создавать, редактировать или удалять контент. Также реализуется возможность архивирования статей для временного скрытия или удаления из общего доступа.

При обработке запросов на редактирование или удаление контента, backend проверяет права доступа текущего пользователя и в соответствии с этим выполняет запрошенные операции. Он также отвечает за обработку комментариев и обратной связи, позволяя пользователям добавлять комментарии к статьям и управлять ими.

3.4. Тестирование

Каждый отдельный модуль тестируется с помощью программы Postman. Для начала определяются различные сценарии использования API, такие как создание, чтение, обновление и удаление контента. Затем создается коллекция тестов в Postman, где для каждого сценария создаются соответствующие запросы с указанием параметров и ожидаемых ответов. Для каждого запроса также могут быть написаны тестовые скрипты на JavaScript, которые выполняются после отправки запроса и проверяют его результаты. После написания тестов и настройки коллекции их можно запустить в Postman, где инструмент выполнит запросы и выполнит проверки согласно заданным скриптам. Анализ результатов тестирования позволяет выявить ошибки и проблемы в API.

Заключение

Результатом исследования стал анализ технологий и средств разработки и тестирования веб-приложений. На основе этого анализа были выбраны оптимальные инструменты, необходимые для реализации серверной части проекта. Этот выбор был основан на критериях эффективности, производительности, расширяемости и совместимости с поставленными требованиями проекта.

Литература

1. Кантелон М. Node.js в действии / М. Кантелон [и др.]. – Санкт-Петербург : Питер, 2014. – 396 с.

2. Mead A. Learning Node.js Development: Learn the fundamentals of Node.js, and deploy and test Node.js applications on the web / A. Mead. – Бирмингем : Packt Publishing Ltd., 2018. – 627 с.

3. Bell J. Express.js: A Progressive Node.js Framework / J. Bell [и др.]. – Бирмингем : Packt Publishing Ltd., 2019. – 317 с.

УДК: 004.627

Алгоритм декомпрессии LZO1X

Д.М. Старухин

Воронежский государственный университет

Введение

Сжатие данных активно используется в вычислительных системах. Для повышения скорости работы с данными, хранящихся на носителях информации, повышения скорости передачи данных по сети и уменьшения размера хранимых данных, используются алгоритмы сжатия данных. В данной работе рассматривается алгоритм декомпрессии LZO1X. Алгоритмы LZO входят в семейство алгоритмов Lempel-Ziv. Декомпрессор LZO1X позволяет быстро распаковывать данные, не используя никаких дополнительных буферов, кроме выходного. По сравнению с другими модификациями алгоритма LZO, LZO1X обладает подходящими параметрами для большинства применений. Указанный алгоритм широко применяется в реальных информационных системах (например, ядро Linux, файловая система BtrFS, база данных MySQL и т. д.). Алгоритм работы декомпрессора LZO1X на сегодняшний день практически не описан в литературе, данная работа призвана исправить данное упущение.

1. Алгоритмы сжатия семейства Lempel-Ziv

В 1977 году два ученых-исследователя из Израиля смогли разработать и предложить совершенно новый подход к теме алгоритмов сжатия данных. Яков Зив и Абрахам Лемпель выдвинули идею формирования «словаря» общих последовательностей данных. Сжатие данных происходит за счёт замены записей во входном потоке кодами заранее известных последовательностей из словаря. Эти алгоритмы называются соответственно LZ77 и LZ78 и являются основными алгоритмами семейства Lempel-Ziv (LZ). LZ77 и LZ78 позволяют передавать, сжатые этими алгоритмами файлы, без передачи словаря. Это означает, что получатель, как и отправитель перед процедурой распаковки формирует точно такой же словарь, следуя заранее известным правилам генерации словаря. Из недостатков данных алгоритмов можно отметить относительно большие временные затраты на формирование эффективного словаря [1].

На основе алгоритмов сжатия Lempel-Ziv были разработаны другие усовершенствованные алгоритмы сжатия. Семейство LZ сейчас является довольно большим – количество различных алгоритмов сжатия насчитывает больше 20. К ним, к примеру, относятся:

- LZ77
- LZ78
- LZW
- LZO
- LZMA
- DEFLATE

- LZSS
- LZ4
- Zstd
- И многие другие [3].

На рис. 1 приведено упрощенное дерево семейства Lempel-Ziv.

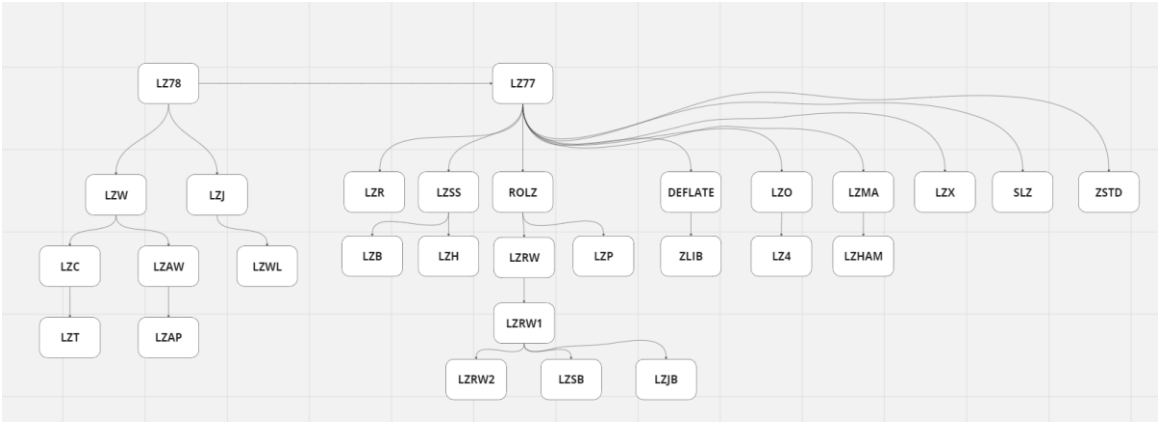


Рис. 1 - Примерное дерево семейства Lempel-Ziv

Семейство LZ активно совершенствуется и расширяется. Различные алгоритмы данного семейства используются в обширном спектре информационных технологий и являются неотъемлемой частью их работы. Можно привести очень большой список примеров, где семейство алгоритмов сжатия LZ применяется [4]:

- В операционных системах (Linux, FreeBSD, Illumos, coreboot, SmartOS, ZephyrOS);
- В файловых системах (OpenZFS, SquashFS, EROFS, HAMMER2, LessFS, LeoFS, GNU GRUB);
- В Big Data (Hadoop, Cassandra, Hbase, Spark, Hustle);
- В поисковых движках (Lucene, solr, Scalyr, Kafka, Groonga);
- В базах данных (MySQL, RocksDB, Tokudb, Redis, Delphix, infiniSQL, Sophia, OlegDB, GemStone, Percona, PostgreSQL);
- В графике (nVidia, Enlightenment, Xpra, OpenVDB, Pixlnsight, Scaplib, kanzi, pgBackRest);
- В интернет-приложениях (openVPN, SimpleLink, Netty, dovecot, virtualHere, linBit, Embulk, NeoMutt);
- И в др.

2. Мультилатформенный алгоритм сжатия данных Lempel-Ziv-Oberhumer

2.1. Общие сведения

Lempel-Ziv-Oberhumer (или же LZO) — это мультилатформенный алгоритм сжатия данных, который позволяет быстро сжимать и распаковывать данные без потерь. Алгоритм

был выпущен в 1997 году Маркусом Оберхеймером. Алгоритмы LZO входят в семейство Lempel-Ziv. Для декомпрессии не нужна дополнительная память, не считая буферов для сжатых и распаковываемых данных. LZO также можно использовать в многопоточной среде [5].

LZO сжимает информацию, путём замены исходных данных инструкциями. Инструкция представляет собой набор байтов, в котором содержится вся необходимая информация, с помощью которой можно распаковать входной поток, содержащий инструкции и литералы в исходные данные. Инструкции состоят из отдельных полей, имеющих определённое значение, некоторые поля занимают всего пару бит. Это позволяет в одном или нескольких байтах инструкции разместить всю необходимую информацию для управления процессом распаковки. В инструкции может содержаться:

- количество литералов во входном потоке, которые нужно скопировать после инструкции;
- расстояние в байтах от указателя на последний байт выходных данных, до места, откуда нужно копировать уже распакованные данные (частично распакованные данные выступают в роли словаря);
- количество байтов, которые нужно скопировать из словаря.

Как было сказано выше, частично распакованные данные могут выступать в роли словаря. Инструкции не только считывают необходимую информацию из входного потока, но и позволяют использовать данные из выходного потока распакованных данных, таким образом можно сказать, что словарь расширяется по мере процесса распаковки.

Входной поток, сжатый алгоритмом LZO, начинается с байта, в котором определяется, какое действие нужно выполнить первым. Обычно первый байт является первой инструкцией, которая требует скопировать определенное количество байт из входного буфера до следующей инструкции. Алгоритм разбивается на состояния, когда алгоритм выполняет действия, указанные в инструкции, и когда алгоритм готов прочесть и разобрать следующую инструкцию. В сжатой информации инструкции идут друг за другом, иногда содержа после себя дополнительные литералы для копирования, а иногда не имея их вовсе. Количество инструкций не указывается в сжатом файле, и их количество может быть произвольным, также как и размер одной инструкции. Сжатый файл всегда завершается последней инструкцией, стоящей в конце файла, которая указывает алгоритму копировать с конца словаря.

Алгоритм распаковки начинается с получения на вход сжатых данных в виде указателя на входной буфер и размер этого буфера. Алгоритму также требуется передать указатель на выходной буфер и размер этого буфера. Алгоритм не изменяет входной буфер [3].

LZO имеет большое количество модификаций и реализаций. В рамках статьи будет рассмотрена версия LZO1X, которая обладает подходящими параметрами для большинства применений [6]. Первая цифра после слов LZO означает категорию данного алгоритма. Данная категория означает, что алгоритм имеет формат сжатых данных, которые строго выровнены по байтам. Различие в логике LZO и LZO1X является минимальным [7].

2.2. Декодирование самого первого байта файла LZO1X

Как говорилось ранее, сжатые данные алгоритмом LZO1X должны содержать первый байт, который может быть полной или сокращенной инструкцией или байтом установки bitstream’a. Всего разновидностей первого байта данных три и каждый вид относится к конкретному диапазону. Разберем каждый диапазон первого байта данных.

Если первый байт имеет значение в диапазоне [0..16], это означает, что нужно принять первый байт, как инструкцию.

Если первый байт имеет значение 17, то это означает, что следующий байт указывает версию bitstream. Если первый байт файла будет иметь это значение и длина входящих данных больше 5 байт (т.к. это минимально возможная длина файла с включенным bitstream'ом), то следующий байт содержит версию bitstream. Если в следующем байте будет находиться 0, то версия bitstream будет 0, то есть как будто bitstream не установлен. Для простоты, будет рассматриваться только 1-ая версия bitstream. Bitstream 1-ой версии необходим, когда в данных содержится много нулей.

Если первый байт имеет значение в диапазоне [18..255], это означает, что нужно скопировать от 1 до 237 литерала из входного буфера in. Количество копируемых литералов будет равно:

$$state = Byte_{First} - 17$$

где $Byte_{First}$ — значение первого байта [3].

2.3. Виды инструкций LZO1X

Возможности каждой инструкции ограничиваются её типом. Каждый тип инструкции может копировать определенное количество байт данных на допустимом расстоянии от конца выходного буфера и копировать определенное число литералов из входного буфера. Тип инструкции определяется позицией первого ненулевого бита в первом байте инструкции. На рис.2 приведен пример нахождения первого значащего бита в числе 67

Рис. 24 Пример нахождения первого значащего бита в числе 67

Существует пять видов инструкций. Для удобства понимания первый байт инструкции будет представлен в двоичном представлении. Это необходимо для возможности просмотреть расположения битов отдельных параметров инструкции. В скобках будет указан возможный диапазон значений этого типа инструкций. Перечислим возможные типы инструкций.

0 0 0 0 X X X X [0..15] – этот тип инструкций является единственным видом инструкций, который требует от алгоритма обратить внимание на количество копируемых литералов из входного буфера предыдущей инструкцией (другими словами, состояние предыдущей инструкции). В данном случае представлен общий вид инструкции этого диапазона. От состояния предыдущей инструкции будет зависеть расположение битов инструкции и ее поведение. Поэтому стоит обратить внимание на всевозможные случаи поведения инструкции из этого диапазона.

Если состояние предыдущей инструкции было равно нулю, то первый байт инструкции

раскладывается следующим образом:

0 0 0 0 L L L L [0..15] – 4 младших бита первого байта инструкции выделены под поле «L». Данный подвид инструкции может скопировать длинную строку литералов из входного буфера. Данный вид инструкции занимает минимум 1 байт. Количество копируемых литералов может определятся в зависимости от значения L. Если $L = 0$, то алгоритм, начинает считать нулевые байты во входном потоке после данной инструкции, пока не встретит ненулевой байт. После вычисления количества нулевых байтов, высчитывается количество байтов, которые нужно скопировать из входного буфера в словарь по формуле:

$$length = 18 + (C_{zero\_bytes} * 255) + Byte_{non\_zero}$$

где $length$ — количество копируемых литералов, C_{zero_bytes} — количество нулевых байт, и $Byte_{non\_zero}$ — значение встреченного алгоритмом ненулевого байта. Если $L \neq 0$ то количество байт высчитывается под другому:

$$length = 3 + L$$

Приведём два примера. Если первый байт инструкции равен 0x0D(13), то т. к. первый байт ненулевой, то количество необходимых копируемых литералов равно 16 ($13+3$). Если же алгоритм, ожидая инструкцию, обнаруживает последовательность байтов 0x00 0x00 0x13, то, т. к. первый байт нулевой, алгоритм запускает подсчёт нулевых байтов и считает количество нулей игнорируя первый байт, поэтому количество необходимых копируемых литералов будет равно:

$$length = 18 + (1 * 255) + 19 = 292$$

Состояние инструкции или же количество копируемых литералов из входного буфера будет стоять 4. Это в данном случае означает, что инструкция не будет копировать дополнительные литералы из входного буфера.

Если предыдущая инструкция копировала от 1 до 3 литералов из входного буфера, то первый байт инструкции 0 0 0 0 X X X X декодируется другим образом:

0 0 0 0 D D S S [0..15]: - Данный вид инструкции приведёт к копированию двух байт из выходного потока на расстоянии до 1 КБ. Количество копируемых байт в этом виде инструкции всегда фиксировано. Первый байт определяет количество копируемых литералов с помощью битового поля «S» и расстояние в байтах от указателя на конец выходного потока, откуда необходимо копировать данные из выходного буфера. Помимо первого байта инструкции для вычисления расстояния используется второй байт, поэтому данная инструкция будет содержать два байта. Второй байт полностью выделен под битовое поле «H», которая необходима для расчёта расстояния в байтах, откуда нужно копировать данные из выходного буфера. Такое расстояние высчитывается по следующей формуле:

$$dist = (H << 2) + D + 1$$

Промежуточные значения «D» и «S» могут иметь значения от 0 до 3, «H» может принимать значения 0 до 255. Количество копируемых литералов из входного буфера будет полностью взято из битового поля «S».

К примеру, алгоритм обнаружил инструкцию из следующих байт: 0x0D 0x10. Для удобства будем использовать временный указатель с именем m_pos, который изначально равен указателю конца выходного буфера. Также необходимы указатели ip — указатель входного буфера на начале инструкции и op — указатель на конец выходного буфера. Разложив первый байт инструкции, поле «D» имеет значение 12, а поле «S» имеет значение 1. Поле «H» приняло значение 16. Определив все промежуточные значения, можно получить расстояние от указателя выходного буфера:

$$dist = (16 << 2) + 12 + 1 = 77$$

Алгоритму необходимо передвинуть временный указатель m_pos от указателя конца выходного буфера на 77 байтов назад ($m\_pos = op - 77$). После этого начиная от m_pos нужно скопировать в конец выходного буфера два байта вперед, до позиции $m\_pos + 2$. В последнюю очередь, ip должен переместится за инструкцию и считать «S» литералов из входного буфера до следующей инструкции. Считав все литералы, можно считывать следующую инструкцию.

Бывают ситуации, когда предыдущая инструкция копировала 4 или даже больше литералов (состояние в данном значении выставлено в 4). Следующая инструкция будет двухбайтовой, где первый и второй байт будут выглядеть также, как и в предыдущем случае:

0 0 0 0 D D S S [0..15] - Данная инструкция в отличие от предыдущей, копирует 3 байта на расстоянии от 2 до 3 КБ. Количество копирования байтов также фиксировано. Расстояние до адреса, откуда нужно копировать байты из выходного буфера, вычисляется по следующей формуле:

$$dist = (H << 2) + D + 2049$$

Другие диапазоны первого байта инструкции не имеют зависимости от предыдущих инструкций. Вычисления количества копируемых литералов из входного потока необходимо для первых инструкций из диапазона [0..15]. Все следующие виды инструкций имеют единственный вид разложения (расположение битовых полей) в каждом диапазоне. Количество копируемых литералов из входного буфера всегда будет браться полностью из битового поля «S» (т.е. $state = S$).

0 0 0 1 H L L L [16..31]: - является инструкцией, которая состоит, как минимум, из 3 байт. Эта инструкция копирует блок байтов на расстоянии от 16 до 48 КБ. В первом байте инструкции есть два битовых поля: «H» и «L». Если L будет равно нулю, то алгоритм начнёт подсчёт количества нулей до первого ненулевого байта и будет вычисляться по формуле:

$$length = 9 + (C_{zero\_bytes} * 255) + Byte_{non\_zero}$$

где C_{zero_bytes} — количество нулевых байт и $Byte_{non\_zero}$ — значение ненулевого байта. Если же L больше нуля, то количество копируемых байт будет другое:

$$length = 2 + L$$

Если же версия bitstream'а стоит 1, то вычисление количества байт для копирования меняется. Если все биты битовых полей «D» и «H» выставлены в 1, то у инструкции появляется 4-ый байт, который заполняется полностью битовым полем «X». Количество байт для копирования вычисляется по следующей формуле:

$$length = ((X << 3) \mid L) + 4$$

К примеру, если bitstream установлен в 1, а алгоритм прочитал инструкцию 0x12 0x10 0x00 0x05, то последний байт будет являться «X», который равен 5. С помощью него можно высчитать $length = ((5 << 3) \mid 2) + 4 = (40 \mid 2) + 4 = 42 + 4 = 46$. Если бы bitstream был установлен в 0, то последний байт не входил бы в инструкцию и length было бы равно $2 + 2 = 4$.

Зачастую инструкции для вычисления количества байт для копирования из выходного буфера достаточно одного байта, но если количество байт для копирования будет необходимо больше, чем 10, то будут добавляться дополнительные байты. После вычисления длины идут два байта, склеенных вместе. Два байта имеют порядок Little-Endian (в данном случае это LE16) и имеют следующий формат: «D D D D D D D D : D D D D D D S S». Для того, чтобы получить промежуточные значения, хранящиеся в двух байтах, нужно изменить порядок между двумя байтами, к примеру, если два байта будут 0x01 и 0x02, то для того, чтобы с помощью них можно было получить два битовых поля «D» и «S» их нужно представить как 0x02 и 0x01. Получив оставшиеся переменные в двух байтах, можно вычислить расстояние, откуда нужно будет копировать байты из выходного буфера по формуле:

$$dist = 16384 + (H << 14) + D$$

0 0 1 L L L L L [32..63]: - следующий диапазон инструкций копирует маленький блок байтов на расстоянии до 16 КБ. Эта инструкция схожа с предыдущей в реализации подсчета нужного числа байтов из выходного буфера. Алгоритм вычисления количества байт необходимых для копирования вычисляется по схожей логике: если $L = 0$, то:

$$length = 33 + (C_{zero\_bytes} * 255) + Byte_{non\_zero}$$

Если $L \neq 0$:

$$length = 2 + L$$

После вычисления расстояния идут также два байта в формате LE16. Расстояние откуда нужно будет копировать данные будет вычисляться так:

$$dist = D + 1$$

0 1 L D D D S S [64..127] — инструкция будет копировать 3-4 байта расстоянием до 2КБ. Инструкция состоит из двух байт: второй байт будет полностью выделен под переменную H. Дистанция будет вычислена по формуле:

$$dist = (H << 3) + D + 1$$

Количество копируемых байт будет вычислено уже по другой формуле:

$$length = 3 + L$$

1 L L D D D S S [128..255] — Копирует 5-8 байтов до 2КБ расстоянием. Эта инструкция состоит из двух байт, второй байт выделен под переменную H. Количество копируемых байт будет равно:

$$length = 5 + L$$

Расстояние, откуда будут копироваться байты, будут вычисляться по следующей формуле:

$$dist = (H << 3) + D + 1$$

Это все возможные варианты инструкций. Некоторые из них чаще используются других, даже возможен вариант, когда не каждый тип инструкций может быть встречен в сжатом файле [3].

2.4. Представление работы алгоритма LZO1X в виде блок схемы

Для более лучшего понимания, в этом разделе представлены блок-схемы алгоритма декомпрессии LZO1X. Последовательность действий может отличаться от реализации алгоритма, а также некоторые общие шаги у разных видов инструкции обобщены, для простоты восприятия. Разделим всю блок-схему на более простые и рассмотрим каждую по отдельности. На рис. 3 можно увидеть начало алгоритма.

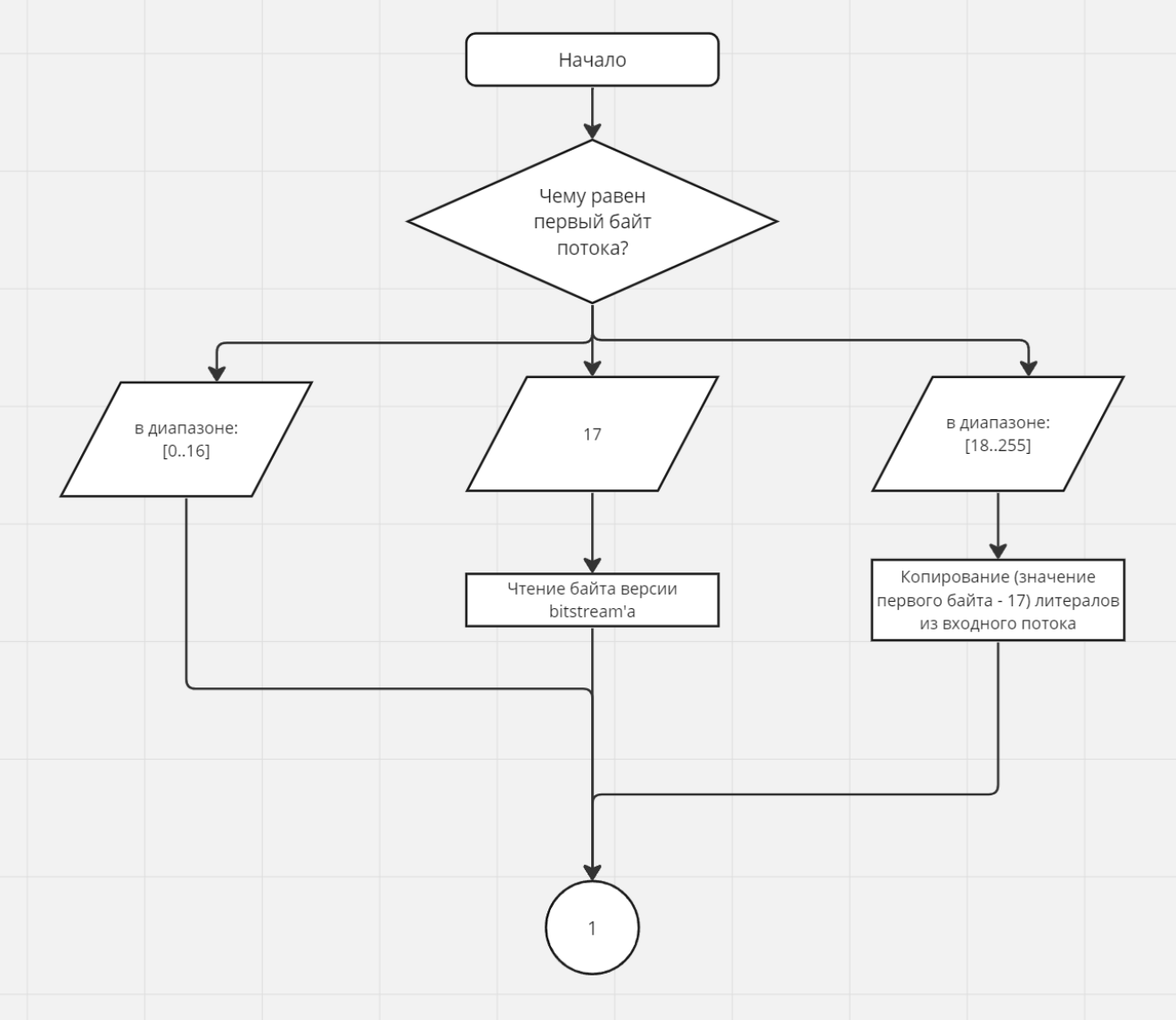


Рис. 3 Начало алгоритма декомпрессии LZO1X

На рис.4. изображена часть определения первого байта к определенному диапазону значения.

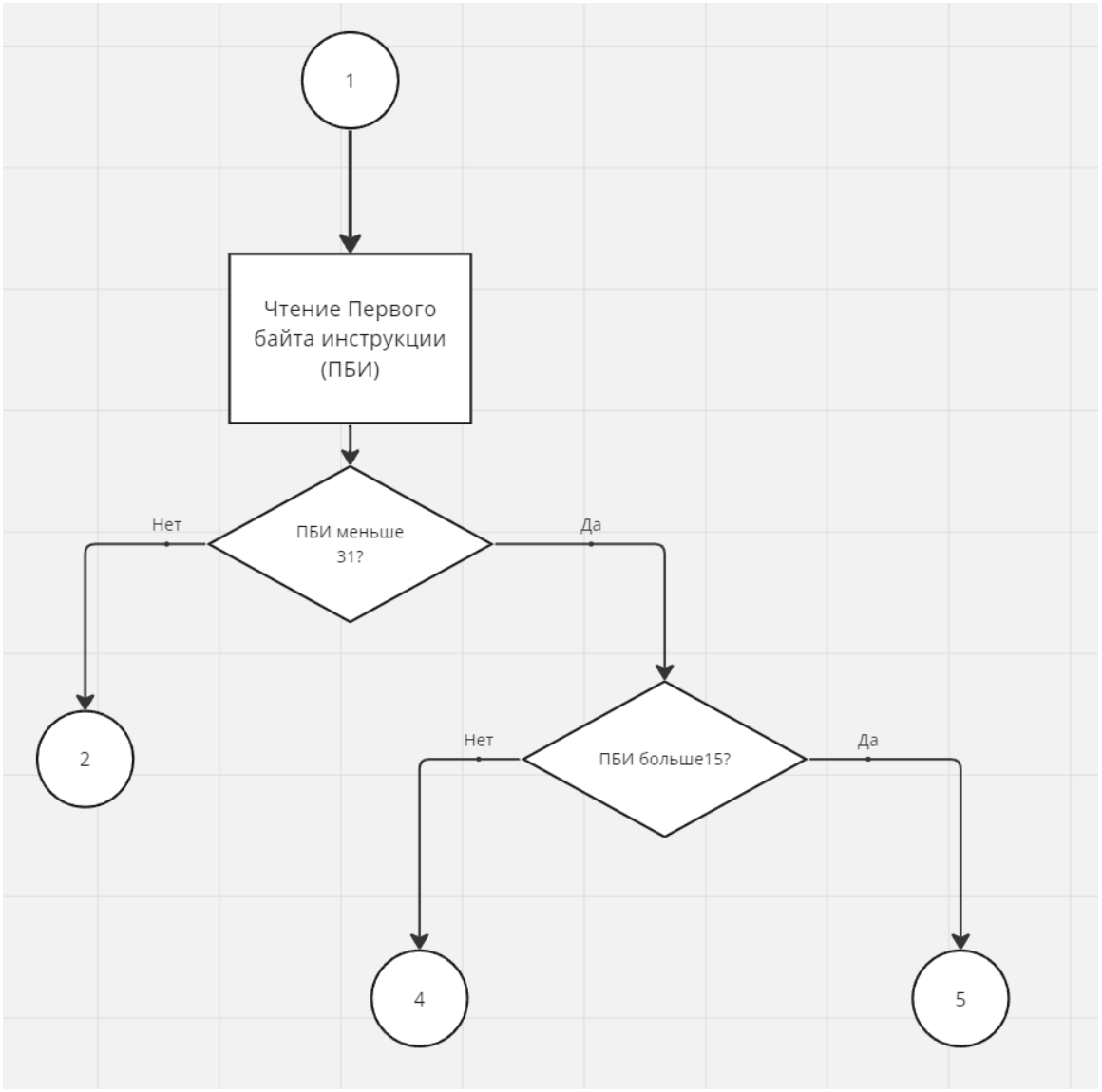


Рис. 4 Определение принадлежности первого байта к определенному диапазону

В зависимости от значения первого байта инструкции, выполняются определенная часть блок-схемы. На рис. 5 изображена часть блок-схемы, которая выполняется, если первый байт инструкции находится в диапазоне [32..255].

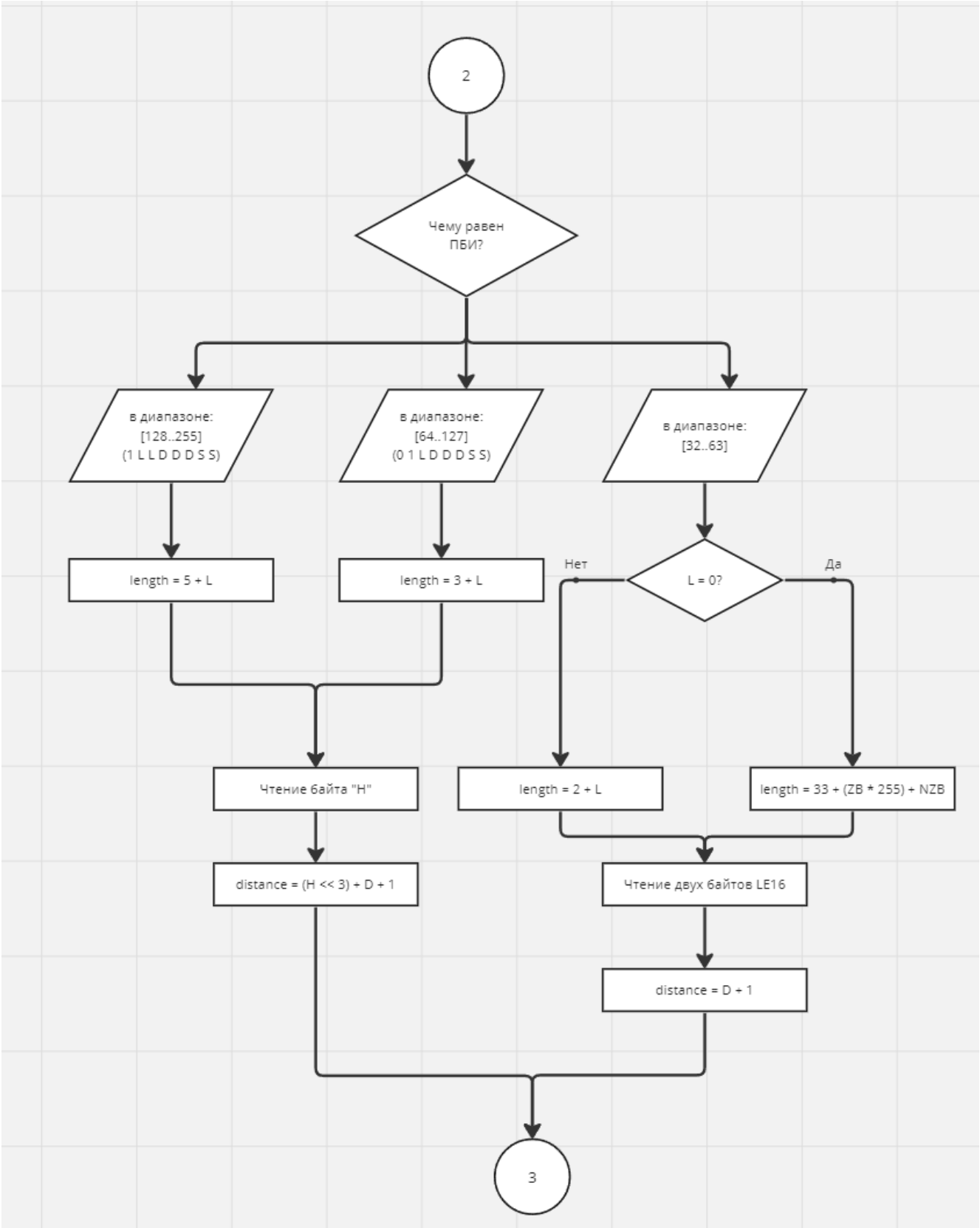


Рис. 5 Выполняемая блок-схема, когда первый байт инструкции в диапазоне [32..255]

После выполнения блок схемы из рис.6, выполняется заполнение выходного буфера распакованными данными. Блок-схему последовательности этих действий можно рассмотреть на рис.6.

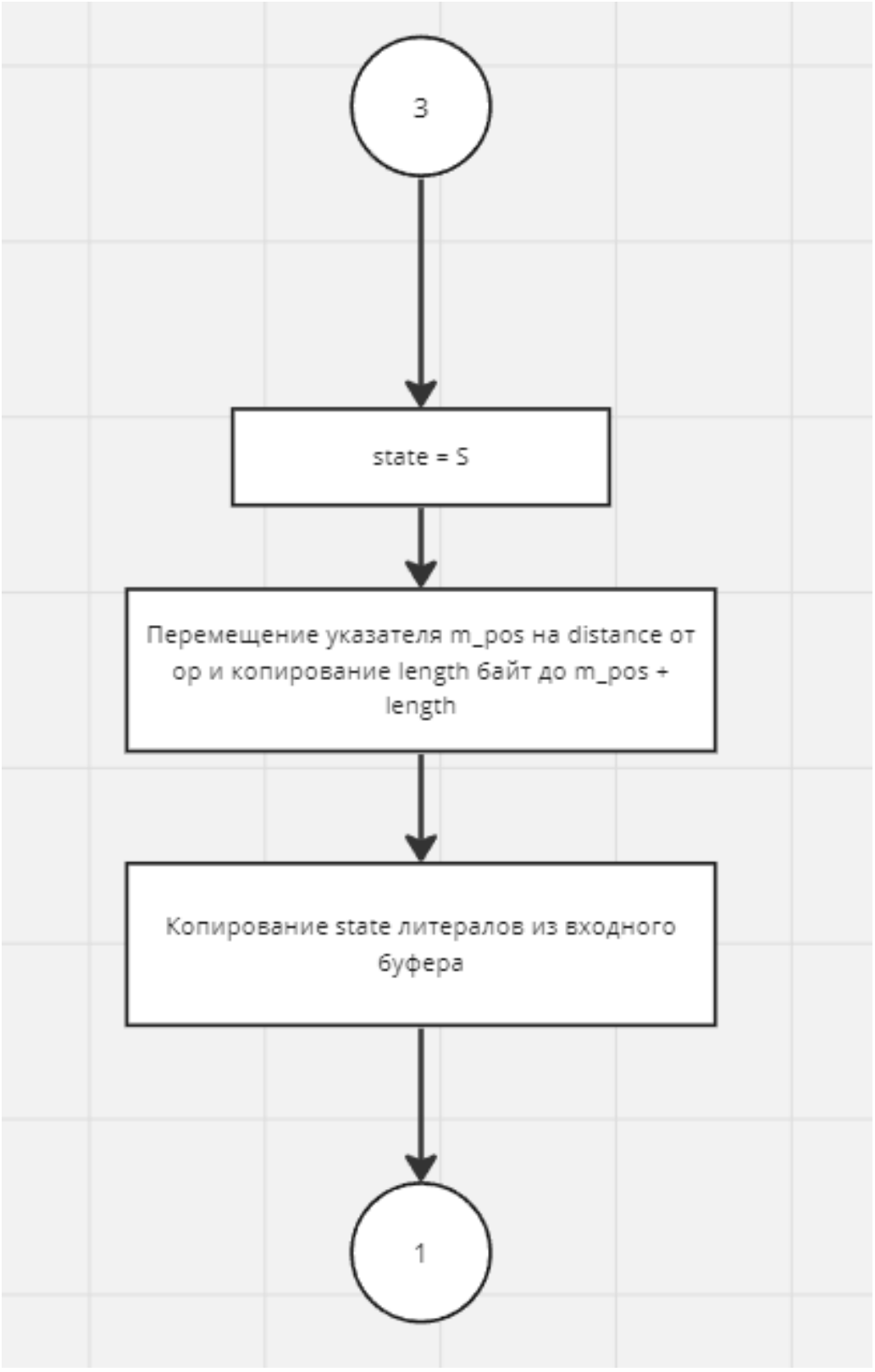


Рис. 6 Блок схема добавления распакованных данных в выходной буфер

Если первый байт инструкции является числом меньшим или равным 15, то выполняется блок-схема, изображенная на рис.7.

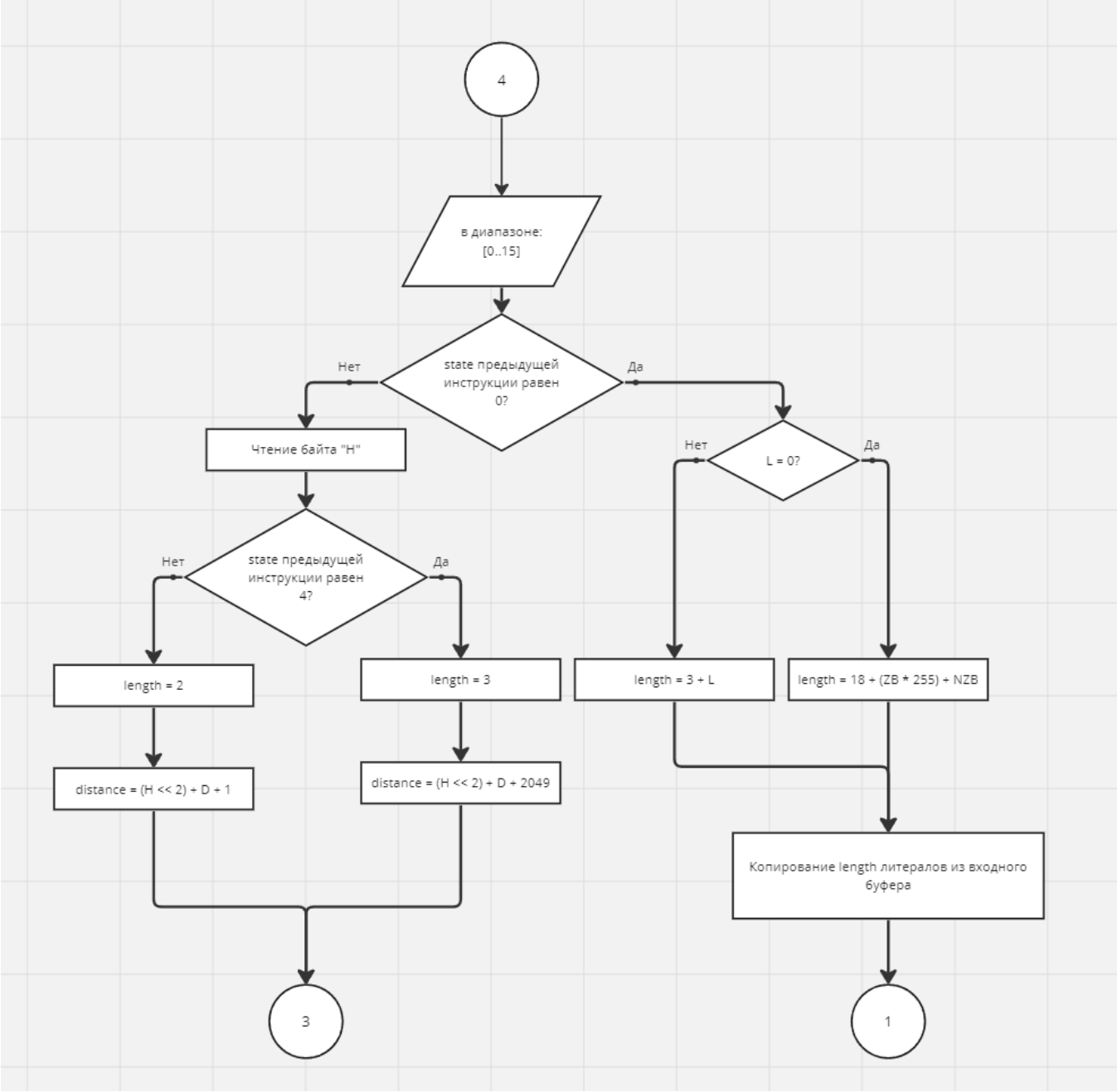


Рис. 7 Выполняемая блок-схема, когда первый байт инструкции меньше или равен 15

Если первый байт инструкции является числом в диапазоне [16..31], то выполняется блок-схема, которую можно увидеть на рис.8.

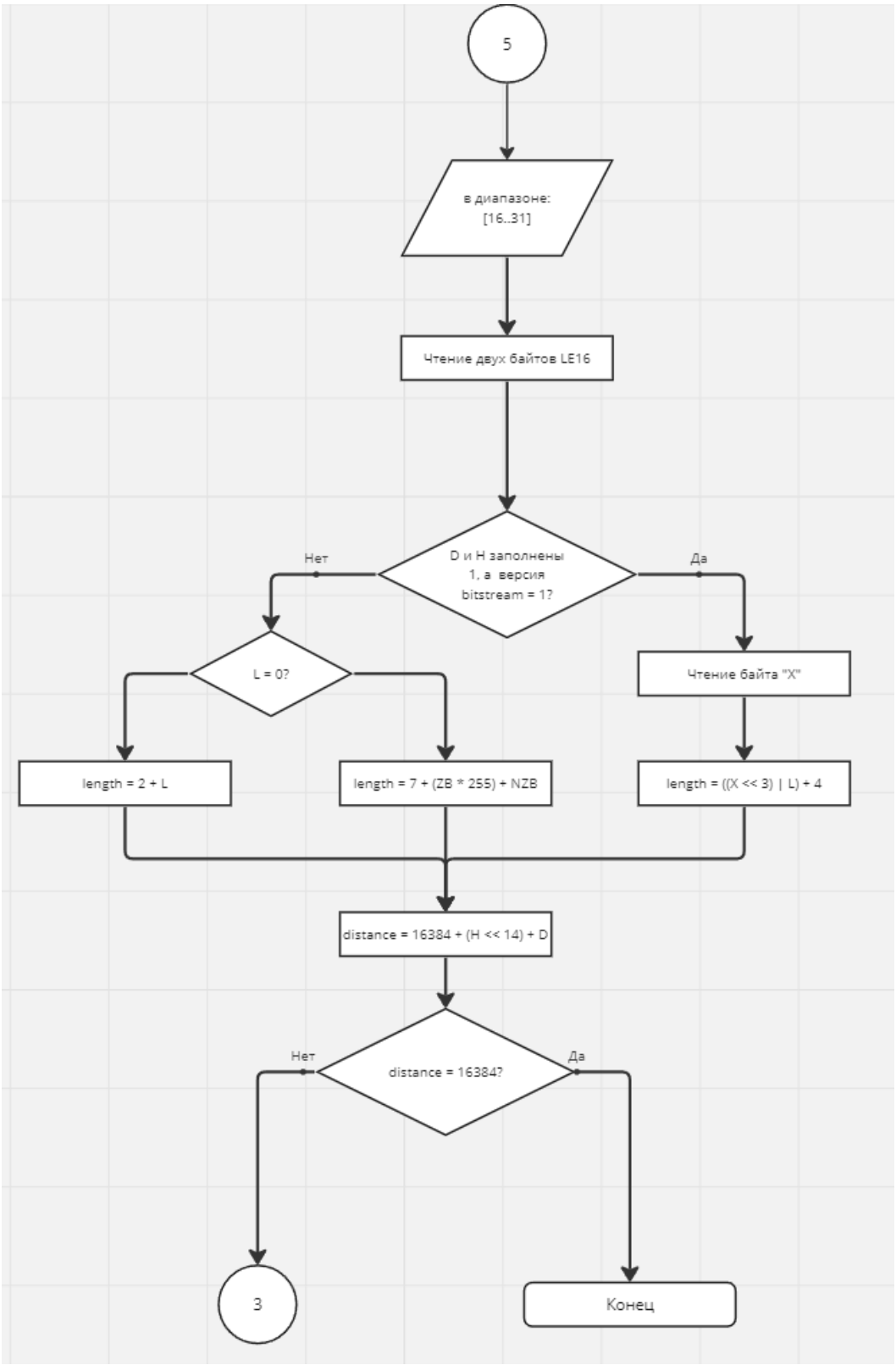


Рис. 8 Выполняемая блок-схема, когда первый байт инструкции в диапазоне [16..31]

По рис.3-8, можно полностью получить алгоритм декомпрессии LZO1X.

Заключение

В рамках статьи была разобрана работа алгоритма декомпрессии LZO1X. Была рассмотрена логика распаковки алгоритма, а также ее тонкости работы.

Список использованных источников

1. История развития теории сжатия информации [Электронный ресурс]. - Режим доступа: http://compression.ru/arctest/descript/comp-hist.htm - (Дата обращения 18.04.2024)
2. History of Lossless Data Compression Algorithms [Электронный ресурс]. - Режим доступа: https://ethw.org/History_of_Lossless_Data_Compression_Algorithm — (Дата обращения 18.04.2024)
3. LZO stream format as understood by Linux's LZO decompressor [Электронный ресурс]. - Режим доступа: https://www.kernel.org/doc/Documentation/lzo.txt — (Дата обращения 18.04.2024)
4. LZ4 [Электронный ресурс]. - Режим доступа: https://lz4.org/
5. LZO — Режим доступа: https://ru.wikipedia.org/wiki/LZO (Дата обращения 18.04.2024)
6. LZO Use and introducion — Режим доступа: https://topic.alibabacloud.com/a/lzo-use-and-introduction_8_8_31155441.html — (Дата обращения 18.04.2024)
7. LZO.FAQ — Режим доступа: https://github.com/nemequ/lzo/blob/master/doc/LZO.FAQ — (Дата обращения 18.04.2024)

УДК 004.932.2

ИССЛЕДОВАНИЕ АЛГОРИТМОВ СГЛАЖИВАНИЯ ДЛЯ ВОССТАНОВЛЕНИЯ 3D МОДЕЛИ КОСТЕЙ ПО СНИМКАМ КТ

В. Н. Сырых, Е. В. Трофименко

Воронежский государственный университет

Введение

В данной работе было проведено исследование алгоритмов фильтрации, которые могут быть применены для восстановления трехмерных моделей костей по снимкам компьютерной томографии (КТ). Были рассмотрены такие методы фильтрации, как Гауссово размытие, медианный фильтр и билатеральный фильтр. В данной статье анализируется эффективность предложенных методик на основе анализа результатов полученных, при применении того или иного алгоритма сглаживания.

1. Алгоритмы фильтрации поверхности

Методы предобработки изображений позволяют преобразовывать изображения для улучшения их визуального восприятия, а также решать задачи изменения представления изображений для обеспечения их хранения, передачи, визуализации в электронном виде и дальнейшего анализа заложенной в них информации.

Несмотря на ряд преимуществ снимков КТ и возможность получения изображения частей тела в любой плоскости, заданная интерпретация получения данных, а именно установление соответствия пикселей изображений анатомическим участкам тела, остается актуальной. На Рис.1 пример КТ снимка без применения фильтрации

Проблема фильтрации компьютерным томографом осложняется тем, что функциональные структуры с одинаковой рентгеновской плотностью могут иметь различный диапазон значений интенсивности в зависимости от параметров проведенного КТ-исследования. Проблема сглаживания и шумоподавления является одной из самых актуальных и распространенных проблем в области обработки изображений.

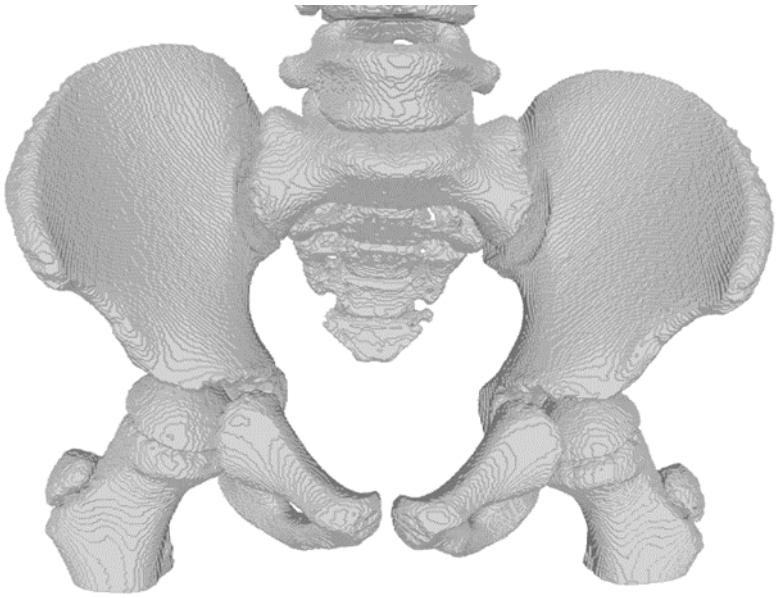


Рис 1. Модель без применения фильтрации.

1.1. Гауссово размытие

Гауссово размытие – это один из наиболее широко используемых методов размытия изображений, применяемый в цифровой обработке изображений. Он основан на математическом преобразовании, которое применяется к каждому пикселю изображения с учетом его окружения. При этом используется функция Гаусса, которая создает плавный, постепенно убывающий эффект размытия [1].

Преимущество гауссова размытия заключается в его способности сохранять детали изображения, при этом смягчая острые края и шум. Это делает его идеальным выбором для многих задач обработки изображений, таких как устранение шума, сглаживание кожи в портретах или создание эффекта глубины в фотографиях.

Несмотря на свою эффективность, гауссово размытие также имеет свои ограничения. Оно может быть недостаточно эффективным при размытии изображений с большим количеством деталей или шумом, и в таких случаях могут потребоваться более сложные методы обработки. Однако, в большинстве сценариев, гауссово размытие остается важным инструментом для достижения желаемого визуального эффекта.

Пример работы алгоритма с сигмой равной 1,5 приведён на Рис. 2.

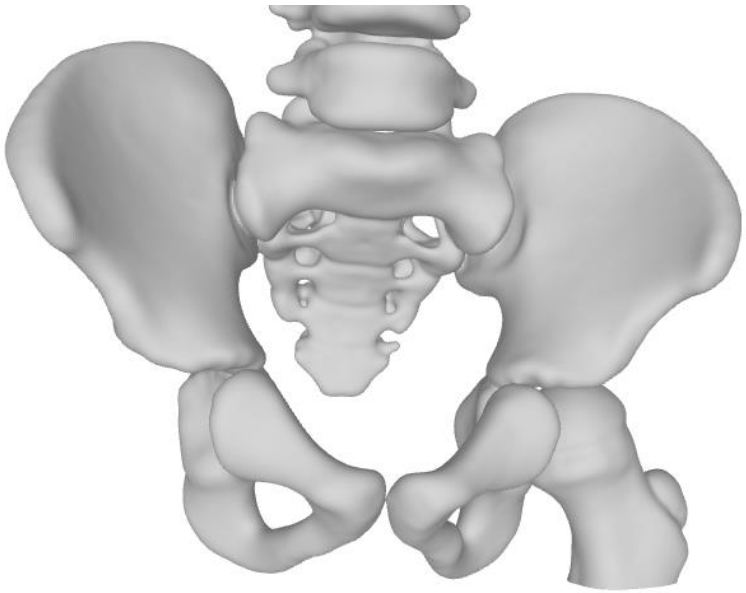


Рис 2. Результаты сглаживания Гауссовым размытием с ядром равным 1,5.

1.2. Медианный фильтр

Медианный фильтр – это метод обработки изображений, треугольных сеток, поверхностей, который используется для снижения шума и улучшения качества. В отличие от других методов размытия, медианный фильтр не использует математические операции, такие как среднее значение или функция Гаусса [2]. Вместо этого он заменяет каждый пиксель изображения медианным значением яркости в его окрестности.

Основная математическая идея медианного фильтра заключается в том, что он использует медиану, то есть значение, которое находится посередине упорядоченного списка значений. Это позволяет эффективно удалять выбросы или аномальные значения, которые могут быть вызваны шумом на изображении. Таким образом, медианный фильтр обеспечивает более точное сохранение краев и деталей изображения по сравнению с другими методами размытия.

Преимущество медианного фильтра заключается в его способности эффективно справляться с различными типами шума на изображении, включая шум соль и перец, который представляет собой случайное появление черных и белых пикселей. Благодаря этой особенности медианный фильтр часто используется в областях, где важно сохранить высокую четкость и детализацию изображения.

Так же есть и недостатки, самый главный недостаток – время работы, 30 секунд обрабатывается окно 3x3x1 (x, y, z) на Ryzen 7 5700x, что даёт очень слабый результат сглаживания (Рис. 3.), на больших окнах время работы исчисляется десятками минут.

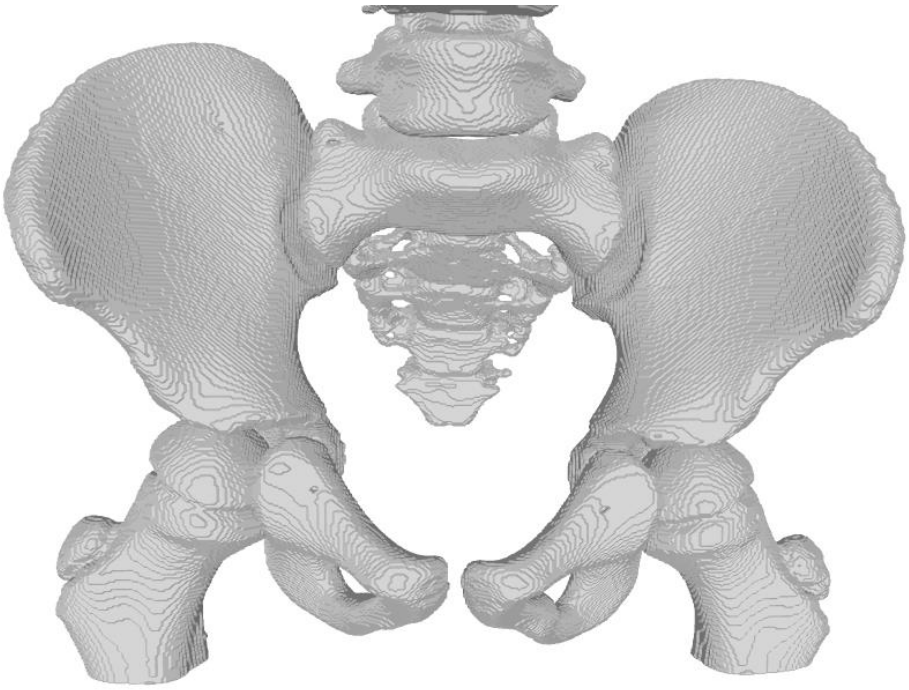


Рис 3. Результат сглаживания медианным фильтром при размере окна 3x3x1.

1.3 Билатеральный фильтр

Билатеральный фильтр – это метод обработки изображений, который сочетает в себе эффективные способы размытия и снижения шума, при этом сохраняя края и детали изображения. В отличие от простых методов размытия, таких как среднее значение или размытие Гаусса, билатеральный фильтр учитывает не только пространственное расположение пикселей, но и их интенсивность [3].

Основная математическая идея билатерального фильтра состоит в том, чтобы применять различные весовые коэффициенты к пикселям изображения в зависимости от их расстояния и интенсивности. Этот метод позволяет сохранять резкие края объектов, уменьшая размытие в этих областях, и одновременно сглаживать шум в менее важных участках изображения.

Преимущество билатерального фильтра заключается в его способности эффективно улучшать качество изображений без потери деталей и текстур. Он находит применение во многих областях, включая компьютерное зрение, фотографию и графический дизайн, где важно сохранить высокую четкость и естественный вид изображения при снижении шума. Однако билатеральный фильтр требует более вычислительных ресурсов по сравнению с некоторыми другими методами размытия изображений.

Недостатком данного фильтра так же является скорость работы, в алгоритме приходится обрабатывать слишком много параметров, что очень сильно повышает временную сложность алгоритма, на DICOM изображениях не удалось получить результат, поэтому на Рис. 4 показан общий пример работы алгоритма.

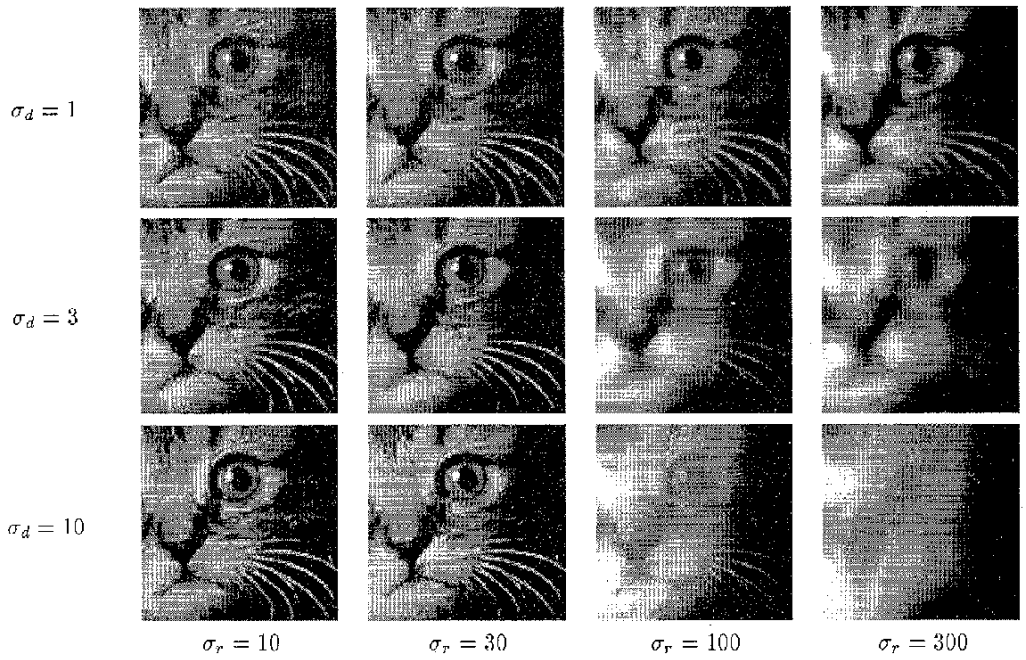


Рис 4. Результаты сглаживания билатеральным фильтром при различных значениях окрестности.

1.4 Полигональный фильтр с оконной функцией синуса

Полигональный фильтр с оконной функцией синуса – это метод обработки полигональных данных, который применяет технику сглаживания на основе синхронизированной функции (sinc), дополненной оконной функцией. Этот метод отличается от традиционных подходов сглаживания тем, что он более эффективно сохраняет важные геометрические характеристики объектов, такие как края и углы, при этом уменьшая шероховатости и артефакты на поверхности.

Идея метода заключается в применении фильтра синхронизации с модифицированным оконным профилем (например, Хэмминга или Велча), который помогает контролировать ширину полосы пропускания и уменьшить явления, такие как "звон", обычно ассоциируемые с синхронизированными фильтрами. Это позволяет более аккуратно обрабатывать геометрически сложные структуры, сохраняя их основные особенности при снижении шума и искажений.

Преимущество алгоритма заключается в его способности обеспечивать высококачественное сглаживание, что делает его идеальным для применений, где важно сохранение точности и четкости геометрических деталей, таких как медицинская визуализация, компьютерная графика и CAD системы. Он широко используется для предобработки данных перед выполнением более сложных операций, таких как реконструкция поверхностей или 3D печать. Так же данный алгоритм сглаживания сопоставим по скорости с алгоритмом Гаусса.

Результаты обработки приведены на рисунках 5 и 6.

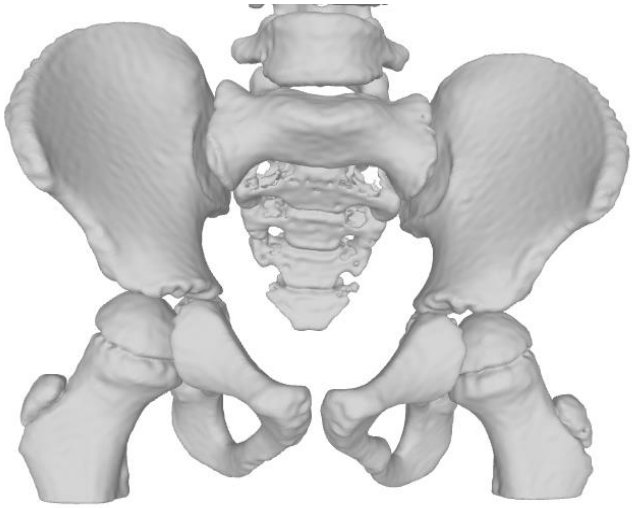


Рис 5. Результаты сглаживания полигональным фильтром с оконной функцией синуса при ста итерациях.

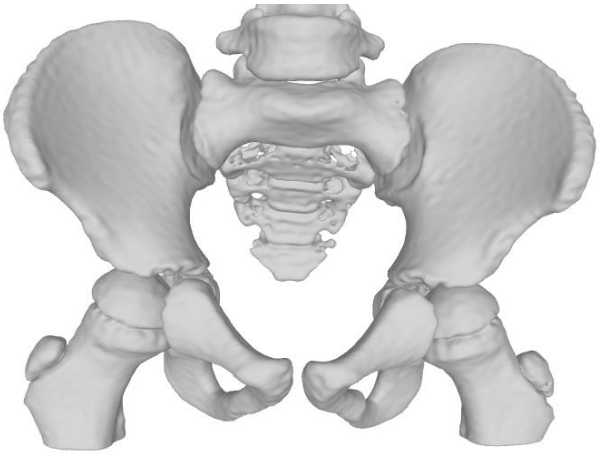


Рис 6. Результаты сглаживания полигональным фильтром с оконной функцией синуса при пятистах итерациях.

2. Анализ результатов

Применение сглаживания для улучшения качества 3D моделей значительно повышает их визуальное восприятие, делая модели более реалистичными и пригодными для детального анализа. Сглаживание можно применять в несколько этапов или итераций, чтобы достичь желаемого уровня детализации и качества.

Произведём сравнение полигонального фильтра с оконной функцией при 100 итерациях сглаживания к 3D модели и 500 из прошлого параграфа (Рис 5, 6) с полученной моделью без сглаживания (Рис. 1). Исходная модель без сглаживания содержит множество геометрических несовершенств, таких как острые углы, неровные поверхности и визуальный шум, что делает её неадекватной для визуального представления и анализа. Шум этот обусловлен дистанцией между снимками в DICOM изображениях.

При применении сглаживания, особенно после 500 итераций, можно заметить улучшение качества модели. Это достигается за счет сглаживания неровностей и уменьшения видимых дефектов поверхности. В результате модель выглядит более гладкой и четкой.

Сглаживание помогает устранить "шероховатости" поверхности, делая модель более эстетически привлекательной и удобной для работы.

При всех плюсах сильные неровности не были удалены и часть элементов выглядят угловатыми даже при 500 итерациях, следовательно, для улучшения качества изображения необходимо либо увеличивать количество итераций, либо комбинировать данный фильтр с каким-либо ещё.

Тем не менее полигональный фильтр с оконной функцией синуса показал себя как отличный инструмент, который способен быстро работать, сохранять детали модели, но при этом можно заметить, что сильные локальные аномалии данный алгоритм сгладить неспособен, требуется большое количество итераций, что сильно увеличивает время его работы.

Размытие Гаусса показало себя как отличный инструмент сглаживания, при подборе сигмы можно получить отличный баланс между количеством сохранённых деталей и качеством полученной модели.

В зависимости от конкретной задачи следует выбирать соответствующий метод сглаживания. Для общего снижения шума и быстрой обработки можно использовать гауссово размытие. Для удаления аномального шума на изображении лучше всего подходит медианный фильтр (но с оговоркой ресурсоёмкости). В случае, когда важно сохранить высокую четкость и детализацию изображения при снижении шума, рекомендуется применять полигональный фильтр с оконной функцией синуса.

Заключение

В результате проделанной работы были исследованы четыре алгоритма сглаживания: гауссово размытие, медианный фильтр, билатеральный фильтр, полигональный фильтр с оконной функцией синуса. Исходя из полученных результатов можно сделать вывод о том, что медианный фильтр и билатеральный фильтр являются очень ресурсоёмкими и их какая-либо эффективность сглаживания нивелируется продолжительностью их работы, что приводит даже к невозможности получить результаты, для этого требуется уменьшение количества полигонов модели, но это приведёт к тому, что она будет выглядеть нереалистично и схематично, а остальные алгоритмы вполне подходят для эффективного применения.

Литература

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.
2. Фурман Я.А., Кревецкий А. В., Передреев А. К., Роженцев А. А., Хафизов Р. Г., Егошина И. Л., Леухин А. Н. Введение в контурный анализ и его приложения к обработке изображений и сигналов. Изд. 2-е, испр. М.: Физматлит, 2003. 592 с.
3. Feudjio C. K., Tiedeu A., Noubeg M.-L., Gordan M., Vlaicu A., Domngang S. Extracting and smoothing contours in mammograms using Fourier descriptors // Biomedical Science and Engineering, 2014. No. 7. P. 119-129.
4. Hobby J. D. Smoothing Digitized Contours // Theoretical Foundations of Computer Graphics and CAD, Springer Berlin Heidelberg, 1988. P. 777–793.
5. Hu J., Yu D., Yan H. Structural Boundary Feature Extraction for Printed Character Recognition // Advances in Pattern Recognition: Joint IAPR International Workshops, SSPR’98 and SPR’98, Sydney, Australia, August 11-13, 1998, Proceedings, P. 500-507

УДК 004.273

РЕАЛИЗАЦИЯ ПАТТЕРНОВ «МЕДИАТОР» И «ДЕКОРАТОР» В C#

В. К. Сычев, М. В. Матвеева

Воронежский государственный университет

Введение

Нередко при программировании возникает потребность в использовании различных способов организации кода, таких как паттерны проектирования. Два из таких полезных паттернов – это «Медиатор» (Mediator) и «Декоратор» (Decorator).

Паттерн «Медиатор» [1] позволяет уменьшить связанность между объектами и централизовать логику взаимодействия в отдельном классе-медиаторе. Это особенно полезно, когда объекты имеют сложные взаимосвязи и требуется упорядоченное и централизованное управление коммуникацией. Медиатор выступает в роли посредника между объектами, обрабатывая передачу сообщений, и позволяет объектам работать независимо друг от друга.

Паттерн «Декоратор» [2] позволяет динамически добавлять новую функциональность или изменять поведение существующих объектов, не изменяя их исходного класса. Декоратор использует принцип композиции и оборачивает объект в другие объекты-декораторы, дополняя его функциональность. Этот паттерн эффективно применяется, когда требуется добавить функциональность объектам без внесения изменений в их исходный код. Он позволяет гибко комбинировать различные декораторы, чтобы получить необходимое поведение.

Использование паттернов проектирования, таких как «Медиатор» и «Декоратор», помогает создавать гибкие, расширяемые и модульные системы. Эти паттерны предлагают организационные решения, которые упрощают разработку, поддержку и масштабирование кода. Однако, всегда необходимо тщательно оценивать ситуацию и применять паттерны там, где они действительно целесообразны и улучшают архитектуру системы.

1. Паттерн «Медиатор»

На рис. 1 [1] представлен поведенческий паттерн проектирования «Медиатор» (Mediator), который позволяет установить связь и координацию между различными объектами, путем вынесения логики взаимодействия в отдельный класс-медиатор. Он помогает уменьшить зависимости между классами, распределяет ответственность за взаимодействие и упрощает поддержку кода.

Основная идея паттерна заключается в том, что все коммуникации и взаимодействие между объектами происходят через централизованный класс-медиатор, который понимает, какие объекты взаимодействуют между собой и как передавать им сообщения. Вместо того, чтобы объекты напрямую обращались друг к другу, они отправляют сообщения медиатору, который затем решает, какой объект должен получить это сообщение.

Преимущества использования паттерна «Медиатор» включают:

- уменьшение связанности между объектами, так как они не зависят напрямую друг от друга;
- централизация и управление взаимодействиями между объектами в одном месте, что делает код более читаемым и поддерживаемым;

- упрощение добавления новых компонентов или изменения логики взаимодействия без необходимости изменения всех классов-участников;
- снижение дублирования кода, поскольку логика взаимодействия находится в медиаторе.

Однако, следует помнить, что паттерн «Медиатор» может добавить сложность и зависимость от медиатора. Поэтому он должен использоваться тогда, когда коммуникация между объектами становится сложной и требует упорядоченного и централизованного подхода.

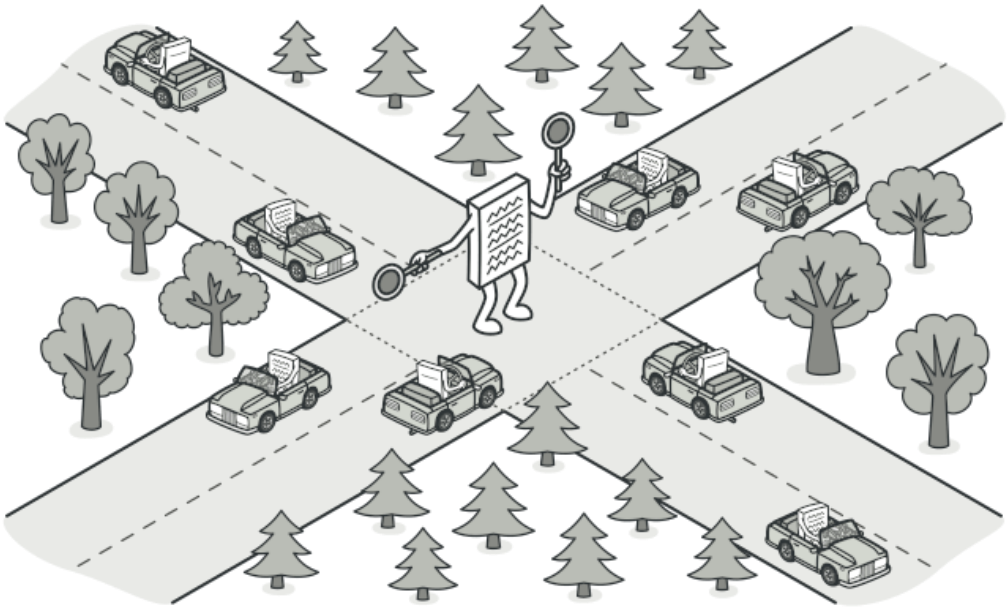


Рис. 1. Паттерн «Медиатор»

2. Пример использования паттерна «Медиатор»

Рассмотрим пример использования паттерна «Медиатор» в системе умного дома. В данной системе «Умный дом» (SmartHome) действует в качестве класса-медиатора и координирует взаимодействие между несколькими компонентами, такими как датчики движения (MotionSensor), устройства освещения (Lighting) и термостаты (Thermostat).

На рис. 2 изображена диаграмма классов умного дома. Датчик движения обнаруживает движение и отправляет уведомление медиатору – умному дому. Умный дом, в свою очередь, решает, какие действия должны быть выполнены на основе полученной информации о движении. Он может управлять освещением, включая или выключая свет в зависимости от наличия движения, а также может управлять термостатом, регулируя температуру в комнате.

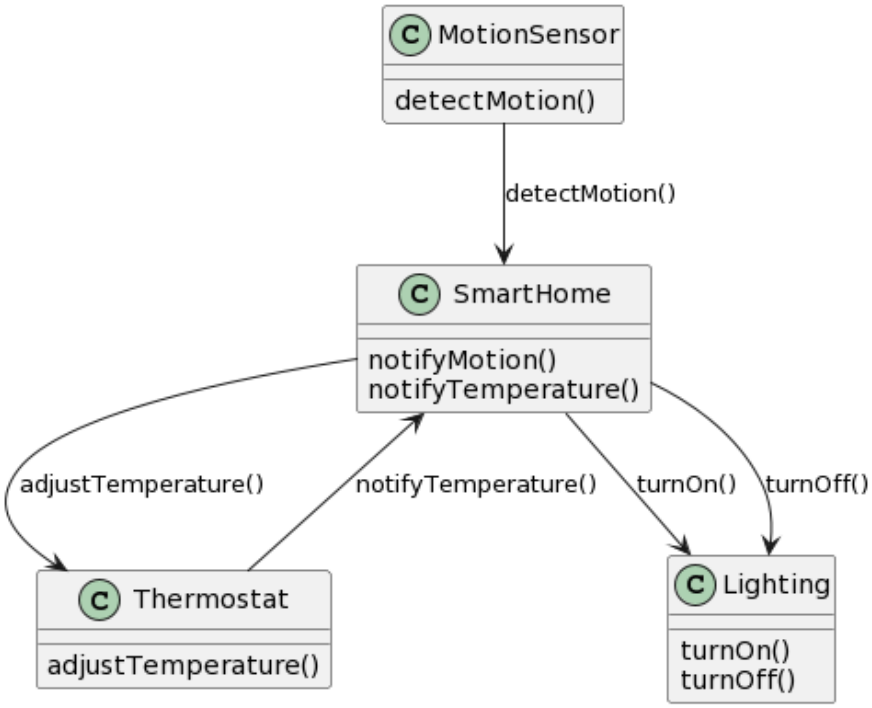


Рис. 2. Диаграмма классов системы умного дома

Паттерн «Медиатор» позволяет упростить взаимодействие между различными компонентами системы умного дома. Применение этого паттерна позволяет уменьшить связанность между классами, делает код более модульным и облегчает добавление новых компонентов или изменение логики взаимодействия без необходимости изменения всех классов-участников.

3. Паттерн «Декоратор»

На рис. 3 [2] представлен структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки» – матрешки (рис. 3).

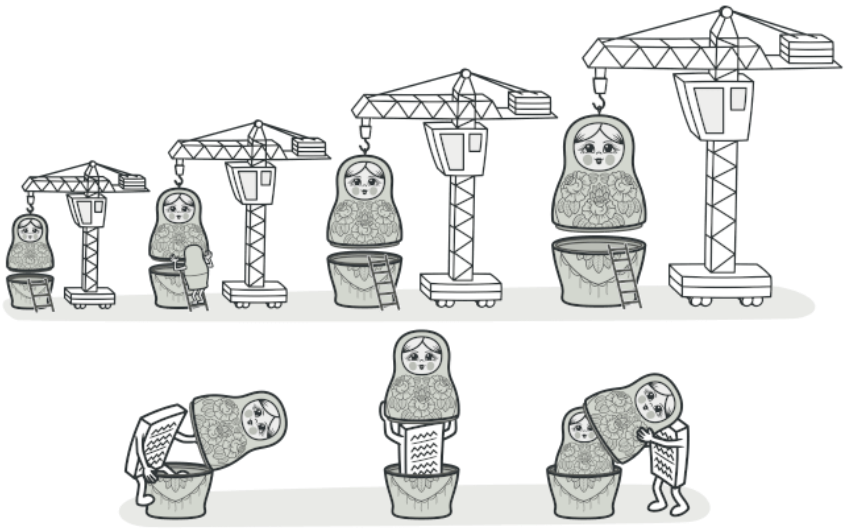


Рис. 3. Паттерн «Декоратор»

Концепция паттерна «Декоратор» заключается в использовании композиции вместо наследования. Паттерн позволяет создавать гибкие системы, в которых объекты могут быть обернуты в различные декораторы, добавляя им новые возможности.

Использование паттерна "Декоратор" обладает следующими преимуществами:

- гибкое добавление функциональности: позволяет добавлять новую функциональность объектам динамически без изменения их основной структуры, делая систему более гибкой;
- избегание создания множества подклассов: позволяет обертывать объект в различные декораторы, добавляя функциональность по мере необходимости, снижая количество классов и делая код более поддерживаемым;
- расширение функциональности без нарушения принципа открытости/закрытости: позволяет добавлять новую функциональность к объектам без изменения существующего кода, соблюдая принцип открытости/закрытости.
- возможность комбинирования различных декораторов: позволяет комбинировать различные декораторы между собой, создавая разнообразные комбинации функциональности и достигая большей гибкости в проектировании и использовании объектов.
- сохранение единообразия интерфейса: декораторы имеют тот же интерфейс, что и компоненты, что позволяет свободно заменять объекты декораторами, не нарушая работу клиента и сохраняя единообразие интерфейса.

Использование паттерна «Декоратор» способствует созданию гибкого и расширяемого кода, где функциональность объектов может быть динамически добавлена или изменена на лету. Это обеспечивает высокую степень адаптивности и позволяет легко вносить изменения в систему.

4. Пример использования паттерна «Декоратор»

Рассмотрим случай использования паттерна «Декоратор» на примере кофейного автомата. На рис. 4 изображена диаграмма классов кофейного аппарата.

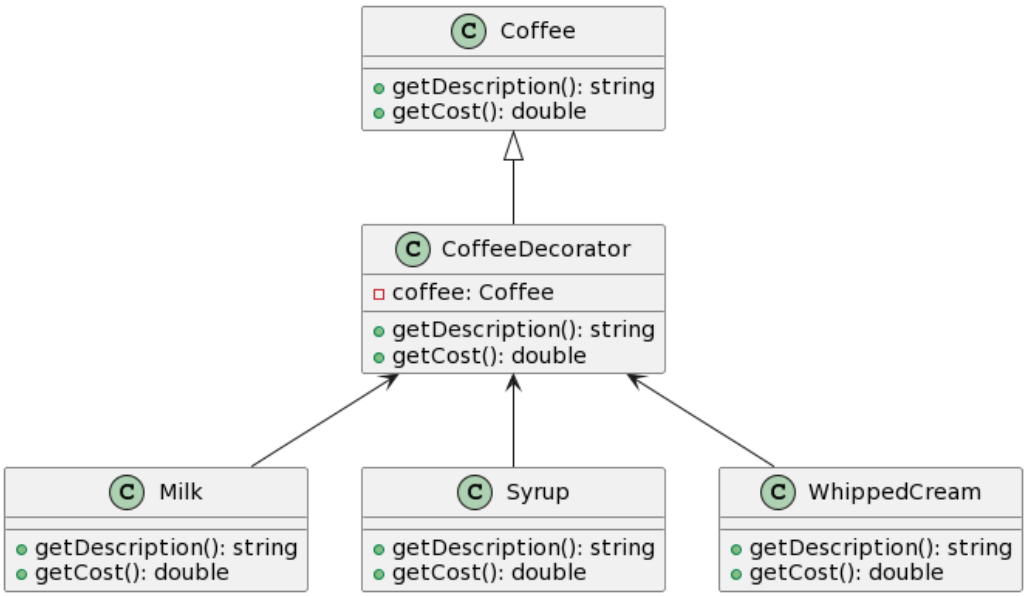


Рис. 4. Диаграмма классов кофейного аппарата

В этом примере класс Coffee представляет основной компонент, который имеет методы getDescription() и getCost(), которые возвращают описание кофе и его стоимость.

Класс CoffeeDecorator является абстрактным декоратором, который содержит ссылку на объект coffee, а также реализует тот же интерфейс, что и Coffee. Это позволяет обернуть кофе в другие декораторы и добавлять функциональность.

Декораторы Milk, Syrup и WhippedCream наследуются от CoffeeDecorator и реализуют дополнительную функциональность, добавляя соответствующие ингредиенты к описанию и стоимости кофе.

Применение паттерна «Декоратор» позволяет динамически добавлять различные ингредиенты к кофе, не меняя основной класс «Кофе».

5. Библиотека «MediatR»

Библиотека «MediatR» [3] является реализацией паттернов «Медиатор» и «Декоратор» в .NET. Основной смысл заключается в том, чтобы устранить прямую зависимость между отправителем запроса и обработчиком запроса. Вместо этого, MediatR работает через медиатор, который принимает запросы, находит соответствующего обработчика и передает запрос обработчику.

5.1. Реализация паттерна «Медиатор»

Для реализации паттерна «Медиатор» библиотека предоставляет следующие абстракции:

1. IMediator: Главный интерфейс MediatR, который служит как посредник для отправки запросов и событий. Интерфейс определяет методы для отправки запросов и событий.
2. IRequest<TResponse>: Интерфейс, который представляет запрос и определяет тип ожидаемого ответа. Реализация данного интерфейса требуется для каждого запроса в вашем приложении.
3. IRequestHandler<TRequest, TResponse>: Интерфейс, который представляет обработчик запроса. В контракте определен метод Handle, который принимает запрос TRequest и возвращает ожидаемый ответ TResponse.
4. INotification: Интерфейс, который представляет уведомление или событие в вашем приложении. Реализация данного интерфейса требуется для каждого события.
5. INotificationHandler<TNotification>: Интерфейс, который представляет обработчик события. В контракте определен метод Handle, который принимает событие TNotification и выполняет соответствующую обработку.

На рис. 5 изображен пример отправки запроса с помощью представленных абстракций. Код контроллера создает запрос, и с помощью посредника в виде IMediator отправляет запрос обработчику. Контроллеру не нужно напрямую знать про обработчик, его задача вызвать метод Send с нужным запросом или командой. Медиатор же в свою очередь определяет какой обработчик нужно вызвать для запроса.

На рис. 6 представлен пример отправки события с помощью метода Publish. Для одного события может быть зарегистрировано несколько INotificationHandler. В данном случае публикуется событие о заказе, медиатор передает это событие каждому из обработчиков. Следует знать, что существуют различные стратегии для вызова обработчиков события: по очереди или параллельно, с ожиданием результата и без.

```
// Использование MediatR для отправки запроса
public class UserController : ControllerBase
{
    private readonly IMediator _mediator;

    [HttpGet(template: "api/v1/get-user")]
    public async Task<IActionResult> GetUser(int userId)
    {
        var query = new Query(userId);
        var user = await _mediator.Send(query);
        return Ok(user);
    }
}

// Определение запроса
3 usages
public record Query(int UserId) : IRequest<User>;

// Обработчик запроса
public class Handler : IRequestHandler<Query, User>
{
    public Task<User> Handle(
        Query query,
        CancellationToken cancellationToken){...}
}
```

Рис. 5. Отправка запроса с помощью MediatR

```
public class EmailNotificationHandler
    : INotificationHandler<OrderNotification>
{
    public Task Handle(
        OrderNotification notification,
        CancellationToken cancellationToken){...}
}

public class TextMessageNotificationHandler
    : INotificationHandler<OrderNotification>
{
    public Task Handle(
        OrderNotification notification,
        CancellationToken cancellationToken){...}
}

5 usages
public record OrderNotification(int OrderId) : INotification;

public class OrderService
{
    public async Task CreateOrder(int orderId)
    {
        var notification = new OrderNotification(orderId);
        await _mediator.Publish(notification);
    }
```

Рис. 6. Отправка события с помощью MediatR

5.2. Реализация паттерна «Декоратор»

Для реализации паттерна «Декоратор» используется интерфейс IPipelineBehavior<TRequest, TResponse>, который предоставляет возможность реализации общей логики (поведения) для обработки запросов и уведомлений в пайплайне перед и после вызова обработчиков.

IPipelineBehavior позволяет добавлять общие операции, такие как валидация [4], логирование, обработка исключений, аудит и другие, к цепочке обработки запросов и уведомлений.

На рис. 7 представлен пример использования декоратора для управления транзакциями. Перед тем как вызвать следующий делегат открывается транзакция, после обработки всех последующих делегатов транзакция фиксируется. В случае исключения транзакция откатывается. Стоит отметить, что IPipelineBehavior можно использовать только для обработчиков запросов, а не событий.

```
public class TransactionPipelineBehaviour<TRequest, TResponse>
    : IPipelineBehavior<TRequest, TResponse>
    where TRequest : ITransactionWrappedCommand<TResponse>
{
    3 references
    private readonly DbContext _db;

    0 references
    public async Task<TResponse> Handle(
        TRequest request,
        CancellationToken cancellationToken,
        RequestHandlerDelegate<TResponse> next)
    {
        TResponse response;
        await _db.BeginTransactionAsync(cancellationToken);

        try
        {
            response = await next();
        }
        catch (Exception)
        {
            await _db.RollbackTransactionAsync(cancellationToken);
            throw;
        }

        await _db.CommitTransactionAsync(cancellationToken);
        return response;
    }
}
```

Рис. 7. Декоратор для управления транзакциями

Заключение

Паттерны «Медиатор» и «Декоратор» предоставляют эффективные инструменты для организации и управления сложной логикой в приложениях на C#.

Паттерн «Медиатор» обеспечивает централизованную точку связи между объектами, позволяя им взаимодействовать друг с другом без явных ссылок. Он способствует уменьшению связанности и повышению гибкости приложений, позволяя изменять коммуникацию между объектами без изменения самих объектов.

В библиотеке MediatR паттерн «Медиатор» реализуется с помощью класса Mediator, который обеспечивает механизм публикации запросов (Send) и уведомлений (Publish) и их дальнейшую маршрутизацию к соответствующим обработчикам.

Паттерн «Декоратор» позволяет динамически добавлять функциональность к объектам, не изменяя их исходного поведения. В библиотеке MediatR паттерн «Декоратор» может быть реализован с использованием декораторов для обработчиков запросов и обработчиков уведомлений. Это позволяет вынести общую логику (например, логирование или валидацию) в декораторы.

Использование паттернов «Медиатор» и «Декоратор» может значительно упростить архитектуру, улучшить разделение ответственности и облегчить поддержку и расширение кода. Они помогают создать гибкий и расширяемый код, который легко модифицировать и адаптировать к новым требованиям бизнеса.

Библиотека MediatR предоставляет удобные механизмы для реализации паттернов «Медиатор» и «Декоратор» в .NET-приложениях, позволяя быстро разрабатывать продукты с хорошей архитектурой и высокой гибкостью.

Литература

1. Mediator Pattern – URL: https://refactoring.guru/ru/design-patterns/mediator (дата обращения 26.03.2024).

2. Decorator Pattern – URL: https://refactoring.guru/ru/design-patterns/decorator (дата обращения 26.03.2024).
3. MediatR – URL: https://github.com/jbogard/MediatR (дата обращения 26.03.2024)
4. Вам нужен медиатор – URL: https://habr.com/ru/articles/734302/ (дата обращения 09.04.2024)

УДК 519.25

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СРЕСТВ ЗАЩИТЫ ПЕРСОНАЛЬНЫХ ДАННЫХ ИНТЕРНЕТ-ПОЛЬЗОВАТЕЛЯ

М. Ю. Талагаев

Воронежский государственный технический университет

Введение

За последние десятилетия Интернет превратился в неотъемлемую часть нашей повседневной жизни, став ключевым инструментом взаимодействия, информационного обмена и коммерческих операций. Однако с ростом цифровой активности возросла и потребность в защите конфиденциальности пользователей. С каждым годом случаи утечек данных, хакерских атак и нарушений конфиденциальности становятся все более распространенными и разрушительными.

1. Постановка проблемы

Несмотря на меры предосторожности и технические инновации, злоумышленники постоянно находят новые способы проникновения в защищенные системы. Так, например, среди наиболее громких утечек данных пользователей можно выделить утечку в компании Equifax [1]. В 2017 году компания Equifax допустила утечку личной и финансовой информации почти 150 млн человек в результате обновления одной из своих баз данных. Компания не смогла устранить критическую уязвимость спустя несколько месяцев после выпуска патча и не информировала общественность о взломе в течение нескольких недель после его обнаружения. В июле 2019 года кредитное агентство согласилось выплатить сумму $575 млн в рамках урегулирования конфликта. Еще одна подобная ситуация произошла с компании Facebook в 2021 году, когда около 540 миллионов пользователей были подвергнуты риску из-за недостатков в системе безопасности [2]. Facebook понесли убытки, связанные с судебными тяжбами в размере $5 млрд.

Среди российских компаний можно выделить ряд случаев нарушения конфиденциальности данных [3], таких как: утечку данных с сервера ОФД «Дримкас», когда в общей сложности более 90 млн записей с данными различного рода о юридических и физических лицах оказались в Сети. При этом самой крупной утечкой считается кража данных клиентов Сбербанка, 60 млн записей. База, содержащая информацию о десятках миллионов держателей кредитных карт, оказалась в продаже. А также, утечка базы данных клиентов «Билайн» в 2019 году.

Кибербезопасность остается одним из самых сложных и важных направлений в деятельности организаций и государственных структур, а также источником беспокойства

обычных интернет-пользователей [4]. Так, согласно исследованию, Тинькофф в 2022 году эксперты в сфере кибербезопасности зафиксировали 710 случаев умышленной утечки информации что в два раза больше, чем в 2021 году. Общее количество утекших данных равно – 667,6 млн записей. Чаще всего злоумышленники получают доступ к персональным данным – 88.0%, коммерческой тайне компаний – 9.1%, государственной тайне – 1.5% и платежной информации – 0.5%.

Наравне с кражей данных крупных компаний, хакеры стремятся получить данные и у рядовых пользователей при помощи фишинг-техник, таких как ложные электронные письма, веб-сайты или сообщения. Кроме того, хакеры используют вредоносные программы, такие как вирусы, черви или троянские программы, для заражения компьютеров и мобильных устройств пользователей. После заражения такие программы могут собирать и передавать персональные данные без ведома и согласия пользователя третьим лицам, что может привести к непринятым. И если с утечкой данных крупных компаний интернет-пользователь мало что может сделать, то защитой от атак второго типа он может и должен озаботиться сам.

Существует разнообразный арсенал средств защиты данных. К наиболее популярным можно отнести использование паролей длинной и сложной структуры, регулярное резервное копирование, установку паролей на рабочие папки и файлы, двухфакторную аутентификацию и защиту с помощью антивирусных программ. Однако, не все эти средства защиты известны широкому кругу пользователей, и даже если известны, многие люди их не применяют.

2. Результаты исследования

Нами было проведено исследование, целью которого стало определить, какие средства защиты используют обычные интернет-пользователи в повседневной жизни и какие из них считают наиболее эффективными. В исследовании приняли участие 33 человека в возрасте от 19 до 61 года. Им предлагалось ответить на вопросы составленной нами анкеты, которые позволяли выяснить, какие меры по обеспечению кибербезопасности они знают и применяют.

Анализ результатов исследования показал, что наиболее часто используемые средства защиты следующие: использование антивирусной программы – 66.7%, использование сложных паролей – 60.6%, регулярное резервное копирование – 36.4% и двухфакторная аутентификация – 36.4%. На вопрос о том используют ли респонденты платный антивирус 78.8% ответили, что не используют, при этом 65.4% полагают, что бесплатные версии не менее эффективны. Почти треть опрошенных (26.9% считают платную версию слишком дорогой) и 7.7% прекрасно обходятся без антивирусной программы. На вопрос о том, как часто пользователи проводят резервное копирование 48.5% ответили, что никогда этого не делают, 33.3% выполняют его ежемесячно, 9.1% – еженедельно и 12.1% – ежедневно. При этом, наиболее часто, резервные копии сохраняют в облачных средствах – 46.7% и на самом устройстве 36.7%. На отдельном устройстве резервные копии сохраняют только 16.9%.

Что касается паролей, которые сегодня необходимы для входа в практически любой сервис, то 30.3% опрошенных используют один и тот же пароль на всех сайтах и сервисах, в то время как 69.7% предпочитают устанавливать разные. Вместе с этим, подавляющие количество задают в качестве пароля простые сочетания цифр или значимые для себя слова – 69.7%, и только 30.3% используют случайные наборы букв и цифр. Также две трети респондентов отметили, что не рассказывают свои пароли родными и близким – 66.7%, в то время как 33.3% доверяют им эту информацию.

На вопрос о том, где пользователи хранят свои пароли 57.6% ответили, что хранят их «в голове», 21.2% используют сервисы-помощники от Google или Yandex, 18.2% предпочитают хранить пароли в блокноте, 12.1% сохраняют пароли в отдельный файл на компьютере и всего 3% записывают на листок бумаги и хранят рядом с компьютером. Среди респондентов всего

21.2% используют отдельную учетную запись при работе против 78.8%, которые подобным не утруждаются.

На вопрос о том, какие средства защиты информации они считают обязательными, пользователи отвечали следующее – 53.1% написали антивирус, 18.9% считают обязательным сложный пароль и по 6.6% респондентов соответственно ответили о необходимости иметь внешний диск или двойную аутентификацию.

Таким образом, наиболее часто используемыми средствами защиты, согласно полученным данным исследования, оказались антивирусные программы и сложные пароли. Интересным фактом является отношение респондентов и к выбору антивирусного программного обеспечения. Большинство не использующих платные версии антивирусов делают это по причине убежденности в эффективности бесплатных версий или из-за считающегося высоким уровня цены на платные версии. Кроме того, данные исследования указывают на недостаточное осознание рисков, связанных с несоблюдением базовых правил создания паролей, а также недостаточное внимание к уровню безопасности хранения паролей.

При выборе антивирусной программы пользователь может лишь выбрать одну из представленных на рынке без знания механизмов работы, тогда как с создание правильного пароля лежит на самом человеке**.** Для определения достаточно ли сложен и безопасен пароль требуется погрузится в работу сервисов и изучить принципы хранения конфиденциальных данных пользователей в базах данных. Когда пользователь создаёт аккаунт, сервис сохраняет пароль в одном из многочисленных форматов. Сервис может занести пароль прямо в базу данных (в виде простого текста) или сгенерировать из него хеш с помощью одного из многочисленных алгоритмов. Самые популярные алгоритмы для хеширования пароля это: MD5, SHA-1, Argon2, Bcrypt и Scrypt. Главным преимуществом такого подхода является отсутствие паролей в базе данных, поскольку вместо них хранится хеш. Таким образом, когда пользователь входит пароль на сайте серверу отправляется хеш, который сравнивается с имеющимся в базе и в случае совпадения пропускает пользователя дальше (рис.1).

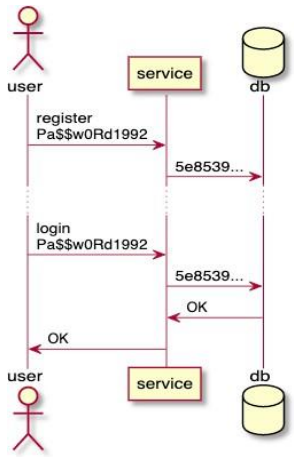


Рис. 1. Схема проверки пароля сервисом

Помимо скорости проверки пароля такой подход осуществляет и максимальную безопасность, поскольку если база данных будет взломана, злоумышленники не получат конкретных данных, а подбор пароля по хешу будет невозможен. Однако, многие компании чаще всего используют для хеширования паролей наиболее быстрые алгоритмы такие как MD5 и SHA-1 или не используют их вовсе, поскольку, чем сложнее алгоритм, тем большие

вычислительные мощности требуются.

Хоть компании иногда и пренебрегают безопасностью хранимых данных, пользователь может обезопасить себя, используя пароль большой длины. Сила пароля характеризуется его энтропией, то есть количество случайностей которая хранится в пароле [5-7]. Для вычисление точного значения используется формула:

$$E = \log_2(S^l), \quad (1)$$

где S – размер пула уникальных символов, а l – длина пароля.

Таким образом, чем больше энтропия пароля, тем сложнее его подобрать, зная его хеш. Например, вычислим энтропию пароля формата "слово123" учитывая возможность появления каждого символа. Предположим, что в пароле используются символы русского алфавита (33 символа), а также цифры (10 символов). Тогда $S = 33+10 = 43$, l же будет равно 8. Подставив значения в формулу, получаем $E = \log_2(43^8) = 43.41$ бит, для того чтобы высчитать удельную энтропию потребуется разделить 43.41 бита на длину пароля l, таким образом $\frac{43.41}{8} = 5.42$ удельная энтропия.

Ниже представлена таблица различных классов, количество возможных переменных и энтропии для каждого класса русского алфавита (табл.1).

Таблица 1

Энтропия для различных типовых данных с учетом русского алфавита

Класс символов	**Пример**	**Количество переменных**	**Энтропия**
Только нижний регистр	дуб	33	5.04
Верхний и нижний регистр	дУб	66	6.04
Верхний, нижний регистр и цифры	дУб1	76	6.24
Верхний, нижний регистры, цифры и специальные символы	дУб1?	98	6,61

Таким образом, пароль, состоящий из 10 символов прописных и строчных букв, будет иметь энтропию 60.4 бита. Поскольку на большинстве сервисов для пароля используются буквы латинского алфавита, приведем таблицу и для него (табл. 2):

Таблица 2

Энтропия для различных типовых данных с учетом латинского алфавита

Класс символов	**Пример**	**Количество переменных**	**Энтропия**
Только нижний регистр	abc	26	4.7
Верхний и нижний регистр	aBc	52	5.7
Верхний, нижний регистр и цифры	aBc1	62	5.95
Верхний, нижний регистры, цифры и специальные символы	aBc1?	84	6.4

Согласно исследованиям Национального института стандартов и технологий, безопасным паролем в период с 2019 год по 2030 год считается последовательность, имеющая энтропию 128 и более.

Таким образом безопасный пароль будет выглядеть примерно так:

1. qxcrasimtpaocstbzibmsnxdkbq при использовании только строчных букв (длина 27).

2. jbVaxFFuOIkwnBQBAexmlG при использовании прописных и строчных букв (длина 22).

3. IhBSAtorlHrnDnU7Y7oae при использовании прописных, строчных букв и цифры (длина 21).

4. tkWwhFQXDMLokioPefLG при использовании прописных, строчных букв, цифр и специальных символом (длина 20).

Заключение

В современном информационном пространстве, где безопасность данных является одним из приоритетных аспектов, вопрос обеспечения надежности паролей и их хранения находится в центре внимания. Для обеспечения наиболее надежной защиты данных пользователю необходимо не только использовать антивирусные программы, но и уделить внимание выбору пароля. Эффективное применение алгоритмов хеширования в сочетании с использованием паролей высокой энтропии становится ключевым фактором в обеспечении безопасности информационных систем и защите персональных данных.

Литература

1. Утечка данных Equifax в 2017 году – Режим доступа: https://en.wikipedia.org/wiki/2017_Equifax_data_breach

2. Утечка Facebook в 2021 году – Режим доступа: https://en.wikipedia.org/wiki/2021_Facebook_leak

3. Топ 10 крупнейших утечек информации в России – Режим доступа: https://www.anti-malware.ru/analytics/Threats_Analysis/Top-10-data-leakage-in-Russia

4. Белоус, А. И. Кибероружие и кибербезопасность. О сложных вещах простыми словами / А. И. Белоус, В. А. Солодуха. – Москва, Вологда: Инфра-Инженерия, 2020. – 692 с.

5. Password Entropy: The Value of Unpredictable Passwords – Режим доступа: https://www.okta.com/identity-101/password-entropy

6. Password entropy – Режим доступа: https://www.techtarget.com/whatis/definition/password-entropy

7. All About Password Entropy – Режим доступа: https://specopssoft.com/blog/pass-word-entropy

УДК 004.415.2.031.43

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА И УПРАВЛЕНИЯ ПРОМЫШЛЕННЫМ ОБОРУДОВАНИЕМ

С. В. Тимофеенко, К. Г. Резников

Воронежский государственный университет

Введение

Разработка приложений для потребностей современной промышленности является сложной и комплексной задачей. В ряде задач возникает потребность ручного управления системой специалистом, а не полностью автоматизированный алгоритм. Для реализации такой системы необходимо учитывать масштаб предприятия, количество оборудования, а также возможность дистанционного управления через веб-приложение [1].

Цель данной работы — разработать веб-приложение для гибридного управления промышленным оборудованием, которое сочетает в себе автоматический контроль и ручное управление в реальном времени. Приложение должно предоставлять пользователям возможность не только наблюдать за текущим состоянием машин и производственных процессов через удобный интерфейс, но и быстро вмешиваться в управление при необходимости.

1. Программное обеспечение для промышленного оборудования

Программное обеспечение представляет собой систему для контроля и сбора данных во время работы с оборудованием, адаптированную для работы через веб-интерфейс. Система должна позиционировать себя как решение для управления процессами и мониторинга в реальном времени, обеспечивая пользователю возможность получения актуальной информации о параметрах работы устройств и процессов. Важными характеристиками являются оперативность реагирования на изменения состояний и возможность предотвращения потенциальных нештатных ситуаций путем отправки команд управления.

Также система должна учитывать требований масштабируемости и адаптивности, позволяя интегрировать дополнительное оборудование и расширять функциональные возможности с минимальными усилиями и затратами. Безопасность передачи данных и управляющих команд обеспечивается современными методами шифрования и аутентификации, предотвращая несанкционированный доступ и обеспечивая целостность данных.

Главной идеей программного обеспечения это ориентированность на потребности малых предприятий, ключевыми критериями для которых являются доступность и простота использования. В этом контексте предлагается сравнение с альтернативными решениями от крупных производителей, таких как система WinCC от Siemens и платформа FactoryTalk от Rockwell Automation. Эти решения характеризуются мощными функциональными возможностями и предназначены для крупномасштабных производственных процессов, предлагая глубокую интеграцию с различным оборудованием.

Основным недостатком таких систем является их высокая стоимость, делающая их недоступными для малых предприятий, а также сложность внедрения и требования к

квалификации персонала для поддержки и обслуживания. Кроме того, некоторые производители предлагают станки с встроенными системами управления, что обеспечивает высокую степень интеграции, но сопряжено с высокими затратами, что делает такие решения экономически нецелесообразными для малого бизнеса.

В отличие от упомянутых решений, программное обеспечение должно представлять собой гибкое и экономически выгодное решение, позволяющее малым предприятиям автоматизировать свои процессы без значительных капитальных вложений в специализированное оборудование и ПО. Оно характеризуется низкой стоимостью владения, простотой настройки и возможностью адаптации под конкретные потребности предприятия, что делает его идеальным выбором для бизнесов с ограниченным бюджетом.

2. Требования к ПО для управления и мониторинга в реальном времени

Проект нацелен на создание уникальной системы управления для малых предприятий, задействующих небольшое оборудование без встроенных интерактивных способов управления. Основная цель – предоставить инструмент, который не только облегчает работу в реальном времени через веб-интерфейс, но и отличается простотой интеграции и использования.

Одной из главных функций является мониторинг в реальном времени, который позволяет предоставлять актуальную информацию о состоянии оборудования, включая параметры такие как температура, скорость работы и давление. Эта возможность критически важна для оперативного реагирования на любые изменения и поддержания оптимальных условий работы.

Удалённое управление дополняет функционал мониторинга, давая возможность отправлять управляющие команды оборудованию прямо с любого устройства, подключенного к интернету. Это значительно увеличивает гибкость управления процессами, позволяя операторам вмешиваться в работу оборудования на расстоянии, будь то запуск, остановка или изменение рабочих параметров.

Планирование задач и автоматизация рутинных процессов становится простым и интуитивно понятным благодаря возможности создания расписаний и автоматической настройке параметров работы в соответствии с планом производства. Эта функция предоставляет предприятиям инструменты для повышения эффективности и минимизации затрат времени на управление процессами.

Анализ данных и отчётность играют ключевую роль в оптимизации производственных процессов, позволяя собирать и анализировать данные о работе оборудования. Выявление тенденций и предоставление отчётов о производительности, потреблении ресурсов и возможных проблемах помогает в принятии обоснованных решений для улучшения процессов.

Функция предупреждений и уведомлений является неотъемлемой частью системы, обеспечивая своевременное информирование ответственных лиц о любых нештатных ситуациях или отклонениях от заданных параметров работы. Это гарантирует возможность быстрого реагирования на возможные проблемы и способствует предотвращению серьёзных сбоев в работе.

Отличие от более дорогих и сложных систем, предлагаемых крупными производителями, заключается в простоте и экономической доступности. Проект нацелен на обеспечение малых предприятий эффективным, но вместе с тем простым в освоении решением, которое не требует крупных инвестиций или специализированных знаний для начала работы.

3. Инструменты и технологии для разработки

Для разработки проекта выбраны современные технологии, обеспечивающие как высокую производительность веб-приложения, так и удобство в его разработке и поддержке. Язык программирования Python [2] в сочетании с фреймворком Flask [3] используется для создания серверной части приложения, что позволяет разрабатывать масштабируемую и гибкую серверную логику. Flask отличается легковесностью и гибкостью, предоставляя разработчикам свободу в выборе инструментов и расширений для реализации необходимого функционала.

Среди ключевых расширений Flask, Flask-SQLAlchemy играет важную роль, облегчая взаимодействие с базой данных PostgreSQL. Это обеспечивает эффективное управление данными, гарантируя их надежность и доступность. Flask-RESTful, в свою очередь, используется для создания REST API, что делает обмен данными между сервером и клиентом еще более гибким и удобным.

Разработка пользовательского интерфейса основана на использовании React.js [4] и D3.js, что позволяет создавать высокоинтерактивные интерфейсы и эффективно визуализировать данные. Эти технологии идеально подходят для построения интуитивно понятных приложений, способствуя быстрой ориентации пользователя в управлении процессами. В будущем, планируется расширение функционала интерфейса за счёт внедрения дополнительных фреймворков, например Redux, что улучшит управление состоянием приложения и его производительность.

Для организации взаимодействия в реальном времени между сервером и клиентами в проекте используется Flask-SocketIO. Это расширение позволяет реализовать веб-сокеты для мгновенной передачи данных, что является ключевым для функций мониторинга и получения уведомлений о событиях в работе оборудования. Использование Node-RED облегчает интеграцию с различными каналами уведомлений, как E-Mail или SMS. Эта функция критически важна для обеспечения оперативного реагирования на изменения состояния оборудования. Также рассматривается включение MQTT как дополнительного канала для усиления системы уведомлений, обеспечивая её надёжность даже при нестабильных интернет-соединениях.

Выбор PostgreSQL в качестве системы управления базами данных подчеркивает стремление к надежности и эффективности работы с большими объемами данных [5]. PostgreSQL предлагает расширенный набор функциональностей для комплексной работы с данными, что идеально подходит для задач, стоящих перед веб-приложением

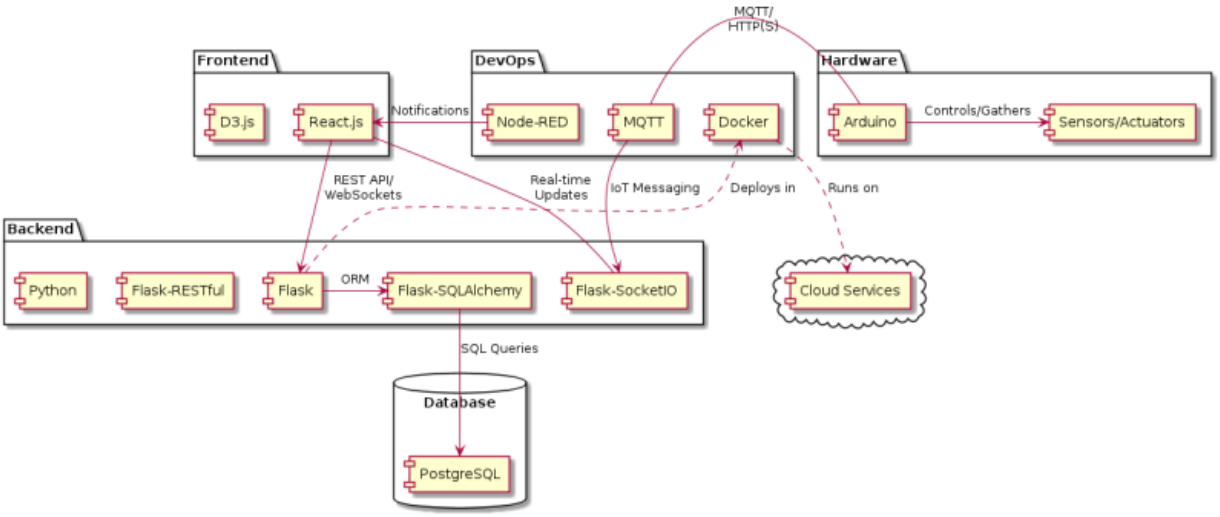


Рис. 1. UML-диаграмма архитектуры системы

Интеграция с микроконтроллерами, в частности с Arduino, осуществляется через стандартные библиотеки и HTTP/HTTPS протоколы, что обеспечивает эффективный сбор данных и обмен информацией с сервером. В процессе разработки уделено особое внимание созданию прошивки для Arduino, оптимизирующей потребление энергии и повышающей надёжность работы устройств. Рассматривается возможность расширения функционала за счёт технологий дальней связи, таких как LoRaWAN, и использование MQTT для оптимизации сетевого взаимодействия, что позволит значительно увеличить дальность и качество передачи данных от датчиков [6].

Таким образом, выбранные технологии обеспечивают прочную основу для создания адаптивной и надёжной системы управления, однако постоянный анализ и стремление к улучшению позволят сделать систему ещё более устойчивой.

Представленная на рисунке 1 UML-диаграмма архитектуры системы управления демонстрирует продуманный выбор технологий и структуры.

4. Принцип работы и реализация

На рисунке 2 представлена диаграмма последовательности, которая иллюстрирует полный процесс взаимодействия пользователя с системой управления промышленным оборудованием, начиная с момента регистрации в системе и заканчивая выполнением команд управления. Весь процесс разбивается на несколько ключевых этапов, включающих регистрацию, аутентификацию, запрос текущего состояния оборудования, отправку команд на его запуск и получение подтверждения о выполнении данных команд.

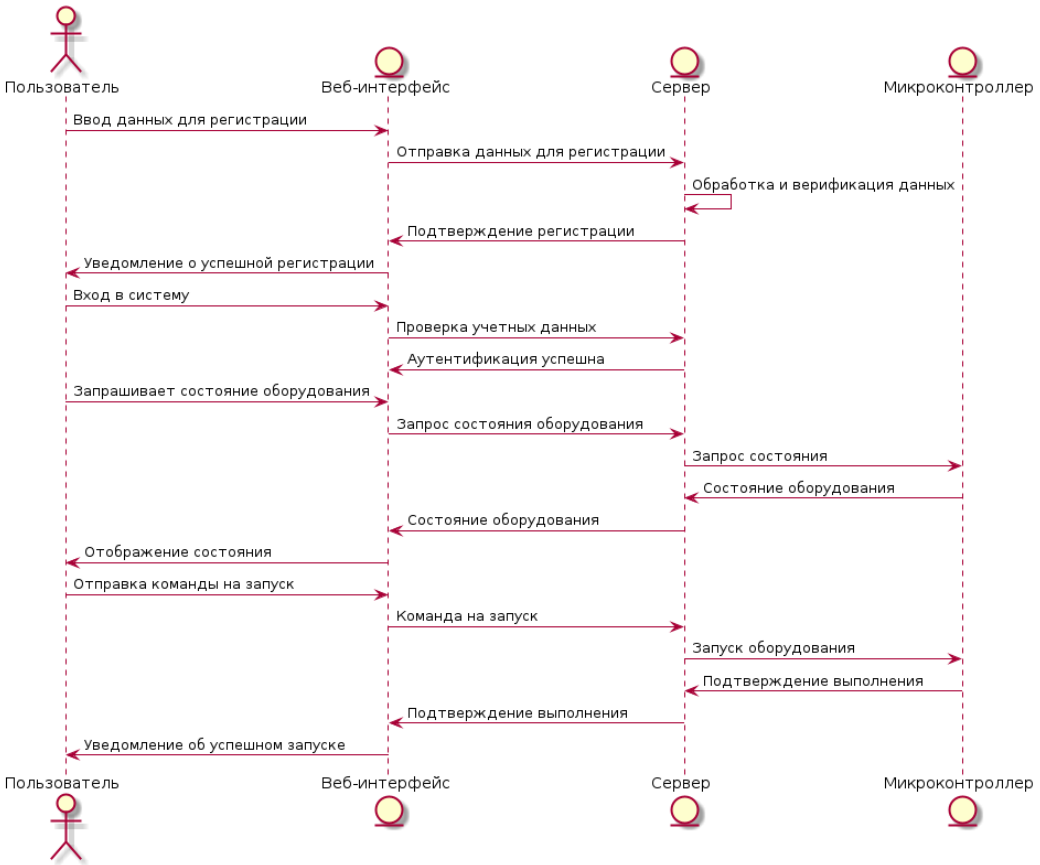


Рис. 2. UML-диаграмма последовательности

Процесс начинается с того, что новый пользователь вводит необходимые данные для регистрации через веб-интерфейс, который, в свою очередь, передает эти данные на сервер. Сервер обрабатывает полученные данные, создает новую учетную запись и отправляет обратно подтверждение о успешной регистрации. Этот шаг критически важен для идентификации пользователей и обеспечения безопасности системы, так как позволяет контролировать доступ к функциям управления оборудованием и обеспечивать персонализированный опыт использования.

После успешной регистрации пользователь может войти в систему, используя свои учетные данные. Система аутентифицирует пользователя и предоставляет доступ к интерфейсу управления. Пользователь может запросить текущее состояние оборудования, и система, взаимодействуя через сервер с микроконтроллером, предоставляет актуальные данные. Эта возможность наблюдения за состоянием оборудования в реальном времени является ключевой для оперативного реагирования на изменения и поддержания оптимальных условий его работы.

Реализована функция регистрации нового устройства в системе. Пользователь может зарегистрировать новые устройства, вводя информацию в необходимые поля: название устройства, тип (в данном случае предварительно выбран Arduino Uno), серийный номер, данные для подключения к Wi-Fi сети, включая название сети и пароль. После заполнения формы пользователь нажимает кнопку "Зарегистрировать" для завершения процесса добавления устройства в систему. Это позволяет системе идентифицировать и управлять новым оборудованием.

В качестве демонстрационного примера добавлены два устройства: "Arduino Uno Фрезерный станок" и "Raspberry PI Лазерный станок" [7]. Они подключены к системе через Wi-Fi, что предполагает возможность беспроводного обмена данными между устройствами и сервером. Это удобно, так как позволяет размещать устройства в разных точках без необходимости проводного подключения.

Система отображает текущее состояние каждого устройства по нескольким показателям: потребление энергии, температура, уровень давления. Такие данные могут быть получены с помощью датчиков, подключенных к устройствам, и отправляются на сервер, где они визуализируются в удобном для пользователя виде. Это позволяет оператору в реальном времени отслеживать критические параметры и эффективность работы оборудования. На рисунке 3 отражены добавленные устройства и статистические данные работы одного из них.

Для демонстрации системы оповещения виртуализирована критическая ситуация. В системе мониторинга активировалось предупреждение: *«Внимание: температура превысила допустимый порог!»*. Это сообщение высвечивается на экране в виде всплывающего окна и является частью системы управления, направленной на предотвращение возможных нештатных ситуаций.

Также добавлена функция планирования задач. Здесь пользователь может задать расписание для определенных действий оборудования. В предложенном поле можно выбрать одно из зарегистрированных устройств. Для него предусмотрено планирование действий, например, *«Старт»,* и установка конкретного времени выполнения через встроенный календарь и часы. После нажатия кнопки *«Запланировать»,* задание вносится в расписание и будет выполнено системой автоматически в указанное время

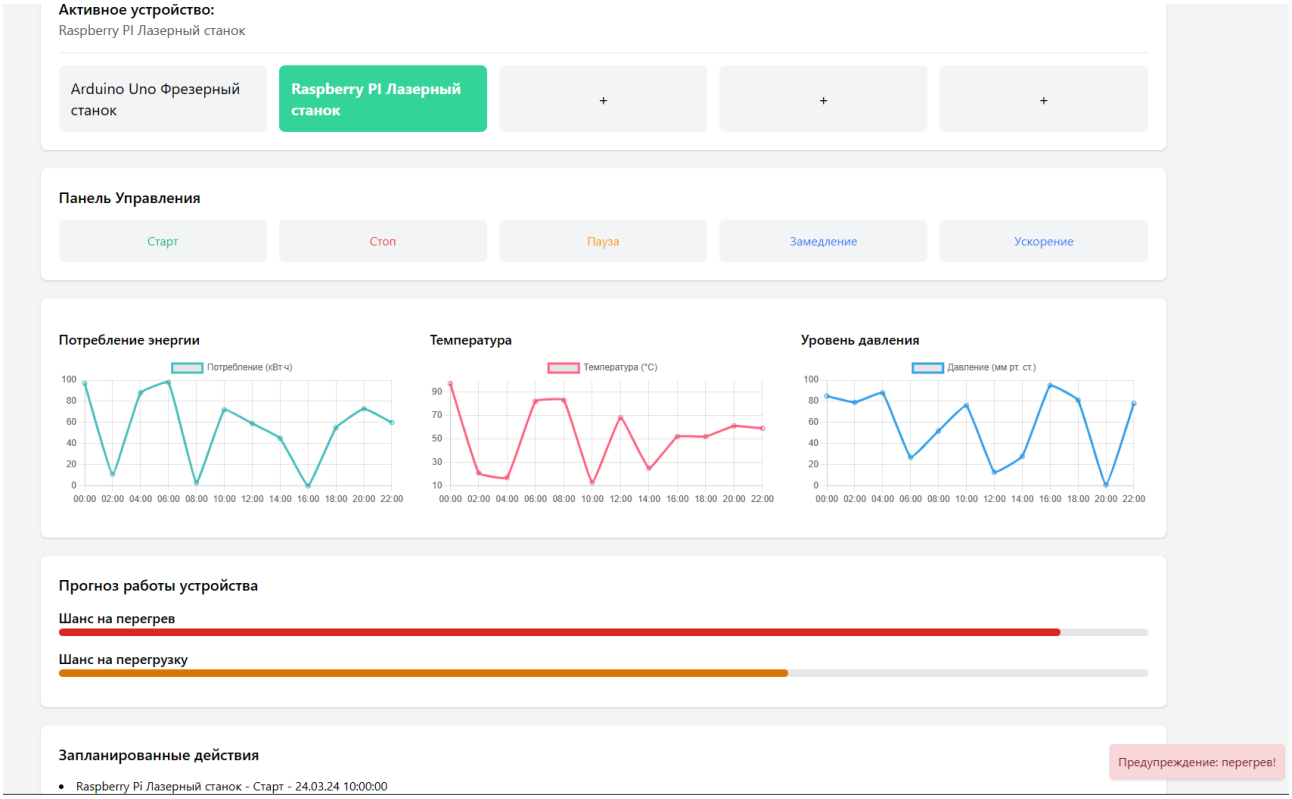


Рис. 3. Интерфейс веб-приложения

Рисунок 3 демонстрирует главный интерфейс пользователя, где в нижней части экрана отображается панель *«Запланированные действия»*, в которую добавлена описанная выше задача старта лазерного станка в заданное время. Эта система планирования позволяет автоматизировать процессы управления оборудованием, повышая эффективность и уменьшая вероятность человеческой ошибки.

Заключение

В рамках статьи рассмотрена разработка и реализация веб-приложения для мониторинга и управления промышленным оборудованием в реальном времени, основываясь на современных подходах и технологиях. Актуализирована важность оперативного контроля и гибкости управления в производственных процессах. Также стоит отметить, что выбор данных технологий позволяет не только решить поставленные задачи, но и предоставляет фундамент для будущего масштабирования и интеграции с другими системами. Эта гибкость и открытость к новым возможностям делает данный подход оптимальным выбором для малых предприятий, стремящихся к модернизации своих производственных процессов и повышению их эффективности.

Список литературы

1. Резников К. Г. Разработка программного обеспечения для визуализации трехмерных поверхностей в веб-браузере / К. Г. Резников, С. Н. Медведев // Вестник ВГТУ.

Том 17, №6. – Воронеж: ВГТУ, 2021 - С. 13-19.
2. Гринберг М. Flask Web Development: Создание веб-приложений на языке Python / М. Гринберг. – СПб. : Питер, 2021. – 350 с.
3. Официальная документация Flask. – Режим доступа: https://flask.palletsprojects.com/en/2.0.x/. – (Дата обращения: 14.03.2024).
4. Официальная документация React. – Режим доступа: https://reactjs.org/docs/getting-started.html. – (Дата обращения: 14.03.2024).
5. Грофф, Д. Р. SQL: полное руководство / Д. Р. Грофф, П. Н. Вайнберг, Э. Д. Оппелъ. – 3-е изд. – М. : ООО "И.Д. Вильяме", 2015. – 960 с.
6. Лукас, П. MQTT в действии: Сообщения и уведомления для Интернета вещей / П. Лукас. – СПб. : Питер, 2023. – 256 с.
7. Петин В. А. Arduino и Raspberry Pi в проектах Интернета вещей. – 2-е изд. – М.: БХВ-Петербург, 2017 – 432.

УДК 004.415.2.031.43

РАЗРАБОТКА МОДУЛЯ ARIMA ДЛЯ МОНИТОРИНГА СОСТОЯНИЯ ПРОМЫШЛЕННОГО ОБОРУДОВАНИЯ

С. В. Тимофеенко, Е. В. Трофименко

Воронежский государственный университет

Введение

В современной промышленности актуальность разработки и внедрения алгоритмов для мониторинга и прогнозирования состояния оборудования неуклонно растет. Эти алгоритмы позволяют не только предотвратить потенциальные сбои и аварии, но и значительно повысить эффективность использования машин и оборудования. Внедрение передовых технологий и инновационных решений в этой области открывает новые возможности для улучшения надежности, безопасности и продолжительности эксплуатации промышленных активов.

В данной статье рассмотрены популярные алгоритмы применяемые для мониторинга оборудования, описана методика ARIMA и приведены результаты диагностики работы станка, проведенные с помощью разработанного модуля, который реализовал метод ARIMA.

5. Обзор алгоритмов

Существует множество алгоритмов, разработанных для мониторинга оборудования, каждый из которых имеет свои особенности, преимущества и сферы применения. Наиболее популярными являются методы по машинному обучению, сети с долгой краткосрочной памятью [1], преобразования Фурье [2], а также ARIMA[3].

ARIMA (Авторегрессионные интегрированные модели скользящего среднего) — это широко используемый статистический метод для анализа и прогнозирования временных рядов [4]. Метод основан на идее, что будущие значения серии можно предсказать на основе её прошлых значений и ошибок прогноза. ARIMA эффективно применяется для моделирования и прогнозирования временных рядов в случаях, когда данные показывают тенденции и сезонность, что делает его особенно полезным для предсказания технического состояния оборудования на основе исторических данных.

Решающие деревья — это метод машинного обучения, который использует структуру дерева решений для классификации или регрессии. Каждый узел дерева представляет собой точку принятия решения по одному из атрибутов, а ветви дерева — возможные значения этих атрибутов, ведущие к листьям с прогнозируемыми значениями или классами. Этот метод широко применяется для диагностики состояния оборудования, поскольку позволяет легко интерпретировать, какие факторы влияют на вероятность отказа.

Сети с долгой краткосрочной памятью (LSTM) — это вид рекуррентных нейронных сетей, специально разработанный для анализа временных последовательностей и данных с долгосрочными зависимостями. LSTM способны запоминать информацию на продолжительные периоды, что делает их идеальными для задач, связанных с прогнозированием состояния оборудования, где важно учитывать, как недавние, так и более старые данные для точного прогноза.

Преобразование Фурье — это математический инструмент, который преобразует сигнал из временной области в частотную, позволяя анализировать частотный состав сигналов. В контексте мониторинга оборудования это преобразование используется для анализа вибрационных сигналов, позволяя обнаруживать аномалии и предсказывать отказы на основе изменений в частотных характеристиках. На рисунке 1 представлены диаграммы, отображающие характеристики каждого из методов.

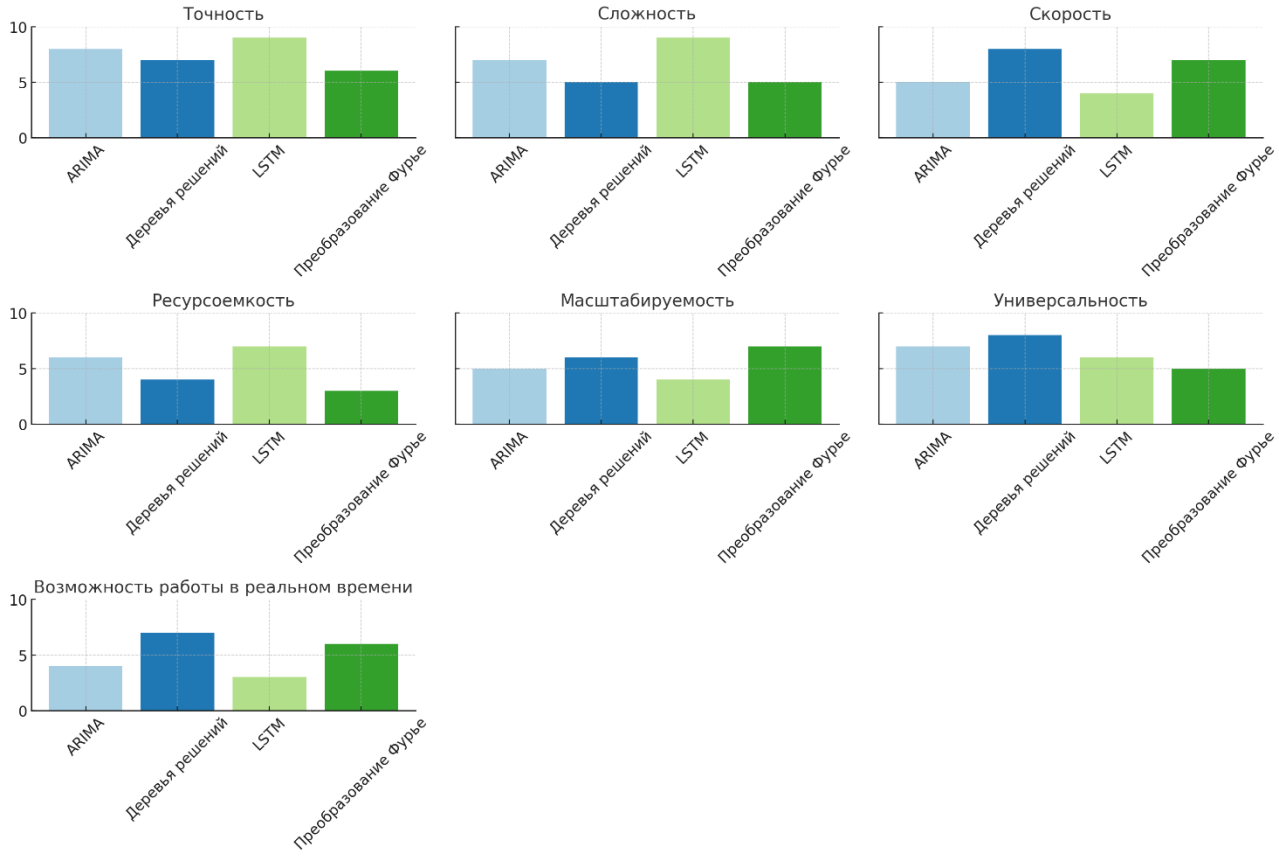


Рис. 1. Диаграммы характеристик

Каждый из рассмотренных алгоритмов обладает своим набором достоинств и недостатков, но именно использование ARIMA для моделирования основной тенденции временного ряда предсказывать неисправности оборудования, учитывая, как долгосрочные тенденции, так и краткосрочные изменения. Таким образом, выбор ARIMA в качестве основного инструмента для анализа обусловлен его эффективностью в предсказании будущих событий на основе прошлых данных.

6. Метод ARIMA

В методе ARIMA (Autoregressive Integrated Moving Average), каждый компонент служит определенной цели в анализе и прогнозировании временных рядов [5]. Эта модель используется для анализа временных рядов, корректно проанализировать данные или предсказать будущие точки в серии. Она сочетает в себе три основных аспекта: авторегрессионный (AR), интегрированный (I) и скользящее среднее (MA).

Авторегрессионный компонент (AR) полагается на зависимости между наблюдениями и некоторыми их предыдущими значениями. Этот компонент определяется параметром p, который указывает число лагов (отступов назад по временной шкале), используемых в модели. Например, если p=3, текущее значение временного ряда будет рассчитываться с использованием значений на три шага назад.

Интегрированный компонент (I) отражает количество раз, которое данные временного ряда должны быть дифференцированы до достижения стационарности. Нестационарные данные характеризуются трендами и сезонностью, которые могут быть устранены путем дифференцирования. Параметр d определяет, сколько раз временной ряд должен быть дифференцирован.

Компонент скользящего среднего (MA) моделирует ошибку временного ряда как линейную комбинацию ошибок прошлых временных точек. Параметр q указывает количество терминов скользящего среднего в модели.

Определение параметров p, d, и q является ключевым в процессе моделирования ARIMA и требует тщательного анализа временного ряда. Возможны три основных случая:

- выбор параметра p основан на анализе автокорреляционной функции (ACF) временного ряда;
- выбор параметра d определяется необходимостью достижения стационарности ряда, часто проверяемой с помощью расширенного теста Дики-Фуллера;
- выбор параметра q основывается на анализе частичной автокорреляционной функции (PACF);

Модель ARIMA может быть настроена так, чтобы учитывать только прошлые значения (если q=0), только ошибки (если p=0), или оба этих фактора одновременно. Настройка этих параметров зависит от специфических требований к анализу и может варьироваться в зависимости от прикладной задачи. Это позволяет достигать компромисса между точностью модели и её сложностью, так как каждый из параметров вносит свой вклад в общую способность модели к прогнозированию.

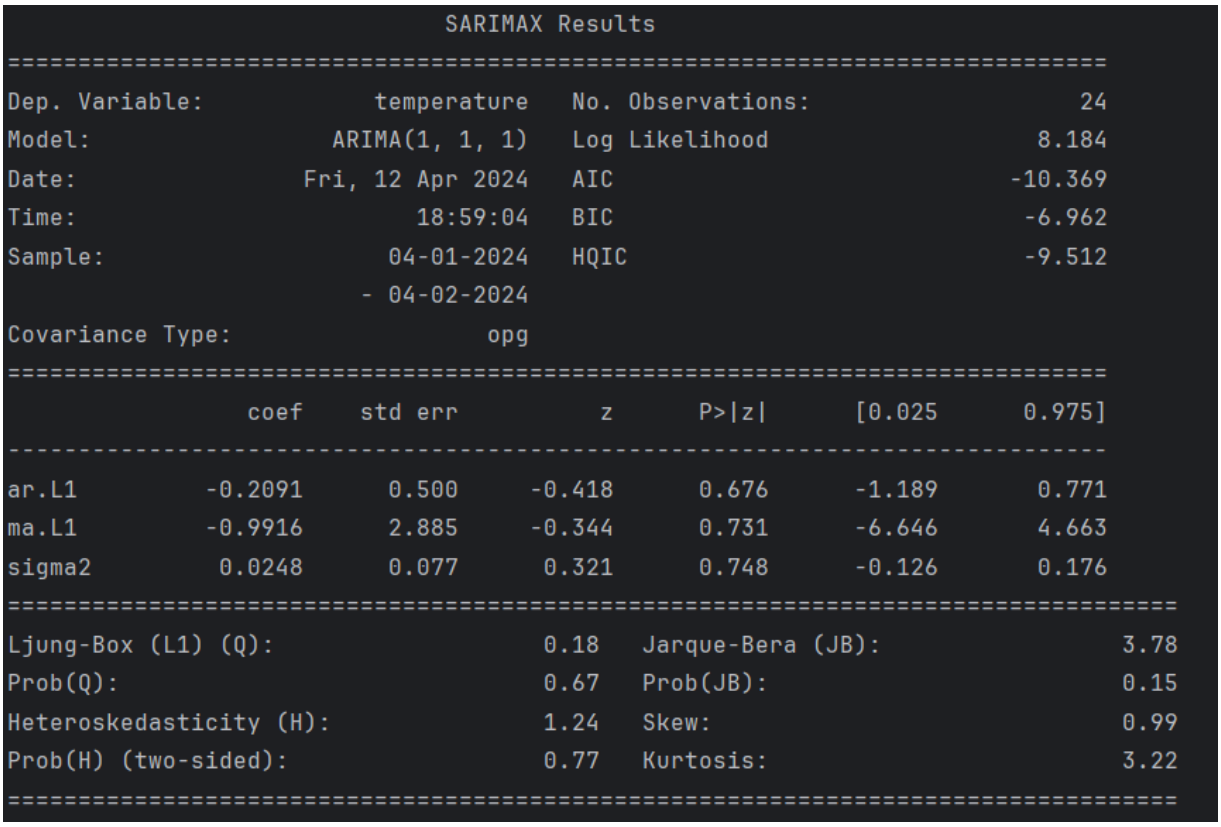
7. Пример прогнозирования с помощью модуля ARIMA

Модуль анализа состояния промышленного оборудования был реализован на языке программирования Python [6]. Были использованы стандартные библиотеки, такие как Pandas [7] для обработки временных рядов и Statsmodels [8] для построения ARIMA моделей. Программное обеспечение было спроектировано с учётом требований к масштабируемости и интеграции, позволяя легко адаптировать модуль под различные промышленные среды. В процессе разработки особое внимание уделялось точности и скорости вычислений, что критически важно для обеспечения обработки данных в реальном времени. Результаты были представлены на наборе данных, отражающих реальные условия эксплуатации оборудования, что подтвердило эффективность подхода. В качестве демонстрационного примера Тестирование разработанного приложения было проведено на лазерном станке Raspberry Pi в течении 24 часов. Были записаны ключевые параметры работы станка, включая температуру и потребление электроэнергии, с почасовой периодичностью. Целью данного мониторинга было создание модели, способной предсказывать потенциальные сбои в работе станка, что позволяет предпринимать профилактические меры и минимизировать риск внезапных остановок производства. На рисунке 2 представлены результаты замеров, которые сохраняются в базу данных.

	timestamp [PK] timestamp without time zone	temperature numeric (5,2)	pressure numeric (5,2)	electricity numeric (5,2)
1	2024-04-01 00:00:00	20.50	1.02	0.30
2	2024-04-01 01:00:00	20.70	1.03	0.31
3	2024-04-01 02:00:00	20.60	1.04	0.29
4	2024-04-01 03:00:00	20.80	1.02	0.32
5	2024-04-01 04:00:00	20.90	1.05	0.33
6	2024-04-01 05:00:00	21.00	1.03	0.34
7	2024-04-01 06:00:00	21.10	1.02	0.35
8	2024-04-01 07:00:00	20.90	1.04	0.36
9	2024-04-01 08:00:00	21.00	1.05	0.37
10	2024-04-01 09:00:00	20.80	1.03	0.38
11	2024-04-01 10:00:00	20.70	1.02	0.39
12	2024-04-01 11:00:00	20.60	1.05	0.40
13	2024-04-01 12:00:00	20.50	1.02	0.41
14	2024-04-01 13:00:00	20.80	1.03	0.42
15	2024-04-01 14:00:00	20.70	1.04	0.43
16	2024-04-01 15:00:00	20.60	1.05	0.44
17	2024-04-01 16:00:00	20.50	1.02	0.45
18	2024-04-01 17:00:00	20.80	1.03	0.46
19	2024-04-01 18:00:00	20.70	1.04	0.47
20	2024-04-01 19:00:00	20.60	1.05	0.48
21	2024-04-01 20:00:00	20.50	1.02	0.49
22	2024-04-01 21:00:00	20.80	1.03	0.50
23	2024-04-01 22:00:00	20.70	1.04	0.51

Рис. 2. Результаты замеров

Для анализа собранных данных была использована разработанный модуль авторегрессии интегрированного скользящего среднего. На рисунке 2 представлены результаты анализа данных.



SARIMAX Results

Dep. Variable:	temperature	No. Observations:	24
Model:	ARIMA(1, 1, 1)	Log Likelihood	8.184
Date:	Fri, 12 Apr 2024	AIC	-10.369
Time:	18:59:04	BIC	-6.962
Sample:	04-01-2024 - 04-02-2024	HQIC	-9.512
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.2091	0.500	-0.418	0.676	-1.189	0.771
ma.L1	-0.9916	2.885	-0.344	0.731	-6.646	4.663
sigma2	0.0248	0.077	0.321	0.748	-0.126	0.176

Ljung-Box (L1) (Q):	0.18	Jarque-Bera (JB):	3.78
Prob(Q):	0.67	Prob(JB):	0.15
Heteroskedasticity (H):	1.24	Skew:	0.99
Prob(H) (two-sided):	0.77	Kurtosis:	3.22

Рис. 3 Результаты анализа

На рисунке 3 можно выделить следующие ключевые показатели:

- коэффициенты модели;
- диагностику остатков;
- проверки стабильности ;
- значение сигма-квадрат;

Коэффициенты авторегрессии ar.L1 и скользящего среднего ma.L1 имеют p-значения, превышающие порог в 0.05, что не позволяет признать их статистически значимыми. Такие результаты не выявляют надежных зависимостей между последовательными измерениями во временном ряду, что может свидетельствовать о стабильности изучаемого процесса, поскольку отсутствует выраженная автокорреляция.

Тест Льюнга-Бокса (Ljung-Box Test) с p-значением 0.67 указывает на отсутствие автокорреляции остатков модели, подтверждая адекватность модели в отношении рандомизированных временных шумов. Тест Жарка-Бера (Jarque-Bera Test) с p-значением 0.15 говорит о том, что остатки распределены нормально, отсутствуют скошенность и аномальные всплески.

Коэффициент гетероскедастичности (H) с p-значением 0.77 не демонстрирует наличия изменчивости дисперсии остатков модели, что характеризует систему как стабильную и невосприимчивую к внезапным колебаниям, которые могут указывать на риски перегрева или перенапряжения.

Низкое значение параметра сигма-квадрат свидетельствует о том, что модель имеет малую ошибку прогнозирования. Это свидетельствует о высокой точности модели и предполагает, что система работает стабильно, без непредвиденных скачков или сбоев.

В совокупности эти результаты представляют собой убедительные доказательства надежности и эффективности мониторинга системы с использованием выбранной

статистической модели. Отсутствие признаков потенциального перегрева или перенапряжения в оборудовании позволяет предположить, что текущие условия эксплуатации находятся в приемлемых рамках, и система может продолжать функционировать без значительного риска нежелательных инцидентов.

Заключение

В данной статье была представлена концепция мониторинга состояния промышленного оборудования с применением метода ARIMA для анализа временных рядов. Реализация подобного подхода в виде системы мониторинга позволяет проводить непрерывное наблюдение за параметрами оборудования, что является ключевым аспектом для предотвращения непредвиденных остановок и сбоев в работе. Показано, что использование статистических методов анализа временных рядов способствует оперативному выявлению и устранению потенциальных проблем, а также обеспечивает важное преимущество в плане оптимизации работы оборудования.

Список литературы

1. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – 2-е изд. – М.: Вильямс, 2006. – 1103 с.
2. Бокс Дж., Дженкинс Г. Анализ временных рядов: прогнозирование и управление / Дж. Бокс, Г. Дженкинс. – М.: Мир, 1974. – 550 с.
3. Брош Дж. Применение ARIMA-моделей для анализа временных рядов / Дж. Брош. – СПб.: Питер, 2022. – 287 с.
4. Эндерс У. Анализ временных рядов / У. Эндерс. – М.: Пиндар, 2020. – 712 с.
5. Чатфилд К. Анализ временных рядов: теория и практика / К. Чатфилд. – М.: Финансы и статистика, 2019. – 304 с.
6. Официальная документация Python. – Режим доступа: https://docs.python.org/3/. – (Дата обращения: 14.03.2024).
7. Официальная документация библиотеки pandas. – Режим доступа: https://pandas.pydata.org/pandas-docs/stable/. – (Дата обращения: 15.03.2024).
8. Официальная документация библиотеки Statsmodels. – Режим доступа: https://www.statsmodels.org/stable/index.html. – (Дата обращения: 17.03.2024).

УДК 004.94

ГЕНЕРАЦИЯ ЛАБИРИНТА НА ИГРОВОМ ДВИЖКЕ UNITY3D

А.М. Титова, Е.В. Трофименко

Воронежский государственный университет

Введение

В последние годы появляется все больше новых игр в жанре лабиринт. Эти игры способствуют развитию зрительно-моторной координации, когнитивного развития и критического мышления, поскольку игроки проходят сложные лабиринты, чтобы достичь желаемого результата. Таким образом целью разработчика становится создание интересного, не тривиального, но реального в прохождении лабиринта. Для этого используются различные алгоритмы генерации лабиринта.

В настоящей статье рассматривается процесс генерации лабиринта на игровом движке Unity3D.

1. Описание метода

Существует два основных подхода для генерации лабиринта: метод вырезания проходов и метод добавления стен. Метод вырезания проходов означает, что изначально создаются сплошные блоки и в процессе работы алгоритма в них удаляются некоторые стены, таким образом генерируя лабиринт. В противоположном случае метод генерации стен означает, что изначально создаются пустые области и в процессе работы алгоритма они заполняются стенами, создавая лабиринт. Существует множество различных алгоритмов генерации лабиринта. Каждый из них имеет свои определённые характеристики. Большинство из них предполагает создание лабиринта путем вырезания проходов, однако почти каждый можно реализовать и путём добавления стен, если не указано обратное. К примеру для генерации лабиринта можно использовать алгоритм «Рекурсивный бэктрекер», алгоритм Эллера, алгоритм Крускала и многие другие. В данной статье будет использован алгоритм Эллера для генерации лабиринта основанный на подходе вырезания проходов.

При создании лабиринта с помощью данного алгоритма процент тупиковых ячеек составляет 28%. Память требующаяся для реализации алгоритма составляет N, где N – длина одной строки. Также при использовании этого алгоритма нет необходимости хранить весь лабиринт целиком, и есть возможность бесконечного добавления новых строк. Время

требующееся для создания лабиринта размерности 100x100 равно 20 секунд. Процент ячеек через, которые проходит путь решения составляет 4.2%.

Алгоритм Эллера был выбран для реализации так как занимает наименьшее количество памяти, а так же является одним из самых быстрых по работе. Количество тупиковых ячеек является достаточным для создания лабиринта, который будет являться не тривиальным, а интересным в прохождении.

1.1. Алгоритм Эллера

Шаги алгоритма:

1. Создать первую строку. Ни одна ячейка не будет являться частью ни одного множества.
2. Присвоить ячейкам, не входящим в множество, свое уникальное множество.
3. Создать правые границы, двигаясь слева направо:
    3.1. Случайно решить, добавлять границу или нет.
        3.1.1. Если текущая ячейка и ячейка справа принадлежат одному множеству, то создать границу между ними (для предотвращения зацикливаний).
        3.1.2. Если не добавили границу, то объединить два множества, в которых находится текущая ячейка и ячейка справа.
4. Создать границы снизу, двигаясь слева направо:
    4.1. Случайно решить, добавлять границу или нет. Необходимо убедиться, что каждое множество имеет хотя бы одну ячейку без нижней границы (для предотвращения изолирования областей).
        4.1.1. Если ячейка в своем множестве одна, то не нужно создавать границу снизу.
        4.1.2. Если ячейка одна в своем множестве без нижней границы, то не нужно создавать нижнюю границу.
5. Решить будут ли добавлены ещё строки или лабиринт закончен.
    5.1. Если будет добавлена строка, то:
        5.1.1. Вывести текущую строку.
        5.1.2. Удалить все правые границы.
        5.1.3. Удалить ячейки с нижней границей из их множества.
        5.1.4. Удалить все нижние границы.
        5.1.5. Продолжить с шага 2.
    5.2. Если лабиринт закончен, то:
        5.2.1. Добавить нижнюю границу к каждой ячейке.
        5.2.2. Двигаясь слева направо:
            5.2.2.1. Если текущая ячейка и ячейка справа члены разных множеств, то:
                5.2.2.1.1. Удалить правую границу.
                5.2.2.1.2. Объединить множества текущей ячейки и ячейки справа.
                5.2.2.1.3. Вывести завершающую строку.

2. Реализация

При генерации лабиринта в игровом движке Unity3D необходимо реализовать следующие этапы:

1. Необходимо написать код алгоритма генерации лабиринта (алгоритма Эллера).

2. Чтобы показать пример работы необходимо создать базовый объект - «префаб», в данном случае им будет является неполный куб, состоящий из нижней, задней и левой граней, а также сцену, в которой будет создан лабиринт.

Задача была реализована с использованием языка программирования C# и возможностей движка Unity3D.

При создании данного набора инструментов (далее — ассетов) для игрового движка Unity3D была разработана следующая диаграмма вариантов использования, представленная на рисунке 1.

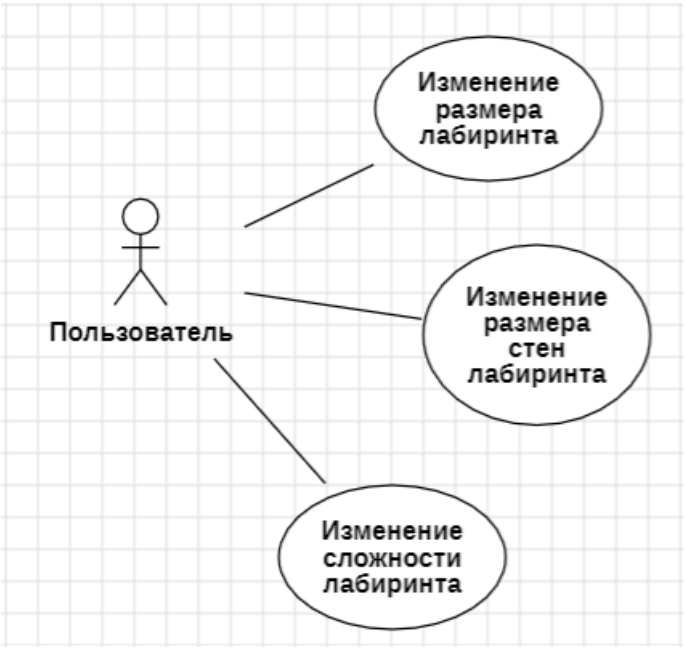


Рис. 1. *Диаграмма вариантов использования проекта*

На данной диаграмме обозначены все необходимые варианты использования ассетов, которые могут пригодится пользователю во время игры. У него есть возможность управлять размером лабиринта и высотой стен, из которых он состоит. А также изменять сложность лабиринта, путём изменения частоты генерируемых тупиковых проходов.

Для генерации лабиринта по алгоритму Эллера на движке Unity3D были реализованы следующие шаги:

1. Создать «префаб» - неполный куб, состоящий из нижней грани, а также задней и левой. «Префаб», а точнее стены и пол лабиринта созданы из встроенного в Unity3D 3D объекта куб (3D Object → Cube). Таким образом для создания одного «префаба» было использовано три куба, которым был задан нужный размер, они соединены определённым образом, показанном на рисунках 2 и 3. Из этих «префабов» будет состоять лабиринт. Один «префаб» — одна ячейка лабиринта, состоящая из пола и двух стен.

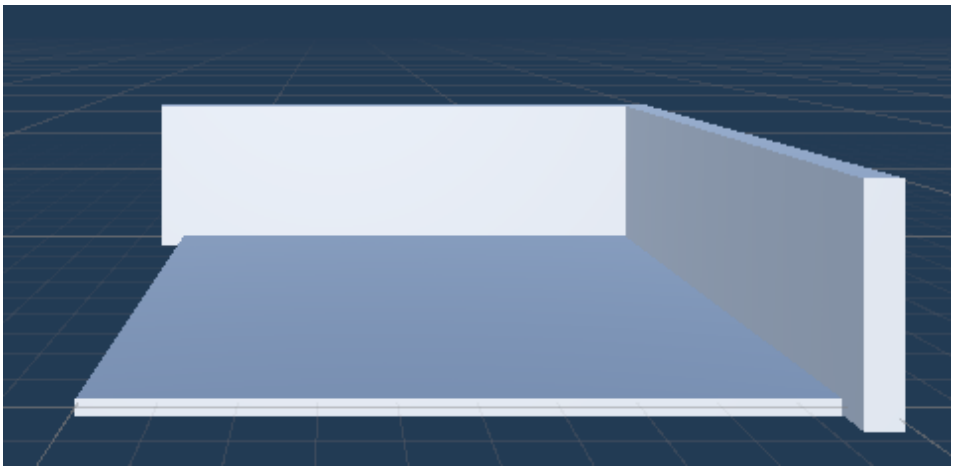


Рис. 2. *«Префаб» - вид прямо*

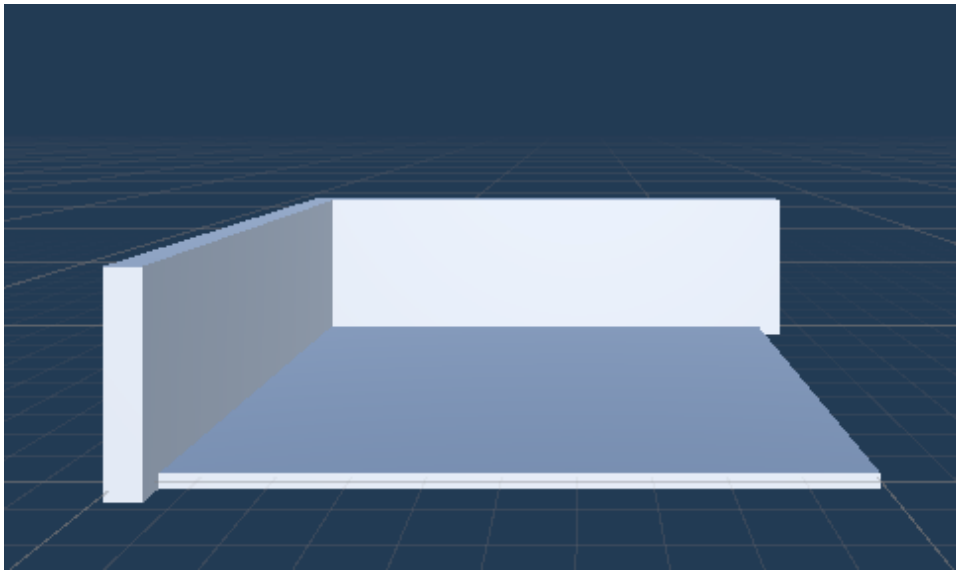


Рис.3. *«Префаб» - вид сбоку*

2. Для генерации лабиринта были реализованы три класса:

- MazeSpawnerEller

Отвечает за отображение лабиринта на сцене.

- _MazeGeneratorCell

Хранит информацию о ячейке лабиринта (расположение ячейки и видимость стен).

- MazeGeneratorEller

Содержит реализацию алгоритма Эллера. Полями класса являются ширина и длина лабиринта.

В результате работы получается сцена, представленная на рисунках 5, 6 и 7. На ней сгенерирован лабиринт размерности 23x15.

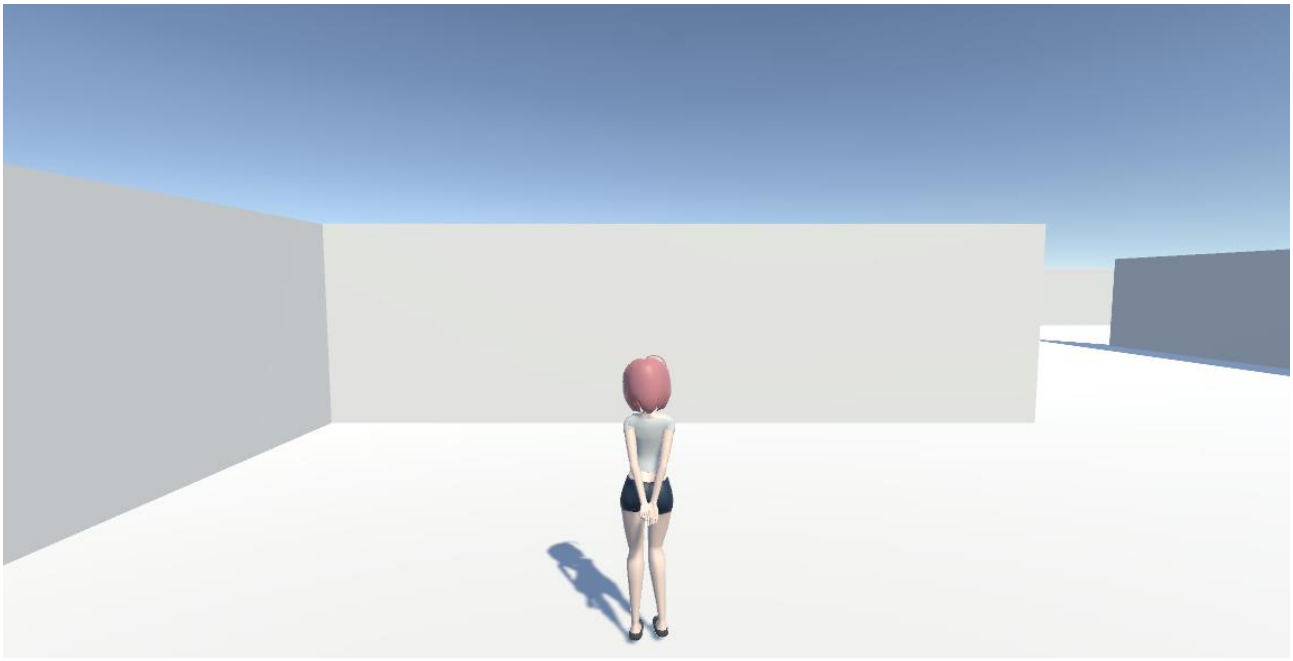


Рис. 5. *Сгенерированный лабиринт — вид от лица игрока*

Рис. 6. *Сгенерированный лабиринт — вид сверху*

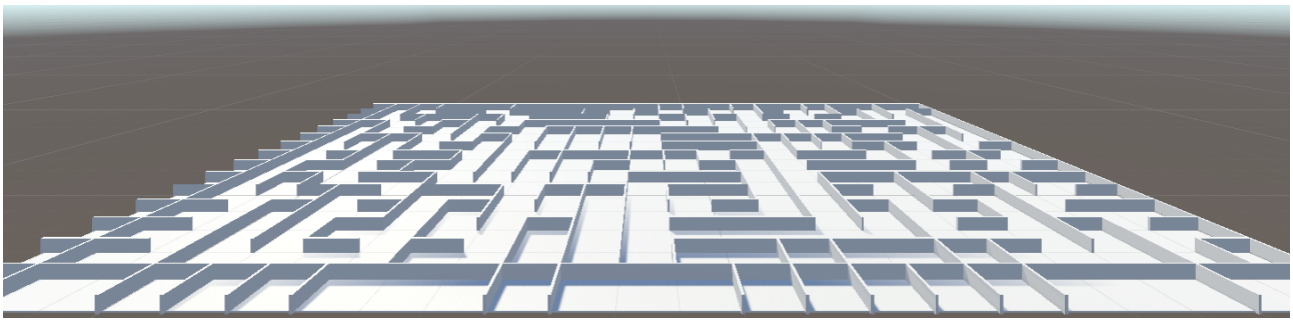


Рис.7. *Сгенерированный лабиринт — вид спереди*

Заключение

В данной статье была рассмотрена генерация лабиринта по алгоритму Эллера на игровом движке Unity3D. В результате работы был реализован алгоритм, созданы «префаб» и сцена для демонстрации его работы.

Литература

1. Maze Classification. – Режим доступа: https://www.astrolog.org/labyrnth/algrithm.html. – (Дата обращения: 27.03.2024).
2. Maze Algorithms. – Режим доступа: http://www.jamisbuck.org/mazes/. – (Дата обращения: 27.03.2024).
3. Maze Generation: Eller's Algorithm. – Режим доступа: https://weblog.jamisbuck.org/2010/12/29/maze-generation-eller-s-algorithm. – (Дата обращения: 27.03.2024)
4. Хоккинг, Дж. Unity в действии. Мультиплатформенная разработка на C# / Дж. Хоккинг. - 3-е изд., перераб. И доп. – Санкт - Петербург: Питер, 2023. – 448 с.
5. Корнилов, А. Unity. Полное руководство / А. Корнилов. – 2-е изд., перераб. И доп. – Москва: ДМК Пресс, 2021. – 496 с.

УДК 004.8

Обнаружение и классификация дефектов на поверхности стали.

Д. Д. Толмачев

Воронежский государственный университет

Введение

В современных условиях цифровая обработка и анализ изображений приобрели огромную популярность, находя применение в различных областях. Начиная от распознавания жестов и эмоций, и беспилотного управления автомобилями, и заканчивая диагностикой и анализом медицинских изображений, эти технологии стали неотъемлемой частью нашей повседневной жизни.

Компьютерное зрение оказывает значительное влияние на различные отрасли, облегчая и автоматизируя множество задач, которые ранее требовали человеческого вмешательства. Уникальность данной технологии заключается в его многогранности, и решения, разработанные для одной области, часто могут успешно адаптироваться для других прикладных областей. Несмотря на многочисленные исследования в области обработки изображений, универсального решения, подходящего для всех задач, на данный момент не существует. Каждая сфера требует особого внимания к своим уникальным условиям.

Важной задачей является разработка алгоритма, способного решать поставленные задачи с высокой скоростью и точностью. Однако даже при современных технологиях остаются условия, представляющие определенные трудности. Например, многие методы могут быть неустойчивы к изменениям интенсивности света или его направления, в то время как другие могут сталкиваться с проблемами при изменении масштаба.

Целью статьи является реализация алгоритма, способного распознавать дефекты на изображении стали, а именно определять наличие, точное расположение и вид дефекта на стальной заготовке.

1. Постановка задачи

За основу данной работы была взята задача обнаружения дефектов на поверхности стали. Задача была поставлена компанией “Северсталь” в качестве конкурсной работы на платформе дочерней компании Google LLC под названием Kaggle, являющейся интернет-сообществом специалистов в области обработки данных и машинного обучения. Главная цель этой задачи — автоматизировать обнаружение и классификацию дефектов стальной продукции на комбинате.

На входе система должна получать изображение стальной заготовки, а на выходе давать информацию о наличии или отсутствии дефектов и принадлежность к определенным классам, при наличии дефектов.

2. Исходные данные

Исходные данные представлены в виде набора обучающих изображений с координатами масок для каждого класса дефектов и набора тренировочных изображений. Классы в наборе данных для обучения распределены неравномерно (рис. 1).

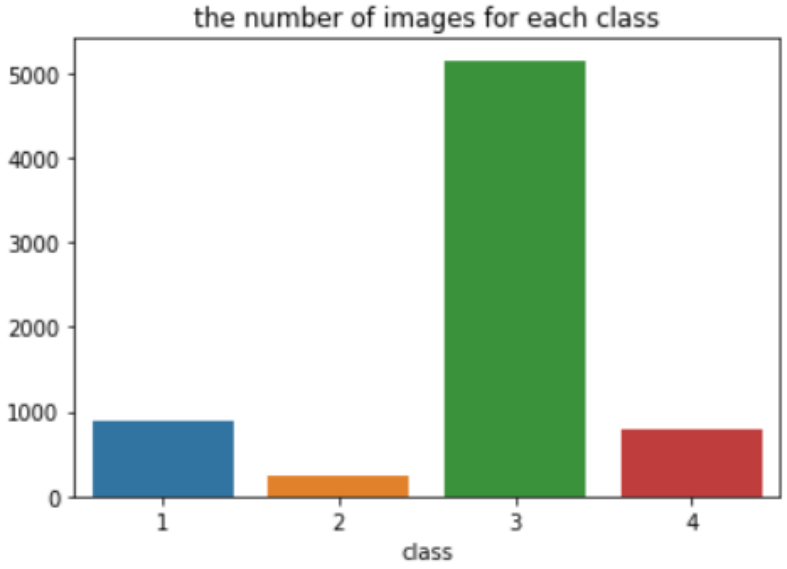


Рис. 1. Распределение классов в исходном наборе данных

На рис. 2 приведен пример изображения из обучающего набора данных с наложенной маской на дефект класса 3.

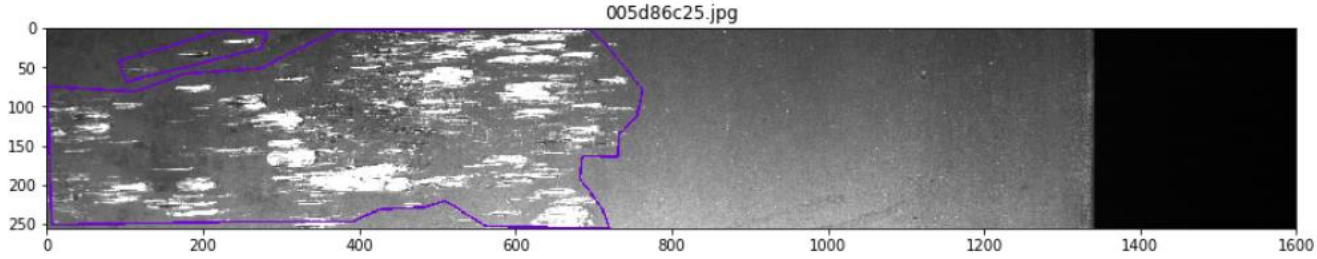


Рис. 25. Пример изображения из обучающего набора данных

3. Обзор используемой модели

Решение поставленной задачи реализовано с использованием модели для сегментации изображений — YOLOv8n-seg (YOLO — название серии моделей; v8 — 8-я версия; n — сокращение от nano, является указанием на размер модели; -seg — вариант модели, ориентированный на сегментацию изображений).

YOLOv8 — это современный алгоритм обнаружения объектов, известный своей высокой точностью и производительностью в реальном времени. Это особенно эффективно, когда дело доходит до сегментации экземпляров, которая включает в себя идентификацию и разграничение отдельных объектов на изображении.

YOLO (You Only Look Once) несет в себе философию смотреть на картинку один раз, и за этот один просмотр (то есть один прогон картинки через одну нейронную сеть) делать все необходимые определения объектов.

4. Алгоритм работы YOLOv8

Первым делом изображение разбивается на клетки (рис. 3), которые называются grid cells. Они лежат в основе идеи YOLO.

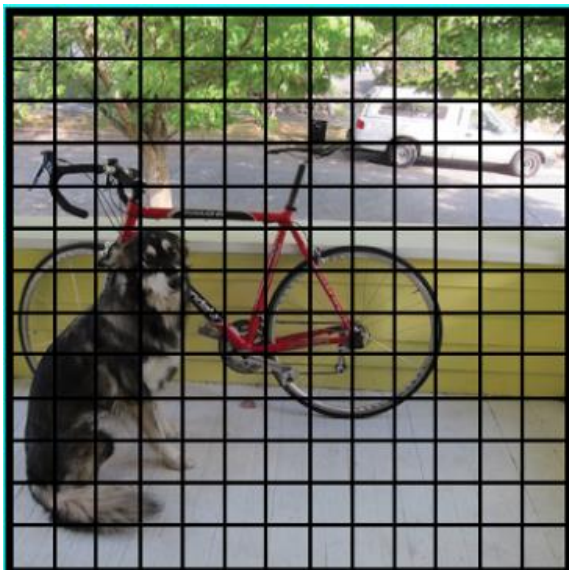


Рис.3. Изображение разбитое на клетки (grid cells)

Каждая клетка является «якорем», к которому прикрепляются ограничивающие прямоугольники (bounding boxes), что представлено на рис. 4. То есть вокруг клетки рисуются несколько прямоугольников для определения объекта (поскольку непонятно, какой формы прямоугольник будет наиболее подходящим, их рисуют сразу несколько и разных форм), и их позиции, ширина и высота вычисляются относительно центра этой клетки.

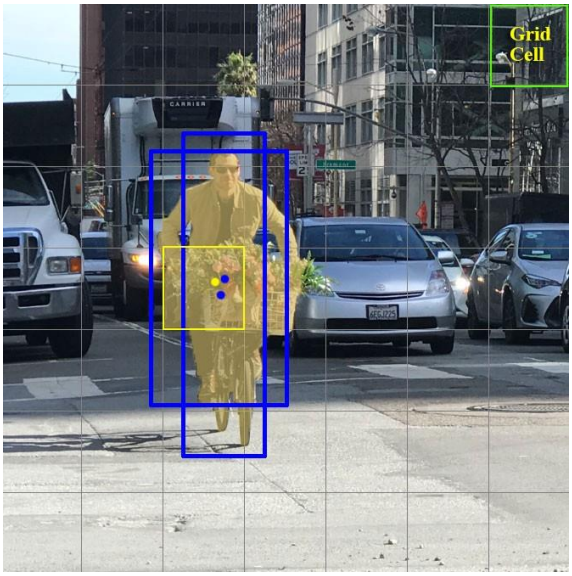


Рис. 4. Изображение с прикрепленными ограничивающими прямоугольниками

Ограничивающие прямоугольники определяются с помощью якорных прямоугольников (anchor boxes). Якорные прямоугольники задаются в самом начале либо самим пользователем, либо их размеры определяются исходя из размеров ограничивающих прямоугольников, которые есть в наборе данных, на котором будет тренироваться модель. Обычно задают порядка 3 различных якорных прямоугольников, которые будут нарисованы вокруг (или внутри) одной клетки (рис. 5).

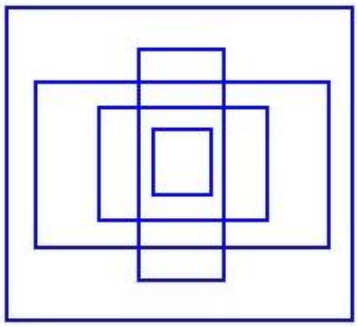


Рис.5. Пример заданных якорных прямоугольников

Затем изображение из набора данных прогоняется через нейронную сеть и для каждой клетки определяются две вещи (рис. 6):

1. Какой из 3-х, нарисованных вокруг клетки, якорных прямоугольников подходит больше всего и как его можно немного подправить для того, чтобы он хорошо вписывал в себя объект.
2. Какой объект находится внутри этого якорного прямоугольника и есть ли он вообще.

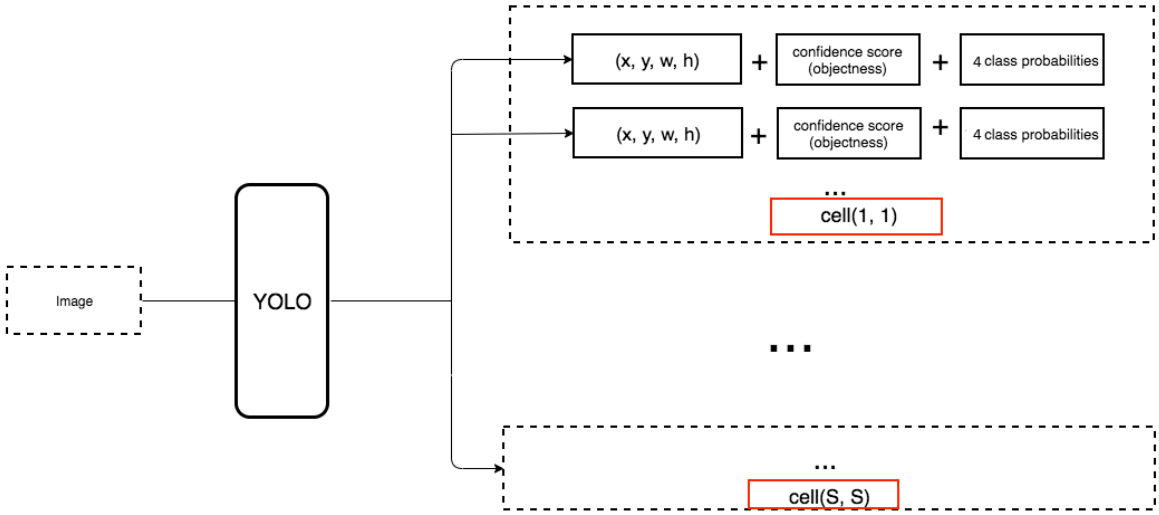


Рис.6. Вывод для каждой клетки

Параметр objectness (предметность) определяется с помощью метрики IoU во время обучения. Метрика IoU представляет собой отношение пересечения областей якорных прямоугольников к объединению этих областей.

В начале можно выставить порог для этой метрики, и если предсказанный ограничивающий прямоугольник будет выше этого порога, то у него будет предметность равна единице, а все остальные ограничивающие прямоугольники, будут исключены. Величина предметности необходима при подсчете показателя уверенности — вероятности того, что это именно нужный объект расположен внутри предсказанного прямоугольника.

YOLO предсказывает 5 параметров (для каждого якорного прямоугольника для определенной клетки):

1. Примерное расположение искомого объекта
2. Координата верхней левой точки нужной клетки
3. Ширина и высота определенного якорного прямоугольника
4. Параметры для предположительного ограничивающего прямоугольника
5. Показатель уверенности

Задача YOLO — максимально точно предсказать эти параметры, чтобы достоверно определять объект на картинке. А показатель уверенности, который определяется для каждого предсказанного ограничивающего прямоугольника, является неким фильтром для того, чтобы отсеять совсем неточные предсказания. Каждый предсказанный ограничивающий прямоугольник умножается на IoU вероятность того, что это определенный объект (вероятностное распределение рассчитывается во время обучения нейронной сети). Затем берется лучшая вероятность из всех возможных, и если число после умножения превышает определенный порог, то этот предсказанный ограничивающий прямоугольник оставляется на изображении.

На рис. 7 представлено изображение, на котором остались только предсказанные ограничивающие прямоугольники с высоким показателем уверенности.

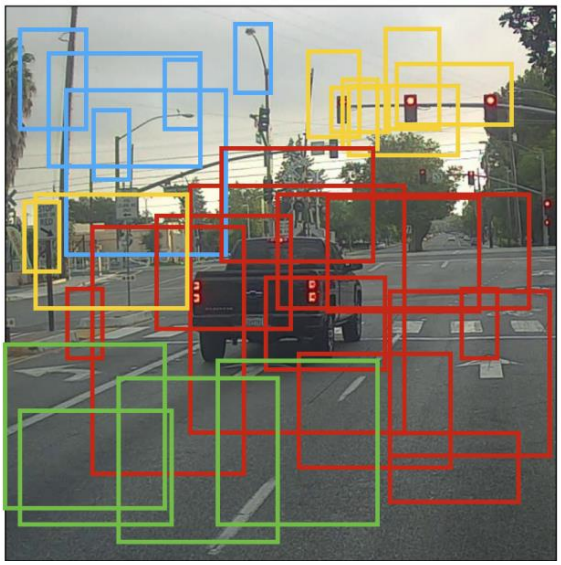


Рис.7. Изображение с самыми точными предсказанными ограничивающими прямоугольниками

В завершение используется техника NMS (non-max suppression), чтобы отфильтровать ограничивающие прямоугольники таким образом, чтобы для одного объекта был только один предсказанный ограничивающий прямоугольник. На рис. 8 представлен результат работы NMS.

Рис. 8. Пример результата работы NMS

5. Обучение модели

В обучении модели используется набор данных в формате YAML, изображения в котором имеют размер 640x640 пикселей. Размер обучающего набора данных равен 5999, а валидационного - 667. Обучение модели было проведено на 40 эпохах. Параметру batch было присвоено значение 16. Этот параметр является размером партии для обучения, указывающим, сколько изображений обрабатывается перед обновлением внутренних параметров модели. В

ходе данной работы не была использована предобученная модель, так как она была обучена на 80 классах изображений, что не соответствует условию поставленной задачи. Параметры модели и фрагмент обучения представлены на рис. 9 и рис. 10 соответственно.

```
results = model.train(data='train.yaml', epochs=40, batch=16, pretrained=False)
```

Рис. 926. Параметры обучения модели

```
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       1/40      3.11G      3.467      4.994      4.066      3.531         63        640: 100%|██████████| 1875/1875 [08:48<00:00,
3.55it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95):
100%|██████████| 21/21 [00:07<00:00,  2.83it/s]
                   all        667       2438      0.285     0.0286     0.0114     0.0031      0.281     0.0284    0.00793    0.00189

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       2/40      3.37G      2.391      3.365      2.795      2.287         62        640: 100%|██████████| 1875/1875 [08:39<00:00,
3.61it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95):
100%|██████████| 21/21 [00:07<00:00,  3.00it/s]
                   all        667       2438      0.342      0.132     0.0547     0.0157       0.34      0.125     0.0447     0.0136

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       3/40      3.12G      2.074       3.16      2.494       1.95         47        640: 100%|██████████| 1875/1875 [08:41<00:00,
3.60it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95):
100%|██████████| 21/21 [00:06<00:00,  3.02it/s]
                   all        667       2438      0.392      0.124     0.0797     0.0265      0.377      0.115     0.0656     0.0206

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       4/40      3.23G      1.954      3.046      2.367      1.805         47        640: 100%|██████████| 1875/1875 [08:39<00:00,
3.61it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95):
100%|██████████| 21/21 [00:06<00:00,  3.06it/s]
                   all        667       2438       0.44      0.176      0.117     0.0399      0.432      0.169      0.104      0.033
```

Рис. 10. Фрагмент обучения модели

6. Результат обучения модели

Метрики по классам для обученной модели представлены на рис. 11.

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)	Mask(P	R	mAP50	mAP50-95)
all	667	2438	0.465	0.364	0.356	0.157	0.38	0.287	0.273	0.102
0	667	365	0.467	0.438	0.42	0.166	0.373	0.301	0.267	0.0959
1	667	45	0.403	0.111	0.124	0.0326	0.201	0.0444	0.0673	0.0162
2	667	1840	0.487	0.428	0.402	0.168	0.418	0.339	0.307	0.116
3	667	188	0.503	0.479	0.477	0.261	0.528	0.463	0.449	0.181

Рис. 11. Метрики по классам для обученной модели

Precision - точность модели в обнаружении дефектов в среднем составляет 0.45 для ограничивающего прямоугольника и 0.38 для маски.

Recall - способность модели идентифицировать все экземпляры объектов на изображении составляет 0.37 для ограничивающего прямоугольника и 0.29 для маски.

На рис. 12 представлен график метрики F1, на котором заметно, что максимальная точность F1 = 0.39 достигается при показателе уверенности равняющимся 0.2.

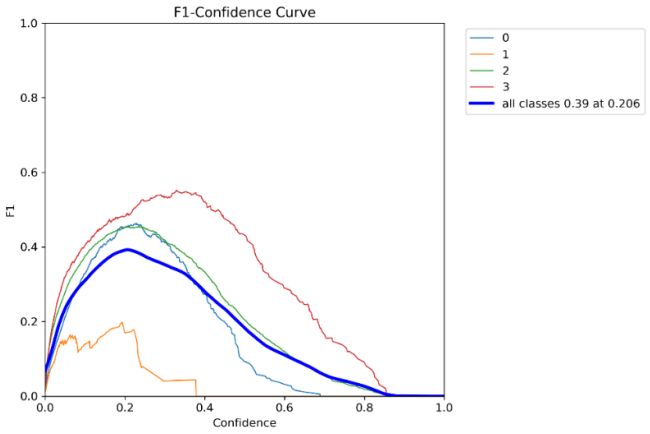


Рис. 12. График метрики F1 относительно показателя уверенности

Низкие показатели уверенности связаны с большим количеством ложных обнаружений при обучении модели, которые представлены на рис. 13.

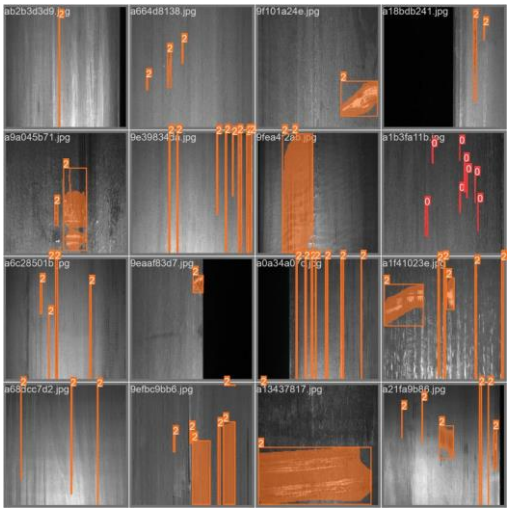
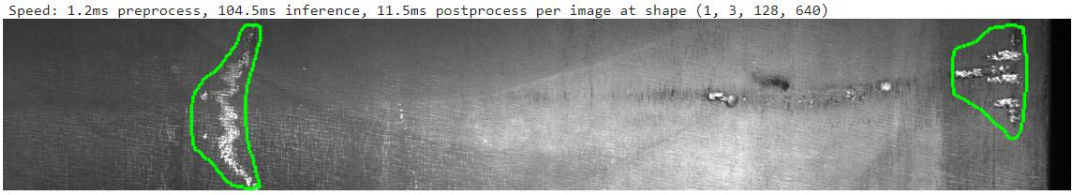


Рис. 127. Одна из партий обучения модели

Однако, как можно заметить на рис. 14, модель справляется с обнаружением и классификацией дефектов без ложных обнаружений.

```
image 1/1 /kaggle/input/severstal-steel-defect-detection/test_images/00513039a.jpg: 128x640 2 2s, 104.5ms
Speed: 1.2ms preprocess, 104.5ms inference, 11.5ms postprocess per image at shape (1, 3, 128, 640)
```



```
pred_results[0].boxes[1]
```

```
ultralytics.engine.results.Boxes object with attributes:

cls: tensor([2.], device='cuda:0')
conf: tensor([0.3825], device='cuda:0')
data: tensor([[1.4121e+03, 1.6410e+00, 1.5315e+03, 1.8656e+02, 3.8247e-01, 2.0000e+00]], device='cuda:0')
id: None
is_track: False
orig_shape: (256, 1600)
shape: torch.Size([1, 6])
xywh: tensor([[1471.7798,   94.1009,  119.3506,  184.9197]], device='cuda:0')
xywhn: tensor([[0.9199, 0.3676, 0.0746, 0.7223]], device='cuda:0')
xyxy: tensor([[1412.1045,    1.6410, 1531.4551,  186.5607]], device='cuda:0')
xyxyn: tensor([[0.8826, 0.0064, 0.9572, 0.7288]], device='cuda:0')
```

Рис. 14. Результат работы модели

Заключение

В ходе проведенного исследования была рассмотрена и обучена модель YOLOv8n-seg, а также описан алгоритм её работы. Обученная модель способна распознавать дефекты на поверхности стали и классифицировать их.

Стоит отметить, что разработка модели, способной точно обнаруживать дефекты на производстве, является важной задачей в области компьютерного зрения. Подобные разработки могут сильно снизить затраты на контроль качества на производствах и уменьшить количество брака за счет исключения человеческого фактора.

Литература

1. Бурков А. Машинное обучение без лишних слов / А. Бурков – Санкт-Петербург : Питер, 2020. – 192 с.

2. Гласснер Э. Глубокое обучение без математики. Т.1: Основы / пер. с анг. В.А. Яроцкого. – Москва : ДМК Пресс, 2019. – 584 с.

3. Гласснер Э. Глубокое обучение без математики. Т.2: Практика / пер. с анг. В.А. Яроцкого. – Москва : ДМК Пресс, 2020. – 610 с.

4. Коул А. Искусственный интелект и компьютерное зрение. Реальные проекты на Python, Keras и TensorFlow / А. Коул, С. Ганджу, М. Казам – Санкт-Петербург : Питер, 2023. – 624 с.

5. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow / Ж. Орельен – Санкт-Петербург : Диалектика, 2020. – 1040 с.

УДК *004.623*

АНАЛИЗ СПОСОБОВ ЗАГРУЗКИ ДАННЫХ В ОПЕРАТИВНЫЙ СЛОЙ КОРПОРАТИВНОГО ХРАНИЛИЩА ДАННЫХ БАНКА

П. Е. Толстых

Воронежский государственный университет

Введение

Для сбора, хранения и обработки информации, корректной работы банковских процессов и быстрого формирования аналитической, регламентированной и управленческой отчетности бизнес-пользователям необходимо единое корпоративное хранилище данных (далее — КХД).

Системами-источниками (далее — СИ) данных для КХД являются информационные системы, которые выполняют задачи обеспечения процессов банка. Формат хранения информации для систем может сильно различаться.

СИ используют следующие основные форматы хранения данных:

- реляционная база данных;
- файлы различных форматов;
- веб-интерфейс — данные хранятся на сервере, обращение к ним происходит с помощью *HTTP*-запросов.

В зависимости от способа хранения информации в СИ для загрузки данных в КХД используются различные алгоритмы и программные решения.

1. Загрузка из реляционной базы данных

Большинство СИ являются *OLTP*-системами.

OLTP (англ. *Online Transaction Processing*) или реляционные базы данных — это базы данных, основная цель которых — ввод, редактирование и удаление данных в режиме онлайн [1]. Такие системы позволяют быстро обрабатывать информацию и отслеживать порядок её появления с высокой точностью.

Загрузка информации оперативного слоя из СИ, представляющих собой *OLTP*-системы, наименее затратна со стороны КХД. Извлечение данных источника осуществляется с помощью

специальных программных решений. Для КХД, основанном на СУБД *Oracle*, такими решениями являются *Oracle Data Integrator* и *Oracle GoldenGate*.

1.1. Oracle Data Integrator

Oracle Data Integrator (далее — *ODI*) — решение, которое обеспечивает извлечение, преобразование и загрузку данных из разнообразных источников [2].

Для обеспечения загрузки из любой другой СУБД необходимо на стороне КХД создать таблицу, в которую будут загружены данные с источника. Обычно такие объекты называют *SNP*-таблицами (англ. snapshot — снимок), т. к. они сохраняют данные в том виде, в котором они находятся на источнике в текущий момент времени.

Для таблицы-источника и созданной *SNP*-таблицы необходимо создать *Datastore*. Это структура данных, которая совпадает по атрибутному составу с соответствующим (в основном, по названию) объектом СИ или КХД.

Основной процесс загрузки данных из таблицы системы-источника выполняет *Mapping*. Mapping в *ODI* — это логическая и физическая организация источников данных, целевых объектов и преобразований, посредством которых данные передаются от источника к целевому объекту [3].

Mapping сопоставляет атрибуты *Datastore* источника данных и атрибуты *Datastore* целевой области с помощью модулей знаний (шаблонов кода). Один из встроенных модулей знаний *LKM SQL to Oracle* позволяет загружать данные из любой реляционной СУБД в область преобразования данных *Oracle*. Информация загружается с помощью Агента *ODI*, который устанавливает связь с обеими СУБД.

После завершения работы *Mapping*, данные из таблицы системы источника будут загружены в соответствующую *SNP*-таблицу на стороне КХД.

После извлечения данных из СИ выполняется сравнение данных таблицы-снимка «*SNP*» с таблицей-зеркалом, содержащей актуальный срез источника, сформированный в результате прошедших загрузок. Результатом сравнения является набор изменённых, добавленных и удалённых записей источника. По полученному набору записей с помощью дополнительных *Datastore* и *Mapping* выполняется актуализация таблицы-зеркала источника с обязательной отметкой времени полученного изменения и флагом операции.

1.2. Oracle GoldenGate

Для таблиц источника, которые занимают более 30 гб памяти, а также имеют большую динамику роста — быстрый ежедневный прирост данных (например, таблицы, хранящие информацию о банковских договорах) в КХД используется загрузка средствами *Oracle GoldenGate* (далее — *OGG*). Данный инструмент является дорогостоящим продуктом, поэтому его применяют только в тех случаях, когда извлечение данных средствами *ODI* может нагрузить СИ и КХД и тем самым привести к значительному снижению их производительности.

OGG в КХД реализует механизм *CDC* (*Change Data Capture*) — решение для получения изменений данных и структур данных на основе анализа лог-файлов СУБД в режиме реального времени. Инструмент обогащает измененные данные метаданными, которые позволяют ответить на вопросы кто, когда и какие изменения выполнил.

Таким образом, *OGG* автоматически выполняет извлечение данных из системы источника в онлайн-режиме (что позволяет отследить абсолютно все изменения таблицы источника, в том числе ручные правки пользователей), а также проводит сравнение версий строк и построение истории по ключу таблицы.

Все необходимые настройки и таблицы для использования *OGG* выполняются и создаются на стороне источника. На стороне КХД для дальнейшей загрузки используется представление, основанное на этих таблицах.

2. Загрузка из файлов различных форматов

Некоторые системы хранят данные в файлах различного формата. В основном используются следующие форматы:

- *.TXT* — файл, который содержит текст, упорядоченный по строкам;
- *.CSV* — текстовый файл, в котором информация разделена символами-разделителями (запятая, точка с запятой и т. д.).

Из-за простоты и удобства записи информации СИ предпочитают именно такой способ хранения данных. Однако за счёт того, что в большинстве случаев заполнение производится пользователем увеличивается вероятность ошибки: например, случайно введённый лишний пробел, запятая или другая подобная опечатка. С точки зрения КХД такие данные довольно сложно извлекать из СИ (необходимо дополнительно проверять и обрабатывать ошибки ввода), а также структурировать согласно архитектуре хранилища.

Извлечение данных можно выполнять с помощью специального модуля знаний *ODI*, однако из-за разных форматов файлов, разделителей и т. д. большинство КХД создают собственные инструменты для загрузки, основанные на этом модуле знаний.

Для извлечения данных из файлов в КХД банка разработан специальный пакет. Пакет — это объект схемы, который группирует логически связанные типы, курсоры, константы, исключения, глобальные переменные и подпрограммы в единое целое [4]. Для его использования необходимо добавить в специальную настроечную таблицу (реализована на стороне КХД) следующую информацию:

- путь к директории, в которую ежедневно будут выкладываться файлы со стороны СИ;
- маску имени архива, который содержит файлы;
- маску имени файла, который необходимо загрузить в КХД;
- идентификатор типа файла (в основном берется из фиксированного списка типов, при необходимости с согласования бизнес-пользователей вводится новый тип).

Также необходимо во всех таблицах оперативного слоя добавить новое поле *ID_FILE* — уникальный идентификатор файла и создать *Datastore* для *SNP*-таблицы и для файла. В параметрах *Datastore* для файла обязательно указать *ResourceName = #GLOBAL.GV_FILE_NAME* и символы-разделители и создать *Mapping* сопоставления полей файла и *SNP*-таблицы.

Алгоритм работы механизма извлечения данных из файла СИ и загрузки этих данных в *SNP*-таблицу:

1) по заданному пути выполняется поиск архива, название которого удовлетворяет маске, если такой архив не найден, выполнение пакета завершается с ошибкой, иначе п. 2;

2) выполняется распаковка архива и поиск файла, название и тип которого удовлетворяют значениям этих параметров в настроечной таблице, если такой файл не найден, выполнение пакета завершается с ошибкой, иначе п. 3;
3) запуск сценария Mapping, в результате которого будет выполнен парсинг содержимого файла и запись информации в *SNP*-таблицу.

Дальнейшие действия (обновление зеркала и выстраивание истории) аналогично загрузке таблицы из реляционной базы данных с помощью *ODI*.

3. Загрузка из веб-интерфейса

Некоторые СИ являются веб-интерфейсом, который позволяет предоставить данные с сервера пользователю. В основе большинства таких веб-интерфейсов лежит *REST API*.

REST (англ. *Representational State Transfer* — передача состояния представления) *API* (англ. *Application Programming Interface* — программный интерфейс приложения) — это архитектурный подход, который устанавливает некие правила для интерфейса *API*. Это позволяет стандартизировать работу программных интерфейсов, сделать их более удобными и производительными. [5]

Основные принципы и ограничения, которым должен удовлетворять такой интерфейс:

- клиент-серверное взаимодействие: система *A* делает *HTTP*-запрос на *URL*, по которому развернута система *B*, система *B* отвечает;
- не хранит состояние: запрос от клиента должен содержать всю информацию, необходимую для ответа;
- возможность кэшируемости: запрос должен помечаться как кэшируемый или нет;
- наличие слоёв, скрытых от клиента [6].

Для извлечения данных из СИ, формат хранения данных которых является веб-интерфейсом, вместо пакета или *Mapping ODI* используются разработанные на стороне хранилища процедуры, позволяющие из КХД получить данные с сервера.

В этом случае КХД выступает в качестве клиента и выполняет запросы к СИ. Для реализации такого способа получения данных схеме, под которой будет выполняться процедура, необходимо выдать доступ в *ACL* (*Access Control List*) — списку правил, запрещающих или разрешающих использование ресурсов сети [7] (выполняется администратором базы данных), а также получить логин и пароль для доступа к серверу (запрашивается у СИ). Логин и пароль, а также *URL*, по которому будет происходить обращение, записывается в специальную настроечную таблицу.

Извлечение данных выполняется по следующему алгоритму с использованием встроенного пакета *Oracle* для работы с *REST API — UTL_HTTP* [8]:

1) со стороны КХД происходит обращение к серверу по *URL*;

2) выполняется запрос к веб-интерфейсу: чаще всего это *GET*- или *POST*-запрос (зависит от СИ и обычно указывается в документации);
3) сервер отправляет ответ, содержимое ответа приходит в формате *JSON* и записывается в таблицу на стороне КХД;
4) завершение запроса к веб-интерфейсу;
5) парсинг полученного ответа в соответствии с требованиями к атрибутному составу конечной таблицы.

Дальнейшие действия (обновление зеркала и формирование истории) аналогично загрузке таблицы из реляционной базы данных с помощью *ODI*.

Заключение

Для полноценной и качественной работы банка необходимо использовать корпоративное хранилище данных, которое позволяет обрабатывать данные из различных источников. Системы-источники могут хранить информацию в разных форматах и представлениях, поэтому важно подобрать способ загрузки, подходящий под конкретную СИ.

В ходе работы были проанализированы СИ, а также алгоритмы и программные решения, позволяющие загрузить информацию в оперативный слой КХД.

Большинство СИ представляют собой реляционную базу данных. Загрузка из них происходит посредством создания таблицы-снимка и формирования истории с помощью средств Oracle Data Integrator или Oracle GoldenGate.

Некоторые СИ хранят информацию в файлах различного формата или на сервере (веб-интерфейс). Загрузка из таких источников встречается редко и является более затратной со стороны КХД, т. к. требует дополнительной разработки и использования специальных механизмов для извлечения данных, а также обработок ошибок ввода.

Проведённый анализ позволяет корректно определить способ загрузки в случае появления новой СИ и минимизировать временные затраты при разработке новой таблицы из существующего источника.

Информация о научном руководителе: ст. преп. кафедры программного обеспечения и администрирования информационных систем ВГУ факультета ПММ Матвеева Мария Валерьевна.

Литература

1. Различия OLAP и OLTP. – Режим доступа: https://dzen.ru/a/W_VVflGEzACpkCj_. – (Дата обращения: 18.03.2024).

2. Oracle Data Integrator. – Режим доступа: https://www.fors.ru/businesssolutions/analytical-systems-and-data-warehouse/oracle-data-integrator / – (Дата обращения: 18.03.2024).

3. Mapping. – Режим доступа: https://docs.oracle.com/en/middleware/fusion-middleware data-integrator/12.2.1.3/using/mappings – (Дата обращения: 18.03.2024).

4. PL/SQL Packages. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/18/lnpls/plsql-packages.html – (Дата обращения: 19.03.2024).

5. REST API: что это такое и как работает. – Режим доступа: https://skillbox.ru/

media/code/rest-api-chto-eto-takoe-i-kak-rabotaet – (Дата обращения: 19.03.2024).

6.Что такое REST API? – Режим доступа: https://webformyself.com/chto-takoe-rest-api – (Дата обращения: 19.03.2024).

7. Access Control List – Режим доступа: https://itglobal.com/ru-ru/company/glossary/access-control-list / – (Дата обращения: 20.03.2024).

8. Фейерштейн, С. Oracle PL/SQL. Для профессионалов/ С. Фейерштейн, Б. Прибыл. – 6-е изд. – СПб.: Питер, 2015. – С. 765–773.

УДК 004.94

ОБЗОР АЛГОРИТМОВ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ КАРТ

Т.В. Тонких, Е.В. Трофименко

Воронежский государственный университет

Введение

В настоящее время процедурная генерация используется для создания уникальных карт, объектов и текстур. Процедурная генерация также используется в разработке игр как для формирования реиграбельности, так и для облегчения процесса создания объектов и игровых карт в частности. Процедурная генерация карт так же, как и ручное составление карт, позволяет генерировать закрытые области, например: пещеры, туннели, лабиринты, этажи с комнатами и коридорами и открытые пространства с чередующейся высотой, биомами, водоемами и природными объектами.

В настоящей статье приводится обзор трех алгоритмов процедурной генерации карты: алгоритм двоичного разбиения пространства, алгоритм туннелирования и шум Перлина.

1. Обзор алгоритмов процедурной генерации карты

Процедурная генерация позволяет создать карту с нуля и дает большую вариативность как разработчику из-за большого спектра и гибкости настроек, влияющих на конечный вид карты, так и пользователю, потому что при каждом запуске игры он получает новую область для исследования, которая никогда не повторится.

1.1 Алгоритм двоичного разбиения пространства

Алгоритм двоичного разбиения пространства является одним из распространенных способов процедурной генерации карты [1]. Данный метод позволяет генерировать карту, состоящую из комнат, соединенных коридорами на основе области, разделенной на части. В отличие от случайной генерации комнат, при использовании алгоритма двоичного разбиения пространства в результате получается конечное количество непересекающихся комнат. Количество коридоров можно настраивать в зависимости от задачи, но данный алгоритм позволяет соединить комнаты так, чтобы не образовывались циклы, то есть чтобы из каждой комнаты в любую другую существовал единственных проход. Также этот алгоритм позволяет настраивать величину пространства, минимальный размер комнат и ширину коридоров. Генерацию карт с помощью BSP-деревьев можно применять на двумерное и трехмерное пространство. Карты, сгенерированные данным способом, можно так же случайно заполнять игровыми объектами, окружением и игровыми персонажами.

1.2 Алгоритм туннелирования

Алгоритм туннелирования также широко используется для процедурной генерации карт [4]. В ходе работы данного алгоритма последовательно случайным образом генерируется положение, размер и форма комнаты, причем она не может пересекаться с ранее сгенерированной, и соединяется с получившейся на предыдущем шаге комнатой посредством вычисления минимальной стоимости пути. Преимуществом алгоритма туннелирования является широкая вариативность формы комнаты при ее генерации, она может получиться любой формы из стандартных фигур: прямоугольной, круглой, треугольной или составной как комбинация нескольких одинаковых или разных фигур. Это преимущество и обеспечивает сходство получившейся карты с туннелем. Также алгоритм позволяет гибко настраивать стоимость пути из комнаты в комнату, можно задавать разную цену единицы расстояния для перемещения вдоль стен комнаты, по полу комнаты и по коридору. Есть возможность настраивать размер пространства, минимальную и максимальную величину комнат, варианты форм комнат и ширину коридоров. Таким образом часто получаются комнаты, имеющие не только вход и выход. Алгоритм туннелирования можно применять как для генерации двумерных, так и трехмерных карт.

1.3 Шум Перлина

Шум Перлина — это градиентный шум, состоящий из набора псевдослучайных единичных векторов, расположенных в определенных точках пространства и интерполированных функцией сглаживания между этими точками [3]. Алгоритм шума Перлина считается самым популярным и распространенным способом процедурной генерации карт. Данный алгоритм позволяет создавать карты с изменением высот, или с изменением климата, или с изменением погоды, или с комбинацией нескольких видов карт путем наложения нескольких шумов. Карты, получающиеся в результате работы шума Перлина, имеют бесконечный потенциал к генерации на них природных объектов, игровых объектов и игровых персонажей. Данный алгоритм позволяет настраивать размер пространства, разницу высот, варианты разной погоды или климата. Алгоритм шума Перлина можно применять в одномерных, двумерный и трехмерных пространствах и возможно добавить дополнительное временное измерение для того, чтобы изменять текстуры с течением времени.

2. Реализация

Реализация алгоритма двоичного разбиения пространства состоит из следующий шагов:

1. Разделение начальной области на конечное число частей. Данный метод выполняется рекурсивно, на каждой итерации выбирается направление разрезания: вертикальное, если ширина пространства много больше длины и горизонтальное, если длина пространства много больше ширины, иначе направление выбирается случайным образом. Алгоритм продолжается до тех пор, пока длина или ширина разрезанного пространства не достигнут заданного минимального. В результате работы метода получается двоичное дерево, в листовых узлах которого хранятся готовые для генерации в них комнат подпространства [2].
2. Генерация комнат внутри получившихся на предыдущем шаге подпространств. Начальная точка, длина и ширина комнаты генерируются случайным образом так, чтобы размер комнаты получился больше или равен заданному минимальному.

   Данный метод также выполняется рекурсивно. В конце работы данного алгоритма получаем конечное число непересекающихся комнат.
3. Соединение комнат коридорами. Для этого сначала создаются коридоры между теми комнатами, которые находятся в листовых узлах, имеющих одного родителя в

двоичном дереве. Начальная точка коридора генерируется случайным образом так, чтобы из обеих комнат был проход в соседнюю. Далее на этом шаге коридорами соединяются те комнаты, которые до этого не были соединены. Для этого используется дополнительные методы для поиска таких комнат и для нахождения ближайших к ним комнат, с которыми они будут соединены коридором. После этого так же случайным образом генерируется начальная точка коридора. На рисунке 1 представлен пример работы алгоритма двоичного разбиения пространства.

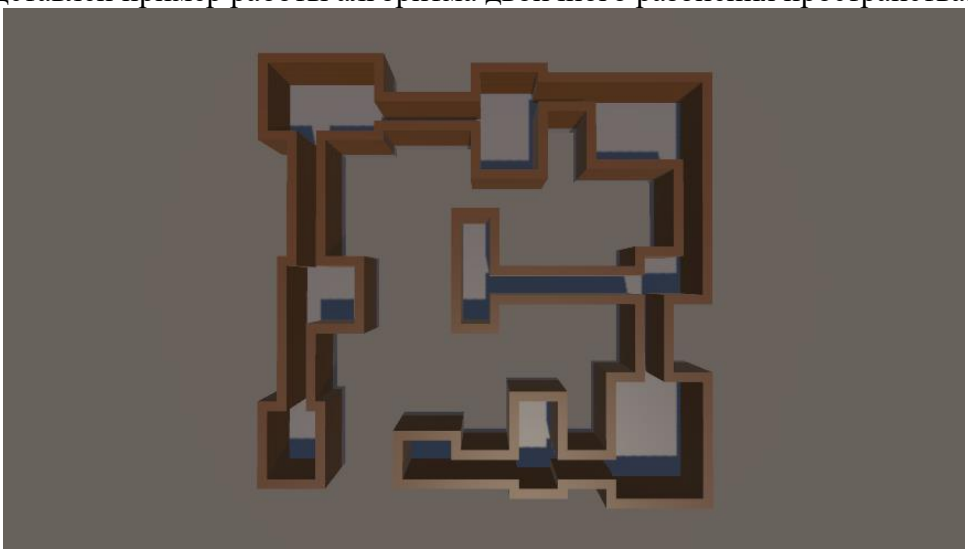


Рис. 1 Пример готовой карты, сгенерированной с помощью BSP-деревьев

Реализация алгоритма туннелирования состоит из следующих шагов:

1. Генерация комнаты. Случайным образом генерируется число от 1 до заданного максимального количества комнат, из которых может состоять составная комната. На основе этого числа генерируется данное количество комнат, также для каждой комнаты случайным образом генерируется ее форма и размер. Комнаты могут как накладываться друг на друга, так и соприкасаться.
2. Случайная генерация положения комнаты в пространстве. Необходимо сгенерировать координату левого верхнего угла комнаты так, чтобы новая комната не пересекалась с уже сгенерированными комнатами.
3. Вычисление минимальной стоимости пути. Для этого применяется алгоритм поиска A^*. Также стоимость единицы расстояния, находящиеся у стены и в центре комнаты задаются заранее.
4. Отрисовка недостающего коридора между данной комнатой и комнатой, сгенерированной на предыдущем шаге на основе минимального вычисленного пути. На рисунке 2 представлен пример работы алгоритма туннелирования.

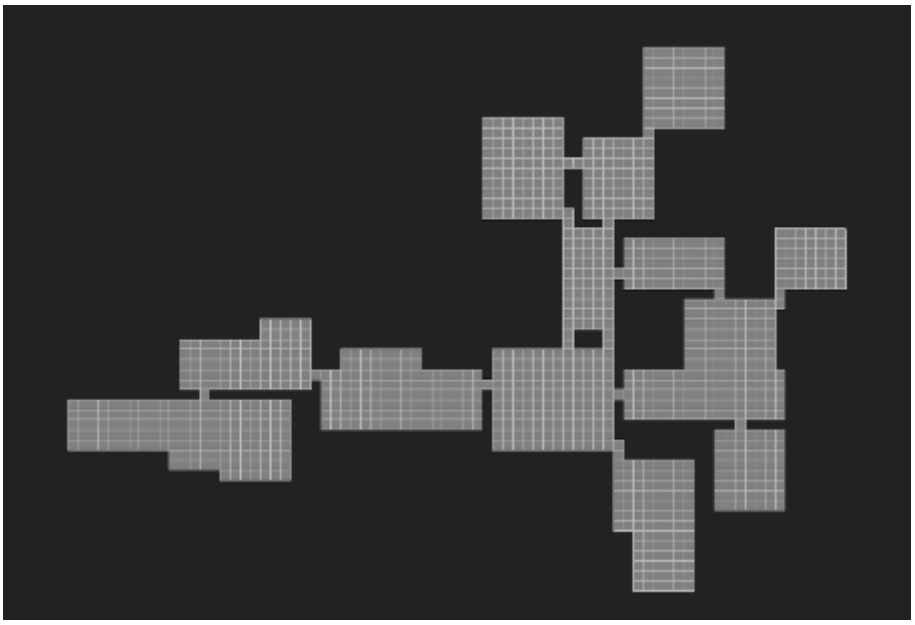


Рис. 2 Пример готовой карты, сгенерированной с помощью алгоритма туннелирования

Реализация алгоритма шума Перлина состоит из следующих шагов:

1. Создание сетки. Сначала создается сетка узлов или градиентов, эти узлы образуют основу для генерации шума.
2. Генерация случайный векторов. Для каждого узла сетки генерируется случайный вектор, который представляет собой направление изменения интенсивности шума.
3. Интерполяция значений. Для заданной точки в пространстве определяются ближайшие узлы сетки и вычисляется их воздействие на данную точку.
4. Смешивание значений. В результате этого шага путем смешивания значений, полученных из различных узлов сетки, получаем окончательное значение шума для данной точки. На рисунке 2 представлен пример работы алгоритма шума Перлина.

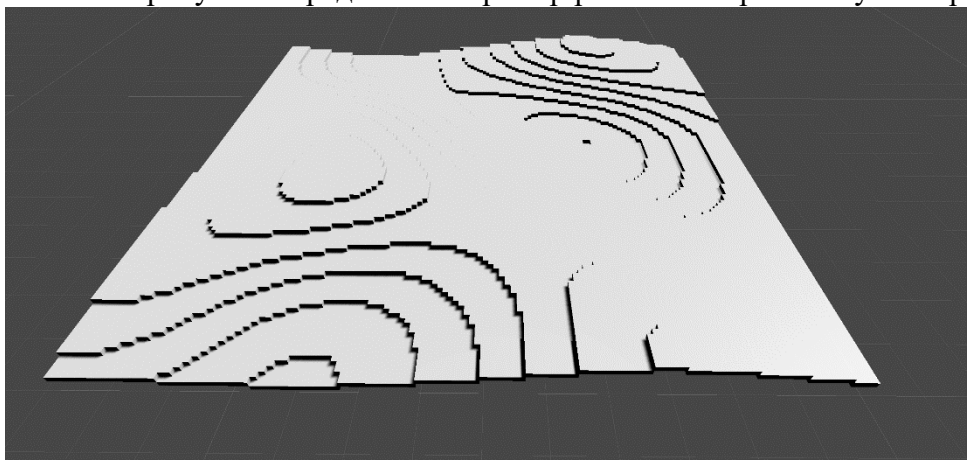


Рис. 3 Пример готовой карты, сгенерированной с помощью алгоритма шума Перлина

Заключение

В данной статье был приведен анализ обзор трех алгоритмов процедурной генерации карт. Можно сделать вывод о том, что каждый алгоритм выполняет свою конкретную задачу, причем все способы гибко настраиваются. Алгоритм двоичного разбиения пространства подойдет для реализации этажа с комнатами и коридорами, алгоритм туннелирования

подходит для реализации пещерообразных запутанных карт, алгоритм шума Перлина подходит для реализации открытой местности. Также были рассмотрены реализации каждого из способов и приведены примеры готовых карт.

Литература

1. Боресков А. Программирование компьютерной графики. Современный OpenGL / А. Боресков. — Москва : ДМК Пресс, 2019. — 372 с.
2. Берг де М. Вычислительная геометрия. Алгоритмы и приложения / М. де Берг, О. Чеонг, М. ван Кревельд, М. Овермаре. — Москва : ДМК Пресс, 2017. — 438 с.
3. Фар М. GPU Gems 2 / М. Фар, Р. Фернандо. — Бостон : Addison-Wesley, 2005. — 814 с.
4. Шелл Д. Искусство геймдизайна / Д. Шелл. — Бока-Ратон : CRC Press, 2008. — 520 с.

УДК 519.854.2

РЕШЕНИЕ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТНЫХ СРЕДСТВ С ЧЕРЕДУЮЩИМИСЯ ОБЪЕКТАМИ С ИСПОЛЬЗОВАНИЕМ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

А.Г. Трифонов

Воронежский государственный университет

Введение

Несмотря на активное применение информационных технологий, актуальность проблемы решения задачи маршрутизации транспорта не снижается с течением времени, поскольку практика выдвигает все более сложные задачи как по количеству оптимизируемых параметров, так и по количеству ограничений, учитываемых при ее решении.

Задачи маршрутизации автотранспорта являются ключевыми задачами в области логистики. Доставка товара в большинстве случаев добавляет к его стоимости сумму, сравнимую со стоимостью самого товара, поэтому поиск рационального решения в целях снижения транспортных затрат является актуальным. Задача маршрутизации транспорта – это типичная задача комбинаторной оптимизации, в которой необходимо определить набор маршрутов от парка транспортных средств до нескольких отдаленных точек-потребителей.

В данной статье будет рассмотрена задача маршрутизации транспортных средств с несколькими центрами и чередованием объектов двух типов, для решения которой будет использоваться генетический алгоритм.

1. Постановка задачи

Имеется совокупность неподвижных объектов, которые делятся на два типа: целевые объекта и центры. Расстояния между всеми объектами известны. Заданное количество транспортных средств должно в совокупности посетить все целевые объекты таким образом, чтобы минимизировать разброс длин результирующих маршрутов транспортных средств. Причем целевые объекты и центры должны чередоваться в их маршрутах. Каждый целевой объект можно посетить только один раз, а любой центр можно посетить неограниченное количество раз. При этом все транспортные средства начинают и заканчивают свой маршрут в единой точке сбора.

Содержательная постановка задачи может быть представлена как погрузка стогов сена с поля в грузовые машины сельскохозяйственной техникой [1].

2. Математическая модель

Введем следующие обозначения:

m — количество целевых объектов;

n — количество центров;

p — количество транспортных средств;

$(c_{ij})_{n \times m}$ — матрица, задающая затраты для перемещения между $i-$м центром и $j-$м

целевым объектом.

Введем переменные:

$x_{ijk}^{t} \in \{0,1\}, \quad i = 1,...,n, \quad j = 1,...,m, \quad k = 1,...,n, t = 1....,p,$ причем $x_{ijk}^{t} = 1,$ если $t-$ое транспортное средство перемещается от $i-$го центра к $j-$му целевому объекту и перемещается от него к $k-$му центру, и $x_{ijk}^{t} = 0,$ в противном случае [2].

Учитывая введенные переменные, суммарные затраты для перемещения между $i-$м центром, $j-$м целевым объектом и $k-$м центром, будут задаваться матрицей $(c_{ij})_{n \times m \times n}$ с элементами вида

$$c_{ijk} = c_{ij} + c_{kj}, \quad i = 1,...,n, \quad j = 1,...,m, \quad k = 1,...,n.$$

Место сбора представляется, как совокупность фиктивного целевого объекта и центра с индексами $j = m + 1$ и $i, \ k = 0$ соответственно.

В рамках заданных обозначений целевая функция для поставленной задачи примет следующий вид:

$$\max_{t=1,...,p} \left\{ \sum_{i=o}^{n} \sum_{j=1}^{m+1} \sum_{k=0}^{n} c_{ijk} x_{ijk}^{t} \right\} \to \min \qquad (1)$$

Она отвечает за минимизацию расхождений в длинах маршрутов всех транспортных средств.

Ограничения данной задачи были рассмотрены в статье [2].

3. Алгоритм решения

Для решения задачи, описанной выше, будет использован генетический алгоритм. Генетический алгоритм в общем виде состоит из следующих основных блоков:

1. Формирование начальной популяции;

2. Селекция хромосом с наилучшим значением целевой функции;

3. Скрещивание хромосом-родителей с целью получения новые хромосом-потомков;

4. Мутация хромосом.

5. Остановы.

Рассмотрим подробнее каждый из них применительно к поставленной задаче.

1.1. Формирование начальной популяции

Рассмотрим вид хромосомы для поставленной задачи.

Гены, стоящие на четных местах, будут соответствовать целевым объектам, а на нечетных – центрам. В каждой хромосоме будут присутствовать маршруты всех транспортных средств. Разделять отдельные маршруты будут гены, соответствующие депо (рис. 1).

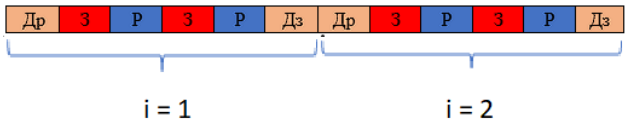


Рис. 1. Представление хромосомы (Д – депо, З – загрузка (целевой объект), Р – разгрузка (центр), i – индекс транспортного средства)

Депо представляется двумя генами, что необходимо для сохранения чётности внутренних элементов маршрутов. Исключение составляют начало и конец хромосомы.

С целью равномерного распределения целевых объектов и центров между транспортными средствами, в каждый маршрут отбирается $\left[m/p \right]$ пар: целевой объект и центр.

1.2. Селекция хромосом

Процедуру селекции будет осуществляться методом ранжирования. Он заключается в выборе особей с наибольшим значением функции приспособленности.

Для рассматриваемой задачи в качестве значения функции приспособленности хромосомы будет рассматриваться величина, обратная наибольшей из длин маршрутов, входящих в текущую хромосому. А именно

$$f_s = \frac{1}{L_s},$$

где s – индекс особи в популяции, а величина L_s вычисляется по формуле

$$L_s = \max_p \left\{ \sum_{i=0}^{n} \sum_{j=1}^{m+1} \sum_{k=0}^{n} c_{ijk} x_{ijk}^p \right\}.$$

1.3. Скрещивание

Для реализации скрещивания особей используется модифицированный равномерный кроссовер. Идея равномерного кроссовера заключается в случайном выборе каждого гена для потомка у одного из его родителей.

Отдельно следует рассмотреть ситуацию, когда на текущем месте у одного из родителей содержится ген, соответствующий одной из частей депо. Такие пары генов разъединять недопустимо. Для гарантированного выполнения этого условия при попадании на ген, содержащий депо, предлагается осуществлять выбор между парами генов родителей. При следующем появлении у одного из родителей гена, отвечающего депо, пару генов для потомка необходимо взять у того же родителя. Это связано с тем, что количество генов, отвечающих за депо, во всех хромосомах должно быть одинаково.

На рисунке 1 проиллюстрирован пример работы алгоритма скрещивания.

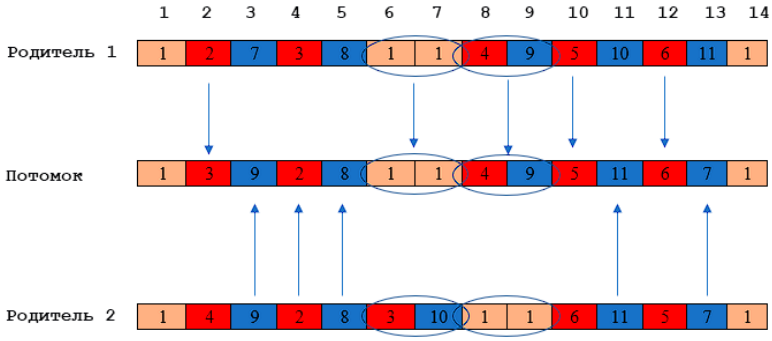


Рис. 1. Иллюстрация работы оператора скрещивания

Здесь гены потомка с индексами 1 и 14 по умолчанию отвечают за депо. Гены с индексами со 2 по 5 и с 10 по 13 случайным образом выбираются из хромосом родителей. В гене с индексом 6 у первого родителя встречается депо, поэтому происходит случайный выбор из пары генов с индексами 5 и 6. При следующей встрече у родителей гена, соответствующего депо, пара генов берётся у того же родителя.

1.4. Мутация

В соответствие с принципом работы генетического алгоритма каждая особь с определенной вероятностью должна подвергаться мутации.

В связи с особенностями построения хромосомы для рассматриваемой задачи для достижения максимального многообразия особей популяции необходимо предусмотреть следующие варианты операторов мутации:

1. Случайным образом выбираются два гена одного типа и меняются местами. Данный оператор необходим, чтобы менять объекты внутри пар вида: целевой объект и центр (рис. 2).

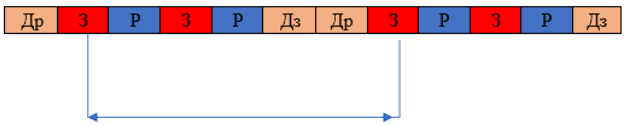


Рис. 2. Иллюстрация работы первого оператора мутации

2. Случайным образом выбираются две пары подряд идущих генов и меняются местами. Этот оператор позволит менять местоположение депо (рис. 3).

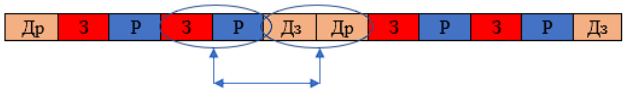


Рис. 3. Иллюстрация работы второго оператора мутации

3. Случайным образом выбирается ген, содержащий центр, и его значение случайным образом меняется на другой допустимый номер центра (рис. 4).

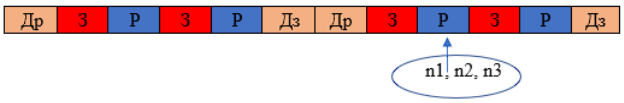


Рис. 4. Иллюстрация работы третьего оператора мутации

1.5. Остановы

Основные этапы генетического алгоритма повторяются, пока не будет достигнуто условие останова. В качестве остановов можно использовать ограничение по числу итераций и длительное отсутствие прогресса.

4. Пример работы предложенного алгоритма

Для исследования предложенного алгоритма на языке python была разработана его программная реализация.

На входе программа получает следующие данные: координаты и типы всех точек, количество особей в популяции, количество транспортных средств, процент мутации, количество итераций алгоритма.

Рассмотрим пример работы алгоритма на следующих входных данных: вероятность мутации - 30%, количество тракторов - 3, количество особей в популяции - 10, координаты и типы точек заданы в таблице 1.

Таблица 1

Вводные данные

Индекс	**X**	**Y**	**Тип**
0	5	5	стог
1	10	5	стог
2	25	10	стог
3	100	100	стог
4	10	25	стог
5	10	40	стог
6	5	100	стог
7	15	100	стог
8	15	90	стог
9	85	90	стог
10	20	100	стог
11	100	10	стог

12	90	25	стог
13	90	5	стог
14	10	55	машина
15	85	100	машина
16	100	30	машина
17	55	50	депо

На рисунке 5 показано графическое представление полученных маршрутов.

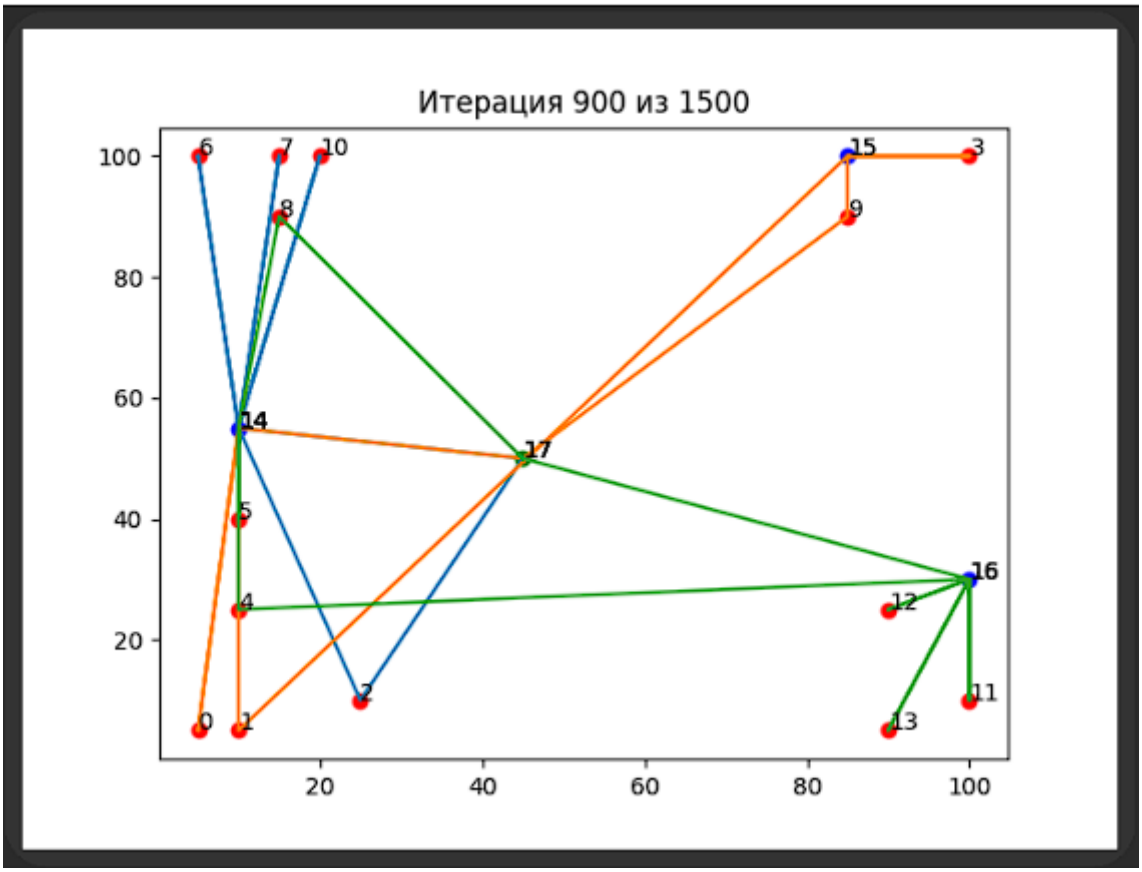


Рис. 5. Результат работы генетического алгоритма

Таким образом, в результате работы алгоритма транспортные средства получили следующие маршруты, представленные в таблице 2.

Таблица 2

Результирующие маршруты транспортных средств

№	Маршрут	Длина пути	Отклонение от среднего
1	17-2-14-10-14-6-14-7-14-17	400,81	1%
2	17-9-15-3-15-1-14-0-14-17	403,46	0,3%
3	17-8-14-5-14-4-16-12-16-13-16-11-16-17	410,23	1,4%

Как видно из таблицы 2, для транспортных средств получены маршруты, которые имеют длины, отличающиеся от среднего значения не более, чем на 1,5%

Заключение

В статье рассматривается задача маршрутизации транспортных средств с чередованием объектов двух видов. В качестве целевой функции рассматривается минимизация разброса длин результирующих маршрутов транспортных средств. Для решения поставленной задачи предложен генетический алгоритм, который был программно реализован и показал свою работоспособность.

Литература

1. Domínguez-Martín, B. The driver and vehicle routing problem / B. Domínguez-Martín, I. Rodríguez-Martín, J.-J. Salazar-González // Computers & Operations Research. – 2018. – V. 92. – P. 56–64. DOI: 10.1016/j.cor.2017.12.010.

2. Медведев, С. Н. Математическая модель и алгоритм решения задачи маршрутизации транспортных средств с несколькими центрами с чередованием и единым местом сбора / С. Н. Медведев // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – Воронеж: Воронеж. гос. ун-т., 2021. – № 1. – С. 21- 32. DOI: https://doi.org/10.17308/sait.2021.1/3368.

3. Medvedev S. N. (2019) Greedy and adaptive algorithms for multi-depot vehicle routing with object alternation // Automation and Remote Control. 84(3). P. 341–364. Available from: DOI: 10.25728/arcRAS.2023.81.72.001.

4. Корбут, А. А. Дискретное программирование / А. А. Корбут, Ю. Ю. Финкельштейн [под ред. Д. Б. Юдина]. – Москва : Наука, 1969. – 368 с.

5. В. А. Бойков. "О применении жадных алгоритмов в некоторых задачах дискретной математики" Программные продукты и системы, vol. 32, no. 1, 2019, pp. 55-62. doi:10.15827/0236-235X.125.055-062

УДК 004.42

РАСПРЕДЕЛЕННОЕ КОРПОРАТИВНОЕ ФАЙЛОВОЕ ХРАНИЛИЩЕ

В. С. Тройнин

Воронежский государственный университет

Введение

В любом предприятии или организации, в том числе в небольших, возникает проблема безопасного и надёжного хранения файлов. Эта задача может решаться разными способами, однако все они являются либо организационно, либо технически сложными. Далее приведено несколько примеров подобных решений:

1. Использовать несколько виртуальных дисков внешних компаний. Это сложно организационно, так как кто-то должен их вручную заполнять и администрировать.
2. Создать локальную сеть, добавить сетевой диск и настроить его репликацию. Такой подход требует дополнительных затрат на приобретение дополнительного оборудования.
3. Использовать Apache Cassandra, но это довольно тяжеловесное и сложное в эксплуатации программное решение.

Однако, можно ли создать достаточно простое в использовании, и при этом устойчивое решение для безопасного хранения файлов? Да, если веб-приложение будет выстроено на основе *Blackboard-архитектуры* и алгоритма *Consistent hashing ring*, а также оно использует алгоритмы шифрования данных. [3–5, 8]

1. Постановка задачи

Необходимо спроектировать и реализовать веб-приложение на основе *Blackboard-архитектуры*, алгоритма *Consistent hashing ring* и с использованием алгоритмов шифрования данных, чтобы безопасно хранить файлы в виде децентрализованного хранилища данных из множества пользователей системы. [3–5, 8]

2. Взаимодействие с пользователями

Приложение работает с пользователями, которые имеют одну из двух ролей — *участник* или *администратор*.

2.1. Функциональные возможности участника

Участник имеет следующий набор возможностей:

1. Авторизация.
2. Получение файла по имени.
3. Получение списка имен всех собственных файлов.
4. Сохранение файла.
5. Удаление файла.

Взаимодействие участника с приложением осуществляется посредством *RESTful API*. [7]

2.2. Функциональные возможности администратора

Администратор обладает следующими функциональными возможностями:

1. Авторизация.
2. Добавление пользователя.
3. Смена пароля пользователя.
4. Удаление пользователя.
5. Получение статистики по пользователю.

Взаимодействие администратора с приложением осуществляется посредством *RESTful API*. [7]

3. Архитектура приложения

Приложение состоит из двух основных компонентов — *узла* (условная серверная часть) и *поставщиков* (условная клиентская часть). Взаимодействие между ними осуществляется посредством протокола *WebSocket*, реализуя архитектурный паттерн *Blackboard*. Сами компоненты реализованы на платформе *Spring Boot 2*, основанной на *Spring Framework 5*. [1, 4, 9–10, 12]

Пользователи обращаются к компонентам посредством *RESTful API*, безопасность подключения обеспечивается благодаря протоколу *https* и *Spring Security*. [1, 7, 9]

В качестве СУБД для *узла* была использована *PostgreSQL*, как бесплатная и открытая СУБД «по умолчанию». [6]

В качестве СУБД для *поставщика* была использована *SQLite*, так как это СУБД запускается не как отдельный сервис, а как библиотека основного приложения, что позволяет упростить установку и использование клиентской части приложения. Она также является бесплатной и открытой. [11]

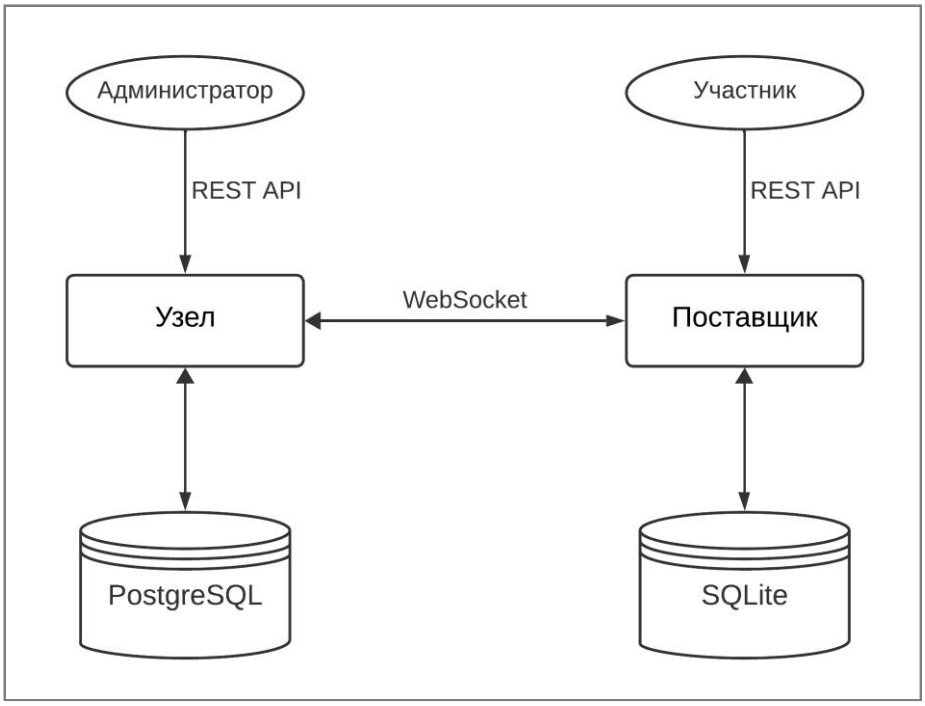


Рис. 1. Общая архитектура веб-приложения

4. Модель хранения данных

Хранение данных осуществляется посредством алгоритма *consistent hashing ring*. Данный алгоритм позволяет распределённо хранить данные в произвольном количество хранилищ, количество которых может изменяться в процессе работы приложения. [3, 4]

В файловых хранилищах поставщиков данные хранятся в зашифрованном виде. В качестве алгоритма блочного симметричного шифрования используется *AES*. [5]

Электронные подписи создаются с помощью алгоритма создания асимметричных подписей *RS256*. [8]

4.1. Структура хранимого блока данных

Данные хранятся в виде блока хранения, именуемого *storageBlock*, который состоит из четырёх элементов:

1. *dataBlock* — массив, являющийся зашифрованным блоком данных исходного файла.
2. *infoBlock* — строка, являющаяся зашифрованным *JSON*-объектом, состоящим из следующих полей:
   1) *userName* — строка, являющаяся именем пользователя, которому принадлежит файл;
   2) *fileName* — строка, являющаяся именем исходного файла;
   3) *blockNumber* — число, являющееся номером блока данных в исходном файле.
3. *infoSignature* — строка-подпись *infoBlock*.
4. *storageSignature* — строка-подпись *infoBlock* + *dataBlock*.

4.2. Алгоритм получения данных пользователем

Алгоритм получения данных поставщиком работает следующим образом: [2, 3]

1. Поставщик вычисляет идентификатор *storageBlock* как хеш-сумму от зашифрованного с помощью секретного ключа *infoBlock*.
2. Запрос с вышеописанным идентификатором и публичным ключом пользователя передаётся на узел.
3. Узел вычисляет адрес *storageBlock* как хеш-сумму от идентификатора *storageBlock* и номера его репликации.
4. Узел осуществляет поиск активного поставщика с наибольшим адресом, не превышающим рассчитанный адрес *storageBlock*, до тех пор, пока не найдёт первый корректный *storageBlock* посредством проверки подписи *storageSignature* с помощью переданного клиентом публичного ключа.
5. Узел передаёт полученный *storageBlock* ожидающему поставщику.

Поставщик повторяет вышеперечисленные шаги до тех пор, пока не соберет все *storageBlock* искомого файла. Затем требуется расшифровать блоки с помощью имеющегося секретного ключа и расположить их в правильной последовательности. [5]

Если требуется получить от поставщиков только *infoBlock*, то достаточно осуществлять проверку *infoSignature* вместо *storageSignature*.

4.3. Алгоритм сохранения данных пользователем

Алгоритм сохранения данных поставщиком работает следующим образом: [2, 3]

1. Поставщик делит файл на равные блоки (например, 1 Мб) и шифрует их с помощью своего секретного ключа.
2. Поставщик для каждого *dataBlock* создаёт *storageBlock* и по очереди передаёт их на узел.

3. Для каждого полученного *storageBlock* узел высчитывает адрес как хеш-сумму от *infoBlock* и номера репликации *storageBlock.*
4. Узел осуществляет поиск активного поставщика с наибольшим адресом, не превышающим рассчитанный адрес блока, и затем передаёт этому поставщику *storageBlock.*

Вышеперечисленные пункты повторяются до тех пор, пока поставщик не сохранит все блоки файла.

4.4. Алгоритм разделения данных при добавлении нового поставщика

Алгоритм разделения данных между поставщиками при добавлении нового выглядит следующим образом: [3]

1. Осуществляется поиск поставщика с наибольшим адресом, не превышающим адрес нового поставщика.
2. У исходного поставщика запрашиваются все *storageBlock,* чей адрес превышает адрес нового поставщика, и передаются новому поставщику для сохранения.

4.5. Алгоритм переливания данных при потере одного из поставщиков

Алгоритм переливания данных в оставшегося поставщика при потере одного из других поставщиков выглядит следующим образом: [3]

1. Узел осуществляет поиск поставщика с наибольшим адресом, не превышающим адрес выбывшего поставщика.
2. Узел проводит ревизию всех реплик блоков по всем оставшимся поставщикам, то есть считает количество реплик для каждого уникального *storageBlock.*
3. Те блоки, у которых количество ревизий меньше их требуемого количества, передаются на сохранение оставшемуся поставщику, найденному в 1-м пункте.

5. Программный интерфейс пользователей

5.1. API пользователя «участник»

В табл. 1 представлен перечень основных эндпоинтов и набор поддерживаемых для них *HTTP*-методов для пользователя с ролью «участник».

Таблица 1.

Краткая информация о поддерживаемом REST API для участника

Эндопоинт	**GET**	**POST**	**PUT**	**DELETE**
files	+	+	–	+
/files/names	+	–	–	–

5.2. API пользователя «администратор»

В табл. 2 представлен перечень основных эндпоинтов и набор поддерживаемых для них *HTTP*-методов для пользователя с ролью «администратор».

Таблица 2.

Краткая информация о поддерживаемом REST API для администратора

Эндопоинт	**GET**	**POST**	**PUT**	**DELETE**

/users	+	+	+	+
/user/statistics	+	–	–	–

Заключение

Было спроектировано и реализовано веб-приложение на основе *Blackboard-архитектуры*, алгоритма *Consistent hashing ring* и с использованием алгоритмов шифрования данных, которое способно безопасно хранить файлы в виде децентрализованного хранилища данных из множества пользователей системы.

Мельников Вадим Митрофанович (научный руководитель) — старший преподаватель кафедры ПОиАИС Воронежского государственного университета.

Литература

1. Гутьеррес Ф. Spring Boot 2: лучшие практики для профессионалов / Ф. Гутьеррес – СПб.: Питер, 2020. – 464 с.
2. Тройнин В. С. Распределенное сетевое файловое хранилище / В. С. Тройнин // Математика, информационные технологии, приложения: сборник трудов межвузовский научной конференции молодых ученых и студентов. – 2023. – С. 474–477.
3. A guide to consistent hashing [Электронный ресурс]. Режим доступа: https://www.toptal.com/big-data/consistent-hashing. – (Дата обращения: 12.04.2024).
4. Blackboard architecture: Documentation [Электронный ресурс]. Режим доступа: https://social.technet.microsoft.com/wiki/contents/articles/13215.blackboard-design-pattern.aspx. – (Дата обращения: 12.04.2024).
5. Java AES encryption and decryption [Электронный ресурс]. Режим доступа: https://www.baeldung.com/java-aes-encryption-decryption. – (Дата обращения: 12.04.2024).
6. PostgreSQL: Documentation [Электронный ресурс]. Режим доступа: https://www.postgresql.org/docs. – (Дата обращения: 12.04.2024).
7. RESTful API: Documentation [Электронный ресурс]. Режим доступа: https://restfulapi.net. – (Дата обращения: 12.04.2024).
8. RSA in Java [Электронный ресурс]. Режим доступа: https://www.baeldung.com/java-rsa. – (Дата обращения: 12.04.2024).
9. Spring: Documentation [Электронный ресурс]. Режим доступа: https://docs.spring.io/spring-framework/docs/current/reference/html. – (Дата обращения: 12.04.2024).
10. Spring 5 для профессионалов / Ю. Козмина, Р. Харроп, К. Шефер, Х. Кларенс — СПб.: ООО «Диалектика», 2020 — 1120 с.
11. SQLite: Documentation [Электронный ресурс]. Режим доступа: https://www.sqlite.org/docs.html. – (Дата обращения: 12.04.2024).
12. WebSockets: Documentation [Электронный ресурс]. Режим доступа: https://docs.spring.io/spring-framework/reference/web/websocket.html. – (Дата обращения: 12.04.2024).

УДК 004

Серверное приложение по парсингу данных
с маркетплейсов на основе Django Rest Framework

Тупикин Е. И.

Воронежский государственный университет

Введение

При проведении парсинга данных с маркетплейсов используются специальные технологии и программные решения. Серверное приложение по парсингу данных с маркетплейсов предоставляет эффективное решение для автоматизации процесса сбора и обработки информации. Оно обеспечивает централизованный доступ к данным, что позволяет бизнесу быстро получить актуальную информацию и анализировать ее для выработки стратегических решений.

В настоящее время имеются решения для получения данных из разных маркетплейсов: Upwork, Freelancer, Fiverr, DataParser. Их особенность в том, что они имеют ограниченный набор инструментов, например, чаще всего умеют парсить только легкие сайты без защиты.

В статье рассматриваются некоторые аспекты разработки серверного приложения для парсинга данных с маркетплейсов, которая позволит бизнесам автоматизировать и упростить процесс сбора и анализа информации о товарах и конкурентных предложениях.

1.1 Технические требования

Целью работы является создание серверного приложения, позволяющего получать данные с различных маркетплейсов и выдавать пользователю в легко читаемом формате (JSON).

Перечень функциональных требований к приложению:

1) возможность авторизации и аутентификация в системе;

2) возможность парсить цены товаров с маркетплейсов;

3) возможность парсить характеристики товаров с маркетплейсов;

3) возможность получения данных о том, сколько раз были собраны данные с тех или иных сайтов;

4) возможность менять статус задачи в зависимости от того, на каком этапе находится задача парсинга;

5) возможность парсить данные асинхронно, чтобы пользователь не ждал в реальном времени окончания решения задачи.

1.2 Обзор существующих решений

Разработан ряд решений по парсингу данных в интернете, приведем некоторые из них.

1) Парсинг поисковой выдачи. Данный метод заключается в извлечении информации из результатов поиска веб-страниц, связанных с товарами или услугами. Алгоритм включает в себя следующие шаги:

- Отправка запроса на поиск по заданным ключевым словам.
- Анализ полученных результатов, выделение сниппетов или блоков, содержащих нужные данные.
- Извлечение и обработка данных, например, удаление символов валюты или форматирования.

2) Использование регулярных выражений. Данный алгоритм основан на использовании регулярных выражений для поиска и извлечения даныъ информации из текста. Шаги алгоритма включают:

-Определение шаблона регулярного выражения, соответствующего формату данных (например, десятичное число с символом валюты).

-Применение регулярного выражения к тексту и поиск соответствующего формата.

-Извлечение найденных данных и их обработка (например, преобразование в числовой формат).

3) Анализ структуры HTML-страниц. Данный метод основан на анализе структуры HTML-страниц для определения местоположения и извлечения данных. Шаги алгоритма включают:

-Загрузка и разбор HTML-страницы с использованием библиотеки парсинга HTML.

-Исследование структуры страницы и определение характерных признаков, указывающих на наличие нужной информации (например, классы CSS или теги HTML).

-Извлечение данных из соответствующих элементов страницы.

4) API парсинга. Существуют сторонние сервисы и инструменты, предоставляющие API для парсинга из различных источников, таких как веб-страницы или онлайн-торговые площадки. Алгоритм включает в себя следующие шаги:

-Получение доступа к API сервиса парсинга цен.

-Определение параметров запроса (например, URL веб-страницы или идентификатор товара).

-Отправка запроса на API и получение ответа, содержащего ценовую информацию.

-Обработка и сохранение полученных данных.

1.3 Описание проблемы и ее решения

Исходя из вышеизложенных решений, которые позволяют получать данные из интернета, для решения задачи парсинга данных из маркетплейсов был выбран самый оптимальный метод – API парсинг.

Этот метод является эффективным решением для получения и анализа данных о ценах, как с точки зрения удобства использования, так и с точки зрения надежности результатов.

Метод API парсинга отличается простотой в реализации и использовании. Он позволяет получать информацию о ценах из внешних источников путем отправки запросов API и получения структурированных данных. Такой подход обеспечивает быстрое получение актуальных данных о ценах и удобное их сравнение.

Метод также обладает высокой масштабируемостью, что позволяет обрабатывать большие объемы данных о ценах. Благодаря использованию стандартизированных форматов данных, таких как JSON или XML, обеспечивается надежность и точность полученной информации.

Заключение

На основе разработанной системы для парсинга цен и характеристик с маркетплейсов были достигнуты следующие результаты:

1) Создана возможность авторизации и аутентификация в системе;

2) Создана возможность парсить цены товаров с маркетплейсов

3) Создана возможность парсить характеристики товаров с маркетплейсов

3) Создана возможность получения данных о том, сколько раз были собраны данные с тех или иных сайтов

4) Создана возможность менять статус задачи, в зависимости от того, на каком этапе находится задача парсинга

5) Создана возможность парсить данные асинхронно, чтобы пользователь не ждал в реальном времени окончания решения задачи

Литература

1. Фримен Эрик. Head First. Паттерны проектирования. 2-е издание / Фримен Эрик. – СПб: Питер, 2023. – 640 с.

2. Лусиану Рамальо. PYTHON. К вершинам мастерства. Лаконичное и эффективное программирование / Лусиану Рамальо. – СПб: ДМК Пресс, 2021. – 898 с.

3. Яворски Михал. Python. Лучшие практики и инструменты. 4-е изд. / Яворски Михал. – СПб: Питер, 2024. – 592 с.

4. Celery: лучшие практики. – URL: https://habr.com/ru/articles/269347/ (дата обращения: 03.02.2024)

5. Полное практическое руководство по Docker: с нуля до кластера на AWS. – URL: https://habr.com/ru/articles/310460/ (дата обращения: 03.02.2024)

6. Art Yudin. Building Versatile Mobile Apps with Python and REST. / Art Yudin. – Springer Nature, 2024. – 366 с.

7. Gastón C. Hillar. Django RESTFul Web Services / Gastón C. Hillar. – Packt Publishing, 2023. – 326 с.

УДК 004.41

Разработка информационной системы мониторинга временных затрат сотрудников IT-компании

А. И. Турбабин

Воронежский государственный университет

Введение

В современном мире эффективное управление временем сотрудников играет ключевую роль в успехе организаций. Основываясь на этом, разработка информационных систем для мониторинга временных затрат и оценки эффективности сотрудников становится неотъемлемым элементом стратегии управления персоналом.

Целью данной работы является разработка информационной системы, которая позволит вести учет рабочего времени, а также провести необходимый анализ эффективности сотрудников. В ходе работы будут рассмотрены основные компоненты, технологии и методологии, необходимые для создания такой системы.

1. Функциональность информационной системы

Основной задачей информационной системы будет автоматизация учета времени работы сотрудников и разработка механизмов аналитики для оценки производительности, в соответствии с этим разрабатываемое приложение должно иметь следующий функционал:

1. Учет временных затрат:
    1.1. Возможность отмечать начало и окончание работы над задачей;
    1.2. Ввод временных затрат вручную.
2. Оценка эффективности сотрудников:
    2.1. Сравнение плановых и фактических временных затрат;
    2.2. Расчет KPI (ключевых показателей эффективности) сотрудника.
3. Отчетность и мониторинг:
    3.1. Создание отчетов о затраченном времени и эффективности;
    3.2. Мониторинг временных затрат проектов, отделов, компании.

Таким образом, мы можем декомпозировать нашу изначальную задачу на три меньших, с которыми нам будет удобнее работать. Построим их при помощи графической модели idef0. На рис. 1 представлена общая, контекстная диаграмма.

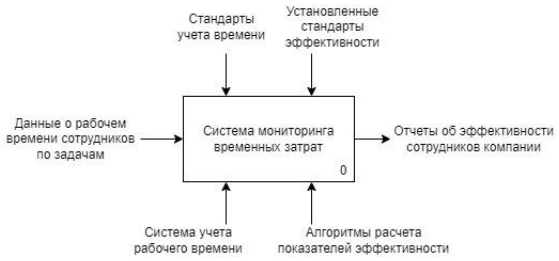


Рис. 1. Контекстная диаграмма.

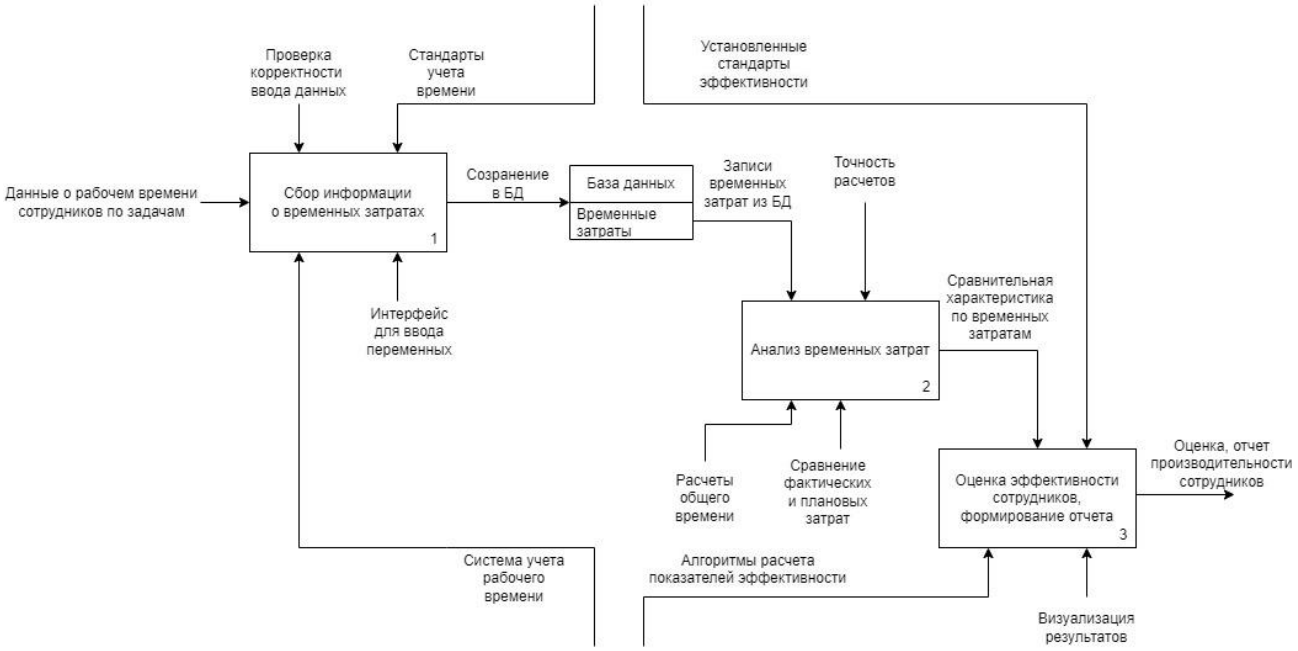


Рис.2. Диаграмма декомпозиции A0.

На рис. 2 мы видим уже меньшие задачи, на которые была разбита изначальная. Здесь, помимо входных и выходных данных, изображены и средства контроля, а также механизмы, с помощью которых будет получен необходимый результат. Правильно подобрав средства разработки, можно достичь наилучших результатов.

2. Компоненты системы

При реализации проекта будут разработаны следующие компоненты:

Интерфейс пользователя (UI). Пользователи могут взаимодействовать с системой через веб-интерфейс, где будет доступен функционал по управлению задачами, вводу времени и просмотру статистики.

Пользовательский интерфейс будет разработан при помощи средств ReactJS, язык гипертекстовой разметки HTML и CSS.

Серверная часть.

- База данных (использование PostgreSQL):
  Хранение информации о задачах, временных затратах и сотрудниках.
- API Layer:
  RESTful API для взаимодействия между клиентской частью и сервером.
  Реализация на Python (с использованием Django).
- Бизнес-логика:
  Обработка данных о времени, задачах и оценка эффективности.
  Механизм расчета KPI и других метрик эффективности.

Аналитика и отчетность. Модуль аналитики **о**беспечивает анализ временных затрат сотрудников, а также визуализацию данных в виде графиков, диаграмм и отчетов, для получения точной, а также доступной для восприятия информации.

В качестве инструментов для аналитики будут использоваться библиотеки Python, используемые повсеместно в аналитике: Pandas - для подготовки данных, NumPy - для углублённых расчётов, MatPlotLib - для визуализации.

3. Оценка эффективности сотрудника

Ключевые показатели эффективности (KPI) сотрудников представляют собой метрики, которые оценивают достижение определенных целей и результатов работы каждого сотрудника в организации. KPI помогают измерить производительность и вклад сотрудников в достижение общих бизнес-целей компании. Эти показатели могут варьироваться в зависимости от специфики деятельности и роли сотрудника, однако основные принципы оценки остаются общими.

KPI должны быть конкретными, измеримыми, достижимыми, релевантными и своевременными (SMART). Они могут включать в себя такие аспекты как выполнение задач в срок, качество продукта или услуги, уровень обслуживания клиентов, уровень продаж, соблюдение бюджета или другие ключевые результаты, важные для конкретного подразделения или процесса.

Использование KPI позволяет определить, насколько успешно сотрудник выполняет свои обязанности и какие области требуют улучшения или дополнительного внимания. Кроме того, правильно подобранные и измеряемые KPI могут стать инструментом мотивации и стимулирования профессионального роста сотрудников.

Определение ключевых показателей эффективности (KPI) для сотрудника требует системного и аналитического подхода. Важно начать с понимания стратегических целей компании или организации. Затем необходимо проанализировать роль и ответственности сотрудника в контексте этих целей. Это позволит идентифицировать ключевые области, в которых необходимо измерять эффективность работы сотрудника. После определения роли сотрудника и его места в бизнес-процессах, следующим шагом будет анализ основных задач и функций, которые выполняет сотрудник. Важно понять, какие из этих задач критически влияют на достижение бизнес-целей организации.

Далее необходимо установить конкретные метрики или показатели, которые могут отражать эффективность выполнения этих задач. Ключевые показатели должны быть измеримыми, достижимыми, релевантными и ориентированными на результаты. Это могут быть, например, количество выполненных задач за определенный период, уровень качества или точности работы, степень удовлетворенности клиентов и другие факторы, зависящие от специфики деятельности сотрудника.

Критически важно также обеспечить согласованность и прозрачность при установлении KPI. Сотрудник должен понимать, какие результаты и метрики отражают его эффективность, и как их измерять. Важно также учитывать возможность корректировки KPI в зависимости от изменяющихся условий и целей организации.

Заключение

Результатом данной статьи стала модель информационной системы для мониторинга временных затрат и определения эффективности сотрудников. В заключении хотелось бы подчеркнуть важность создания вышеописанной системы для повышения эффективности и управления ресурсами в организации.

Литература

1. Блинкова, О.Н. Необходимость анализа производительности труда на предприятии / О.Н. Блинкова, О.Н. Ганюта // Синергия Наук. — 2019. — № 31. — С. 222-227.

2. Фримен Э., Робсон Э., Сьерра К., Бейтс Б. Head First. Паттерны проектирования. Обновленное юбилейное издание. – СПб.: Питер, 2018. – 656 с.

3. Баранов Ю.В. Актуальные проблемы в сфере оценки труда персонала / Ю.В. Баранов // Социально-трудовые исследования. — 2021. — № 1 (42). — С. 64-74.

УДК 004.852

ПРОГНОЗИРОВАНИЕ КУРСА ВАЛЮТ И АКЦИЙ С ИСПОЛЬЗОВАНИЕМ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ СТАТИСТИКИ

Д. С. Тырышкин

Воронежский государственный университет

Введение

Вместе с усовершенствованием технологий меняются и различные области человеческой деятельности, в частности — рынок ценных бумаг. Появляется все больше факторов, которые могут косвенно или напрямую повлиять на ситуацию на рынке, поэтому прогнозировать его изменение становится все сложнее, что создает спрос на новые механизмы прогнозирования курса ценных бумаг. Одним из наиболее перспективных направлений в этой области является применение машинного обучения, которое позволяет анализировать большие объемы данных и выявлять скрытые закономерности, которые тяжело учитывать в традиционном анализе.

Цель данного исследования заключается в изучении возможностей машинного обучения в прогнозировании курса ценных бумаг на основе общедоступной информации о статистических данные и новостей. Данное исследование разделено на две части: модель анализа статистических данных и модель анализа новостей, которые позднее будут объединены в одну модель. В данной части внимание будет сконцентрировано на анализе исторических данных, разработке модели машинного обучения и исследовании того, как влияет архитектура модели на качество прогноза.

1. Постановка задачи

Чтобы реализовать часть прогнозирования на данных статистики необходимо разработать алгоритм машинного обучения на основе рекуррентной нейронной сети. Поскольку данная модель будет соединена с моделью прогнозирования новостей требуется выбрать одинаковый временной интервал сбора данных, который подойдет обоим моделям. После сравнения результатов прогноза на разных интервалах сбора данных и просмотра новостей, которые подходят по критериям задачи, было решено установить интервал сбора данных величиной в один день.

Реализация модели прогнозирования статистики состоит из следующих этапов:

1. Сбор и анализ данных;
2. Анализ и подготовка данных для передачи в модель машинного обучения;
3. Разработка возможных архитектур нейросети;
4. Исследование изменения качества моделей при изменении параметров
5. Выбор наилучшей архитектуры и наилучших параметров модели.

2. Сбор и анализ данных

Для сбора данных было разработано несколько вариантов: основной и запасной. В качестве основного источника данных выступает библиотека moexalgo[1], которая позволяет собирать данные с Московской биржи. Период сбора данных составил 3 года (с 01.03.21 по

01.03.24), в качестве целевой акции была выбрана акция Сбербанка. В полученном наборе данных содержится 746 записи (без выходных дней, в которые не велась торговля на бирже). Каждая запись содержит следующую информацию:

- Цена на начало дня;
- Цена на окончание дня;
- Самая высокая цена за день;
- Самая низкая цена за день;
- Суммарная стоимость проданных за день акций;
- Количество проданных за день акций;
- Дата начала дня;
- Дата окончания дня.

Разрабатываемая модель будет предсказывать цену закрытия дня, поэтому подробнее проанализируем данное значение. Для визуализации данных была использована библиотека plotly[2]. На рис. 1 представлена диаграмма распределения цены закрытия дня, по данной диаграмме видно, что цены большинства акций находятся в диапазонах 105–160 и 230–330.

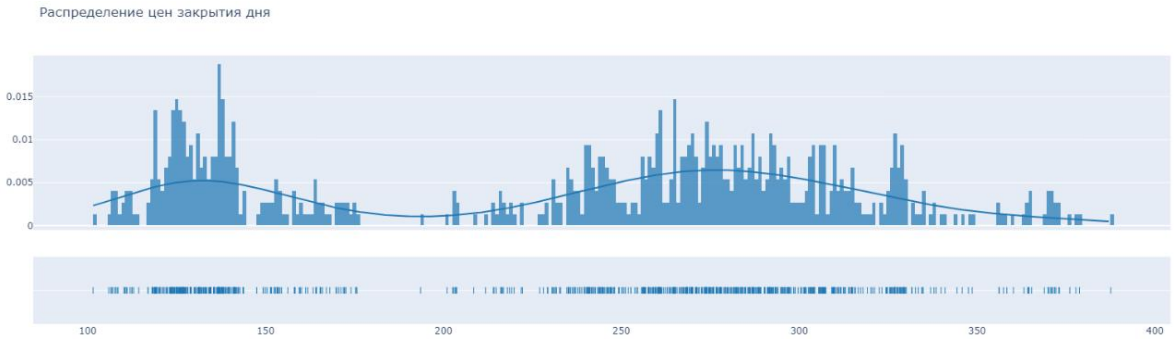


Рис. 1. Диаграмма распределения цены закрытия дня.

На рис. 2 представлен график вида «ящик с усами», на котором сравниваются распределения значений цены закрытия дня и количества проданных акций, исходя из диаграммы максимальная цена закрытия дня составляет 387,6, минимальная — 101,5, среднее значение цены равняется 257,8, а половина значений цен находится в диапазоне 141–291.

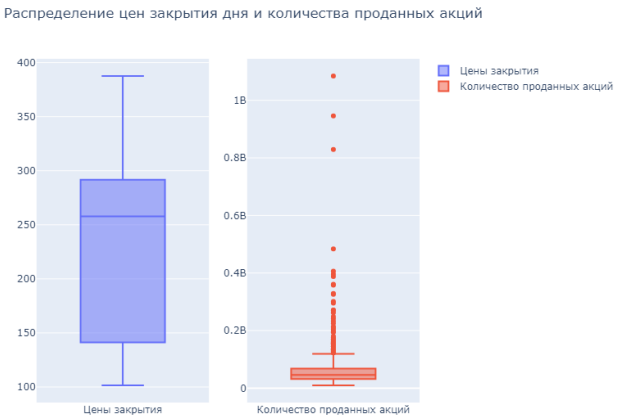


Рис. 2. Сравнение распределения значений

На рис. 3 построены графики изменения цены закрытия дня (синий график, левая ось) и количества проданных акций (красный график, правая ось), на основе совпадения резких изменений цены акции и количества проданных акций на данном графике можно сделать вывод, что изменения цены неестественны и были продиктованы внешними факторами. Такими факторами могут служить изменения геополитической обстановки, стихийные бедствия, изменение экономики и т. д. Данные события обычно освещаются в новостях, поэтому предполагается, что совмещение прогнозирования на основе статистики и новостей повысит качество прогнозов.

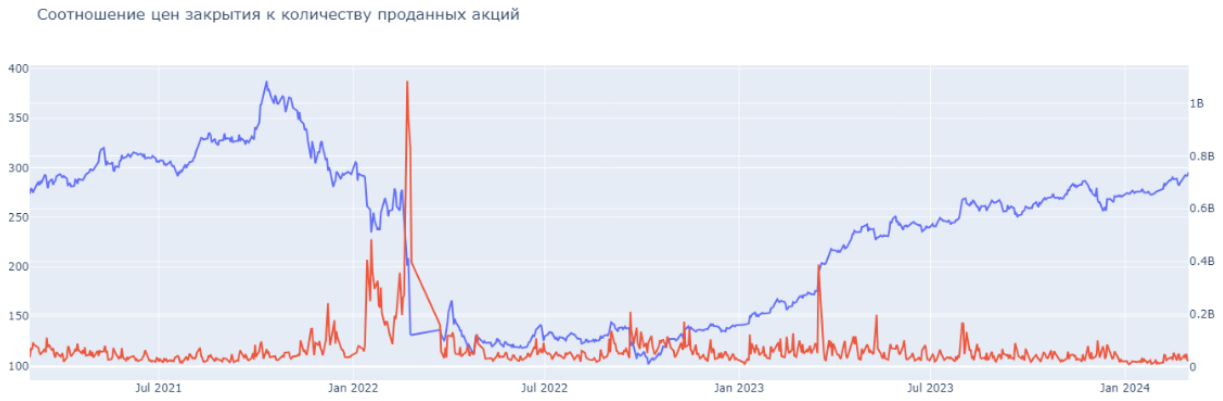


Рис. 3. Графики цены акции и количества проданных акций.

3. Разработка архитектур

Для сравнения было разработано 3 модели. Архитектура первой модели (рис. 4) содержит один LSTM слой с 256 нейронами и функцией активации selu, а также выходной полносвязный слой.

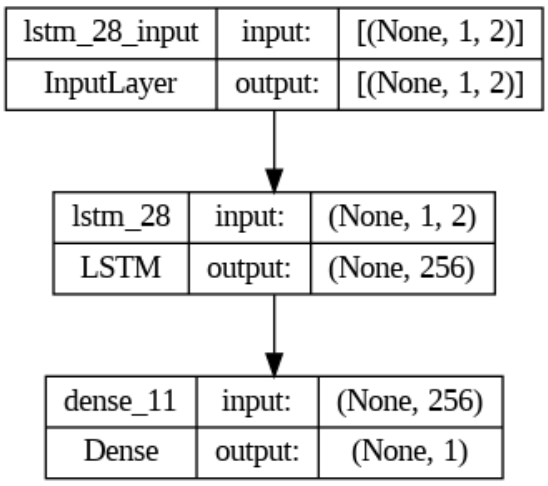


Рис. 4. Архитектура 1 модели.

Вторая модель (рис. 5) содержит 2 слоя LSTM по 128 нейронов каждый с той же функцией активации, слой регуляризации (dropout) и полносвязный слой на выходе.

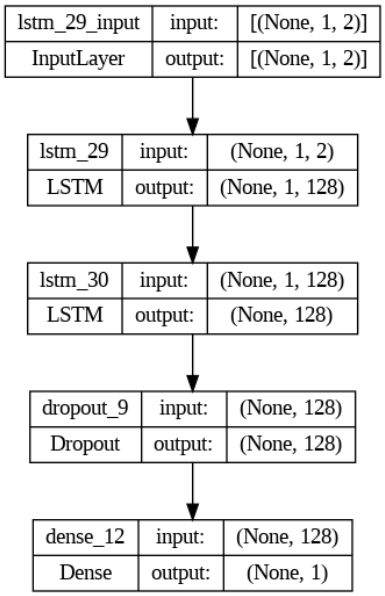


Рис. 5. Архитектура 2 модели.

Третья модель (рис. 6) имеет 3 LSTM слоя с 64 нейронами и той же функцией активации, 2 слоя регуляризации (после 2-го и 3-го рекуррентных слоев) и полносвязный слой на выходе.

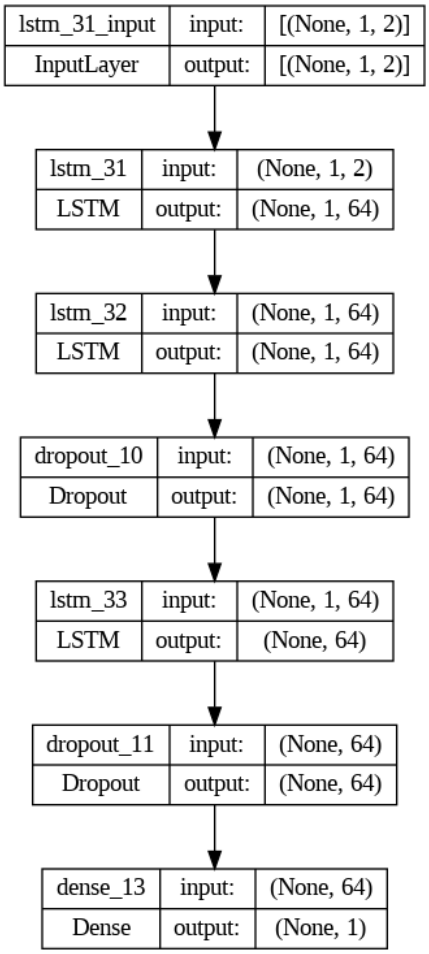


Рис. 6. Архитектура 3 модели.

В качестве метрик точности были использованы Mean Square Error (MSE), Mean Absolute Error (MAE) и R^2, которые были взяты из библиотеки scikit-learn[3]. В таб. 1 приведены значения метрик для каждой из моделей после 10 эпох обучения.

Таблица 1

Метрики точности моделей

Метрика \ Архитектура	Модель 1	Модель 2	Модель 3
MSE	9.7378	7.1024	12.6741
MAE	2.3764	1.9115	3.0229
R^2	78.39%	84.24%	71.88%

Графики прогноза всех моделей трудно отличить друг от друга, поэтому вместо демонстрации всех графиков продемонстрирован только 1 (рис. 7), от модели 2, так как она обладает наилучшими показателями. На данном графике видно, что график, построенный моделью, имеет высокую степень наложения на график реальных значений.

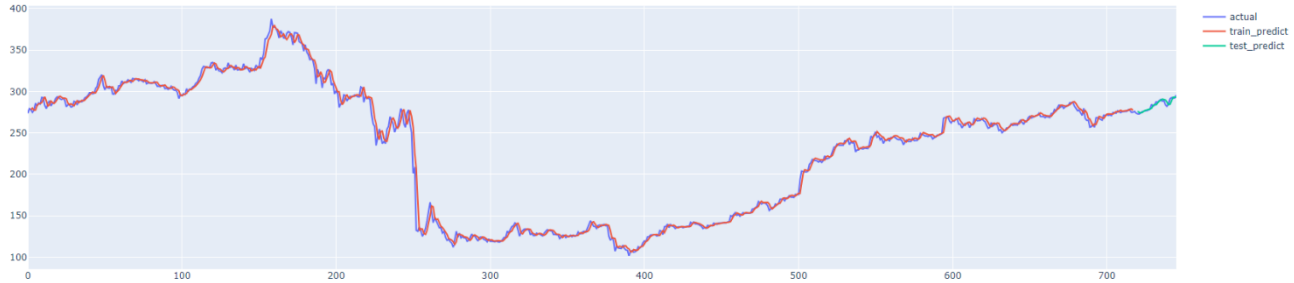


Рис. 7 График предсказания второй модели.

4. Исследование влияния параметров

После подготовки трех архитектур моделей был проведен анализ точности при изменяющихся параметрах. В качестве изменяющихся параметров были выбраны размер входного окна и количество нейронов, которое будет использоваться на каждом слое. Для комбинации параметров была создана функция, которая перебирает значения параметров из множеств, после чего преобразует набор данных в подходящий формат, с установкой входного окна, и передает данные модели. Множество размера входного окна: 2, 4, 6, 8, 10, 12, 14; а множество количества нейронов: 64, 128, 256, 512. Комбинации параметров и точности моделей представлены в таб. 2, отрицательные значения означают что данное сочетание параметров не подходит для текущей модели и, для использования таких параметров, требуется изменить архитектуру модели или количество эпох обучения, количество которых, как и в начальных моделях, было 10. Из таблицы следует, что модель 1 имеет наивысшую точность при размере входного окна 2 и 512 нейронах (79,7%), модель 2 — при 2 и 256 (83,11%), а модель 3 — при 2 и 64 (83,07%). Так же, проанализировав таблицу, можно сделать вывод, что для всех моделей оптимальным размером входного окна является 2 на всех вариантах количества нейронов на слоях и на всех вариантах архитектур, которые были рассмотрены в исследовании.

Таблица 2

Комбинации параметров и точности моделей

Размер входного окна	Количество нейронов на каждом слое	Точность модели 1	Точность модели 2	Точность модели 3
2	64	-145,56	82,49	83,07
4	64	32,49	74,23	-172,04
6	64	30,16	-53,63	65,32
8	64	0,03	60,67	-218,82
10	64	-132,33	-131,19	-124,88
12	64	-254,44	-453,14	-1345,96
2	128	59,73	79,7	74,13
4	128	59,43	43,14	74,96
6	128	31,45	64,31	71,8
8	128	-32,42	51,61	53,2
10	128	-106,6	24,02	-222,53
12	128	-116,94	-236	-75,93
2	256	77,27	83,11	67,9
4	256	63,81	-7,18	19,78
6	256	50,78	54,59	48,82
8	256	25,03	30,7	61,75
10	256	-174,48	0,74	28,44
12	256	-142,49	-337,92	-706,39
2	512	79,77	82,89	80,29
4	512	57,7	60,19	50,96
6	512	43,99	-5,57	-38,02
8	512	-52,11	-61,68	61,9
10	512	-104,67	-24	-85,94
12	512	-148,71	24,5	-628,71

Заключение

В ходе данного исследования была разработана модель на основе рекуррентной нейронной сети, прогнозирующая курс акций Сбербанка, которая имеет достаточную точность прогноза, чтобы считать его достоверным. Наиболее точный прогноз строит вторая модель, а оптимальный результат достигается при размере входного окна 2 и 256 (и 128) нейронах на каждом слое.

Так как у прогнозирования по историческим данным есть недостаток — не учитывание внешних факторов, то для улучшения показателей данной модели будет добавлена обработка новостей. Несмотря на то, что LSTM модель не нуждается в переобучении, планируется до обучать модель каждый месяц для сохранения точности прогноза.

Литература

1. Репозиторий библиотеки moexalgo на GitHub. — URL: https://github.com/moexalgo/moexalgo (дата обращения 20.03.24);
2. Документация библиотеки plotly. — URL: https://plotly.com/python/ (дата обращения 25.03.24);
3. Документация библиотеки scikit-learn. — URL: https://scikit-learn.org/stable/modules/model_evaluation.html (дата обращения 26.03.24).

УДК 004.032.26 : 004.932 : 004.925.8 : 519.85

ПЕРЕНОС ДЕФОРМАЦИИ ТРЁХМЕРНОЙ МОДЕЛИ ЛИЦА АКТЁРА НА ТРЁХМЕРНУЮ МОДЕЛЬ ЛИЦА ДРУГОГО АКТЁРА С УЧЁТОМ АНАТОМИЧЕСКИХ ОСОБЕННОСТЕЙ

А. М. Усачев

Воронежский государственный университет

Введение

Создание цифрового дублёра — достаточно сложная задача, которая занимает при этом очень много времени как с точки зрения подготовки данных, так и с точки зрения времени расчётов компьютера. Для того чтобы создать цифрового дублёра, необходимо произвести несколько видов съёмок, обработать полученные данные, создавая для каждого кадра скан актёра, отмечать на них контуры лица, губ, положение маркеров, и далее выполнить несколько этапов ретопологии. Полученный результат необходимо стабилизировать, высчитать для него положение моделей глаз, зубов, посчитать текстурные изображения конкретно для каждого кадра. И такую последовательность действий необходимо повторять при создании каждого компьютерного персонажа, а для одного и того же персонажа — повторять большинство из этих действий. Но производственных возможностей для выполнения подобных задач часто может не хватать: например, если необходимо создать большое количество второстепенных персонажей, или же если эти персонажи не являются людьми, а лишь обладают чертами человеческого лица. В таком случае технология захвата движений — лишь часть процесса создания персонажа. При этом удобно было бы создать программу (или подпрограмму в уже имеющейся программе), которая будет использовать уже созданных компьютерных персонажей для создания новых, минимизировав тем самым затраты времени на его создание.

Настоящая работа посвящена реализации алгоритма, от которого требуется переносить выражение лица с уже созданного компьютерного персонажа на нового, для которого на данный момент есть только нейтральное выражение лица. Также необходимо, чтобы при переносе эмоции она соответствовала анатомии лица создаваемого персонажа, учитывала форму и масштаб лица. В перспективе необходимо будет разработать помимо алгоритма ещё и полную последовательность действий, которая будет приводить к созданию полностью анимированного компьютерного персонажа, то есть также анимирование глаз, челюсти (перенос вращений глаз и движения челюсти достаточно простые задачи, чтобы выделять их в отдельную работу), создание изменяемых текстурных изображений. При создании алгоритма необходимо учитывать, что если он создается для последующего использования пользователем, то стоит минимизировать число входных данных, чтобы сделать работу с алгоритмом удобной.

1. Обзор литературы

В работе [1] предлагается решение о переносе деформации между трёхмерными объектами, созданное не на основе переноса повершинных изменений между геометриями. В статье вместо повершинных изменений рассматриваются изменения между полигонами. Задача сводится к решению системы линейных алгебраических уравнений, что позволяет рассматривать эту задачу как задачу оптимизации, и далее модифицировать её, добавляя дополнительные невязки в функцию ошибки. Такой способ, в отличие от переноса

повершинных изменений, учитывает также соседние вершины, поэтому данный метод выдаёт результат, более подходящий для сильно отличающихся нейтральных геометрий. В своей работе мы будем использовать этот метод как основой для переноса деформаций трёхмерных моделей лиц.

В работе [2] предлагаются решения некоторых проблем трансфера деформаций, а именно:

1. Основная проблема метода — итоговая трёхмерная модель может быть произвольно смещена в пространстве, и это не является ошибкой, так как такое поведение появляется при сведении задачи к решению системы линейных алгебраических уравнений. В данной статье предлагается отключить изменение тех вершин модели, которые меняются меньше заданного порога.

2. В применении к лицевой анимации возникает следующая проблема: верхняя и нижняя губа человека, а также верхние и нижние веки глаз движутся вне зависимости друг от друга. Из-за этого могут возникать ситуации, когда нижняя губа пересекает верхнюю (или наоборот), либо наоборот: у исходного персонажа закрыт рот или глаза, но при работе алгоритма у результата они не закрыты. Для этого предложено решение: создать в геометрии дополнительные треугольники, соединяющие верхнюю и нижнюю губу. При тестировании этого метода результат становится лучше, но при этом всё ещё возможны проблемные ситуации, изложенные выше.

3. При применении трансфера деформации у модели возможны заломы, и поэтому предлагается сглаживать изменение модели. На данный момент в работе нет планов по применению сглаживания.

В работе [3] приводится пример решения задачи переноса деформации на основе не одного, а нескольких примеров, которые можно смешивать между собой. На данный момент данное решение не актуально для настоящей работы, потому что в качестве входных данных мы предполагаем отсутствие дополнительных примеров изменений геометрии. Безусловно, дополнительные примеры можно создать, но для них как раз придется использовать алгоритм, который этих дополнительных примеров не требует.

2. Постановка задачи

В рамках работы будет решаться задача, целью которой является перенос деформации с трёхмерной модели лица одного актера на трехмерную модель лица другого актера с учетом анатомических особенностей. Учёт анатомических особенностей необходим для создания более реалистичной трехмерной модели лица актёра. Под анатомическими особенностями понимаются:

1. Сохранение объема головы. Это означает, что у деформированной модели лица актёра не должно быть вмятин, которых бы не появилось у реального актёра из-за черепа и челюсти.

2. Деформируемая модель лица актёра должна быть стабилизирована, то есть подразумевается, что при изменении времени положение черепа не изменяется (за исключением нижней челюсти).

3. Веки деформируемой модели лица актёра при своём смещении должны учитывать существование глаз актера как твёрдых объектов.

4. Веки и губы деформируемой модели лица актёра должны закрываться тогда, когда закрываются веки и губы на модели лица актёра, с которой берётся деформация.

5. При переносе деформации необходимо учитывать индивидуальные особенности лиц актёров, которые меняют трехмерную модель лица.

Решением задачи будет создание алгоритма, который принимает на вход следующие

данные:

1. Нейтральная трёхмерная модель лица актёра, с которого берется деформация.
2. Деформированная трёхмерная модель лица актера, с которого берётся деформация.
3. Нейтральная трёхмерная модель лица актера, к которой будет применена деформация.
4. Вспомогательная информация, определяющая анатомически характеристики трёхмерных моделей лиц актёров (на данный момент можно к ним отнести трёхмерные модели глаз, черепа для каждого актера; номера полигонов на геометрии, относящихся к губам; точки на веках для каждого актера, определяющие линию пересечения век).

На выходе алгоритма получается стабилизированная трёхмерная модель актера, содержащая ту же деформацию, что и трёхмерная модель лица другого актёра, и при этом были учтены анатомические особенности, о которых сказано выше, что улучшает качество результата.

3. Реализация поставленных задач

Поставленные задачи решаются с помощью методов оптимизации, то есть с помощью минимизации функции ошибки. В качестве метода оптимизации был выбран метод оптимизации второго порядка Гаусса — Ньютона. Для решения задачи переноса деформации между моделями используется алгоритм, предложенный в [2]. В ней вводится понятие деформационный градиент — матрица трансформации треугольника, включающая в себя вращение и масштабирование. Статья предлагает требовать, чтобы деформационные градиенты между топологиями моделей совпадали. Разница между деформационными градиентами моделей является первой невязкой в оптимизации. Второй невязкой в оптимизации является копирование расстояния от трёхмерной модели лица актера до трехмерной модели черепа актера между персонажами. Третьей невязкой является копирование расстояний между вершинами верхней и нижней губы актеров.

Программная реализация подразумевает под собой реализацию поставленной задачи на языке программирования C++ с использованием фреймворков Qt и CasADi, где CasADi — фреймворк автоматического дифференцирования, позволяющий достаточно быстро высчитывать аналитические производные заданных функций и использовать их при минимизации функции ошибки.

Пример работы алгоритма представлен на рис. 1. Пример взаимного расположения с черепом представлен на рис. 2. Пример работы невязки копирования расстояния представлен на рис. 3.

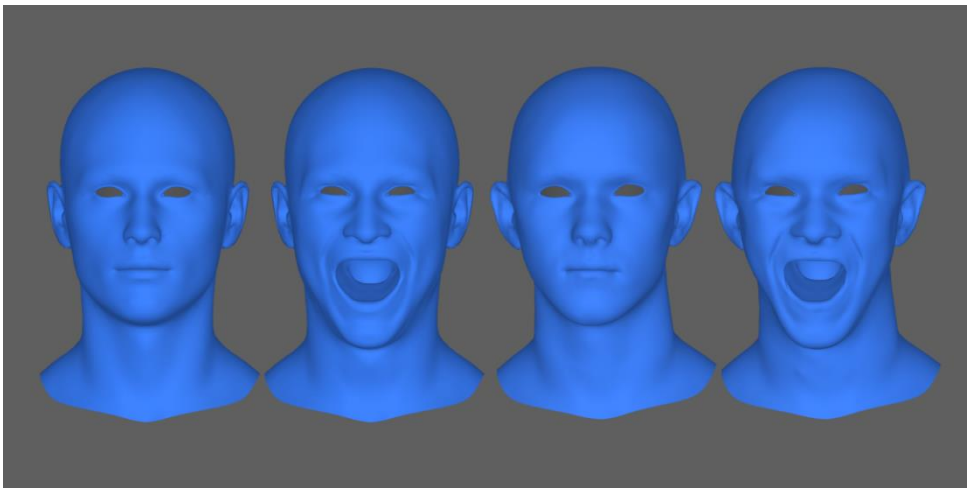


Рис. 1. Слева направо: нейтральная 3D-модель актёра, эмоция актера, нейтральная 3D-модель персонажа, эмоция персонажа, полученная предложенным алгоритмом

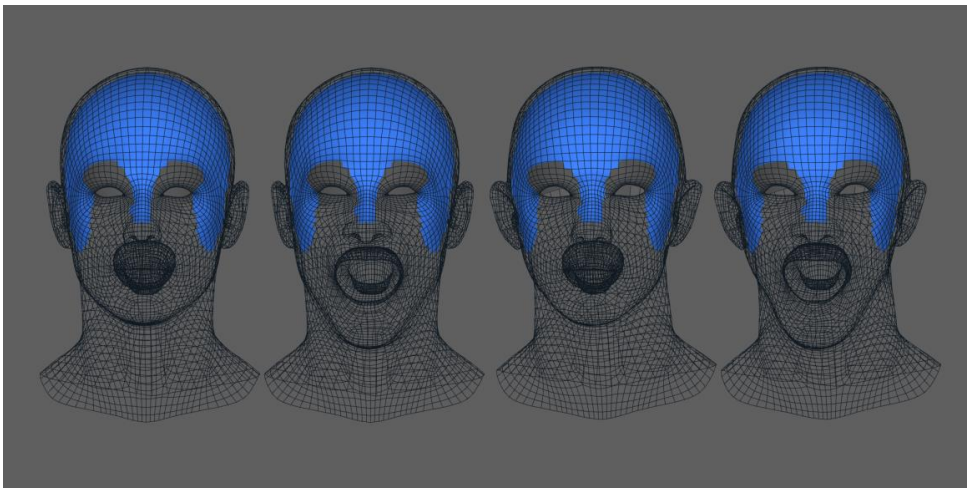


Рис.2. Пример, аналогичный представленному на рис. 2., с визуализацией 3D-модели черепа, используемой при решении задачи

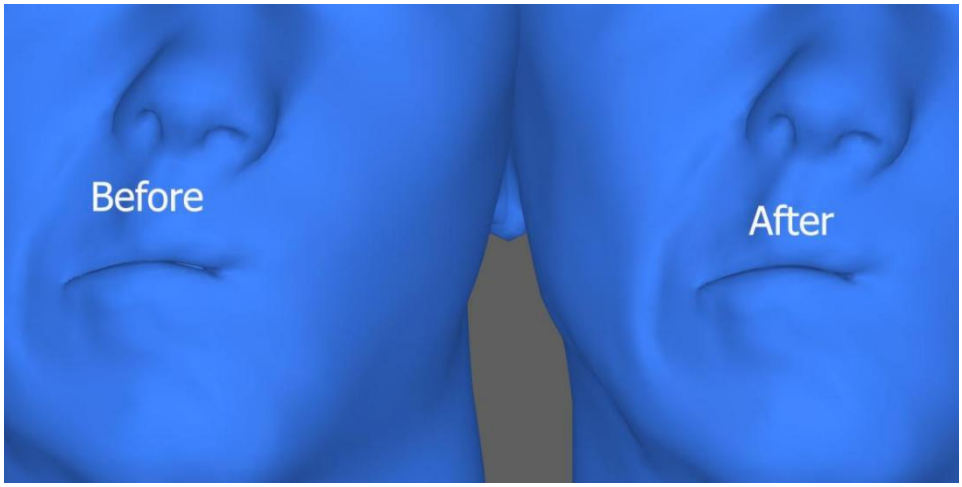


Рис. 3. Слева изображен результат работы алгоритма без использования невязки по копированию расстояния между губами, справа — с использованием невязки

Заключение

В рамках работы был предложен алгоритм по переносу деформации трёхмерной модели лица актера на трёхмерную модель лица другого актера с учетом анатомических особенностей. Предложенный алгоритм был реализован с использованием фреймворка CasADI и применён к имеющимся в распоряжении трёхмерным моделям.

Литература

1. Saito J. Smooth contact-aware facial blendshapes transfer / J. Saito // Proceedings of the Symposium on Digital Production, 2013.

2. Sumner R. W. Deformation transfer for triangle meshes / R. W. Sumner, J. Popović // ACM Transactions on Graphics, 2004. — Vol. 23(3). — P. 399–405.

3. Chu H.-K. Example-based Deformation Transfer for 3D Polygon Models / Hung-Kuo Chu, Chao Hung Lin // Journal of Information Science and Engineering, 2014. — Vol. 26(2). — P. 379–391.

УДК 004

РАЗРАБОТКА ИГРЫ ОТ ПЕРВОГО ЛИЦА НА ОСНОВЕ UNITY

А. В. Фалалеева, Т. В. Курченкова

Воронежский государственный университет

Введение

Современная игровая индустрия — одна из областей, которая не сбавляет темпы динамичного развития в течение уже долгого времени, привлекая миллионы игроков по всему миру. Существует множество разнообразных игр, начиная от визуальных новелл и заканчивая полнометражными сюжетными играми с реалистичной графикой. Каждый человек может выбрать то, что будет ему по вкусу: один игрок предпочтёт жанр платформеров [1], второй — многопользовательскую игру, в которой необходимо кооперироваться с товарищами по команде, чтобы победить противника, третий же игрок может играть ради увлекательного сюжета. Помимо различных жанров, пользователи выбирают комфортный для них способ управления (клавиатура и мышь, геймпад или VR), а также по представлению игрового мира и взаимодействию с ним (игры от первого лица, игры от третьего лица и др.). Игры от первого лица являются одним из наиболее захватывающих и востребованных жанров. Людей может привлекать не только кинематографичность игр, но и возможности различных игровых механик, которые позволяют игроку влиться в атмосферу игры на более глубоком уровне.

Разработка игр от первого лица на Unity стала актуальной темой не только благодаря растущему интересу со стороны игроков, но и в связи с тем, что Unity стал популярным игровым движком, предоставляя широкий спектр инструментов для реализации идей разработчиков. Unity славится не только своими доступностью и гибкостью, но и способностью обеспечивать высокое качество графики, анимации и функциональность в создании игр от первого лица.

1. Постановка задачи

Требуется разработать игру, в которой игрокам предстоит пройти через препятствия, решить различные загадки и найти выход из запертого пространства.

Игра представляет собой квест, суть которого состоит в том, чтобы игроки нашли выход, путём нахождения подсказок и раскрывая сюжет, а также преодолевая преграды и открывая различные двери с помощью ключа, правильного решения загадки или кода доступа.

Функциональность игры должна включать в себя разработку механик игры, включая управление персонажем и взаимодействие с окружением, преодоление препятствий и решение головоломок, а так же хранение игровых элементов.

2. Механики игры

Для разработки игры требуется определить механики, которые необходимы для реализации задуманной идеи. Необходимо иметь чёткое представление о структуре и взаимодействии её компонентов [2]. Диаграмма классов помогает в этом, иллюстрируя отношения между классами игры.

В основные игровые механики игры от первого лица в Unity включаются управление персонажем, систему сбора ключей и взаимодействие с дверьми, рычагами и активируемыми объектами (рис. 1). Далее представлено более подробное описание каждого из аспектов разработки игровых механик.

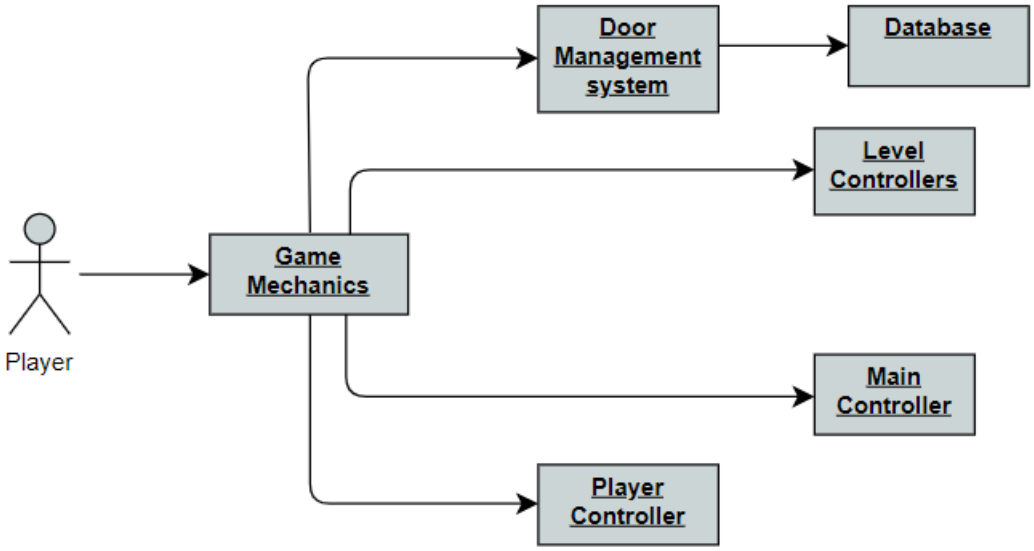


Рис. 1. Механики игры

2.1. Управление персонажем от первого лица

Для управления персонажем от первого лица необходимо разработать систему управления персонажем с помощью клавиатуры и компьютерной мыши. Персонаж может передвигаться после получения команд с соответствующих кнопок с клавиатуры. Обычно за передвижение вперёд, назад, влево и вправо отвечают клавиши W, A, S, D соответственно, а за поворот камеры — движение мыши. Также для преодоления различных препятствий необходимо реализовать возможность прыжков, приседаний и бега.

2.2. Система сбора ключей

Для разработки игровой механики, позволяющей открывать двери с помощью ключа были созданы объекты-ключи. Также необходимо разработать механику взаимодействия с ключами, позволяющей игроку подбирать и хранить их в инвентаре. Чтобы ключи открывали соответствующие двери была реализована проверка наличия ключа в инвентаре.

2.3. Взаимодействие с дверьми, рычагами и активируемыми объектами

В игре присутствуют различные типы дверей, каждая из которых требует определённого взаимодействия для её открытия. Вот основные типы дверей и их механика открытия:

Обычные двери. При приближении к двери игрок может активировать соответствующую клавишу, которая запустит анимацию открытия двери.

Двери, требующие ключа. Для таких дверей игрок должен найти соответствующий ключ и сохранить его в своём инвентаре. Игрок сможет открыть дверь только тогда, когда у него есть ключ.

Двери, открывающиеся после нажатия на рычаг. В игровом пространстве

размещены рычаги, которые игрок должен найти и активировать, после чего дверь, за которую отвечает рычаг, откроется.

Двери, открывающиеся после решения загадки. Игрок столкнётся с загадкой или головоломкой, которую нужно решить, чтобы получить доступ к открытию двери.

Таким образом, игра предлагает игроку интересный вызов в виде лабиринта с различными типами дверей, требующих разных методов открытия. Чтобы достичь выхода, игрок должен проявить наблюдательность, решать головоломки, искать ключи и взаимодействовать с окружающим миром.

Были созданы различные типы дверей, такие как обычные двери, двери, требующие ключей, двери, открывающиеся после нажатия на рычаг, и двери, открывающиеся после решения загадок. Также были реализованы механики взаимодействия с дверьми, позволяющей игроку открывать и закрывать их при выполнении определённых условий.

Созданы рычаги и объекты, которые игрок может активировать для изменения состояния дверей или других игровых элементов.

Для улучшения игрового опыта были разработаны анимации открытия и закрытия дверей.

2.4. Обработка столкновений и физики

Rigidbody является наиболее распространенным компонентом, используемым при интерактивной разработке. Динамика Rigidbody основана на ньютоновском принципе движения и массы. Rigidbody — это идеализированное твёрдое тело, размер и форма которого фиксированы и остаются неизменными при воздействии некоторых внешних сил и которое используется в ньютоновской механике для моделирования реальных объектов [3]. В разработке интерактивных приложений невозможно использовать абсолютно точную физику из-за естественных ограничений, таких как стандартная частота кадров. Однако для достижения реалистичности моделирования физические законы должны быть представлены насколько это возможно.

Реализация обработки столкновений игрового персонажа с препятствиями и объектами в игровом мире выполнена с использованием Rigidbody. Также прописана физическая симуляция для достижения реалистичного поведения объектов и персонажа, для чего была проведена настройка физических параметров, таких как масса, трение и упругость, для достижения желаемого игрового поведения.

В процессе разработки игровых механик важно обеспечить плавность и отзывчивость управления персонажем, а также убедиться в правильной работе системы сбора ключей и взаимодействия с дверьми, рычагами и активируемыми объектами. Важно провести тестирование и отладку разработанных механик для обнаружения и устранения возможных ошибок или несоответствий.

2.5. Создание локальной базы данных для хранения объектов

В проекте присутствует локальная база данных, где хранятся загадки для реализации дверей, открывающихся после её решения. При старте игры загадка выбирается случайным образом.

База данных была создана внутри Unity через ScriptbleObject.

ScriptableObject — это класс движка Unity, который позволяет определить пользовательский объект данных, который может быть создан во время выполнения и сохранен как ресурс Unity [4]. Это позволяет управлять ресурсами, не создавая каждый раз новый скрипт на C#, что может упростить разработку и упростить управление проектом и его

масштабирование.

Объекты ScriptableObject определяются их собственными автономными файлами, которые могут быть размещены в папке Assets проекта точно так же, как и любой другой ресурс Unity. Вместо того, чтобы наследоваться от класса `MonoBehaviour`, они наследуются от класса `ScriptableObject`.

Скриптовые объекты могут быть использованы в качестве контейнеров данных, например, можно создать класс ScriptableObject для хранения предметов инвентаря или, в данном проекте, базы данных загадок. Можно создавать экземпляры этого класса ScriptableObject во время выполнения, изменять их свойства и сохранять эти изменения в качестве ресурса Unity.

2. Разработка классов приложения

Основным элементом игры являются двери разных типов, каждая из которых требует определённого взаимодействия для открытия. Для их реализации были созданы: интерфейс IInteractable и классы, которое реализуют его методы (рис. 2).

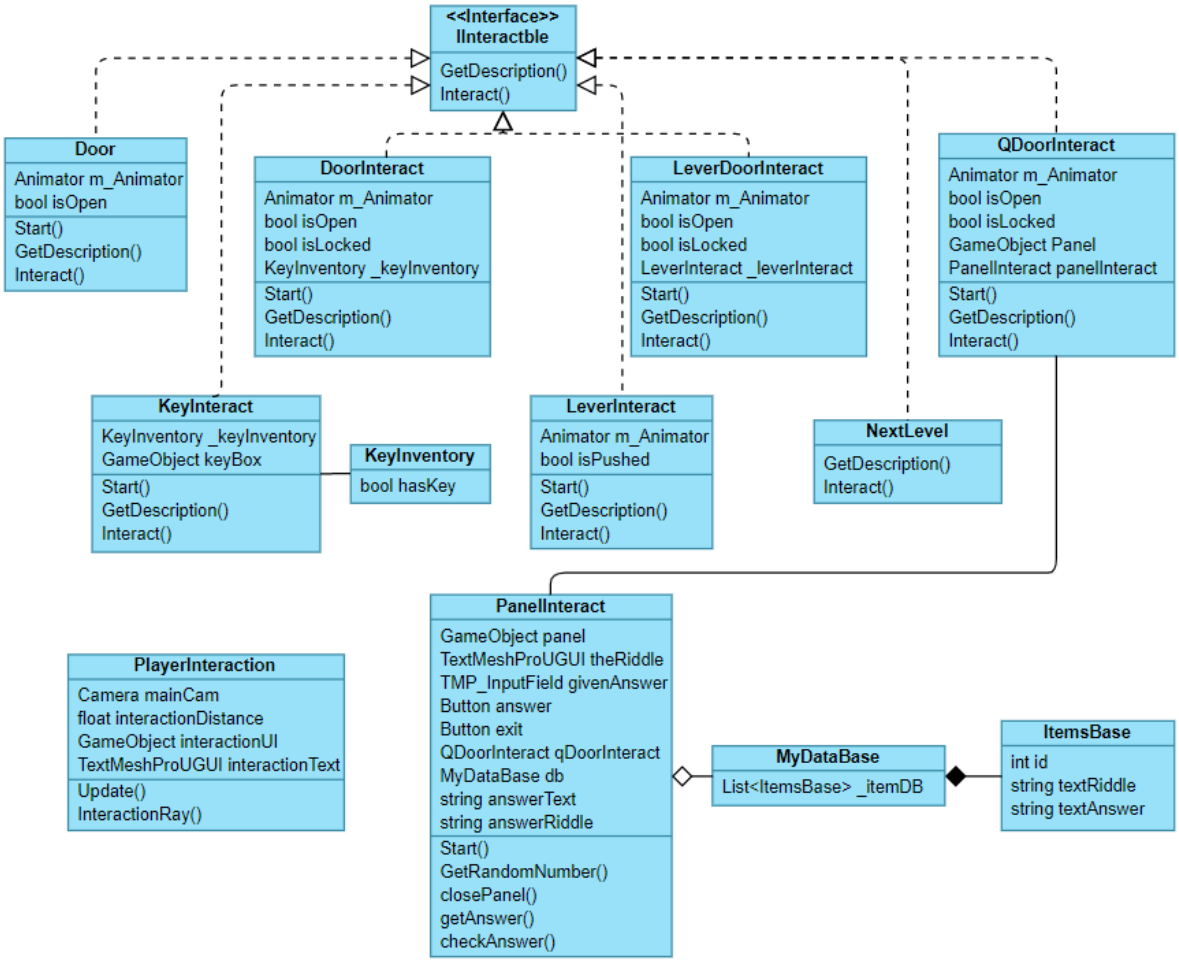


Рис. 2. Диаграмма классов

Интерфейс IInteractable определяет два метода: Interact() и GetDescription(). Каждый вид дверей реализуется в отдельном классе, наследующий интерфейс IInteractable. Метод Interact() вызывается при взаимодействии с дверью. После проверок соответствующих типам дверей он изменяет значение isOpen, инвертируя его (если дверь открыта, то закрывает, и наоборот),

после чего вызывается анимация открытия/закрытия двери. Метод GetDescription() предназначен для получения описания действия, которое может быть выполнено с объектом, на котором расположен скрипт.

Практически каждый класс имеет метод Start(), в котором происходит инициализация скрипта и проверка того, чтобы в начале игры все двери были закрыты.

2.1. Механика обычной двери

Поведение обычной двери описано в классе Door. Метод GetDescription() возвращает строку, описывающую действие, которое можно выполнить с дверью, как подсказка игроку. Если дверь открыта, возвращается строка "Press [E] to <color=red>close</color> the door". В обратном случае возвращается строка "Press [E] to <color=green>open</color> the door".

Метод Interact() вызывается при взаимодействии с дверью. Он изменяет значение isOpen, инвертируя его (если дверь открыта, то закрывает, и наоборот). Затем он устанавливает соответствующее значение параметра "isOpen" в аниматоре. Если дверь открыта, то аниматору устанавливается значение true для параметра "isOpen", чтобы проиграть анимацию открытия двери. Если дверь закрыта, то аниматору устанавливается значение false для параметра "isOpen", чтобы проиграть анимацию закрытия двери.

Таким образом, данный класс позволяет игроку взаимодействовать с дверью в игре, открывая и закрывая ее при нажатии на клавишу "E". Анимация открытия или закрытия двери воспроизводится с использованием компонента аниматора (`Animator`).

2.2. Механика двери, отпирающейся ключом

Для реализации дверей, которые открываются с помощью ключа, необходимо добавить дополнительную логику в классе DoorInteract. В этом случае, каждая дверь будет иметь уникальный идентификатор ключа. Также будет присутствовать инвентарь игрока, который будет хранить информацию о наличии ключей.

При подходе к двери, происходит проверка наличия ключа с соответствующим идентификатором в инвентаре игрока. Если ключ найден, игрок может активировать дверь и открыть ее, используя ту же логику, как и в случае с обычными дверями. В противном случае, дверь останется закрытой.

Для реализации этой механики были созданы классы KeyInventory, KeyInteract. В KeyInventory хранится значение, показывающее наличие в инвентаре игрока нужного ключа для открытия двери.

Класс KeyInteract имеет реализации методов, определенных в интерфейсе IInteractable. Метод `GetDescription() возвращает строку "Press [E] to get key". Метод Interact() вызывается при взаимодействии с объектом. Он устанавливает значение переменной “hasKey” в объекте KeyInventory равным true, что указывает на наличие ключа в инвентаре, и делает объект “keyBox” неактивным (выключает его).

2.3. Механика двери, отпирающейся рычагом

Класс LeverInteract отвечает за взаимодействие с рычагом в игре. Метод Interact() вызывается при взаимодействии с рычагом. Значение isPushed устанавливается в true, чтобы указать, что рычаг был нажат. Затем устанавливается соответствующее значение параметра "isPushed" в аниматоре, чтобы активировать соответствующую анимацию рычага.

Метод GetDescription() возвращает строку, описывающую действие, которое можно выполнить с рычагом. Если рычаг уже был нажат, возвращается строка "Lever was already pushed". В противном случае, возвращается строка "Press [E] to push lever".

При активации рычага игроком, запускается анимация активации рычага с помощью компонента Animator. Следующим шагом будет открытие соответствующей двери, используя аналогичную логику, как и в предыдущих случаях. Сначала проверяется был ли нажат рычаг: если да, то можно открывать дверь, если же нет – необходимо найти рычаг.

2.4. Механика двери, открывающейся после решения загадки

Класс PanelInteract отвечает за взаимодействие с панелью загадок в игре. Метод Start() выполняется при запуске скрипта. Если база данных пуста, то загружается предустановленная загадка и правильный ответ. В противном случае, выбирается случайная загадка и ответ из базы данных db, и они отображаются на панели.

Метод GetRandomNumber() возвращает случайное число в заданном диапазоне min и max, чтобы выбрать случайную загадку из базы данных.

Метод closePanel() вызывается при нажатии на кнопку закрытии панели. Он делает панель неактивной и устанавливает режим блокировки курсора на "заблокирован".

Метод getAnswer() вызывается при получении ответа от игрока. Он сохраняет введенный текст ответа в переменную answerText.

Метод checkAnswer() вызывается когда игрок нажимает на кнопку “Ответить”. Он сравнивает текст ответа answerText с правильным ответом answerRiddle. Если ответ верный, панель закрывается и устанавливается isLocked в false для объекта qDoorInteract, разблокировав при этом дверь.

Отличие класса QDoorInteract от предыдущих реализаций поведения дверей состоит в том, что помимо стандартных переменных m_Animator, isOpen и isLocked, он содержит ссылку на объект панели с загадкой и ссылка на класс PanelInteract, отвечающий за взаимодействие с панелью загадки.

В методе Interact()проводится проверка: если дверь заблокирована (isLocked == true), активируется панель с загадкой Panel и устанавливается режим блокировки курсора на "разблокирован". Если пользователь ввёл правильный ответ на загадку, дверь разблокируется (isLocked становится равным false). В противном случае, дверь остаётся заблокированной. Если дверь не заблокирована, происходит переключение её состояния (открыто/закрыто) и соответствующая анимация.

Таким образом, реализация различных типов дверей в игре от первого лица в Unity требует создания соответствующих игровых объектов, анимаций и взаимодействия с игроком. Это позволяет создать разнообразные и интересные игровые ситуации в лабиринте, где игрок должен применять различные навыки и решать задачи для продвижения вперёд.

Заключение

В ходе работы были разработаны игровые механики и классы, которые являются основой для сюжета игры. Благодаря разработке этих классов была создана игра, наполненная различными элементами геймплея. Разнообразие используемых механик дверей и взаимодействия пользователя с игровым миром расширяет игровой контент, что позволяет заинтересовать игрока на более глубоком уровне, а решение загадок для продвижения по сюжету игры добавляет дополнительный элемент взаимодействия и интриги, стимулируя игрока к размышлениям и принятию творческих решений.

Литература

1. Макнил, С. Hey! Listen!: путешествие по золотому веку видеоигр / С. Макнил; [перевод с английского С. А. Евтушенко]. – Москва : Эксмо, 2020. – 368 с.

2. Бонд, Дж. Г. Unity и C#. Геймдев от идеи до реализации / Дж. Г. Бонд ; 2-е изд. – СПб. : Питер, 2019. – 928 с.

3. Aava Rani, K. Learning Unity Physics / K. Aava Rani. – Birmingham : Packt Publishing Ltd., 2014. – 128 с.

4. Unity – Manual: ScriptableObject. – Режим доступа: https://docs.unity3d.com/Manual/class-ScriptableObject.html

УДК *004.4*

ФАЗЗИНГ МЕТОДОМ ЧЕРНОГО ЯЩИКА.

Фирстов В.В.

Воронежский государственный университет

Введение

Тема моего выступления – фаззинг методом черного ящика. Особое внимание уделим его использованию, принципу работы и техникам тестирования.

Фаззинг - техника тестирования программного обеспечения, часто автоматическая или полуавтоматическая, заключающаяся в передаче приложению на вход неправильных, неожиданных или случайных данных. Предметом интереса являются падения и зависания, нарушения внутренней логики и проверок в коде приложения, утечки памяти, вызванные такими данными на входе.

В зависимости от того, насколько доступны сведения о проверяемом объекте, выделяют три подхода к тестированию: Black-box, Gray-box и White-box. В случае с White-box фаззингом специалист понимает, как устроена и реализована его цель. Он проводит тестирование на уровне исходного кода. При реализации Gray-box специалист имеет доступ к скомпилированной программе, но не к ее исходникам. Обладая неполным представлением о внутреннем устройстве цели, он пытается получить недостающую информацию о ней в процессе анализа.

1.BLACK-BOX ФАЗЗИНГ

1.1. Что такое и для чего нужен

Согласно терминологии ISTQB(Международный квалификационный совет по тестированию программного обеспечения) *Black-box тестирование* – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы.

Метод используется в:

1. Интеграционное тестирование.

Тестирование, в котором программные и аппаратные компоненты объединяются и тестируются для оценки взаимодействия между ними. При использовании метода «черного ящика» тестировщик проверяет, корректно ли работают все компоненты в целом тогда, когда они интегрированы в большую систему. И действительно, нормальная работа каждой составляющей по отдельности – это еще не гарантия того, что они будут работать вместе в рамках всего проекта. Например, данные могут не отправиться через интерфейс, или интерфейс не отработает согласно документации. При планировании таких тестов тестировщики опираются на спецификацию.

2. Функциональное тестирование.

Используя этот метод, тестировщик проверяет, выполняет ли программное обеспечение все заявленные функции и требования клиента в полном объеме согласно документации.

3. Стресс-тестирование.

Предположим, что у нас есть букмекерская онлайн-контора, в документации к которой заявлена возможность одновременной регистрации 1000 пользователей. В этом случае

стрессовым тестированием будет непрерывный поток автоматизированных регистраций (как минимум, 1000 регистраций в минуту) на протяжении 12 часов.

4. Usability-тестирование.

Пусть в упомянутой букмекерской конторе есть функционал «Купон»: мы проверяем, сколько времени уходит у пользователя для добавления ставки в купон, ввода суммы и завершения ставки.

5. Тестирование производительности.

Таким видом тестирования мы можем проверить: есть ли утечки памяти, насколько быстро система работает и выдает обратную связь, не потребляет ли наше ПО слишком много трафика и не создает ли избыточное количество подключений.

6. Приемочное тестирование.

После проверки ПО тестировщиками его отдают заказчику, который запускает приемочные тесты «черного ящика» на основе ожиданий от функциональности. Как правило, набор тестов в этом случае определяет сам заказчик, за ним же остается право отказаться от приемки (если его не устроили результаты тестирования).

7. Регрессионное тестирование.

Проводится на протяжении всего цикла разработки. Цель такого тестирования – проверить работоспособность нового кода и выяснить, не привел ли он к ошибкам или поломкам в старом функционале.

При выборе набора регрессионных тестов следует использовать следующие рекомендации:

- делается репрезентативная выборка тестов, в которой используются все функции ПО;
- выбираются тесты, сосредоточенные на программных компонентах/функциях, которые подверглись изменениям;
- используеются дополнительные тестовые примеры, уделяя основное внимание функциям, на которые с наибольшей вероятностью повлияли изменения.

Для регресса также используется метод «белого ящика», особенно при поиске функций, на которые с большой вероятностью повлияли изменения.

8. Beta-тестирование.

Практически готовое ПО отдают для «обкатки» желающим для выявления максимального количества ошибок еще до того, как оно попадет к конечному пользователю.

Это дает:

- идентификацию непредвиденных ошибок (так как бета-тестеры используют ПО нестандартно);
- широкий набор окружений для проверки, который трудно обеспечить иными методами (разные операционные системы, разные настройки, разные версии браузеров);
- снижение расходов (так как работа бета-тестеров, как правило, не оплачивается).

2.ПРИНЦИПЫ РАБОТЫ

2.1.Фаззер

Ключевую роль в фаззинге играет сам инструмент фаззер, программное обеспечение, предназначенное для автоматизированного тестирования безопасности других программ путем вставки некорректных, случайных данных во входные параметры. Основная цель фаззера - находить уязвимости и ошибки в программном обеспечении, подвергая его воздействию большого количества некорректных, искаженных или неожиданных входных данных. Такие данные могут вызвать сбои, зависания, утечки памяти или другие нежелательные последствия, указывающие на наличие уязвимостей.

Первым делом определяются все входные точки целевой программы, такие как интерфейсы, файлы, сетевые соединения или другие источники, через которые программа принимает данные. Далее фаззер генерирует большое количество разнообразных тестовых данных, которые могут включать в себя случайные данные, некорректные форматы, чрезмерно длинные строки, специальные символы и другие потенциально проблемные входные данные. Сгенерированные тестовые данные вводятся в целевую программу через определенные входные точки. Это может быть автоматизировано с помощью скриптов или инструментов фаззинга. Во время ввода тестовых данных фаззер внимательно отслеживает поведение программы, ища признаки ошибок, аварийных остановок, утечек памяти, отказов в обслуживании или других нежелательных реакций. Когда фаззер обнаруживает ошибку или нежелательное поведение, он собирает всю возможную информацию, такую как стек вызовов, регистры процессора, входные данные, приведшие к сбою, и другие диагностические данные. Собранная информация анализируется для определения причины сбоя и типа обнаруженной уязвимости. Это может включать анализ бинарных файлов, отладку кода или обратный инжиниринг. Процесс фаззинга повторяется с новыми наборами тестовых данных, основанных на полученных результатах. Фаззер может использовать эвристические методы, такие как мутация входных данных или направленное нечеткое тестирование, чтобы сфокусироваться на определенных областях кода или типах уязвимостей.

2.2.Санитайзер

Одним из важнейших компонентов фаззера является санитайзер, необходимый при подготовке тестовых данных и обеспечении безопасности процесса фаззинга. Он выполняется следующие функции:

1. Мутация входных данных: санитайзер применяет различные техники мутации для генерации новых тестовых данных на основе существующих примеров или семплов входных данных. Это может включать вставку, удаление, замену или перестановку байтов, битов или последовательностей.
2. Генерация случайных данных: санитайзер может генерировать полностью случайные данные, не основанные на существующих семплах. Это помогает расширить покрытие тестовых случаев.
3. Форматирование данных: санитайзер может преобразовывать сырые бинарные данные в форматы, релевантные для тестируемой программы, такие как HTTP-запросы, файлы различных типов или специальные протоколы.

4. Фильтрация данных: санитайзер может фильтровать вредоносные или запрещенные данные, чтобы избежать нежелательных последствий при фаззинге, таких как выполнение вредоносного кода или повреждение системы.
5. Управление состоянием: в некоторых случаях санитайзер может управлять состоянием тестируемой программы, сбрасывая ее в определенные точки или восстанавливая предыдущие состояния для эффективного тестирования.
6. Параллелизация: современные санитайзеры часто используют параллельную обработку для одновременной генерации и мутации большого количества тестовых данных, ускоряя процесс фаззинга.
7. Инструментация: санитайзер может инструментировать двоичный код или среду выполнения для сбора дополнительной информации о поведении программы во время фаззинга, такой как покрытие кода или состояние памяти.

2.3. Техники тестирования

1. Эквивалентное разбиение.
Эта техника включает в себя разделение входных значений на допустимые и недопустимые разделы и выбор репрезентативных значений из каждого раздела в качестве тестовых данных. Она может быть использована для уменьшения количества тестовых случаев. Допустим, у нас есть целая переменная N в диапазоне от -99 до 99: позитивными классами эквивалентности будут [-99, -10], [-9, -1], 0, [1, 9], [10, 99], а недействительными (негативными) – <-99, >99, пустое значение, нечисловые строки.

2. Анализ граничных значений.
Техника, которая включает в себя определение границ входных значений и выбор в качестве тестовых данных значений, находящихся на границах, внутри и вне границ. Многие системы имеют тенденцию вести себя некорректно при граничных значениях, поэтому оценка значений границ приложения очень важна. При проверке мы берем следующие величины: минимум, (минимум-1), максимум, (максимум+1), стандартные значения. Например, в том же случае -99 <= N <= 99 будет использоваться набор: -100, -99, -98, -10, -9 -1, 0, 1, 9, 10, 98, 99, 100.

3. Тестирование таблицы переходов.
При данной технике сценарии тестирования выбираются на основе выполнения корректных и некорректных переходов состояний. Допустим, мы хотим записаться на прием к врачу и зарезервировать время своего приема: заходим в форму, выбираем удобное для нас время и нажимаем кнопку «Записаться». Сразу после этого выбранное нами время становится недоступно для другой записи, так как первая запись привела к изменению в базе.

4. Тестирование по сценариям использования.
Эта техника используется при написании тестов для индивидуального сценария пользователя с целью проверки его работы.

3.ВЫВОДЫ

3.1. Преимущества

Достоинства метода

1. Тестирование методом «черного ящика» позволяет найти ошибки, которые невозможно обнаружить методом «белого ящика». Простейший пример: разработчик забыл добавить какую-то функциональность. С точки зрения кода все работает идеально, но с точки зрения спецификации это – сверхкритичный баг.
2. «Черный ящик» позволяет быстро выявить ошибки в функциональных спецификациях (в них описаны не только входные значения, но и то, что мы должны в итоге получить). Если полученный при тестировании результат отличается от заявленного в спецификации, то у нас появляется повод для общения с аналитиком для уточнения конечного результата.
3. Тестировщику не нужна дополнительная квалификация. Часто мы пользуемся различными сервисами и приложениями, не очень в них разбираясь.
4. Тестирование проходит «с позиции пользователя». Пользователь всегда прав, он конечный потребитель практически любого ПО, а значит, ему должно быть удобно, комфортно и понятно.
5. Составлять тест-кейсы можно сразу после подготовки спецификации. Это значительно сокращает время на тестирование: к тому моменту, как продукт готов к тестированию, тест-кейсы уже разработаны, и тестировщик может сразу приступать к проверке.

3.2. Недостатки

Недостатки метода

1. Основным недостатком метода «черного ящика» является возможность пропуска границ и переходов, которые не очевидны из спецификации, но есть в реализации кода (собственно, это и заставляет тестировщиков использовать метод «белого ящика»).
2. Можно протестировать только небольшое количество возможных вводных (входящих) значений; многие варианты остаются без проверки.
3. Тесты могут быть избыточными, если разработчик уже проверил данную функциональность (например, Unit-тестом).
4. При отсутствии четкой и полной спецификации проектировать тесты и тест-сценарии оказывается затруднительно.

Заключение

Из представленной информации можно сделать следующий вывод: метод «черного ящика» является эффективным при различных видах тестирования, но следует помнить, что некоторые ошибки невозможно найти, используя только этот метод (например, ошибки во внутренней структуре кода).

Проведение «black-box» тестирования увеличивает уверенность в том, что приложение надежно работает на широком диапазоне входных данных, так как набор тестовых данных зависит только от спецификации, а не от особенностей внутренней реализации продукта (как в случае применения методов «белого» и «серого» ящиков).

Метод «черного ящика» выгодно применять, если необходимо найти:

1. неправильно реализованные функции приложения или сервиса;
2. ошибки в пользовательском интерфейсе;

3. ошибки в функциональных спецификациях.

Для реализации наиболее полной проверки рекомендуется использовать методы «черного» и «белого» ящиков одновременно. Это позволит увеличить покрытие возможных сценариев, снизить риск пропуска ошибки, а также качественно улучшить результаты тестирования, так как приложение или сервис будет проверено двумя разными методами – с позиций пользователя и внутреннего устройства системы.

Литература

1. Особенности тестирования «черного ящика». – Режим доступа: https://quality-lab.ru/blog/key-principles-of-black-box-testing/
2. Незаменимый фаззинг. – Режим доступа: https://securitymedia.org/info/nezamenimyy-fazzing.html

УДК *004.42*

ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ИГР С ПОМОЩЬЮ ИНСТРУМЕНТОВ ИГРОВОГО ДВИЖКА UNITY

А. Е. Холодова, Е. В. Трофименко

Воронежский государственный университет

Введение

Компьютерные игры стали неотъемлемой частью современной культуры и занимают значительное место в повседневной жизни миллионов людей по всему миру. Оптимизация компьютерных игр играет ключевую роль в обеспечении плавного и комфортного игрового процесса, повышении производительности и качества графики, а также обеспечивает оптимальное функционирование игровых приложений на различных устройствах, начиная от персональных компьютеров и заканчивая мобильными устройствами.

Игровая индустрия постоянно развивается, требования к графике и производительности игр увеличиваются. В условиях быстрого прогресса задача оптимизации становится все более значимой для разработчиков игр, стремящихся сделать графику более реалистичной и ее визуализацию без задержек и сбоев.

Поэтому вопрос производительности и скорости обработки рендеринга стоит на первом месте. Данная статья посвящена изучению цикла оптимизации игры и вопроса измерения производительности игр с помощью предоставляемых игровым движком Unity инструментов.

Компания Unity для тестов производительности предлагает Unity Profiler. Есть и другие ранее популярные инструменты: от PIX до Intel VTune. Однако современный профайлер Unity предлагает все, что необходимо для внесения наиболее важных изменений. Для некоторых конкретных платформ можно использовать дополнительные инструменты (XCode или Android Studio и другие), чтобы упростить доступ к информации об устройстве. Но обычно встроенных инструментов Unity достаточно для выполнения оптимизации.

1. Unity Profiler

Unity Profiler — это инструмент игрового движка Unity, который можно использовать для получения информации о производительности приложения [3]. Его также можно подключать к устройствам в сети или к подключенным к компьютеру для профилирования разных платформ, или запустить в редакторе Unity, чтобы получить представление о распределении ресурсов во время разработки игры.

Unity Profiler собирает и отображает данные о производительности вашего приложения. Например, в областях ЦП, памяти, рендеринга и звука. Это полезный инструмент для определения мест, требующих улучшения производительности игры, и последующей оптимизации. Профилировщик помогает точно определить, как код, ресурсы, настройки сцены, рендеринг камеры и настройки сборки влияют на производительность игры. Он отображает результаты в виде серии диаграмм, поэтому наглядно видно, где происходят пики и спады производительности приложения (рис.1).

Помимо использования встроенного профилировщика Unity, можно использовать API-интерфейс профилировщика для экспорта данных профилирования в сторонние инструменты, а также пакет Profiling Core для настройки анализа профилирования. Также есть возможность

добавить в проект мощные инструменты профилирования, такие как Memory Profiler и Profile Analyser, для более детального анализа данных производительности.

Чтобы получить доступ к окну Unity Profiler, нужно перейти в меню: Окно > Анализ > Профилировщик.

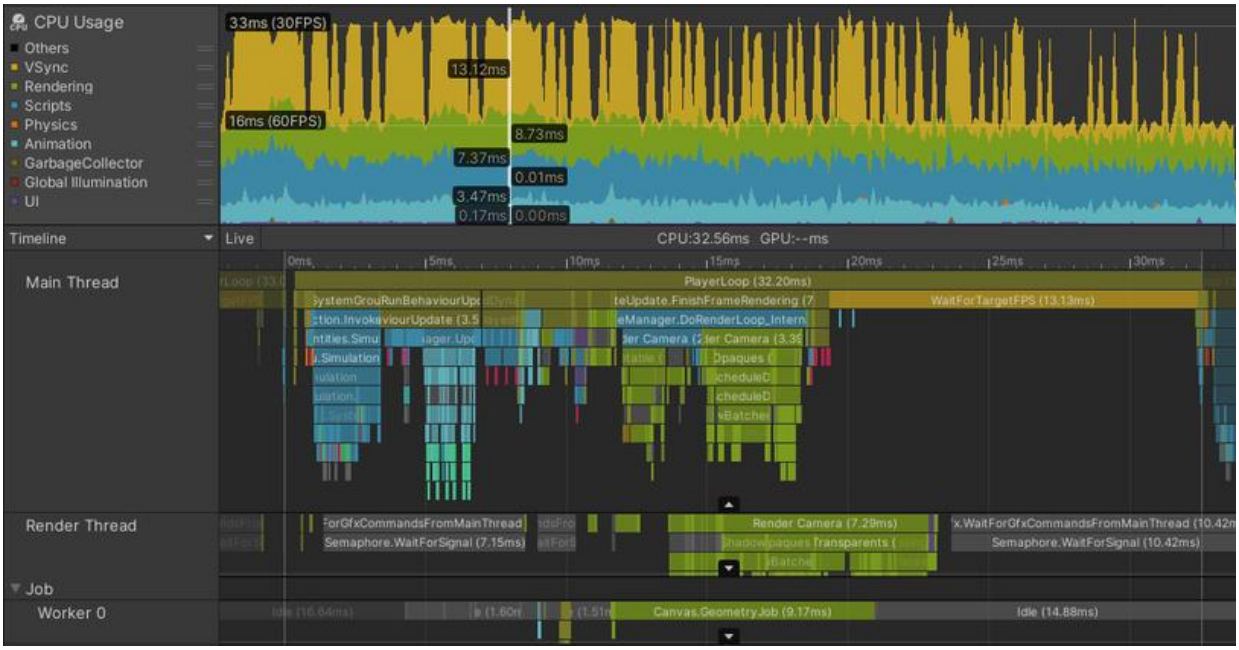


Рис. 1. Пример работы Unity Profiler

На рис.1 показан пример работы *Unity Profiler.* Верхний график – график использования игрой ЦП. На нем отмечены оба параметра и *миллисекунды на кадр,* и соответствующие им кадры в секунду. Каждый кадр длится не более 22 ms, то есть игра работает стабильно без перегрева устройства при частоте 30 кадров в секунду. Под графиком использования ЦП показаны важные для оптимизации приложения потоки (или треды). Можно заметить, что основной тред проводит 13 ms времени в желтом маркере WaitForTargetfps. То есть реальная работа потока завершается на 19 ms и остальное время до 33 ms основной поток проводит в состоянии покоя, ожидая начало следующего кадра. Это хорошая ситуация, так как время простоя ЦП может использовать для охлаждения и расходовать меньшее количество электроэнергии [3].

2. Этапы цикла оптимизации игры с использованием Unity Profiler

Первый этап оптимизации - профилирование игры. Процесс профилирования нужно начинать с просмотра fps в Unity Profiler. Если игра выдает fps, которые разработчиков устраивают и подходят по лимиту, то далее можно анализировать расход памяти игрой с помощью Memory Profiler и проанализировать, не превышает ли он установленного лимита.

Memory Profiler [2] - пакет, предлагаемый Unity Engine для захвата, изучения и сравнения Snapshot’ов памяти. Snapshot - запись того, как была организована память, используемая приложением, когда Memory Profiler захватил снимок. Окно Memory Profiler также показывает распределение собственной и управляемой памяти, что позволяет оценить использование памяти игрой и выявить потенциальные проблемы, например утечки памяти.

Если fps игры не входят в лимит, то необходимо найти, где происходит эффект “узкого горлышка”. Для этого нужно проверить не ожидает ли главный поток окончание выполнения любого другого треда. Можно выделить три основных типа тредов:

1. Главный поток или main thread: основной или главный тред, в котором выполняется основная логика написанных скриптов игры и где проводит большинство времени системы физики, анимаций, рендеринга и UI.
2. Поток рендера или render thread: главный поток, выполняя разные оптимизационные функции для рендера, создает список объектов для рендеринга. Этот лист передается в поток рендера, который вызывает необходимые методы из графического API Unity, управляющего графическим процессором платформы, на которой запущена игра.
3. Потоки задач или Job worker threads: эти потоки будут в приложении, если оно использует C# Job System. Эта система позволяет разработчикам запускать треды-работники, выполняющие побочную работу, уменьшая нагрузку на основной тред. Также физика, анимации и рендеринг Unity используют C# Job System.

Наиболее частые ситуации ожидания главного потока [3]:

1. Главный тред ждет render тред. Тогда ожидается ли GPU?
    a. Да. Значит, проблема в GPU. Такую ситуацию называют GPU bound
    b. Нет. Проблема в рендер треде - Render thread bound
2. Главный тред ждет поток задач - Job worker thread bound
3. Главный тред свободен. Значит, проблема в самом главном потоке - Main thread bound

Для каждого из выделенных случаев нужно посмотреть на разные статистики для решения проблем производительности.

Для ситуации GPU bound важно посмотреть настройки графики и качества и проанализировать данные из отладчика кадров (Frame Debugger).

Для ситуации Render thread bound важно посмотреть шейдеры, настройки графики и качества, провести анализ данных из отладчика GPU (GPU Debugger).

Для ситуаций Job worker thread bound и Main thread bound нужно изучить наиболее затратные места с помощью Unity Profiler и Profile Analyzer. Profile Analyzer визуализирует данные полученные в процессе профилирования, а также поддерживает режим сравнения нескольких результатов разных сессий профилирования, что облегчает поиск время затратных процессов. Если проблемное место сложно найти, то можно посмотреть стек вызовов или нагрузку на нативный CPU. Можно включить настройку Deep Profiling. При включенном Deep Profile, Unity Profiler профилирует по всему написанному коду и записывает все вызовы функций и даже первый уровень запросов к Unity API. Но этот метод очень ресурсо емкий и памяти затратный, так как Unity добавляет ко всем скриптовым методам инструментарий профайлера, для записи всех вызовов функций.

После анализа данных профилировщиков нужно исправлять выявленную проблему. Обычно решения для выделенных ситуаций такие:

1. Render thread bound: необходимо оптимизировать камеры, использовать culling (процесс, по которому объекты вне видимости не рендерятся) и вызывать запросы на отрисовку одной пачкой(batch) запросов.
2. GPU bound: меши и шейдеры, эффекты постобработки требуют оптимизации. Можно также посмотреть разрешение и сжатие текстур, изучить, нет ли нигде в коде повторной отрисовки(overdraw).
3. Job worker thread bound и Main thread bound: нужно обратить внимание на физику игры и написанные скрипты, выделение и сборку мусора, камеры и UI игры, анимации и зависимости job’ов и их параллелизацию.

Следующий этап после внесения изменений - повторное профилирование для сравнения с помощью Profile Analyzer того, как новые правки повлияли на производительность игры.

Этот оптимизационный цикл можно повторять много раз, пока разработчик не будет доволен производительностью игры.

3. Рекомендации и советы для процесса профилирования

1. Лучше измерять бюджет игры в миллисекундах на кадр вместо привычных кадров в секунду.

Если измерять производительность игры в кадрах в секунду, то можно упустить следующую неприятную для игроков ситуацию. Если пара кадров, находящихся рядом будут рендериться разное время (например, один кадр - 0,5 секунды, а второй - 0,1 секунды), то переход к новому более быстрому кадру будет заметно менее плавным для игрока. Но при замерах средних кадров в секунду это будет упущено.

Для расчета бюджета игры в миллисекундах на кадр можно воспользоваться простой формулой $\mathrm{fps}' = \frac{1000\ \mathrm{ms}}{\mathrm{DesiredFPS}}$, где DesiredFPS - желаемый бюджет по кадрам в секунду.

Таким образом для цели в 30 кадров бюджет миллисекунд на кадр составит 33.33 миллисекунды, для 60 - 16.66. Таким образом на отрисовку одного кадра игры для достижения задуманного бюджета должно уходить меньше 33.33 и 16.66 миллисекунд соответственно.

Этот лимит можно превысить во время показа сцен с низкой интерактивностью, но не во время геймплея. Любой кадр, который рендерится дольше установленного бюджета, будет сказываться на производительности игры. Это особенно важно для игр виртуальной реальности (VR), так как для прохождения сертификации игры при ее публикации необходимо поддерживать fps на уровне 120 кадров без прерываний.

2. Профилировать и оптимизировать игру стоит начинать как можно раньше.

Чем раньше будут выявлены и решены проблемы, тем проще будет работать с проектом и в дальнейшем поддерживать его. Заранее и часто проводите профилирование, чтобы был понятен стандартный уровень производительности проекта. Если он пойдет на спад, то это можно быстро и легко заметить и устранить проблему.

Нужно стремиться к состоянию, когда потоки могут комфортно находится в состоянии покоя (idle), и при этом не превышен бюджет игры по кадрам.

3. Определить, что задерживает исполнение программы — GPU или CPU— можно, используя Timeline профайлера CPU:

• Gfx.WaitForPresent: ограничения GPU, CPU ожидает ответа от GPU;

• Gfx.WaitForCommands: ограничения CPU, GPU ожидает ответа от CPU.

4. Профилировать лучше, начиная от наиболее общего к частному.

Хорошо работает подход профилирования сверху вниз, начиная с отключенного Deep Profiling. На этом верхнем уровне полезно собрать основные данные и наметить возможные проблемные области.

Далее стоит проанализировать потенциальные проблемы, прибегая к уже описанным выше методам для нахождения источника замедления. Можно провести вторую сессию профилирования с уже включенным Deep Profiling.

5. *Рекомендуется также закрыть все другие приложения во время профилирования и отключите регулировку частоты процессора, например, Intel SpeedStep или Turbo Boost, чтобы избежать разгона процессора.*

Заключение

В статье была рассмотрена возможность оценивания производительности игры с помощью предоставляемых игровым движком Unity инструментов, таких как Unity Profiler и Profile Analyzer. Был изучен предлагаемый Unity оптимизационный цикл и приведены

рекомендации к процессу профилирования. Далее планируется применить эти знания на практике при замере производительности трех игр-примеров для оценки разных подходов для разработки игр.

Литература

1. Оптимизация игр на Unity: проверенный в деле план. – Режим доступа: https://habr.com/ru/companies/playgendary/articles/582950/ – (Дата обращения: 27.03.2024).
2. Memory Profiler. – Режим доступа: https://docs.unity3d.com/Packages/com.unity.memoryprofiler@1.0/manual/index.html – (Дата обращения: 12.04.2024).
3. Performance profiling tips for game developers. – Режим доступа: https://unity.com/ru/how-to/best-practices-for-profiling-game-performance – (Дата обращения: 10.04.2024).
4. Оптимизируйте игру с помощью Profile Analyzer. – Режим доступа: https://unity.com/ru/how-to/optimize-your-game-profile-analyzer – (Дата обращения: 01.04.2024).
5. Performance and Optimization. – Режим доступа: https://learn.unity.com/course/performance-and-optimisation?uv=2022.2 – (Дата обращения: 01.04.2024).
6. Профайлеры. – Режим доступа: https://unity.com/ru/features/profiling – (Дата обращения: 01.04.2024).

УДК 004.77

ОСНОВЫ РАЗРАБОТКИ СЕРВЕРНОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ НА ЯЗЫКЕ «JAVA»

Д. В. Хохлов, С. Ю. Болотова

Воронежский государственный университет

Аннотация: *Статья представляет основы разработки серверной части веб-приложений на языке Java. Она охватывает ключевые аспекты, начиная с обзора архитектур и переходя к конкретным этапам разработки. Особое внимание уделено обеспечению безопасности данных. Эта статья призвана помочь разработчикам создавать эффективные и надежные веб-приложения с использованием современных методов защиты данных.*

Ключевые слова: *веб-приложение, серверная часть, архитектура, этап разработки, безопасность, компоненты, данные.*

Введение

В мире, насыщенном виртуальными инновациями, веб-приложения становятся неотъемлемой частью повседневной жизни, предоставляя нам сложные системы, где клиентская и серверная части тесно взаимосвязаны. Разделение на эти две части обеспечивает высокий уровень безопасности и управляемости системы. Однако, для достижения эффективности необходима тщательная настройка и согласованность всех ее компонентов.

Серверная часть выступает в роли фундамента всей системы, обеспечивая обработку данных, их безопасное хранение и надежную работу приложения. Вся функциональность, доступная пользователю, зависит от производительности и надежности серверной части, что в современном мире является ключевым аспектом.

Правильно спроектированная серверная часть обеспечивает не только высокий уровень безопасности, но и гибкость и масштабируемость приложения. Ее эффективная архитектура снижает вероятность возникновения ошибок и упрощает процесс внесения изменений. Это, в свою очередь, экономит время и ресурсы при долгосрочном сопровождении приложения.

1. Основные архитектуры

Серверная часть веб-приложений представляет собой незаменимую основу современных информационных систем, обеспечивающую взаимодействие с клиентами посредством HTTP протокола. Серверы могут использовать разнообразные архитектурные концепции, каждая из которых соответствует определенным потребностям и задачам. Монолитная архитектура [1] объединяет все компоненты в одном приложении, что упрощает начальную разработку и тестирование. В то время как микросервисная архитектура разбивает функциональность на небольшие, независимые сервисы, что обеспечивает высокую гибкость, масштабируемость и обслуживаемость.

Кроме того, архитектура сервера может также делиться на два ключевых типа: stateful (с сохранением состояния) и stateless (без сохранения состояния) [1]. В первом случае сервер хранит информацию о состоянии клиента между запросами, что позволяет поддерживать долгосрочные взаимодействия и сложные бизнес-процессы. Во втором случае сервер не сохраняет состояние клиента, что делает его идеальным выбором для создания высокопроизводительных и масштабируемых веб-сервисов. Таким образом, выбор архитектуры серверной части зависит от уникальных требований проекта и целей, которые

необходимо достичь.

2. Разработка серверной части

В основе современной системы используется REST архитектура [2], которая является лучшим выбором для разработки веб-сервисов, благодаря своей простоте и эффективности. Её ключевые особенности включают легкость масштабирования, понятные и читаемые URL для взаимодействия с ресурсами, а также поддержку разделения клиентской и серверной частей, что способствует улучшению обслуживаемости и переносимости приложений. Отсутствие состояния между запросами делает RESTful идеальным выбором для создания высокопроизводительных и надежных веб-сервисов.

В основе веб-приложения используется фреймворк Java Spring Boot [3]. Он является мощным инструментом для разработки веб-сервисов, который обеспечивает простоту и эффективность создания RESTful архитектуры. Его надежная интеграция с PostgreSQL делает работу с базой данных легкой и эффективной. Также Spring Boot обеспечивает высокую защиту от внешних воздействий благодаря встроенным механизмам безопасности и поддержке аутентификации [4]. Кроме того, его высокая масштабируемость и возможность использования микросервисной архитектуры позволяют удовлетворить потребности разрастающихся проектов. Все это делает Spring Boot привлекательным выбором для создания надежных, безопасных и масштабируемых веб-сервисов

При разработке веб-приложения используется концепция MVC [5] — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер. Такой подход обеспечил независимое проектирование системных компонентов.

3. Основные этапы разработки серверной части

Основными этапами разработки любого сервера являются:

3.1. *Выделение основных сущностей.*

На этом этапе происходит определение основных объектов данных, которые будут использоваться в приложении. Это могут быть пользователи, продукты, заказы, комментарии и т. д. Выделение основных сущностей позволяет проектировать структуру базы данных и определять связи между ними.

3.2. *Создание базы данных.*

После определения основных сущностей необходимо создать базу данных PostgreSQL и создать таблицы, отражающие структуру данных приложения [6]. Каждая таблица соответствует определенной сущности, а столбцы в таблице представляют атрибуты этой сущности. Также на этом этапе может потребоваться создание индексов и ограничений для оптимизации работы с данными.

3.3. *Организация корректного приема и обработки запросов.*

Этот этап включает разработку механизмов обработки HTTP-запросов от клиентов. В Spring Boot это может быть реализовано с использованием контроллеров , которые обрабатывают запросы от клиентов и вызывают соответствующие сервисы для выполнения необходимых операций. Контроллеры должны обеспечивать корректную обработку запросов, проверку прав доступа и валидацию входных данных.

3.4. *Реализация внутренней логики обработки данных.*

На этом этапе разрабатывается бизнес-логика [7] приложения, которая определяет, как данные будут обрабатываться и взаимодействовать друг с другом. Это может включать в себя реализацию алгоритмов обработки данных, бизнес-правил, проверок и преобразований.

3.5. *Обеспечение взаимодействия с базой данных.*

Взаимодействие с базой данных PostgreSQL осуществляется с помощью механизмов доступа к данным, таких как Spring Data JPA или JDBC Template [8]. На этом этапе разрабатываются и реализуются методы доступа к данным (репозитории), которые позволяют выполнять операции чтения, записи, обновления и удаления данных из базы данных.

4. Обеспечение безопасности и защиты данных

Защита информации является критическим аспектом в разработке серверных приложений, особенно в контексте современной цифровой среды, где угрозы кибербезопасности постоянно усиливаются. Эффективная стратегия безопасности должна охватывать широкий спектр аспектов, включая аутентификацию, авторизацию, защиту данных от несанкционированного доступа и атак, а также обеспечение целостности и конфиденциальности информации.

Spring Boot [9], популярный фреймворк для разработки веб-приложений, предоставляет мощные инструменты для реализации безопасности. Он включает в себя Spring Security, специализированный модуль, который обеспечивает множество функций безопасности, начиная от аутентификации и авторизации пользователей и заканчивая защитой данных от различных видов атак [10].

Spring Security обеспечивает удобные средства для настройки аутентификации пользователей, поддерживая различные методы, такие как базовая аутентификация, форма входа, аутентификация по токену [11] и другие . Кроме того, он позволяет настроить различные стратегии авторизации, регулируя доступ пользователей к различным ресурсам на основе их ролей и привилегий. Таким образом, использование Spring Boot и Spring Security обеспечивает надежную защиту данных и приложений в целом, делая их более устойчивыми к угрозам кибербезопасности.

Заключение

В данной статье были рассмотрены основы разработки серверной части веб-приложений на языке Java. Начиная с обзора основных архитектур и переходя к конкретным этапам разработки серверной части, был провелен глубокий анализ ключевых аспектов этого процесса.

Понимание основных архитектур и принципов разработки серверной части позволяет разработчикам создавать эффективные и надежные веб-приложения. Важные этапы, такие как выделение основных сущностей, создание базы данных и организация обработки запросов, играют критическую роль в успешной реализации проекта.

Особое внимание уделено обеспечению безопасности и защиты данных, что является важным аспектом разработки любого веб-приложения. Использование современных методов аутентификации, авторизации и шифрования данных помогает обеспечить надежную защиту информации и предотвратить возможные угрозы безопасности.

Список литературы

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб. : Питер, 2016.
2. Фримен Э., Сьерра К., Бейтс Б. Паттерны проектирования. Переиздание. – СПб. : Питер, 2016.
3. Spring Framework Documentation. – URL: https://spring.io/guides
4. Уоллс К. Spring Boot в действии. – М. : ДМК Пресс, 2016.
5. Кэри Д. Изучаем паттерны проектирования. – СПб. : Питер, 2018.
6. PostgreSQL Documentation. – URL: https://www.postgresql.org/docs/
7. Официальная документация Java. – URL: ttps://docs.oracle.com/en/java/
8. PostgreSQL Tutorial. – URL: https://www.tutorialspoint.com/postgresql/index.htm
9. Spring Boot Tutorials. – URL: https://www.baeldung.com/spring-boot
10. Джошуа Блох. Java. Эффективное программирование. – СПб. : Питер, 2019
11. Официальная документация Spring Boot. – URL: https://docs.spring.io/spring-boot/docs/current/reference/html/

УДК 004

НАГРУЗОЧНОЕ ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ И АНАЛИЗ РЕЗУЛЬТАТОВ

М. А. Хрипунов

Воронежский государственный университет

Введение

Производительность веб-приложений напрямую влияет на качество предоставляемых услуг. Чтобы обеспечить достаточную производительность веб-приложения, необходимо протестировать его перед запуском под нагрузкой. Тесты продемонстрируют способность приложения обслуживать нужное количество одновременных пользователей и/или запросов, в то же время сохраняя приемлемое время отклика, а также позволят найти и устранить узкие места в сетевой инфраструктуре.

Данная статья посвящена проектированию, проведению и анализу нагрузочного тестирования.

1. Постановка задачи

Первоначально необходимо выбрать объект нагрузочного тестирования. Затем определить цели и требования предъявляемые приложению. После чего требуется настроить инструменты для сохранения и наглядного демонстрирования метрик и результатов тестов, спроектировать и провести нагрузочное тестирование, проанализировать полученные результаты, то есть сравнить их с изначальными ожиданиями. В конце принять решение по оптимизации и улучшению приложения.

2. Инструменты

В качестве инструмента для проведения нагрузочного используется программа - Apache JMeter. Apache JMeter является одним из популярных инструментов нагрузочного тестирования. Он обладает следующими особенностями: бесплатность и открытый исходный код; развитие и поддержка; масштабируемость; графический интерфейс и командная строка; интеграция с другими инструментами. Есть и другие инструменты для проведения нагрузочного тестирования, такие как Gatling и WebLOAD. WebLOAD распространяется на платной основе и обладает ограниченной документацией, можно столкнуться с ограниченным или недостаточно подробным описанием некоторых функций и возможностей. В свою очередь, Gatling не имеет такого большого сообщества и поддержки, как JMeter. Необходимо некоторое время для освоения и изучения специфического DSL (Domain Specific Language - язык предметной области), используемого в Gatling для создания тестовых сценариев. В некоторых случаях отладка и поиск ошибок может быть более сложным, чем в других инструментах.

Для визуализации и анализа метрик используется Chronograf - это наглядный веб-интерфейс для отображения и мониторинга данных, разработанный для работы с базой данных временных рядов InfluxDB. Особенности Chronograf: простота использования; интеграция с InfluxDB; удобство установки. Этот инструмент подходит для анализа небольших проектов, однако, если потребности в анализе данных и масштабировании будут расти, стоит изучить

альтернативные инструменты, например, ELK и Grafana. ELK(Elasticsearch, Logstash, Kibana) и Grafana сложнее в плане настройки и конфигурации. Могут потреблять больше ресурсов, чем Chronograf, особенно при обработке больших объемов данных или запросах.

3. Этапы реализации работы

3.1. Выбор объекта тестирования

В качестве объекта тестирования было выбрано приложение, предоставляющее возможности аналогичные социальной сети, например, регистрация и авторизация, добавление в друзья, отправка сообщений и т.д. Серверная часть написана на языке Java, с использованием фреймворка Spring Boot. Для хранения данных используется СУБД PostgreSQL. JWT применяется для безопасной передачи информации между двумя сторонами в форме токена. Docker обеспечивает автоматизацию развёртывания и управления приложением. Swagger позволяет автоматически описывать API. Liquibase является инструментом миграции баз данных. Клиентская часть реализуется на языке JavaScript с использованием библиотеки React.

3.2. Определение целей и требований

Цели:

1. Понять, как ведет себя система под нагрузкой.
2. Определить возможности веб-приложения.

Требования:

1. Так как приложение находится на раннем этапе разработки, достаточно, чтобы оно выдерживало минимальную нагрузку.

3.3. Создание тестового сценария

Самая важная часть работы – создание профилей виртуальных пользователей. Каждый виртуальный пользователь эмулирует действия пользователя реального. Профиль определяет его маршрут по веб-сайту и другие параметры. Каждое выполнение профиля во время теста создает одну сессию пользователя. Для этого буду использовать расширение предоставляемое Google – Blazemeter.

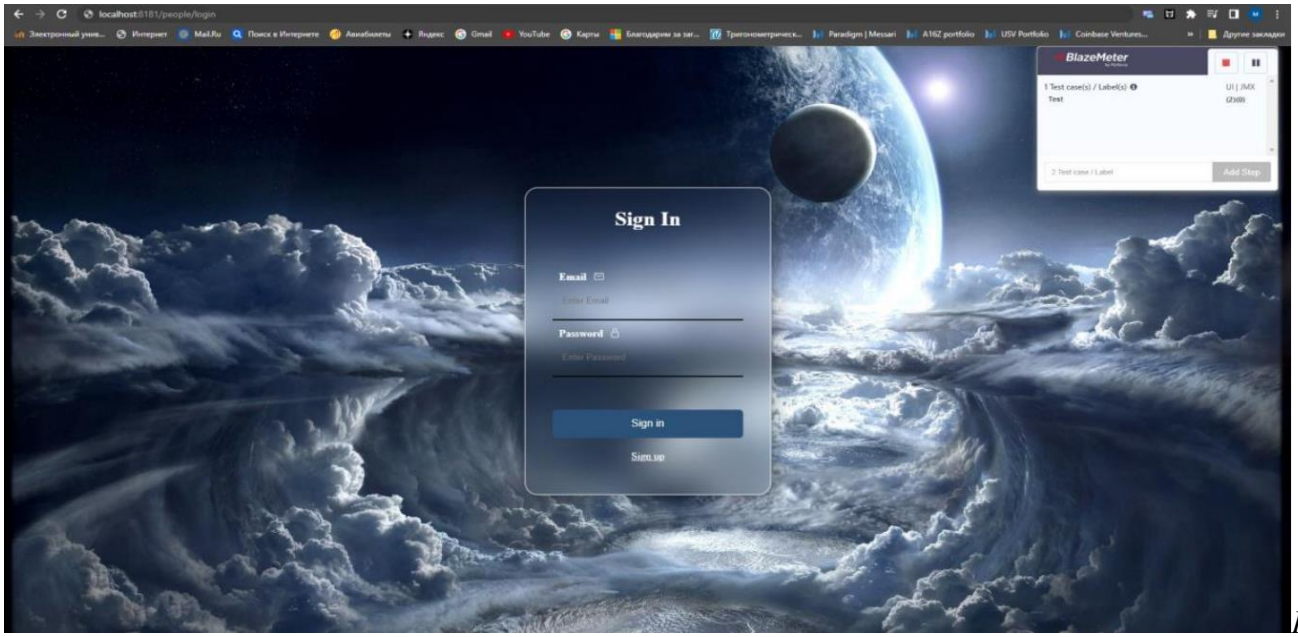


Рис. 1. Создание профилей виртуальных пользователей

Нужно пошагово совершить все действия обычного пользователя сайта, в то время как Blazemeter записывает HTTP-запрос. Когда профиль выполняется во время теста, на сервер посылается тот же самый запрос. В этом случае на сервер отправляется POST запрос с данными из формы входа.

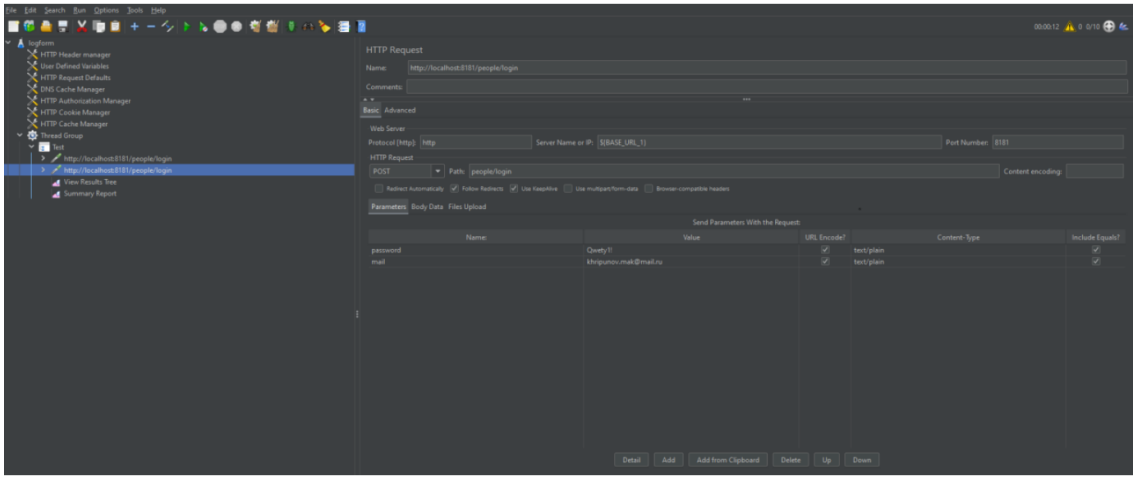


Рис. 2. План тестирования

В итоге получается план тестирования представленный выше.

3.4. Настройка нагрузки

Теперь можно более детально настроить тесты, например, указать количество потоков, задержка перед тем, как каждая группа начнет добавляться к выполнению теста, период нарастания для группы, время удержания группы перед остановкой, скорость отключения всех потоков указанной группы и т.д.

3.5. Запуск теста

После настройки можно запускать тест.

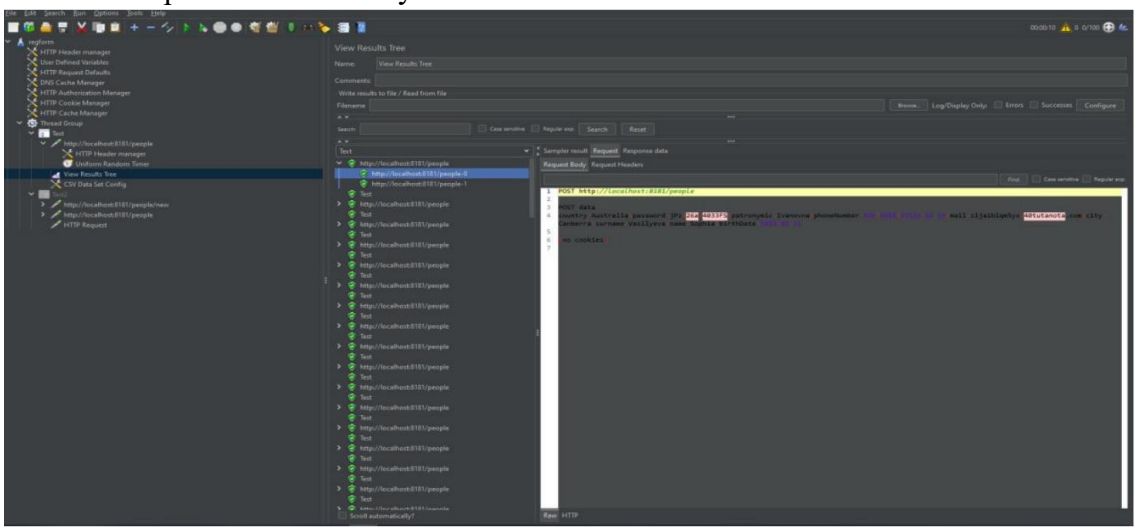


Рис. 3. Результаты теста

3.6. Анализ результатов

Рис. 4. Метрика запросов

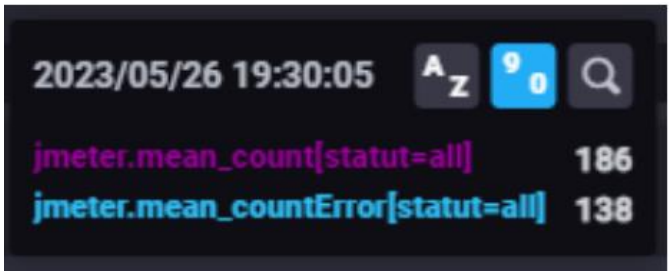


Рис. 5. Числовая характеристика метрики

Count - количество всех запросов

CountError - количество неудачных запросов

По результатам теста можно сказать, что количество ошибок равно 0.81% от общего числа запросов. Пришлись они на исход временного интервала, в котором нагрузка на приложение была максимальной.

3.7. Оптимизация и улучшение

Так как целью было узнать возможности приложения, которое находится на раннем этапе разработки, и посмотреть, как оно ведёт себя под нагрузкой, то дальнейшая работа над оптимизацией и улучшением будет проводиться по мере готовности самого приложения.

Заключение

В ходе работы было проведено нагрузочное тестирование. Целью проекта является проведение нагрузочного тестирования веб-сайта с помощью специального инструмента Apache JMeter.

Литература

1. Meier J.D. Web Application Performance Testing / J.D. Meier [и др.]. - Сан-Франциско : Microsoft Press, 2007. - 228 с.
2. Molyneaux I. The Art of Application Performance Testing, 2nd Edition / I. Molyneaux – Себастопол : O'Reilly Media, Inc., 2014. - 114 с.
3. Allspaw J. The Art of Capacity Planning / J. Allspaw - Себастопол : O'Reilly Media, Inc., 2008. - 80 с.

УДК 004.41

СОЗДАНИЕ ВИЗУАЛЬНЫХ ИНТЕРФЕЙСОВ ПРОГРАММНЫХ ПРОДУКТОВ С ПРИМЕНЕНИЕМ МАСШТАБИРУЕМОЙ РЕАКТИВНОЙ АРХИТЕКТУРЫ

А. А. Худяков

Воронежский государственный университет

Введение

Пользовательский интерфейс удобнее всего представлять в виде функции, принимающей данные (часть текущего состояния приложения) и возвращающей готовую для отрисовки структуру, обычно описанную на одном из языков разметки (XML, HTML и др.). Многие современные программные средства для создания интерфейсов основаны на идее реактивности – автоматического распространения изменений в данных ко всем местам, где эти данные используются, и последующего пересчета зависимых значений.

В данной статье предлагается новый подход для создания реактивных интерфейсов, в котором помимо данных и их отображения вводятся дополнительные примитивы – генератор сигналов, генератор данных и преобразователь данных. Новые примитивы позволяют эффективно обновлять и синхронизировать отображаемые данные между разными элементами интерфейса, делая возможным реализацию сложных приборных панелей и других нетривиальных задач, значительно упрощая их разработку и поддержку.

1. Проблемы актуальности и дублирования отображаемых данных

Большинство приложений с пользовательским интерфейсом имеют небольшое количество основных источников данных на странице. Одностраничные сайты и блоги часто являются статическими и получают все нужные данные еще на стороне сервера, интернет-магазины работают со списком товаров, системы бронирования – с текущими предложениями. Данные интерфейсе этих систем можно получить один раз и необязательно поддерживать в актуальном состоянии без непосредственного вмешательства пользователя.

Однако в ряде случаев, где информация обновляется в режиме реального времени, такой модели недостаточно. Примерами таких систем могут служить сводные панели индикаторов для отображения текущего состояния системы, интерфейсы биржевой торговли, моделирование и визуализация данных о каком-либо объекте для изучения его свойств.

Источников данных в подобных системах может быть очень много, а управление их обновлением и синхронизацией в реальном времени становится нетривиальной задачей. Если несколько блоков интерфейса по-разному визуализируют одни и те же данные, велика вероятность, что эти данные будут дублироваться, что приведет к дополнительной нагрузке и проблемам с производительностью. Более того, структура страницы может быть динамической, если блоки интерфейса и логика для их работы хранится в базе данных [1].

2. Формализация классической реактивной модели

В классической реактивной модели создания интерфейсов присутствуют 4 основных элемента – состояние (*State*), представление (*View*), действие (*Action*) и диспетчер изменений (*Dispatcher*). Представление (интерфейс) визуализирует состояние приложения и позволяет

пользователю с ним взаимодействовать. Взаимодействие (например, нажатие кнопки) может привести к вызову действия, которое будет обработано диспетчером. Диспетчер в зависимости от типа действия обновит состояние, что в свою очередь запустит повторную визуализацию, и цикл начинается снова.

Формализуем понятие состояния. Для упрощения будет считать, что любое состояние приложения может быть представлено в виде вектора из нулей и единиц произвольного размера *n*:

$$S = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix},\ s_j \in \{0,1\}$$

Введем понятие вектора визуализации – это вектор из нулей и единиц произвольного размера *m*, который уже может быть использован для отрисовки изображения. Например, так двоичным кодом можно представить содержание HTML-разметки страницы. Вектор визуализации задается следующим образом:

$$V = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix},\ v_j \in \{0,1\}$$

Формализуем понятие интерфейса. Интерфейс – это функция, которая преобразует вектор состояния в вектор визуализации:

$$I : S \to V$$

Диспетчер изменений состояния аналогично можно представить в виде функции, преобразующей вектор действия *A* и старое состояние S' в новый вектор состояния S''.

В данной модели в приложении может быть одно глобальное состояние, за изменением которого следят все блоки интерфейса, но, если блокам нужны разные данные для работы, каждый может иметь свое изолированное, локальное состояние. Второй поход намного лучше масштабируется с ростом количества строк кода в проекте, так как минимизирует изменения в уже написанной логике и позволяет поддерживать низкую связанность [2] модулей программы между собой. Однако, если несколько блоков по-разному визуализируют одно и то же состояние, это может приводить к ненужному дублированию больших объемов данных, что негативно сказывается на производительности.

3. Введение новых примитивов для масштабирования архитектуры

Введем понятие рабочей области – это изолированное окружение, в котором создаются реактивные примитивы. Примитивы из разных областей не могут взаимодействовать друг с другом, что соответствует принципу низкой связанности и позволяет разделять контексты выполнения, относящиеся к разным страницам, бизнес-доменам или просто группам блоков.

3.1. Генератор данных

Для эффективного использования данных и исключения их дублирования вводится генератор данных – примитив, который хранит в себе реактивное состояние и имеет функцию для его обновления. Это может быть асинхронная функция получения данных со стороннего сервера, либо функция, заданная каким-либо правилом для вычисления значений (например, генератор случайных чисел).

Генератор данных является единым источником истины для данных, которые определяет, и может использоваться другими примитивами для получения актуального состояния.

3.2. Генератор сигналов

Чтобы поддерживать актуальность нескольких генераторов данных и запускать обновление, вводится генератор сигналов. Сигнал может быть вызван вручную (например, при нажатии кнопки в интерфейсе) или задаваться временным интервалом – каждые 10 секунд, каждую минуту или любое другое значение.

3.3. Преобразователь данных

Преобразователь производит конвертацию данных из одного вида в другой. Это особенно полезно, если несколько блоков интерфейса визуализируют одну и ту же информацию, но делают это по-разному и с разной предобработкой. Вынесение этой логики в отдельный примитив позволяет не хранить логику конвертации в модулях самих элементов интерфейса, а также комбинировать несколько преобразователей между собой. Формализация преобразователя данных выглядит следующим образом:

$$C : S' \to S''$$

3.4. Комбинирование примитивов

Использование примитивов в разных комбинациях и объединение их в разные структуры в пределах одной рабочей области позволяет лучше планировать и декомпозировать задачи, а также более эффективно использовать вычислительные ресурсы.

В качестве примера рассмотрим информационную панель, которая визуализирует и сравнивает данные по продажам за день в единицах товара в двух торговых точках – старой и новой. Необходимо отобразить следующую информацию:

- совокупный объем продаж по двум точкам;
- разницу между продажами точек в абсолютных единицах;
- разницу между продажами точек в процентах;
- график продаж первой точки;
- график продаж второй точки.

Каждые 10 секунд последняя версия статистики должна загружаться из базы данных, а визуализация соответствующе обновляться. Структура интерфейса показана на рис. 1.

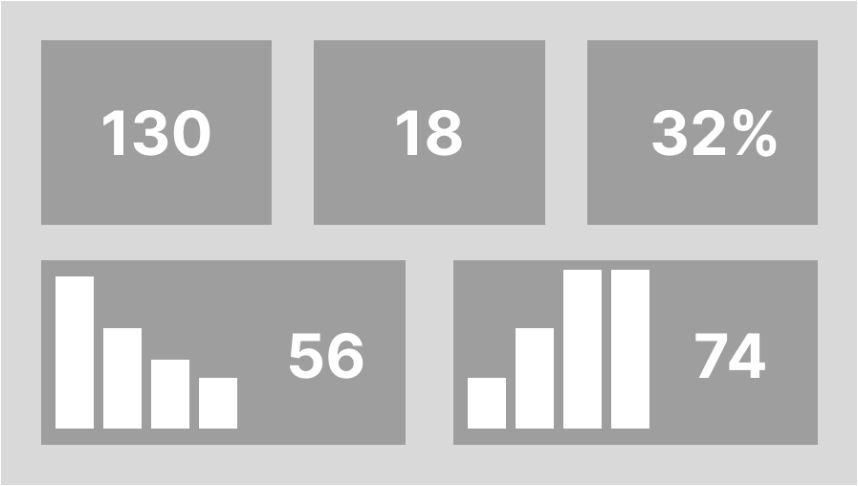


Рис. 1. Информационная панель с данными о продажах

Для реализации этого интерфейса с использованием новых примитивов понадобятся:

- генератор сигналов *S*, каждые 10 секунд подающий сигнал на генераторы данных;
- генератор данных *D1* о продажах первой точки;
- генератор данных *D2* о продажах второй точки;
- преобразователь *C1*, считающий сумму проданных в первой точке товаров;
- преобразователь *C2*, считающий сумму проданных во второй точке товаров;
- блок интерфейса *W1* для отображения совокупного объема продаж;
- блок интерфейса *W2* для отображения разницы в абсолютных единицах;
- блок интерфейса *W3* для отображения разницы в процентах;
- блок интерфейса *W4* с графиком продаж первой точки;
- блок интерфейса *W5* с графиком продаж второй точки.

Связи примитивов между собой образуют направленный ациклический граф вычислений и показывают, откуда берутся данные в интерфейсе приложения. Граф вычислений показан на рис. 2.

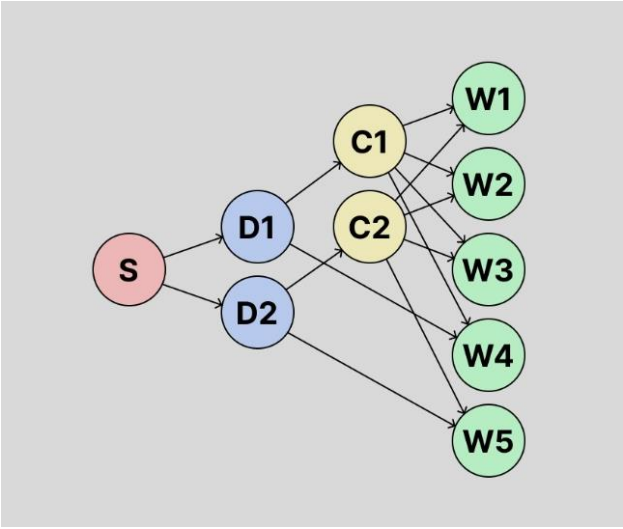


Рис. 2. Граф вычислений для интерфейса информационной панели

Благодаря графу вычислений и используемым примитивам было исключено дублирование данных, а обновления данных из разных источников синхронизированы. Также стало проще планировать, декомпозировать и оценивать задачу по созданию информационной панели.

Полученный граф зависимостей между примитивами можно легко программным образом визуализировать и использовать для отладки программы.

Заключение

Использующийся в данной статье подход к созданию масштабируемых реактивных пользовательских интерфейсов позволяет эффективно обновлять и синхронизировать отображаемые данные между разными блоками с заданной частотой. Он также помогает выстраивать в проекте правильную архитектуру, упрощает декомпозицию и отладку приложения, повышает производительность за счет повторного использования ресурсов и состоит из простых и понятных примитивов, создающих гибкую, легко визуализируемую структура зависимостей.

Литература

1. Худяков, А. А. Создание визуальных интерфейсов программных продуктов с применением динамически подключаемых модулей / А. А. Худяков // Математика, информационные технологии, приложения: сборник трудов межвузовской научной конференции молодых ученых и студентов. – 2023. – С. 528–531.

2. Ларман, К. Применение UML и шаблонов проектирования / К. Ларман – 2-е изд. – Москва: Вильямс, 2004. – 620 с.

УДК 519.713.5

Алгоритм группового взаимодействия искусственных агентов

У. В. Чеботарев

Воронежский государственный университет

Аннотация. В статье предложена модель интеллектуального агента для группового поиска в лабиринте на прямоугольном поле с препятствиями. Идея алгоритма опирается на критерий доверия, значение которого повышается при успехе и понижается при неудачном действии.

Ключевые слова: имногоагентные системы, интеллектуальный агент, маршрутизация

Введение

Одним из сложных видов распределенных систем являются многоагентные системы. Идейная основа для них – тезис о том, что решение ряда сложных технических задач может быть получено путем использования большой совокупности взаимодействующих между собой сравнительно простых индивидов, каждый из которых носит свой вклад в достижение некоторой глобальной цели. Помимо того, что такие групповые системы из агентов должны обладать свойствами надежности, масштабируемости, универсальности и т.д., неизбежно возникают эмерджентные эффекты, определяющие новое качество подобного рода комплексов. Данный подход имеет место в групповой робототехнике и нейронных сетях.

Таким образом, проблема создания эффективных принципов и моделей группового взаимодействия агентов в многоагентных системах, является актуальной. Статья посвящена обзору существующих подходов к построению методов группового взаимодействия агентов и иллюстрации такого взаимодействия на примере игры Pac-Man.

1. Основные понятия и определения многоагентных систем

Обычно агент – это некоторая абстрактная, программная сущность. П. Маес определяет автономных агентов как "вычислительные системы, которые населяют некоторую сложную динамическую среду, ощущают и действуют автономно в этой окружающей среде, реализуя множество своих целей и задач, для которых они разработаны" (1995). С другой стороны, у С. Рассела и П. Норвига агент понимается как некоторая сущность, воспринимающая среду с помощью своих датчиков и воздействующая на эту среду своими эффекторами (2010). Определение агента по М. Минскому (2018) формулируется следующим образом: агент – это любая совокупность элементов, которая способна действовать как единое целое вне зависимости от функций ее элементов. Если необходимо подчеркнуть биологическую инспирированность исследуемых принципов и механизмов, целесообразно использовать термин «анимат» – искусственную модель живого существа – животного. С. Вильсон (1986) определяет базовые характеристики такого животного, при этом это животное живет в мире сенсорных сигналов; оно способно к действиям, изменяющим эти сигналы; ряд сигналов является значимым для животного; деятельность животного направлена на оптимизацию уровня некоторых сигналов. Впервые о создании некой виртуальной сущности, моделирующей организацию поведения, упоминается в работах М. Бонгарда с коллегами (1975). В ней описывалась «попытка построить модель, воспроизводящую в грубых чертах поведение

человека или хотя бы «разумных» животных». Важно, что данный проект во многом опирался на проблему распознавания и обучения. В настоящее время тематика МАС развивается в работах В. Редько, К. Анохина, В. Карпова. В дальнейших рассуждениях будем использовать терминологию, предложенную В. Карповым, согласно которому термины «робот» и «агент» считаются синонимами, при этом если необходимо подчеркнуть технические аспекты, то используется термин «робот»; когда же больше интересует некоторое абстрактное, модельное поведение, используется термин «агент».

Многоагентной системой (МАС) называется совокупность или группа самостоятельных агентов, объединённых общей целью, которую невозможно достичь и реализовать ресурсами одного агента или результаты по достижению цели являются неудовлетворительными, например, по времени, или качеству, или другим критериям. Для успешного взаимодействия, а, значит, и реализации итоговой задачи, необходимо наличие коммуникации между такими интеллектуальными агентами. Возможны ситуации, при которых члены МАС могут не иметь единой цели, но их объединяет общая система, например, общая территория. Если в такой системе организовано взаимодействие между агентами, то она тоже может считаться МАС.

Одной из принципиальных особенностей, например, роевой робототехники является локальный характер взаимодействия роботов друг с другом, а также роботов со средой. Важнейшим видом взаимодействия является коммуникация. При этом коммуникация может быть как явной – в виде непосредственного обмена сообщениями между агентами, так и неявной. Примеры неявной коммуникации – это наблюдение друг за другом, оставление знака, который может быть найден и опознан другими (например, феромонный след муравья). В любом случае нас интересует, прежде всего, ситуация, когда каждый робот группы непосредственно взаимодействует лишь со своими соседями, находящимися в некоторой ограниченной зоне видимости. Отсюда обычно следует, что в такой системе роботы самостоятельно принимают решения о дальнейших действиях, опираясь на некоторые простые правила локального взаимодействия. Правила локального взаимодействия могут быть самыми разными, от сугубо формальных до весьма экзотических. Например, правила стайного движения основаны на модели пружин и амортизаторов. «Пружинная» составляющая модели определяет притяжение особей к лидеру, а «амортизационная» – отталкивание от лидера. В этой статье будем рассматривать, прежде всего, механизмы локального взаимодействия, основанные на том, что агенты могут обмениваться сигналами (сообщениями) с другими особями, находящимися в непосредственной близости.

Одним из вариантов программного обеспечения, позволяющего разрабатывать МАС, является среда NetLogo, в которой реализован широкий технический инструментарий для моделирования ситуаций и феноменов, происходящих в природе и обществе; для моделирования сложных, развивающихся во времени систем.

Многоагентный подход актуален для многих задач, и особое значение имеет его применение в задачах маршрутизации для моделирования группового согласованного движения группы агентов. В классическом случае задача группового движения сводится к составлению компактного контура некоторой геометрической фигуры, вмещающей в себя членов всей группы, но куда интереснее рассмотреть случай, где каждый такой объект – *интеллектуальный агент*, который определяет своё поведение самостоятельно, исходя из личного опыта, условий окружения и коммуникации с другими агентами. Примером использования данного подхода для решения задачи является метод группового взаимодействия агентов для нахождения своей цели в дискретном лабиринте.

2. Задача группового взаимодействия интеллектуальных агентов в игре «Pac-Man»

Для реализации модели взаимодействия интеллектуальных агентов с использованием инструментария NetLogo и решения задачи маршрутизации использовалась классическая аркадная игра Pac-Man (Рис. 1). В ней персонаж перемещается по лабиринту, избегая контакта с призраками и собирая разбросанные по лабиринту гранулы. Игровое поле дискретно и разбито на ячейки, которые могут или быть свободными, или содержать препятствие. Препятствия непроходимы как для игрока, так и для призраков. Перемещение доступно только по вертикали или по горизонтали, при столкновении игрока с призраком наступает поражение. Сам персонаж – единственный в своём роде и управляется игроком, а призраки управляются программно. Далее они будут реализовываться и восприниматься как интеллектуальные агенты.

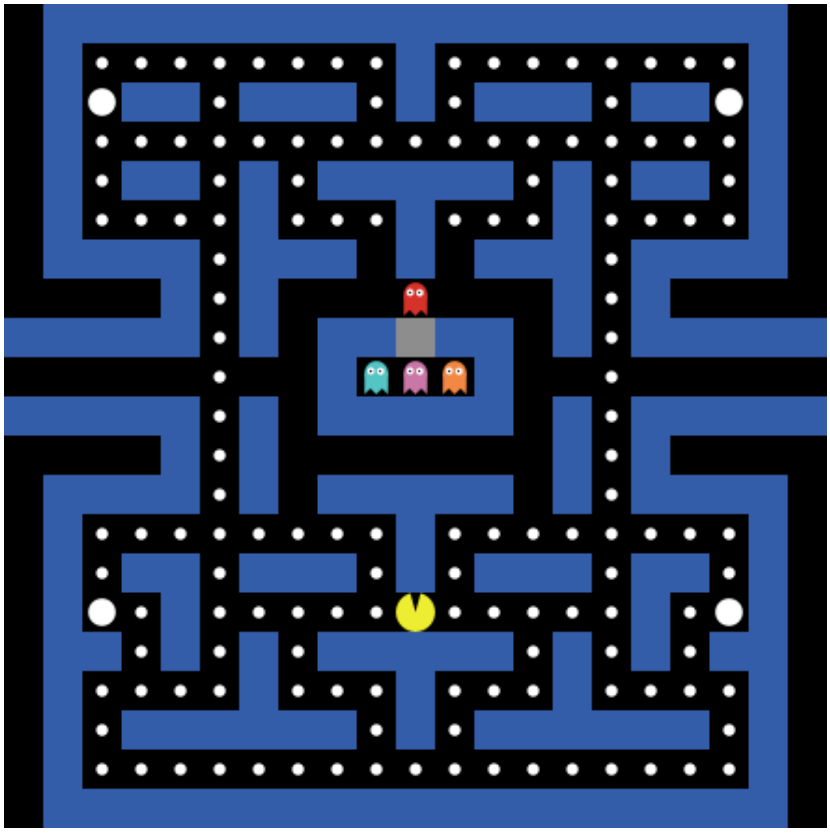


Рис. 1. Лабиринт в игре Pac-Man

Пусть имеется прямоугольное поле, разбитое на квадратные ячейки, по которому перемещается мобильный объект, способный двигаться горизонтально и вертикально. Размеры мобильного объекта строго меньше размера плитки, и в каждый момент времени он находится строго внутри одной из ячеек, поэтому, по сути, мобильный объект ассоциируется с одной из ячеек. Предположим, что на поле имеются статические препятствия, каждое из которых занимает ровно одну ячейку. В связи с этим, оказывается, что некоторые ячейки для объекта оказываются запрещенными. Препятствия не могут изменить никаких своих характеристик. На одной ячейке не может одновременно находиться более одного агента.

Интеллектуальный агент (он же мобильный объект) имеет своё поле обзора и некоторую изменяемую степень доверия к другим агентам. Если в поле обзора одного из агентов попадает другой, то они могут общаться. Призраки не запоминают карту местности, но если игрок попал

в их область видимости, то они сразу начинают преследование, пока он не скроется или не будет пойман. Игрок управляется оператором, он представляется исключительно как мобильный объект и не является интеллектуальным агентом. Необходимо разработать метод группового взаимодействия мобильных интеллектуальных агентов для обмена информацией при поимке игрока.

Пусть задано плоское прямоугольное поле $F = \{(x_i, y_j)\}_{n \times m}$, разбитое на $n \cdot m$ квадратных ячеек, каждая из которых задается своими координатами (x_i, y_i). На поле имеются препятствия $P = \{p_1, ..., p_r\}$, каждое p_k из которых задается координатами ячейки (x_i^k, y_j^k), в которой оно находится.

Мобильный объект может перемещаться в четырех направлениях по свободным ячейкам поля: вверх (↑), вправо (→), вниз (↓) и влево (←), но не может находиться в ячейке, занятой препятствием.

Агенты видят на расстояние N ячеек вертикально и горизонтально, если есть натуральное число N незанятых препятствиями, если же есть сторона с препятствием на расстоянии до агента, меньшем чем N, и это препятствие ближайшее к агенту по этой стороне, то в такую сторону обзор сократится до ближайшего препятствия. Агенты на расстоянии меньше N могут коммуницировать. Если в поле зрения агента попадает игрок, он следует за ним до поимки или до достижения последней ячейки, где видел игрока.

Решением задачи будет являться модель социального взаимодействия интеллектуальных агентов, которая позволит не использовать знание фактических координат игрока или препятствий, а только обмен информацией между агентами.

Моделью решения данной задачи может служить алгоритм обмена информацией между агентами для поимки игрока, который будет основан лишь на взаимодействии с агентами, находящимися в радиусе их видимости. Пусть каждый агент может передавать информацию, видит ли он кого-либо или нет, таким образом, если агент **A** видит игрока, а агент **B** видит агента **A**, то агент **B** последует за агентом **A** до тех пор, пока ему самому не попадётся игрок в поле зрения, или агент **A** не потеряет игрока, или, в свою очередь, **A** скроется от **B**.

Выделим следующие возможные ситуации, которые могут возникнуть при движении мобильного объекта во время нахождения игрока:

- если за игроком следует агент **A**, а агент **B** не видит игрока, но видит **A**, следующего за целью, тогда **B** последует за **A**, за **B** может последовать **C**, если он не видит игрока и агента **A**, но видит **B**, такая цепочка может продолжаться разрастаться;
- если агент, следуя за **B**, сам увидел **A**, тогда он будет следовать за **A** самостоятельно;
- если **C** видит различных агентов, которые движутся в сторону игрока, он выбирает того, кто ближе всех к тому, кто видит игрока;
- агент может стоять на месте или двигаться хаотично, если не находится в состоянии преследования;
- если агент не имеет соседних свободных полей, он будет находиться на месте, пока не появится возможности двигаться;
- если агенты двигаются друг навстречу другу, то они могут поменяться местами.

3. Решение поставленной задачи

3.1. Описание алгоритма

Для организации социального группового взаимодействия предлагается ввести единственный критерий «доверие», который имеет случайное изначальное значение от 0 до натурального числа P для каждого призрака на старте игровой сессии. Это число будет отражать то, в насколько длинную очередь видящих лишь впереди идущего агента данный индивид готов вступить и начать преследование. Если длина последовательности агентов превышает это ненулевое число, то такие преследователи агентом будут игнорироваться.

Значение критерия «доверие», равное нулю, в свою очередь, будет означать асоциальный настрой агента. Такой индивид, увидев любой продолжительности очередь, незамедлительно начнёт двигаться в любом направлении от агентов, состоящих в очереди. Но, если асоциальный агент сам увидит игрока, он так же будет сигнализировать окружающим следовать за собой. Все агенты получают случайное значение критерия «доверие» при своей инициализации по нормальному закону распределения. Под хаотичным движением рассматривается случайный выбор последующего направления движения, одно из четырёх направлений выбирается по равномерному закону распределения, если же все соседние ячейки недоступны, агент останется на месте. На рис. 2 представлен разработанный алгоритм поведения игрока и призраков.

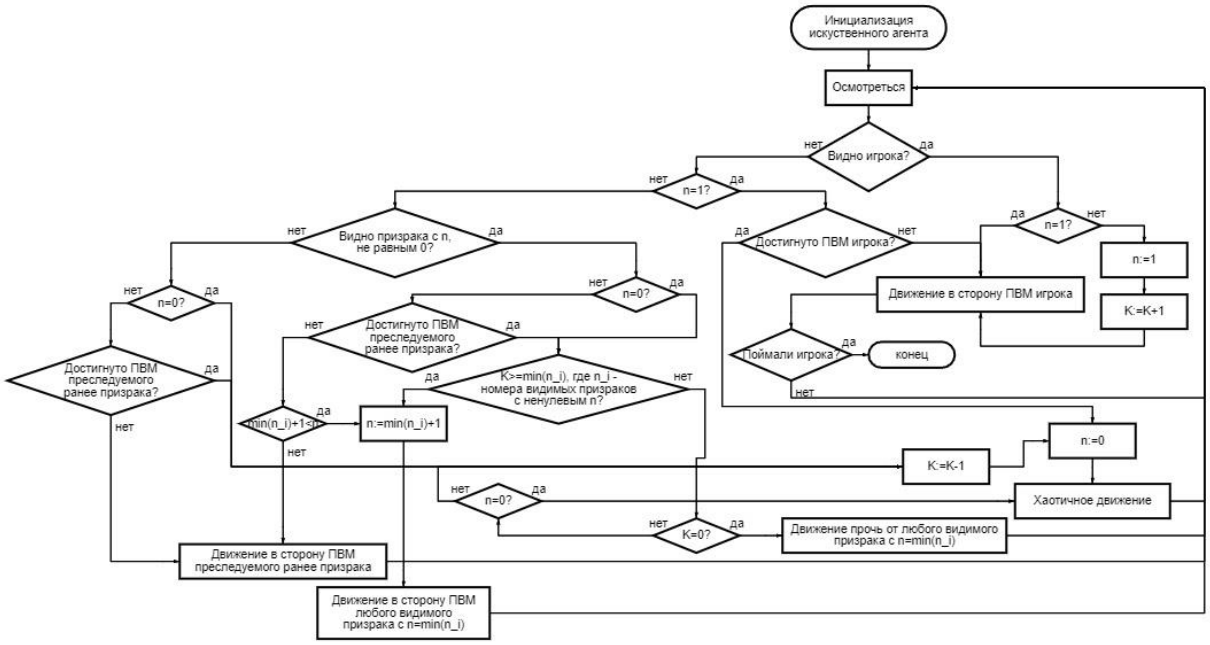


Рис. 2. Алгоритм взаимодействия агентов в игре Pac-Man

В алгоритме используются следующие обозначения: n – номер в очереди, если он нулевой, то объект не состоит в очереди, ПВМ (последнее видимое место) – последняя ячейка, на который находился преследуемый мобильный агент до выхода из поля видимости, K – значение критерия «доверие», призрак – интеллектуальный мобильный агент, реализующий предложенную модель поведения.

В алгоритме считается, что устройство агентов одинаково. Алгоритм начинается с момента появления агента и заканчивается поимкой игрока любым из призраков. Важно отметить, что ниже минимального и выше максимального значение критерия «доверие» опускаться/подниматься не должно. Таким образом, группа, следующая в очереди, сможет

покрыть большую площадь при потере игрока для его поисков, а асоциальные агенты будут мешать игроку специально выстроить легко прогнозируемую очередь, следующую за ним.

На данный момент архитектура устройства агента отличается простотой, но её можно разнообразить, добавив новые критерии, например, статус, сколько агент вёл очередь за собой правильно, создать индивидов-обманщиков, специально дающих ложные сигналы другим агентам с целью понизить их статус и(или) доверие. Можно заменить игрока на игровой объект, который будет сам избегать призраков, тогда было бы интересно использовать обучение с подкреплением для новых объектов.

Заключение

Предложенный алгоритм описывает примитивную логику поведения агента, но уже этого достаточно для того, чтобы заставить игрока подстраивать свою стратегию поведения под других агентов, а отсутствие общего алгоритма движения, (например, A*), усложняет распознавание действий интеллектуальных объектов, бег кругами не поможет. Кроме того, при решении данной задачи использовался инструмент NetLogo, который при дальнейшем развитии проекта и новых реализациях придется заменить на более высокоуровневый, так как без гибкой настройки классов довольно трудно сделать агента со сложной логикой социального поведения.

Литература

1. Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – Москва : Мир, 1978. – 427 с.

2. Леденева, Т. М. Некоторые алгоритмы прикладной теории графов / Т.М. Леденева. – Воронеж : Издательский дом ВГУ, 2021. – 35 с.

3. Wilensky, U. (2001). NetLogo Pac-Man model. http://ccl.northwestern.edu/netlogo/models/Pac-Man. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

4. Карпов В. Э. Методы группового управления искусственными агентами на основе биологически инспирированных моделей поведения : дис. докт. техн. наук. / В.Э. Карпов – Москва : НИИ ИТ, 2020. – 311 с.

5. Варшавский В.И. Коллективное поведение автоматов. / В. И. Варшавский – Москва: Наука, 1973. 408 с

УДК: 004.08+004.048

Проблема планирования работы гидроэлектростанций

М.А. Чекменев

Воронежский государственный университет

Введение

Современное общество в значительной степени зависит от функционирования различных электронных устройств и систем. Электроника используется для приготовления пищи, отопления, связи, управления различными системами безопасности и другими необходимыми или даже жизненно важными системами, использующими электричество (например, медицинское оборудование). В данном отчёте будет представлено решение М.В.Ф. Перейры и Л.М.В.Г. Пинто проблемы планирования энергоснабжения. Цель оптимальной работы гидротермальной системы заключается в обеспечении энергетических потребностей в заданный период времени при минимизации затрат на эксплуатацию. Затраты на эксплуатацию в данном случае включают в себя стоимость топлива для термальной генерации электроэнергии и штрафы за недостаточную подачу энергии за определённый период времени.

1. *Проблемы планирования*

1.1. Оценка потребности электроэнергии в будущий период

Как уже упоминалось, наличие достаточного запаса энергии может быть жизненно важным для индивидуума или общества в целом. Хотя, эта задача не является невозможной [6], сама по себе задача оценки того, сколько и когда потребуется электроэнергии, представляет собой вызов для современных математиков.

1.2. Прогнозирование погоды

Поскольку работа гидроэлектростанции в основном сводится к решению о том, сколько воды следует выпустить (если вообще выпускать), чтобы использовать её для генерации энергии, возникает вопрос: "Можем ли мы надёжно предсказать, когда и сколько воды поступит в резервуар?" Ответ, опять же, нет.[5]

1.3. Поиск оптимальной стратегии управления

Ни одно из ограничений решения, включая верхние и нижние пределы уровней водохранилищ и векторы оттока, не может быть нарушено.

1.4. Минимизация затрат

Кроме того, приемлемое решение этой задачи оптимизации должно минимизировать затраты на эксплуатацию генераторов (которые являются нелинейными). В то же время затраты на вычисления должны оставаться разумно низкими.

Обратите внимание, что некоторые критически важные данные, необходимые для принятия решений, такие как потребность в энергии и будущие погодные условия, не могут быть предсказаны с высокой степенью уверенности. Именно поэтому задача оптимизации будет сформулирована как стохастическая проблема.

2. Формулировка задачи

2.1. Теория двойственности

Прежде чем решать задачи двойственного динамического программирования, мы продемонстрируем достоверность метода, доказав несколько результатов о двойственности линейных задач.

Теорема слабой двойственности

Если x *и* y *являются допустимыми решениями прямой и двойственной задач соответственно, тогда* $cT\ x\ \leq\ bT\ y$.

Доказательство слабой двойственности

Доказательство в приложении после Словаря.

Теорема сильной двойственности

Для прямой линейной задачи

$$\begin{aligned} &\text{Макс} && c^T x \\ &\text{при условии} && Ax \leq b. \end{aligned} \tag{1}$$

Если x^* *является оптимальным решением задачи в стандартной форме, тогда оптимальное решение*

$$\begin{aligned} &\text{Мин} && b^T y \\ &\text{при условии} && A^T y \geq c \end{aligned} \tag{2}$$

Является $y^* = B^{-T} c_B$, *где* B^{-T} — *обратно транспонированная матрица* B *(матрица коэффици-ентов, соответствующих базисным переменным). Оптимальные значения целевых функций равны:* $c^T x^* = b^T y^*$.

Доказательство сильной двойственности

Доказательство в приложении после Словаря.

2.2. Двойственное динамическое программирование (Детерминированный случай)

Для лучшего понимания стохастической модели сначала рассмотрим детерминированный случай, т. е. будущие события не случайны. Рассмотрим следующую задачу оптимизации:

$$\begin{aligned} &\text{Мин} && c_1x_1 + c_2x_2 \\ &\text{при условии} && A_1x_1 \geq b_1, \\ & && E_1x_1 + A_2x_2 \geq b_2. \end{aligned} \tag{3}$$

Задача (6) может быть решена путем разложения на две меньшие линейные задачи следующим образом:

1. **Этап 1**

$$\begin{aligned} &\text{Мин} && c_1x_1 + \alpha_1(x_1) \\ &\text{при условии} && A_1x_1 \geq b_1; \end{aligned} \tag{4}$$

2. **Этап 2**

$$\begin{aligned} \alpha_1(x_1) = &\text{Мин} && c_2x_2 \\ &\text{при условии} && A_2x_2 \geq b_2 - E_1x_1, \end{aligned} \tag{5}$$

где $\alpha_1(x_1)$ представляет будущую стоимость решения x_1 и может быть представлена как кусочно-линейная функция.[2]

Мы выбираем набор испытательных допустимых значений для x_1. Затем решаем двойственную задачу второго этапа для оптимального решения, используя испытательные значения $\hat{x}_1$, полученные на первом этапе. Таким образом, мы получаем приближения функций будущих затрат. Теперь мы решаем приблизительную задачу первого этапа.

А затем вычисляем нижнюю границу $z_{min} = c_1\hat{x}_1 + \hat{\alpha}_1$ и обновляем верхнюю границу $z_{max} = c_1\hat{x}_1 + \alpha(x_1)$. Обратите внимание: что наши начальные границы могут быть установлены как $\pm\infty$ соответственно.

По сути, на каждой итерации алгоритм обновляет верхние и нижние границы оптимального решения исходной задачи и завершается, когда разница между границами находится в пределах приемлемого уровня толерантности, т. е. $z_{max} - z_{min} \leq \epsilon$ для некоторого ϵ нашего выбора.

3 Стохастическое Двойственное Динамическое Программирование (SDDP)

3.1. Улучшение качества обслуживания

Одной из привлекательных особенностей алгоритмов динамического программирования (DP) является их способность обрабатывать стохастические задачи. Следовательно, подход DDP, описанный в предыдущем разделе, также может быть расширен на стохастический случай. Это будет проиллюстрировано на следующей двухэтапной задаче:

$$\begin{aligned} \text{Мин} \quad & c_1x_1 + p_1c_2x_{21} + p_2c_2x_{22} + \ldots + p_mc_2x_{2m} \\ & A_1x_1 \geq b_1, \\ & E_1x_1 + A_2x_{21} \geq b_{21}, \\ \text{при условии} \quad & E_1x_1 + A_2x_{22} \geq b_{22}, \\ & \vdots \\ & E_1x_1 + A_2x_{2m} \geq b_{2m}. \end{aligned} \tag{6}$$

Здесь x_1 — решение на первом этапе, а $x_{21},x_{22},\ldots,x_{2m}$— решения на втором этапе, которые зависят от реализации неопределенных параметров $p_1,p_2,\ldots,p_m$. Эти параметры представляют собой вероятности, с которыми возникают различные сценарии, и соответственно влияют на целевую функцию и ограничения.

Приведенная выше задача может быть интерпретирована как решение x_1, принятое на первом этапе. Исходя из испытательного решения x_1, на втором этапе возникнет m подзадач:

$$\begin{aligned} \alpha_{1j}(x_1) = & \text{Мин} \quad c_2x_{2j} \\ & \text{при условии} \quad A_2x_{2j} \leq b_{2j} - E_1x_1. \end{aligned} \tag{7}$$

Для всех $j = 1, \ldots, m$. Цель состоит в минимизации $c_1x_1 + E\left(\sum_{j=1}^{m} p_jc_2x_{2j}\right)$, что является суммой затрат первого этапа и ожидаемых затрат второго этапа. Обратите внимание, что каждый p_j представляет собой вероятность каждой подзадачи. Сумма вероятностей равна 1, т.е. $\sum_{j=1}^{m} p_j = 1$.

Исходная задача оптимизации, представленная (6), может быть решена с помощью стохастической рекурсии DP, где будущие функции затрат будут даны формулой:

$$\overline{\alpha_1}(x_1) = \sum_{j=1}^{m} p_j\alpha_{1j}(x_1).$$

Здесь мы используем тот же концепт, который был представлен в дискретном случае:

- Определить набор испытательных решений $\{\widehat{x_{ti}}: i = 1,\ldots,n; t = 1,\ldots,T\}$
- Повторять для каждого t:
- Повторять для каждого сценария b_{ti}
- Решить задачу оптимизации:

$$\begin{aligned} & \text{Мин} \quad c_tx_t + \bar{\alpha}_t(x_t) \\ & \text{при условии} \quad A_tx_t \geq b_{ti} - E_{t-1}\hat{x}_{t-1i}. \end{aligned} \tag{8}$$

для t, $\hat{x}_{t-1i}$ и b_{ti}.

- Аппроксимировать функцию будущих затрат для этапа $\overline{\alpha_{t-1}}(x_{t-1})$

- Вернуться к первому шагу.
- Нужно отметить, что возможно избежать повторения прямого моделирования для каждой комбинации сценариев b_{tj}, выполнив Монте-Карло прямое моделирование для выборки сценариев [2], что снижает вычислительные затраты.

Заключение

"Методология для решения многоступенчатых стохастических задач оптимизации, основанная на аппроксимации функций ожидаемой стоимости оставшихся операций стохастического динамического программирования с помощью кусочно-линейных функций. Не требуется дискретизация состояний, и избегается комбинаторный «взрыв» с увеличением числа состояний (известное как «проклятие размерности» динамического программирования). Кусочно-линейные функции получаются из двойственных решений оптимизационной задачи на каждом этапе и соответствуют разрезам Бендерса в рамках стохастической, многоступенчатой декомпозиционной схемы."[2] Мы можем заключить, что решение М.В.Ф. Перейры и Л.М.В.Г. Пинто по планированию энергоснабжения является оптимальным, поскольку оно было принято для промышленного использования. [3] Следует отметить, что ни один из элементов представленной методологии не специфичен для гидроэлектростанций, что подразумевает возможность его применения для планирования других источников энергии, таких как ветрогенераторы или солнечные панели.

Неучтенные аспекты:
При планировании энергоснабжения также необходимо учитывать альтернативный план на случай, если некоторые из генераторов в системе не смогут достичь желаемой мощности. Системы, требующие таких мер безопасности, могут быть такими большими, как сеть страны, или такими маленькими, как одна электростанция.

Словарь

- π таково, что $B^T\pi = c_B$
- $t \in [1, T]$ — индекс этапа для планирования горизонта H
- x_t — вектор состояния в начале этапа t
- $\alpha_t(x_t)$ — стоимость операции с состояния t
- u_t — вектор решений для этапа t
- $c_t(u_t)$ — немедленные затраты, вызванные решением ut
- $\alpha_{t+1}(x_{t+1})$ — будущие затраты, вызванные решением $xt + 1$
- $\beta \in (0; 1)$ — коэффициент дисконтирования будущих решений

Приложение

Доказательство слабой двойственности

$$
\begin{aligned}
& c^T x = x^T c \\
& \leq x^T (A^T y) \\
& = (x^T A^T) y \\
& = (Ax)^T y \\
& \leq b^T y,
\end{aligned}
\tag{9}
$$

где на второй строке используется $c \leq ATy \Rightarrow y$ является допустимым, и $Ax \leq b \Rightarrow x$ является допустимым на последней строке.

Доказательство сильной двойственности

Сначала вводим вспомогательные переменные s и z в прямую и двойственную задачи соответственно, чтобы привести их к стандартной форме.

Для прямой линейной задачи:

$$
\begin{aligned}
\text{Макс} \quad & c^T x \\
& Ax + s = b, \\
\text{при условии} \quad & x \geq 0, \\
& s \geq 0.
\end{aligned}
\tag{10}
$$

Для двойственной линейной задачи:

$$
\begin{aligned}
\text{Мин} \quad & b^T y \\
& A^T y - z = c, \\
\text{при условии} \quad & y \geq 0, \\
& z \geq 0.
\end{aligned}
\tag{11}
$$

Теперь,

$$
c^T x = x^T c = x^T (A^T y - z) = (Ax)^T y - x^T z = (b - s)^T y - z^T x = b^T y - s^T y - z^T x.
$$

Поскольку все из x, s, y и z неотрицательны, $s^T y \geq 0$ и $z^T x \geq 0 \Rightarrow c^T x \leq b^T y$ (по теореме слабой двойственности). Далее мы хотим показать, что для оптимального решения прямой задачи можно построить вектор y, так что $s^T y = z^T x = 0$, следовательно $c^T x = b^T y$ (эквивалентность целевых функций).

Предположим, x^* — оптимальное базисное решение для прямой задачи с пониженными затратами $\widehat{c_N} \leq 0$, удовлетворяющее $N^T \pi = c_N - \widehat{c_N}$,

$$\begin{vmatrix} B^T \\ N^T \end{vmatrix} * \pi = \begin{vmatrix} c_B \\ c_N \end{vmatrix} - \begin{vmatrix} 0 \\ \hat{c_N} \end{vmatrix}.$$

Рассматривая строки, соответствующие оригинальным переменным, мы видим, что $AT\pi = c + z \Rightarrow AT\pi - z = c$, где, поскольку пониженные затраты $\widehat{c_N}$ неположительные, компоненты z, $z_i = 0 : i \in B$ (множество базисных индексов) и $z_i = -\hat{c_i} \geq 0$ в противном случае. Следовательно, $z \geq 0$ и π удовлетворяют двойственной задаче $\Longrightarrow \pi$ допустим для двойственной задачи.

Для оптимального значения x^* прямой задачи рассмотрим значения вспомогательных переменных $s_i : i = 1, \ldots, m$. Поскольку x^* допустим, $s_i \geq 0$. Если $s_i > 0$, то это базовая переменная. В противном случае, $s_i = 0 \Longrightarrow s^T y = 0$. Для x^*, $x_i > 0$ $\Rightarrow$ это базовая переменная, в противном случае, $x_i = 0$ $\Rightarrow z^T x = 0$ и, поскольку $s^T y = 0$, $c^T x = b^T y$. Поскольку $b^T y = c^T x^*$ для $y = \pi$, не существует допустимого значения y с большей целевой стоимостью. Следовательно, $y^* = \pi$ решает двойственную задачу. Таким образом, обе задачи, двойственная и прямая, имеют одно и то же оптимальное решение**.** [4]

Литература

1. Перейра М.В.Ф. Оптимальное стохастическое планирование операций на крупных гидроэлектрических системах // International Journal of Electrical Power & Energy Systems. – 1989. – Т. 11, № 3. – С. 161–169.
2. Перейра М.В.Ф., Пинто Л.М.В.Г. Многоступенчатая стохастическая оптимизация, примененная к планированию энергии // Mathematical Programming. – 1991. – Т. 52, № 1-3. – С. 359–375.
3. Презентация EMS. Исследование случая 3: Предпринимательство в математических науках (2021-2022) [SEM1] / Университет Эдинбурга. – 2021.
4. Лекционные материалы по дисциплине "Линейное программирование, моделирование и решение" (LPMS), MATH10073 / Университет Эдинбурга. – 2021.
5. Пауэлл У. Стохастические оптимизационные проблемы в энергетике [Электронный ресурс]. – 2016. – Режим доступа: https://www.youtube.com/watch?v=dOIoTFX8ejMamp;list=WLamp;index=197amp;abchannel=CompSustNet, свободный. – Загл. с экрана. – Дата доступа: 2021.
6. Ю Ш., Вэй Й.-М., Ван К. Оптимальная модель PSO–GA для оценки первичного энергопотребления Китая // Energy Policy. – 2012. – Т. 42. – С. 329-340.

УДК 004.4

СОЗДАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ КОРПОРАТИВНОГО УВЕДОМЛЕНИЯ СОТРУДНИКОВ

Р. А. Черваков, Е. В. Трофименко

Воронежский государственный университет

Введение

Современные организации стремятся к эффективной коммуникации и управлению информацией среди своих сотрудников. В данной статье рассматривается разработка веб-приложения информационной системы, предназначенной для корпоративного уведомления, создания заданий и замечаний для отслеживания и оценки работы сотрудников.

1. Постановка задачи

Необходимо разработать веб-приложение информационной системы, предназначенной для корпоративного уведомления, и которая должна удовлетворять следующим требованиям:

1. Должны быть разделены роли пользователей:
   - общие пользователи, которые обладают функциями создания и редактирования уведомлений;
   - функциональные пользователи, которые могут только просматривать уведомления и помечать их как ознакомленные.
2. Виды уведомлений, должны быть следующего типа:
   - документация;
   - объявления;
   - сообщения;
   - задания;
   - замечания.
3. Должна быть реализована адресация уведомлений:
   - замечания, задания и сообщения могут быть адресованы конкретным сотрудникам или группам сотрудников;
   - документация и объявления адресованы подразделениям.
4. Все сотрудники должны быть закреплены к подразделению организации, в котором они работают.
5. Ко всем типам уведомлений можно прикреплять файлы.
6. В замечаниях, заданиях, сообщениях и объявлениях можно ссылаться на уведомления типа «документация».
7. Функциональные пользователи могут просматривать уведомления глобально.
8. Общие пользователи могут просматривать только уведомления, адресованные им или подразделению организации, к которому они относятся.
9. Требования к интерфейсу:
   - система должна предоставлять удобный интерфейс для создания, редактирования и просмотра уведомлений;
   - реализовать возможность фильтрации уведомлений по типу, статусу, дате и подразделению.

10. Безопасность:
    - обеспечить безопасность системы с использованием аутентификации и авторизации;
    - запись действий пользователей для отслеживания изменений и обеспечения безопасности.
11. Реализовать систему уведомлений для пользователя о новых уведомлениях и приближающихся сроках исполнения.
12. Система разрабатывается на базе Flask [1] в качестве серверной технологии, а для фронтенда используются JavaScript, jQuery [2], HTML и Bootstrap CSS [3].
13. Веб-приложение должно быть реализовано с использованием архитектурного подхода MVC (Model-View-Controller) для обеспечения модульности и удобства поддержки.
14. Для хранения данных можно использовать реляционную базу данных, такую как PostgreSQL [4] или SQLite.

2. Анализ функциональности системы

Данная информационная система позволяет регистрировать два типа пользователей: адресат (общий пользователь), уведомитель (функциональный пользователь).

Пользователь типа «адресат» не является отдельным человеком, это единый пользователь для целого подразделение организации. «Адресат» видит в системе два раздела: «для всех» и «отдельным сотрудникам». В разделе «для всех» сотрудники могут ознакомится с общей документацией и объявлениями, опубликованными для их подразделения. В разделе «отдельным сотрудникам», конкретный сотрудник может воспользоваться поиском или выбрать свое имя из списка и ознакомится с уведомлениями, адресованными ему лично.

Пользователь типа «уведомитель» — это отдельный человек, например, руководитель организации или подразделения, проверяющий или любой другой сотрудник организации, обладающий правами «уведомителя». «Уведомитель» может создавать и редактировать уведомления для адресатов, а также просматривать все не защищенные или созданные им уведомления в системе используя фильтры поиска различные для каждого типа уведомлений. В системе существует пять типов уведомлений.

Тип уведомлений «документация» — это общие уведомления адресованные конкретному или нескольким подразделениям организации. Они имеют название и должны обязательно содержать прикрепленные файлы, а также могут содержать небольшое описание этих файлов. Такие уведомления могут быть помечены флажком «важное» и могут быть сортированы по папкам. Уведомления этого типа не могут быть защищены паролем.

Тип уведомлений «объявление» — это общие уведомления адресованные конкретному или нескольким подразделениям организации. Они имеют название, должны обязательно содержать текстовое сообщение, могут содержать прикрепленные файлы и ссылки на уведомления типа «документация». Такие уведомления могут быть помечены флажком «важное». Уведомления этого типа не могут быть защищены паролем.

Тип уведомлений «сообщение» — это частные уведомления адресованные отдельному или нескольким сотрудникам организации. Они должны обязательно содержать текстовое сообщение, могут содержать прикрепленные файлы, ссылки на уведомления типа «документация». Такие уведомления могут быть помечены флажком «важное». Уведомления этого типа могут быть защищены паролем.

Тип уведомлений «задание» — это частные уведомления адресованные отдельному или нескольким сотрудникам организации. Они должны обязательно содержать текстовое сообщение, могут содержать прикрепленные файлы, ссылки на уведомления типа

«документация». Такие уведомления могут иметь срок исполнения и могут быть помечены флажками: «выполнено», «не выполнено». Уведомления этого типа могут быть защищены паролем.

Тип уведомлений «замечание» — это частные уведомления адресованные отдельному или нескольким сотрудникам организации. Они должны обязательно содержать текстовое сообщение, могут содержать прикрепленные файлы, ссылки на уведомления типа «документация». Такие уведомления могут иметь срок исполнения и могут быть помечены флажками: «исправлено», «не исправлено», а также могут быть классифицированы по видам. Уведомления этого типа не могут быть защищены паролем.

3. Разработка веб-приложения

3.1. Структура базы данных

В проектируемой информационной системе, где пользователь не ограничивается индивидуальным лицом, а может охватывать целые группы людей, структура базы данных предусматривает две отдельные таблицы. Первая таблица «person» отражает конкретного сотрудника и содержит информацию о его личных данных. Вторая таблица «user» описывает сущность пользователя, к которому привязан данный сотрудник, позволяя эффективно управлять группами пользователей и их доступом к функциональности системы.

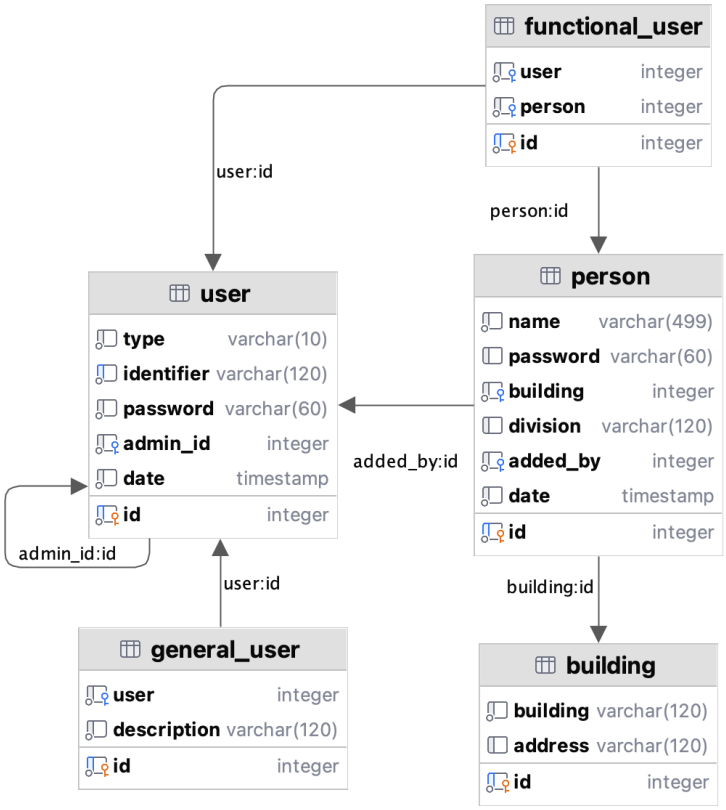


Рис. 1. UML диаграмма, отображающая представление данных о пользователе зарегистрированном в информационной системе

Система предусматривает упорядоченное разделение уведомлений на различные типы, каждый из которых обладает сходной, но индивидуальной структурой данных. Это позволяет обеспечить гибкость функционала системы, сохраняя при этом ее эффективность и удобство использования. Для достижения этой цели таблицы уведомлений должны быть автономными, в то время как другие дополнительные структуры данных привязаны к ним [4].

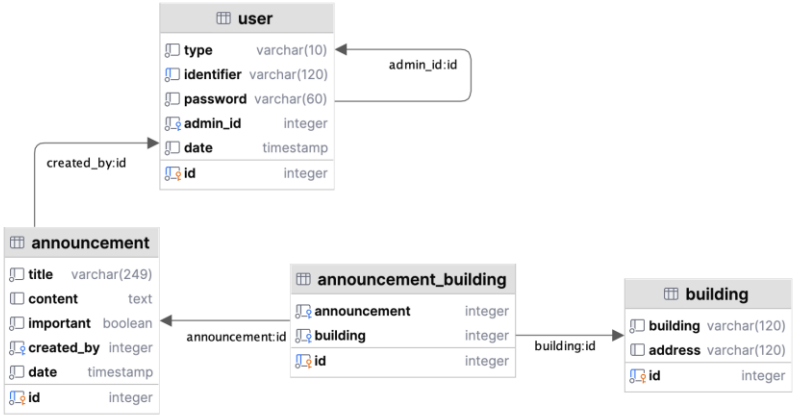


Рис. 2. UML диаграмма отображающая связь общего уведомления типа «объявление» с подразделением, которому оно адресовано

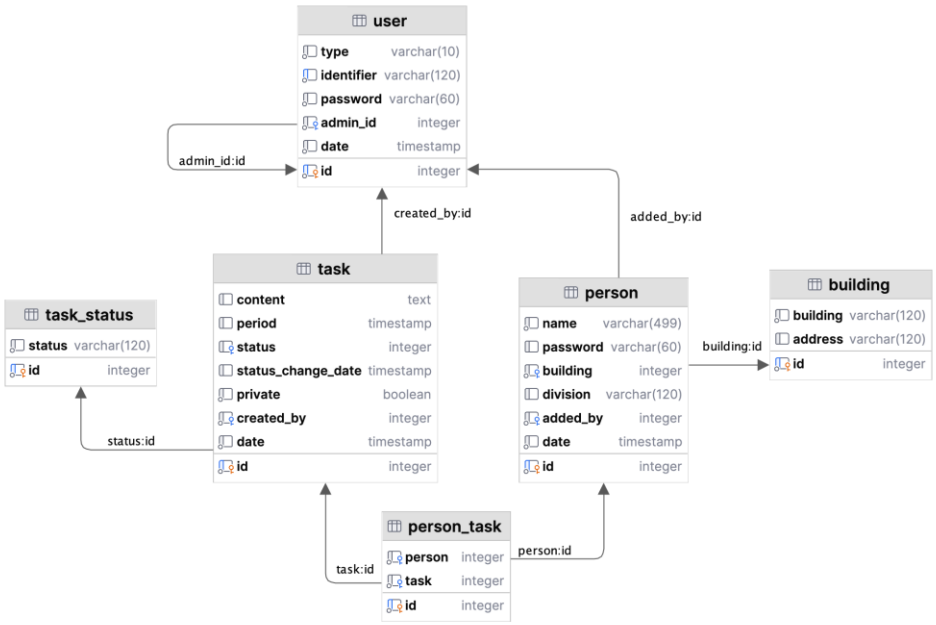


Рис. 3. UML диаграмма отображающая связь частного уведомления типа «задание» с сотрудником, которому оно адресовано

В качестве примера, общие уведомления связаны с определенными подразделениями, представленными в таблице «building», тогда как частные уведомления привязаны к индивидуальным сотрудникам, описанным в таблице «person». Кроме того, все уведомления связаны с таблицей «user», с которой в свою очередь ассоциируются таблицы «functional_user» и «person», указывая на идентификацию «уведомителя», который создал данное уведомление.

Приведенные архитектурные решения обеспечивают гибкость и масштабируемость базы данных, соответствуя сложности и потребностям современных организаций.

3.2. Пользовательский интерфейс

Интерфейс веб-приложения был разработан с акцентом на минимальное количество переходов между страницами, представляя собой динамическую одностраничную платформу, где области изменяются в зависимости от раздела, выбранного пользователем [3]. Для реализации такой клиент-серверной архитектуры применяются методы jQuery — AJAX [2].

Основой интерфейса является центральная лента уведомлений, которая обеспечивает наглядное представление последних событий и действий. На левой панели размещаются индивидуализированные элементы управления для различных типов пользователей, а справа доступны функциональные возможности информационной системы, подстраивающиеся под потребности и роли пользователя.

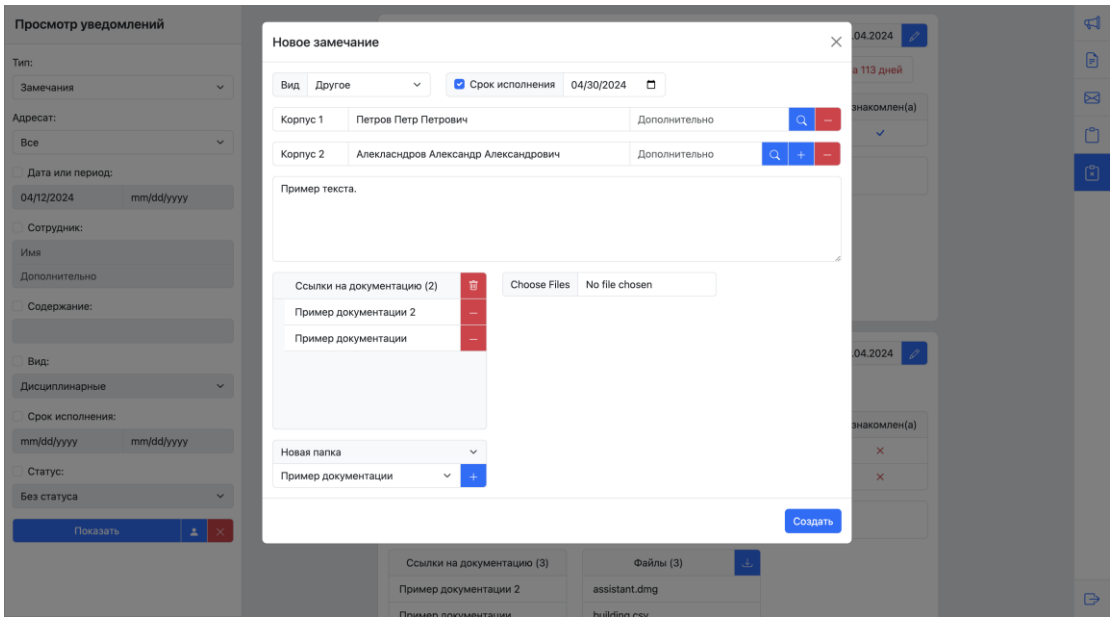


Рис. 4. Модальное окно создания уведомления типа «замечание»

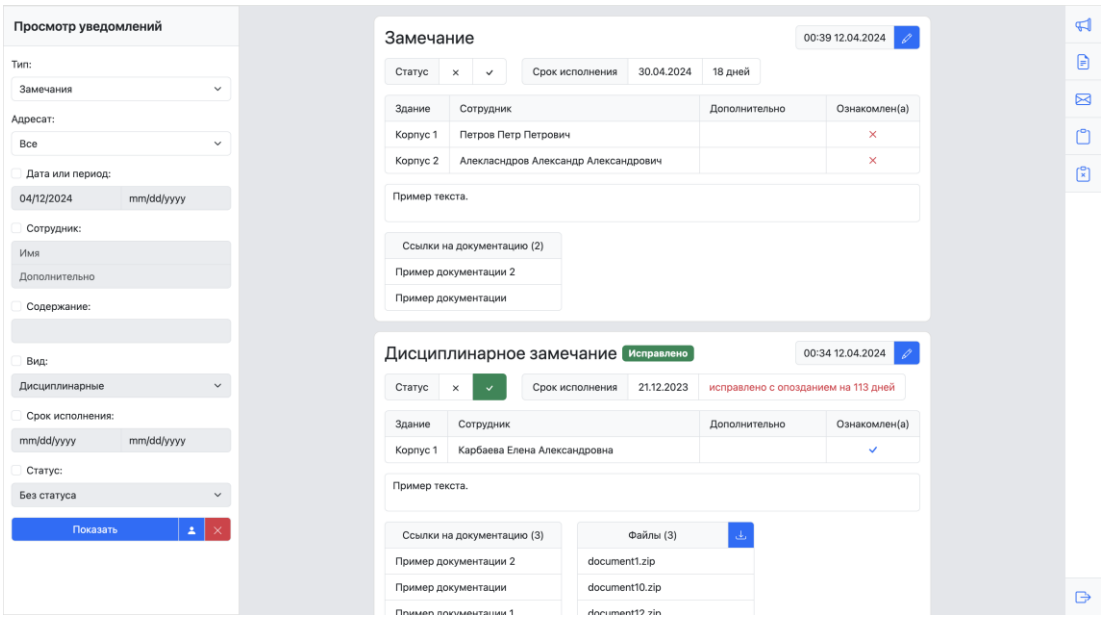


Рис. 5. Интерфейс пользователя типа «уведомитель»

На левой панели интерфейса «уведомителя» обозначены фильтры уведомлений, настраиваемые в соответствии с типами уведомлений. При входе в систему открывается раздел объявлений, что обусловлено его первоочередным положением в списке типов уведомлений, это является как удобным, так и логичным решением.

Лента уведомлений интерактивна, обеспечивает возможность изменения статусов, загрузки файлов и перехода по ссылкам на «документацию», а также перехода к редактированию уведомлений. На правой панели располагаются кнопки, предназначенные для создания уведомлений, а также кнопка выхода из системы.

Модальное окно [3] создания уведомления содержит форму уведомления и разнообразные виджеты. Например, для уведомления типа «замечание» доступна функция добавления и удаления дополнительных «адресатов» и «документации» из соответствующего раздела, в который она сортирована.

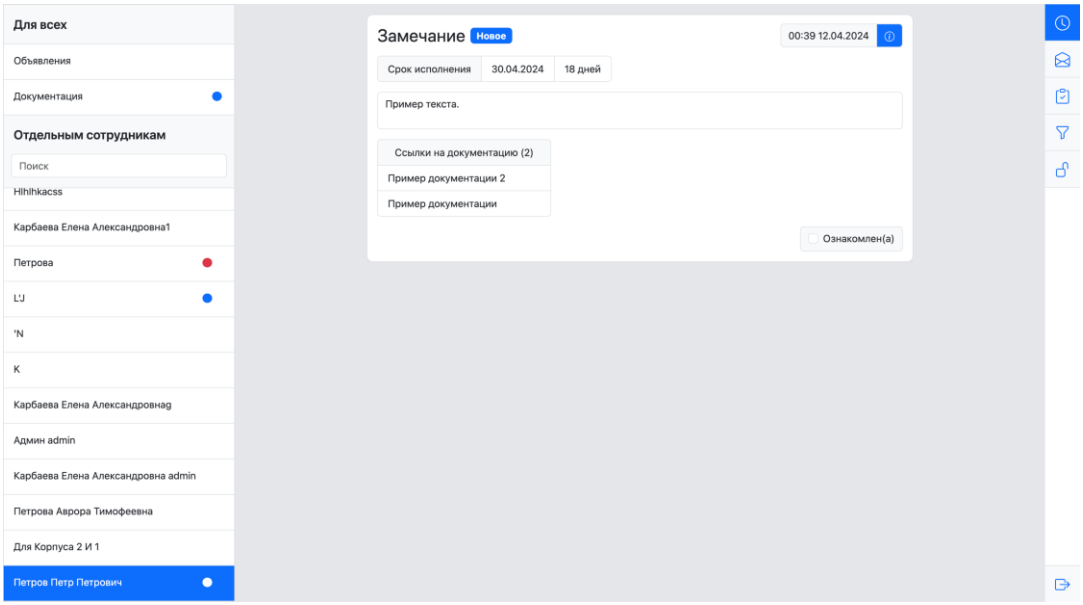


Рис. 6. Интерфейс пользователя типа «адресат»

Для «адресата» интерфейс имеет схожий облик, однако в ленте уведомлений отсутствует возможность перехода к редактированию уведомлений или изменения их статуса. Левая панель включает ссылки на общие разделы и на отдельных сотрудников, а также имеет поле для поиска сотрудников. Справа, первые три кнопки представляют собой разделы сверху вниз: новые уведомления, прочитанные сообщения, выполненные задания и исправленные замечания. Затем следует виджет фильтрации уведомлений в разделе, а также виджет ввода пароля для доступа к защищенным уведомлениям и снизу кнопка выхода из системы.

Заключение

В результате проделанной работы было создано веб-приложение, которое полностью соответствует всем поставленным требованиям. Оно представляет собой инструмент для корпоративного уведомления, управления задачами и оценки эффективности работы сотрудников.

Это приложение разработано с учетом потребностей различных организаций, предоставляет масштабируемую структуру данных, простой и удобный интерфейс, способствует повышению эффективности внутренней коммуникации и координации задач, что делает его ценным инструментом для современных организаций.

Литература

1. Документация Flask. – Режим доступа: https://flask.palletsprojects.com/en/3.0.x (Дата обращения: 12.04.2024).

2. Документация jQuery – Режим доступа: https://api.jquery.com (Дата обращения: 12.04.2024).

3. Документация Bootstrap. – Режим доступа: https://getbootstrap.com/docs/5.3 (Дата обращения: 12.04.2024).

4. Г. Домбровская Оптимизация запросов PostgreSQL / Г. Домбровская, Б. Новиков, А. Бейликова – Москва : ДМК Пресс, 2022. – 278 с.

УДК 004.832.32

АЛГОРИТМ ПРЯМОГО ВЫВОДА НА ОСНОВЕ ПРЕДСТАВЛЕНИЯ БАЗЫ ЗНАНИЙ В ВИДЕ ЕДИНОГО ГРАФА

В.В. Черкасов, О. В. Авсеева

Воронежский государственный университет

Введение

Задача прямого продукционного вывода [1] формулируется следующим образом: пусть есть база знаний B о некоторой системе, а также некоторое множество C заранее известных фактов о состоянии системы, которые представлены в виде A или $\overline{A}$ и не являются составными выражениями алгебры логики. Необходимо, используя продукции из базы знаний B, вывести новые факты о состоянии системы. Задачи прямого продукционного вывода решаются во многих сферах: экспертные системы, постановка диагноза в медицине, программное обеспечение для принятия решений, обработка естественного языка и другие [2].

В данной работе проводится анализ существующих алгоритмов решения задачи прямого продукционного вывода, а также представлены разработанные алгоритмы и структуры данных для решения данной задачи. Алгоритм, предложенный в данной работе, не ограничивается применимостью только к продукциям, но и к любым множествам выражений математической логики.

Более сложные задачи часто можно свести к такой дискретной задаче, используя предикаты, и устанавливая связи между этими предикатами для уменьшения количества оценок. Добавляя связи результата некой продукции с выполнением или невыполнением определенного условия другой продукции, например, если результатом продукции является операция - для некой характеристики системы выполняется оценка - факт F, то условие C истина, может быть представлено в виде дополнительной продукции в системе $F \to C$. Это позволяет отразить характеристику в виде набора важных для модели фактов и их взаимоотношений.

1. Существующие алгоритмы решения задачи

Алгоритмы решения задачи прямого продукционного вывода будем называть решателями.

Одним из самых первых использований решателей с прямым выводом, которое не ограничивается только целями изучения, является система OPS3, в которой используется алгоритм на основе RETE [3]. Значительное большинство решателей основаны именно на RETE и являются его модификациями. Также среди довольно известных применений решателей можно назвать язык программирования prolog, в котором используется алгоритм решения задач прямого и обратного вывода.

Одним из подходов к решению рассматриваемой задачи также является использование формальной логики на строках, содержащих высказывания. В то время, как это самый очевидный подход, он является очень медленным и занимает много памяти. Его необходимо упомянуть, так как это "идеальная" реализация алгоритма, которая обладает высокой объяснительной ценностью для такого рода задачи. Алгоритм этой научной имплементации представлен в работе [4]. Также пример использования логики первого порядка для решения

задачи человеком можно найти в статье [5], которая дает альтернативное определение задачи прямого продукционного вывода.

Так как наивный подход к реализации имеет крайне плохую эффективность, было создано множество других алгоритмов для решения задачи прямого продукционного вывода. Их представителями являются RETE, RuleSolver, TREAT, Leaps, RETE-II и многие другие.

Алгоритм RETE [6] рассматривается как значительный прорыв в сфере, а потому можно найти огромное количество его модификаций: RETE-II, RETE-III, RETE-NT и другие [7]. Также существует огромное количество алгоритмов, вдохновленных им. Он продемонстрировал высокую производительность и скорость работы в неупорядоченном наборе правил.

Из минусов алгоритма можно выделить так называемый "комбинаторный взрыв", который возникает, когда среди правил слишком много составных условий. В результате уровень вершин-комбинаций (бета-память) очень быстро разрастается, что негативно сказывается на производительности алгоритма.

Алгоритм RETE все еще является крайне популярным выбором. Он используется в таких проектах, как SPARKLING LOGIC SMARTS, которая использует модификацию RETE-NT; представляется в виде отдельного фреймворка для визуального программирования и других применений. Невозможно переоценить влияние этого алгоритма.

RuleSolver [8] интуитивен в использовании, но имеет ряд ограничений, самым сильным из них является необходимость порядка в оценке правил. Среди всех возможных задач прямого вывода можно выделить задачи, которые хорошо представляются в этом алгоритме и эффективно им решаются. Такие задачи характеризуются множеством групп правил, в каждой группе которых решается задача выбора альтернативы, и всегда выполняется не более одного правила. В таком случае эти правила можно разделить на разные группы, а каждое решение (группу правил) можно оценивать по-отдельности. Еще одним серьезным ограничением в использовании этого алгоритма можно назвать неоптимальность самого алгоритма в оценке составных условий, что сказывается на производительности.

Алгоритм RuleSolver является специализированным алгоритмом прямого продукционного вывода, который отлично справляется с определенным подмножеством задач и удобен для их программирования. Однако он не нацелен на решение общего круга задач, а потому уступает другим алгоритмам в общем случае, поэтому перед его использованием стоит оценить конкретно стоящую задачу и решить, является ли RuleSolver оптимальным решателем или стоит выбрать другой алгоритм.

Алгоритм RuleSolver лицензирован и доступен для скачивания с сайта владельца прав в виде библиотеки для Java.

Одной из модификаций алгоритма RETE является алгоритм TREAT, который ставит своей целью оптимизировать скорость оценки на основе сортировки правил по их условиям и векторам, что позволяет сократить количество ненужных проверок условий. Также алгоритм не использует вершины для комбинации, что уменьшает количество используемой памяти. Конкретный механизм топологической сортировки правил довольно интересен, но не является фокусом этой работы; он лучше представлен в работе [9].

TREAT является модификацией алгоритма RETE, а потому сохраняет все положительные и отрицательные черты алгоритма - "родителя", хотя утверждается, что он решает проблему полной неупорядоченности условий в RETE и благодаря этому работает быстрее. Однако согласно работе [10] было показано, что RETE работает быстрее, потому что сохраняет промежуточные результаты, а не заново их рассчитывает.

Leaps [11] представляется как альтернатива алгоритму RETE, однако необходимо заметить, что он основан на алгоритме TREAT, что означает, что он все еще является наследником RETE. Основным отличием от TREAT является добавление ленивой оценки для

разрешения конфликтов, что снижает сложность и объем памяти, и время работы алгоритма к линейной сложности [12].

Leaps, являясь модификацией RETE, значительно улучшает скорость работы и размер используемой памяти, поэтому в общем случае рекомендуется выбирать его перед RETE и TREAT.

Можно заметить, что алгоритмы, основанные на RETE, доминируют в этой сфере, поэтому целью данной работы было предложение алгоритма, который кардинально отличается от RETE. Этот алгоритм заключается в создании графа, который выполняет роль базы знаний и позволяет легко выбирать применимые правила, упрощая их до бинарных отношений.

2. Описание разработанного алгоритма

Алгоритм, предлагаемый в данной работе, основан на преобразовании базы знаний в вид ориентированного графа с несколькими видами узлов и даже наличием пары узлов, концы и начало которых отличаются только порядком(дизъюнкция). Для удобства они будут отображаться в виде ребер, так как различие в направлениях не содержит полезной информации и/или отличий в применении алгоритма. Этот алгоритм подразумевает два отдельных этапа работы с базой знаний: предварительный (построение) и вычислительный.

Алгоритм направлен на сосредоточение вычислительных затрат на предварительном этапе. Другими словами, основной задачей становится именно составление графа, особенности которого на этапе вычислений графа можно будет использовать для минимизации затрат на поиск применимых правил и других вычислительно затратных задач.

Введем некоторые определения и свойства единиц, которые будут использоваться в дальнейшем.

Узел графа — это некоторый факт, описываемый базой знаний системы, который может либо выполняться, либо не выполняться. Он будет описываться двумя переменными:

1. Истинность ξ — сам факт выполнения или не выполнения этого факта.
2. Применимость η — искусственная булева характеристика, которая добавляет информацию, необходимую для вычисления графа.

Пусть x — некоторая общая характеристика системы, а $x > 0$ это условие, которое представляется в виде узла. Тогда истинность узла полностью соответствует истинности высказывания в обычном смысле с добавлением 2х значений: не определено означает, что значение истинности высказывания не известно, а Противоречие означает, что выполнены высказывание и ему противоположное. Множество значений тогда $\{0, 1, Противоречие, Не\ определено\}$. Если не указано значение, то считается, что $\xi = Не\ определено$.

Для всех фактов системы, которые поступают в базу знаний до выполнения этапа построения графа, применимость узла графа считается истинной величиной, ложными в применимости будут считаться все созданные узлы на этапе построения графа. Также особым случаем является узел 0, который всегда присутствует в графе и характеризуется ($\xi = 0$, $\eta = 1$).

Дуги представлены в виде нескольких видов, определения которых даются в качестве таблиц истинности движения в разделе вычисление графа:

1. Импликация — дуга.
2. Дизъюнкция — ребро.
3. Импликативная активация — дуга.
4. Дизъюнктивная активация — дуга.

Замечание: Математическое описание алгоритма требует соблюдения непротиворечивости системы, однако различные подходы к решению этой задачи для противоречивых систем в реальной имплементации допускают обход этой проблемы.

2. 2.1. Подготовительный этап

Этап построения графа можно разделить на два отдельных подэтапа: подготовительный и создание графа.

Подготовительный этап. Перед началом построения графа необходимо привести базу знаний к особому виду. Все выражения должны состоять из дизъюнкций и импликаций, и не должны включать в себя конъюнкции, отрицания и скобки.

Теорема 4.1: Любое высказывание может быть представлено в необходимом для построения виде.

Доказательство: Пусть p — рассматриваемое высказывание. По теореме из математической логики любое выражение можно представить в конъюнктивной нормальной форме, пусть

$$p_c = (p_1) \cap (p_2) \cap \ldots \cap (p_n)$$

где p_c — конъюнктивная нормальная форма выражения p и p_i – выражения вида

$$p_i = p_{i_{1,0}} \cup p_{i_{2,0}} \cup \ldots \cup p_{i_{k,0}} \cup \overline{p}_{i_{1,1}} \cup \overline{p}_{i_{2,1}} \cup \ldots \cup \overline{p}_{i_{m,1}}$$

Преобразуем выражение так, чтобы в нем не было отрицаний, заменив дизъюнкции на импликации. Если в выражении находятся только отрицания литералов, то одно из отрицаний литералов можно заменить на $p_{i_{j,1}} \to 0$. Теперь выражение имеет вид

$$p_i = p_{i_{1,1}} \to p_{i_{2,1}} \to \ldots \to p_{i_{m,1}} \to p_{i_{1,0}} \cup p_{i_{2,0}} \cup \ldots \cup p_{i_{k,0}}$$

Определение: Для удобства правилом графа в данной работе будем называть выражение приведенного выше вида.

Повторить такой алгоритм для всех дизъюнкций литералов и рассматривать получившиеся выражения p_i, как отдельные правила, которые можно использовать для построения графа.

Используя алгоритм и доказательство теоремы, необходимо привести все выражения к нужному виду. Теперь можно перейти к построению самого графа.

3. 2.2. Построение графа

Граф выступает как база знаний для алгоритма, и поэтому следующие алгоритмы не создают отдельные графы каждый раз, когда используются, а дополняют один большой граф. Тогда алгоритм построения графа базы знаний следующий:

1. Пусть G — пустой граф, а R - множество правил графа, полученных на подготовительном этапе.
2. Добавить узел 0 с параметрами истинности $\xi = 0$ и применимости $\eta = 1$.
3. Если $R = \varnothing$, то останов.
4. Добавить правило $r \in R$ в граф G, $R = R/\{r\}$ и шаг 3.

Добавление правила в граф зависит от его вида. Для удобства алгоритмизации добавления правил будет введено следующее определение.

Определение: Импликация и дизъюнкция будут обозначаться @, когда такое различие не важно. Если указан индекс, то он используется только для различия положения операции в выражении.

Если правило графа имеет вид A, то алгоритм имеет вид:

1. Найти в графе G узел с названием A, если такого нет, то добавить его в граф G, указать применимость $\eta = 1$ для него.

2. Изменить истинность узла A, $\xi = 1$.

Если правило графа имеет вид $A@B$, то следует применить следующий алгоритм:

1. Найти в графе G узлы с названием A и B, если какого-то из них нет, то добавить такие узлы в граф G, указать применимость $\eta = 1$ для узлов A и B.

2. Если правило графа имеет вид $A \to B$, то добавить дугу Импликация в граф с началом в A и концом в B.

3. Если правило графа имеет вид $A \cup B$, то добавить ребро Дизъюнкция в граф с концами A и B.

Если правило графа имеет вид $A@_1B@_2...@_{n-3}W@_{n-2}X@_{n-1}Y@_nZ$, то применяется следующий рекурсивный алгоритм:

1. Найти в графе G узел с названием $[Y@_nZ]$, если такого нет, то добавить его в граф G. $[Y@_nZ]$ в дальнейшем означает именно узел с таким названием.

2. Добавить правило $[Y@_nZ]@_nZ$ в граф G.

3. Если $@_n$ это импликация, то добавить дугу импликативной активации с началом в Y и концом в $[Y@_nZ]$ в граф G.

4. Если $@_n$ это дизъюнкция, то добавить дугу дизъюнктивной активации с началом в Y и концом в $[Y@_nZ]$ в граф G.

5. Изменить применимость узла $[Y@_nZ]$ на $\eta = 0$.

6. Добавить правило $A@_1B@_2...@_{n-3}W@_{n-2}X@_{n-1}[Y@_nZ]$ в граф G.

Таким образом, по завершении алгоритма все правила будут добавлены в граф, и построение графа завершено, можно приступить к его вычислению.

4. 2.3. Вычисление графа

После создания графа G, который включает в себя все правила, можно указывать, какие факты верны или не верны для определенной ситуации. Пусть наша ситуация характеризуется A, тогда мы находим в графе G узел с названием A и изменяем его истинность $\xi = 1$, если же A не выполняется, то аналогично находим узел с названием A и изменяем его истинность $\xi = 0$.

Повторяем это действие для всех характеристик ситуации, которые отображены в графе G. Теперь мы хотим узнать все следствия — вычислить граф.

Алгоритм вычисления графа:

1. Положим $i = 0$. Из множества всех узлов графа G составим подмножество узлов, для которых выполняется $\xi \neq Не\,определено$. Назовем его G_a.

2. Произвольным образом возьмем узел $g \in G_a$ и положим $G_a = G_a / \{g\}$, $i = i + 1$.

3. Для всех узлов p_j, смежных g, обновим их значения согласно виду их связи. Если для p_j ξ или η изменили свое значение, то $G_a = G_a \cup \{p_j\}$.

4. Если G_a — пустое множество, то останов, иначе шаг 2.

Ниже будут представлены объяснения движения по дугам и ребрам в зависимости от вида связи.

Дуга Импликация из a в b:

1. Если a имеет истинность $\xi_a = 1$ и применимость $\eta_a = 1$, а истинность $\xi_b = Не\ определено$, то обновить истинность $\xi_b = 1$.

2. Если a имеет истинность $\xi_a = 1$ и применимость $\eta_a = 1$, а истинность $\xi_b = 0$, то обновить истинность $\xi_b = Противоречие$.

3. Если b имеет истинность $\xi_b = 0$ и применимость $\eta_b = 1$, а истинность $\xi_a = Не\ определено$, то обновить истинность $\xi_a = 0$.

4. Если b имеет истинность $\xi_b = 0$ и применимость $\eta_b = 1$, а истинность $\xi_a = 1$, то обновить истинность на $\xi_a = Противоречие$.

5. Иначе ничего не менять.

Ребро Дизъюнкция с концами a, b:

1. Если a имеет истинность $\xi_a = 0$ и применимость $\eta_a = 1$, а истинность $\xi_b = Не\ определено$, то обновить истинность $\xi_b = 1$.

2. Если a имеет истинность $\xi_a = 0$ и применимость $\eta_a = 1$, а истинность $\xi_b = 0$, то обновить истинность $\xi_b = Противоречие$.

3. Если b имеет истинность $\xi_b = 0$ и применимость $\eta_b = 1$, а истинность $\xi_a = Не\ определено$, то обновить истинность $\xi_a = 1$.

4. Если b имеет истинность $\xi_b = 0$ и применимость $\eta_b = 1$, а истинность $\xi_a = Не\ определено$, то обновить истинность $\xi_a = Противоречие$.

5. Иначе ничего не менять.

Дуга Импликативная активация из a в b:

1. Если a имеет истинность $\xi_a = 1$ и применимость $\eta_a = 1$, то обновить применимость $\eta_b = 1$.

2. Если b имеет истинность $\xi_b = Противоречие$, а истинность $\xi_a = Не\ определено$, то обновить истинность a на $\xi_a = 0$.

3. Если b имеет истинность $\xi_b = Противоречие$, а истинность $\xi_a = 1$, то обновить истинность a на $\xi_a = Противоречие$.

4. Иначе ничего не менять.

Дуга Дизъюнктивная активация из a в b:

1. Если a имеет истинность $\xi_a = 0$ и применимость $\eta_a = 1$, то обновить применимость $\eta_b = 1$.

2. Если b имеет истинность $\xi_b = Противоречие$, а истинность $\xi_a = Не\ определено$, то обновить истинность $\xi_a = 1$.

3. Если b имеет истинность $\xi_b = Противоречие$, а истинность $\xi_a = 0$, то обновить истинность $\xi_a = Противоречие$.

4. Иначе ничего не менять.

Как только применение вышеуказанного алгоритма завершится, остается только достать рассчитанную информацию, рассматривая значения истинности и применимости среди узлов графа.

Если для узла графа, который назван литералом A, значение истинности $\xi = x$, то и соответствующий факт A имеет истинность x. Если для узла графа, который назван составным именем в этой работе вида [...], значение применимости $\eta = 1$, то это выражение было выведено в результате работы алгоритма.

3. Пример работы алгоритма

Пример работы алгоритма для произвольной системы математической логики, взятой из логической задачи:

1. $AB \to C$
2. $A \to B$
3. $D \to A$
4. $D \vee \overline{C}$

Первое и последнее высказывания находятся в форме, которую необходимо преобразовать. Для приведения можно воспользоваться двумя способами: составить КНФ с помощью таблицы истинности или использовать эквивалентные преобразования. Здесь будет использован второй способ.

Первое выражение:

$$AB \to C = \neg(AB) \vee C = \overline{A} \vee \overline{B} \vee C = A \to B \to C$$

Также преобразуется последнее выражение, здесь представляется лучше $C \to D$, но для демонстрации работы с узлом 0 и работы с дизъюнкцией, будет использовано другое представление $D \vee (C \to 0)$, здесь скобки используются только для простоты восприятия.

Теперь система имеет вид:

1. $A \to B \to C$
2. $A \to B$
3. $D \to A$
4. $D \vee (C \to 0)$

Теперь можно построить граф следующего вида (рис. 1):

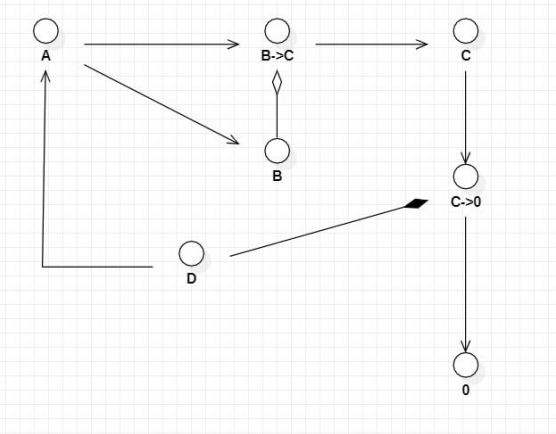


Рис. 1. Граф системы:

Здесь:

1. Дуга со стрелкой — Импликация.
2. Дуга с не закрашенным ромбом — Импликативная активация.
3. Дуга с закрашенным ромбом — Дизъюнктивная активация.

предложенный алгоритм был реализован в виде Python-скрипта, который вычисляет получившийся граф и выводит в консоль истинность простых фактов A, B, C, D на основе предоставленной информации.

Пусть известно, что $\overline{\mathrm{A}}$. Тогда результатом вычисления является (рис. 2).

```
A = FALSE
C = FALSE
B = UNDEFINED
D = FALSE
```

Рис. 2. Результат вычисления

Рассуждения, которые произвел скрипт:

1. Обратное движение по дуге импликации от D к A, $\overline{D}$.
2. Обратное движение по дуге импликации от 0 к $C \to 0$, Узел не активен.
3. Прямое движение по дуге дизъюнктивной активации от D к $C \to 0$.
4. Обратное движение по дуге импликации от $C \to 0$ к C, $\overline{C}$.
5. Обратное движение по дуге импликации от C к $A \to C$, Узел не активен.
6. Больше доступных движений нет, остается.

Для проверки полученного результата проведем рассуждения над системой вручную:

1. $A \to B \to C$
2. $A \to B$
3. $D \to A$
4. $D \vee (C \to 0)$
5. $\overline{\mathrm{A}}$
6. (3, 5) modus tollens, $\overline{D}$
7. (4, 6), $C \to 0$
8. (7) modus tollens, $\overline{C}$
9. Останов

Как можно видеть из рассуждений, система выдала значения истинности для всех возможных факторов верно.

Заключение

В ходе работы был проведен анализ существующих алгоритмов для решения задачи прямого продукционного вывода, представлены алгоритмы по созданию и вычислению графа, обоснован механизм работы.

Список использованных источников

1. Прямой логический вывод в продукционных эс. – Режим доступа: https://studfile.net/preview/2582507/page:11/. – (Дата обращения: 20.01.2024).
2. Forward Chaining. – Режим доступа: https://www.doubtly.in/blog/forward-chaining. – (Дата обращения: 20.01.2024).
3. Efficient Matching Algorithms for the SOAR/OPS5 Production System. – Режим доступа: https://apps.dtic.mil/sti/tr/pdf/ADA174277.pdf. – (Дата обращения: 20.01.2024).
4. Artificial Intelligence. – Режим доступа: http://www-personal.umd.umich.edu/~leortiz/teaching/6.034f/Fall05/rules/fwd_bck.pdf. – (Дата обращения: 20.01.2024).
5. Forward Chaining and backward chaining in AI. – Режим доступа: https://www.javatpoint.com/forward-chaining-and-backward-chaining-in-ai. – (Дата обращения: 20.01.2024).
6. How the Rete Algorithm Works. – Режим доступа: https://www.sparklinglogic.com/rete-algorithm-demystified-part-2/. – (Дата обращения: 20.01.2024).
7. Variants. – Режим доступа: https://en.wikipedia.org/wiki/Rete_algorithm\#Variants. – (Дата обращения: 20.01.2024).
8. Rule Solver. – Режим доступа: http://openrules.com/pdf/RulesSolver.UserManual.pdf. – (Дата обращения: 20.01.2024).
9. A Better Match algorithm for AI Production Systems. – Режим доступа: https://cdn.aaai.org/AAAI/1987/AAAI87-008.pdf. – (Дата обращения: 20.01.2024).
10. Comparison of the Rete and Treat Production Matchers for Soar (A Summary). – Режим доступа: https://cdn.aaai.org/AAAI/1988/AAAI88-123.pdf. – (Дата обращения: 20.01.2024).
11. Procedia Computer Scienc. – Режим доступа: https://www.sciencedirect.com/science/article/pii/S1877050914010576?ref=pdf_download\\ \&fr=\RR-2\&rr=83aae4778f91163d. – (Дата обращения: 20.01.2024).
12. TREAT or RETE. – Режим доступа: https://www.cs.utexas.edu/~miranker. – (Дата обращения: 20.01.2024).

УДК 517.927.2

ПОСТРОЕНИЕ ФУНКЦИЙ ГРИНА КРАЕВЫХ ЗАДАЧ С МАТРИЧНЫМ КОЭФФИЦИЕНТОМ

В. Ю. Чурсин
Воронежский государственный университет

Введение

Целью настоящей работы является построение явного представления функции Грина для решения двух краевых задач. Первая представляет собой краевую задачу на отрезке

$$\begin{aligned} x'(t) - Ax(t) &= f(t), \\ \alpha x(a) + \beta x(b) &= 0. \end{aligned}$$

с матричным коэффициентом A, обладающим свойством: $\alpha e^{\lambda a} + \beta e^{\lambda b} \neq 0$ при всех $\lambda \in \sigma(A)$. Второй краевой задачей является задача об ограниченных решениях

$$x''(t) - Ax(t) = f(t), \qquad t \in R,$$

с матричным коэффициентом A, чей спектр не пересекает отрицательную действительную полуось.

Идея решения обеих задач заключается в том, чтобы сначала угадать ответ, а затем уже стандартными методами его обосновать. Для угадывания формулы для решения сначала решается [1, 5, 7] скалярная краевая задача и в ответ вместо скалярного коэффициента λ подставляется матрица A. Затем выполняется проверка.

Краевая задача на отрезке

Обозначим через $C = C([a,b],C^n)$ линейное пространство всех непрерывных векторных функций $f\colon [a,b] \to C^n$, а через $C^1 = C^1([a,b],C^n)$ — линейное пространство непрерывно дифференцируемых функций x, для которых $x, x' \in C$.

Теорема 1. *Пусть* $A \in C^{n\times n}$ *— матрица, обладающая свойством:* $\alpha e^{\lambda a} + \beta e^{\lambda b} \neq 0$ *при всех* $\lambda \in \sigma(A)$. *Тогда краевая задача*

$$\begin{aligned} x'(t) - Ax(t) &= f(t), \\ \alpha x(a) + \beta x(b) &= 0. \end{aligned}$$

при любой $f \in C([a,b],C^n)$ *имеет единственное решение* $x \in C^1([a,b],C^n)$, *и это решение может быть представлено в виде*

$$x(t) = \int_a^b G(t,s) f(s)\, ds,$$

где

$$G(t,s) = \begin{cases} \alpha e^{Aa} e^{A(t-s)} (\alpha e^{Aa} + \beta e^{Ab})^{-1}, & a \le s < t, \\ -\beta e^{Ab} e^{A(t-s)} (\alpha e^{Aa} + \beta e^{Ab})^{-1}, & t < s \le b. \end{cases}$$

По поводу аналитических функций от матриц см. [2, 3, 4, 6].

Для вывода предыдущей формулы сначала рассматривается случай краевой задачи

$$\begin{aligned} x'(t)-\lambda x(t) &= f(t),\\ \alpha x(a)+\beta x(b) &= 0. \end{aligned}$$

с произвольным параметром $\lambda \in C$ и затем доказывается, что при условии $\alpha e^{\lambda a}+\beta e^{\lambda b}\neq 0$ решение существует, единственно и представимо в виде

$$x(t)=\int_a^b G(t,s)f(s)\,ds,$$

где

$$G(t,s)=\begin{cases} \dfrac{\alpha e^{\lambda a}}{\alpha e^{\lambda a}+\beta e^{\lambda b}}e^{\lambda(t-s)}, & a\le s<t,\\ -\dfrac{\beta e^{\lambda b}}{\alpha e^{\lambda a}+\beta e^{\lambda b}}e^{\lambda(t-s)}, & t<s\le b. \end{cases}$$

По аналогии с этой формулой определяется матричный вариант

$$G(t,s)=\begin{cases} \alpha e^{Aa}e^{A(t-s)}(\alpha e^{Aa}+\beta e^{Ab})^{-1}, & a\le s<t,\\ -\beta e^{Ab}e^{A(t-s)}(\alpha e^{Aa}+\beta e^{Ab})^{-1}, & t<s\le b. \end{cases}$$

и доказывается, что решение x, построенное с использованием такой функции Грина, является единственным решением краевой задачи.

Задача об ограниченных решениях

Далее рассматривается дифференциальное уравнение

$$x''(t)-Ax(t)=f(t),\qquad t\in R,$$

с матричным коэффициентом A на всей оси. *Задачей об ограниченных решениях* для такого уравнения называют задачу о нахождении ограниченного решения x, заданного на R, в предположении, что функция f ограничена.

Введем обозначение для отрицательной действительной полуоси:

$$R_- = \{\, z\in R: z\le 0\,\}.$$

Под $\sqrt{(\cdot)}$ будем понимать аналитическую ветвь корня, определенную в $C\backslash R_-$ и принимающую значения в открытой правой комплексной полуплоскости.

Обозначим через $C=C(R,C^n)$ линейное пространство всех непрерывных ограниченных функций $f: R\to C^n$, а через $C^2=C^2(R,C^n)$ — линейное пространство дважды непрерывно дифференцируемых функций x, для которых $x,x',x''\in C$.

Построение решения ведется по той же схеме, что и в случае краевой задачи на отрезке. Сначала рассматривается скалярный аналог задачи. Пусть $\lambda\in C\backslash R_-$ — число. Рассмотрим скалярное дифференциальное уравнение

$$x''(t)-\lambda x(t) = f(t), \qquad t \in R.$$

Нахождение решения этого уравнения стандартными методами из курса дифференциальных уравнений приводит к следующему результату.

Теорема 2. *Пусть* $\lambda \in C\backslash R_-$. *Тогда для любой непрерывной ограниченной функции* $f: R\to C$ *уравнение*

$$x''(t)-\lambda x(t) = f(t), \qquad t \in R.$$

имеет единственное ограниченное решение x *и это решение имеет вид*

$$x(t) = \int_{-\infty}^{+\infty} G(t-s)f(s)\,ds,$$

где

$$G(t) = -\frac{1}{2\sqrt{\lambda}}e^{-\sqrt{\lambda}|t|}, \qquad t \in R.$$

И обратно, если это уравнение при любой ограниченной f *имеет единственное ограниченное на* R *решение* x, *то* $\lambda \in C\backslash R_-$.

Перейдем теперь к построению решения в случае матричного коэффициента. Это делается путем последовательного доказательства приводимых ниже утверждений.

Рассмотрим функции

$$G_+(t) = -\frac{1}{2}e^{+\sqrt{A}t}(\sqrt{A})^{-1}, \qquad t \le 0,$$
$$G_-(t) = -\frac{1}{2}e^{-\sqrt{A}t}(\sqrt{A})^{-1}, \qquad t \ge 0.$$

Поскольку спектр матрицы A не пересекается с множеством R_-, матрица $\sqrt{A}$ определена и обратима, а значит, функции G_+ и G_- определены корректно. Определим функцию

$$G(t) = \begin{cases} G_-(t) & \text{при } t \ge 0, \\ G_+(t) & \text{при } t \le 0. \end{cases}$$

Функцию G назовем *функцией Грина* задачи об ограниченных решениях.

Лемма 3. *Пусть* $A \in C^{n\times n}$ — *матрица, спектр которой не пересекает множество* R_-. *Тогда для любой* $f \in C(R,C^n)$ *функция*

$$x(t) = \int_{-\infty}^{+\infty} G(t-s)f(s)\,ds,$$

дважды непрерывно дифференцируема. При этом

$$x''(t) = f(t) + A\int_{-\infty}^{+\infty} G(t-s)f(s)\,ds.$$

Лемма 4. *Пусть* $A \in C^{n\times n}$ — *матрица, спектр которой не пересекает множество* R_-. *Тогда для любой* $f \in C(R,C^n)$ *функция*

$$x(t) = \int_{-\infty}^{+\infty} G(t-s)f(s)\,ds$$

ограничена при $t\to-\infty$ *и* $t\to+\infty$.

Лемма 5. *Пусть* $A \in C^{n\times n}$ *— матрица, спектр которой не пересекает множество* R_-. *Тогда общее решение однородного уравнения*

$$x''(t) = Ax(t)$$

представимо в виде

$$\psi(t) = e^{\sqrt{A}t}C_1 + e^{-\sqrt{A}t}C_2,$$

где $C_1, C_2 \in C^n$ *— произвольные векторы.*

Лемма 6. *Пусть* $A \in C^{n\times n}$ *— матрица, спектр которой не пересекает множество* R_-. *Тогда справедливы следующие утверждения:*

$$\lim_{t\to+\infty} \| e^{\sqrt{A}t}C_1 \| = +\infty,$$
$$\lim_{t\to-\infty} \| e^{-\sqrt{A}t}C_2 \| = +\infty.$$

где $C_1, C_2 \in C^n$ *— произвольные ненулевые векторы.*

Теорема 7. *Пусть* $A \in C^{n\times n}$ *— матрица, спектр которой не пересекает множество* R_-. *Тогда для любой ограниченной* $f \in C(R, C^n)$ *уравнение*

$$x''(t) - Ax(t) = f(t), \qquad t \in R,$$

имеет единственное ограниченное решение $x \in C^2(R, C^n)$, *и это решение может быть представлено в виде*

$$x(t) = \int_{-\infty}^{+\infty} G(t-s)f(s)\,ds.$$

И обратно, если это уравнение при любой ограниченной f *имеет единственное ограниченное на* R *решение* x, *то спектр* A *не пересекает множество* R_-.

Литература

1. Боровских, А. В. Лекции по обыкновенным дифференциальным уравнениям / А. В. Боровских, А. И. Перов. – М.–Ижевск : Регулярная и хаотическая динамика, 2004. – 540 с. – ISBN: 5-93972-327-6.

2. Далецкий, Ю. Л. Устойчивость решений дифференциальных уравнений в банаховом пространстве / Ю. Л. Далецкий, М. Г. Крейн. Нелинейный анализ и его приложения. – М. : Наука, 1970. – 536 с.

3. Курбатов, В. Г. Основы спектральной теории / В. Г. Курбатов, И. В. Курбатова. – Воронеж : Научная книга, 2015. – 122 с. – ISBN: 978-5-4446-0611-7.

4. Курбатов, В. Г. Вычислительные методы спектральной теории : учебное пособие / В. Г. Курбатов, И. В. Курбатова. – Воронеж : Издательский дом ВГУ, 2022. – 326 с. –ISBN: 978-5-9273-3384-4.

5. Наймарк, М. А. Линейные дифференциальные операторы / М. А. Наймарк. – Второе изд. – М. : Наука, 1969. – 528 с.

6. Рудин, У. Функциональный анализ / У. Рудин. – М. : Мир, 1975. – 443 с.

7. Хартман, Ф. Обыкновенные дифференциальные уравнения / Ф. Хартман. – М. : Мир, 1970. – 720 с.

УДК 004.42

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ КОММУНИКАЦИИ УЧАСТНИКОВ ОБРАЗОВАТЕЛЬНОЙ СИСТЕМЫ

К. А. Шанина

Воронежский государственный университет

Введение

В современное время особое внимание уделяется образованию. Учебная система, которая основывается на прогрессивных технологиях, нуждается в инструментах, позволяющих оперативно поддерживать связь и учебный процесс вне зависимости от формы обучения. Наличие сегодня практически у каждого человека мобильного телефона позволяет ему находиться на связи в любое время, а также иметь доступ ко всем материалам, находящимся в памяти или на интернет-платформах.

Многие технологии помогают реализовывать мобильное обучение, а разнообразные мобильные приложения позволяют решать широкий спектр задач в обучении, в том числе задачи коммуникации и самообразования. Создание обучающих платформ помогает повысить качество и доступность образования. Жители отдаленных регионов, которые не имеют возможности получать хорошее образование из-за недостатка кадров, могут получать как классическое, так и дополнительное образование с помощью средств для коммуникации. Помимо этого, качественное образование смогут получать люди с ограниченными возможностями.

В данной работе представлено мобильное приложение для коммуникации между членами образовательного процесса и организации обучения школьников, студентов, а также участников дополнительного образования.

1. Анализ задачи

Мобильное приложение для коммуникации должно позволять комфортно общаться в приложении и иметь возможность качественно организовывать учебный процесс. Для этого необходимо провести анализ задачи и выявить все необходимые задачи и требования к приложению.

1.1. Функциональные требования

Перед началом разработки надо выполнить анализ необходимой функциональности приложения.

Задачами данного программного обеспечения являются:

- регистрация пользователя;
- выполнение входа в кабинет пользователя;
- создание курсов (для преподавателей и администраторов);
- удобный поиск курсов;
- возможность приглашать по ссылке и добавлять пользователей к курсу;
- возможность написания личных сообщений;
- возможность написания сообщений в чат курса;

- просмотр учебных материалов с имеющихся у пользователя курсов;
- удобный интерфейс;
- возможность расширения в дальнейшем;

Для достижения этих целей выделяются необходимые требования:

1. Простой и понятный интерфейс;
2. Возможность сдачи и проверки домашней работы;
3. Доступ к учебный материалам вне сети интернет;
4. Возможность объединения учеников в группы;
5. Разграничение доступа к учебным материалам;

Также определяются 3 необходимые роли пользователей: студент, преподаватель и администратор. У всех пользователей изначально есть возможность зарегистрироваться и указать свою роль. Основные действия пользователя при входе указаны на рис. 1. После входа в приложение у ролей появляются отличающиеся друг от друга действия. На рис.2-3 представлены диаграммы основных действий пользователей в зависимости от роли.

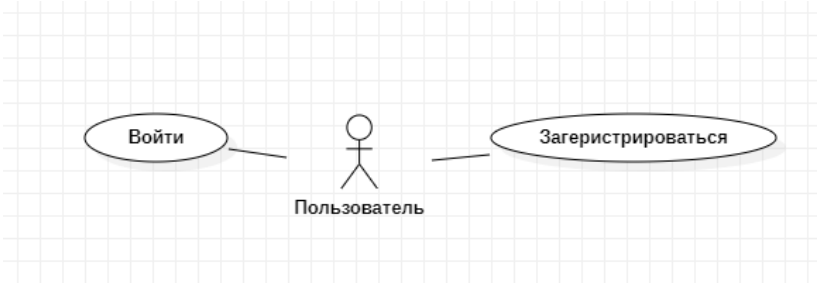


Рис. 1 Диаграмма основных действий пользователя

Рис. 2. Диаграмма основных действий студента

Рис. 3. Диаграмма основных действий преподавателя

Рис. 4. Диаграмма основных действий администратора

1.2. Выбор платформы и средств реализации

Для создания приложения выбрана платформа Android, потому что она является наиболее популярной среди пользователей мобильных телефонов, что позволит приложению быть универсальным и доступным.

Для разработки необходимо подобрать инструменты разработки:

1. Kotlin. Это статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine. Преимуществами данного языка являются: поддержка от Google, совместимость с Java и библиотеками языка, регулярные обновления и дополнения к языку;
2. Firebase. Платформа для разработки мобильных приложений, которая используется в качестве хранилища данных, позволяющая работать как с маленькими проектами, так и с крупными, что в дальнейшем позволит расширять приложение;
3. Picasso. Мощная библиотека Kotlin для работы с изображениями;
4. Локальный контроль версий с помощью Git и выгрузка версий на GitHub для удаленного доступа к проекту;

2. Реализация мобильного приложения

Перед реализацией мобильного приложения необходимо определить структуру базы данных, используемой в приложении, а также способ разделения приложения на логические части, для удобной разработки и в дальнейшем для удобной работы с кодом при расширении приложения.

2.1. База данных

Основные таблицы, которые необходимы для хранения данных и находятся в базе:

1. Profiles – таблица для хранения информации профилей.
2. Courses – таблица для хранения информации о курсах.
3. Chats – таблица для хранения чатов.

На рис. 5 представлена полная схема базы данных со всеми вспомогательными таблицами, которая используется в приложении.

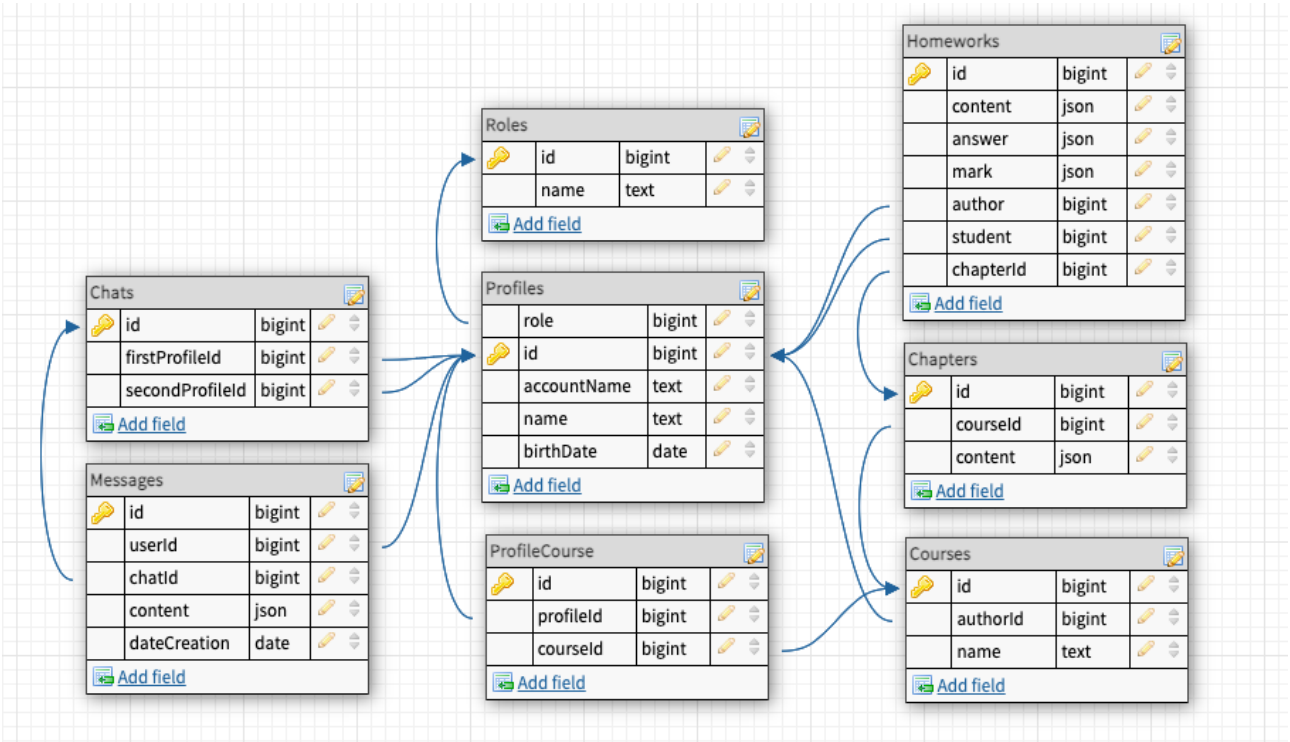


Рис. 5 Полная схема базы данных

2.2. Шаблон MVVM

При проектировании приложения используется шаблон MVVM, который позволяет разделить логические части проекта на различные объекты. Основной целью использования данного шаблона является отделение бизнес-логики от пользовательского интерфейса. Компоненты внутри каждого из блоков имеют связи с другими компонентами благодаря механизму связывания. Согласно данному шаблону приложение разделяется на три главных компонента: View, соответствующий интерфейсу пользователя, Model, содержащий основную бизнес-логику и данные, и ViewModel, обеспечивающий связь предыдущих двух компонентов. Компонент Model в данной задаче представлена базой данных, указанной в пункте 2.1.

3. Интерфейс пользователя

Приложение состоит из основных экранов:

1. Экран входа

   Разрешен вход пользователя по номеру телефона или по адресу электронной почты. Если пользователь был зарегистрирован ранее, он будет перенаправлен на домашнюю страничку. Иначе пользователь будет направлен на страницу создания профиля.

2. Страница создания и редактирования профиля

   Пользователю необходимо внести или изменить данные о себе. После этого он переносится на домашнюю страницу. При сохранении в первый раз роль пользователя должна быть подтверждена администратором.

3. Домашняя страница

   На этом экране пользователь может просматривать курсы, чаты и выполнять поиск. С данного экрана есть возможность перейти в редактирование своего профиля.

4. Страница просмотра чата или курса

   На данной странице пользователь может просматривать данные чата и отправлять сообщения, или просматривать данные курса. С данного экрана преподаватель и администратор могут перейти на страницу редактирования курса.

5. Страница создания и редактирования курса

   Экран доступный для администраторов и преподавателей. Тут можно вносить данные по курсу: название курса, список участников курса, разделы. В разделы можно добавить: описание, текстовые поля, учебные материалы в виде файлов, изображений, видеоматериалов, формы для сдачи домашних работ.

6. Страница домашней работы

   Представляет собой экран с полями: для прикрепления и просмотра файлов, для оценки. Оценку преподаватель может изменять, а студент просматривать.

Заключение

В данной работе был описан проект разработки мобильного приложения для коммуникации и организации обучения. Практическим результатом является рабочее мобильное приложение.

Литература

1. Android Developers // URL: https://developer.android.com

2. Использование шаблона MVVM // URL: https://habr.com/ru/companies/dataart/articles/272737/

3. Дейтл П., Дейтл Х., Уолд А., Android для разработчиков. //3-е издание – СПб.: Питер, 2016. – 512 с.

4. Мартин Р. «Чистый код. Создание, анализ и рефакторинг». 2019 — 464 с.

УДК 004.891

ПОСТРОЕНИЕ ДЕРЕВА РЕШЕНИЙ ДЛЯ ДИАГНОСТИКИ ТИПА ЛЕЙКОПЛАКИИ РОТОВОЙ ПОЛОСТИ

М.А. Шашкина

Воронежский государственный университет

Введение

Лейкоплакия полости рта — поражение слизистой, возникающее вследствие постоянного раздражения и сопровождающееся повышенным ороговением (гиперкератозом) [1]. К основным раздражителям относятся курение, злоупотребление алкоголем, пряностями, употребление слишком холодной или слишком горячей пищей, длительный прием некоторых лекарственных препаратов, стоматологические проблемы. Кроме того, основными факторами риска для данного заболевания являются наследственная предрасположенность, наличие вируса папилломы человека (ВПЧ), сахарный диабет, болезни органов желудочно-кишечного тракта. Опасность лейкоплакии заключается в том, что она может переродиться в рак, поэтому ее ранняя диагностика крайне важна. Под медицинской диагностикой будем понимается процесс установления диагноза — заключения о болезни и состоянии пациента, выраженное в принятой медицинской терминологии [2]. Задача медицинского диагноза заключается в том, чтобы на основе совокупности показателей, характеризующих состояние пациента, из некоторой совокупности заболеваний выделить то, которое в максимальной степени соответствует состоянию пациента. Сложность постановки диагноза заключается в том, что некоторые симптомы могут быть характерными сразу для нескольких заболеваний. Кроме того, даже при одном и том же заболевании у двух пациентов симптомы могут проявляться с разной интенсивностью. Опытный врач принимает во внимание весь комплекс проявляющихся симптомов, его опыт позволяет сократить множество возможных заболеваний до минимума, а затем, применяя дополнительные методы диагностики, установить заболевание и назначить пациенту лечение. Процесс рассуждений врача относительно тех показателей, которые выделены, можно формализовать в виде диагностического дерева, которое, по сути, относится к деревьям решений [3].

Статья посвящена описанию алгоритма построения диагностического дерева, который определяет набор показателей и последовательность их рассмотрения при постановке диагноза.

1. Алгоритм построения диагностического дерева

Предположим, что для группы заболеваний $Z_1,...,Z_N$ на основе консультаций с опытным врачом выделен некоторый набор существенных для учета показателей $P_1,...,P_n$, характеризующих состояние пациента, причем для каждого показателя $i=\overline{1,n}$ определена шкала в виде конечного множества возможных значений $ScaleP_i=\left\{zP_j^i\right\}\left(j=\overline{1,n_i}\right)$, где $|ScaleP_i|=n_i$. Например, для некоторой группы заболеваний имеет смысл выделить показатель «наличие головной боли» $\left(P_i\right)$, тогда его шкала будет содержать, например, следующие значения:

$$ScaleP_i=\{\textit{очень сильная};\textit{сильная};\textit{умеренная};\textit{слабая};\textit{отсутствует}\},$$

здесь $n_i=\left|ScaleP_i\right|=5$. В качестве возможных значений может выступать логическая $\{0,1\}$, количественная, лингвистическая, интервальная информация. Значения zP_j^i показателя P_i будем также называть специфичными.

Для заболеваний $Z_1,...,Z_N$ каждый из показателей проявляется в различной степени, а состояние пациента может быть описано вектором возможных значений показателей. Этот вектор будем называть векторной оценкой или экземпляром. Данная оценка позволяет предположить о наличие у пациента некоторого заболевания из группы заболеваний $Z_1,...,Z_N$.

Под *диагностическим деревом* подразумевается граф, который удовлетворяет следующим условиям:

1) вершины графа соответствуют специфичным значениям показателей;

2) граф является ориентированным деревом и разложен по уровням, причем каждый уровень соответствует показателю, и на данном уровне располагается столько вершин, сколько специфичных значений имеет данный показатель;

3) показатели-уровни дерева упорядочены по значимости для данной группы заболеваний, что позволяет оптимизировать количество шагов, необходимых для постановки диагноза;

4) в дереве имеется конечное множество висячих вершин, которые соответствуют конкретным заболеваниям и являются основой для постановки диагноза.

Диагностическое дерево определяет взаимосвязи между значениями показателей и позволяет определить тип заболевания. Однако зачастую сопутствующие заболевания создают «смазанную» картину, значения диагностических показателей в той или иной степени отклоняются от тех специфичных значений, на основании которых ставится диагноз, поэтому целесообразно использовать статистические оценки появления специфичных значений у конкретного пациента.

Диагностическое дерево относится к деревьям решений, которые используются для решения задач прогнозирования и классификации [4]. Алгоритм построения деревьев решений относится к жадным алгоритмам. Свойство «жадности» означает, что локально-оптимальные решения на каждом шаге (разбиения в узлах), приводят к искомому оптимальному решению. В случае деревьев решений это означает, что если один раз был выбран атрибут, и по нему было произведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другой атрибут, который дал бы лучшее итоговое разбиение. В каждом узле дерева находится предикат, истинность или ложность которого является основой для ветвления. Прогнозирование (предсказание) или классификация на основе дерева решений осуществляется как проход по дереву от корня к некоторой висячей вершине.

На рис. 1 представлено дерево решений для разбиения на 5 классов объектов, которые характеризуются двумя количественными признаками x и y.

Для построения диагностического дерева необходимо задать обучающее множество $LSet$, состоящее из экземпляров, каждый из которых относится к некоторому заболеванию (классу) из группы $Z_1,...,Z_N$. Важно, чтобы каждое заболевание было представлено необходимым и достаточным количеством экземпляров. Пусть обучающее множество $LSet$ включает экземпляры, относящиеся к заболеваниям $Z_1,...,Z_N$. Каждый экземпляр соответствует конкретному пациенту, обозначим его $\boldsymbol{x}^{\Pi_k} = \left(x_1^{\Pi_k},...,x_n^{\Pi_k}\right)$, где $x_i^{\Pi_k} \in ScaleP_i$ – значение показателя P_i для пациента Π_k. Обозначим I_{Z_k} – множество пациентов в обучающем множестве $LSet$, у которых диагностировано заболевание Z_k, тогда $LSet = I_{Z_1} \cup ... \cup I_{Z_N}$.

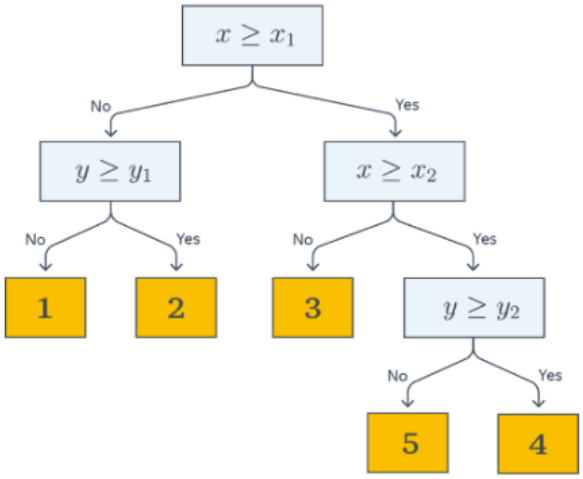


Рис. 1. Пример диагностического дерева
(https://education.yandex.ru/handbook/ml/article/reshayushchiye-derevya)

В алгоритме построения дерева лежит принцип жадной максимизации прироста информации – на каждом шаге выбирается тот показатель/признак, при разделении по которому прирост информации оказывается наибольшим.

Рассмотрим алгоритм построения диагностического дерева.

Шаг 1. Зафиксировать множество заболеваний Z. Для каждого заболевания $Z_k \in Z$ отобрать относящиеся к нему экземпляры $\left\{x^{\Pi_s}\right\}_{s \in I_{Z_k}}$ и определить относительную частоту p_{Z_k} появления пациентов с заболеванием Z_k в обучающем множестве. Найти оценку в виде энтропии неопределенности диагноза на основе данного обучающего множества $LSet$

$$H(Z) = -\sum_{k=1}^{N} p_{Z_k} \log_2 p_{Z_k}.$$

Шаг 2. Для каждого показателя P_i рассмотреть последовательно специфичные значения $zP_j^i \left(j = \overline{1, n_i}\right)$ из его шкалы значения $ScaleP_i$ и выполнить следующие действия:

a) Найти относительную частоту $p_j^i = P\left(zP_j^i\right)$ появления значения zP_j^i в экземплярах обучающего множества и относительную частоту $p\left(Z_k / zP_j^i\right)$ диагностики заболевания $Z_k \in Z$ при фиксированном значении zP_j^i.

b) Определить следующие условные энтропии:

- неопределенность диагноза заболеваний из группы $Z_1, ..., Z_N$ при использовании специфичного значения zP_j^i

$$H\left(Z / zP_j^i\right) = -\sum_{k=1}^{N} p\left(Z_k / zP_j^i\right) \log_2 p\left(Z_k / zP_j^i\right);$$

- неопределенность диагноза заболеваний из группы $Z_1, ..., Z_N$ при использовании показателя P_i

$$H\left(Z / P_i\right) = \sum_{j=1}^{n_i} p_j^i H\left(Z / zP_j^i\right).$$

c) Определить величину прироста информации по формуле

$$J\left(Z / V_i\right) = H\left(Z\right) - H\left(Z / P_i\right).$$

Шаг 3. Выбрать показатель с максимальным приростом информации (принцип жадной максимизации прироста информации – на каждом шаге выбирается тот признак, при разделении по которому прирост информации оказывается наибольшим) в качестве корня дерева, при этом из корня выходит столько дуг, сколько специфичных значений содержит шкала соответствующего показателя.

Шаг 4. Затем процесс продолжается в каждой из полученных висячих вершин до тех пор, пока не будет определено такое значение текущего показателя, которое позволит однозначно поставить диагноз.

В результате применения данного алгоритма к конкретному набору данных будет построено диагностическое дерево, в котором каждому уровню ставится в соответствие показатель P_i, всего уровней n. Уровни располагаются в соответствии с порядком, который сформировался при работе алгоритма. Из каждой вершины (показатель P_i) выходит столько дуг, сколько специфичных значений содержит шкала $ScaleP_i$. Каждой дуге – значению zP_j^i – приписывается относительная частота появления этого значения в редуцированных экземплярах обучающего множества. Под редуцированным экземпляром обучающего множества подразумевается экземпляр, в котором оставлены компоненты векторной оценки, соответствующие пути из корня в данную вершину диагностического дерева.

В общем случае выбор вершины для ветвления может осуществляться на основе различных критериев (*Gain, GainRatio, SplitInfo, Gini*). Пусть множество S содержит экземпляры всех классов C_k. Для характеристики экземпляров используются показатели $\{P\}$, каждый из которых имеет шкалу $Values\left(P\right) = \{v\}$. Требуется разбить S на подмножества, ассоциированные с классами C_k. При построении дерева решений будем использовать критерий $Gain$: выбирается та вершина, у которой критерий прироста информации максимален

$$Gain\left(S, P\right) = H\left(S\right) - \sum_{v \in Values(P)} \frac{|S_v|}{|S|} H\left(S_v\right),$$

где S_v – множество экземпляров, у которых показатель P принимает значение v;

Поскольку энтропия, по сути, степень хаоса (или неопределенности) в системе, то уменьшение энтропии называют приростом информации.

2. Пример построения диагностического дерева для лейкоплакии

Лейкоплакия относится к одной из разновидностей кератозов, характеризующихся хроническим течением и поражающих слизистую оболочку полости рта и красную кайму губ. Одним из современных методов ее диагностики является оптическая когерентная томография (ОКТ) – получение прижизненных изображений клеточных структур с высокой разрешающей способностью [5]. Метод ОКТ способен визуализировать морфологические структуры слизистой оболочки полости рта на уровне слоев и внутритканевых элементов в норме и при патологии. ОКТ-изображения отличаются наличием слоистой структуры, характером границы между слоями, степенью однородности/неоднородности, наличием очагов и другими

признаками. Различают следующие типы лейкоплакии: простая (начальная) форма (два слоя, граница между слоями ровная и непрерывная, в нижнем слое видны области низкой яркости) – **ПФ** и верукозная форма, при этом существуют две разновидности: бляшечная форма (визуализируются три горизонтальных слоя; граница между слоями – нерезкая, извилистая; верхний слой неоднородный, высота увеличена по сравнению с нормой; яркость высокая; нижний слой неоднородный) – **БФ** и эрозивная форма (визуализируются три горизонтальных слоя; присутствуют линейные очаги затемнения; контраст между слоями значительно снижен или потерян; плотность клеточных компонентов увеличена) – **ЭФ**. Таким образом предполагается, что по изображениям ОКТ можно сделать предварительный диагноз относительно формы лейкоплакии.

Выделим следующие признаки на ОКТ:

- *количество слоев*: два (1), более двух (0);
- *уровень однородности клеточных структур между слоями*: Сниженный (С), Нормальный (Н), Повышенный (П);
- *контрастность*: Низкая (Н), Нормальная (Норм), Высокая (П).

Задача заключается в установлении диагноза: ПФ, БФ, ЭФ. Обучающее множество задано в табл. 1, количество пациентов равно 8.

Таблица 1

Обучающее множество

Количество слоев	Уровень однородности клеточных структур	Контрастность	Диагноз
0	Н	В	БФ
1	П	Н	ПФ
0	Н	Норм	БФ
0	Н	Н	ЭФ
0	С	В	БФ
1	С	Н	ЭФ
0	П	Норм	БФ
0	Н	Н	ЭФ

Рассчитаем прирост информации.

$$H\left(Диагноз\right) = -\sum_{i=1}^{s} \frac{m_i}{n} \ln \frac{m_i}{n} = -\frac{4}{8}\ln\frac{4}{8} - \frac{1}{8}\ln\frac{1}{8} - \frac{3}{8}\ln\frac{3}{8} = 0,9743.$$

$$Gain\left(количество\ слоев\right) = H\left(диагноз\right) - \frac{2}{8}H(количество\ слоев\ (2), диагноз) -$$

$$-\frac{6}{8}H(количество\ слоев\ (>2), диагноз) =$$

$$= 0,9743 - \frac{2}{8}\left(-\frac{1}{2}\ln\frac{1}{2} - \frac{1}{2}\ln\frac{1}{2}\right) - \frac{6}{8}\left(-\frac{4}{6}\ln\frac{4}{6} - \frac{2}{6}\ln\frac{2}{6}\right) = 0,3239.$$

$$Gain(УОКС) = H(диагноз) - \frac{2}{8}H(УОКС(Сниженный), диагноз) -$$

$$-\frac{4}{8}(УОКС(Нормальный), диагноз) - \frac{2}{8}(УОКС(Повышенный), диагноз) =$$

$$= 0,9743 - \frac{2}{8}\left(-\frac{1}{2}\ln\frac{1}{2} - \frac{1}{2}\ln\frac{1}{2}\right) - \frac{4}{8}\left(-\frac{2}{4}\ln\frac{2}{4} - \frac{2}{4}\ln\frac{2}{4}\right) - \frac{2}{8}\left(-\frac{1}{2}\ln\frac{1}{2} - \frac{1}{2}\ln\frac{1}{2}\right) = 0,2811.$$

Таблица 2

Обучающее множество после редуцирования (*Контрастность = Низкая* (*Н*))

Количество слоев	Уровень однородности клеточных структур	Контрастность	Диагноз
1	П	Н	ПФ
0	Н	Н	ЭФ
1	С	Н	ЭФ
0	Н	Н	ЭФ

Таблица 3

Обучающее множество после редуцирования (*Контрастность=Норм* (*Норм*))

Количество слоев	Уровень однородности клеточных структур	Контрастность	Диагноз
0	Н	Норм	БФ
0	П	Норм	БФ

Таблица 4

Обучающее множество после редуцирования (*Контрастность=Высокая* (*В*))

Количество слоев	Уровень однородности клеточных структур	Контрастность	Диагноз
0	Н	В	БФ
0	С	В	БФ

$$Gain(Контрастность) = H(диагноз) - \frac{4}{8}H(Контрастность(Низкая), диагноз) -$$

$$-\frac{2}{8}(Конрастность(Нормальная), диагноз) - \frac{2}{8}(Контрастность(Высокая), диагноз) =$$

$$= 0,9743 - \frac{4}{8}\left(-\frac{1}{4}\ln\frac{1}{4} - \frac{3}{4}\ln\frac{3}{4}\right) - \frac{2}{8}\left(-\frac{2}{2}\ln\frac{2}{2}\right) - \frac{2}{8}\left(-\frac{2}{2}\ln\frac{2}{2}\right) = 0,6931.$$

Согласно вычислениям, в качестве корня диагностического дерева нужно выбрать показатель «Контрастность». Разделим обучающее множество на подмножества.

Как видно из таблиц 3 и 4 постановка диагноза для этих случаев тривиальна. Для таблицы 2 постановка диагноза неоднозначная. Рассмотрим таблицу 2 и рассчитаем прирост информации. Рассчитаем прирост информации.

$$H\left(Диагноз\right)=-\sum_{i=1}^{s}\frac{m_i}{n}\ln\frac{m_i}{n}=-\frac{3}{4}\ln\frac{3}{4}-\frac{1}{4}\ln\frac{1}{4}=0,5624$$

$$Gain\left(\ количество\ слоев\right)=H\left(диагноз\right)-\frac{1}{2}H\left(количество\ слоев\ (2),диагноз\right)-$$

$$-\frac{1}{2}H\left(количество\ слоев\ (>2),диагноз\right)=0,5624-\frac{1}{2}\left(-\frac{1}{2}\ln\frac{1}{2}-\frac{1}{2}\ln\frac{1}{2}\right)-\frac{1}{2}\left(-\frac{2}{2}\ln\frac{2}{2}\right)=0,2158.$$

Таблица 5

Обучающее множество после редуцирования (*Контрастность=Н* и *УОКС=С*)

Количество слоев	Уровень однородности клеточных структур	Диагноз
1	С	ЭФ

Таблица 6

Обучающее множество после редуцирования (*Контрастность=Н* и *УОКС=Н*)

Количество слоев	Уровень однородности клеточных структур	Диагноз
0	Н	ЭФ
0	Н	ЭФ

Таблица 7

Обучающее множество после редуцирования (*Контрастность=Н* и *УОКС=П*)

Количество слоев	Уровень однородности клеточных структур	Диагноз
1	П	ПФ

$$Gain\left(УОКС\right)=H\left(диагноз\right)-\frac{1}{4}H\left(УОКС\left(Сниженный\right),диагноз\right)-$$

$$-\frac{2}{4}\left(УОКС\left(Нормальный\right),диагноз\right)-\frac{1}{4}\left(УОКС\left(Повышенный\right),диагноз\right)=$$

$$=0,5624-\frac{1}{4}\left(-\frac{1}{2}\ln\frac{1}{2}\right)-\frac{2}{4}\left(-\frac{2}{2}\ln\frac{2}{2}\right)-\frac{1}{4}\left(-\frac{1}{2}\ln\frac{1}{2}\right)=0,3891.$$

Таким образом, следующим критерием в случае, если Контрастность *Низкая*, является Уровень однородности клеточных структур (УОКС). В результате получим следующие подмножества.

Анализ приведенных таблиц позволяет сформулировать следующие правила принятия решений:

1) Если **Количество слоев** = *0* (больше двух слоев) и **Контрастность** = *Нормальная* или *Высокая*, то **Диагноз** = БФ.

2) Если **Количество слоев** = *1*(2 слоя) и **УОКС** = *Повышенный*, то **Диагноз** = ПФ,

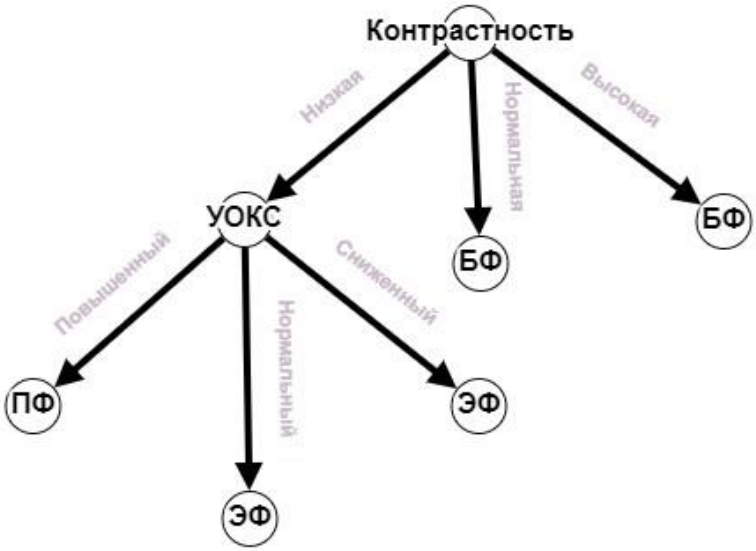


Рис. 2. Диагностическое дерево для лейкоплакии ротовой полости (по ОКТ)

3) Если **Количество слоев** = 2(больше двух слоев) и **УОКС** = *Сниженный* или *Нормальный*, то **Диагноз** = ЭФ.

На рис. 2 представлено диагностическое дерево для лейкоплакии ротовой полости.

Заключение

На основе разработанного алгоритма было построено диагностическое дерево, которое может быть использовано для обоснования постановки конкретного диагноза по изображениям ОКТ, а также для построения продукционных правил экспертной системы диагностики типа лейкоплакии ротовой полости.

Литература

1. Григорьев С.С. Гиперкератозы слизистой оболочки рта (красный плоский лишай, лейкоплакия): Учебно-методические рекомендации / С.С. Григорьев, Г.И. Ронь, А.А. Епишова. – Екатеринбург : Издательский дом «Тираж», 2019. – 72 с.

2. Гончарик П.В. Лейкоплакия слизистой оболочки полости рта / П.В. Гончарик, Р.Н. Супруновский, Г.Д. Панасюк. – Гомель : ГУ «РНПЦ РМиЭЧ», 2019. – 27 с.

3. Вероятностно-статистические методы принятия решений: теория, примеры, задачи : учебное пособие / А. П. Науменко, И.С. Кудрявцева, А. И. Одинец ; Минобрнауки России, ОмГТУ. – Омск : Изд-во ОмГТУ, 2018. – 85 с.

4. Дюк В.А. Информационные технологии в медико-биологических исследованиях / В.А. Дюк, В. Эммануэль. – Санкт-Петербург : Питер, 2003. – 528 с.

5. Рабинович О.Ф. Оценка эффективности комплексного лечения тяжелых форм лейкоплакии слизистой оболочки рта /О.Ф. Рабинович, И.М. Рабинович, А.Д. Островский, А.А. Тогонидзе // Клиническая стоматология, 2014. – №2. – С. 110-114.

УДК 004

СПОСОБЫ ПОВЫШЕНИЯ ДОСТУПНОСТИ ПРИ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ

Н. В. Шебекина, К. Г. Резников

Воронежский государственный университет

Введение

Интернет ресурсы повсеместно вошли в жизнь современного человека. Интернет компании в сфере услуг, магазины, разработчики и другие продвинутые авторы активно и широко используют интернет площадки для своего продвижения. Благодаря веб-разработчикам, которые в свою очередь воплощают идеи дизайнеров, перед пользователями открывается продуманный интерфейс сервисов. При разработке проекта важно помнить о многогранности общества и особенностях многих его представителей, и о том, что ресурс, должен быть доступен для всех. Особенно приоритетен в наше время вопрос социализации людей с особенностями развития и ограниченными возможностями. А также о продуктивной доступности ресурса для широкого международного сообщества.

Данная статья раскрывает важность ряда особенностей, которые при разработке веб-приложений позволяет повысить их информативность и доступность для более широкого круга пользователей. Рассматривается ряд подходов и методологий, которые помогут каждому разработчику в данной сфере. Начиная с простых и интуитивно понятных способов, заканчивая более индивидуальными и точечными функциями работы с HTML и CSS [1].

1. Базовые принципы верстки

Рассмотрим первичные принципы для повышения доступности.

Фокусные состояния элементов. В зависимости от браузера, такие состояния изменяются, в браузере доступна возможность это состояние изменять вручную. Рекомендуется делать объекты более заметными и не убирать фокусное состояние совсем, как показано на рис. 1 и рис. 2.

Рис. 1. Внешний вид фокусного состояния с стилизацией

Рис. 2. Внешний вид фокусного состояния без стилизации

Разметка. Под разметкой подразумеваются специализированные теги и их размещение в HTML документе. Самыми важными аспектами является то, что на странице должен быть только один заголовок первого уровня, а остальные должны располагать иерархично в каждом

блоке страницы. Также, на сколько это возможно, необходимо стараться верстать страницу с использованием специализированных тегов. В браузере разрешено использовать и стандартный блочный тег *div*, но использование специализированных конструкций позволяет сделать страницу более валидной и более читабельной не только для пользователя, но и для сборщика и поисковых сервисов. В первую очередь такой подход упростит слушание при помощи плагинов для чтения для людей с нарушением зрения [2]. Также это повысит доступность навигацию сайту для всех пользователей, а поиск в браузере чаще будет основан не только на ключевых словах, но и на тематических блоках тегов

Атрибуты тегов. Например, атрибут *alt* тега *img*, который позволяет задать описание изображения для плагина чтения или в случае если изображение не удалось загрузить. Такой атрибут лучше всего использовать на важных изображениях [3]. В случае декоративных изображений, допустимо оставить тег пустым. Также у тегов есть атрибут *type*, который позволяет конкретизировать область использования тега. Чаще применяется для указания кнопок и полей ввода разного вида.

2. Паттерны повышения доступности

Часто в процессе верстки, разработчик сталкивается с необходимостью скрывать объекты на странице. Например, с помощью методов *visibility: hidden и display: none*, а также атрибута hidden в html разметке. У этих способов есть свои достоинства и недостатки. Как было сказано выше, такие методы скрывают элемент из дерева доступности, а свойства, *display: none* и *hidden* также убирают элементы из потока, в то время как *visibility: hidden* скрывает элементы со страницы. Такие способы скрытия могут быть полезны в случаях, когда элементы страницы должны появляться по клику, а до этого плагин чтения не должен распознавать его на странице.

Возникают случаи, когда требуется скрыть элемент визуально, но при этом оставить его для чтения. Один из примеров могут стать элементы навигации, которые часто выполнены в виде изображений и помещены не как картинка, а как фон контейнера. В таком случае плагин чтения не прочитает изображение или иконку и, соответственно, пропустит его. Данную проблему решает паттерн *visually-hidden*. Рассмотрим свойства для описания данного паттерна подробнее:

– *position: absolute* – скрывает элемент из потока, но сохраняет в дереве;

– *width: 1px и height: 1px* – задают ширину и высоту элемента соответственно, делая элемент самого маленького размера;

– *overflow: hidden* – при увеличения размеров блока, скрывает прокрутку;

– *clip: rect(0 0 0 0)* – видимая область обрезается до нулевого значения;

– *margin: 1px* – элемент сдвигается на размер самого себя, тем самым удаляется из поля зрения;

– *padding: 0* – обнуление внутренних отступов, при наличии по умолчанию;

– *border: none* – удаление границы элемента.

Таким образом элемент может быть скрыт из потока и видимо не заметен, но дерево доступности по прежнему будет его распознавать и плагин сможет его озвучить. Помимо доступности, данный паттерн может быть применен для декоративных целей, например, при создании дизайна для чекбоксов и радиокнопок. Чтобы прописать им новый стиль, необходимо скрыть изначальный вид и реализовать новый дизайн с помощью CSS.

Следующий важный набор паттернов – *ARIA. Accessible Rich Internet Applications* (перевод с англ. – доступные многофункциональные интернет-приложения»). Данной

аббревиатурой характеризуют набор дополнительных инструментов, которые помогают сделать интерфейс более доступным в сравнении с классическими инструментам HTML [4].

ARIA–ролей достаточно много и для верного их использования необходимо досконально в них погружаться и знать некоторые аспекты их использования.

Изначально, у каждого элемента уже есть своя роль, которую можно изучить в инструментах разработчика. Некоторые роли будут более точно передавать назначение того или иного элемента на странице. ARIA–роли также помогают в рефакторинге, а именно, когда необходимо исправлять или дополнять устаревший код или код который был сверстан практически использую только теги *div* без учета принципов семантики. При помощи ролей возможно изменять непосредственные роли в дереве доступности и сделать код более подходящим для особенных пользователей.

Помимо ARIA–ролей, существуют также ARIA–атрибуты. Атрибутов многим больше чем ролей и область применения у них шире. Давайте рассмотрим несколько атрибутов, функции которые они выполняют и то, как это помогает сделать страницу более доступной.

– *aria-hidden* – атрибут принимает значение булевое значение. Чаще используется для скрытия изображений, которые невозможно скрыть иными способами. Например, иконки ссылок или другие изображения невыносимые как фоновые;

– *aria-label* – атрибут, значения которого указывает как озвучивать элементы;

– *aria-description* – который в качестве значения принимает описание которое будет необходимо озвучить, как расширенное. Также возможно и дорабатывать озвучание состояний и указывать что изменится при нажатии той или иной кнопки при изменении фокусного состояния.

Существуют и другие атрибуты, для повышения доступности страницы, а выше приведены те, которые используются чаще всего.

3. Прочие методы повышения доступности

В современных компьютерах широкий спектр настроек для адаптации рабочего процесса под особенности каждого пользователя. Не секрет, что многие современные браузеры умеют отслеживать состояния которые применяются в системе. Для адаптации веб приложений под пользователя может помочь директива *@media,* которой в качестве параметра передается ряд параметров.

Например, *prefers-reduced-motion,* который указывает на то, что пользователь предпочитает минимизировать резкие движения и анимации [5]. Либо, *prefers-color-scheme,* который показывает предустановленные предпочтения пользователя в цветовой схеме, темной или светлой. Проверить работоспособность данных пареметров можно в инструментах разработчика в расширенных настройках, изображенные на рис. 3.

Все изображения необходимо строго делить на две категории:

– контентные изображения, которые необходимо озвучивать, т.е добавлять им *alt* и при этом заполнять его правильно;

– декоративные изображения, которые необходимо скрыть из дерева доступности и озвучить при помощи других методов, если необходимо.

Например, иконки раскрывающихся меню правильно указывать как фоновое изображение контейнера, а логотипы на веб-сайтах правильно задавать как объект для его большей заметности плагинами для чтения.

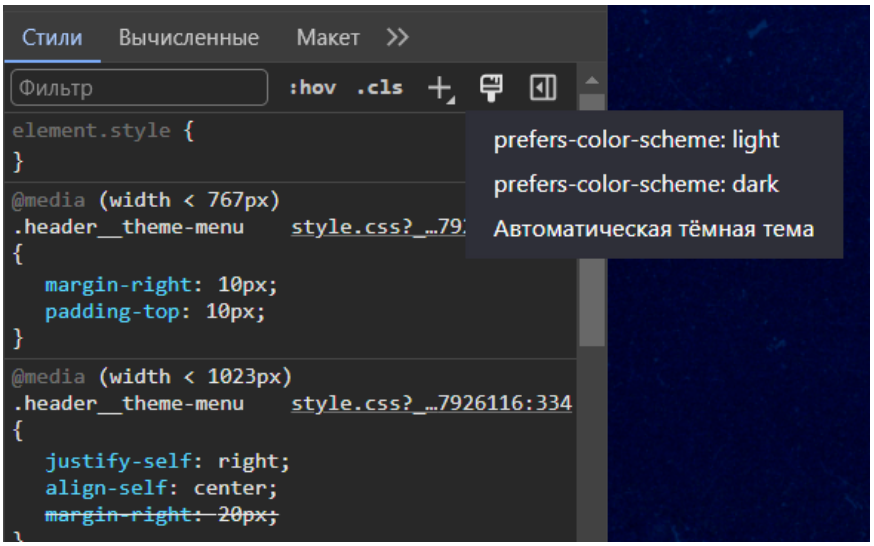


Рис. 3. Параметр отображения системной светлой или темной темы

Заключение

Таким образом, на данный момент в сообществе веб-разработчиков существует достаточно много способов с помощью которых можно упростить использование интернет ресурсов пользователям с ограниченными возможностями, а также сделать веб-приложение более универсальным для пользователей из других стран.

Методы представленные в статье являются основой. Разработчикам рекомендуется придерживаться их, чтобы сделать сайт практичнее и комфортнее для всех участников интернет сообщества.

Литература

1. Резников К. Г. Разработка программного обеспечения для визуализации трехмерных поверхностей в веб-браузере / К. Г. Резников, С. Н. Медведев // Вестник ВГТУ. Том 17, №6. – Воронеж: ВГТУ, 2021 - С. 13-19.

2. Как скрыть содержимое от скринридеров. – Режим доступа: https://doka.guide/a11y/content-hidden/. – (Дата обращения: 02.04.2024).

3. Скринридеры. – Режим доступа: https://doka.guide/a11y/screenreaders/ – (Дата обращения: 02.04.2024).

4. Основы доступного HTML – Режим доступа: https://doka.guide/a11y/a11y-html/– (Дата обращения: 02.04.2024).

5. Доступность – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Accessibility/ – (Дата обращения: 02.04.2024).

УДК 004.62

РАЗРАБОТКА ПЛАНА АВТОМАТИЧЕСКОЙ СБОРКИ ПРОЕКТА ПО МЕТОДОЛОГИИ CI/CD С ИСПОЛЬЗОВАНИЕМ СЕРВИСА BAMBOO

С. А. Шебуняева

Воронежский государственный университет

Введение

Работа представляет собой решение на основе инструмента непрерывных интеграции и развертывания (CI/CD), который объединяет автоматизированную сборку, тестирование и развертывание программного обеспечения, тем самым упрощая и ускоряя разработку ПО.

В настоящее время все новые компоненты проекта и изменения в коде не обновляются автоматически и развертываются разработчиками вручную на тестовой и производственной среде после каждого изменения. Непрерывная интеграция обеспечит постоянное автоматическое слияние рабочих копий проекта в общую основную ветвь и выполнение частых автоматических сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.

Чтобы обеспечить непрерывную сборку компонентов проекта, создание необходимых пакетов и своевременную интеграцию вносимых в код проекта изменений, была использована методология CI (непрерывная интеграция), которая направлена на проверку кода, модульное тестирование, компиляцию и автоматическое развертывание кода проекта (рис. 1).

Рис 1. Схема непрерывной интеграции

Для CI/CD существуют четыре руководящих принципа:

1. Разделение ответственности. Каждый участник процесса ответственен за тот или иной этап жизненного цикла продукта.
2. Снижение рисков. Контроль корректности бизнес-логики, проверка этапов разработки, улучшение хранения и обработки данных и тд.
3. Сокращение цикла обратной связи. Стремление к увеличению скорости внесения изменений и согласования правок.
4. Реализация среды. Команде разработки необходимо единое рабочее окружения для контроля версий и качества, масштабируемости, отказоустойчивости и других критериев.

Методология CI/CD подразумевает разделение процесса разработки на этапы (рис. 2):

- Планирование
- Написание кода
- Сборка
- Ручное тестирование
- Развертывание
- Поддержка и мониторинг

Рис 2. Этапы процесса разработки по методологии CI/CD

1. Инструменты реализации

Для разработки плана автоматической сборки проекта использован инструмент CI/CD — Bamboo. Bamboo предоставляет ряд функций, которые делают его предпочтительным выбором для реализации непрерывной интеграции:

1. Автоматизированная сборка и тестирование. Bamboo автоматизирует процессы сборки, тестирования и верификации, снижая риск ошибок, связанных с человеческим фактором, и обеспечивая качество кода.
2. Комплексное развертывание. Bamboo легко интегрируется со средствами развертывания, упрощая развертывание в различных средах.
3. Параллельные сборки. Обеспечивается более быструю обратную связь по изменениям кода и сокращается общее время сборки.
4. Управление артефактами. Bamboo управляет артефактами сборки, упрощая их отслеживание и распространение на разных этапах.
5. Проверка развертывания. Bamboo может выполнять автоматические тесты проверки развертывания, чтобы гарантировать, что развернутый код функционирует должным образом.
6. Отчетность. Bamboo предоставляет подробные отчеты и информацию о состоянии и работоспособности конвейеров CI/CD.

Схема сборки разрабатываемого проекта, включающая используемые инструменты разработки и развёртывания, представлена на рис. 3.

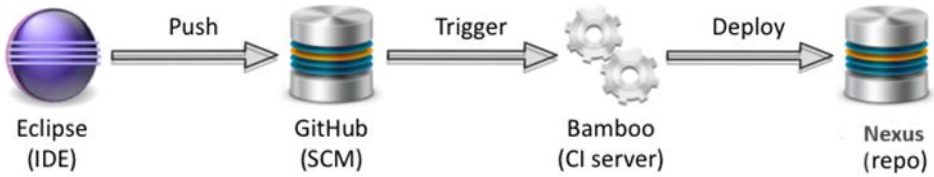


Рис 3. Схема сборки разрабатываемого проекта

Bamboo использует концепцию проектов, планов, этапов, заданий и задач, представленных в иерархической структуре (рис. 4). Проект — это пространство имен, содержащее разное количество планов. Каждый план может включать несколько этапов, выполняемых последовательно. Каждый этап, в свою очередь, состоит из нескольких параллельно выполняемых заданий. Наконец, серия задач составляет задание. Они выполняются последовательно в запланированном рабочем узле.

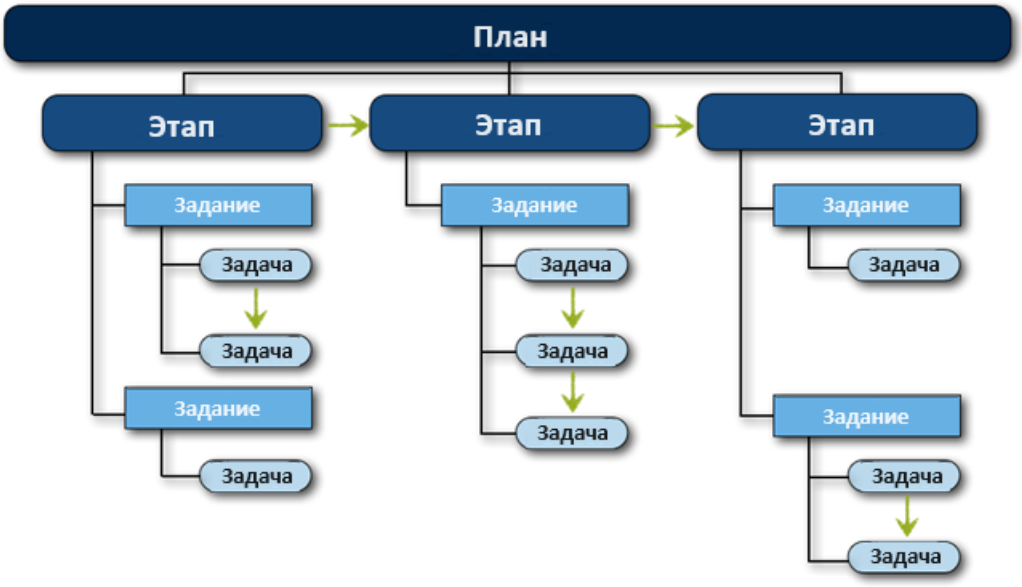


Рис 4. Концепция плана построения проекта

2. Выгрузка исходного кода из удалённого репозитория

Для сборки проекта необходима интеграция с системой контроля версий. После установки и настройки Bamboo, для работы с исходным кодом, настраивается доступ к удаленному репозиторию посредством авторизации и интегрирования репозитория в проект (рис. 5).

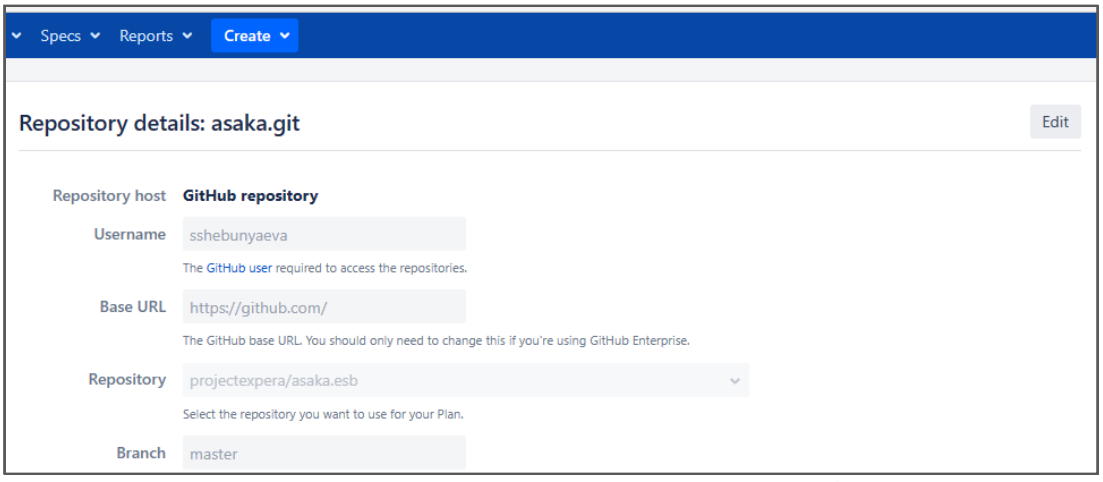


Рис 5. Настройка доступа к репозиторию GitHub

3. Создание плана конфигурации

В сервисе Bamboo был создан план, который определяет шаги, необходимые для сборки и тестирования компонентов проекта. В плане указывается репозиторий по умолчанию и описывается, как запускается сборка. В нем же в дальнейшем отображаются уведомления о результатах сборки.

В плане конфигурации создан этап «Build Delivery» - этап сборки поставки компонентов проекта, который включает в себя следующие задачи (рис. 6):

- Source code checkout — выгрузка исходного кода из репозитория в GitHub, проверка загруженных коммитов и возможных конфликтов кода;
- Build bar-file — генерация исполняемого файла с расширением bar. Это формат файлов brew mobile platform для упаковки одного или более модулей в одном архиве, чтобы развертывание различных модулей на сервере приложений происходило одновременно и согласовано;
- Build delivery — сборка необходимых для развертывания компонентов проекта файлов (.bar и .mqs) и выгрузка их на локальной машине по заданному пути;
- Upload to Nexus — выгрузка собранного пакета файлов на платформу Sonatype Nexus Repository.

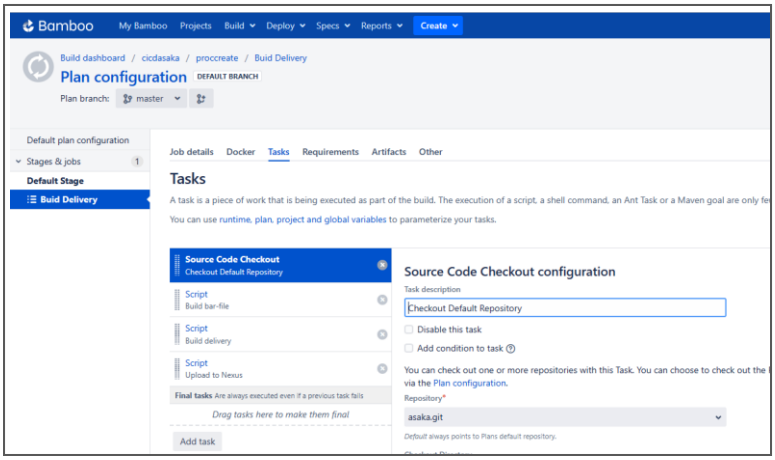


Рис 6. План конфигурации и его задачи

Задачи Build bar-file, Build delivery и Upload to Nexus выполняются с помощью соответствующих bash-скриптов. Исходный код скриптов расположен в репозитории проекта, в настройка выполнения задачи указывается путь к пакетному скрипту, который необходимо выполнить во время выполнения той или иной задачи плана. Так же указываются аргументы, которые подаются на вход пакетного скрипта для его выполнения.

Например, для выполнения задачи Build bar-file необходимо выполнить скрипт build.bat с аргументами -c ${bamboo.COMPONENT_NAME} (имя компонента для сборки и развертывания), -obd ${bamboo.build.working.directory}\delivery (путь для выгрузки файла на локальную машину), -tp 'C:\Program Files\IBM\ACE\11.0.0.16' (актуальная версия среды разработки). После настройки всех необходимых параметров, данная задача будет выполняться автоматически при запуске плана. Данная задача настройки плана представлена на рис. 7.

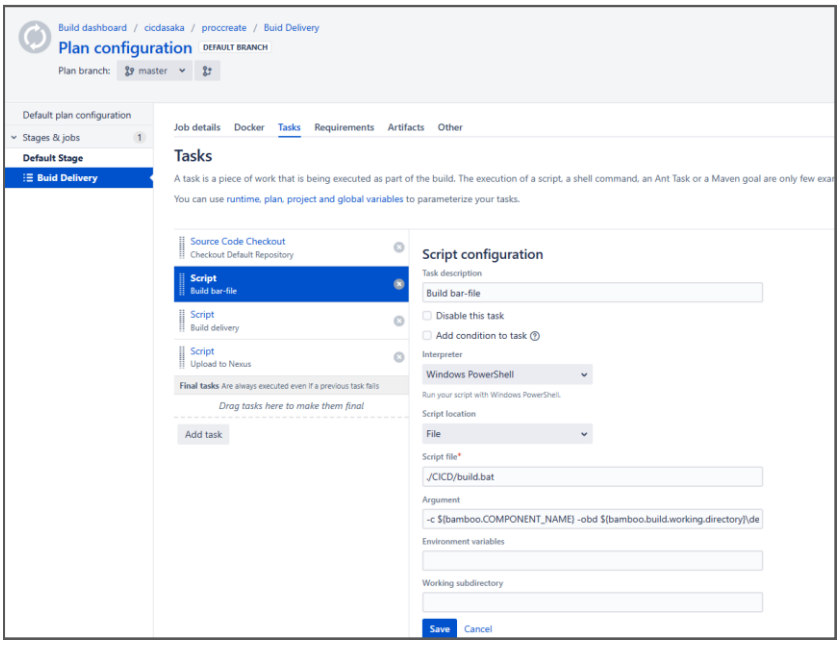


Рис 7. Задача Build bar-file

Фрагмент пакетного скрипта, выполняющего сборку bar-файла, представлен ниже.

```
REM Get libraries from .project
set /a index=0
if exist ".\%componentName%\.project" (
      set "xml_project_file=.\%componentName%\.project"
)else (
      echo directory "%componentName%" doesn't exist
      set "xml_project_file=.\.project"
)
for /f "tokens=1-9 delims=></" %%A in (%xml_project_file%) do (
      if "%%B" == "project" (
            set projects[!index!]=%%C
            set /a index+=1
      )
)

REM Make links for libraries
echo Create links:
set /a index=0
:mkLinksLibsLoop
if defined projects[%index%] (
            echo !projects[%index%]!
            set javaproject=!projects[%index%]!Java
            if /i "!projects[%index%]:~0,2!"=="Mc" (
                        MKLINK /j .\!projects[%index%]! ..\..\..\Mc
                        \!projects[%index%]!\src\!projects[%index%]!
            if exist "..\..\..\Mc\!projects[%index%]!\src\!javaproject!" (
                        echo !javaproject!
                        MKLINK /j .\!javaproject! ..\..\..\Mc
                              \!projects[%index%]!\src\!javaproject!
                  )
            ) else if /i "!projects[%index%]:~0,3!"=="Lbt" (
                  MKLINK /j .\!projects[%index%]! ..\..\..\Lbt\
                        !projects[%index%]!\src\!projects[%index%]!
                  if exist
```

```
                    "..\..\..\Lbt\!projects[%index%]!\src\!javaproject!" (
                    echo !javaproject!
                    MKLINK /j .\!javaproject! ..\..\..\Lbt\
                          !projects[%index%]!\src\!javaproject!
                )
            )
            set /a index+=1
            goto :mkLinksLibsLoop
)

REM Create bar
echo Create bar %componentName%:
CALL "mqsicreatebar" -data "." -b %outBarDir%\%componentName%\%componentName%.bar
-a %componentName%
-deployAsSource -cleanBuild
```

4. Выполнение тестов и просмотр результатов

После создания пакетных скриптов и успешной настройки всех шагов созданного плана, необходимо запустить сборку компонентов проекта (рис. 8). Для тестирования задач плана была запущена параметризированная сборка с указанием конкретного компонента для сборки и развертывания — Ac.ABS.Account.V1 — сервис доступа для автоматизированной банковской системы (АБС).

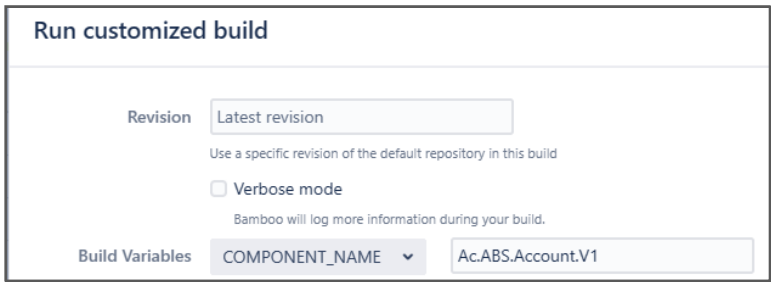


Рис 8. Запуск плана

После выполнения первой задачи плана — выгрузки исходного кода — в меню выполняемой тестовой сборки отображаются последние изменения в репозитории, включающие измененные в коммите файлы и авторы изменений (рис. 9).

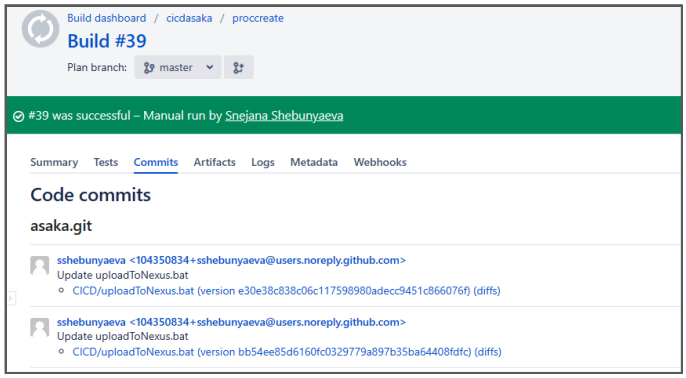


Рис 9. Отображение последних изменений исходного кода

Далее собранный пакет созданных файлов сохраняется по заданному пути на локальной машине – для каждого компонента в отдельной директории с именем компонента. После

выполнения данного шага по указанному пути создается директория с названием последнего заданного в параметрах плана компонента Ac.ABS.Account.V1 (рис. 10).

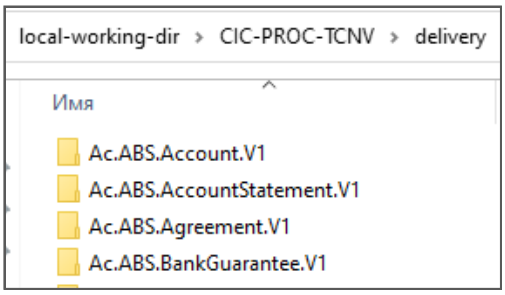


Рис 10. Директория с собранными пакетами файлов

Следующий шаг выполняет выгрузку сгенерированных файлов на платформу Sonatype Nexus Repository. Файлы для указанного компонента размещены в директории uz/asakabank/esb/Ac.ABS.Account.V1/1.00.00 (рис. 11).

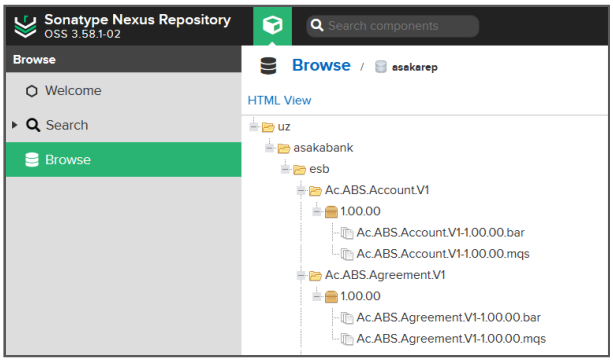


Рис 11. Репозиторий Nexus

После выполнения плана во вкладке последних выполненных планов появляется запись об успешной сборке компонента (рис. 12).

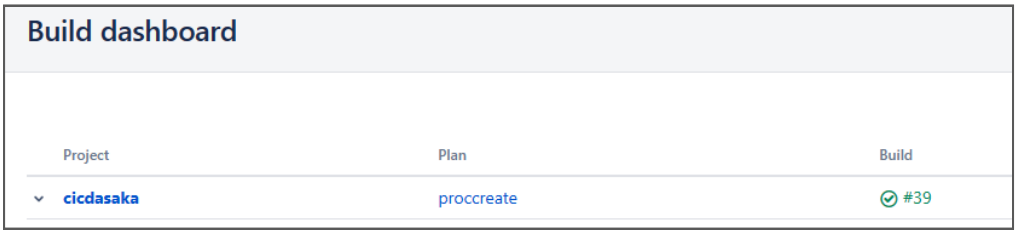


Рис 12. Таблица выполненных сборок

Заключение

CI/CD — набор методов и практик, отвечающий требованиям современной ПО-разработки. Принципы непрерывной интеграции и доставки помогают внедрять решения быстро, оперативно согласовывать их и доводить до релиза, не рискуя при этом качеством продукта.

Bamboo CI из среды Atlassian в сочетании с системой управления тестированием обеспечивает более быстрый, продуктивный и экономичный подход к внедрению новых функций продукта и устраняет риски для производственной среды.

Разработанный план сборки компонентов проекта позволяет автоматически отслеживать изменения в исходном коде, создавать необходимые для развертывания ПО файлы и выгружать их в выбранный репозиторий, а также на локальную машину, сведя к минимуму вероятность ошибок и потери данных. Тестирование выполняемых в плане задач показало, что данные функции выполняются корректно.

Литература

1. Chris Timberlake, Automating DevOps with GitLab CI/CD Pipelines: Build efficient CI/CD pipelines to verify, secure, and deploy your code using real-life examples / Chris Timberlake ; Packt Publishing Ltd., 2023. — 348 p.
2. Jez Humble, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation / Jez Humble, David Farley ; Pearson Education, Inc., 2011. — 497 p.
3. Джез Хамбл, Непрерывное развертывание ПО / Джез Хамбл, Дейвид Фарли ; Пер. с англ. — М. : ООО «И.Д. Вильямс», 2011. — 432 с.
4. Sander Rossel, Continuous Integration, Delivery, and Deployment / Sander Rossel ; Packt Publishing Ltd., 2017. — 458 p.
5. Дюваль Поль М. Непрерывная интеграция. Улучшение качества программного обеспечения и снижение риска / Дюваль Поль М., Матиас Стивен, Гловер Эндрю ; Пер. с англ. — М. : ООО «И.Д. Вильямс», 2016. — 240 с.
6. Документация сервиса Bamboo. – Режим доступа: https://confluence.atlassian.com/bamboo/bamboo-documentation – (Дата обращения: 04.01.2024).

УДК 621.3.019.3

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ДИАГНОСТИКИ ДАТЧИКОВ СЛОЖНЫХ ТЕХНИЧЕСКИХ УСТРОЙСТВ.

К.Л. Шкут, С.А. Маяцкий, Иванов И.А.

ВУНЦ ВВС «ВВА» им. проф. Н.Е. Жуковского и Ю.А. Гагарина.

Введение

Современные тепловые двигатели, газотурбинные установки, реакторы атомных электростанций являются сложными техническими устройствами, от надежной работы которых зависит выполнение частных и глобальных задач производства. Однако, ужесточение технических требований для таких устройств, обусловленное потребностью в повышении их эффективности, вызывает необходимость в усложнении конструкции, способов организации рабочего процесса, увеличения параметров рабочего процесса.

Так, например, для газотурбинных двигателей (ГТД) государственной авиации наблюдается устойчивая тенденция к увеличению параметров рабочего процесса для повышения коэффициента полезного действия (КПД) двигателя. (Рис. 1)

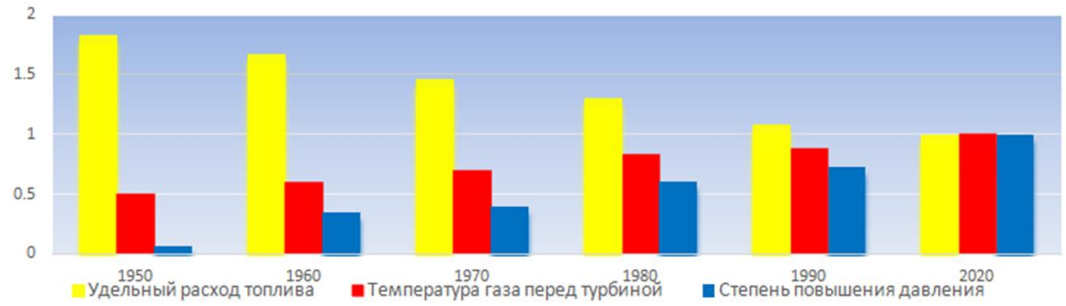


Рис. 1. Относительное изменение параметров рабочего процесса авиационных ГТД в процессе их технического совершенствования.

Из рисунка 1 видно, что удельный расход топлива, параметр характеризующий экономичность двигателя, уменьшается. Это происходит благодаря увеличению степени повышения давления в осевом компрессоре и температуры газа перед турбиной. Повышение технического совершенства авиационных ГТД достигается за счет усложнения конструкции двигателя (переход к двухконтурной схеме, применение нескольких каскадов осевых компрессоров, применения развитой механизации осевого компрессора). Для обеспечения устойчивой работы такого двигателя необходима сложная система автоматического управления (САУ) с надежными и точными датчиками.

Аналогичная ситуация для водо-водяных энергетических реакторов (ВВЭР). С 1995 года наблюдается рост мощности реакторов, температуры теплоносителя на выходе из активной зоны, массы загрузки двуокиси урана, но при этом габариты реактора остаются при мерно на там же уровне. При росте мощности в 5 раз за более чем 50 лет, размер реактора изменился в 1.1 раза. (Рис.2)

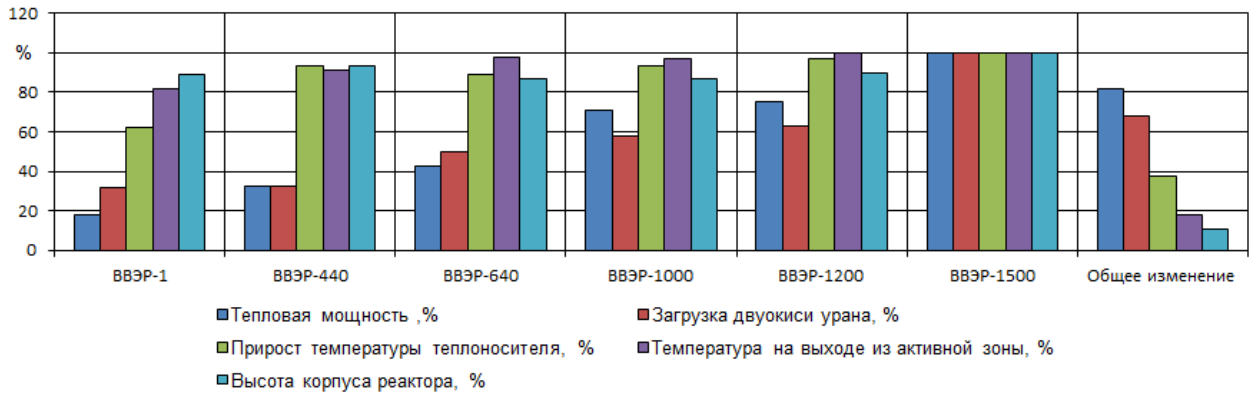


Рис. 2. Относительное изменение технических характеристик реакторов ВВЭР в процессе их технического совершенствования.

Для обеспечения надежности этих и подобных им технических систем, развивающихся по аналогичному принципу необходимо обеспечить точное управление параметрами рабочего процесса, а так же заведомо точное измерение этих параметров. Однако датчики часто необходимо устанавливать в высоконагруженные зоны, что негативно сказывается на их надёжности. Для обеспечения надежности датчика, и системы его автоматического управления в состав которой он входит, предлагается способ диагностики датчика по его собственным параметрам и с способ диагностики технического состояния объекта управления за счет реализации концепции интеллектуальной системы диагностики датчиков сложных технических устройств.

1. Обеспечение точности измерений

1.1. Определение потребного числа измерений датчиком

Датчик в качестве выходного сигнала имеет физическую величину, представляющую собой, интерпретацию случайной текущей величины, зарегистрированной чувствительным элементом датчика. Таким образом, выходной сигнал датчика можно представить как: математическое отображение значений показаний датчика $h(t)$ во времени, тогда показания датчика можно представить, как

$$u(t) = h(t) + f(t) \tag{1}$$

где – математическое отображение показаний датчика; $f(t)$ – функция равномерного распределения случайной величины, характеризующая шум датчика.

Так выходной сигнал с чувствительного элемента датчика является случайная текущая величина, то для повышения точности работы датчика необходимо обеспечить достаточное количество измерений, соответствующее закону больших чисел, согласно которому при увеличении числа измерений значение математического ожидания приближается к истинному значению измеряемой величины. Следовательно, необходимо определить количество опросов чувствительного элемента датчика ϑ, которая зависит от величины его погрешности ξ, и требуемого уровня точности измерения ψ.

Тогда число возможных событий Z вычисляется как:

$$Z = \frac{2 \cdot \xi}{\psi} \tag{2}$$

Отсюда вероятность получения случайного значения из диапазона измерений P_Z вычисляется как:

$$P_Z = \frac{1}{Z} = \frac{\psi}{2 \cdot \xi} \tag{3}$$

А число потребных измерений равно Z, следовательно, математическое ожидание измеряемой истинной физической величины равно:

$$M_{\text{фв}} = P_z \sum_{1}^{Z} x_i \tag{4}$$

где x_i – полученное с датчика действительное значение случайной текущей физической величины.

Для согласования частот опросов чувствительных элементов, и частот опросов самого датчика в составе САУ необходимо определить параметры частоты опросов при диагностировании его технического состояния по собственным данным.

1.2. Набор текста

Для диагностики отказа датчика по его собственным данным разработан способ анализа параметров на основе линейной экстраполяции экспоненциальной скользящей средней медианы сигнала датчика. Предлагаемый способ основан на способе фильтрации сигнала при помощи медианного фильтра, отличается введение экспоненциально скользящего среднего по медианам сигнала датчика и линейной экстраполяцией последних *n* значений скользящей медианы датчика, что позволяет прогнозировать тенденцию изменения показания сигнала без влияния на нее шумов, искажающих сигнал.

Первым этапом в обработке сигнала является вычисление медианы в заданном окне сигнала. Размер окна *N* определяется тактовой частотой опроса датчика САУ и скоростью переходных процессов у измеряемого параметра.

$$N = 0,25 \cdot \nu \tag{5}$$

где ν – частота опроса датчика.

По полученным значениям медианы строится экспоненциально скользящая средняя *S*:

$$S(t) = M \cdot \beta + (1 - \beta) \cdot S(t - 1) \tag{6}$$

где β – коэффициент сглаживания в диапазоне [0;1].

Чем ближе коэффициент β к 0, тем большее влияния на последующие значения оказывают значения с предыдущих итераций.

По последним двум значениям экспоненциально скользящей средней вычисляются параметры наклона *K* и смещения *B* линейной экстраполирующей функции. Применение линейной экстраполяции обусловлено тем, что частота опроса датчика значительно больше частоты колебаний исследуемого сигнала и точность линейной экстраполяции удовлетворительна и обеспечит максимальное быстродействие алгоритма.

$$K = \frac{M(t) - M(t-1)}{t - (t-1)} \tag{7}$$

$$B = M(t) - K \cdot t \tag{8}$$

Затем строится линейная экстраполяционная функция на расстояние $\frac{1}{\nu}$ от текущей точки измерений при известной частоте дискретизации ν.

$$y_{\text{экс}}(t) = K \cdot \left(t + \frac{1}{\nu} \right) + B \tag{9}$$

Если рассогласование показаний $\varsigma(t)$ с датчика в момент времени $t + \frac{1}{\nu}$ превышает заданное рассогласование $\varsigma_{\text{зад}}(t)$ скользящей медианы то датчик считается отказавшим.

$$\varsigma(t) = \left| y(t) - y_{\text{экс}}(t) \right| \tag{10}$$

Однако такой подход позволяет диагностировать только отказы датчиков, заключающиеся в резком изменение значений параметров рабочего процесса, а так же не позволяет отличить отказ датчика от отказа объекта, параметры рабочего процесса которого изменяются. Для решения этой проблемы дополнительно вычисляется уровень рассогласования между датчиками и их цифровыми двойниками.

2. Разработка цифровых двойников

Для диагностирования отказа датчика контроля параметров рабочего процесса необходимо вычислить рассогласования между датчиками и их цифровыми двойниками, по следующей схеме. (Рис. 8)

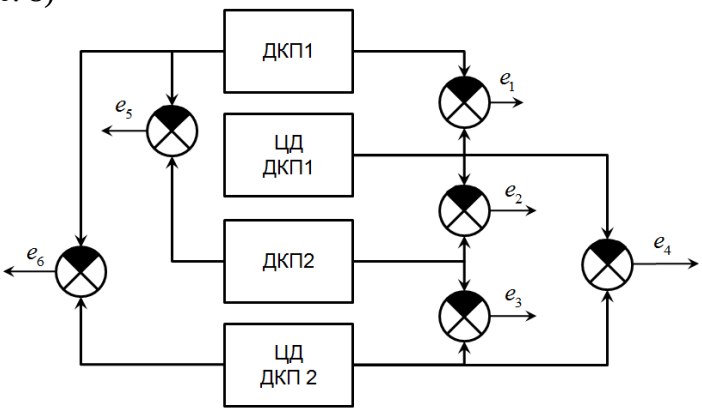


Рис.3. Схема вычислений предельных рассогласований датчиков контроля параметров рабочего процесса и их двойников.

На рис. 8 приняты следующие обозначения: ДКП 1 – первый датчик контроля параметров рабочего процесса; ДКП 2 – второй датчик контроля параметров рабочего процесса; ЦД ДКП1 – цифровой двойник первого датчика; ЦД ДКП2 – цифровой двойник второго датчика; e_1 – рассогласование между ДКП1 и ЦД ДКП1; e_2 – рассогласование между ДКП2и ЦД ДКП1; e_3 – рассогласование между ДКП2 и ЦД ДКП2; e_4 – рассогласование между

ЦД ДКП1 и ЦД ДКП2; e_5 – рассогласование между ДКП2 и ДКП1; e_6 – рассогласование между ДКП1 и ЦД ДКП2;

Полученные рассогласования поступают в нейросетевой анализатор решающий задачу анализа данных, а именно поиска аномалий среди потоков данных с цифровых двойников и датчиков контроля параметров рабочего процесса.

В случае отказа первого датчика контроля параметров рабочего процесса рассогласования e_1, e_5, e_6 выходя за пределы заданных значений и нейросетевой анализатор выдает сигнал об отказе ДКП1. В случае отказа второго датчика контроля параметров рабочего процесса за заданные пределы выходят рассогласования e_2, e_3, e_5 и нейросетевой анализатора выдает сигнал об отказе второго датчика контроля параметров рабочего процесса, при отказе обоих датчиков контроля параметров рабочего процесса за пределы рассогласования выходят параметры e_1, e_2, e_3, e_5, e_6. В данных случаях рассогласование e_5 указывает на отказ хотя бы одного из датчиков контроля параметров рабочего процесса, рассогласования e_1, e_6, e_2, e_5 локализуют отказ и указывают на конкретный датчик.

При отказе первого цифрового двойника за заданные пределы выходят рассогласования e_1 e_2 e_4 для второго соответственно e_3 e_6 e_4. Здесь на отказ хотя бы одного из цифровых двойников указывает рассогласование e_4, а рассогласования e_1 e_2 e_3 e_6 локализуют отказ и указывают конкретный отказавший цифровой двойник.

3. Экспериментальное исследование оценки работы датчика

В целях апробации методики был проведен полный факторный эксперимент для оценки эффективности работы разработанной методики обеспечения требуемого уровня надежности газотурбинного двигателя.

Для проведения эксперимента разработана экспериментальная установка имитирующая контур управления $\pi_T = f(T_в^*)$ (Рис. 4).

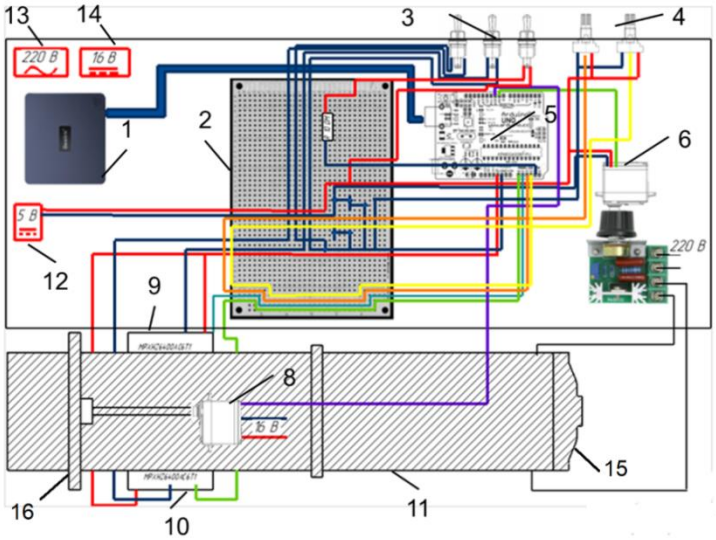


Рис. 4. Конструктивная схема экспериментальной установки

где 1- вычислитель; 2 – плата; 3 – управляющие переключатели; 4 – управляющие потенциометры; 5 – ммикроконтроллер; 6,8 – сервомеханизмы; 7 - регулятор мощности тока; 9,10 – датчики давления; 11- канал полунатурной модели ГТД; 12,13,14 – источники тока; 15 – ээлектроприводной компрессор; 16 – регулируемое сопло

Изначально были собраны материалы о работе полунатурной модели контура управления степени понижения давления в газовой турбине. На основе которых происходило обучение нейросетевых моделей датчиков и анализатора параметров рабочего процесса (Рис. 5).

Рис. 5. Результат обучения искусственных нейронных сетей

После обучения моделей датчиков и нейросетевого анализатора выполнялось оценка точности управления при исправных датчиках давления и при неисправных датчиках с подключением нейросетевых моделей.

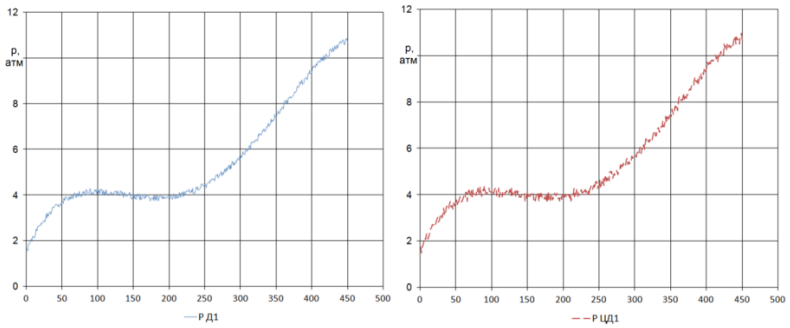
При проведении эксперимента сначала проверялась работа САУ со стабильно работающими датчиками, затем физически отключались клеммы питания от датчиков давления 9,10 при помощи управляющих переключателей 3. (Рис. 6)

Рис 6. переходный процесс в контуре управления $\pi_T = f(T_в^*)$

где 1-значение полного давления с отказавших датчиков; 2 – отклик нейросетевых моделей;
3 – заданное значение полного давления.

Для проведения оценки повышения уровня надежности системы, представленной в экспериментальной установке выполнен расчет вероятности безотказной работы полунатурной модели контура управления $\pi_T = const$.

Была проведена статистическая оценка точности работы математических моделей датчиков давления, при которой оценивалась проверка гипотезы статистического равенства выборок значений давления получаемых с датчика (Рис.7. а) и его нейросетевой математической модели (Рис. 7. б). Оценка производилась при помощи Т-критерия Стьюдента.



а) б)

Рис. 7. Значения давления, регистрируемые датчиком и вычисленные математической моделью.

где Р Д1 –математическое ожидание значение с датчика давления; Р ЦД1 – математическое значение вычисленное математической моделью

Каждая точка на графиках является математическим ожиданием результатов 384 измерений для обеспечения точности наблюдений в эксперименте. Такое число измерений обусловлено минимизацией ошибки измерений $\varepsilon = 0,05$ и обеспечением доверительной вероятности $P = 0,95$ на основе таблицы достаточно больших чисел. При сравнении выборок значение Т-критерия Стьюдента составило $t = 1,85$ что меньше критического значения $t_{\text{кр}} = 1,96$, а следовательно, с доверительной вероятностью $\alpha_T = 0,95$ можно считать что статистической значимости различий между выборками нет.

Заключение

Предложенный способ разработки интеллектуального датчика с цифровым двойником и возможностью самодиагностики по собственным параметрам позволяет минимизировать влияние надежности датчиков контроля параметров рабочего процесса на надежность объекта управления и его системы автоматического управления.

Разработанный способ позволяет повысить число каналов дублирования в два раза, обеспечить работу системы автоматического управления в условиях неопределенности диагностировать отказ датчика без использования опорных значений.

Литература

1. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. – М.: Вильямс, 2006. –1104 с.

2. Гмурман, В. Е. Теория вероятностей и математическая статистика :учебное пособие / В. Е. Гмурман. – М.: Высш. шк., 2000.

3. Мхитарян, В. С. Теория вероятностей и математическая статистика :учебник / В. С. Мхитарян, В. Ф. Шишов, А. Ю. Козлов. – М.: Академия, 2012.

4. Коновалов В.И. Идентификация и диагностика систем: учебное пособие; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2010. – 163 с.

УДК 004.6

СОСТАВЛЕНИЕ БАЗЫ ДАННЫХ ДЛЯ РЕЙТИНГОВОЙ СИСТЕМЫ УЧАЩИХСЯ

Н. В. Щербаков

Воронежский государственный университет

Введение

Рейтинговые системы для оценки учащихся являются важным инструментом в образовательном процессе, с помощью которого можно не только более объективно оценить полученные знания и навыки, а также личностные качества студента, но и значительно активизировать его работу и сделать ее более равномерной в течение всего семестра по всем дисциплинам. Они могут способствовать мотивации студентов к достижению лучших результатов, а также помогать преподавателям и администрации школы или университета выявлять потенциальные проблемы в учебном процессе и поощрять учащихся с высоким рейтингом, мотивируя остальных.

В данной статье описывается вариант базы данных для применения в рейтинговых системах образовательных учреждений.

1. Параметры оценивания

По каким параметрам оценивать учащихся? Первым и самым очевидным ответом на этот вопрос являются их оценки. Для объективности оценивания знаний мало знать его оценку на экзамене или зачете, не исключаются ситуации, когда студент мог получить оценку по дисциплине ниже или наоборот выше того, что заслуживает. Немалую роль в этом играют, с одной стороны, так называемый «сессионный стресс», а с другой, личностно-психологические особенности самого студента и преподавателя. В балльно-рейтинговой системе экзамен или зачет уже не является единственной возможностью дифференцированной оценки знаний студента за семестр. Он становится только еще одной формой контроля, результаты прохождения которого лишь добавляют определенное количество баллов к набранным студентом в течение семестра. Рейтинговая система должна учитывать все работы, выполняемые учащимся в течении семестра. Для обеспечения более равномерной работы студента в рейтинговой системе учитываются посещаемость и своевременная сдача проверочных работ. Это показывает насколько своевременно и успешно он этими знаниями овладевал в течение семестра. Также в рейтинговой системе немаловажным является участие в различных мероприятиях (доклады, рефераты, предметные олимпиады). Это позволяет увеличить активность учащихся. Для возможности отражать в рейтинге не только полученные знания и навыки, но и личностные качества студента можно использовать мнение преподавателей.

Рейтинговая система содержит много настраиваемых параметров. Небольшая часть из них устанавливается едиными для всего университета, а остальные параметры выбирает кафедра для своих дисциплин.

Для наиболее полной оценки деятельности учащихся будем учитывать следующее:

1. Оценки по предметам.
2. Посещение занятий.
3. Своевременная сдача работ.

4. Участие в мероприятиях.
5. Мнение преподавателей (опоздания, поведение, списывания и прочее)

2. Составление базы данных

Для данной задачи будем использовать самые распространенные и подходящие базы данных – реляционные. Исходя из вышесказанных параметров оценивания были сформированы следующие таблицы:

ATTENDANCE – таблица посещаемости. Поля:

ID: целочисленный, первичный ключ, идентификатор студента;
STUDENT_FIO: строка, ФИО студента;
LESSON_1: логический, посещение занятия 1;
LESSON_2: логический, посещение занятия 2;
…
LESSON_N: логический, посещение занятия N;

EVENTS – таблица мероприятий. Поля:

ID: целочисленный, первичный ключ, идентификатор студента;
STUDENT_FIO: строка, ФИО студента;
EVENT_1: целочисленный, участие в мероприятии 1;
EVENT_2: целочисленный, участие в мероприятии 2;
…
EVENT_N: целочисленный, участие в мероприятии N;

DELAY – таблица задержки сдачи предметных заданий. Поля:

ID: целочисленный, первичный ключ, идентификатор студента;
STUDENT_FIO: строка, ФИО студента;
WORK_1: целочисленный, время задержки сдачи работы 1;
WORK_2: целочисленный, время задержки сдачи работы 2;
…
WORK_N: целочисленный, время задержки сдачи работы N;

SUBJECT – таблица оценок по предмету. Поля:

ID: целочисленный, первичный ключ, идентификатор студента;
STUDENT_FIO: строка, ФИО студента;
WORK_1: целочисленный, оценка за выполнение работы 1;
WORK_2: целочисленный, оценка за выполнение работы 2;
…
WORK_N: целочисленный, оценка за выполнение работы N;
FINALWORK: целочисленный, оценка за выполнение итоговой работы (зачета/экзамена);

ACADEMIC_PERFORMANCE – таблица оценок по предметам. Поля:

ID: целочисленный, первичный ключ, идентификатор студента;
STUDENT_FIO: строка, ФИО студента;
SUBJECT_1: целочисленный, оценка за выполнение работы 1;
SUBJECT_2: целочисленный, оценка за выполнение работы 2;
…
SUBJECT_N: целочисленный, оценка за выполнение работы N;

TEACHER_RATING – таблица оценок студента преподавателями. Поля:

N: целочисленный, первичный ключ;
TEACHER: строка, ФИО преподавателя;
STUDENT: строка, ФИО студента;
BEHAVIOR: целочисленный, оценка поведения;
PUNCTUALITY: целочисленный, оценка пунктуальности;
RESPONSIBILITY: целочисленный, оценка ответственности;

Вывод итогового рейтинга производится в таблицу с полями:

N: целочисленный, первичный ключ, номер в рейтинге;
STUDENT_FIO: строка, ФИО студента;
RATING: целочисленный, итоговая оценка;

Вычисление итогового рейтинга производится с помощью задаваемых весов параметров, уникальных для каждого случая.

Заключение

В данной работе были изучены критерии оценивания студентов и создана база данных для рейтинговой системы. Составление базы данных является ключевым шагом для эффективного мониторинга и оценки успеваемости студентов. Правильно организованная база данных позволяет не только отслеживать академические достижения студентов, но и выявлять области для улучшения образовательного процесса. Грамотное использование данных из базы позволяет принимать обоснованные решения по повышению качества обучения и поддержке студентов в достижении своих образовательных целей.

Литература

1. Соболев С.К. Рейтинговая система оценки знаний: общие принципы и выбор параметров. Инженерный журнал: наука и инновации, 2014, вып. 1.
2. Зайцева Н.А. "Балльно-рейтинговая система: особенности и практика применения" Современные проблемы сервиса и туризма, №. 4, 2011, с. 98-105.
3. Ключевые аспекты при выборе базы данных для вашего приложения. – Режим доступа: https://habr.com/ru/companies/otus/articles/562852/ – (Дата обращения: 10.04.2024).

УДК 004.4

РАЗРАБОТКА ПРОГРАММНОЙ РЕАЛИЗАЦИИ МЕТОДА TOPSIS ДЛЯ ФОРМИРОВАНИЯ ОЦЕНКИ ПЕРСОНАЛА КОМАНИИ НА ОСНОВЕ ОБРАБОТКИ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

М.А. Япринцев, Ю.В. Бондаренко

Воронежский государственный университет

Введение

Ежедневно различные компании набирают новых сотрудников в штат. Для оценки их личностных и межличностных навыков рекрутеры с помощью большого количества тестов [1-2]. Одной из основных проблем, с которой сталкиваются менеджеры HR-отделов компании и рекрутинговых агентств, является обработка результатов тестирования и построения комплексной (интегральной) количественной оценки каждого кандидата, на основании которой возможно принять решение о выборе подходящего соискателя на вакантную должность. Одним из методов формирования комплексной оценки альтернатив и выбора лучшей из них, является метод TOPSIS [3], который может быть эффективно применен и в процессе подбора персонала компаний.

Целью настоящей работы является разработка программного обеспечения, позволяющего на основе метода TOPSIS сформировать комплексную оценку кандидатов на вакантную должность и принять обоснованное решение о выборе соискателя.

1. Материалы и метод

1.1. Постановка задачи

Предположим, что на вакантную должность в компании претендуют n кандидатов. Каждый из кандидатов прошел собеседование по тестам, по отобранным менеджерами HR-отдела или рекрутингового агентства. Известны частные оценки каждого кандидата по каждому из пройденных им тестов. Для формирования комплексной оценки кандидатов на основе частных оценок тестирования, воспользуемся методом TOPSIS.

Метод TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) - это метод многокритериального принятия решений, который помогает определить оптимальное решение с учетом нескольких критериев [1]. Состоит он из следующих шагов:

1. Формирование матрицы решений
2. Нормализация матрицы решений
3. Определение весов критериев
4. Применение весов к нормализованной матрице
5. Нахождение идеальных решений
6. Расчет расстояния от идеальных решений
7. Нахождение ближайшего решения к положительному идеалу и самого далекого отрицательного идеала

Данный метод реализован на языке Python версии 3.12, в среде разработки PyCharm 2023.1.3. Вначале нужно перенести таблицу excel в нашу среду разработки, в этом нам поможет библиотека pandas с помощью команды pd.read_excel(«Название файла»). Далее рассмотрим остальные шаги поэтапно.

1.2. Нормализация таблицы результатов

Необходимо нормализовать таблицу результатов так, чтобы значения матрицы были в пределе от 0 до 1. Это делается путем деления каждого значения в матрице на сумму квадратов значений в каждом столбце и извлечения квадратного корня из этой суммы.

Для нормализации будем использовать функцию Normalize, которая в качестве аргумента принимает исходную таблицу и количество тестов. Реализация метода изображена на рисунке 1.

```
def Normalize(dataset, nCol):
    for i in range(1, nCol):
        temp = 0
        # Вычисляем корень из суммы квадратов значений в определенном столбце
        for j in range(len(dataset)):
            temp = temp + dataset.iloc[j, i]**2
        temp = temp**0.5
        # Нормализуем элемент
        for j in range(len(dataset)):
            dataset.iat[j, i] = (dataset.iloc[j, i] / temp)
    print(dataset)
```

Рис. 1. Функция для нормализации исходной матрицы

1.3. Определение весов критериев.

Зададим веса каждого критерия, отражающие их относительную важность. Умножим каждое значение в нормализованной решающей матрице на соответствующий вес критерия. Для данной работы используем функцию MultiplyByWeights, который в качестве аргументов принимает нормализованную матрицу, количество тестов и массив весов. Реализация метода, описанного выше, изображена на рисунке 2.

```
def MultiplyByWeights(dataset, nCol, weights):
    for i in range(1, nCol):
        # Умножаем каждый элемент в столбце на соответствующий вес
        for j in range(len(dataset)):
            dataset.iat[j, i] *= weights[i-1]
    print(dataset)
```

Рис. 2. Функция для умножения весов на критерии

1.4. Нахождение идеальных решений.

Определим идеальное решение, которое является максимальным в каждом столбце. Так же сделаем с негативно идеальным решением, только в этом случае ищем минимальные значения.

Для этого используем функцию Calc_Values, принимающую на вход матрицу, количество столбцов и переменную impact, которая может быть равна «+» или «-». В первом случае ищем идеально положительное, во втором противоположное ему. Реализация метода, описанного выше, изображена на рисунке 3.

```
def Calc_Values(dataset, nCol, impact):
    ideal_best = (dataset.max().values)[1:]
    ideal_worst = (dataset.min().values)[1:]
    for i in range(1, nCol):
        if impact[i-1] == '-':
            ideal_best[i-1], ideal_worst[i-1] = ideal_worst[i-1], ideal_best[i-1]
    return ideal_best, ideal_worst
```

Рис. 3. Функция для нахождения идеальных решений

1.5. Расчет расстояния от идеальных решений.

На этом этапе для каждой альтернативы вычислим расстояние до положительного и негативного идеалов для каждого критерия, используя формулу Евклидова расстояния: $d_i^+ = \sqrt{\sum_{j=1}^{m}(x_{ij} - P_j)^2}$, где d_i^+ – расстояние от альтернативы i до положительно идеала, x_{ij} – значение альтернатив i по критерию j, P_j – значение положительного идеала по критерию j, m – количество критериев. С негативно идеальным решением поступаем аналогично

1.6. Получение итогового результата.

На заключительным этапе нам необходимо найти ближайшие альтернативы к положительному идеалу и наиболее удаленной альтернативы от отрицательного идеала. Для этого воспользуемся формулой: $\frac{d_i^-}{(d_i^+ + d_i^-)}$, где негативно идеальное решение делим на позитивное в сумме с негативным. Полученный результат для каждой строки сравниваем, наибольший результат и будет для нас являться лучшим вариантом сотрудника на нанимаемую должность. Алгоритмы, описанные выше изображены на рисунке 4.

```
# Расчет положительных и отрицательных значений
p_values, n_values = Calc_Values(temp_dataset, nCol, impact)
# Расчет значения TOPSIS
result = []  # Результат TOPSIS
pp = []  # Положительное расстояние
nn = []  # Отрицательное расстояние
# Расчет расстояний и значения TOPSIS для каждой строки
for i in range(len(temp_dataset)):
    temp_p, temp_n = 0, 0
    for j in range(1, nCol):
        temp_p += (p_values[j-1] - temp_dataset.iloc[i, j])**2
        temp_n += (n_values[j-1] - temp_dataset.iloc[i, j])**2
    temp_p, temp_n = temp_p**0.5, temp_n**0.5
    результат.append(temp_n/(temp_p + temp_n))
    nn.append(temp_n)
    pp.append(temp_p)
# Добавление новых столбцов в набор данных
dataset['Положительное расстояние'] = pp
dataset['Отрицательное расстояние'] = nn
dataset['Результат TOPSIS'] = result
# Расчет ранга согласно результату TOPSIS
dataset['Ранг'] = dataset['Результат TOPSIS'].rank(method='max', ascending=False)
dataset = dataset.astype({"Ранг": int})
```

Рис. 4. Алгоритмы *для получения итогового результата*

2. Пример работы программы

2.1. Подготовка данных к исследованию

Для примера возьмем трех кандидатов, которые прошли четыре теста, для определения уровня мотивации, стрессоустойчивости, памяти, способность творческого мышления и получили.

В результате прохождения кандидатами тестирований нам была предоставлена таблица с различными числовыми представлениями, которая будет использована для дальнейшей работы и определения наилучшего кандидата. Таблица представлена на рисунке 5.

Кандидаты	Тест А	Тест Б	Тест В	Тест Г
Кандидат 1	8	7	800	8/10
Кандидат 2	7	8	700	4/10
Кандидат 3	9	8	600	6/10

Рис. 5. Данная таблица результатов для исследования

2.2. Нормализация таблицы результатов

На данном этапе наша матрица будет приведена к нормальному виду, чтобы все значения были от 0 до 1. Используем функцию Normalize. В результате в консоли получим таблицу, которая изображена на рисунке 6.

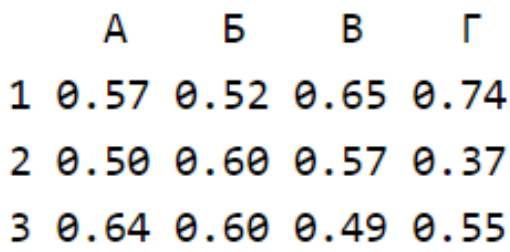


Рис. 6. Нормализованная матрица

2.3. Определение весов критериев

Пусть для нас тест на уровень мотивации самый важный, тогда определим ему вес 1, остальные 0.4, 0.5, 0.3, соответственно. Мы будем использовать их для умножения на каждую ячейку в матрице. После данных операций в консоль будет выведена таблица, которая изображена на рисунке 7.

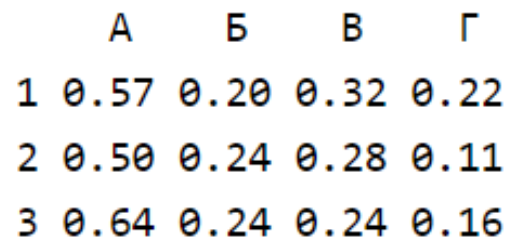


Рис. 7. Нормализованная матрица с учетом весов

2.4. Нахождение идеальных решений.

Для поиска идеальных решений найдем максимальное и минимальное значение из каждого столбца в нормализованной матрице (рис. 7). После выполнения данной процедуры программа выводит в консоль следующие значения, изображенные на рисунке 8.

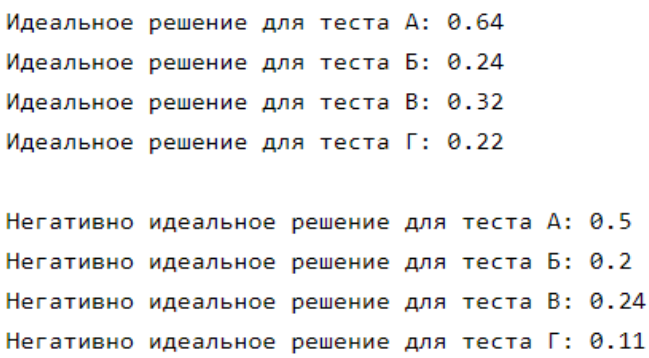


Рис. 8. Идеальные решения

2.5. Расчет расстояния от идеальных решений

С помощью формулы Евклидова расстояния, рассчитаем расстояния от идеальных решений для каждого кандидата. Программа выведет в консоль следующую матрицу, изображенную на рисунке 9.

	А	Б	В	Г	+	-
1	0.57	0.20	0.32	0.22	0.06	0.07
2	0.50	0.24	0.28	0.11	0.10	0.02
3	0.64	0.24	0.24	0.16	0.07	0.07

Рис. 9. Расстояния от идеальных решений

В данной таблице появились два новых столбца «+» и «-». В первом указаны расстояния от положительного идеального решения, во втором от отрицательного.

2.6. Получение итогового результата

При итоговом расчете мы сможем увидеть результаты нашего исследования. В консоль будет выведена конечная таблица, изображенная на рисунке 10.

	А	Б	В	Г	+	-	Result
1	0.57	0.20	0.32	0.22	0.06	0.07	0.67
2	0.50	0.24	0.28	0.11	0.10	0.02	0.50
3	0.64	0.24	0.24	0.16	0.07	0.07	0.72

Подходящий сотрудник на должность - Кандидат 3

Рис. 10. Итог работы

В итоговом результате у нас добавился столбец «Result», в котором указан итоговый рейтинг кандидата. Далее программа находит максимальное значение из них и выводит итоговое решение в виде строки ниже.

Заключение

В ходе проведенного исследования было разработано программное обеспечение на языке Python, обеспечивающее поддержку обработку результатов тестирования кандидатов на вакантную должность с целью выбора наиболее подходящей кандидатуры. Программное обеспечение носит универсальный характер и может быть использовано как инструмент поддержки принятия решений в любой компании.

Литература

1. Батыршев А. В. Тестирование. Основной инструментарий практического психолога: учебное пособие / А. В. Батыршев. – 3-е изд., перераб. и доп. – Москва : Дело, 2003. – 240 с.
2. В. Йеттер. Эффективный отбор персонала. Метод структурированного интервью – Х.: Изд-во «Гуманитарный Центр» – 2011. – 360 с.
3. Джабраилова З.Г. Моделирование процесса выбора кандидатов на вакантные должности с применением нечеткой логики / З.Г. Джабраилова, С.Р. Нобари // Искусственный интеллект: сб. статей. Баку, 2009. – С. 254-259.

Содержание

Научное издание

Математика,
информационные технологии,
приложения

Сборник трудов
Межвузовской научной конференции
молодых ученых и студентов

Воронеж,
24-25 апреля 2024 г.

Подписано в печать 11.06.2024. Формат 60 × 84 / 8.
Усл. печ. л. 33,95. Тираж 100 экз. Заказ № ***.

Отпечатано с готового оригинал-макета

ООО Издательско-полиграфический центр
«Научная книга»
394030, г. Воронеж, ул. Никитинская, д. 38, оф. 308
Тел. +7 (473) 200-81-02, 200-81-04
http://www.n-kniga.ru; e-mail: zakaz@n-kniga.ru

Отпечатано в типографии ООО ИПЦ «Научная книга». 394026, г. Воронеж, Московский пр-т, 11б
Тел. +7 (473) 220-57-15, 238-02-38 http://www.n-kniga.ru; e-mail: typ@n-kniga.ru