

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ
РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ В.Ф. УТКИНА

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ПРИКЛАДНЫХ ИССЛЕДОВАНИЯХ

Межвузовский сборник научных трудов

Рязань 2025

УДК 004

Информационные технологии в прикладных исследованиях, межвузовский сборник научных трудов. – Рязань: ИП Коняхин А.В. (Book Jet), 2025. – 370 с.

ISBN 978-5-908007-16-0

Публикуются статьи о проблемах использования информационных технологий в науке, промышленности и образовании.

Сборник рассчитан на научных сотрудников, преподавателей, аспирантов и студентов высших учебных заведений.

Авторская позиция, орфография, пунктуация и стилистические особенности публикаций сохранены.

Редакционная коллегия:

д-р техн. наук, проф. В.П. Корячко (ответственный редактор), д-р техн. наук, проф. В.А. Минаев (МУ МВД России им. В.Я. Кикотя), д-р техн. наук, проф. Ю.Л. Леохин (Московский технический университета связи и информатики), д-р техн. наук, проф. О.Я. Кравец (Воронежский государственный технический университет), д-р техн. наук, проф. Д.О. Жуков (МИРЭА – Российский технологический университет), д-р техн. наук, проф. С.В. Скворцов (РГРТУ), д-р техн. наук, проф. Д.А. Перепелкин (РГРТУ), канд. техн. наук, доц. А.Н. Сапрыкин (ответственный секретарь).

Рецензенты:

д-р техн. наук, проф. А.Н. Пылькин (Рязанский государственный радиотехнический университет имени В.Ф. Уткина), д-р техн. наук, проф. А.П. Карпенко (Московский государственный технический университета имени Н.Э. Баумана).

ISBN 978-5-908007-16-0

©Рязанский государственный
радиотехнический университет
имени В.Ф. Уткина, 2025
©Коллектив авторов, 2025
©ИП Коняхин А.В. (Book Jet), 2025

СОДЕРЖАНИЕ**Аббасов К.Н., Елесина С.И.**

Сравнительный анализ метода деформируемого многогранника и нейросетевого поиска в задаче корреляционного совмещения изображений.....9

Агеев А.А.

Разработка программного модуля сбора отчётности для автоматизированной информационно-аналитической системы на платформе ODANT..... 13

Акинин А.С.

Аналитические системы на базе 1С:Предприятие и Power BI: применение ODATA и стандартных механизмов отчетности 17

Балоян А.А.

Интеграция API Spotify и Яндекс.Музыки в разработке музыкальных приложений 22

Башкин Ф.И.

Преобразования речевых запросов в текст с использованием обученных моделей нейросетей..... 26

Благодаров Е.А.

Разработка программного модуля передача видеопотока между приложениями..... 29

Богомолов А.С.

Внедрение ERP-систем в малом и среднем бизнесе: возможности и ограничения..... 33

Борзых Д.А.

Проектирование программы сбора и обработки информации систем мониторинга инженерных конструкций..... 39

Бубликов Д.И., Маркин А.В.

Разработка компилятора SQL 43

Васина М.А.

Нейрокомпьютеры, нейроинтерфейсы, нейрочипы и их применение.....58

Венчиков Д.А.

Создание алгоритма для поиска места встречи попутчика и водителя. .65

Гармаш М.Е.

Проектирование информационной системы с микросервисной архитектурой71

Графкина К.В.

Распознавание номерных знаков транспортных средств79

Грошев В.В.

Информационная система для интернет-магазина техники с клиент-серверной архитектурой.....85

Дорин А.А.

Метод роя частиц.....93

Доркин Д.А.

Использование технологии сверточных нейронных сетей в сегментации объектов изображения99

Елатников Н.В., Скворцов С.В.

Проектирование приложения для отслеживания динамических объектов на неподвижном фоне104

Ениватов А.С.

Интеграция интеллектуального ассистента GigaChat в образовательную платформу108

Еськин М.А.

Использование локальных языковых моделей в инженерной среде с применением технологии RAG111

Зайцев Е.С.

Использование нейронных сетей LSTM для прогнозирования временных рядов.....115

Замешаев Д.В., Скворцов С.В.

Выбор хеш-функции для решения задачи балансировки нагрузки в сети.....118

Исайкин А.В.

Искусственные нейронные сети122

Касельский М.П.

Обучение нейронных сетей129

Каширина А.В.

Автоматизированное А/В-тестирование с использованием машинного обучения136

Коротких И.А.

Использование микросервисной архитектуры при создании информационных систем приема и обработки платежей за услуги ЖКХ143

Коротких И.А.

Сравнительный анализ монолитной и микросервисной архитектур ..147

Костин В.Ю., Скворцов С.В.

Исследование эффективности алгоритмов решения NP-полных задач в САПР152

Костылев И.В.

Психоакустические методы оценки качества сжатой речи. Алгоритм PESQ157

Кошелев А.Д., Кошелева М.С.

Взаимосвязь цифровых двойников и киберфизических систем в контексте умного производства161

Крыгина М.К., Кошелева М.С.

Объекты моделирования цифровых двойников.....167

Кузнецов Н.А., Скворцов С.В.

Анализ влияния объема обучающих данных на коэффициент сжатия алгоритма Хаффмана170

Курганова И.И.

Анализ роевых алгоритмов: методы, применение и тенденции.....173

Куркина М.С.

Разработка программного модуля автоматизации формирования и учета платежных транзакций по использованию парковочного пространства181

Маркин А.С.

Сравнительный анализ функций активации при обучении линейного классификатора.....187

Панина А.И.

Туманные и граничные вычисления – анализ преимуществ и перспектив192

Перепелкин А.А.

Мультиагентные системы200

Перепелкин Д.А., Ковердяев А.И.

Динамическое управление потоками данных в мультипровайдерных сетях на основе генетического алгоритма208

Перепелкин Д.А., Никонов К.А.

Применение графовых нейронных сетей для решения задачи маршрутизации в программно-конфигурируемых сетях215

Петренко А.Д.

Влияние развития искусственного интеллекта на жизнь человека.....220

Петричук Д.В.

Разработка системы подбора кандидатов для реализации стартапов в области разработки ПО225

Петрова Е.С.

Разработка пользовательского интерфейса и интерактивной игры на Java229

Печенин И.В.

Использование распределенных вычислительных платформ Spark/Hadoop для машинного обучения237

Половинкин А.В.

Методы идентификации и их применение в библиотечной среде242

Половинкин В.В.

Применение алгоритмов и методов нечеткого сопоставления для очистки неструктурированных массивов данных в различных системах анализа и обработки BIG DATA246

Попова А.С.

Система персонализированных рекомендаций для выбора туристических маршрутов на основе искусственного интеллекта251

Почтарев А.В.

Исследование методов повышения качества и производительности при обработке изображений с глобальным освещением.....259

Прохин А.И.

Трансформер – архитектура нейронных сетей265

Сапрыкин А.Н., Арутюнова А.В.

Методы и алгоритмы преобразования 3D-моделей в управляющие коды для принтеров269

Сапрыкин А.Н.

Применение метода индексирования баз данных для снижения задержек в высоконагруженных информационных системах273

Сенякин К.Д., Тихонов А.А.

Мониторинг систем с помощью Python и Zabbix277

Сидоров А.С.

Сравнительный анализ цветовых моделей в области веб-дизайна282

Соломонов Н.Д.

Использование GPU в обучении искусственного интеллекта293

Сыркин Г.Е.

Особенности разработки серверной части интернет-магазина302

Сысоев М.Э.

Сравнительный анализ HTTP-фреймворков GIN, FIBER и ECHO для высоконагруженных web-приложений311

Тихонов А.А.

Разработка чат-бота Telegram для интернет-магазина314

Тобратов Ю.М., Кошелева М.С.

Обобщенное представление информационной системы обработки заказов с элементами искусственного интеллекта322

Халдеева М.А.

Особенности разработки клиентской части интернет – магазина.....328

Цховребов Р.А.

Анализ современных подходов к оценке характеристик программного обеспечения информационных систем336

Шмаков А.Ю.

Методы Few-Shot Learning для распознавания объектов при ограниченных обучающих данных342

Шошников С.В.

Автоматизация разработки интернет-магазинов с помощью искусственного интеллекта.....345

Щевьев О.И.

Автоматическая транскрипция музыки на основе нейронных сетей и CQT350

Щукарев А.С.

Архитектуры адаптивной системы для работы с потоковой информацией356

Юркова В.И., Маркин А.В.

Разработка системы анализа платежной дисциплины абонентов предприятий ЖКХ361

УДК 004.932

АББАСОВ К.Н., ЕЛЕСИНА С.И.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДА ДЕФОРМИРУЕМОГО
МНОГОГРАННИКА И НЕЙРОСЕТЕВОГО ПОИСКА В ЗАДАЧЕ
КОРРЕЛЯЦИОННОГО СОВМЕЩЕНИЯ ИЗОБРАЖЕНИЙ**

Рассматриваются методы поиска изображения, включая метод деформируемого многогранника (МДМ) и поиск при помощи сверточных нейронных сетей. Анализируются их временные характеристики и эффективность при последовательной и параллельной реализациях с использованием OpenMP.

Совмещение изображений является одной из главных задач в области обработки данных, находящей применение в самых разных отраслях науки и техники. Задача заключается в поиске максимума двумерной функции ковариации текущего и эталонного изображений. Совмещение изображений широко используется в медицинской визуализации, дистанционном зондировании Земли, компьютерном зрении, робототехнике и системах навигации.

В навигации особое внимание уделяется корреляционно-экстремальным методам совмещения, которые легли в основу работы корреляционно-экстремальных навигационных систем (КЭНС) [1]. КЭНС используют эталонные изображения (ЭИ) местности (например, аэрофотоснимки или цифровые карты местности), сравнивая их с текущими изображениями (ТИ), полученными бортовыми камерами или сенсорами. Задача совмещения позволяет определить параметры движения объекта и скорректировать его положение в пространстве.

Для совмещения изображений используется большое количество разнообразных методов, в данной работе исследуется метод локальной оптимизации – метод деформируемого многогранника (МДМ) [2, 3]. Поскольку этот метод определяет только один ближайший экстремум в ограниченной области поиска, а функция в результате совмещения получается многоэкстремальной (рис. 1) [1], то метод деформируемого многогранника необходимо применять несколько раз из различных точек в области поиска и выбрать наилучший результат. Для этого выбран метод глобальной оптимизации – мултистарт [1].

В качестве функционала сравнения была выбрана нормированная корреляционная функция (1) [1]. Она принимает значения в диапазоне от 0 до 1. Значение показывает степень сходства

между двумя изображениями или их фрагментами, где 1 указывает на идеальное совпадение, а 0 – на отсутствие корреляции.

$$F(x, y) = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f_{ЭИ}(i, j) \cdot f_{ТИ}(i + x, j + y)}{\sqrt{(\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f_{ЭИ}^2(i, j)) \cdot (\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f_{ТИ}^2(i + x, j + y))}} \quad (1)$$

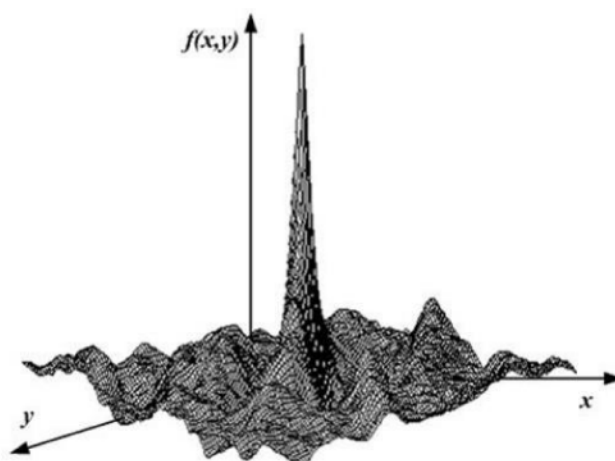


Рисунок 1 – Характерный вид нормированной корреляционной функции

Сверточные нейросети (CNN) [4] являются специализированным типом искусственных нейронных сетей, предназначенным для эффективной обработки данных с двумерной структурой, таких как изображения. Основное отличие сверточных нейросетей от полносвязных заключается в том, что они используют операцию свертки, позволяющую извлекать локальные признаки с помощью фильтров, обучающихся во время тренировочного процесса. Эти фильтры скользят по изображению, выявляя характерные особенности, такие как края, углы и текстуры, на различных уровнях абстракции. За счет того, что одни и те же фильтры применяются ко всей входной области, сверточные нейросети обладают высокой вычислительной эффективностью и устойчивостью к смещению объектов внутри изображения.

Типичная архитектура сверточной нейросети (рис. 2) включает в себя последовательные слои свертки, активации и подвыборки (пулинга), которые постепенно понижают пространственное разрешение изображения, одновременно усиливая его семантическое представление. После серии сверточных блоков информация

передается в полносвязные слои, которые интерпретируют извлеченные признаки и производят окончательные предсказания. Благодаря этому CNN могут успешно решать широкий круг задач компьютерного зрения, включая классификацию изображений, детекцию объектов, сегментацию и поиск шаблонов.

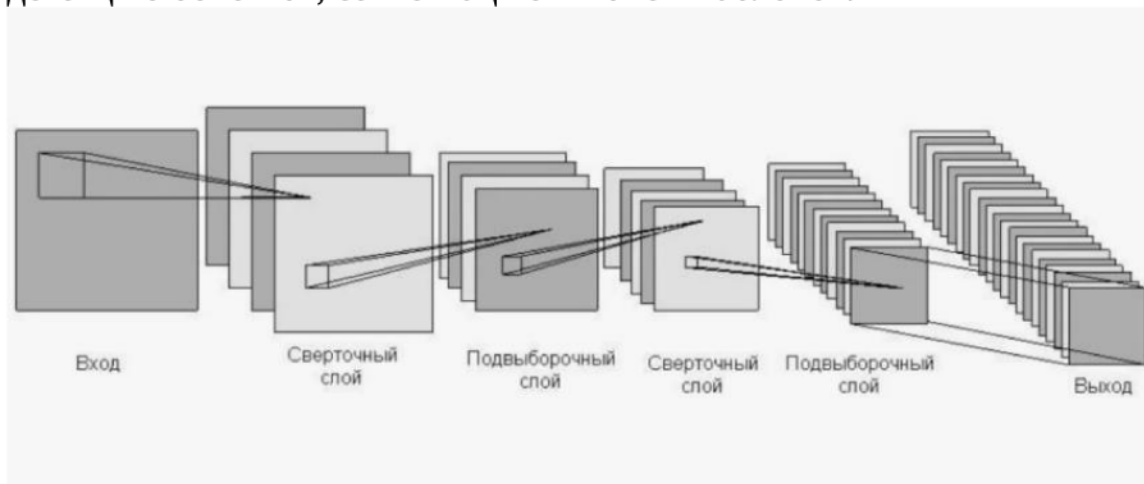


Рисунок 2 – Архитектура сверточных нейронных сетей

Для сравнения используется готовое решение из библиотеки `torch.nn`, являющейся частью фреймворка PyTorch. Она предоставляет удобный и гибкий инструментарий для построения и обучения сверточных нейросетей. В `torch.nn` реализованы все базовые компоненты CNN: слои свертки с настраиваемыми параметрами ядра, функции активации операции подвыборки, нормализация и регуляризация. Кроме того, библиотека поддерживает автоматическое дифференцирование и оптимизацию, что упрощает процесс обучения модели.

Был выполнен ряд экспериментальных запусков процесса совмещения изображений с разными значениями параметров. Эксперименты разбиваются на серии в зависимости от размера ЭИ (33%, 50%, 66% от линейных размеров ТИ). Размер ТИ 1200x700 пикселей. ЭИ имеют разные размеров: 900x500, 700x400 и 500x300 пикселей.

Исследования проводились на вычислительной системе: процессор – Intel Core i5-10210U, установленная память 8 ГБ. Для реализации МДМ был использован язык программирования C++, реализация параллельных вычислений производилась при помощи технологии OpenMP. Для нейросетевого поиска использовался язык программирования Python.

Результаты проведения экспериментальных запусков приведены в таблице 1.

Таблица 1 – Результаты экспериментальных запусков

Размер ЭИ	МДМ (последовательный)	МДМ (параллельный)	Нейросетевой поиск
900×500 (66%)	116.992 с.	33.476 с.	3.1669 с.
700×400 (50%)	61.07 с.	13.119 с.	3.738 с.
500×300 (33%)	34.363 с.	8.259 с.	3.0974 с.

На основе полученных экспериментальных данных был построен график (рис. 3), наглядно демонстрирующий зависимость времени выполнения поиска изображения от размера эталонного изображения для каждого из рассмотренных методов.

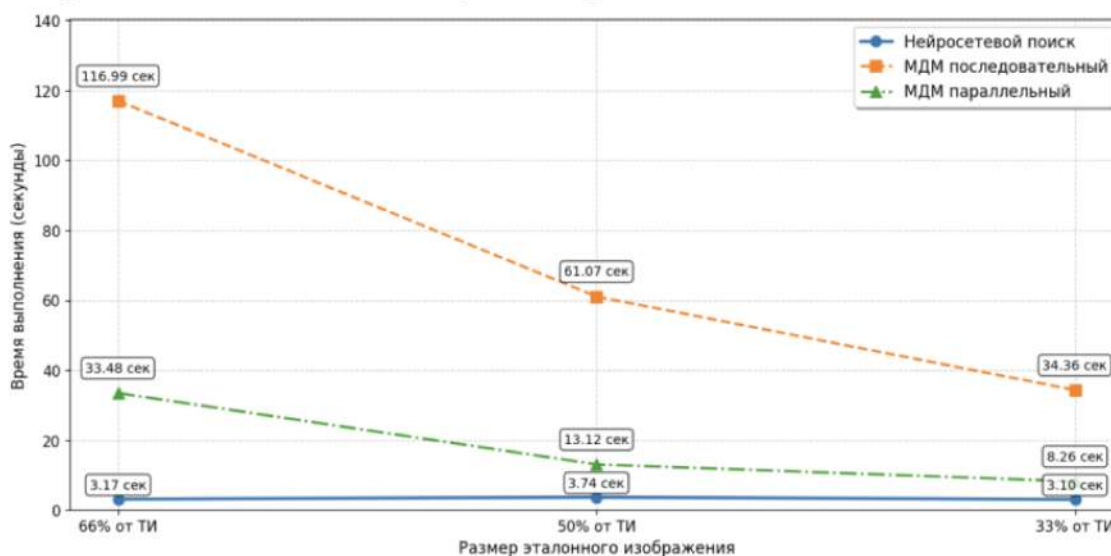


Рисунок 3 – Зависимость времени поиска от используемого метода

Исследования показали, что использование нейросетевого метода для поиска изображения является наиболее эффективным решением по сравнению с методом деформируемого многогранника. В среднем нейросетевой поиск выполняется более чем в 10 раз быстрее, чем параллельная реализация МДМ, и более чем в 30 раз быстрее, чем его последовательная версия. За счет распараллеливания вычислений с использованием OpenMP метод деформируемого многогранника ускорился в среднем в 4 раза, однако даже после оптимизации он значительно уступает нейросетевому подходу по времени выполнения.

Нейросетевой метод, реализованный с помощью библиотеки torch.nn, показал не только высокую скорость работы, но и стабильность результатов при различных входных данных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Елесина С.И., Костяшкин Л.Н, Логинов А. А, Никифоров М.Б. Совмещение изображений в корреляционно-экстремальных навигационных системах: монография. – М.: Изд-во Радиотехника, 2015. – 208 с.
2. Пантелеев А.В. Методы оптимизации в примерах и задачах: учеб. пособие/ А.В. Пантелеев, Т.А. Летова. – М.: Высш. шк., 2002. – 544 с.
3. Елесина С.И., Логинов А.А. Модифицированный метод деформируемого многогранника // Вестник РГРТУ. – 2009. – №4 (выпуск 30). – С. 100-103.
4. Омеляненко, Я. Эволюционные нейросети на языке Python: практическое руководство / Я. Омеляненко; пер. с англ. В.С. Яценкова. – Москва: ДМК Пресс, 2020. – 310 с.

УДК 004.65

АГЕЕВ А.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

**РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ СБОРА
ОТЧЁТНОСТИ ДЛЯ АВТОМАТИЗИРОВАННОЙ
ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ НА
ПЛАТФОРМЕ ODANT**

В статье рассматривается перспектива разработки информационного решения на базе отечественной платформы для органов государственной власти и муниципалитетов.

В условиях политики импортозамещения и санкционных ограничений российский рынок информационных технологий сталкивается с необходимостью разработки и внедрения отечественных решений. Одним из примеров таких систем является система сбора отчетности, созданная на базе отечественной технологической платформы ODANT. Эта платформа предлагает инновационные решения для организации, хранения и обработки данных, удовлетворяя требованиям современных российских компаний и ведомств.

Ключевым преимуществом платформы для разработки системы является древовидная структура данных, где объекты могут иметь неограниченное количество уровней и полей, что позволяет гибко

описывать иерархии данных. Благодаря этому подходу, система может моделировать сложные взаимосвязи и структурировать данные в виде вложенных массивов или таблиц, что особенно полезно при работе с крупными объемами информации.

Платформа ODANT также предоставляет возможности наследования классов и объединения объектов в иерархические структуры. Это обеспечивает не только простоту в администрировании и развитии системы, но и высокий уровень интеграции данных между различными подразделениями и организациями. Домены, в которых хранятся объекты, можно рассматривать как отдельные базы данных, каждая из которых обладает своей структурой и системой безопасности. Это позволяет легко изолировать данные различных уровней и настраивать доступ к ним в зависимости от иерархии [1].

Домены платформы бывают различных типов – от разделов до сервисов и компонентов. Системные домены обеспечивают доступ к классам с системными данными и функционалом, что упрощает управление доступом и данными на уровне всей организации. Гибкость системы позволяет пользователям с административными правами централизованно настраивать справочники, которые затем используются для формирования отчетов и управления данными. Более того, справочники могут быть как общими для всех пользователей системы, так и частными для отдельных ведомств, что позволяет эффективно организовывать работу на всех уровнях организации.

Административный интерфейс платформы обеспечивает полное управление системой через клиентское приложение, что позволяет максимально упростить настройку ролевой модели, структуры данных, иерархии отчетных форм и прав доступа. Это делает платформу ODANT оптимальным выбором для построения информационной системы сбора отчетности в условиях современных требований импортозамещения и безопасности данных [3].

Информационная система сбора отчетности строится по классической клиент-серверной модели с учетом особенностей технологии ODANT. Объекты в системе могут иметь сложную древовидную структуру с неограниченной вложенностью и множеством полей на каждом уровне, включая табличные данные в виде массивов. Их структура определяется классами, которые одновременно служат типами данных, хранилищами и функциональными модулями, способными взаимодействовать между собой. Объекты поддерживают ссылки, наследование и объединение в иерархии [1].

Система организует данные через домены – изолированные логические контейнеры, которые группируют классы и обеспечивают безопасность. Домены бывают трех типов: разделы, сервисы и компоненты, причем системные домены содержат служебные классы, такие как модули безопасности. Каждый элемент ODANT – хост, домен, класс или объект – имеет уникальный 15-значный идентификатор (*_id*), а полный путь в иерархии определяется составным *_fullid* [3].

Конфигурация пользователей формируется динамически на основе их прав доступа. Хосты представляют собой физические серверы с собственными адресами (IP или URL) и могут объединяться в бизнес-сети, предоставляя друг другу данные и структуры. Хранение информации осуществляется в бинарном формате ODB, похожем на XML, а для работы с данными используется язык запросов XQuery. Взаимодействие с системой происходит через REST API с глобальной адресацией, позволяющей обращаться к ресурсам любого узла сети.

В ходе анализа ролевой модели системы были определены три ключевые роли, обеспечивающие многоуровневый контроль доступа к данным.

Администратор системы выполняет централизованное конфигурирование классов отчетности, справочников и агрегационных форм, а также управляет распределением прав доступа между пользователями. Он обладает полным доступом ко всем компонентам системы и может конфигурировать рабочие места для отдельных пользователей или групп.

Заказчик данных, представляющий министерство или головную организацию, координирует процесс сбора информации от подведомственных структур – открывает отчетные периоды, утверждает или отклоняет представленные данные, формирует сводные отчеты на основе первичных сведений. Он работает в рамках главного домена и всех связанных с ним поддоменов, что обеспечивает контроль за всей иерархией отчетности.

Оператор ввода первичных данных, являющийся представителем муниципалитета или дочерней организации, отвечает за внесение исходных данных, их заверение электронной подписью и передачу на проверку заказчику. Он ограничен одним или несколькими поддоменами, что гарантирует четкое разграничение зон ответственности при работе с первичной информацией.

Такой подход обеспечивает не только эффективное распределение функционала между участниками процесса, но и

безопасное взаимодействие с данными на разных уровнях организационной структуры.

Реализация данной модели основана на гибких механизмах платформы ODANT, позволяющих настраивать рабочие места пользователей через систему доменов и поддоменов.

Перспективность разработки данной системы заключается в кардинальном сокращении сроков сбора отчетности ведомств с нескольких рабочих дней до нескольких часов. Столь значительное ускорение процессов достигается за счет комплексной автоматизации всех этапов работы с данными. Система обеспечивает сбор информации за счет шаблона, может производить автоматическую проверку ее корректности и исключает необходимость ручного переноса данных между различными системами.

Ключевым преимуществом является использование современных технологий обработки данных, таких как XPath и XQuery, для работы с информацией, хранящейся в XML-формате [4]. Эти языки запросов позволяют эффективно выполнять сложные выборки данных, осуществлять их многоуровневую агрегацию и автоматически формировать итоговые отчеты. В результате значительно сокращается время на обработку информации и минимизируется вероятность ошибок, неизбежных при ручном вводе. Примеры хранения данных и использования языка запросов с выводом результата представлены на рисунках 1-3.

```
<?xml version="1.0"?>
<PACK>
  <OBJECT Поселение="Караваевское" Остаток="25" Дата="2025-03-03T09:25:06+03:00"/>
  <OBJECT Поселение="Щедринское" Остаток="50" Дата="2025-03-03T09:25:40+03:00"/>
  <OBJECT Поселение="Муромское" Остаток="75" Дата="2025-03-03T09:26:24+03:00"/>
  <OBJECT Поселение="Васильевское" Остаток="100" Дата="2025-03-03T09:27:05+03:00"/>
  <OBJECT Поселение="Олеговское" Остаток="56" Дата="2025-03-10T14:53:15+03:00"/>
  <OBJECT Поселение="Черненькое" Остаток="666" Дата="2025-03-10T14:53:53+03:00"/>
</PACK>
```

Рисунок 1 – Представление модели хранения данных в системе

```
element PACK {
  for $a in PACK/OBJECT
  group by oda:left($a/@date, 6)
  return element OBJECT {
    attribute КоличествоПоселений {count($a/@Посел)},
    attribute СуммарныйОстаток {sum($a/@Остат)}
  }
}
```

Рисунок 2 – Запрос на вывод суммы остатков и количества поселений из объектов с группировкой по году и месяцу при помощи языка Xquery


```
<?xml version="1.0"?>
  <РАСК>
    <ОБЪЕКТ КоличествоПоселений="6" СуммарныйОстаток="972"/>
  </РАСК>
```

Рисунок 3 – Вывод результата запроса

Также преимуществом является возможный обмен данными с другими информационными системами по API в формате XML [3], что позволит данному решению интегрироваться со многими другими информационными системами.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Перепелкин Р.А., Чумаков Г.В., Витухин В.В., Марочкин М.В., Трофимов И.М. ODANT – интернет нового поколения // Проектирование будущего. Проблемы цифровой реальности: труды 2-й Международной конференции (7-8 февраля 2019 г., Москва). – М.: ИПМ им. М.В.Келдыша, 2019. – С. 112-117. – URL: <https://keldysh.ru/future/2019/11.pdf> (дата обращения 01.04.2025).
2. Одиноккина С.В. Основы технологий XML: учебное пособие. – СПб: НИУ ИТМО, 2013. – 56 с.
3. Технологическая платформа ODANT Platform [Электронный ресурс] – URL: <https://odant.ru/> (дата обращения: 02.04.2025).
4. Extensible Markup Language (XML) W3C: официальный сайт. [Электронный ресурс] – URL: <https://www.w3.org/XML/> (дата обращения: 03.04.2025).

УДК 004.4+004.9

АКИНИН А.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

АНАЛИТИЧЕСКИЕ СИСТЕМЫ НА БАЗЕ 1С:ПРЕДПРИЯТИЕ И POWER BI: ПРИМЕНЕНИЕ ODATA И СТАНДАРТНЫХ МЕХАНИЗМОВ ОТЧЕТНОСТИ

В статье рассматриваются подходы к построению аналитических систем на основе интеграции учетной платформы 1С:Предприятие и аналитического инструмента Power BI.

Информационные технологии играют важнейшую роль в поддержке управленческих решений на всех уровнях бизнеса.

Особенно это актуально в условиях цифровой трансформации, когда предприятия стремятся автоматизировать не только учет, но и аналитику. Распространенной ситуацией является использование 1С:Предприятие как учетной платформы и Power BI как системы бизнес-аналитики. Несмотря на высокую популярность обоих инструментов, их интеграция зачастую требует дополнительной настройки.

Настоящая статья направлена на исследование архитектурных подходов к интеграции этих платформ без привлечения промежуточного ПО, с использованием исключительно стандартных средств 1С:Предприятие – механизмов отчетности и публикации данных через OData [2]. В центре внимания – универсальность, простота внедрения и применимость к задачам малого и среднего бизнеса.

Роль аналитических систем в цифровом бизнесе

Современное предприятие, вне зависимости от масштаба, функционирует в условиях постоянного потока информации [5]. Эффективное использование этих данных становится ключевым фактором успешности. В условиях цифровизации возрастает потребность в системах, обеспечивающих не просто учёт и хранение информации, но и её оперативный анализ с возможностью визуализации, прогнозирования и поддержки управленческих решений.

В последние годы получили широкое распространение платформы класса Business Intelligence (BI), позволяющие настраивать гибкие отчёты, интерактивные панели мониторинга и автоматизированные системы контроля показателей. Одной из самых востребованных BI-платформ является Power BI, разработанная компанией Microsoft. В сочетании с учетными системами, такими как 1С:Предприятие, она становится мощным инструментом для построения корпоративной аналитики.

Однако эффективность такой интеграции напрямую зависит от используемой архитектуры и методов обмена данными [3]. В данной работе акцент сделан на использовании стандартных отчетных механизмов 1С и технологии OData, встроенной в платформу 1С, как на наиболее сбалансированном и универсальном решении для построения аналитической системы.

Особенности отчетных механизмов 1С:Предприятие

Платформа 1С:Предприятие традиционно ориентирована на учетно-операционные задачи — ведение складского, бухгалтерского, кадрового и управленческого учета. При этом в ней присутствуют

встроенные средства создания отчетности, которые можно разделить на три группы:

1. Стандартные отчеты (универсальные) – реализованы типовыми средствами и позволяют пользователю без программирования сформировать табличный или графический отчет.

2. Настраиваемые отчеты – создаются разработчиком конфигурации, адаптируются под нужды конкретной организации, используют запросы к внутренним данным.

3. Регистры сведений – накапливают агрегированные или предобработанные данные для использования в аналитических целях.

Эти средства являются внутренними по отношению к платформе 1С и предоставляют значительные возможности по структурированию и предварительной обработке данных. Однако они ограничены в части визуализации, а также не обеспечивают полноценную многомерную аналитику, к которой пользователи привыкли в BI-системах.

Power BI как платформа бизнес-аналитики

Power BI предоставляет пользователю [1]:

- подключение к множеству источников данных (в т.ч. REST, SQL, Excel, XML, JSON, OData и др.);
- трансформацию и очистку данных (Power Query);
- построение визуализаций (графики, диаграммы, KPI-индикаторы, карты и др.);
- использование DAX-выражений для вычислений;
- встроенные функции прогнозирования и машинного обучения;
- автоматическое обновление данных по расписанию.

Ключевым достоинством Power BI является интуитивный интерфейс, возможность создания интерактивных дашбордов и высокая масштабируемость. Однако для эффективной работы Power BI требуется, чтобы данные поступали в «плоском» и агрегированном виде, предпочтительно через формализованный веб-протокол.

Использование OData для передачи данных из 1С в Power BI

OData (Open Data Protocol) – это REST-подобный протокол, разработанный Microsoft, предназначенный для стандартизованного доступа к данным через HTTP. Платформа 1С:Предприятие поддерживает публикацию объектов метаданных в виде OData-источников [4], включая регистры сведений, справочники, документы, таблицы значений и др.

Главные преимущества использования OData в связке 1С + Power BI:

- Стандартизированный формат – не требуется промежуточный сервер или дополнительная база данных.

- Прямая интеграция – Power BI нативно поддерживает подключение к OData-фиду.
- Минимальная настройка – публикация OData в 1С требует лишь базовых знаний конфигуратора.
- Поддержка авторизации и безопасности – можно использовать HTTP Basic, Windows Authentication и HTTPS.
- Автоматизация – Power BI поддерживает обновление OData-источников по расписанию.

Таким образом, использование OData превращает платформу 1С в полноценный источник данных для BI-системы, обеспечивая прозрачный и безопасный канал передачи информации.

Логика построения аналитической системы на базе OData и стандартных механизмов 1С

Предложенный подход строится на идее предобработки данных в 1С с последующим их потреблением Power BI по протоколу OData. Это позволяет снять нагрузку по агрегации и фильтрации с Power BI, перенести ее на внутренние ресурсы 1С и, тем самым, оптимизировать производительность.

Схема алгоритма работы аналитической системы:

1. В 1С создается запрос или отчет, формирующий нужный срез данных (например, продажи по дням, остатки на складах и т.д.).
2. Результаты запроса записываются в регистр сведений, структура которого соответствует аналитической задаче.
3. Регистр публикуется как OData-ресурс в интерфейсе веб-службы 1С.
4. Power BI подключается к опубликованному OData-фиду, извлекает данные и строит модель данных.
5. Создаются визуализации, дашборды и отчеты, которые могут обновляться по расписанию через Power BI Gateway.

Такой подход обеспечивает разделение логики: 1С отвечает за получение и структурирование данных, Power BI – за визуализацию и анализ.

Схема архитектуры:

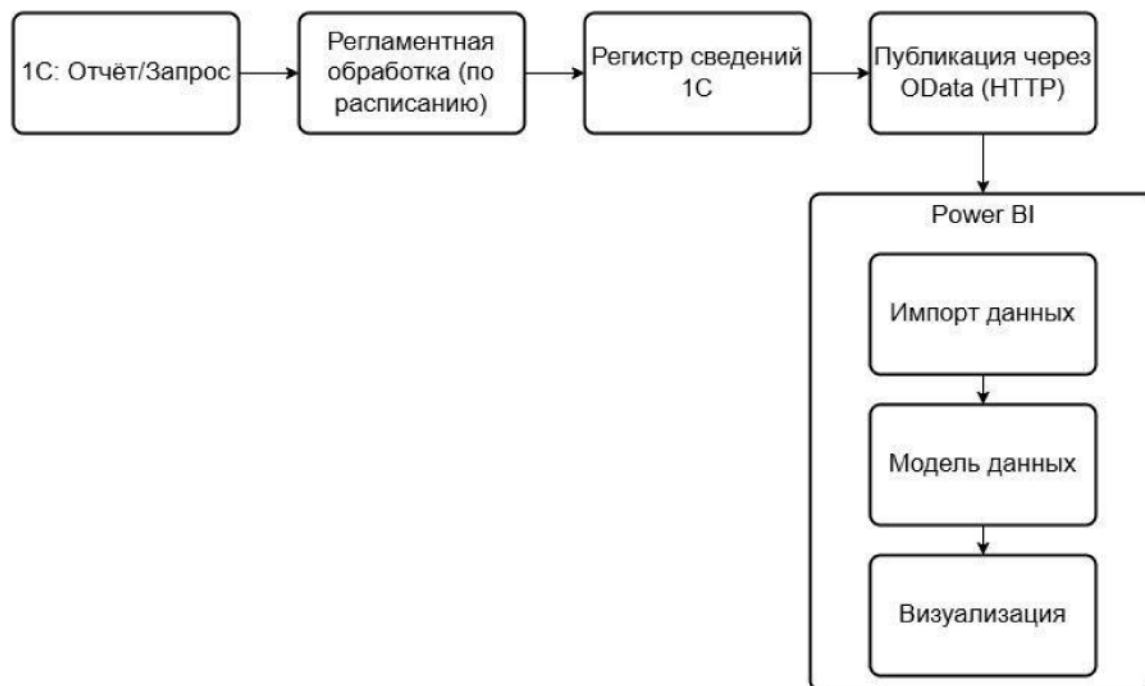


Рисунок 1 – Архитектурная схема интеграции 1С и Power BI

Когда и почему необходимо использовать связку OData и стандартных отчетных механизмов

Использование описанной архитектуры оправдано в следующих случаях:

- Для организаций, не желающих использовать сторонние ETL-средства (например, Talend, Apache Nifi, SSIS).
- Когда важна поддержка обновлений конфигурации 1С без риска потери данных – решение строится на стандартных механизмах и не требует доработки ядра.
- Если пользователи уже используют Power BI – отсутствует необходимость изучения новых инструментов.
- Когда необходимо сократить затраты на разработку отчетов и внедрение аналитики.
- Для оперативного запуска аналитической системы – архитектура позволяет развернуть рабочую систему в течение нескольких дней.

В отличие от более сложных схем, предполагающих выгрузку данных в промежуточные хранилища (например, SQL-сервер), связка OData + стандартные механизмы 1С минимизирует инфраструктурные и организационные издержки.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Microsoft Power BI documentation (Платформа 1С:Предприятие Инструкция разработчика. 1С) [Электронный ресурс]. – URL: <https://learn.microsoft.com/power-bi> (дата обращения: 01.04.2025).
2. Гончаров С.А. Использование OData-сервисов 1С для интеграции с Power BI // Информационные технологии. – 2023. – №4. – С. 63–70.
3. Романов А.Г. Информационно-аналитические системы управления предприятием: учебник. – М.: ИНФРА-М, 2021. – 300 с.
4. Иванов И.Ю. Интеграция 1С: Предприятие с Power BI: учеб. пособие / И.Ю. Иванов. – М.: Питер, 2022. – 320 с.
5. Гринченко Н.Н. Проектирование информационных систем: учебное пособие / Н.Н. Гринченко, А.Ю. Громов, Н.И. Хизриева. – М.: КУРС, 2023. – 176 с.

УДК004.514.6

БАЛОЯН А.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИНТЕГРАЦИЯ API SPOTIFY И ЯНДЕКС.МУЗЫКИ В РАЗРАБОТКЕ МУЗЫКАЛЬНЫХ ПРИЛОЖЕНИЙ

Рассматриваются методы интеграции API Spotify и Яндекс.Музыки для создания кроссплатформенных музыкальных приложений. Анализируются этапы аутентификации, работы с плейлистами, потоковой передачи данных и оптимизации производительности.

Музыкальные платформы

Современные музыкальные приложения требуют интеграции с популярными стриминговыми сервисами для доступа к обширным библиотекам треков, персонализации рекомендаций и синхронизации плейлистов. API Spotify и Яндекс.Музыки предоставляют разработчикам инструменты для реализации этих функций. В статье рассматриваются ключевые аспекты интеграции, включая аутентификацию, управление данными и оптимизацию производительности.

Аутентификация и авторизация

Spotify использует протокол OAuth 2.0, требующий регистрации приложения на платформе разработчика. После получения `client_id` и `client_secret` пользователь перенаправляется на страницу авторизации,

где предоставляет разрешения. Токен доступа, выдаваемый после успешной аутентификации, позволяет выполнять запросы к API, таким как получение истории прослушиваний или доступ к плейлистам.

Пример реализации на Python:

```
from spotipy import SpotifyOAuth
import spotipy
# Настройка OAuth
auth_manager = SpotifyOAuth(
    client_id="BAШ_CLIENT_ID",
    client_secret="BAШ_CLIENT_SECRET",
    redirect_uri="http://localhost:8080/callback",
    scope="user-library-read playlist-modify-private"
)
# Создание клиента
sp = spotipy.Spotify(auth_manager=auth_manager)
# Получение данных пользователя
user_info = sp.current_user()
```

Яндекс.Музыка также применяет OAuth 2.0, но с особенностями: токен запрашивается через код авторизации, который передается вручную или через веб-интерфейс. Для работы с API необходимо зарегистрировать приложение в Яндекс.OAuth и настроить права доступа (например, чтение библиотеки или управление плейлистами).

Пример реализации на Python:

```
import requests
# Получение токена
auth_url="https://oauth.yandex.ru/authorize?response_type=code&client_id=BAШ_ID"
code = input("Введите код из URL: ")
token_response = requests.post(
    "https://oauth.yandex.ru/token",
    data={
        "grant_type": "authorization_code",
        "code": code,
        "client_id": "BAШ_ID",
        "client_secret": "BAШ_SECRET"
    }
)
access_token = token_response.json()["access_token"]
```

Для наглядности на Рис. 1 представлена схема взаимодействия приложения с API Spotify и Яндекс.Музыки.

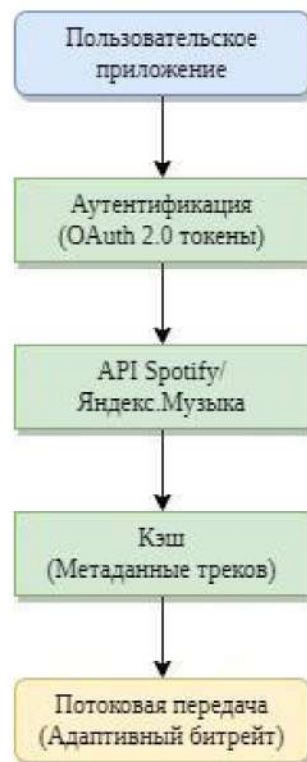


Рисунок 1 – Архитектура взаимодействия приложения с API

Работа с метаданными

Оба сервиса предоставляют доступ к метаданным треков, включая название, исполнителя, альбом, жанр и длительность.

Spotify позволяет получать расширенные данные: битрейт, дату релиза, популярность трека и связи с другими артистами. Например, запрос к эндпоинту `/tracks/{id}` возвращает информацию о треке в формате JSON.

Яндекс.Музыка фокусируется на базовых метаданных, но поддерживает поиск по ключевым словам и фильтрацию по жанрам.

Таблица 1 – Сравнение возможностей API

Параметр	Spotify	Яндекс.Музыка
Глубина метаданных	Высокая (битрейт, дата релиза)	Базовая (название, артист)
Поиск по жанрам	Через категории и теги	Ограниченный функционал
Рекомендации	На основе seed-треков/артистов	На основе истории прослушиваний

Генерация рекомендаций

Spotify предоставляет алгоритмы рекомендаций, основанные на машинном обучении. Система анализирует предпочтения

пользователя, историю прослушиваний и схожие треки. Для генерации рекомендаций используются параметры:

- *seed_tracks* – опорные треки;
- *target_** – целевые характеристики (например, танцевальность или энергетика).

Яндекс.Музыка использует упрощенный подход, предлагая треки на основе популярности и истории прослушиваний. Интеграция рекомендаций требует частых запросов к API, что может привести к превышению лимитов.

Потоковая передача данных

Для минимизации задержек при воспроизведении используются:

1. *Буферизация*: предзагрузка 5-10 секунд аудио.
2. *Кэширование*: локальное хранение часто воспроизводимых треков.
3. *Адаптивный битрейт*: выбор качества в зависимости от скорости сети.

Таблица 2 – Тестирование задержек

Сервис	Средняя задержка (мс)
Spotify	180
Яндекс.Музыка	210

Оптимизация производительности

1. *Пакетные запросы*: объединение запросов к API для снижения нагрузки.
2. *Ограничение частоты запросов*: соблюдение лимитов API (Spotify: 100 запросов/мин, Яндекс: 50 запросов/мин).
3. *Использование WebSocket*: для реального обновления данных (например, синхронизация плейлистов).

Заключение

Интеграция API Spotify и Яндекс.Музыки позволяет создавать мощные музыкальные приложения с доступом к миллионам треков. Ключевые рекомендации:

- Использовать OAuth 2.0 для безопасной аутентификации.
- Оптимизировать запросы для соблюдения лимитов API.
- Реализовать адаптивную потоковую передачу для улучшения пользовательского опыта.

Перспективным направлением является внедрение машинного обучения для персонализации рекомендаций на основе данных обоих сервисов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Spotify Web API Documentation [Электронный ресурс] – URL: <https://developer.spotify.com/documentation/web-api/> (дата обращения: 02.03.2025).
2. Документация API Яндекс.Музыки [Электронный ресурс] – URL: <https://yandex.ru/dev/music/> (дата обращения: 02.03.2025).
3. Макконнелл С. Совершенный код/ Пер. с англ. В.Г. Вшивцева. – М.: Русская редакция, 2023. – 896 с.

УДК 681.3

БАШКИН Ф.И.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

**ПРЕОБРАЗОВАНИЯ РЕЧЕВЫХ ЗАПРОСОВ В ТЕКСТ С
ИСПОЛЬЗОВАНИЕМ ОБУЧЕННЫХ МОДЕЛЕЙ НЕЙРОСЕТЕЙ**

Рассматривается проблема преобразования речевых запросов в текст с использованием обученных моделей нейросетей из библиотек VOSK и Kaldi.

В последние годы наблюдается значительный рост интереса к технологиям распознавания речи, что обусловлено широким распространением голосовых интерфейсов и виртуальных помощников в повседневной жизни. Преобразование речевых запросов в текст является ключевым этапом в разработке систем, способных эффективно взаимодействовать с пользователями на естественном языке. Современные подходы к решению этой задачи все чаще основываются на методах машинного обучения и нейросетевых архитектурах, которые демонстрируют высокую точность и устойчивость к различным условиям записи звука.

Среди множества доступных инструментов для распознавания речи выделяются библиотеки VOSK и Kaldi [1 2], которые предоставляют мощные средства для разработки и внедрения систем автоматического распознавания речи (ASR).

Актуальность данной работы заключается в необходимости повышения качества распознавания речевых запросов в условиях разнообразия акцентов, фонового шума и различных языковых особенностей. В рамках данной статьи будет рассмотрен процесс преобразования речевых запросов в текст с использованием обученных моделей нейросетей из библиотек VOSK и Kaldi. Основная задача исследования заключается в сравнительном анализе

эффективности этих библиотек при решении задачи распознавания речи, а также в выявлении их сильных и слабых сторон. стремящихся улучшить качество взаимодействия человека с машиной через голосовые интерфейсы [1, 2].

В данной статье рассматривается метод преобразования речевых запросов в текст с использованием двух популярных библиотек для распознавания речи: VOSK и Kaldi. Оба инструмента основаны на современных нейросетевых архитектурах и предлагают различные возможности, что позволяет эффективно решать задачу распознавания речи в условиях разнообразия акцентов, фонового шума и различных языковых особенностей.

1. Библиотека VOSK

VOSK – это легковесная библиотека для автоматического распознавания речи, которая поддерживает множество языков и может работать в оффлайн-режиме. Основные возможности VOSK включают:

- **Оффлайн-распознавание:** VOSK позволяет выполнять распознавание речи без необходимости постоянного подключения к интернету, что делает его идеальным для мобильных и встроенных приложений.
- **Поддержка многоязычности:** Библиотека предоставляет предобученные модели для различных языков, что упрощает интеграцию в многоязычные системы.
- **Простота интеграции:** VOSK имеет интуитивно понятный API, что позволяет разработчикам быстро внедрять функциональность распознавания речи в свои приложения без необходимости глубокого изучения теории обработки сигналов.

Процесс работы с VOSK включает запись аудиозапроса, его предварительную обработку (например, нормализацию громкости) и последующее преобразование аудиосигнала в текст с использованием предобученной модели. Это позволяет быстро получать текстовые транскрипции речевых запросов пользователей.

2. Библиотека Kaldi

Kaldi – это более мощный инструмент для распознавания речи, который предоставляет широкие возможности для настройки и оптимизации моделей. Он ориентирован на исследователей и разработчиков, которые нуждаются в гибкости при работе с аудиоданными. Основные возможности Kaldi включают:

- **Гибкость настройки моделей:** Kaldi позволяет пользователям разрабатывать собственные модели с использованием различных

архитектур нейронных сетей (например, LSTM или CNN) и применять передовые методы обработки сигналов.

- Поддержка различных форматов данных: Библиотека может работать с различными форматами аудиофайлов и предоставляет инструменты для предварительной обработки данных, такие как фильтрация шума и нормализация громкости.

- Обширная документация и сообщество: Kaldi имеет активное сообщество пользователей и разработчиков, а также обширную документацию, что облегчает процесс обучения и внедрения.

Алгоритм работы с Kaldi включает несколько этапов: сбор данных, предварительная обработка аудиосигналов, настройка модели на основе специфических требований задачи (например, использование пользовательских словарей или акцентов) и выполнение распознавания речи. Это позволяет достичь высокой точности распознавания даже в сложных условиях.

Предлагаемый подход в реализации программы будет заключаться в использовании предобученной модели библиотеки VOSK для извлечения речевых данных, после чего распознавание полученных речевых данных будет осуществляться с применением инструментов и алгоритмов, предоставляемых библиотекой Kaldi. Данный метод позволит эффективно комбинировать преимущества обеих библиотек: VOSK обеспечит высокую скорость и простоту извлечения речевых данных, в то время как Kaldi предоставит возможности для более глубокого анализа и точного распознавания речи, что в свою очередь повысит общую эффективность системы.

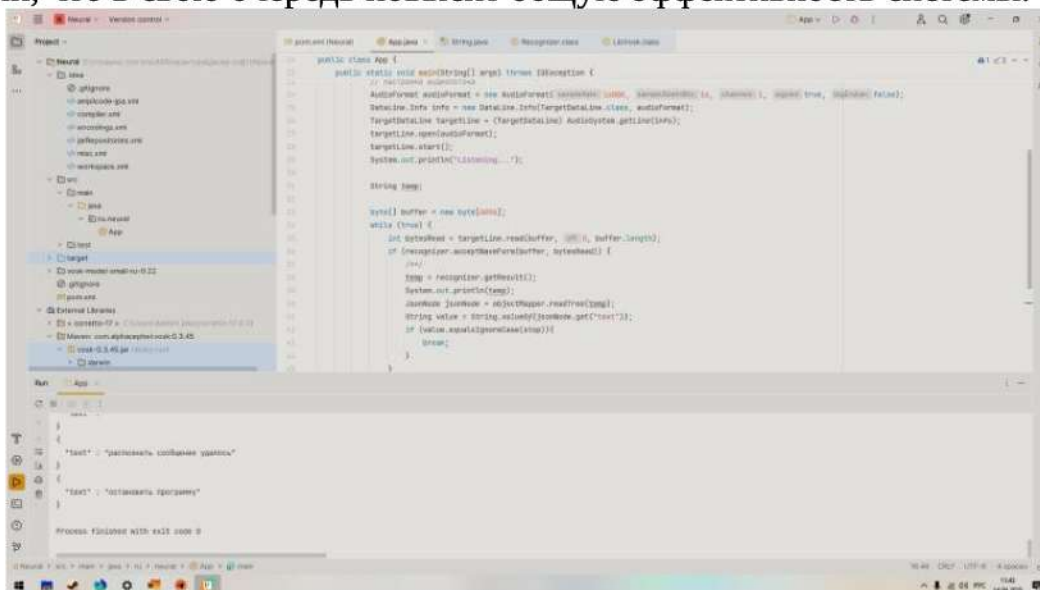


Рисунок 1 – Результат выполнения программы

Полученные результаты – программный модуль, реализующий функцию распознавания речи. Также с помощью команды из распознанной речи возможно остановить программу.

В рамках настоящего исследования был разработан программный модуль, обладающий функциональностью распознавания речи и способный завершать выполнение программы на основе голосовых команд. Данный модуль является составной частью разрабатываемого программного комплекса "Цифровой учитель", предназначенного для проведения уроков и предоставления ответов на вопросы учащихся в процессе обучения. В перспективе планируется интеграция данного модуля с локальной нейросетью, способной обрабатывать и отвечать на входящие речевые запросы, что позволит значительно расширить функциональные возможности системы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. VOSK [Электронный ресурс]. – URL: <https://alphacephei.com/vosk/> (дата обращения: 15.03.2025).
2. Kaldi. The Free Encyclopedia [Электронный ресурс]. – URL: <https://en.wikipedia.org/wiki/Kaldi> (дата обращения: 15.03.2025)

УДК 004.65

БЛАГОДАРОВ Е.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ПЕРЕДАЧА ВИДЕОПОТОКА МЕЖДУ ПРИЛОЖЕНИЯМИ

В данной статье рассматривается алгоритм обмена данными о видеопотоке между модулями системы контроля натекания расплава.

Введение

Разработка программного обеспечения для контроля точки натекания струи расплава в процессе производства минеральной ваты является критически важной задачей для оптимизации всего технологического процесса. Автоматизация этого контроля и управления позволяет значительно повысить качество производимой минеральной ваты, увеличить эффективность производства и, как следствие, улучшить ее эксплуатационные характеристики при одновременном снижении производственных затрат. Это достигается за счет точного регулирования параметров струи расплава, что

исключает отклонения от заданных технологических норм и минимизирует брак.

Система контроля разделена на два модуля обработки видеопотока и модуля контроля натекания расплава. Модуль обработки видеопотока основан на обработке видеопотока, получаемого с камеры, направленной на струю расплава. Для передачи видеоизображения используется протокол RTSP (Real Time Streaming Protocol) с кодеком H.264, обеспечивающий передачу видеоданных в реальном времени с высоким качеством сжатия. Выбор RTSP обусловлен его широкими возможностями по управлению потоком данных, что позволяет эффективно контролировать передачу видеоинформации от камеры к системе обработки. Протокол RTSP не отвечает за непосредственно передачу видеоданных, а управляет потоком, обеспечивая команды запуска, остановки и управления параметрами потока.

В данной системе, видеопоток, передаваемый по протоколу RTSP, принимается приложением, написанным на языке C++ с использованием кроссплатформенной библиотеки Qt. Qt предоставляет обширный набор инструментов и библиотек для работы с RTSP-потоками, что упрощает разработку и обеспечивает надежную работу приложения. После получения видеоданных приложение на C++ передает их в приложение, написанное на языке C#, через механизм разделяемой памяти. Использование разделяемой памяти обеспечивает высокопроизводительную передачу данных между двумя приложениями, работающими на разных языках программирования. Приложение C# отвечает за дальнейшую обработку видеопотока, включая анализ изображения для определения параметров струи расплава, и визуализацию данных для пользователя, предоставляя ему удобный интерфейс для мониторинга и управления процессом. Такой подход позволяет эффективно разделять задачи между приложениями, оптимизируя работу системы в целом.

Алгоритм обмена данными между модулями системы

Пересылка видеопотока между приложением Qt C++ и C# через разделяемую память – сложная задача, требующая внимательного подхода к синхронизации и управлению памятью. Ниже представлен общий план и ключевые моменты реализации:

1. Выбор механизма разделяемой памяти: наиболее распространенный способ: mmap (POSIX) или Create File Mapping/MapViewOfFile (Windows). Это позволяет приложениям на C++ и C# получить доступ к одному и тому же участку памяти. Выбор зависит от целевой платформы.

2. Структура разделяемой памяти: разделяемая память должна иметь четко определенную структуру, понятную как C++-приложению, так и C#-приложению. Она должна включать:

Заголовок: информация о размере видеокадра, формате (например, размер, количество каналов, глубина цвета), timestamp, флаги (например, флаг готовности кадра). Использование структуры данных (struct/class) с четко определенными полями критически важно для корректной работы. Убедитесь, что порядок байтов в структурах одинаков на обеих платформах (используйте `#pragma pack` в C++ или соответствующие атрибуты в C#).

Данные видеокадра: непосредственно пиксельные данные видеокадра. Размер этого блока будет зависеть от размера кадра и формата.

3. Производитель (Qt C++): захват видеопотока: используйте подходящую библиотеку для захвата видео (например, OpenCV, QCamera).

Форматирование видеокадра: преобразуйте видеокадр в формат, понятный обоим приложениям (например, RGB или YUV).

Запись в разделяемую память: заблокируйте доступ к разделяемой памяти (mutex или семафор), запишите заголовок и пиксельные данные, установите флаг готовности кадра и разблокируйте память.

Управление буферизацией: для повышения производительности используйте несколько буферов в разделяемой памяти, позволяя производителю заполнять один буфер, пока потребитель обрабатывает другой. Это потребует механизма круговой буферизации.

4. Потребитель (C#): чтение из разделяемой памяти: ожидание сигнала о готовности кадра (например, с помощью события или опроса флага).

Чтение заголовка и данных: после получения сигнала, чтение заголовка для получения информации о кадре и последующее чтение пиксельных данных.

Обработка видеокадра: отображение кадра на экране или выполнение других действий.

Освобождение памяти: после обработки кадра, сообщить производителю, что буфер свободен.

5. Синхронизация: мьютексы/Семафоры: для предотвращения гонок данных и обеспечения безопасного доступа к разделяемой памяти. Мьютексы (mutex) обеспечивают взаимное исключение, а семафоры позволяют управлять доступом нескольких потоков/процессов.

События (Events): для сигнализации о готовности кадра и освобождении буфера. В C# можно использовать ManualResetEvent или AutoResetEvent. В C++ можно использовать аналогичные механизмы (например, std::condition_variable).

6. Обработка ошибок: обработка исключений: на обеих платформах необходимо предусмотреть обработку возможных ошибок, таких как нехватка памяти, ошибки при доступе к разделяемой памяти и т.д.

Фрагменты кода

Реализация разделяемой памяти в приложении QtC++:

```
// ... создание разделяемой памяти ...
// Запись кадра
mutex.lock();
header.timestamp = getCurrentTime();
header.frameSize = frame.size();
memcpy(sharedMemory + sizeof(Header), frame.data(),
header.frameSize);
header.ready = true;
mutex.unlock();
// ... сигнализировать C# о готовности кадра ...
Реализация разделяемой памяти в приложении C#:
// ... создание разделяемой памяти ...
// Чтение кадра
while (true) {
    if (header.ready) {
        header
        Marshal.PtrToStructure<Header>(sharedMemory);
        byte[] frameData = new byte[header.frameSize];
        Marshal.Copy((IntPtr)((long)sharedMemory
sizeof(Header)), frameData, 0, header.frameSize);
        // ... обработка frameData ...
        header.ready = false; // Освобождаем буфер
    }
    // ... ожидание события ...
}
```

Заключение

Пересылка видеопотока между приложениями Qt и C# через разделяемую память – эффективный способ передачи данных с высокой производительностью. Однако, необходимо учитывать вопросы синхронизации и управления памятью, а также обработки потенциальных ошибок. Правильная реализация требует внимательного подхода и тщательного тестирования. Данный пример является упрощенной иллюстрацией и может потребовать модификаций в зависимости от конкретных требований и

используемых библиотек. Не забудьте установить размер `imageSizeInBytes` соответствующий размеру видеокадра. Также, нужно добавить механизмы обработки ошибок и более robust систему синхронизации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Горяйнов К.Э., Горяйнова С.К. Технология теплоизоляционных материалов и изделий. – М.: Стройиздат, 1982. – 376 с.
2. Джон Смит, Мэри Джонс RTSP: Протокол потоковой передачи в реальном времени// Технологии будущего. – 2023. – 450 с.

УДК 658.5:004.4

БОГОМОЛОВ А.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ВНЕДРЕНИЕ ERP-СИСТЕМ В МАЛОМ И СРЕДНЕМ БИЗНЕСЕ: ВОЗМОЖНОСТИ И ОГРАНИЧЕНИЯ

В статье рассматриваются особенности внедрения ERP-систем в малом и среднем бизнесе. Проанализированы их преимущества, а также определены основные ограничения и сложности внедрения.

ERP-системы (Enterprise Resource Planning) привычно считаются атрибутом крупного бизнеса, где прямая интеграция всех бизнес-процессов в единую информационную среду приносит ощутимый финансовый результат. Однако в последнее время возрастающий интерес к средствам ERP можно наблюдать у малого и среднего бизнеса (МСБ). Подобное явление можно объяснить ростом числа облачных технологий, усилением конкуренции и ужесточением конкурентной борьбы на рынке, а также с цифровизацией, которая помогает компаниям выживать в условиях быстро меняющейся экономической среды [1], [2].

Обзор ERP-систем. ERP-система (Enterprise Resource Planning) – это программное обеспечение, предназначенное для комплексной автоматизации управления предприятием. Она объединяет различные бизнес-процессы и службы компании в единую информационную среду, обеспечивая целостность и актуальность данных во всех отделах [1].

ERP-система представляет из себя сложную структуру, с множеством функциональных модулей, каждый из которых управляет отдельной сферой деятельности. В бухгалтерском и финансовом модуле происходит управление счетами, платежами, налогами, составление финансовой отчетности. Логистика и складской учет управляет запасами, закупками и поставками. Модуль CRM (Customer Relationship Management) предоставляет возможность вести клиентскую базу и управлять продажами. Кадровая система включает в себя учет работников, расчет заработной платы, отпусков и графиков работы. Производственный модуль позволяет планировать производственные действия, контролировать выполнение заказов, распределять ресурсы [3].

ERP-системы различаются по способу развертывания. Локальные решения (on-premise) дают возможность достаточно гибкой настройки за счет того, что устанавливаются на серверах внутри компании, однако, это требует от компании значительных затрат на ИТ-инфраструктуру. Облачные ERP (cloud-based) работают через интернет и предоставляются по модели SaaS. Это делает их куда более доступными для малого и среднего бизнеса. Облачные решения не требуют установки, обеспечивают масштабируемость и снижают нагрузку на внутренние ИТ-ресурсы, однако, они могут иметь ограничения по кастомизации, а также требуют стабильного интернет-соединения [2].

Потребности МСБ и соответствие возможностям ERP. Про малый и средний бизнес (МСБ) можно сказать, что отличительной чертой является относительная ограниченность в финансах, высокая динамичность процессов и необходимость быстрой и гибкой реакции на рыночные изменения. При таких обстоятельствах предприятия особенно остро проявляют потребность в средствах оптимизации управления, уменьшения издержек и повышения качества принятия решений [1].

ERP-системы способны решать широкий спектр задач, с которыми ежедневно сталкиваются компании малого и среднего бизнеса. Среди этих задач – автоматизация бухгалтерского и налогового учёта, контроль движения денежных средств, управление закупками и запасами, планирование продаж, взаимодействие с клиентами, формирование отчётности, а также организация внутреннего документооборота [3]. За счёт интеграции этих процессов в единую систему, ERP позволяет повысить прозрачность и управляемость бизнеса, уменьшить количество ручных операций и снизить риск ошибок.

На современном рынке можно обнаружить очень много решений ERP, которые были адаптированы под МСБ. Одно из широко используемых решений в русскоязычном сегменте – 1С:ERP, которая дает широкий функционал и возможность гибкой настройки под специфику предприятия [4]. Odoo – открытое исходное решение ERP-системы, ориентированное на легкость внедрения и возможности масштабирования. Она особенно интересна малому бизнесу за счёт своей доступности и гибкости [5].

Bitrix24, хоть и нельзя представить как классическую ERP-систему, но она обеспечивает обширный спектр контроля над задачами, проектами, продажами и базой клиентов. Такой набор качеств привел к тому, что она стала популярной у малых компаний. SAP BusinessOne и Microsoft Dynamics 365 – более функционально насыщенные решения, предоставляя более обширный набор функций управления ресурсами, финансами и взаимодействием с клиентами. Эти решения предназначены для предприятий, находящихся на этапе активного роста и требующих более глубокой аналитики и масштабируемости [6].

Преимущества внедрения ERP в МСБ. ERP-системы значительно повышают эффективность работы небольших компаний, так как автоматизируют и объединяют все важные процессы.

Одно из главных достоинств ERP – это централизованное хранение данных. Раньше компании использовали разные таблицы и программы, а информацию передавали вручную между отделами. Теперь же ERP позволяет всем отделам работать с одной базой данных, что устраняет дублирование, уменьшает ошибки и обеспечивает актуальные данные в реальном времени.

ERP также улучшает прозрачность бизнес-процессов. Руководители могут быстро получать свежие отчеты, следить за выполнением задач и анализировать ключевые показатели эффективности. Это помогает быстрее принимать решения и делает бизнес более гибким и управляемым.

Важно отметить улучшенное управление запасами и финансами. С помощью ERP можно точно отслеживать товарные потоки, минимизировать лишние запасы и снижать расходы на хранение. Финансовые функции дают возможность контролировать денежные потоки, автоматизировать расчеты и готовить финансовую отчетность с учетом налогов.

ERP-системы также часто включают в себя возможности для работы с клиентами (CRM). Это позволяет компаниям лучше понимать потребности клиентов, отслеживать историю покупок, предлагать

персональные предложения и строить долгосрочные отношения с клиентами.

Таким образом, ERP значительно упрощает управление бизнесом, повышает его конкурентоспособность и помогает малым и средним предприятиям расти устойчиво, приспосабливаясь к изменениям на рынке.

Ограничения и сложности внедрения ERP в МСБ. Несмотря на то, что внедрение ERP предоставляет МСБ множество преимуществ, но также это сопряжено с некоторыми ограничениями и сложностями, которые могут сильно повлиять на эффективность и успешность проекта не в самую лучшую сторону.

Одной из главных проблем для МСБ является высокая стоимость внедрения и сопровождения. ERP-системы, особенно комплексные, требуют немалых затрат финансов на покупку лицензий, установку, настройку и обучение персонала. Кроме того, необходимо помнить про расходы на техническую поддержку и обновления системы. Для малых и средних предприятий, которые часто ограничены в бюджете, эти расходы могут быть значительным барьером [2].

Сопротивление персонала изменениям – ещё одна сложность с которой можно столкнуться. Внедрение новой системы часто вызывает недовольство со стороны сотрудников, особенно если они долгое время работали с устаревшими методами или программами и впоследствии привыкли к ним. Персонал может быть не готов к изменениям, что замедляет процесс перехода на новую систему и требует дополнительных усилий для обучения и мотивации [1].

Недостаток IT-компетенций в компании тоже является серьёзной проблемой. Для успешного внедрения ERP необходимы специалисты, которые могут правильно настроить систему, интегрировать её с другими инструментами, а также решать возникающие технические проблемы. Малые и средние предприятия в большинстве случаев вряд ли имеют в штате IT-специалистов, достаточно квалифицированных для указанных задач, что требует привлечения внешних консультантов или подрядчиков, а это также увеличивает затраты.

Проблемы кастомизации под уникальные процессы малого бизнеса – ещё одно препятствие. Большинство ERP-систем создаются с оглядкой на потребности крупных компаний, поэтому их стандартная настройка может не подходить для специфики работы малого бизнеса. Кастомизация системы под уникальные процессы требует дополнительных временных и финансовых затрат, а еще это может быть технически сложной задачей [3].

Рекомендации для успешного внедрения ERP-системы в МСБ. Подготовка к внедрению ERP-системы требует комплексного подхода. Чтобы процесс внедрения прошёл максимально гладко и принёс положительные результаты, необходимо заранее подготовить бизнес и сотрудников, а также тщательно выбрать систему.

Для начала важно провести анализ текущих бизнес-процессов компании и определить, те процессы, которые как раз требуют автоматизации. Необходимо выявить слабые места в управлении запасами, финансами, логистикой, производственными процессами или другими аспектами. Важно, чтобы руководство и основные сотрудники были вовлечены в этот процесс. Должно присутствовать понимание, какие задачи ERP-система должна решить. Также стоит учитывать масштабы и особенности бизнеса, так как выбор системы зависит от размера предприятия, его специфики относительно отрасли и бюджета.

Выбор подходящей системы – это ключевой момент в процессе внедрения ERP. На рынке представлено множество решений, как для крупных корпораций, так и для малых и средних предприятий. Для МСБ важно выбрать систему, которая будет соответствовать бюджету и масштабам бизнеса, но при этом обеспечит все необходимые функции. Например, решения 1C:ERP или Odoo – это популярные и относительно доступные варианты для МСБ, которые позволяют автоматизировать все основные бизнес-процессы [4], [5]. Также стоит обращать внимание на гибкость системы: возможность её настройки под уникальные требования бизнеса и масштабы роста компании. Важно также учитывать будущее расширение бизнеса – система должна быть масштабируемой, чтобы поддерживать рост компании в будущем.

Роль обучения персонала и сопровождения системы играют не менее важную роль, чем выбор программного обеспечения. Даже самая продвинутая ERP-система не даст желаемого эффекта, если сотрудники не смогут эффективно работать с ней. Обучение должно быть комплексным и доступным для всех пользователей: от бухгалтеров и логистов до менеджеров и руководителей. Это помогает не только быстро освоить систему, но и минимизировать количество ошибок на ранних стадиях внедрения. Также важно организовать постоянную техническую поддержку и сопровождение системы на всех этапах работы. Это гарантирует своевременное решение проблем, связанных с работой программы, а также обеспечивает необходимое обновление и настройку системы в соответствии с изменяющимися требованиями бизнеса [2].

Для успешного внедрения также рекомендуется работать с профессиональными консультантами или партнёрами ERP-поставщика, которые помогут в настройке системы и обеспечат минимальные риски в процессе внедрения.

Выводы. Внедрение ERP-систем в малых и средних предприятиях представляет собой важный шаг к автоматизации бизнес-процессов, повышению эффективности и прозрачности работы. Обобщая все рассмотренное выше, можно сказать, что правильный выбор ERP-системы, подготовка бизнеса, обучение сотрудников и сопровождение внедрения являются ключевыми факторами для успешного использования ERP в МСБ. Системы, такие как 1C:ERP, Odoo и другие, могут существенно улучшить управление запасами, производственными процессами, финансами и взаимоотношениями с клиентами, что позволяет предприятиям значительно снижать операционные издержки и улучшать качество обслуживания.

Однако внедрение ERP-системы в МСБ не обходится без определённых сложностей, таких как высокая стоимость внедрения, сопротивление персонала изменениям и необходимость адаптации системы под уникальные потребности бизнеса. Эти проблемы требуют тщательной подготовки, внимательного подхода и наличия квалифицированной поддержки.

Перспективы дальнейших исследований могут быть направлены на изучение влияния ERP-систем на конкретные отрасли и их долгосрочную эффективность в различных условиях. Помимо всего прочего стоит обратить внимание на новые подходы к внедрению облачных ERP-решений, которые становятся всё более популярными среди малых и средних компаний благодаря своей гибкости, снижению затрат на инфраструктуру и возможности быстрого масштабирования. Особое внимание в будущих исследованиях следует уделить вопросам кастомизации ERP-систем для нужд специфических бизнес-моделей и интеграции с другими современными IT-решениями.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Абрамов, Р.Н. ERP-системы: теория и практика внедрения / Р.Н. Абрамов. – М.: Инфра-М, 2020. – 264 с.
2. Чубуков, В.Н. Управление предприятием на основе ERP-систем / В.Н. Чубуков. – СПб.: Питер, 2019. – 320 с.
3. Официальный сайт 1C:ERP [Электронный ресурс]. — URL: <https://v8.1c.ru/erp/> (дата обращения: 05.04.2025).
4. OdooDocumentation [Электронный ресурс]. — URL: <https://www.odoo.com/documentation> (дата обращения: 05.04.2025).

5. Бочаров, А.Н. Информационные технологии в экономике и управлении / А.Н. Бочаров. – М.: Юрайт, 2021. – 412 с.

6. SAP Business One – Product Information [Электронный ресурс]. – URL: <https://www.sap.com/products/business-one.html> (дата обращения: 05.04.2025).

УДК 004.422.81

БОРЗЫХ Д.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРОЕКТИРОВАНИЕ ПРОГРАММЫ СБОРА И ОБРАБОТКИ ИНФОРМАЦИИ СИСТЕМ МОНИТОРИНГА ИНЖЕНЕРНЫХ КОНСТРУКЦИЙ

Рассматриваются вопросы разработки программы для регистрации данных от систем мониторинга инженерных конструкций.

В современном мире системы мониторинга инженерных конструкций становятся неотъемлемой частью управления безопасностью и надежностью различных объектов. Системы мониторинга инженерных конструкций (СМИК) представляют собой системы, осуществляющие в режиме реального времени контроль состояния оснований и несущих конструкций зданий и сооружений. Эти системы предназначены для предупреждения чрезвычайных ситуаций, что делает их критически важными для обеспечения безопасности жизнедеятельности населения. В данной статье рассматривается основная теория для разработки модуля регистрации информации с датчиков систем мониторинга инженерных конструкций.

Описание систем мониторинга инженерных конструкций

Система мониторинга инженерных конструкций – это совокупность технических и программных средств, позволяющая осуществлять сбор и обработку информации о различных параметрах строительных конструкций (геодезические, динамические, деформационные и др.). Цель СМИК – оценка технического состояния зданий и сооружений и предупреждение чрезвычайных ситуаций.

Системы мониторинга инженерных конструкций играют критически важную роль в обеспечении безопасности, функциональности и долговечности особо важных сооружений. К таким сооружениям могут относиться мосты, жилые здания, заводские

сооружения, высотные конструкции и т.д. Системы мониторинга инженерных конструкций состоят из нескольких ключевых компонентов, каждый из которых играет важную роль в обеспечении эффективного сбора и анализа данных.

1. Датчики

Системы используют различные датчики для сбора данных в режиме реального времени. Основные виды датчиков включают:

- Датчики измерения угла наклона (инклинометры и др.): позволяют измерять угол наклона несущих конструкций в двух плоскостях.

- Датчики вибрации (акселерометры): выявляют аномальные колебания, которые могут указывать на структурные проблемы.

- Датчики температуры и влажности: контролируют стабильность параметров окружающей среды и условия эксплуатации конструкций из древесины и других материалов.

- Датчики деформации: измеряют изменения в форме или размере конструкций, что может указывать на их ослабление или повреждение.

2. Системы передачи данных

После сбора данных датчики передают информацию на центральный сервер или в облачное хранилище. Для передачи данных зачастую используются проводные решения.

3. Центральный сервер или облачное хранилище

Здесь происходит обработка, хранение и анализ полученных данных. В зависимости от объема и сложности данных может использоваться локальный сервер или облачные решения, обеспечивающие масштабируемость и доступность данных.

4. Программное обеспечение для обработки данных

Это специализированные программы, которые позволяют анализировать собранные данные, визуализировать их и генерировать отчеты.

Принцип работы и описание цифрового инклинометра

Одними из основных датчиков в таких системах являются датчики наклона, или инклинометры. Рассмотрим принцип работы цифрового инклинометра.

Принцип действия инклинометра основан на изменении положения гравиметрического маятника в одном направлении под воздействием силы тяжести или ускорения. Маятник расположен внутри двухосевого акселерометра, чувствительным элементом которого является дифференциальный конденсатор. Изменения положения маятника приводит к соответствующему изменению

электрической ёмкости дифференциального конденсатора. Такие изменения ёмкости преобразуются осциллятором в частоты. Выходной сигнал вначале фильтруется, а затем, с помощью специального программного обеспечения, преобразуется в цифровую 16-битовую форму, которая является отображением значений углов наклона.

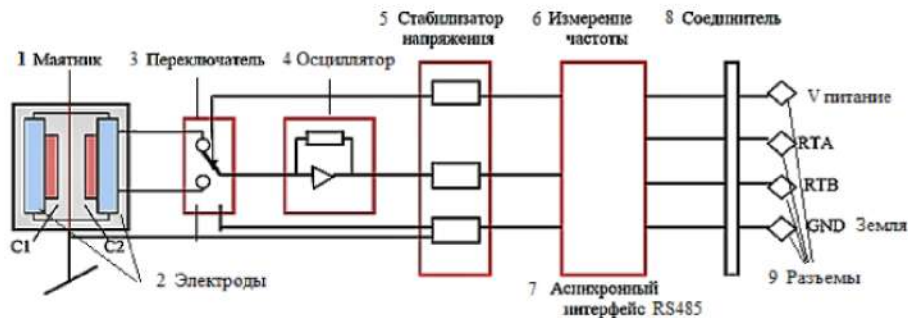


Рисунок 1 – Схема цифрового инклинометра на основе конденсатора

Дальше в работе будут рассматриваться именно датчики наклона, из-за их распространенности. Инклинометры находят широкое применение в различных областях, включая:

Строительство: мониторинг состояния зданий и сооружений в процессе их эксплуатации, а также контроль за деформациями при строительстве.

Геотехника: оценка устойчивости склонов, дамб и других земляных конструкций.

Мостостроение: контроль за деформациями и состоянием мостовых конструкций.

Энергетика: мониторинг состояния башен ветряных электростанций и других объектов.

Анализ существующих программных решений и алгоритмов

Существуют различные программные решения для мониторинга инженерных конструкций, каждое из которых имеет свои особенности и функциональные возможности. Однако многие из них не являются доступными для широкой аудитории. Одним из таких решений является программа Vlogic, которая используется для проверки работоспособности цифровых инклинометров и прошивки датчиков.

Основные функции Vlogic:

- Проверка работоспособности датчиков.
- Проверка корректности данных, поступающих от датчиков по каналам: координаты X, Y и температура.
- Задание логических номеров датчикам.

Недостатки Vlogic:

- Отсутствие возможности сбора данных. Программа предоставляет только возможность просмотра данных в реальном времени, что ограничивает ее функционал.

- Невозможность одновременного отображения данных с нескольких датчиков. Это затрудняет комплексный анализ состояния конструкций.

- Отсутствие визуализации полученных данных. Пользователи не могут удобно анализировать данные, что снижает эффективность мониторинга.

Эти недостатки подчеркивают необходимость разработки более функционального решения, которое будет учитывать современные требования к системам мониторинга.

Алгоритмы сбора и обработки данных являются основой для эффективного мониторинга инженерных конструкций. Они обеспечивают получение, фильтрацию и интеграцию информации с различных датчиков, что позволяет преобразовать сырые данные в полезную информацию для анализа состояния конструкций.

Типы сбора данных:

- Периодический сбор данных

- Событийно-ориентированный сбор данных

В рамках разработки собственного решения планируется использовать периодический сбор данных, так как это обеспечит наибольшую эффективность. Данные в системе СМИК меняются плавно, а краткосрочные аномалии не представляют большого интереса при анализе.

Разработка программного модуля

Разработка программы производится на языке программирования Java с использованием фреймворка Spring. Основной вид датчиков для получения информации – инклинометр цифровой. Данные с датчиков планируется получать в режиме реального времени через СОМ-порт, обрабатываются программой и сохраняются в БД для возможности последующего доступа. В процессе разработки собственного решения планируется избежать недостатков существующих программных решений и разработать программный модуль, выполняющий следующие функции:

- Получение информации с цифровых инклинометров;

- Вывод получаемой информации на экран ЭВМ;

- Построение графиков полученных данных о наклоне сооружений;

- Обработка полученных данных и отображение критических значений;

- Сохранение полученной информации в базу данных;
- Ведение записей о событиях.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Хиллер Бернд. Разработка и исследование автоматизированной системы геодезического деформационного мониторинга инженерных сооружений на основе высокоточной цифровой инклинометрии и тахеометрии. – Москва: МИИГАиК, 2017. – 144 с.

2. Программа по работе с инклинометрами Vlogic. Руководство по эксплуатации [Электронный ресурс]. – URL: <https://vclass.su/software> (дата обращения: 11.04.2025).

3. Егоров Ф.А. Мониторинг инженерных конструкций на примере оценки последствий воздействия челябинского метеорита/ Егоров Ф.А., Неугодников А.П., Быковский В.А., Шерстюк С.П.//Мир измерений. – 2013. – №13. – С. 46-51.

УДК 004.4'422

БУБЛИКОВ Д.И., МАРКИН А.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА КОМПИЛЯТОРА SQL

Рассматривается процесс разработки компилятора SQL для миграции базы данных.

В информационных системах (ИС) иногда приходится мигрировать базу данных (БД). Причиной могут быть изменения в лицензии системы управления базы данных (СУБД), повышение стоимости лицензии, требование оптимизировать бизнес-процессы с помощью функционала целевой СУБД и т.д.

В контексте миграции БД, исходная СУБД – та СУБД, которая используется в информационной системе до окончания миграции, а целевая СУБД – та, которая будет использоваться после завершения миграции.

В качестве примера производится миграция БД web-версии расчётной системы (web PC) комплекса «Абонент+» [1] (рис. 1). В ней в качестве исходной СУБД выступает Firebird 3.0.5 [2], а в качестве целевой – PostgreSQL [3].



Рисунок 1 – Комплекс «Абонент+»

Задача миграции БД состоит из двух подзадач:

- 1) перенос данных из исходной БД в БД целевой СУБД;
- 2) перевод SQL кода [4] исходной СУБД в код целевой СУБД.

Для решения первой подзадачи решено использовать программу Full Convert [5]. Для решения второй подзадачи решено разработать компилятор SQL, переводящий код Firebird в код PostgreSQL. Код Firebird описывается языком PSQL, а код PostgreSQL – языком PL/pgSQL.

Для разработки компилятора необходимо:

- 1) проанализировать переводимый код;
- 2) определить компоненты компилятора;
- 3) выбрать программные средства для его разработки;
- 4) реализовать компилятор;
- 5) протестировать его.

1 Анализ переводимого кода

В БД SQL код – это определение объекта БД. Поэтому перевод кода заключается в переводе определений объектов. В Firebird существуют объекты различных видов: таблицы, хранимые процедуры, последовательности, триггеры, фильтры BLOB и т.д. В БД web PC есть только объекты следующих видов:

- 1) таблицы;
- 2) представления;
- 3) индексы;
- 4) последовательности;
- 5) домены;
- 6) пользовательские исключения;

- 7) хранимые процедуры (ХП);
- 8) хранимые функции (ХФ);
- 9) триггеры.

Помимо переноса данных, Full Convert может также перевести определения таблиц, представлений, индексов и последовательностей. Задача компилятора SQL – перевести определения остальных объектов БД, которые Full Convert перевести не может: доменов; ХП, ХФ, исключений и триггеров.

1.1 Анализ доменов

Домен – пользовательский тип данных, основанный на предопределённом в СУБД типе данных. Домен позволяет совместить тип данных с атрибутами, такими как значение по умолчанию и ограничения NULL и CHECK. Все атрибуты домена, какие есть в Firebird, также имеются в PostgreSQL. Они обрабатываются следующим образом.

1. NOT NULL – остаётся как есть;
2. CHECK – содержит выражение, оно переводится по общим правилам перевода выражений;
3. DEFAULT – содержит либо константу, либо контекстную переменную. Константы переводятся, а контекстные переменные проверяются на совместимость с PostgreSQL, т.к. не у всех контекстных переменных Firebird есть аналог в PostgreSQL.

1.2 Анализ пользовательских исключений

Firebird позволяет определять пользовательские исключения. При создании пользовательского исключения для него определяется его название и сообщение об ошибке. Исключение можно вызвать с помощью оператора EXCEPTION, указав название исключения. При вызове исключения выполнение кода PSQL прерывается и показывается его сообщение об ошибке.

В PostgreSQL нет пользовательских исключений. В PL/pgSQL для вызова исключения применяется оператор RAISE EXCEPTION, и при использовании этого оператора вместо названия исключения указывается само сообщение об ошибке.

Для перевода исключения достаточно при переводе оператора EXCEPTION в оператор RAISE EXCEPTION заменить название исключения его сообщением об ошибке.

1.3 Анализ хранимых процедур

В Firebird существует два вида ХП:

- 1) селективные;
- 2) неселективные.

Селективные ХП возвращают последовательность строк и могут быть использованы как таблица в любом табличном выражении. Например, в операторе FROM. Неселективные ХП не возвращают строк, и они вызываются оператором EXECUTE PROCEDURE.

В PostgreSQL из ХП невозможно возвращать строки. Аналог селективных ХП Firebird – функции, возвращающие последовательность строк, т.е. табличные ХФ. Таким образом, при переводе неселективные ХП остаются ХП, а селективные ХП заменяются табличными ХФ.

Определение ХП состоит из тела, заголовка и списка объявлений. В Firebird тело является оператором BEGIN END, и оно переводится по общим правилам перевода операторов PSQL, за исключением того, что оператор EXIT переводится как оператор RETURN без аргументов.

Заголовок содержит название ХП и описание входных и выходных параметров.

Входной параметр содержит его название, тип и значение по умолчанию. Тип и значение по умолчанию переводится в соответствие с общими правилами. Выходные параметры переводятся аналогично, за исключением того, что не переводится значения по умолчанию, т.к. его нет у выходных параметров.

Список объявлений содержит объявления локальных переменных, курсоров, подпроцедур и подфункций

Объявление локальной переменной содержит её название, тип и значение по умолчанию. Тип и значение по умолчанию переводятся по общим правилам перевода. Название локальной переменной меняется, к нему добавляется приставка «VAR_». В Firebird локальные переменные и столбцы таблиц могут иметь одинаковые названия, но в PostgreSQL они не могут. Поэтому необходимо переименовать все локальные переменные, чтобы их названия не пересекались с названиями столбцов. Это общее правило перевода локальных переменных.

Объявление курсора содержит его название, разрешение или запрет на его прокрутку и запрос SELECT, результаты которого курсов возвращает. Название и разрешение или запрет на прокрутку переводятся как есть. Запрос переводится по общим правилам перевода запросов SELECT.

PostgreSQL не поддерживает ни подпроцедуры ни подфункции, поэтому при переводе они преобразуются в обычные ХП и ХФ соответственно. Названия подпроцедур или подфункций могут повторяться, т.к. они должны быть уникальными только в пределах

ХП или ХФ, которая её определила. Поэтому при их преобразовании в обычные ХП и ХФ может возникнуть ошибка из-за повторяющихся названий. Чтобы избежать ошибки при их преобразовании к их названиям слева добавляется название функции, внутри которой они были определены.

1.4 Анализ хранимых функций

В Firebird ХФ всегда возвращают одно значение, т.е. являются скалярной. В PostgreSQL ХФ является скалярной, если при её определении в секции RETURN не указано ни SET OF ни TABLE.

Определение ХФ состоит из её тела, заголовка, списка объявлений. Тело переводится по тем же правилам, по каким переводится тело ХП, но с одним отличием. Оператор EXIT в ХФ Firebird равнозначен оператору RETURN с аргументом NULL, поэтому для ХФ оператор EXIT переводится как RETURN NULL.

Заголовок ХФ содержит название ХФ, её входные параметры, тип возвращаемого значения и опциональной пометки, что она детерминированная. Входные параметры переводятся так же, как они переводятся для ХП. Возвращаемый тип переводится по общим правилам перевода типов данных. Пометка детерминированности ХФ есть как в Firebird, так и в PostgreSQL, но с другим ключевым словом. В Firebird – DETERMINISTIC, а в PostgreSQL – IMMUTABLE. Следовательно, при переводе DETERMINISTIC заменяется IMMUTABLE.

Список деклараций ХФ переводится так же, как он переводится у ХП.

1.5 Анализ триггеров

В Firebird существует 3 вида триггеров.

1. Data Manipulation Language (DML) триггер. Такой триггер срабатывает при выполнении DML запроса (INSERT, UPDATE или DELETE) к таблице или представлению, на котором он определён.

2. Data Definition Language (DDL) триггер. Такой триггер срабатывает при выполнении запроса CREATE, ALTER или DROP на определённом виде объектов БД. Например, можно создать триггер, реагирующий на создание только индексов, т.е. запрос CREATE INDEX.

3. Триггер БД. Такие триггеры срабатывают при работе с подключениями и транзакциями. Существует 5 видов триггеров БД:

3.1. CONNECT – срабатывает при подключении к БД;

3.2. DISCONNECT – срабатывает при отключении от БД;

3.3. TRANSACTION START – срабатывает при начале транзакции;

3.4. TRANSACTION COMMIT – срабатывает при фиксации транзакции;

3.5. TRANSACTION ROLLBACK – срабатывает при откате транзакции.

В PostgreSQL существует 2 вида триггеров.

1. DML триггер. Как и в Firebird, в PostgreSQL DML триггеры обрабатывают DML запросы, выполняемые к таблицам и представлениям.

2. Событийный триггер. Он реагирует на одно из следующих событий:

2.1. ddl_command_start – срабатывает перед выполнением DML запроса;

2.2. ddl_command_end – срабатывает после выполнения DML запроса;

2.3. table_rewrite – срабатывает после выполнения запросов ALTER TABLE и ALTER TYPE;

2.4. sql_drop – срабатывает после выполнения запроса DROP.

DML триггеры Firebird можно перевести в DML триггеры PostgreSQL.

DLL триггеры можно перевести в событийные триггеры, реагирующие на события ddl_command_start и ddl_command_end.

Триггеры БД невозможно перевести, т.к. в PostgreSQL нет триггеров, реагирующие на те же самые события.

Компилятор SQL не просто переводит код Firebird, но и проводит оптимизацию, триггеры заменяются эквивалентным декларативным кодом. Вследствие этого триггеры можно разбить на две группы: заменяемые декларативным кодом, и не заменяемые. Если триггер является заменяемым, существует эквивалентный ему декларативный код SQL. Триггер и декларативный код являются эквивалентными, если они выполняют одно и то же действие при происхождении одного и того же события.

Чтобы заменить триггер эквивалентным ему декларативным кодом, необходимо сначала определить такой код. Для поиска декларативного кода, которым можно заменить триггеры, использована следующая методика. Поскольку DML триггер производит какое-то действие, при выполнении DML запроса к таблице, эквивалентный ему декларативный код также должен определять, что производится какое-то действие, при выполнении DML запроса. Чтобы найти такой декларативный код, необходимо проанализировать все существующие виды SQL запросов и среди них отобрать те, которые определяют, что при выполнении DML запроса к

таблице производится какое-то действие. Затем необходимо проверить, что возможно создать триггер, эквивалентный отобранному запросу. По результату анализа всех видов запросов в PostgreSQL, отобраны следующие виды запросов, и для каждого проведён анализ, возможно ли создать эквивалентный триггер. Многоточие в виде запроса обозначает часть, где подставляются параметры запроса, например, название таблицы.

1. Запрос вида «ALTER TABLE ... ALTER COLUMN ... SET DEFAULT ...» позволяет декларативно задать для столбца таблицы значение по умолчанию, и это значение будет использовано, если в запросе INSERT не указать значение для столбца или явно запросить значение по умолчанию с помощью ключевого слова DEFAULT. Эквивалентный триггер создать невозможно. Чтобы доказать это, рассмотрим 3 запроса, представленные в листинге 1. Перед запросом указан его номер.

Листинг 1 – Запросы, использующие значение по умолчанию

```
-- 1
INSERT INTO NACHISLSUMMA(SUMMA) VALUES (100);
-- 2
INSERT INTO NACHISLSUMMA(ID, SUMMA) VALUES (DEFAULT,
100);
-- 3
INSERT INTO NACHISLSUMMA(ID, SUMMA) VALUES (NULL, 100);
```

Декларативно заданное значение по умолчанию сработает только для запросов 1 и 2. Реализовать выставление значения по умолчанию в триггере возможно только путём проверки, содержит ли столбец NULL с последующей заменой NULL значением по умолчанию, поэтому выставление значения по умолчанию с помощью триггера сработает для всех трёх запросов. В итоге эквивалентного триггера не существует по следующим причинам.

1.1. В триггере невозможно различить, имеет ли столбец NULL, потому что ему не задали значение (запрос 1), или потому что ему явно задали NULL (запрос 3), а декларативно заданное значение по умолчанию может различить эти два случая.

1.2. Так как триггер заменяет NULL, он не только реализует установление значения по умолчанию, но и не позволяет добавляемой строке содержать NULL в заданном столбце.

2. Запрос вида «ALTER TABLE ... ALTER COLUMN ... ADD GENERATED BY DEFAULT AS IDENTITY ...» определяет столбец как столбец идентификации. Создаётся последовательность и у столбца в качестве значения по умолчанию определяется следующее

значение этой последовательности. Также на столбец накладывается ограничение NOT NULL.

Для этого вида запросов также невозможно создать эквивалентный триггер. Если попытаться явно установить значение NULL для столбца идентификации, как в запросе 3, возникнет ошибка из-за нарушения ограничения NOT NULL, а если использовать триггер, вместо NULL будет выставлено следующее значение последовательности и ошибки не будет.

3. Запрос вида «ALTER TABLE ... ALTER COLUMN ... ADD GENERATED ALWAYS AS IDENTITY ...», как и вид запроса в пункте 2, определяет столбец как столбец идентификации, но запрещает задавать все значения, кроме значения по умолчанию. Явно указывать NULL, как в запросе 3, также запрещается, если это сделать, запрос завершится ошибкой.

Эквивалентного триггера не существует. Если бы он существовал, такой триггер должен был бы различать, содержит столбец NULL, потому что явно задано значение NULL (запрос 3) или потому что значение не задано (запрос 1). Т.к. триггеры не могут различать эти случаи, эквивалентного триггера не существует.

4. Запрос вида «ALTER TABLE ... ALTER COLUMN ... SET NOT NULL» накладывает ограничение «NOT NULL» на таблицу. Это ограничение запрещает столбцу иметь значение NULL, т.е. при выполнении запроса INSERT или UPDATE, если столбцу присваивается значение NULL, запрос завершается ошибкой. Триггер может проверить, присвоено ли столбцу значение NULL и, в таком случае, вызвать ошибку. Поэтому возможно реализовать эквивалентный триггер.

5. Запрос вида «ALTER TABLE ... ADD CHECK ...» накладывает ограничение «CHECK» на таблицу. Это ограничение определяет условие, которое должно выполняться для каждой строки в таблице, т.е. при выполнении запроса INSERT или UPDATE проверяется, выполняется ли это условие для новой (добавляемой или обновлённой) строки, и, если оно не выполняется, запрос завершается ошибкой.

Заданное условие должно быть ретроспективно детерминистическим (retrospectively deterministic) [6], т.е. если условие выполняется в момент обновления или добавления строки и после этого содержимое строки не меняется, то условие должно выполняться и в будущем. Если условие не является ретроспективно детерминистическим, восстановление БД из резервной копии может завершиться ошибкой [7].

Эквивалентный триггер проверяет строку на выполнение условия, и в случае, если условие не выполняется, вызывает ошибку. Чтобы можно было заменить триггер ограничением CHECK, проверяемое им условие должно удовлетворять следующие требования.

5.1. Условие должно быть ретроспективно детерминистическим.

5.2. Условие не должно содержать подзапросы, т.к. PostgreSQL запрещает подзапросы в ограничении CHECK.

6. Запрос вида «ALTER TABLE ... ADD UNIQUE ...» накладывает ограничение уникальности на таблицу. Это ограничение запрещает двум строкам в таблице иметь одни и те же значения для заданных столбцов, т.е. при выполнении запроса INSERT или UPDATE проверяется, есть ли уже в таблице строка, у которой такие же значения заданных столбцов, как у новой строки, и, если есть, вызывается ошибка. Возможно реализовать триггер, проводящий такую проверку. Проверка может быть проведена, например, с помощью запроса SELECT, который ищет в таблице строку с такими же значениями заданных столбцов, как у новой строки.

7. Запрос вида «ALTER TABLE ... ADD PRIMARY KEY ...» накладывает ограничение первичного ключа на таблицу. Ограничение первичного ключа равнозначно ограничению уникальности в паре с ограничением NOT NULL. Поэтому триггер, эквивалентный ограничению первичного ключа, можно создать, объединив логической операцией «И» условия, проверяемые следующими триггерами.

7.1. Триггер, эквивалентный ограничению уникальности.

7.2. Триггер, эквивалентный ограничению NOT NULL.

8. Запрос вида «ALTER TABLE ... ADD FOREIGN KEY ...» накладывает ограничение внешнего ключа на таблицу. Таблица, на которую наложено это ограничение, называется дочерней. Это ограничение требует, чтобы значения группы столбцов дочерней таблицы равнялись значениям группы столбцов другой таблицы, называемой родительской. Если группа столбцов строки дочерней таблицы содержит значения, равные значениям группы столбцов строки родительской таблицы, то строка дочерней таблицы ссылается на строку родительской таблицы.

Это ограничение производит следующие проверки. При выполнении запроса INSERT или UPDATE на дочерней таблице проверяется, что новая строка ссылается на какую-либо строку в родительской таблице. Если такая строка в родительской таблице не найдена, производится действие, поддерживающее ссылочную

целостность. А при выполнении запросов UPDATE и DELETE к родительской таблице проверяется, что в дочерней таблице нет строк, ссылающихся на старую (обновляемую или удаляемую) строку. Если такие строки есть, производится действие, поддерживающее ссылочную целостность.

Действие, поддерживающее ссылочную целостность, задаётся при создании ограничения. Для запроса INSERT есть только одно действие, поддерживающее ссылочную целостность – вызов ошибки. А для запросов UPDATE и DELETE есть следующие действия.

8.1. NO ACTION – вызов ошибки.

8.2. RESTRICT – то же самое, что NO ACTION.

8.3. CASCADE – для запроса DELETE удаляются дочерние строки, которые ссылались на удаляемую строку в родительской таблице, а для UPDATE обновляются значения столбцов строки дочерней таблицы, чтобы они остались равными значениям строки родительской таблицы. Эти действия называются каскадным удалением и обновлением соответственно.

8.4. SET NULL – установление значения NULL столбцам строки дочерней таблицы.

8.5. SET DEFAULT – установление значения по умолчанию столбцам строки дочерней таблицы.

Чтобы реализовать эквивалентный триггер, необходимо, чтобы в триггере было возможно проверить ссылочную целостность и выполнить все перечисленные действия, поддерживающие ссылочную целостность. Поскольку в триггере возможно провести эту проверку и выполнить эти действия, возможно реализовать эквивалентный триггер.

2 Компоненты компилятора

Компилятор SQL состоит из следующих компонентов:

- 1) лексически анализатор;
- 2) синтаксический анализатор;
- 3) генератор декларативного кода;
- 4) генератор процедурного кода.

На вход компилятор принимает код PSQL, в виде последовательности символов.

Лексический анализатор группирует в этой последовательности соседние символы в лексемы, и таким образом создаёт последовательность лексем. Лексический анализатор определяет, какие пробелы в коде являются необязательными. Например, в запросе «SELECT*FROM RDB\$DATABASE» Пробелы между SELECT, FROM и звёздочкой «*» отсутствуют, лексический анализатор и без них

распознаёт их как 3 отдельные лексемы. А пробел между FROM и RDB\$DATABASE – обязателен, в противном случае запрос содержал бы строку «FROMRDB\$DATABASE», и она считалась бы за одну лексему, а не 2.

Синтаксический анализатор преобразует последовательность лексем в абстрактное синтаксическое дерево (АСД). Использование АСД упрощает дальнейший анализ запроса, т.к. оно явно показывает иерархию между конструкциями языка.

Генератор декларативного кода производит обход АСД и проверяет, соответствует ли оно шаблону триггера, заменяемого декларативными правилами. Если соответствует, из АСД извлекается информация, необходимая для генерации декларативного кода, например, названия родительской и дочерней таблицы, и подставляет её в шаблон генерируемого декларативного кода. Если же АСД не соответствует шаблону заменяемого триггера, оно передаётся генератору процедурного кода.

Генератор процедурного кода извлекает информацию из каждой вершины АСД, подставляет её в соответствующий ей шаблон, и затем объединяет все результаты подстановки в одну строку, представляющую собой код PL/pgSQL.

3 Выбор программных средств

Язык программирования (ЯП), на котором разрабатывается компилятор, должен соответствовать следующим требованиям:

1. Должен быть статически типизированным. Статическая типизация позволяет отловить множество ошибок на этапе компиляции.

2. Должен поддерживать функциональное программирование. Функциональное программирование является более декларативным, чем императивное программирование, и минимизирует количество побочных эффектов (side effect [8]). Также работа компилятора заключается в анализе и модификации АСД, т.е. производится работа с данными, а не с объектами. Поэтому для этой задачи функциональное программирование подходит лучше, чем объектно-ориентированное программирование.

3. Должен иметь типы-суммы [9]. Типы-суммы упрощают моделирование АСД.

Этим требованиям удовлетворяют следующие языки программирования: F# [10], OCaml [11], Haskell [12].

Выбран Haskell из-за того, что он является чисто функциональным ЯП, т.е. код не может иметь побочных эффектов, если это не указано явно в типе функции. Также Haskell имеет самую

продвинутую систему типов, среди кандидатов, благодаря чему возможно реализовать шаблон проектирования Trees that Grow [13].

4 Экспериментальная часть

Для примера рассмотрим перевод селективной процедуры GET_MONTHS, генерирующая последовательность месяцев в году за заданный период (листинг 2). Чтобы протестировать её, необходимо:

- 1) перевести её в PostgreSQL;
- 2) выполнить её на наборе входных данных в Firebird и в PostgreSQL;
- 3) проверить, что результаты выполнения ХП одинаковые в обеих СУБД.

Листинг 2 – Определение ХП GET_MONTHS в Firebird

```
CREATE PROCEDURE GET_MONTHS (  
    BEGMONTH integer,  
    BEGYEAR integer,  
    ENDMONTH integer,  
    ENDYEAR integer,  
    STEP integer)  
RETURNS (  
    STATEMONTH integer,  
    STATEYEAR integer)  
AS  
declare variable BEGMONTH_CALC integer;  
declare variable STATEMONTH_CALC integer;  
declare variable ENDMONTH_CALC integer;  
begin  
    STATEMONTH = :BEGMONTH;  
    STATEYEAR = :BEGYEAR;  
    BEGMONTH_CALC = :BEGMONTH + 12 * :BEGYEAR;  
    STATEMONTH_CALC = :STATEMONTH + 12 * :STATEYEAR;  
    ENDMONTH_CALC = :ENDMONTH + 12 * :ENDYEAR;  
    while (STATEMONTH_CALC <= ENDMONTH_CALC) do  
        begin  
            suspend;  
            if (STATEMONTH + STEP <= 12) then  
                begin  
                    STATEMONTH = :STATEMONTH + :STEP;  
                end  
            else  
                begin  
                    STATEMONTH = (:STATEMONTH - 12 + STEP);  
                    STATEYEAR = :STATEYEAR + 1;  
                end  
            STATEMONTH_CALC = :STATEMONTH + 12 * :STATEYEAR;  
        end  
    end
```

end

Результат перевода ХП представлен в листинге 3.

Листинг 3 – Результат перевода ХП GET_MONTHS в PostgreSQL

```
create or replace function GET_MONTHS(
  VAR_BEGMONTH int,
  VAR_BEGYEAR int,
  VAR_ENDMONTH int,
  VAR_ENDYEAR int,
  VAR_STEP int)
returns TABLE(VAR_STATEMONTH int, VAR_STATEYEAR int)
language plpgsql
as $$
declare
  VAR_BEGMONTH_CALC int;
  VAR_STATEMONTH_CALC int;
  VAR_ENDMONTH_CALC int;
begin
  VAR_STATEMONTH = VAR_BEGMONTH;
  VAR_STATEYEAR = VAR_BEGYEAR;
  VAR_BEGMONTH_CALC = VAR_BEGMONTH + 12 *
    VAR_BEGYEAR;
  VAR_STATEMONTH_CALC = VAR_STATEMONTH + 12 *
    VAR_STATEYEAR;
  VAR_ENDMONTH_CALC = VAR_ENDMONTH + 12 *
    VAR_ENDYEAR;
  while VAR_STATEMONTH_CALC <= VAR_ENDMONTH_CALC loop
    return next;
    if VAR_STATEMONTH + VAR_STEP <= 12 then
      VAR_STATEMONTH = VAR_STATEMONTH + VAR_STEP;
    else
      VAR_STATEMONTH = VAR_STATEMONTH - 12 + VAR_STEP;
      VAR_STATEYEAR = VAR_STATEYEAR + 1;
    end if;
    VAR_STATEMONTH_CALC = VAR_STATEMONTH + 12 *
      VAR_STATEYEAR;
  end loop;
end;
$$;
```

Для тестирования ХП GET_MONTHS составлен запрос, представленный в листинге 4. Запрос работает в обеих СУБД. Результат выполнения запроса для Firebird представлен в таблице 1, а для PostgreSQL – в таблице 2.

Листинг 4 – Запрос для тестирования ХП GET_MONTHS

```
select * from get_months('9', '2023', '6', '2024', '1');
```

Таблица 1 – Результаты тестирования функции GET_MONTHS в Firebird

STATEMONTH	STATEYEAR
9	2023
10	2023
11	2023
12	2023
1	2024
2	2024
3	2024
4	2024
5	2024
6	2024

Таблица 2 – Результаты тестирования функции GET_MONTHS в PostgreSQL

var_statemonth	var_stateyear
9	2023
10	2023
11	2023
12	2023
1	2024
2	2024
3	2024
4	2024
5	2024
6	2024

Как следует из таблиц 1 и 2, ХП переведена верно.

Заключение

В итоге:

- 1) проанализирован переводимый код;
- 2) определены компоненты компилятора;
- 3) выбрать программные средства для его разработки;
- 4) компилятор реализован;
- 5) он протестирован на примере хранимой процедуры.

Из результатов тестирования следует, что компилятор работает верно.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Абонент+. Системы для биллинга и приема платежей за услуги ЖКХ [Электронный ресурс] – URL: <https://abonent.plus/> (дата обращения: 16.04.2025).
2. Firebird: About Firebird [Электронный ресурс] – URL: <https://firebirdsql.org/en/about-firebird/> (дата обращения: 16.04.2025).
3. PostgreSQL: About [Электронный ресурс] – URL: <https://www.postgresql.org/about/> (дата обращения: 16.04.2025).
4. Маркин, А.В. Программирование на SQL: учебник и практикум для вузов / А. В. Маркин. – 3-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2024. – 805 с.
5. Overview | Full Convert [Электронный ресурс] – URL: <https://www.fullconvert.com/overview> (дата обращения: 16.04.2025).
6. Breaking your PostgreSQL database with bad CHECK constraints / Laurenz Albe [Электронный ресурс] – URL: <https://www.cybertec-postgresql.com/en/bad-check-constraints-postgresql/> (дата обращения: 16.04.2025).
7. PostgreSQL: Documentation: 14: 5.4. Constraints [Электронный ресурс] – URL: <https://www.postgresql.org/docs/> (дата обращения: 16.04.2025).
8. The Not-So-Scary Guide to Functional Programming [Электронный ресурс] – URL: <https://www.yld.io/blog/the-not-so-scary-guide-to-functional-programming> (дата обращения: 16.04.2025).
9. Теория типов [Электронный ресурс] – URL: <https://habr.com/ru/articles/758542/> (дата обращения: 16.04.2025).
10. What is F# - .NET | Microsoft Learns [Электронный ресурс] – URL: <https://learn.microsoft.com/en-us/dotnet/fsharp/what-is-fsharp> (дата обращения: 16.04.2025).
11. Why OCaml? [Электронный ресурс] – URL: <https://ocaml.org/about> (дата обращения: 16.04.2025).
12. Haskell Language [Электронный ресурс] – URL: <https://www.haskell.org/> (дата обращения: 16.04.2025).
13. Trees that Grow / Shayan Najd, Simon Peyton Jones [Электронный ресурс] – URL: <https://www.microsoft.com/en-us/research/uploads/prod/2016/11/trees-that-grow.pdf> (дата обращения: 16.04.2025).

УДК 004.8

ВАСИНА М.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

НЕЙРОКОМПЬЮТЕРЫ, НЕЙРОИНТЕРФЕЙСЫ, НЕЙРОЧИПЫ И ИХ ПРИМЕНЕНИЕ

Исследование современного состояния и прогнозов развития нейрокомпьютеров, нейроинтерфейсов и нейрочипов.

Человеческий мозг остается одной из самых сложных и малоизученных систем в природе. Стремление понять принципы его работы и воспроизвести их с помощью технологий привело к формированию нескольких научно-технических направлений: созданию нейрокомпьютеров, разработке нейроинтерфейсов и проектированию нейрочипов. Эти технологии находятся на пересечении информатики, нейробиологии, биоинженерии и медицины, образуя мультидисциплинарную область исследований, которая активно развивается в последние десятилетия.

Актуальность данной темы обусловлена не только фундаментальным научным интересом к познанию принципов работы мозга, но и значительным практическим потенциалом нейротехнологий в различных сферах человеческой деятельности: от медицинской реабилитации пациентов с неврологическими нарушениями до создания принципиально новых вычислительных систем и интерфейсов человек-машина.

Основные понятия и принципы нейрокомпьютерных технологий.
Нейрокомпьютеры

Нейрокомпьютер представляет собой устройство переработки информации на основе принципов работы естественных нейронных систем. В отличие от традиционных вычислительных систем, принцип работы которых основан на последовательном выполнении алгоритмов, нейрокомпьютеры реализуют параллельную обработку информации с использованием моделей, имитирующих работу биологических нейронных сетей.

Центральным элементом нейрокомпьютера является искусственная нейронная сеть (ИНС), состоящая из множества взаимосвязанных искусственных нейронов. Искусственный нейрон, как и биологический, получает входные сигналы, обрабатывает их и формирует выходной сигнал. Особенностью искусственных нейронных сетей является их способность к обучению и адаптации на основе обрабатываемых данных [2].

Первые экспериментальные нейрокомпьютеры были созданы еще в 1950-х годах. Перцептрон Розенблатта (1958) и Адалин, разработанный Уидроу и Хоффом (1960), стали первыми практическими реализациями нейрокомпьютеров. Однако активное развитие данного направления началось в 1980-х годах, когда были разработаны эффективные алгоритмы обучения многослойных нейронных сетей.

Основные преимущества нейрокомпьютеров перед традиционными вычислительными системами заключаются в следующем:

- Высокая параллельность обработки информации, что обеспечивает высокое быстродействие;
- Устойчивость к помехам и частичным повреждениям отдельных элементов;
- Способность к обучению и самоорганизации;
- Возможность эффективного решения плохо формализуемых задач, таких как распознавание образов и принятие решений в условиях неопределенности.

Нейроинтерфейсы: соединение мозга с компьютером

Нейрокомпьютерный интерфейс (НКИ), также называемый интерфейсом "мозг-компьютер", представляет собой систему, созданную для обмена информацией между мозгом и электронным устройством. НКИ можно классифицировать по направлению передачи сигнала:

1. Однонаправленные интерфейсы:

- Считывающие (от мозга к компьютеру) – регистрируют нейронную активность и преобразуют ее в команды для внешних устройств;
- Стимулирующие (от компьютера к мозгу) – передают информацию в мозг, например, имитируя сенсорные сигналы.

2. Двухнаправленные интерфейсы – обеспечивают обмен информацией в обоих направлениях.

По степени инвазивности нейроинтерфейсы делятся на:

- Инвазивные – элементы интерфейса имплантируются непосредственно в мозг;
- Полуинвазивные – располагаются под черепом, но не проникают в мозговую ткань;
- Неинвазивные – располагаются на поверхности головы.

История развития нейроинтерфейсов началась в 1970-х годах в Калифорнийском университете в Лос-Анджелесе. После многолетних исследований на животных, в 1990-х годах появились первые

устройства для человека. Первоначально они применялись для восстановления утраченных функций у людей с неврологическими нарушениями, однако сегодня спектр их применения значительно расширился.

Нейрочипы

Нейрочипы представляют собой специализированные микроэлектронные устройства, разработанные для интеграции с нервной системой или для имитации функций нейронов. Они могут быть реализованы как в виде имплантируемых в мозг устройств (нейроимплантатов), так и в виде внешних микропроцессоров с нейроморфной архитектурой.

Одним из значимых достижений в этой области стало создание в 2004 году искусственного кремниевого чипа – аналога гиппокампа, разработанного в Университете Южной Калифорнии. Этот чип был способен имитировать работу гиппокампа – структуры мозга, отвечающей за формирование памяти.

В настоящее время активно развивается направление "wetware" - соединение биологических нейронов с электронными компонентами. Используя современные нанотехнологии, в частности углеродные нанотрубки, исследователи создают интерфейсы между живыми нейронами и электронными устройствами [1].

Современное состояние нейрокомпьютерных технологий. Достижения в области нейрокомпьютеров

На сегодняшний день нейрокомпьютеры активно применяются для решения широкого спектра задач, включая:

- Управление в реальном времени сложными техническими системами;
- Распознавание образов (изображений, речи, текста);
- Прогнозирование временных рядов (финансовые рынки, погода);
- Оптимизация и принятие решений;
- Обработка сигналов при наличии шумов.

Современные нейрокомпьютеры могут быть реализованы как с использованием специализированных нейрочипов, так и на базе обычных компьютеров с программной эмуляцией нейронных сетей. Примерами аппаратных реализаций нейрокомпьютеров являются нейрокомпьютер Synapse (Siemens, Германия) и российский процессор NeuroMatrix.

Важным направлением развития нейрокомпьютеров является создание нейроморфных систем, которые в большей степени имитируют работу настоящего мозга. Такие системы используют

импульсные нейронные сети и механизмы спайковой пластичности, что позволяет им эффективнее решать определенные классы задач при значительно меньшем энергопотреблении по сравнению с традиционными вычислительными системами.

Прогресс в разработке нейроинтерфейсов

В 2020-х годах нейроинтерфейсы прошли значительный путь развития. Среди ключевых достижений можно выделить:

1. Улучшение разрешающей способности и точности неинвазивных методов регистрации мозговой активности, таких как электроэнцефалография (ЭЭГ) и функциональная магнитно-резонансная томография (фМРТ).

2. Разработка малоинвазивных нейроинтерфейсов с высоким пространственным и временным разрешением.

3. Создание двунаправленных нейроинтерфейсов, способных не только считывать сигналы мозга, но и стимулировать нейроны для передачи сенсорной информации.

4. Разработка алгоритмов декодирования сигналов мозга в режиме реального времени с использованием методов машинного обучения [5].

Современные нейроинтерфейсы находят применение в различных областях:

- Медицинская реабилитация (управление протезами, восстановление утраченных функций);
- Обеспечение коммуникации для людей с тяжелыми двигательными нарушениями;
- Мониторинг и модуляция мозговой активности для лечения неврологических и психических расстройств;
- Научные исследования функций мозга;
- Взаимодействие с компьютерными системами и виртуальной реальностью.

Инновации в сфере нейрочипов

Последние годы характеризуются значительным прогрессом в области нейрочипов. Среди ключевых инноваций можно выделить:

1. Разработка нейрочипа Neuralink, который представляет собой беспроводное устройство размером с монету, имплантируемое в череп. В марте 2024 года компания продемонстрировала, как пациент с повреждением спинного мозга смог играть в компьютерные игры, управляя курсором силой мысли.

2. Создание нейроморфных чипов, имитирующих архитектуру и принципы работы мозга. Примерами являются чип TrueNorth от IBM, Loihi от Intel и SpiNNaker от Манчестерского университета.

3. Развитие технологий биосинтетических материалов, которые не отторгаются тканями мозга и могут служить интерфейсом между нейронами и электронными компонентами.

4. Разработка нейрочипов для лечения неврологических расстройств, таких как болезнь Паркинсона, эпилепсия и хронические боли [4].

Современные нейрочипы становятся все более компактными, энергоэффективными и функциональными. Они способны регистрировать активность тысяч нейронов одновременно и передавать данные беспроводным способом, что открывает новые возможности для их применения.

Практическое применение нейротехнологий

1. Медицинские приложения

Одной из наиболее перспективных областей применения нейротехнологий является медицина. Среди ключевых медицинских приложений можно выделить:

- Нейропротезирование – создание и имплантация искусственных устройств для восстановления нарушенных функций нервной системы. Наиболее распространенным примером является кохлеарный имплантат, который используется более чем 100 000 пациентов по всему миру. Также активно разрабатываются зрительные протезы, такие как бионическая система зрения Genparris.

- Нейрореабилитация – использование нейроинтерфейсов для восстановления двигательных функций у пациентов после инсульта или с травмами спинного мозга. Экзоскелеты и протезы, управляемые сигналами мозга, позволяют парализованным пациентам восстановить возможность двигаться.

- Нейромодуляция – применение электрической стимуляции различных структур мозга для лечения неврологических и психических расстройств. Например, глубокая стимуляция мозга применяется для лечения болезни Паркинсона, эпилепсии и обсессивно-компульсивного расстройства [1].

- Диагностика и мониторинг состояния мозга – использование нейроинтерфейсов для раннего выявления и мониторинга прогрессирования нейродегенеративных заболеваний.

2. Применение в исследованиях мозга

Нейротехнологии предоставляют уникальные инструменты для изучения функций мозга:

- Регистрация активности отдельных нейронов и нейронных ансамблей в режиме реального времени, что позволяет лучше понять принципы кодирования и обработки информации в мозге.

- Исследование функциональной связности различных областей мозга и ее изменений при выполнении когнитивных задач.
- Картирование мозга и создание его функциональных и структурных карт с высоким разрешением.
- Моделирование нейронных сетей и проверка теоретических моделей работы мозга.

3. Расширение человеческих возможностей

Нейротехнологии имеют потенциал не только для восстановления утраченных функций, но и для расширения естественных возможностей человека:

- Улучшение когнитивных функций – использование нейростимуляции для улучшения памяти, внимания и скорости обработки информации.
- Создание прямых интерфейсов с компьютерными системами, позволяющих управлять техникой "силой мысли".
- Разработка экзокортекса – внешней системы обработки информации, которая работает в тесной связи с мозгом и расширяет его вычислительные возможности.
- Создание новых каналов коммуникации, включая прямой обмен информацией между мозгом и компьютером или между мозгом и мозгом.

4. Искусственный интеллект и нейроморфные системы

Нейрокомпьютерные технологии играют важную роль в развитии искусственного интеллекта:

- Создание нейроморфных вычислительных систем, которые используют принципы работы мозга для эффективного решения задач обработки данных и принятия решений.
- Разработка новых алгоритмов машинного обучения, основанных на принципах функционирования биологических нейронных сетей.
- Создание гибридных систем, объединяющих возможности биологического и искусственного интеллекта.
- Развитие методов эмуляции отдельных функций мозга и создание когнитивных архитектур, способных моделировать высокоуровневые когнитивные процессы.

Перспективы развития нейрокомпьютерных технологий

Анализ текущих исследований и разработок позволяет выделить следующие тенденции и перспективы развития нейрокомпьютерных технологий на ближайшие десятилетия:

1. Миниатюризация и повышение функциональности нейроинтерфейсов. Ожидается создание полностью имплантируемых

беспроводных устройств с тысячами или даже миллионами каналов регистрации нейронной активности.

2. Развитие неинвазивных технологий считывания и модуляции мозговой активности с высоким пространственным и временным разрешением.

3. Совершенствование алгоритмов декодирования сигналов мозга с использованием методов глубокого обучения и искусственного интеллекта.

4. Создание нейроморфных вычислительных систем, способных эффективно решать когнитивные задачи при минимальном энергопотреблении.

5. Разработка биогибридных систем, объединяющих биологические нейроны и электронные компоненты в единые функциональные сети.

6. Создание более совершенных нейропротезов, обеспечивающих естественное управление и сенсорную обратную связь.

7. Развитие технологий для неинвазивной стимуляции глубоких структур мозга с высокой точностью.

Развитие нейрокомпьютерных технологий сталкивается с рядом серьезных вызовов:

1. Биологические ограничения [3]:

– Иммунный ответ организма на имплантаты и долгосрочная стабильность имплантированных устройств;

– Ограниченное понимание механизмов кодирования информации в мозге;

– Сложность интерпретации сигналов мозга на уровне намерений и абстрактных мыслей.

2. Технические ограничения:

– Необходимость разработки безопасных и долговечных материалов для имплантатов;

– Проблемы энергообеспечения имплантируемых устройств;

– Ограничения в передаче большого объема данных через ткани организма.

3. Алгоритмические вызовы:

– Разработка эффективных алгоритмов для декодирования и интерпретации нейронных сигналов;

– Создание адаптивных систем, способных подстраиваться под изменения в нейронной активности;

– Обеспечение надежности и отказоустойчивости нейрокомпьютерных систем.

Развитие нейрокомпьютерных технологий поднимает ряд важных этических и социальных вопросов:

- Вопросы приватности и безопасности данных мозга;
- Проблемы идентичности и автономии;
- Социальное неравенство и доступность;
- Правовое регулирование (определение границ допустимого использования нейроинтерфейсов; защита прав людей, использующих нейротехнологии).

В заключение, хотелось бы отметить, что нейрокомпьютерные технологии имеют потенциал для революционных изменений в медицине, когнитивных исследованиях, информационных технологиях и других областях. Они способны не только помочь людям с неврологическими нарушениями, но и расширить возможности здоровых людей, создав новые формы взаимодействия человека с технологиями и между людьми.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. – Новосибирск: Наука, 2022. – 220 с.
2. Дунин-Барковский В.Л. Нейрокибернетика,нейроинформатика, нейрокомпьютеры // Нейроинформатика / А.Н. Горбань, В.Л. Дунин-Барковский, А.Н. Кирдин и др. – Новосибирск: Наука, 2023. – 450 с.
3. Маск И. НейрочипNeuralink: исследования и перспективы // Технологии искусственного интеллекта. – 2024. – №2.
4. Галушкин А.И., Савельев А.В. Нейрокомпьютеры в изобретениях // Нейрокомпьютеры: разработка, применение. – 2023. – № 11.
5. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры. – 3-е изд. – М.: МГТУ им. Н.Э. Баумана, 2024. – 400 с.

УДК 004.021

ВЕНЧИКОВ Д.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

СОЗДАНИЕ АЛГОРИТМА ДЛЯ ПОИСКА МЕСТА ВСТРЕЧИ ПОПУТЧИКА И ВОДИТЕЛЯ

Рассматривается алгоритм, позволяющий находить оптимальное место встречи попутчика и водителя в рамках приложения карпулинга.

Современные сервисы совместных поездок сталкиваются с проблемой эффективного выбора точек встречи водителей и пассажиров. Оптимальное место встречи должно учитывать множество факторов: временные затраты всех участников, безопасность локации, доступность парковки и динамические условия движения. В данной работе предлагается гибридный алгоритм, сочетающий методы геопространственного анализа и мультикритериальной оптимизации для решения этой задачи.

Для выбора наиболее ближайших точек из доступных к попутчику можно использовать кластеризацию – принять точку за центр поиска и наращивать кластер возможных мест встречи попутчика и водителя, после чего использовать полученные точки кластера для сортировки по целевой функции.

Для реализации задачи построим целевую функцию для оценки точек:

$$f = w_1 + w_2 * (1 - S) + w_3(1 - P)$$

где T – суммарное время подъезда (нормализованное), S – показатель безопасности (0-1), P – вероятность наличия парковки (0-1), w_i – динамически корректируемые веса.

Однако настроить веса в функции недостаточно, необходимо подбирать наиболее удачную конфигурацию и для этого был выбран эволюционный алгоритм в связке с А/Б тестированием. Такой подход позволит приложению в runtime регулировать веса функции, что сделает алгоритм подбора места встречи попутчиков еще более точным.

Адаптация к условиям:

- Ночное время: увеличение веса безопасности ($w_2 \times 1.5$)
- Погодные условия: корректировка весов на основе данных API погоды

- Час пик: приоритет временным показателям ($w_1 \times 1.2$)

Механизм А/В тестирования:

- Постоянная генерация вариантов с небольшими мутациями коэффициентов

- Автоматический отбор наиболее эффективных конфигураций

Реализация алгоритма

Поиск оптимальной точки встречи:

а) Кластеризация DBSCAN (C#):

```
public class GeoCluster {
    public List<GeoPoint> Points { get; set; }
}

public class GeoService {
    public
```

```
List<GeoCluster> FindClusters(List<GeoPoint> points,
double epsilon = 200, int minPoints = 3) {
    List<GeoCluster> clusters = new
List<GeoCluster>();
    HashSet<GeoPoint> visited = new
HashSet<GeoPoint>();
    foreach (var point in points) {
        if (visited.Contains(point)) continue;
        var neighbors = points.Where(p
=>CalculateDistance(p, point) <= epsilon).ToList();
        if (neighbors.Count < minPoints)
continue;
        var cluster = new GeoCluster { Points =
new List<GeoPoint>() };
        ExpandCluster(point, neighbors, visited, cluster,
epsilon, minPoints);
        clusters.Add(cluster);
    }
    return clusters;
}
private void ExpandCluster(GeoPoint point,
List<GeoPoint> neighbors, HashSet<GeoPoint> visited,
GeoCluster cluster, double epsilon, int minPoints)
{
    cluster.Points.Add(point);
    visited.Add(point);
    foreach (var neighbor in
neighbors.ToList()) {
        if (!visited.Contains(neighbor)) {
            visited.Add(neighbor);
            var subNeighbors = FindClusters(new
List<GeoPoint> { neighbor }, epsilon, minPoints)
                .SelectMany(c => c.Points)
                .ToList();
            if (subNeighbors.Count >= minPoints)
            {
                neighbors.AddRange(subNeighbors);
            }
            if (!cluster.Points.Contains(neighbor))
            {
                cluster.Points.Add(neighbor);
            }
        }
    }
}
```

b) Мультикритериальная оптимизация (TOPSIS):

```

public class TopsisOptimizer {
    public GeoPoint FindBestPoint(List<GeoPoint>
candidates, DynamicWeights weights) {
        var scores = candidates.Select(p => new {
            Point = p,
            Score = weights.W1 * (p.Time / MaxTime)
                + weights.W2 * (1 - p.Safety)
                + weights.W3 * (1 - p.Parking)
        }).OrderBy(x => x.Score).ToList();
        return scores.First().Point;
    }
}

```

Динамическая коррекция весов с A/B тестированием

а) Генерация мутаций:

```

public class WeightMutator {
    private Random _rnd = new Random();
    public DynamicWeights
Mutate(DynamicWeights baseWeights) {
        return new DynamicWeights {
            W1 = Math.Max(0, baseWeights.W1 +
_rnd.NextDouble(-0.1, 0.1)),
            W2 = Math.Max(0, baseWeights.W2 +
_rnd.NextDouble(-0.1, 0.1)),
            W3 = Math.Max(0, baseWeights.W3 +
_rnd.NextDouble(-0.1, 0.1))
        };
    }
}

```

б) Эволюционный A/B тест:

```

public class ABTestManager {
    private List<ABTestVariant> _variants = new
List<ABTestVariant>();
    public void RunEvolutionCycle() {
        // Удаление худших 10%
        var badVariants = _variants.OrderBy(v
=> v.Score).Take(_variants.Count / 10);
        _variants.RemoveAll(v
=> badVariants.Contains(v));
        // Репликация лучших 10%
        var topVariants =
_variants.OrderByDescending(v
=> v.Score).Take(_variants.Count / 10);
        foreach (var variant in topVariants) {
            for (int i = 0; i < 5; i++) {
                var mutated =
Mutate(variant.Weights);
            }
        }
    }
}

```



```

        _variants.Add(new ABTestVariant {
Weights = mutated });
    }
}
}

```

Оценка статистической значимости

а) Расчет t-теста:

```

public class TTest {
    public (double TValue, double PValue)
Calculate(double[] groupA, double[] groupB) {
    double meanA = groupA.Average();
    double meanB = groupB.Average();
    double varA = CalculateVariance(groupA);
    double varB = CalculateVariance(groupB);
    double t = (meanA - meanB) /
Math.Sqrt((varA / groupA.Length) + (varB /
groupB.Length));
    double df = (Math.Pow(varA / groupA.Length
+ varB / groupB.Length, 2))
/ (Math.Pow(varA, 2) /
(Math.Pow(groupA.Length, 2) * (groupA.Length - 1))
+ (Math.Pow(varB, 2) /
(Math.Pow(groupB.Length, 2) * (groupB.Length - 1))));
    // Использование таблиц Стьюдента для p-value (упрощенно)
    double pValue = 2 * (1 -
TDistribution.CDF(Math.Abs(t), df));
    return (t, pValue);
}
}

```

б) Интеграция с A/B тестами:

```

public class ExperimentAnalyzer {
    public bool
IsVariantSignificant(ABTestVariant variantA,
ABTestVariant variantB) {
    var tTest = new TTest();
    var result =
tTest.Calculate(variantA.Metrics.TimeData,
variantB.Metrics.TimeData);
    return result.PValue < 0.05;
}
}

```

Полный рабочий пример

```

public class CarpoolService {
    private GeoService _geoService = new
GeoService();
}

```

```

        private TopsisOptimizer _optimizer = new
TopsisOptimizer();
        private ABTestManager _abTestManager = new
ABTestManager();
        public GeoPointFindBestMeetingPoint(GeoPoint
driver, List<GeoPoint> passengers) {
            // Кластеризация
            var clusters =
            _geoService.FindClusters(passengers);
            // Динамические веса из A/B теста
            var weights = _abTestManager.GetCurrentWeights();
            // Выбор точки
            return
            _optimizer.FindBestPoint(clusters.First().Points,
weights);
        }
    }
    Запуск A/B теста:
    public void RunFullExperiment() {
        var abTest = new ABTestManager();
        var analyzer = new ExperimentAnalyzer();
        abTest.RunEvolutionCycle();
        ...обработка поездки
        var bestVariant = abTest.GetBestVariant();
        bool isSignificant =
analyzer.IsVariantSignificant(abTest.ControlGroup,bestVa
riant);
        if (isSignificant) {
            abTest.UpdateProductionWeights(bestVariant.Weights)
;
        }
    }

```

Преимущества подхода:

- Комплексный учет факторов,
- Временные затраты,
- Безопасность и комфорт,
- Динамические дорожные условия,
- Адаптивность,
- Автоматическая подстройка под изменение условий,
- Возможность интеграции новых параметров,
- Масштабируемость,
- Эффективная работа как для малых городов, так и для мегаполисов,
- Поддержка различных типов транспортных средств.

Предложенный гибридный алгоритм демонстрирует высокую эффективность в решении задачи выбора оптимальных точек встречи для сервисов карпулинга. Комбинация геопространственного анализа с системой динамической оптимизации позволяет учитывать как статические параметры местности, так и изменяющиеся дорожные условия. Дальнейшие исследования могут быть направлены на интеграцию алгоритма с системами машинного обучения для более точного прогнозирования оптимальных точек.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Скворцов С.В., Орехов В.В. Данные и алгоритмы в программном обеспечении САПР: учеб. пособие. – Рязань: РГРТУ, 2009. – 88 с.
2. Многокритериальная оптимизация задач TOPSIS// datafinder.ru [Электронный ресурс]. – URL: <https://cyberleninka.ru/article/n/mnogokriterialnaya-optimizatsiya> (дата обращения: 16.04.2025)
3. Кластеризация DBSCAN в машинном обучении // Хабр [Электронный ресурс]. – URL: <https://datafinder.ru/products/klasterizaciya-dbscan-v-mashinnom-obuchenii> (дата обращения: 16.04.2025)

УДК 004.031.42

ГАРМАШ М.Е.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ

В статье представлен подход к проектированию информационной системы с использованием микросервисной архитектуры на примере стриминговой платформы музыкального контента.

Введение

Современные информационные системы предъявляют высокие требования к масштабируемости, гибкости и скорости внедрения новых функций. Особенно это актуально в индустрии цифрового контента, где требуется быстрая реакция на изменения пользовательских предпочтений и рыночной конъюнктуры. Одним из наиболее эффективных архитектурных подходов для решения этих задач является микросервисная архитектура.

Актуальность. На фоне быстрого роста числа пользователей стриминговых сервисов возникает необходимость в архитектуре, способной эффективно масштабироваться и легко адаптироваться к изменениям требований пользователей и бизнеса. Микросервисная архитектура позволяет не только быстро и эффективно масштабировать ресурсы, но и оперативно внедрять новый функционал, обеспечивая высокую доступность и отказоустойчивость.

Научная новизна. В отличие от традиционных подходов, предложенный в статье проект представляет собой комплексное решение, в котором детально проработаны не только базовые компоненты и их взаимодействия, но и конкретные методики интеграции различных протоколов (REST, gRPC, Kafka) для эффективной обработки и передачи данных. Дополнительно проведена оптимизация системы на уровне базы данных и межсервисного взаимодействия, что позволяет существенно повысить общую производительность системы.

Цель работы. Создание эффективной и масштабируемой информационной системы для стриминговой музыкальной платформы с использованием микросервисной архитектуры, оптимизированной для работы в условиях высокого пользовательского трафика и быстрого изменения бизнес-требований.

Обоснование проектирования на примере музыкальной платформы

Проектирование информационной системы осуществляется на примере музыкальной стриминговой платформы. Рассматривается ее часть, отвечающая за бизнес-логику системы, включая:

- Управление пользователями;
- Администрирование подписок;
- Каталог контента;
- Воспроизведение треков;
- Генерацию рекомендаций;
- Отправку уведомлений.

В данной работе исключены компоненты, связанные с обработкой платежей, рекламой и аналитикой, чтобы сосредоточиться на ключевых аспектах бизнес-логики платформы.

Бизнес-требования

1. Регистрация и авторизация пользователей. Система должна поддерживать регистрацию пользователей, управление их учетными записями и аутентификацию.

2. Управление подписками. Пользователи могут оформлять подписки с различными условиями (например, доступ к премиум-контенту).

3. Каталогизация контента. Система должна обеспечивать поиск, фильтрацию и доступ к трекам, альбомам и плейлистам.

4. Воспроизведение. Возможность воспроизведения музыкального контента с низкой задержкой.

5. Рекомендации. Персонализированные рекомендации на основе истории прослушиваний.

6. Уведомления. Отправка уведомлений о новых треках, изменении статуса подписки.

Выделенные сущности и логическая модель базы данных

Для реализации системы были выделены следующие основные сущности:

- User (Пользователь): хранит информацию о пользователях.
- Role (Роль): определяет уровень доступа пользователя.
- Subscription (Подписка): включает тип подписки, цену и срок действия.
- UserCard (Платежная карта): содержит информацию о привязанных к аккаунту картах.
- Track (Трек): содержит метаинформацию о музыкальных произведениях.
- Album (Альбом): объединяет треки в коллекции.
- Playlist (Плейлист): настраиваемые списки треков.
- Artist (Исполнитель): отдельная сущность, отражающая авторов/исполнителей треков и альбомов, с возможностью учета количества слушателей.
- Genre (Жанр): определяет категорию музыки.
- TypeMedia (Тип Медиа): указывает, является ли трек аудио или видео.

В результате чего была разработана ER-диаграмма базы данных, представленная на рисунке 1:

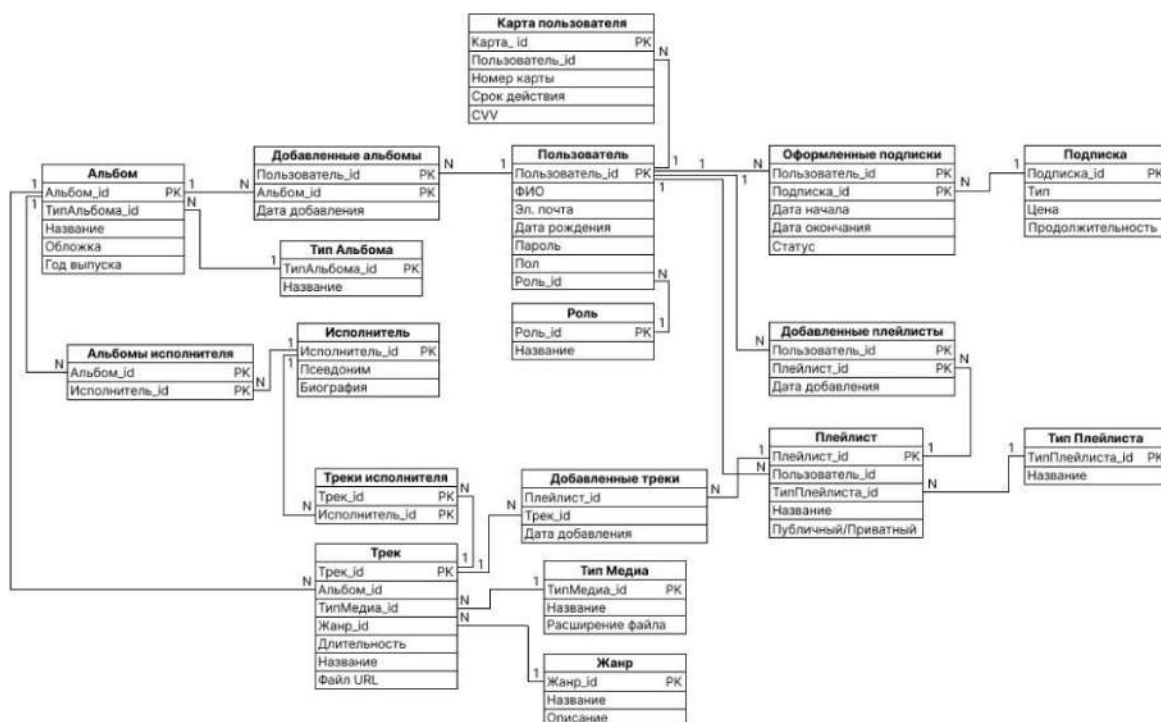


Рисунок 1 – ER -диаграмма

Архитектура системы

Система основана на микросервисной архитектуре, в которой каждый сервис отвечает за строго определенный набор задач, обеспечивая принцип единой ответственности:

- UserService (Сервис пользователей) – отвечает за управление пользователями и их данными. Основные функции включают регистрацию, аутентификацию и управление профилем пользователя, ролями и правами доступа.
- SubscriptionService (Сервис подписок) – занимается обработкой подписок пользователей, включая управление их статусами, проверку срока действия, а также условий предоставления подписки.
- CatalogService (Сервис каталогов) – управляет музыкальным контентом платформы, включая треки, альбомы, плейлисты и артистов. Обеспечивает поиск и фильтрацию контента для пользователей.
- PlaybackService (Сервис воспроизведения) – отвечает за воспроизведение треков.
- UserPreferencesService (Сервис пользовательских настроек) – отвечает за хранение и управление пользовательскими предпочтениями.

- RecommendationService (Сервис рекомендаций) – генерирует персонализированные рекомендации для пользователей на основе их предпочтений, истории прослушиваний и других факторов.

- NotificationService (Сервис уведомлений) – отвечает за отправку уведомлений пользователям через различные каналы.

Такое разделение на микросервисы было выбрано исходя из принципа единой ответственности, что значительно упрощает поддержку, масштабирование и мониторинг.

Взаимодействие микросервисов в системе

1. Клиент-серверные коммуникации

1.1. Взаимодействие между клиентскими приложениями (веб-приложения, мобильные клиенты) и серверными микросервисами реализовано с использованием архитектурного подхода REST (RepresentationalStateTransfer). Данный подход был выбран по следующим причинам:

- Простота реализации и поддержки
- Универсальность и совместимость
- Лёгкость масштабирования

Использование REST делает систему доступной и удобной для интеграции с широким спектром устройств и клиентских приложений.

2. Сервер-серверные коммуникации

2.1 *Синхронное взаимодействие* микросервисов реализовано с помощью RPC (RemoteProcedureCall), а именно его современного и высокопроизводительного варианта gRPC. Причинами выбора gRPC являются:

- Высокая производительность благодаря бинарным протоколам
- Строгие контракты взаимодействия
- Минимизация объёма передаваемых данных

Использование gRPC оптимально подходит для взаимодействий, где важна низкая задержка и высокая производительность (например, PlaybackService и RecommendationService).

2.2. *Асинхронное взаимодействие* реализуется через брокер сообщений (Kafka). Асинхронное взаимодействие микросервисов организовано с помощью брокера сообщений ApacheKafka. Kafka была выбрана по следующим причинам:

- Независимость и слабая связанность сервисов
- Отказоустойчивость и надежность
- Масштабируемость

Использование Kafka эффективно в ситуациях, когда необходимо гарантировать доставку информации нескольким независимым сервисам (например, NotificationService,

SubscriptionService), минимизируя связанность и зависимость между ними.

Комбинирование gRPC для синхронных, высокоскоростных и критичных к задержкам взаимодействий и Kafka для асинхронного, независимого и надежного обмена сообщениями – позволило существенно улучшить эффективность и надёжность общей архитектуры системы.

Основные направления взаимодействия микросервисов

1. UserService–SubscriptionService

Тип взаимодействия: gRPC

Назначение: при регистрации или авторизации пользователя может потребоваться проверка статуса его подписки.

Обоснование: gRPC используется для синхронной и быстрой проверки данных, например, когда пользователь пытается получить доступ к контенту, который доступен только по подписке.

2. Catalog Service – Playback Service

Тип взаимодействия: REST + gRPC

Назначение: PlaybackService получает информацию о треке из CatalogService, включая ссылку на воспроизведение (playback URL), ID медиафайла, длительность и тип.

Обоснование: для первичной загрузки данных используется REST. Для массовой передачи метаданных или во время воспроизведения – gRPC.

3. PlaybackService–UserService

Тип взаимодействия: gRPC

Назначение: подтверждение прав пользователя на воспроизведение (например, наличие активной подписки).

Обоснование: важно иметь низкие задержки и быстрый отклик при старте трека.

4. Playback Service – Recommendation Service

Тип взаимодействия: Kafka

Назначение: отправка информации о прослушанных треках.

Обоснование: нет необходимости обрабатывать рекомендации в реальном времени – они рассчитываются в фоне, на основе истории.

5. User Preferences Service – Catalog Service

Тип взаимодействия: REST

Назначение: получение данных о любимых альбомах, исполнителях, плейлистах и треках, сохранённых пользователем.

Обоснование: предпочтения пользователя часто используются в UI, где REST предоставляет гибкость и простоту.

6. Recommendation Service – Catalog Service

Тип взаимодействия: REST

Назначение: получение метаданных для составления рекомендаций (жанры, исполнители, популярность).

Обоснование: Рекомендации нуждаются в обогащении информации.

7. Subscription Service – Notification Service

Тип взаимодействия: Kafka

Назначение: отправка напоминаний о скором окончании подписки, изменении тарифа и т.д.

Обоснование: асинхронная модель позволяет уведомлять множество пользователей без перегрузки основного потока.

8. User Service – Notification Service

Тип взаимодействия: Kafka

Назначение: уведомления при регистрации, смене пароля и других действиях.

Безопасность и отказоустойчивость системы

Безопасность обеспечена через JWT-токены и OAuth2 для контроля доступа к сервисам. Отказоустойчивость реализована использованием балансировщиков нагрузки и механизма CircuitBreaker, который позволяет системе устойчиво реагировать на сбои и быстро восстанавливаться после них. Дополнительно используется кэширование данных с помощью Redis, сокращающее время ответа системы.

Методики тестирования и мониторинга

В системе реализованы юнит-тесты, интеграционные и нагрузочные тесты, обеспечивающие её надежность и стабильность. Для мониторинга используется Prometheus и Grafana, что позволяет оперативно отслеживать состояние системы и реагировать на возникающие проблемы.

Разработка и развертывание системы

Разработка и доставка системы организованы на основе CI/CD-подхода, использующего Docker и Kubernetes. Это обеспечивает быстрое развертывание, масштабирование и поддержку системы.

Схема архитектуры представлена на рисунке 2.

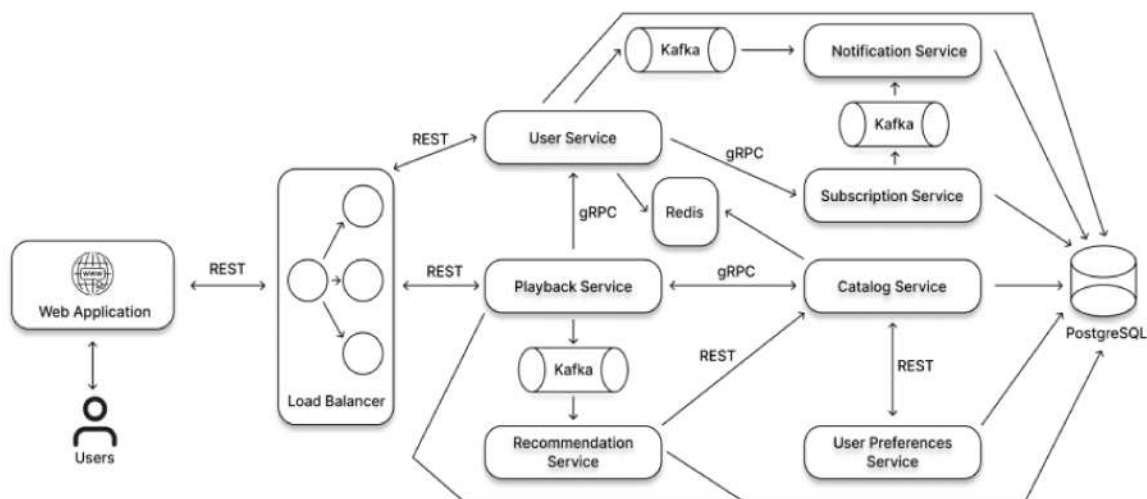


Рисунок 2 – Схема архитектуры

Преимущества:

- Гибкость: возможность добавления нового функционала без влияния на существующие сервисы.
- Масштабируемость: масштабирование отдельных сервисов в зависимости от нагрузки.
- Отказоустойчивость: сбои в одном сервисе не влияют на работу остальных.
- Ускорение разработки: независимость микросервисов позволяет разным командам работать над отдельными частями системы.

- Сложность взаимодействия: распределенные транзакции и управление согласованностью данных требуют применения сложных паттернов.

- Увеличение накладных расходов: необходимость настройки коммуникаций между сервисами.
- Требования к инфраструктуре: высокие требования к оркестрации сервисов и мониторингу.

В рамках данной работы была спроектирована информационная система на основе микросервисной архитектуры. Основные результаты включают:

- Выделение ключевых бизнес-требований.
- Разработку ER-диаграммы и логической структуры базы данных.

- Создание архитектуры системы и описание взаимодействия между микросервисами.

Предложенная система демонстрирует преимущества микросервисного подхода, обеспечивая гибкость, масштабируемость и отказоустойчивость. Тем не менее, использование данной архитектуры требует детального планирования взаимодействий и более высокой квалификации команды разработчиков.

Перспективы использования включают применение предложенного подхода в разработке других систем с высокими требованиями к надежности и производительности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. [Электронный ресурс] – URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата обращения: 01.11.2024).

2. Birman, K. (2012). Reliable Distributed Systems: Technologies, Web Services, and Applications. Springer. [Электронный ресурс] – URL: <https://books.google.ru/books/about/Reliable-Distributed-Systems.html> (дата обращения: 06.11.2024).

3. Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A Distributed Messaging System for Log Processing. Proceedings of the NetDB. [Электронный ресурс] – URL: <https://www.semanticscholar.org/paper/Kafka-A-Distributed-Messaging-System-for-Log-Kreps/> (дата обращения: 06.11.2024).

4. Tanenbaum, A. S., & Van Steen, M. (2016). Distributed Systems: Principles and Paradigms. Pearson. [Электронный ресурс] – URL: <https://github.com/seawish/books/blob/master/books/Distributed-systems-principles-and-paradigms> (дата обращения: 10.11.2024).

УДК 004.93.12

ГРАФКИНА К.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАСПОЗНАВАНИЕ НОМЕРНЫХ ЗНАКОВ ТРАНСПОРТНЫХ СРЕДСТВ

В статье рассматриваются вопросы построения и обучения нейронной сети для распознавания номерных знаков транспортных средств на изображениях, включая архитектуру модели, подготовку данных и специфику задач компьютерного зрения.

Распознавание номерных знаков автомобилей является важной задачей в сфере автоматизации парковок, контроля доступа, систем безопасности и умного города. Одним из наиболее эффективных подходов к решению данной задачи является использование сверточных нейронных сетей (CNN) и метода CTC (Connectionist Temporal Classification), позволяющего корректно обрабатывать последовательности символов переменной длины.

В рамках работы была реализована система, состоящая из двух ключевых компонентов:

- Генератор синтетических изображений номерных знаков с соответствующими метками (labels);
- Обучаемая нейронная сеть, способная распознавать номера на входных изображениях.

Подготовка данных. Для обучения нейронной сети необходимо большое количество размеченных данных. Однако получение большого объема реальных изображений с разметкой (номер + картинка) может быть затруднено. Поэтому на практике используется генерация синтетического dataset. С помощью Python и библиотеки PIL были автоматически сгенерированы изображения номерных знаков, соответствующих ГОСТ Р 50577-2018, с нанесёнными на них символами.

Каждое изображение имело стандартный размер (например, 220x60 пикселей), а к нему сопоставлялась строка из символов (буквы и цифры). Для генерации номеров использовался ограниченный алфавит, включающий кириллические буквы, визуально похожие на латиницу, и цифры, а также в коде применялся шрифт PTSans-Bold.ttf.

Принцип работы слоёв сверточной нейронной сети. Для того, чтобы применить математическую операцию свертки (фильтра) к изображению используется сверточный слой. Под сверткой понимается двумерная матрица коэффициентов. Фрагмент двумерного изображения является входом фильтра, а некоторое число – выходом (рисунок 1). Главным преимуществом применения данных фильтров заключается в том, что число на выходе тем больше, чем больше элемент изображения похож на применяемый к нему фильтр. Используя операцию свертки, возможно получить на выходе изображение, каждый пиксель которого будет соответствовать степени подобия кусочка изображения на фильтр. Таким образом, получается карта признаков.

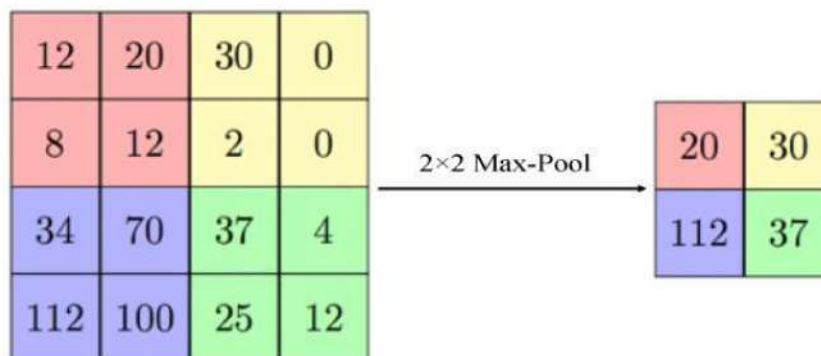


Рисунок 1 – Пример использования двумерной операции свертки для формирования

Для уменьшения размера изображения и увеличения степени инвариантности применяемых к нему фильтров используется субдискретизация. Так как наличие какого-либо признака на изображении гораздо важнее, чем точное знание его координат, то смыслом субдискретизации является выбор максимального нейрона из нескольких соседних. Далее карта признаков уменьшается, и этот нейрон принимается за ее последующий элемент. С помощью этой операции увеличивается инвариантность к масштабу входного изображения.

Благодаря тому, что СНС строится посредством чередования слоев свертки и субдискретизации, появляется возможность строить карты признаков из карт признаков и распознавать иерархии признаков. Концом работы СНС является преобразование карты признаков в вектор или скаляр, для чего и добавляется полносвязный слой на выходе.

Каждый нейрон в полносвязном слое является персептроном с нелинейной функцией активации. Для функции активации используется либо логистическая функция вида:

$$f(x) = \frac{1}{1 + e^{-x}}$$

либо гиперболический тангенс: $f(x) = A * \tanh(Bx)$, где $A, B = \text{const}$.

Каждый нейрон, соединяясь с выходами нейронов предыдущего слоя, суммирует свои входы, умножает на веса, добавляет порог и результат подает на вход функции активации. Выходом нейрона является данное значение. Сверточный слой (Conv2D) является основным блоком свёрточной нейронной сети. Он выполняет обработку изображения с помощью обучаемых фильтров (ядер свёртки), которые проходятся по изображению и формируют так называемые карты признаков. Каждое ядро свёртки применимо к

конкретному каналу входного изображения и выявляет специфические признаки, такие как края, углы и текстуры.

Преимущество свёрточных слоёв заключается в значительно меньшем числе параметров по сравнению с полносвязными слоями: например, при размере фильтра 3×3 и 6 выходных каналах в трёхцветном изображении понадобится всего 162 параметра ($3 \times 3 \times 3 \times 6$).

После свёрточного слоя обычно следует слой активации, чаще всего это функция ReLU (Rectified Linear Unit), определяемая как $f(x) = \max(0, x)$. Она позволяет сделать модель нелинейной, тем самым улучшая способность к обучению.

Далее используется слой субдискретизации (MaxPooling2D), выполняющий уплотнение признаков. Наиболее распространённым методом является пулинг с функцией максимума, при котором каждый непересекающийся блок (например, 2×2 пикселя) заменяется максимальным значением. Это уменьшает размерность данных, снижает вычислительные затраты и помогает бороться с переобучением.

В конце архитектуры применяется полносвязная нейросеть, которая агрегирует признаки и делает окончательное предсказание. Благодаря свёртке и пулингу, к моменту подачи на полносвязный слой изображение превращается в набор высокоуровневых признаков.

Для обучения свёрточных сетей применяются как методы обучения с учителем (например, обратное распространение ошибки), так и подходы без учителя (такие как обучение по фрагментам или автоассоциативные сети). Часто также используется метод исключения (dropout), помогающий бороться с переобучением путём случайного отключения нейронов в процессе обучения.

Модель была построена с использованием Keras и TensorFlow. В качестве базовой архитектуры была выбрана свёрточно-рекуррентная сеть, состоящая из следующих блоков:

Свёрточные слои (Conv2D) с последующим субдискретизированием (MaxPooling2D), отвечающие за выделение признаков изображения [2].

Слой Reshape, преобразующий выход CNN в формат последовательности.

Двунаправленные рекуррентные слои LSTM (Bidirectional LSTM), обеспечивающие обработку последовательности слева направо и справа налево.

Полносвязный слой (Dense) с softmax-активацией, прогнозирующий вероятность появления каждого символа в последовательности.

Специальный CTC-слой, рассчитывающий функцию потерь для последовательностей переменной длины, где известен только конечный результат, но не позиция каждого символа во входной последовательности.

Ниже представлена схема архитектуры модели, построенной по принципу CRNN (сверточно-рекуррентной нейросети, которая показана на рисунке 2). Она демонстрирует прохождение изображения через свёрточные слои, последующую обработку выходных признаков рекуррентными слоями и финальную классификацию символов с помощью полносвязных слоёв [1].

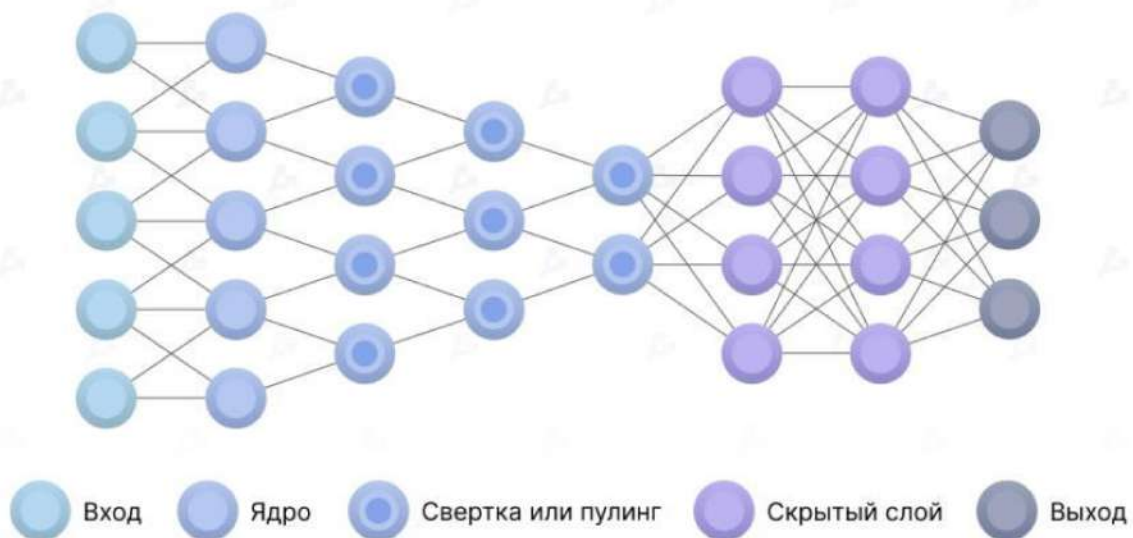


Рисунок 2 – Архитектура сверточно-рекуррентной нейронной сети (CRNN)

Обучение модели происходило на синтетическом dataset объёмом ~200 изображений (пример одного из генерированных изображений показан на рисунке 3). Из-за небольшого размера dataset для начала тестирования применялись упрощённые архитектурные параметры (меньше фильтров, меньше эпох). В дальнейшем возможно масштабирование модели на полноценный dataset и до обучения на изображениях машин.

M923CA05

Рисунок 3 – Пример изображения из dataset

Функция потерь – CTC Loss – позволяла модели предсказывать последовательность символов без необходимости вручную размечать

точные границы символов на изображении. Предполагается, что совпадение входных данных с целевыми является «много к одному», что ограничивает длину целевой последовательности таким образом, чтобы она была много меньше длины входных данных.

Оптимизатор – Adam, количество эпох – 10. Размер батча – 16. Обученная модель сохранялась в формате H5, и в дальнейшем может быть подключена к API-серверу для распознавания реальных номеров с камер.

Разработанная система распознавания номерных знаков интегрируется в более широкую интеллектуальную инфраструктуру учёта времени пребывания транспортных средств на парковке. При въезде на охраняемую территорию или парковочную зону камера фиксирует автомобиль, делается снимок номерного знака, и далее изображение автоматически отправляется в модуль нейросетевого распознавания.

После обучения модель разворачивается в виде полноценного веб-сервиса с использованием фреймворка *FastAPI*, одного из самых производительных и удобных инструментов для создания REST API на Python. Благодаря FastAPI, становится возможным организовать взаимодействие с моделью в реальном времени: клиентское приложение, в том числе внешняя система или мобильный frontend, может отправить HTTP-запрос типа *POST* на специальную конечную точку (endpoint) сервиса.

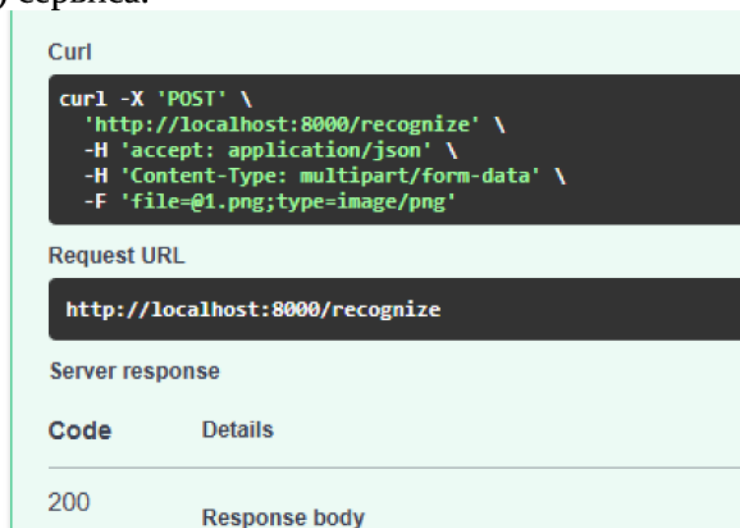


Рисунок 4 – Пример взаимодействия клиента с REST API FastAPI сервиса распознавания номерных знаков

На рисунке 4 представлен пример запроса, отправляемого на адрес `http://localhost:8000/recognize`. Он содержит заголовки Content-Type и Асепт, а также вложенный файл изображения номерного знака. После получения изображения, FastAPI обрабатывает его и передаёт

внутри модели для предсказания текста номера. В ответ сервис возвращает JSON-объект, содержащий строку с распознанным номером. Пример структуры ответа:

```
{  
  "plate_number": "A123BC77"  
}
```

Интеграция этого модуля в систему учёта позволяет реализовать автоматическое определение момента въезда автомобиля. В момент выезда, по сохранённому номеру, производится вычисление времени нахождения ТС на парковке. Далее, пользователю направляется уведомление через Telegram-бот со сгенерированной ссылкой на оплату – всё это без участия человека и без необходимости установки дополнительных устройств в автомобиле.

Данное решение масштабируемо и может быть дополнено новыми модулями:

Автоматическая оплата с привязкой банковской карты;

Хранение и аналитика истории посещений;

Интеграция с городскими платформами (например, "Умный город");

Поддержка мультикампусных парковочных пространств.

Таким образом, предложенная система демонстрирует эффективное использование компьютерного зрения и нейросетевых технологий в реальных прикладных задачах автоматизации городской инфраструктуры.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация PyTorch [Электронный ресурс]. – URL: <https://pytorch.org/docs/stable/nn.html> (дата обращения: 02.04.2025).

2. Официальная документация Keras [Электронный ресурс]. – URL: <https://keras.io/> (дата обращения: 02.04.2025).

3. ГОСТ Р 50577-2018. Государственный стандарт Российской Федерации. Знаки регистрационные транспортных средств.

УДК 004.415.22

ГРОШЕВ В.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

**ИНФОРМАЦИОННАЯ СИСТЕМА ДЛЯ ИНТЕРНЕТ-
МАГАЗИНА ТЕХНИКИ С КЛИЕНТ-СЕРВЕРНОЙ
АРХИТЕКТУРОЙ**

Рассматривается проектирование и реализация системы, включая разработку базы данных, клиентского приложения и интеграцию функций управления заказами, товарами и пользователями.

Современные интернет-магазины техники требуют комплексной автоматизации процессов управления товарами, клиентами и заказами для обеспечения конкурентоспособности в условиях растущего рынка электронной коммерции. Согласно исследованиям, к 2027 году объем онлайн-продаж электроники достигнет \$1,5 трлн, что делает критически важным внедрение систем, способных не только обрабатывать транзакции, но и анализировать данные для персонализации взаимодействия с пользователями. Однако многие существующие платформы, такие как готовые CMS-решения, зачастую ограничены в гибкости: они не позволяют адаптировать функционал под специфику техники (например, управление техническими характеристиками товаров, интеграция с поставщиками комплектующих) и не обеспечивают глубокую аналитику покупательского поведения. Кроме того, отсутствие встроенных механизмов прогнозирования спроса и управления скидками снижает эффективность маркетинговых стратегий. Целью данной работы стала разработка информационной системы для интернет-магазина техники, которая сочетает двухуровневую клиент-серверную архитектуру с расширенным функционалом: управление каталогом товаров (включая категоризацию по типам устройств и фильтрацию по параметрам), автоматизация процессов оформления заказов, реализация системы рекомендаций на основе истории покупок, а также интеграция модуля аналитики для администраторов. Для достижения этой цели была создана система, где серверная часть реализована на Microsoft SQL Server с использованием триггеров для поддержания целостности данных (например, автоматическое обновление остатков на складе при покупке) и хранимых процедур, а клиентское приложение разработано на C# с применением WindowsForms, обеспечивающего интуитивный интерфейс для пользователей и администраторов. Клиентская часть поддерживает функции добавления товаров в корзину, выбора способа оплаты (карта, наличные), отслеживания статуса заказа и взаимодействия со службой поддержки. В статье подробно описывается архитектура системы, включая ER-диаграмму базы данных, нормализованную до БКНФ, механизмы безопасности (валидация данных на стороне клиента, параметризованные SQL-запросы), а также результаты тестирования ключевых сценариев работы приложения. Отдельное внимание уделено перспективам развития системы, таким как внедрение рекомендательных алгоритмов

машинного обучения для персонализации ассортимента и создание мобильной версии приложения для увеличения охвата аудитории.

Описание основного функционала приложения

Разработанная информационная система для интернет-магазина техники обеспечивает взаимодействие пользователей с каталогом товаров, оформление заказов и управление бизнес-процессами через интуитивный интерфейс. При запуске клиентского приложения пользователь попадает на главный экран (рисунок 1), где представлены кнопки для входа, регистрации или выхода из приложения. Для доступа к полному функционалу система требует авторизации: новые пользователи проходят регистрацию, указывая электронную почту и пароль, а существующие – вводят учетные данные для входа. После успешной аутентификации открывается доступ к персональным данным, истории заказов и возможности редактирования профиля.

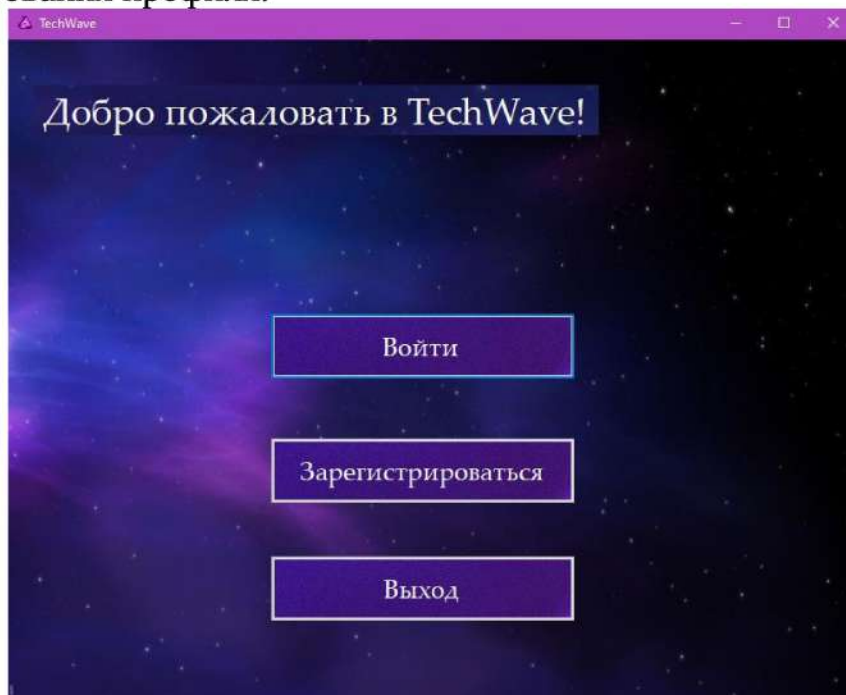


Рисунок 1 – Главный экран приложения

После успешного прохождения процедуры аутентификации (ввода логина и пароля) или регистрации (создания новой учетной записи с валидацией email и телефона) пользователь перенаправляется на главный экран системы (рисунок 2). Интерфейс экрана структурирован для оптимизации навигации: в верхней части расположен каталог товаров. В левой панели размещены элементы управления: интерактивная корзина с отображением текущего количества товаров и их общей стоимости, раздел «Заказы» с кратким описанием, а также кнопки быстрого доступа к функциональным

модулям – «Отзывы и оценки», «Обращение в техподдержку» и «Аккаунт». Последний предоставляет доступ к редактированию персональных данных. Все элементы интерфейса взаимодействуют с серверной частью, обеспечивая мгновенное обновление данных (например, синхронизация корзины при добавлении товара) и минимизацию задержек за счет кэширования часто запрашиваемой информации.

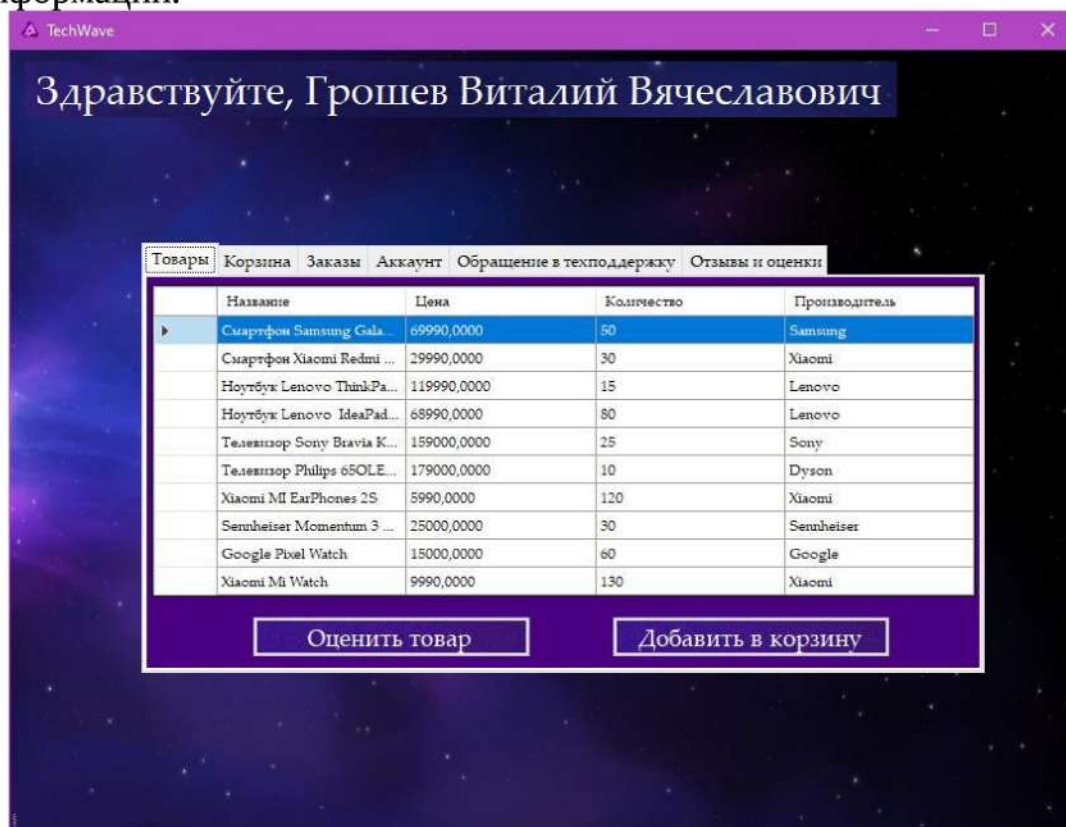


Рисунок 2 – Интерфейс зарегистрированного пользователя

При открытии вкладки «Товары» система автоматически загружает актуальный список доступных товаров из базы данных. Пользователь видит карточки товаров с названиями, ценами, количеством и производителями. Каждый товар можно добавить в корзину – для этого достаточно выбрать товар и нажать кнопку «Добавить в корзину», после чего система проверит доступное количество на складе и обновит корзину, отобразив новую общую стоимость (рисунок 3). Если товар закончился, пользователь получит уведомление о невозможности добавления. Рядом с каждым товаром есть кнопка «Оценить товар», которая открывает форму для оценки (от 1 до 5 звезд) и написания комментария (рисунок 4). После отправки отзыва система сохраняет его и автоматически обновляет средний рейтинг товара, чтобы другие покупатели могли видеть актуальную оценку.

Название	Количество	Цена
Ноутбук Lenovo ThinkPad X1 Carb...	2	119990,0000

Итого:

ОбщееКоличество	ОбщаяСтоимость
2	239980,0000

Рисунок 3 – Корзина пользователя

Отзыв о товаре: Ноутбук Lenovo ThinkPad X1 Carbon

Классно!

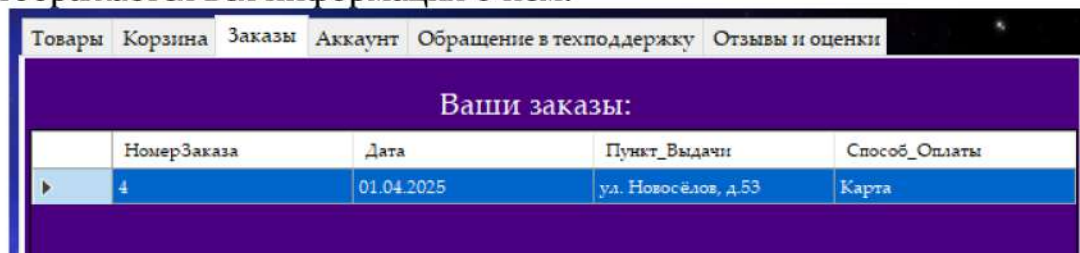
Выберите оценку 3

Отмена Отправить

Рисунок 4 – Форма оценки товара

При переходе в корзину пользователь видит список выбранных товаров с указанием количества, цены за единицу и общей суммы заказа. Каждая позиция в списке сопровождается кнопкой «Удалить товар» – при ее нажатии товар мгновенно исчезает из корзины, а система автоматически пересчитывает итоговую стоимость. Если пользователь случайно удалил нужный товар, его можно снова добавить через каталог. Кнопка «Перейти к оформлению» запускает процесс подтверждения заказа. Сначала система проверяет, доступны ли выбранные товары на складе в нужном количестве. Если все в порядке, пользователь выбирает удобный способ оплаты (например, картой или наличными при получении) и указывает адрес доставки или пункт самовывоза. После подтверждения заказ сохраняется.

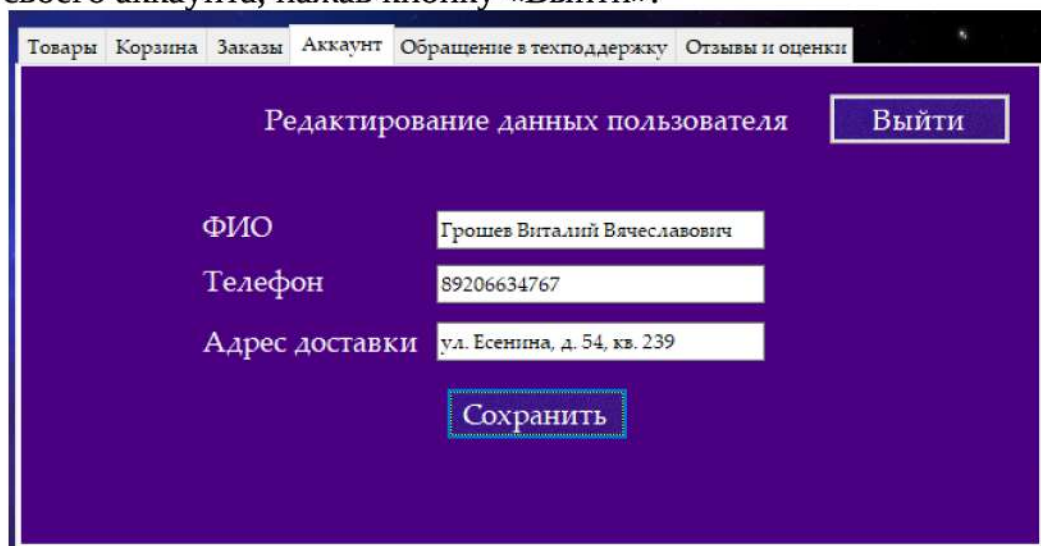
Оформленный заказ появляется на вкладке «Заказы» (рисунок 5), где отображается вся информация о нём.



Ваши заказы:				
	НомерЗаказа	Дата	Пункт_Выдачи	Способ_Оплаты
▶	4	01.04.2025	ул. Новосёлов, д.53	Карта

Рисунок 5 – Заказы пользователя

На вкладке «Аккаунт» (рисунок 6) пользователь может изменить свои личные данные, такие как имя, контактный телефон и адрес доставки. Для этого необходимо изменить нужные данные и нажать кнопку «Сохранить», также в этом разделе пользователь может выйти из своего аккаунта, нажав кнопку «Выйти».



Редактирование данных пользователя Выйти

ФИО

Телефон

Адрес доставки

Сохранить

Рисунок 6 – Вкладка с данными аккаунта пользователя

На вкладке «Обращение в техподдержку» (рисунок 7) пользователь может описать возникшую проблему, например, вопросы по заказу, технические неполадки или возврат товара. Для этого необходимо написать тему обращения (например, «Проблемы с доставкой», «Ошибка в заказе») и кратко изложить суть вопроса в текстовом поле. После отправки обращение поступает в службу поддержки, где сотрудники обрабатывают его в порядке очереди. На вкладке «Отзывы и оценки» отображаются все опубликованные отзывы о товарах от других покупателей, эта информация может помочь пользователям принимать решения о покупке товаров (рисунок 8).

Рисунок 7 – Вкладка обращения в техподдержку

Оценки и отзывы					
	Товар	ФИО_клиента	Текст_отзыва	Дата_отзыва	Оценка_товара
▶	Смартфон Samsu...	Соломонов Никит...	Отличный смар...	21.05.2024	5
	Ноутбук Lenovo T...	Соломонов Никит...	Пойдет	17.06.2024	4
	Ноутбук Lenovo T...	Соломонов Никит...	Пойдет	17.06.2024	3
	Ноутбук Lenovo I...	Соломонов Никит...	Классно	17.06.2024	5
	Ноутбук Lenovo T...	Соколов Никита ...	Хороший ноутб...	22.05.2024	4
	Ноутбук Lenovo T...	Соколов Никита ...	Хороший ноутб...	22.05.2024	3
	Sennheiser Mome...	Светиков Дмитри...	Наушники прос...	22.05.2024	4
	Xiaomi Mi Watch	Светиков Дмитри...	Отличные часы, ...	23.05.2024	5

Рисунок 8 – Вкладка отзывов и оценок товаров

Таким образом, разработанная информационная система предоставляет пользователю комплексный набор функций: от удобного просмотра товаров с мгновенным добавлением в корзину до прозрачного оформления заказов. Интеграция модулей техподдержки и отзывов позволяет оперативно решать вопросы и формировать доверие на основе оценок других покупателей, что в совокупности повышает эффективность работы интернет-магазина и удовлетворенность клиентов.

Архитектура и планы развития системы

Система реализована по двухуровневой клиент-серверной архитектуре, где клиентское приложение взаимодействует с серверной частью, обеспечивая управление товарами, заказами и

пользовательскими данными. Клиентская часть разработана на языке C# с использованием WindowsForms, что позволяет создавать интуитивно понятный интерфейс для десктоп-устройств. Выбор этой технологии обусловлен простотой интеграции с базой данных Microsoft SQL Server, поддержкой богатого набора элементов управления (таблицы, кнопки, формы) и возможностью быстрой адаптации под требования бизнес-логики. Приложение включает вкладки для работы с каталогом товаров, корзиной, историей заказов, профилем пользователя и техподдержкой. Данные отображаются в удобном виде, за счёт чего достигается простота взаимодействия.

Серверная часть системы построена на базе Microsoft SQL Server, обеспечивающего надежное хранение структурированных данных. База данных включает 17 взаимосвязанных таблиц, нормализованных до БКНФ для исключения избыточности.

В ближайших планах развития системы предусмотрено расширение функционала и повышение доступности сервиса для разных категорий пользователей. Первым шагом станет создание веб-версии интерфейса на основе ASP.NET, которая позволит работать с системой через браузер без установки десктоп-приложения. Адаптивный дизайн позволит корректно отображать интерфейс на любых устройствах – от компьютеров до смартфонов. Для авторизации пользователей будет реализована поддержка социальных сетей (например, вход через Google или VK), а также двухфакторная аутентификация для повышения безопасности.

Вторым ключевым направлением станет внедрение модуля рекомендаций, основанного на анализе истории покупок и поведения пользователей. Система будет собирать данные о просмотренных товарах, частоте покупок, оценках и отзывах, чтобы формировать персонализированные подборки.

Третьим этапом станет разработка мобильного приложения для iOS и Android, которое предоставит клиентам возможность совершать покупки в любом месте. Приложение будет синхронизироваться с текущей базой данных, сохраняя историю заказов, корзину и настройки пользователя.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация Microsoft SQL Server [Электронный ресурс] – URL: <https://docs.microsoft.com/sql> (дата обращения: 09.12.2024).
2. Руководство по разработке на C# и WindowsForms [Электронный ресурс] – URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms> (дата обращения: 09.12.2024).

3. Руководство по EntityFramework [Электронный ресурс] – URL: <https://docs.microsoft.com/ru-ru/ef/core/> (дата обращения: 09.12.2024).

4. Троелсен, Э. Язык программирования C# 6.0 и платформа .NET 4.6 / Э. Троелсен, Ф. Джепикс; пер. с англ. – М.: Вильямс, 2016. – 1400 с.

5. Троелсен Э. Язык программирования C# 7 и платформы .NET и .NET Core / Э. Троелсен, Ф. Джепикс; пер. с англ. – М.: Вильямс, 2018. – 400 с.

УДК 004.021

ДОРИН А.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

МЕТОД РОЯ ЧАСТИЦ

Рассматривается метод роя частиц и возможность его применения для нахождения оптимального уровня сигнала маршрутизаторов в компьютерной сети.

Алгоритм роя частиц (PSO) [1] был предложен в 1995 году Джеймсом Кеннеди и Расселом Еберхартом. Идея алгоритма была частично заимствована из исследований поведения скоплений животных (косяков рыб, стай птиц и т.п.), модель была несколько упрощена, а также были добавлены элементы поведения толпы людей, поэтому, в отличие, например, от алгоритма пчел агенты алгоритма (возможные решения) и были названы нейтрально - частицы.

Метод основан на утверждении, что знание заключается в социальном обмене информацией не только между поколениями, но и между элементами одного и того же поколения. Хотя PSO имеет индивидуальные характеристики, этот алгоритм также обладает некоторым сходством с характеристиками, обнаруженными в других вычислительных моделях, основанных на популяции, таких как генетические алгоритмы и другие эволюционные вычислительные методы. Преимущества метода PSO заключается в том, что он относительно прост, и его алгоритм сравнительно легко описать и реализовать.

Его простота и очевидная компетентность в нахождении оптимальных решений в сложных пространствах поиска привели к тому, что алгоритм PSO стал хорошо известен в научном сообществе, что способствовало его изучению и совершенствованию. Таким образом, было предложено множество подходов и протестированы

различные приложения, особенно ха последнее десятилетие.



Рисунок 1 – Блок-схема метода роя частиц

Для того, чтобы понять алгоритм роя частиц, нужно представить себе многомерное пространство (область поиска), в котором располагаются частицы (агенты алгоритма). Начальный шаг алгоритма предполагает расположение частиц случайным образом по всей

области поиска. При этом каждая частица в начале имеет нулевой вектор скорости. В каждой точке пространства, где оказывается частица, происходит расчёт значения целевой функции. При этом каждая частица запоминает, какое лучшее значение целевой функции она лично нашла и координаты соответствующей точки в пространстве. Также каждая частица знает где расположена точка, являющаяся лучшей среди всех точек, в которых находились частицы. На каждой последующей итерации алгоритма частицы корректируют свою скорость (модуль и направление), чтобы, с одной стороны, оказаться ближе к лучшей точке, которую частица нашла сама (авторы алгоритма называют этот аспект поведения "ностальгией"), и, в то же время, приблизиться к точке, которая в данный момент является глобально лучшей. Спустя некоторое количество итераций частицы концентрируются в некоторой области около глобального оптимума, однако существует вероятность, что часть частиц может оказаться где-то в относительно неплохом локальном экстремуме, но главное, чтобы хотя бы одна частица оказалась вблизи глобального экстремума.

Самое интересное в алгоритме – это коррекция скорости частицы, именно от этого шага зависит сходимость алгоритма. В первоначальном виде алгоритма коррекция скорости определяется следующим образом:

$$v_{i,j}^t = v_{i,j}^{t-1} + \varphi_1 r_{p,j}^t (p_{best,i,j}^{t-1} - x_{i,j}^{t-1}) + \varphi_2 r_{g,j}^t (g_{best,i,j}^{t-1} - x_{i,j}^{t-1}),$$

где $v_{i,j}^t$ – j-я компонента скорости i-ой частицы на t-ой итерации,

φ_1 и φ_2 – весовые коэффициенты, определяемые программистом, r_p^t и r_g^t – независимые случайные векторы, значения компонент которых находятся в диапазоне [0,1].

Перемещение частицы в новую точку области поиска происходит по следующей формуле:

$$x_{i,j}^t = x_{i,j}^{t-1} + v_{i,j}^t$$

По завершению процедуры коррекции рассчитывается значение целевой функции в каждой новой точке, каждая частица проверяет, не стала ли новая координата лучшей среди всех точек, где она побывала. Затем среди всех новых точек проверяется, не нашлось ли новой глобально лучшей точки, и, если это произошло, запоминаются ее координаты и значение целевой функции в ней.

Существуют также различные модификации данного алгоритма [2]. По большей части эти модификации были предназначены для того, чтобы сходимость алгоритма не так сильно зависела от выбора

весовых коэффициентов φ_1 и φ_2 .

В одной из модификаций алгоритма роя частиц предполагается введение еще одного весового коэффициента, называемого коэффициентом инерции. Таким образом, формула коррекции скорости частицы принимает следующий вид:

$$v_{i,j}^t = wv_{i,j}^{t-1} + \varphi_1 r_{p,j}^t (p_{best_{i,j}}^{t-1} - x_{i,j}^{t-1}) + \varphi_2 r_{g,j}^t (g_{best_{i,j}}^{t-1} - x_{i,j}^{t-1}),$$

Введение нового коэффициента способствует более плавному изменению скорости частицы. Коэффициент инерции может иметь константное значение или изменяться от шага к шагу.

Еще одной распространенной модификацией данного алгоритма является канонический алгоритм. В нем вводится нормировка коэффициентов φ_1 и φ_2 . Это сделано для того, чтобы уменьшить зависимость сходимости алгоритма от выбора коэффициентов.

Формула коррекции скорости частиц приобретает следующий вид:

$$v_{i,j}^t = \chi [v_{i,j}^{t-1} + \varphi_1 r_{p,j}^t (p_{best_{i,j}}^{t-1} - x_{i,j}^{t-1}) + \varphi_2 r_{g,j}^t (g_{best_{i,j}}^{t-1} - x_{i,j}^{t-1})],$$

$$\chi = \frac{2k}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|},$$

$$\varphi = \varphi_1 + \varphi_2, \varphi > 4$$

Коэффициент k определяется значением из интервала $(0,1)$.

Существуют также и такие модификации алгоритма PSO, учитывающие лучшей не точку, в которой каждая частица оказывалась, а точку, которая является лучшей для самой частицы и ближайших ее соседей.

Если говорить о недостатках алгоритма роя частиц, то нельзя не упомянуть, что, как и в других стохастических алгоритмах оптимизации, не гарантируется сходимость алгоритма при конечном количестве частиц. Также по причине ориентации алгоритма на одну лучшую точку увеличивается вероятность остановки алгоритма в относительно неплохом, но не глобальном оптимуме. К плюсам алгоритма можно отнести его простоту и возможность достаточно быстро модифицировать алгоритм на применение иной формулы расчета скорости частиц, если этого требует решаемая задача.

Помимо всего вышеперечисленного, несомненным достоинством данного алгоритма является то, что для его использования не требуется заранее знать точного градиента оптимизируемой функции. Данная особенность позволяет использовать алгоритм роя частиц для

определения оптимальных значений уровня сигналов маршрутизаторов в компьютерной сети.

Сформулируем решаемую задачу.

Пусть имеется некоторое множество маршрутизаторов $M = \{m_1, \dots, m_i\}$, где $i = \overline{1..n}$. Каждый маршрутизатор характеризуется уровнем сигнала p_i , значение которого лежит в диапазоне от 0 до 10. От уровня сигнала зависит площадь зоны покрытия сети. Зона покрытия представляет собой круг с центром в точке нахождения маршрутизатора. Зона перекрытия – область пересечения двух или более зон покрытия. Таким образом, задача сводится к нахождению оптимальных значений уровней сигналов каждого маршрутизатора таких, чтобы суммарная площадь зон покрытия за вычетом штрафов зон перекрытия была максимальной.

Целесообразно будет ввести по отношению к данной задаче систему бонусов и штрафов. Благодаря этому задача оптимизации становится однокритериальной, что и дает возможность использования метода роя частиц. Целевая функция будет иметь вид:

$$C = \sum S_{\text{покр}i} - \sum S_{\text{перекр}j}$$

Рассмотрим подробнее части этого выражения.

Зона покрытия сетью – зона, в которой уровень сигнала не ниже некоторого граничного значения l . Уровень сигнала в некоторой точке обратно пропорционален квадрату расстояния от этой точки до источника сигнала. Таким образом, радиус зоны покрытия определяется по формуле $r = \sqrt{\frac{x}{l}}$. Тогда площадь зоны покрытия

будет определяться равенством: $S_{\text{покр}i} = \pi * r_i^2 = \pi * \frac{x}{l}$.

Перекрытие – зона, в которой действуют несколько сигналов, уровень которых не ниже граничного значения. В этой зоне также есть доступ к сети, однако уровень сигнала ниже из-за помех, возникающих из-за того, что присутствует несколько сигналов на одном канале. Штраф за перекрытие будет определяться как произведение площади перекрытия на величину, обратную значению соотношения сигнал/шум (ОСШ, S/R, SNR).

$$SNR = \frac{P_{\text{сигнал}}}{P_{\text{шум}}}$$

При этом сигналом считаем сильнейший сигнал, действующий в

области перекрытия, а шумом – остальные сигналы в области.

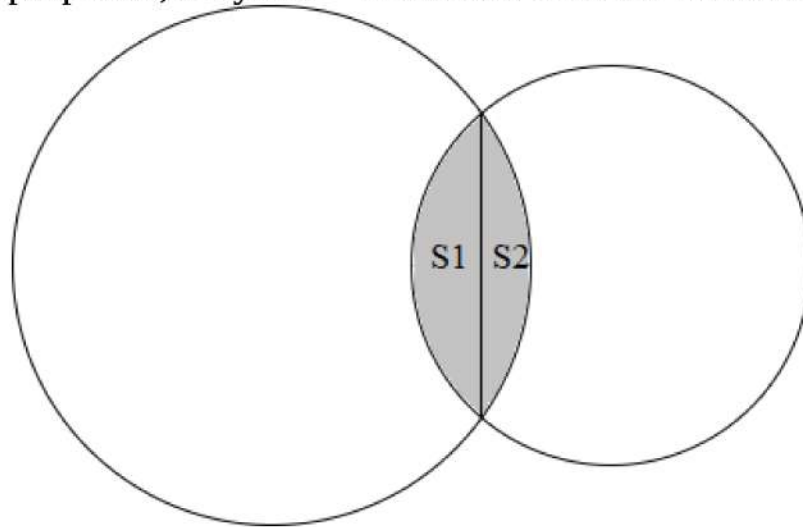


Рисунок 2 – Пример перекрытия

Площадь перекрытия вычисляется следующим образом:

$$S_{\text{перекр}} = S_1 + S_2$$

$$S_1 = \frac{R_1^2 * (F_1 - \sin(F_1))}{2}$$

$$S_2 = \frac{R_2^2 * (F_2 - \sin(F_2))}{2},$$

где

$$F_1 = 2 * \arccos\left(\frac{R_1^2 - R_2^2 + D^2}{2 * R_1 * D}\right)$$

$$F_2 = 2 * \arccos\left(\frac{R_2^2 - R_1^2 + D^2}{2 * R_2 * D}\right)$$

где R_1 – радиус первой окружности, R_2 – радиус второй окружности, D – расстояние между центрами окружностей.

Штраф за пересечение 3 зон и более определяется как сумма штрафов за перекрытие попарно взятых зон.

Пересечения находятся следующим образом. Рассматриваются все возможные пары зон, проверяется условие пересечения. Если условие выполняется, то находятся точки пересечения зон и рассчитывается площадь зоны перекрытия.

Тогда целевая функция определяется следующим образом:

$$C = \sum_{i=1}^n \pi * \frac{x}{l} - \sum \left(S_{\text{перекр}} * \frac{1}{SNR_j} \right)$$

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Kennedy J., Eberhart R. Particle swarm optimization/ In Proceedings of IEEE International Conference on Neural Networks. – 1995. – Vol. 4. – P. 1942-1948.
2. McCaffrey J.D. Graph partitioning using a Simulated Bee Colony algorithm»/ IEEE International Conference on Information Reuse & Integration. – Las Vegas, 2011. – P. 400-405.

УДК 004.032.26

ДОРКИН Д.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ В СЕГМЕНТАЦИИ ОБЪЕКТОВ ИЗОБРАЖЕНИЯ

Рассматриваются использование технологии сверточных нейронных сетей в сегментации объектов изображения, применение нейронных сетей для создания систем выделения конкретных объектов в потоке изображений.

В контексте современных экономических реалий значительное число предприятий активно внедряют автоматизированные решения. Данный подход способствует снижению издержек и росту операционной эффективности, что усиливает их рыночные позиции.

Однако в производственной сфере нередко возникают нестандартные ситуации, не поддающиеся формализации и регулированию с помощью традиционных автоматизированных систем, контроллеров и регуляторов. Для решения подобных задач целесообразно применение интеллектуальных технологий в производственных процессах либо разработка на их основе инновационных систем управления.

Искусственный интеллект охватывает широкий спектр научных дисциплин, каждая из которых ориентирована на решение специфических задач. Ключевыми направлениями в данной области являются машинное обучение, компьютерное зрение и искусственные

нейронные сети, играющие важную роль в развитии интеллектуальных систем.

Машинное обучение обеспечивает автоматическое улучшение алгоритмов через анализ больших данных. В области компьютерного зрения разрабатываются методы для автоматического понимания и обработки изображений. Нейросетевые архитектуры, воспроизводящие принципы биологических нейронов, демонстрируют способность к адаптивному обучению. Указанные подходы часто комбинируются в интеллектуальных системах, где нейросети существенно повышают точность и скорость принятия решений.

Современные исследования в сфере искусственного интеллекта предлагают разнообразные архитектурные решения нейронных сетей, способные решать схожие задачи обработки информации. Среди этого многообразия особое место занимают свёрточные нейронные сети (CNNs), доказавшие свою исключительную эффективность в задачах машинного анализа визуальных данных.

Данная архитектура использует специализированные математические операции для иерархического выделения признаков – от простых текстур до сложных объектов – через анализ пространственных взаимосвязей пикселей.

Для эффективного обнаружения объектов на изображении алгоритм должен обладать инвариантностью к их пространственному положению. Оптимальное решение предполагает анализ не только локальных признаков объекта, но и его взаимосвязей с другими элементами изображения в рамках общего контекста.

Для демонстрации рассмотрим аэрофотосъёмку урбанизированной зоны с множеством архитектурных объектов (см. рис. 1). Цель исследования – идентификация конкретного сооружения (например, девятиэтажного здания) среди визуального шума, создаваемого окружающими постройками.



Рисунок 1 – Аэрофотоснимок городской территории

Свёрточные нейронные сети обеспечивают пространственную инвариантность, формируя компактные и эффективные признаковые представления [3]. На начальных слоях CNN реализует принцип трансляционной эквивариантности, обрабатывая локальные области независимо от их положения. По мере углубления сети происходит интеграция локальных признаков в глобальное семантическое представление объекта.

В процессе решения задач машинного обучения, особенно в области компьютерного зрения, можно выделить несколько основных этапов [1].

Первоначально осуществляется формирование обучающей выборки. Объектами для классификации могут являться, например, фотографии растений, посаженных в дендрарии. Собранные данные подвергаются аннотированию: выделяются ключевые морфологические признаки (форма и размер листьев, характерные особенности), которые фиксируются в виде координатных меток. Полученная разметка сохраняется в стандартных форматах (.txt, .csv, .xml) без модификации исходных файлов. Данный этап предшествует непосредственному обучению нейросетевой модели.

На втором этапе необходимо определить структуру нейронной сети, в частности, свёрточную. Эта структура позволяет выделить из изображения ключевые особенности (признаки) с помощью свёрточных слоёв.

Свёрточные нейронные сети обладают сложной конфигурацией с множеством настраиваемых параметров [2], включающих топологию сети (количество и размерность слоёв), характеристики свёрточных операций (размер, количество фильтров и шаг их перемещения), параметры субдискретизации (тип и степень уменьшения

размерности), используемые функции активации, а также структуру финального классификационного модуля. Такая архитектура обеспечивает гибкость при проектировании CNN для различных задач компьютерного зрения.

Для реализации поставленной задачи предлагается нейросетевая модель на базе архитектуры Mask R-CNN, интегрирующей функции детекции и классификации объектов. Данная архитектура обеспечивает, во-первых, двойное представление признаков: карту фоновых характеристик и карту признаков всего изображения; во-вторых, сохранение пространственной информации об объектах (см. рис. 2).

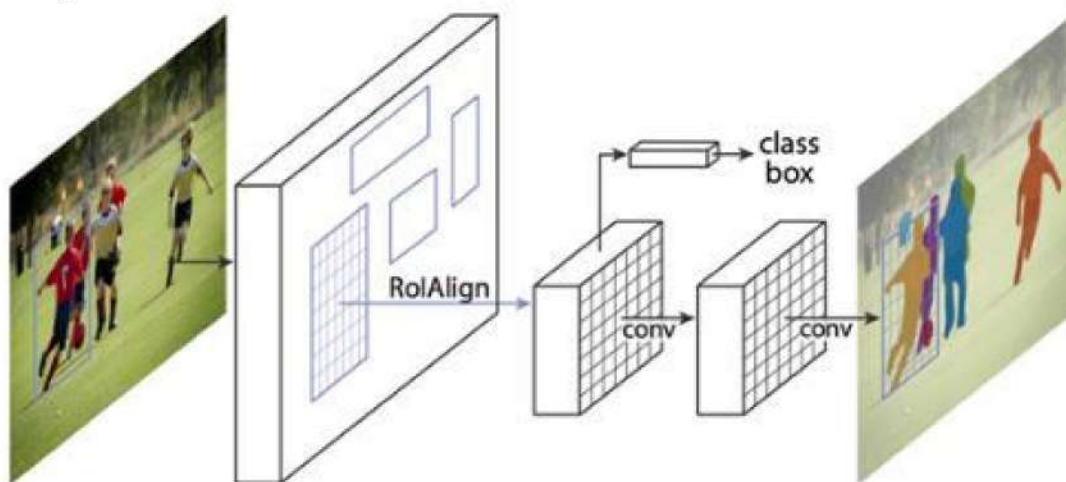


Рисунок 2 – Архитектура Mask R-CNN: RoIAlign – слой сети с информацией о расположении объекта; Classbox – набор данных о расположении объектов на исходных изображениях

Третий этап – этап обучения нейросетевой модели – включает загрузку аннотированных данных и их обработку нейронной сетью. Качество обучения определяется тремя ключевыми факторами: точностью предварительной разметки; объемом обучающей выборки; корректным распределением данных на тренировочный, валидационный и тестовый наборы.

Выборки распределяются на три вида: обучающая (для настройки параметров модели), валидационная (для подбора гиперпараметров), тестовая (для финальной оценки на неизвестных данных).

Для повышения эффективности обучения применяется аугментация – искусственное расширение обучающей выборки за счёт геометрических преобразований (к примеру, повороты на 90 градусов), фотометрических изменений (коррекция яркости/контраста) и прочих модификаций изображений.

Важно отметить, что аугментация используется только при обучении и преимущественно для тренировочного набора данных.

Независимо от предварительной обработки, все изображения преобразуются в 4D-тензор с размерностями: `batch_size` – количество параллельно обрабатываемых изображений (1 для одиночного изображения); RGB – три цветовых канала и H, W – пространственные размерности (высота и ширина)

В процессе обучения используются только два набора данных: обучающий (`train`) и валидационный (`validate`). Это позволяет минимизировать функцию потерь при обучении с помощью метода градиентного спуска. Выбор функции потерь зависит от специфики задачи в области компьютерного зрения.

Таким образом, для обнаружения нескольких классов можно использовать функцию потерь `Multi-ClassCrossEntropyLoss`:

$$L(\hat{y}, y) = - \sum_k y^k \log \hat{y}^k$$

В рамках данной функции осуществляется расчёт функции потерь для K классов. В качестве входных данных используются y и $\hat{y}$ – количество объектов каждого класса.

В процессе обучения нейронной сети необходимо вычислить функцию потерь по определённой формуле. Полученное значение следует сохранить. Оптимальными для решения задачи распознавания растений на общем изображении будут являться веса модели, соответствующие минимальному значению функции потерь.

Четвертый этап – апробация модели. Обученная нейросеть тестируется на независимой выборке данных. Полученные выходные данные анализируются для оценки точности работы модели и возможности её практического внедрения.

Разработанная CNN показывает высокую производительность при детекции объектов в промышленных средах. Использование эволюционных алгоритмов оптимизации позволяет автоматизировать подбор оптимальной структуры сети, упростить процесс обучения модели

Решение задачи оптимизации конфигурации нейронной сети с учётом множества критериев позволяет снизить вычислительную сложность модели, сохраняя при этом необходимую точность результатов [2].

Необходимо подчеркнуть, что подобные нейронные сети могут найти применение в различных областях, например, для идентификации музейных экспонатов, анализа состояния различных

поверхностей, а также для мониторинга дорожно-транспортной ситуации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Клехо Д.Ю., Карелина Е.Б., Батырев Ю.П. Использование технологии сверточных нейронных сетей в сегментации объектов изображения // Лесной вестник. – 2021. – Т. 25., № 1. – С. 140–145.

2. Нишчхал. Особенности организации и применения сверточных нейронных // Решетневские чтения. – 2017. – №21-2. – С. 356 – 357.

3. Скрипачев В.О., Гуйда М.В., Гуйда Н.В., Жуков А.О. Особенности работы сверточных нейронных сетей // International Journal of Open Information Technologies. – 2022. – Т. 10., № 12. – С. 53 – 61.

УДК 004.932

ЕЛАТНИКОВ Н.В., СКВОРЦОВ С.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ ОТСЛЕЖИВАНИЯ ДИНАМИЧЕСКИХ ОБЪЕКТОВ НА НЕПОДВИЖНОМ ФОНЕ

Рассматривается разработка программы для отслеживания движущихся объектов на неподвижном фоне, что актуально в задачах обеспечения безопасности, организации транспорта, спорта и медицины. Основными особенностями программы являются корреляционная функция, лежащая в основе реализованного алгоритма и позволяющая выявлять перемещения объектов, а также модульная структура для удобства модификации.

В условиях стремительного роста объема визуальной информации, генерируемой современными системами видеонаблюдения, задачи анализа и интеллектуальной обработки видеоданных приобретают первостепенное значение. Одной из наиболее актуальных и востребованных задач в этой области является отслеживание динамических объектов на фоне статических сцен. Эффективное решение этой задачи требует значительных вычислительных ресурсов, что затрудняет их применение в системах реального времени. Кроме того, многие существующие решения недостаточно устойчивы к различным внешним факторам, таким как изменения освещенности, шумы и перекрытия объектов. Поэтому

разработка новых, более эффективных и устойчивых алгоритмов особенно важно для выявления перемещений объектов.

В данной работе предлагается использовать взаимно-корреляционную функцию:

$$p(\alpha_{ij}, \beta_{ij}) = \frac{1}{(\sigma_{ij} \sigma_{vj}) \left(\frac{1}{M_F N_F} \sum_{m=0}^{M_F-1} \sum_{n=0}^{N_F-1} Q_{vj}(m, n) Q_{ij}(m + \alpha_{ij}, n + \beta_{ij}) - \bar{Q}_{vj} \bar{Q}_{ij} \right)}$$

где σ_{ij} , σ_{vj} – среднеквадратическое отклонение; $\bar{Q}_{vj}$, $\bar{Q}_{ij}$ – среднее значение яркости j -й пары фрагментов изображений; M_F , N_F – размеры фрагментов; α_{ij} , β_{ij} – сдвиги фрагмента по осям x и y .

Она устойчива к изменениям освещенности и контрастности изображений. Алгоритм основан на вычислении взаимосвязи между пикселями текущего и предыдущего кадров видеопотока. Высокое значение корреляции указывает на то, что объект в текущем кадре соответствует объекту, ранее зафиксированному в предыдущем кадре. Это позволяет определить местоположение отслеживаемого объекта и траекторию его движения.

Повышение точности определения местоположения объектов, может быть достигнуто за счет использования методов субпиксельной интерполяции, позволяющих оценивать координаты объектов с точностью, превышающей размер одного пикселя.

Кроме того, реализация алгоритмов предобработки изображений, основанных на методах фильтрации и выделения контуров, позволит повысить устойчивость программы к шумам и улучшить качество отслеживания. Для подавления различных типов шумов, присутствующих в видеоизображениях, могут быть использованы различные типы фильтров, например, оператор Собеля:

$$f = \sqrt{G_x^2 + G_y^2},$$

где G_x , G_y – изображения, полученные путем применения «горизонтальной» и «вертикальной» масок, фильтр Гаусса и вейвлет-преобразование, что требует проведения дополнительных исследований.

Для программной реализации поставленных задач выбран язык C++, известный своей высокой производительностью и гибкостью, что критически важно для обработки видеопотоков в реальном времени. С целью эффективной работы с изображениями и реализации стандартных алгоритмов обработки может использоваться библиотека

OpenCV (Open Source Computer Vision Library), предоставляющая широкий набор оптимизированных функций для работы с изображениями и видео.

Для достижения максимальной производительности и ускорения вычислений, особенно в ресурсоемких алгоритмах корреляции и фильтрации, предлагается использовать возможности параллельных вычислений. В частности, технология CUDA (Compute Unified Device Architecture) от NVIDIA основана на использовании вычислительной мощности графических процессоров (GPU), что позволяет значительно увеличить скорость обработки данных. Кроме того, для эффективного распараллеливания вычислений на многоядерных центральных процессорах (CPU) может быть применена технология OpenMP (Open Multi-Processing), обеспечивающая возможность параллельного выполнения участков кода на нескольких ядрах процессора, что также способствует повышению общей производительности программы.

Модульная архитектура программы позволит легко добавлять новые функциональные возможности и заменять существующие модули на более эффективные. При этом использование модульного подхода существенно упростит дальнейшее развитие программы, обновление и модификацию. Основные модули программы обеспечивают выполнение следующих функций:

- **Захват изображений.** Модуль отвечает за загрузку и обработку видеоданных из различных источников, таких как веб-камеры, IP-камеры, файлы и видеопотоки. Он обеспечивает совместимость с различными форматами файлов и типами камер, а также выполняет первичную обработку изображений, такую как изменение размера, цветокоррекция и преобразование в другие цветовые пространства.
- **Обработка изображений.** Модуль выполняет предобработку изображений с целью улучшения качества отслеживания. Он включает в себя различные алгоритмы фильтрации шумов, выделения контуров, нормализации освещенности и другие методы, направленные на повышение устойчивости к внешним воздействиям.
- **Алгоритмы отслеживания.** Модуль реализует основные алгоритмы вычисления корреляции, необходимые для отслеживания объектов в видеопотоке. Он включает в себя алгоритмы поиска объектов, оценки их местоположения и предсказания их дальнейшего движения.
- **Интерфейс пользователя.** Модуль обеспечивает взаимодействие с пользователем, позволяя ему задавать параметры отслеживания, просматривать результаты отслеживания в режиме реального времени и экспортировать данные.

Перспективным направлением дальнейших исследований представляется повышение интеллектуальных возможностей программы. В частности, улучшение алгоритмов отслеживания может быть достигнуто с использованием методов глубокого обучения и нейронных сетей, что позволит повысить точность и устойчивость к сложным внешним условиям.

Практическое применение программы возможно в различных областях, таких как системы безопасности и видеонаблюдения для автоматического обнаружения и отслеживания нарушителей и предотвращения преступлений, в транспортных системах для мониторинга трафика и предотвращения аварий, в спортивном анализе для оценки эффективности тренировок и тактики игры, а также в медицинской визуализации для мониторинга движения органов и тканей. Результаты работы программы также могут быть полезны для специалистов, занимающихся разработкой и внедрением современных средств анализа видеоданных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Гусев С.И., Егошкин Н.А., Еремеев В.В., Москвитин А.Э., Кузнецов А.Е., Ушенкин В.А. Цифровая обработка данных радиолокационного зондирования Земли из космоса. – М.: КУРС, 2023. – 272 с.
2. Лобанов А.Н. Фотограмметрия. – М.: НЕДРА, 1984. – 552 с.
3. Антонушкина С.В., Гуров В.С., Егошкин Н.А. и др. Современные технологии обработки данных дистанционного зондирования Земли. – М.: ФИЗМАТЛИТ, 2015. – 460 с.
4. Прохоренок Н.А. OpenCV и Java. Обработка изображений и компьютерное зрение. – СПб.: BHV, 2018. – 320 с.
5. Бакулева М.А., Скворцов С.В., Хрюкин В.И. Методы оптимизации. – Рязань: Рязанский филиал МЭСИ, 2015. – 160 с.
6. Страуструп Б. Язык программирования C++. – М.: БИНОМ, 2006. – 1104 с.
7. Модель программирования CUDA [Электронный ресурс]: учебник / В.В. Коробицын [и др.]. Электрон. текстовые данные. – Омск: Омский государственный университет им. Ф.М. Достоевского, 2012. – 256 с.
8. Параллельные вычисления на GPU. Архитектура и программная модель CUDA [Электронный ресурс]: учебное пособие / А.В. Боресков [и др.]. Электрон. текстовые данные. – М.: Московский государственный университет имени М.В. Ломоносова, 2015. – 336 с.

УДК 004.8

ЕНИВАТОВ А.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИНТЕГРАЦИЯ ИНТЕЛЛЕКТУАЛЬНОГО АССИСТЕНТА GIGACHAT В ОБРАЗОВАТЕЛЬНУЮ ПЛАТФОРМУ

В статье рассматриваются теоретические основы интеграции интеллектуального ассистента GigaChat в цифровую образовательную среду, а также его потенциальное влияние на вовлечённость студентов.

Современное образование всё активнее переходит в цифровой формат, и на этом фоне особую значимость приобретают технологии, способные сделать обучение более гибким, персонализированным и эффективным. Одной из таких технологий являются интеллектуальные ассистенты на основе искусственного интеллекта.

GigaChat — отечественный ИИ-ассистент, способный поддерживать диалог на естественном языке и отвечать на учебные вопросы. Его внедрение в структуру образовательной платформы открывает новые возможности для поддержки студентов.

Цифровые образовательные платформы давно перестали быть просто электронными дневниками или хранилищами файлов. Сегодня это полноценные экосистемы, объединяющие средства коммуникации, планирования и поддержки обучения. Современный подход к созданию таких платформ основан на идее модульности, где каждый компонент может быть заменён или дополнен. Интеллектуальные ассистенты в этой логике представляют собой один из модулей поддержки, обеспечивающий помощь студенту в процессе взаимодействия с системой и учебным контентом.

Языковая модель *GigaChat* обучена на русскоязычных данных, что делает её особенно релевантной для локальных образовательных задач. В контексте обучения он может объяснять сложные понятия, помогать в навигации по платформе, давать рекомендации по выполнению заданий и отвечать на организационные вопросы. *GigaChat* не заменяет преподавателя, а дополняет его, особенно в ситуациях, когда студент сталкивается с трудностями вне аудитории.

Интеграция интеллектуального ассистента требует не только технического подключения через API, но и методической проработки его роли в системе. Характеристики такие как, контекстности, адаптивности, непрерывности и интерактивности становятся основой

такой интеграции. Эти характеристики позволяют сделать взаимодействие со студентом более человечным и поддерживающим.

Для взаимодействия с интеллектуальным ассистентом GigaChat используется RESTAPI, доступ к которому реализуется через механизм аутентификации по протоколу OAuth 2.0 (clientcredentialsflow). Получение токена осуществляется посредством отправки POST-запроса на адрес <https://ngw.devices.sberbank.ru:9443/api/v2/oauth> с указанием области доступа (scope=GIGACHAT_API_PERS) и авторизационных данных клиента, закодированных в формате Base64. В ответ возвращается JSON-объект с полем access_token, который используется в заголовке Authorization: Bearer<токен> при последующих обращениях к API.

Основной рабочий эндпоинт POST /api/v1/chat/completions предназначен для генерации ответа на основе введённой пользователем реплики в контексте предыдущего диалога. Структура запроса включает обязательное поле model (например, "GigaChat"), массив messages, содержащий историю общения, а также дополнительные параметры: temperature, top_p, n и max_tokens.

- temperature – коэффициент случайности при выборе слов. Чем выше значение (например, 0.8–1.0), тем более креативным и разнообразным будет ответ, но при этом увеличивается вероятность отклонения от фактической информации. Низкие значения (0.1–0.3) обеспечивают более сдержанный и предсказуемый результат.

- top_p (также называемый nucleus sampling) – параметр обрезки распределения вероятностей. Вместо выбора из всех возможных слов, модель выбирает из наименьшего множества, суммарная вероятность которого превышает top_p. Например, значение 0.9 означает, что модель выбирает среди самых вероятных 90% слов. Это позволяет контролировать баланс между разнообразием и надёжностью ответа.

- n – количество генерируемых вариантов ответа. Значение n = 1 означает, что возвращается один вариант. При необходимости сравнения или выбора наилучшего ответа можно установить n > 1, и в ответе API будут содержаться несколько альтернатив.

Пример запроса:

```
{
  "model": "GigaChat",
  "messages": [
    { "role": "system", "content": "Ты – помощник в учебной платформе." },
    { "role": "user", "content": "Объясни, что такое нейронная сеть." }
  ]
}
```

```
],  
  "temperature": 0.7,  
  "top_p": 0.9,  
  "n": 1,  
  "max_tokens": 512  
}
```

Ответ API приходит в виде JSON-объекта, содержащего поле `choices`, где в каждом элементе представлен сгенерированный ответ ассистента:

```
{  
  "choices": [  
    {  
      "message": {  
        "role": "assistant",  
        "content": "Нейронная сеть — это модель,  
вдохновлённая работой человеческого мозга..."  
      }  
    }  
  ]  
}
```

Такой формат общения позволяет гибко встраивать GigaChat в интерфейс образовательной платформы, создавая иллюзию живого диалога с цифровым помощником, способным учитывать контекст и специфику запроса.

Вовлечённость давно признана одним из главных факторов успешного обучения. Она включает интерес, активность и эмоциональное включение в процесс. Использование ИИ-ассистента может повышать вовлечённость за счёт оперативной обратной связи и адаптивирования ответов на запросы студентов.

Эмпирические исследования подтверждают эффективность применения интеллектуальных ассистентов в образовательной среде. Согласно данным StanfordHAI, использование языковых моделей способствует повышению академических результатов на 18–24% за счёт улучшения понимания и ускорения выполнения заданий. Исследование MicrosoftResearch показало, что внедрение ИИ-ассистентов увеличивает среднюю вовлечённость студентов на 35%, а 63% респондентов отметили снижение тревожности при самостоятельной работе. Эти данные указывают на позитивное влияние ИИ на мотивацию и активность студентов.

Таким образом, интеграция GigaChat в образовательную платформу укладывается в современные подходы к цифровой

педагогике и может служить важным инструментом трансформации учебного взаимодействия.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Шульженко А.В., Грудницкая Н.Н., Кутьин И.В. Концепции цифровой педагогики и их влияние в системе образования // Психология и педагогика служебной деятельности. – 2024. – №2.
2. Русскоязычная мультимодальная нейросеть от Сбера/Код Дурова [Электронный ресурс]. – URL: <https://kod.ru/gigachat-sber-neuro> (дата обращения: 29.03.2025).
3. Stanford Institute for Human-Centered Artificial Intelligence. [Электронный ресурс]. – URL: <https://hai.stanford.edu/ai-index/2023-ai-index-report> (дата обращения: 29.03.2025).
4. Microsoft Research. Educational AI Assistants: Pilot Study on Impact in Digital Learning Environments [Электронный ресурс]. – URL: <https://www.microsoft.com/research> (дата обращения: 29.03.2025).

УДК 004.8

ЕСЬКИН М.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ ЛОКАЛЬНЫХ ЯЗЫКОВЫХ МОДЕЛЕЙ В ИНЖЕНЕРНОЙ СРЕДЕ С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ RAG

Рассматривается возможность использования локальных языковых моделей для автоматизации взаимодействия с технической документацией в инженерных задачах. Приводятся примеры конфигураций оборудования и моделей.

В последние годы языковые модели искусственного интеллекта (LLM) стали ключевым элементом автоматизации многих процессов на предприятиях, упрощая и ускоряя работу с большими объемами текстовой информации. Однако широкое внедрение подобных технологий в инженерной и промышленной сферах ограничено требованиями безопасности и конфиденциальности данных. В связи с этим особое значение приобретают локальные решения, позволяющие использовать интеллектуальные системы внутри организации, без передачи данных во внешние облачные сервисы.

В данной работе рассматривается практическая возможность использования локальных LLM в связке с технологией RAG,

обеспечивающей взаимодействие с внешними документами (PDF, DOCX и т.п.) для генерации ответов.

Основной целью исследования являлась оценка целесообразности и эффективности применения локальных языковых моделей формата GGUF на предприятии инженерного профиля. Исследование было сфокусировано на:

- обработке технической и нормативной документации;
- автоматизации рутинных задач сотрудников;
- ускорении и оптимизации рабочих процессов при помощи интеллектуального ассистента.

Для оценки реальной применимости была изучена технология Retrieval-Augmented Generation (RAG) [10], предполагающая использование базы загруженных документов для точного и информативного ответа нейросети.

Используемые модели и конфигурация оборудования

Для проведения исследования были выбраны модели формата GGUF, наиболее подходящие под задачи инженерного сектора и русскоязычные особенности:

- Saiga LLaMA3 8B – специализированная русскоязычная модель на основе LLaMA3 [5].
- YandexGPT-5 Lite 8B – официальная русскоязычная модель Яндекса, ориентированная на диалог и инструкции [6].
- Granite3.1 8B – модель с явной поддержкой технологии RAG, подходящая для технических задач [7].
- Qwen2.5 7B – мультиязычная модель с увеличенной длиной контекста [8].
- DeepSeek R1 8B – модель, оптимизированная для аналитических и логических задач [9].

Тестирование проводилось на типовом оборудовании среднего класса:

- Процессор: AMD Ryzen серий 5000/7000
- Оперативная память: 16 ГБ DDR4
- Видеокарта (для дополнительного тестирования): Radeon серии 7000, с 16 ГБ видеопамати, использующая Vulkan API.

Результаты практического тестирования

Проведённые испытания подтвердили стабильную работу локальных моделей в условиях предприятия, обеспечивая скорость генерации порядка 10–11 токенов в секунду при использовании только центрального процессора.

Дополнительные тесты с видеокартой Radeon серии 7000 (16 ГБ видеопамати, Vulkan API) показали существенное ускорение до 90

токенов в секунду, что примерно в 8–9 раз превышает производительность на CPU. Этот факт свидетельствует о возможности использования аппаратного ускорения от различных производителей GPU, включая как AMD, так и NVIDIA.

Рекомендуемая архитектура системы для предприятия

На основании результатов тестирования рекомендована серверная платформа AnythingLLM [11], развёртываемая в Docker-контейнере. Данная платформа идеально подходит для интеграции технологии RAG и позволяет:

- безопасно хранить документы и модели на локальном сервере предприятия;
- исключить передачу данных за пределы организации, полностью обеспечивая конфиденциальность;
- организовать удобный веб-интерфейс для сотрудников;
- поддерживать разнообразие локальных моделей формата GGUF с учётом специфики задач предприятия;
- использовать встроенную векторную базу данных для поиска и генерации ответов на основании внутренней документации.

Несмотря на то, что первичные испытания были проведены с помощью интерфейса LM Studio [12], выбор платформы AnythingLLM обусловлен необходимостью многопользовательского доступа, расширенных возможностей интеграции и простоты администрирования.

Практическая значимость и выводы

Исследование наглядно показало, что локальные языковые модели, интегрированные с технологией RAG, представляют собой эффективное и безопасное решение для предприятий, предъявляющих высокие требования к безопасности и конфиденциальности своих данных. Данный подход позволяет существенно сократить время работы с документацией, снизить нагрузку на сотрудников и улучшить качество принимаемых решений благодаря быстрому доступу к релевантной информации.

Отдельным преимуществом является универсальность используемого оборудования: описанная технология не требует дорогостоящих специализированных решений и позволяет использовать GPU разных производителей, что делает систему доступной и гибкой с точки зрения аппаратного обеспечения.

Таким образом, внедрение предложенной архитектуры на предприятиях существенно оптимизирует рабочие процессы, повышая эффективность инженерных и технических подразделений.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Савченко Е.Ю. Основы и применение нейронных сетей: теория, практика, инновации: учеб. пособие / Е.Ю. Савченко. – Бишкек: Нео Принт, 2024. – 116 с.
2. Крылов И.В., Сафронов А.Н. Применение ИИ в инженерной практике. – М.: Техносфера, 2023. – 224 с.
3. Информационные технологии в промышленности: сб. науч. тр. – Рязань: РГРТУ, 2024. – 160 с.
4. TheBloke. GGUF Model Repository [Электронный ресурс]. – URL: <https://huggingface.co/TheBloke> (дата обращения: 16.04.2025).
5. IlyaGusev. saiga_llama3_8b [Электронный ресурс]. – URL: https://huggingface.co/IlyaGusev/saiga_llama3_8b (дата обращения: 16.04.2025).
6. Yandex. YandexGPT-5-Lite-8B-Instruct [Электронный ресурс]. – URL: <https://huggingface.co/yandex/YandexGPT-5-Lite-8B-instruct> (дата обращения: 16.04.2025).
7. Vikram-fresche. granite-3.1-8b-instruct [Электронный ресурс]. – URL: <https://huggingface.co/vikram-fresche/granite-3.1-8b-instruct> (дата обращения: 16.04.2025).
8. BabaK07. Qwen2.5-7B-Instruct-1M-GGUF [Электронный ресурс]. – URL: <https://huggingface.co/BabaK07/Qwen2.5-7B-Instruct-1M-GGUF/tree/main> (дата обращения: 16.04.2025).
9. CasiAC. deepseek-r1-8b-ciberseguridad [Электронный ресурс]. – URL: <https://huggingface.co/CasiAC/deepseek-r1-8b-ciberseguridad/tree/main> (дата обращения: 16.04.2025).
10. Mintplex Labs. AnythingLLM GitHub Repository [Электронный ресурс]. – URL: <https://github.com/Mintplex-Labs/anything-llm> (дата обращения: 16.04.2025).
11. Mintplex Labs. AnythingLLM Docker Hub [Электронный ресурс]. – URL: <https://hub.docker.com/r/mintplexlabs/anythingllm> (дата обращения: 16.04.2025).
12. LM Studio [Электронный ресурс]. – URL: <https://lmstudio.ai/> (дата обращения: 16.04.2025).
13. lmstudio-ai. GitHub Repository [Электронный ресурс]. – URL: <https://github.com/lmstudio-ai> (дата обращения: 16.04.2025).

УДК 004.021

ЗАЙЦЕВ Е.С.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ LSTM ДЛЯ
ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ**

В статье рассмотрено применение нейросетевой архитектуры LSTM (Long Short-Term Memory) для прогнозирования временных рядов. Представлены основные этапы подготовки данных и построения модели.

Введение

Временные ряды представляют собой последовательности данных, упорядоченные по времени. Такие данные часто встречаются в реальной жизни. Например, к ним относятся показания счётчиков электроэнергии, курсы валют, движение транспорта, данные медицинского мониторинга.

Прогнозирование таких данных помогает заранее определить, как будет меняться наблюдаемая величина. Это может быть полезно при планировании ресурсов, управлении рисками или принятии решений.

Ранее аналитики часто использовали классические статистические методы – такие как ARIMA. Эти методы эффективны, если данные стабильны и линейны. Однако на практике данные часто нестабильны, содержат шум и нелинейные зависимости. В таких случаях статистические подходы становятся недостаточно точными.

Современные нейросетевые модели, такие как LSTM, дают лучшие результаты. Они запоминают значения, наблюдаемые в прошлом, и используют эту информацию при прогнозировании. Это особенно важно, если поведение объекта во времени зависит не только от последних значений, но и от событий, которые произошли давно.

Подготовка и обработка данных

Перед обучением модели данные нужно подготовить. Первый шаг – нормализация. Все значения приводятся к одному масштабу, обычно в диапазоне от 0 до 1. Это помогает ускорить и стабилизировать обучение модели.

Далее временной ряд делится на две части: одна используется для обучения, другая – для проверки точности прогноза. Используется метод скользящего окна: создаются последовательности фиксированной длины, которые служат входными данными для нейросети. Каждой такой последовательности соответствует одно

целевое значение – это следующий элемент ряда, который модель должна предсказать.

Такая подготовка позволяет превратить простой список чисел во входную и выходную обучающие выборки, с которыми нейросеть может работать.

Построение и обучение модели

Модель создаётся с использованием библиотеки Keras. Основным элементом – слой LSTM с 50 нейронами. За ним размещён выходной слой, который формирует одно прогнозируемое значение. В качестве функции потерь используется среднеквадратичная ошибка. Для настройки весов применяется оптимизатор Adam.

Обучение модели заключается в многократном показе ей обучающих примеров. После каждой эпохи сеть корректирует свои внутренние параметры, стремясь уменьшить ошибку предсказания.

После завершения обучения модель проверяют на тестовой выборке. Результаты прогнозов денормализуют, чтобы сравнить с реальными значениями. Для оценки точности используются такие показатели, как RMSE и MAE. Они помогают понять, насколько хорошо модель справляется со своей задачей.

Применение модели и результаты

Модель была протестирована на нескольких наборах реальных временных рядов. Результаты показали высокую точность предсказаний. Даже при наличии шума и нестабильности данные интерпретировались корректно. LSTM смог уловить сложные зависимости, которые не всегда видны при беглом анализе.

Модель хорошо справилась с задачами в сферах энергетики, финансов и логистики. Также она показала хорошие результаты при анализе медицинских данных.

Качество прогноза сильно зависит от того, как подготовлены данные и насколько правильно подобраны параметры модели. Длина входного окна, число нейронов и скорость обучения влияют на итоговую точность. При оптимальной настройке LSTM даёт стабильный результат даже при больших объёмах данных.

Преимущества и ограничения

Главное преимущество LSTM – способность запоминать информацию на длительное время. В отличие от обычных рекуррентных сетей, LSTM не теряет важные данные, если они были получены много шагов назад.

Модель можно легко адаптировать под разные задачи. Количество слоёв и нейронов можно менять, в зависимости от сложности данных. Но обучение таких сетей требует значительных

ресурсов. Особенно при работе с большими наборами данных и длинными входными последовательностями.

Также важно правильно настроить параметры. Если, например, выбрать слишком большое количество нейронов, модель может переобучиться. Это значит, что она будет хорошо работать на обучающих данных, но плохо – на новых.

Текущие направления развития

Сегодня исследователи стремятся улучшить LSTM-модели. Один из способов – объединить LSTM с другими типами слоёв. Например, можно использовать сверточные слои (CNN) перед LSTM. Это помогает выделить важные особенности входных данных.

Также применяется механизм внимания (attention). Он позволяет модели определять, какие части последовательности важнее при прогнозе. Бинаправленные LSTM дают возможность учитывать информацию как из прошлого, так и из будущего контекста.

Кроме того, развивается использование трансформеров. Эти модели пришли из области обработки текста и показали отличные результаты. Временные ряды по структуре похожи на последовательности слов, поэтому трансформеры начинают активно применять и здесь.

Заключение

Нейронные сети LSTM – это мощный инструмент для анализа временных рядов. Они позволяют строить точные прогнозы, даже когда данные содержат шум или имеют сложные зависимости.

Эксперименты показали, что модели на базе LSTM успешно справляются с задачами в разных прикладных областях. Их можно применять в экономике, здравоохранении, логистике и других сферах.

Будущее таких моделей связано с развитием гибридных архитектур, использованием более мощных вычислительных средств и интеграцией с новыми алгоритмами. Всё это делает LSTM важным элементом современного инструментария анализа данных.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Гаврилов Д.М., Соколов А.В. Машинное обучение в задачах прогнозирования временных рядов // Вестник НИУ ВШЭ. – 2020. – №1. – С. 56–65.
2. Махортов А.А., Орлов И.Н. Прогнозирование временных рядов с использованием глубокой нейронной сети LSTM // Научно-технические ведомости СПбГПУ. – 2019. – №3. – С. 145–154.

УДК 004.75

ЗАМЕШАЕВ Д.В., СКВОРЦОВ С.В.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**ВЫБОР ХЕШ-ФУНКЦИИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ
БАЛАНСИРОВКИ НАГРУЗКИ В СЕТИ**

Рассмотрены задачи балансировки нагрузки в сети и показаны возможные варианты её решения в рамках модели OSI. Выполнен анализ методов вычисления хеш-функций и обоснован выбор наиболее предпочтительных для решения задачи балансировки нагрузки в сети.

Для эффективного использования сетевых ресурсов (кластеров, каналов и т.д.) решается задача балансировки нагрузки. В зависимости от уровня по модели OSI, она может формулироваться разными способами.

На сетевом уровне требуется обеспечить передачу данных в сети таким образом, чтобы за конкретные вычислительные узлы, представленные IP-адресами, отвечали разные физические машины. Данный вариант балансировки может быть осуществлён различными способами: DNS-балансировка, балансировка по территориальному признаку, построение NLB-кластера и т.д. [1].

На прикладном уровне задача балансировки нагрузки может формулироваться исходя из тех или иных показателей, которые выбраны целевыми. Например, пусть требуется достичь минимального отклика системы, состоящей из нескольких серверов с различными вычислительными мощностями. Тогда задачу балансировки нагрузки можно сформулировать следующим образом: организовать распределение запросов внутри системы таким образом, чтобы обеспечивалась равномерная нагрузка вычислительных узлов. Решение данной задачи может быть выполнено в виде специального узла – «балансировщика», чаще всего представляющего собой прокси-сервер, анализирующий запросы и перенаправляющий их тому или иному серверу в зависимости от результата анализа и данных о сервере. Предположим, что имеется три сервера с вычислительными мощностями, выраженными некоторым абстрактным числовым показателем, равными 3, 5 и 10 единицам. В таком случае, «балансировщик» будет отправлять относительно простые запросы предпочтительнее первому серверу, запросы средней мощности – предпочтительнее второму серверу, и достаточно тяжеловесные – предпочтительнее третьему серверу. Представленный подход может

использоваться для балансировки нагрузки между серверами асимметричной фермы [2].

Не меньший интерес представляет задача балансировки нагрузки в сети на канальном и на сетевом уровне применительно к сетевому оборудованию: коммутаторам и маршрутизаторам. При таком рассмотрении следует уделить внимание такому понятию как ЕСМР.

ЕСМР (Equal-Cost Multi-Path) – это метод маршрутизации, который обеспечивает распределение трафика по нескольким равнозначным путям с одинаковой стоимостью (метрикой) между отправителем и получателем [3]. Равнозначные пути формируют ЕСМР-группу. ЕСМР предназначен для равномерного распределения трафика между существующими маршрутами. Это достигается путём применения различных алгоритмов. Один из них – RoundRobin. Данный алгоритм относится к по пакетным алгоритмам балансировки и заключается в циклическом переборе имеющихся каналов/путей в группе и передаче через них поступающих пакетов данных [4]. Этот алгоритм обеспечивает равномерное распределение нагрузки на все имеющиеся каналы, однако имеет ряд существенных недостатков. В частности – данный метод может привести к деградации производительности и увеличению нагрузки на сетевое оборудование в сетях TCP из-за переупорядочивания пакетов. Данный недостаток вызван тем, что алгоритм RoundRobin не учитывает такое понятие как поток данных, в связи с чем пакеты одного и того же потока (от клиента А до клиента В) всегда передаются по разным каналам.

Для устранения данного недостатка активно используется по потоковый алгоритм балансировки, основанный на использовании хеш-функции. С их помощью появляется возможность достаточно быстро определить принадлежность пакета к тому или иному потоку данных, что позволяет направлять пакеты одного потока данных в один и тот же канал из ЕСМР-группы.

Хеширование представляет собой математическое преобразование, которое отображает данные произвольной длины (входной набор данных в битовую строку фиксированной длины (хеш-значение) [5].

Существует множество различных алгоритмов вычисления хеш-функции. Применительно к задаче балансировки нагрузки в сети, требуется выбрать такие методы, которые обеспечат равномерное распределение, низкую вероятность коллизий и достаточную устойчивость к изменениям в системе. Также следует уточнить, что в

контексте решаемой задачи, вычисляемый ключ, равно как и входные данные, будет представлять собой целое число.

Укажем основные методы вычисления хеш-функции.

1. Модульный метод. Является наиболее простым способом вычисления функции и осуществляется по следующей формуле:

$$h(x) = x \bmod n ,$$

где x – ключ (IP-адрес, номер порта и т.п.), n – количество участников в ЕСМР-группе [6]. Данный метод считается эффективным для получения равномерного распределения ключей, однако обладает рядом недостатков, наиболее существенный из которых (применительно к поставленной задаче) – неустойчивость к изменению числа узлов. При добавлении или удалении узла возникнет необходимость перерасчёта хеш-функции для всех ключей.

2. Комбинированный метод. Является объединением модульного и мультипликативного методов и применяется для хеширования ключей, заданных числами с плавающей точкой из фиксированного диапазона. Значение хеш-функции вычисляется по следующей формуле:

$$h(x) = \text{trunk}(x * \alpha) \bmod N ,$$

где x – ключ, α – константа, равная золотому сечению: 0,618033..., N – некоторое целое число, которое желательно должно быть простым. Данный метод обладает теми же недостатками, что и модульный метод.

3. Криптографические хеш-функции (SHA-1, SHA-256, MD5 и т.д.). Обеспечивают достаточно хорошую равномерность распределения и минимум коллизий, однако являются избыточными для решения задачи балансировки нагрузки в сети, а также обладают ощутимо высокой трудоёмкостью с точки зрения вычислительных ресурсов.

4. Консистентное хеширование. По сути, не является методом вычисления хеш-функции. Заключается в отображении узлов и ключей на некоторое замкнутое (закольцованное) пространство хешей [7]. Под узлами понимаются члены ЕСМР-группы. Для каждого из них вычисляется хеш-функция одним и тем же методом. Далее вычисленные узлы фиксируются на выбранном пространстве. Далее для входящего пакета также вычисляется хеш-функция тем же методом, и определяется, к какому из узлов на данном пространстве вычисленное значение ближе. Такой подход является достаточно устойчивым к изменениям в системе, однако также не лишён недостатков, к которым относятся достаточно ощутимая сложность

реализации, а также необходимость виртуальных узлов для более равномерного распределения.

Рассмотренные особенности методов хеширования позволяют сделать вывод, что наиболее предпочтительным вариантом в контексте задачи балансировки нагрузки в сети представляется консистентное хеширование с использованием комбинированного метода вычисления хеш-функции. Такое сочетание является достаточно удобным для первичной реализации, позволяет сделать балансировку устойчивой к изменениям в системе, а простота вычисления хеш-функции снижает время обработки пакета. Также, несмотря на недостатки, достаточно перспективным выглядит вариант использования криптографических функций вместо комбинированного метода вычисления, ввиду его активного применения в реальном сетевом оборудовании. В дальнейшем возможно проведение эксперимента с целью выявления преимущества одного метода над другим в условиях практического решения поставленной задачи.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Балансировка нагрузки: основные алгоритмы и методы / [Электронный ресурс] // Habr – URL: <https://habr.com/ru/companies/selectel/articles/250201/> (дата обращения: 29.10.2024).
2. Протасов С.С., Белоусов С.М., Тормасов А. Г. Математическая модель и метод построения сервиса балансирования нагрузки между серверами асимметричной фермы [Текст] / С.С. Протасов, С.М. Белоусов, А.Г. Тормасов // ИССЛЕДОВАНО В РОССИИ. – 2024. – № 4. – С. 21-66
3. RFC 2992: Analysis of an Equal-Cost Multi-Path Algorithm [Электронный ресурс]. – URL: <https://datatracker.ietf.org/doc/html/rfc2992> (дата обращения: 10.04.2025).
4. ECMP и балансировка нагрузки в датацентрах [Электронный ресурс] – URL: <https://linkmeup.ru/blog/903/> (дата обращения: 10.04.2025).
5. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке Си / Пер. с англ. – М.: Вильямс, 2015. – 816 с.
6. Таненбаум Э. Распределённые системы. Принципы и парадигмы: пер. с англ. / Э. Таненбаум, М. Стин. – СПб.: Питер, 2017. – 848 с.
7. Karger D., Lehman E., Leighton T., Panigrahy R., Levine M., Lewin D. Consistent Hashing and Random Trees: Distributed Caching

Protocols for Relieving Hot Spots on the World Wide Web // In: ACM Symposium on Theory of Computing. – 1997. – P. 654-663.

УДК 004.896

ИСАЙКИН А.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Рассматриваются искусственные нейронные сети (ИНС), включая их структуры, методы обучения, ключевые вызовы, историческое развитие, достижения и направления, ориентированные на повышение их эффективности, адаптивности и устойчивости.

Искусственные нейронные сети (ИНС) представляют собой мощный инструмент в области искусственного интеллекта, который нашел широкое применение в различных сферах человеческой деятельности. С их помощью решаются задачи анализа данных, распознавания образов, обработки естественного языка, управления роботами и многие другие. Несмотря на значительные достижения, перед исследователями все еще стоят задачи, связанные с улучшением эффективности нейросетей, расширением их возможностей и повышением уровня понимания их работы.

История искусственных нейронных сетей. История искусственных нейронных сетей насчитывает более полувека. В начале 20 века ученые начали изучать функции нейронов и их взаимосвязь в головном мозге. Важным моментом в этой области стала работа Уоррена МакКаллока и Уолтера Питтса, опубликованная в 1943 году, в которой была представлена первая математическая модель искусственного нейрона. В 1958 году Фрэнк Розенблатт представил персептрон – простейшую форму нейронной сети, способную обучаться на основе обратной связи. Однако в 1960-е годы исследования в области ИНС пережили спад из-за ограниченных вычислительных ресурсов и недостаточного понимания их возможностей. В 1980-х годах интерес к нейросетям возрос, частично благодаря развитию вычислительной техники и алгоритмов обучения, таких как метод обратного распространения ошибки. С начала 21 века, благодаря развитию глубокого обучения и использованию графических процессоров, ИНС достигли новых высот в области

распознавания образов, обработки естественного языка и других приложений [1].

Основные принципы работы ИНС. Искусственные нейронные сети моделируют работу биологических нейронных сетей, которые состоят из нейронов, связанных между собой синапсами. Основные компоненты ИНС включают нейроны (или узлы), веса связей и функции активации. Нейрон принимает входные сигналы, обрабатывает их и генерирует выходной сигнал. Веса связей определяют силу влияния входных сигналов на выход нейрона, а функции активации преобразуют взвешенную сумму сигналов в выходной сигнал, вводя нелинейность в модель. Это позволяет нейросетям решать сложные задачи [3].

Процесс работы искусственной нейронной сети включает несколько ключевых этапов, каждый из которых играет важную роль в обработке информации и получении результатов. На первом этапе, называемом прямым распространением, входные данные поступают в сеть и передаются через несколько слоев нейронов. Каждый нейрон на каждом слое получает входной сигнал, который подвергается обработке. Веса связей между нейронами определяют силу воздействия одного нейрона на другой. Для каждого нейрона вычисляется взвешенная сумма входных сигналов, которая затем проходит через функцию активации. Эта функция отвечает за преобразование результата в выходное значение, которое передается дальше по сети. В зависимости от типа задачи функции активации могут быть линейными или нелинейными. На выходе первого слоя получают значения, которые становятся входами для следующего слоя [4].

После того как выходные данные сети получены, наступает этап обратного распространения ошибки. На этом этапе вычисляется разница между реальным результатом и ожидаемым, то есть ошибка. Эта ошибка распространяется обратно через все слои сети, начиная с выходного и двигаясь к входному. На каждом шаге определяется, как ошибка каждого нейрона влияет на ошибку предыдущих нейронов. В процессе обратного распространения на основе вычисленной ошибки корректируются веса связей между нейронами, чтобы минимизировать общую ошибку и улучшить производительность сети. Этот процесс позволяет нейросети адаптироваться и улучшать точность своих предсказаний [3].

Архитектуры и типы искусственных нейронных сетей. Искусственные нейронные сети могут иметь различные архитектуры, каждая из которых предназначена для решения определённых задач.

Однослойные нейронные сети, или перцептроны, представляют собой простейшую форму нейронной сети, состоящую из одного слоя нейронов. Эти сети обычно используются для бинарной классификации и решения простых задач, где данные могут быть разделены линейно. Однако однослойные нейронные сети ограничены в своих возможностях и не могут эффективно решать более сложные задачи [2].

Многослойные нейронные сети, или многослойные перцептроны (MLP), включают несколько слоёв нейронов, включая один или несколько скрытых слоёв. Это позволяет моделям изучать более сложные зависимости в данных, что делает их подходящими для задач, требующих обработки более сложных паттернов, таких как распознавание образов, классификация или регрессия. С помощью алгоритма обратного распространения ошибки многослойные сети могут корректировать свои веса и улучшать результат на протяжении обучения [1].

Рекуррентные нейронные сети (RNN) имеют уникальную архитектуру, где выход нейронов на одном временном шаге передается обратно в качестве входа на следующем шаге. Это позволяет таким сетям учитывать временные зависимости, что делает их особенно полезными для обработки последовательных данных, таких как текст, речь или временные ряды. Однако стандартные RNN сталкиваются с проблемой исчезающего градиента, что ограничивает их способность захватывать долгосрочные зависимости. Для решения этих проблем были разработаны более сложные архитектуры, такие как LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Units), которые используют механизмы, позволяющие более эффективно извлекать долгосрочные зависимости из последовательностей.

Сверточные нейронные сети (CNN) – это специализированные архитектуры для обработки изображений и других структурированных данных. В этих сетях используется операция свертки, которая позволяет автоматически извлекать признаки, такие как края, текстуры и формы, из изображений. CNN эффективно работают с изображениями различных разрешений, что делает их основой для многих современных решений в области компьютерного зрения. Эти сети могут быть использованы не только для распознавания объектов, но и для задач, связанных с анализом видео или распознаванием речи, где важно выявление пространственных или временных признаков.

Кроме того, существуют и другие типы нейронных сетей, такие как автоэнкодеры, которые применяются для сжатия данных или удаления шума, а также генеративные состязательные сети (GANs),

где две сети обучаются генерировать и отличать фальшивые данные, например изображения. Трансформеры, использующие механизм внимания, заменили рекуррентные сети в задачах обработки последовательностей, таких как машинный перевод и обработка текста, благодаря своей способности фокусироваться на различных частях входных данных.

Каждая из этих архитектур обладает своими особенностями и используется в зависимости от специфики решаемой задачи и типа обрабатываемых данных. Развитие и усовершенствование нейронных сетей позволяют значительно улучшить результаты в различных областях, от компьютерного зрения до обработки естественного языка.

Применение ИНС в различных областях. Искусственные нейронные сети активно применяются в различных областях. В обработке естественного языка они используются для автоматического перевода, анализа тональности текста и генерации текста. В области компьютерного зрения – для распознавания объектов, сегментации изображений и распознавания лиц. В робототехнике нейросети применяются для автоматического управления, разработки автономных систем и роботизированных технологий. В медицине ИНС используются для анализа медицинских изображений, прогнозирования заболеваний и создания индивидуальных схем лечения. В финансовой аналитике они помогают в прогнозировании рыночных трендов и обнаружении мошенничества.

Однако, несмотря на успехи, искусственные нейронные сети сталкиваются с рядом вызовов. Одной из основных проблем является интерпретируемость моделей. Современные нейросети зачастую представляют собой "черные ящики", что затрудняет объяснение их решений, особенно в критических областях, таких как медицина и финансы. Другим вызовом является обучение нейросетей при недостатке данных. В некоторых областях, например, в медицине, сбор данных может быть дорогостоящим или труднодоступным, что требует разработки новых методов обучения с использованием малых объемов данных. Также нейросети могут быть чувствительны к изменениям в данных или окружающей среде, что делает необходимым создание методов, которые позволят адаптировать сети к изменяющимся условиям [5].

Перспективы и вызовы развития ИНС. С развитием ИНС также встают важные вопросы этики и прозрачности, включая проблемы конфиденциальности данных, справедливости алгоритмов и их влияния на общество. Это поднимает необходимость создания

этичных и прозрачных методов использования ИНС, что становится важным направлением в их развитии [4].

Современные исследования ИНС охватывают несколько ключевых направлений. Одним из них является обучение с недостаточным надзором, когда нейросеть обучается на неразмеченных данных или на малых объемах размеченных данных. Также активно развиваются методы обучения с подкреплением, которые позволяют системам самостоятельно обучаться на основе взаимодействия с окружающей средой. В последние годы набирает популярность мультизадачное обучение, которое позволяет нейросетям одновременно решать несколько связанных задач, что улучшает их обобщающие способности. Еще одним важным направлением является использование графовых нейросетей, которые эффективно работают с данными, представленными в виде графов, что открывает новые возможности для анализа сложных структурированных данных [6]. В области генеративных моделей продолжаются исследования автоэнкодеров, которые могут генерировать новые данные, например, изображения или текст, на основе обучающих примеров. Все эти направления направлены на повышение универсальности, гибкости и эффективности искусственных нейронных сетей.

Искусственные нейронные сети (ИНС) продолжают развиваться и находят все больше применений в самых различных областях. Одной из ключевых особенностей нейронных сетей является их способность к обучению на больших объемах данных, что позволяет использовать их для обработки информации, которая слишком сложна или объемна для традиционных алгоритмов. Современные нейросети способны эффективно выявлять скрытые закономерности в данных, что делает их идеальными инструментами для анализа и предсказания в таких областях, как экономика, маркетинг, медицина и науки о данных.

Одной из важнейших причин, по которой нейросети становятся столь востребованными, является их способность к решению задач, для которых не существует стандартных алгоритмов. Например, в области компьютерного зрения нейросети используются для распознавания объектов на изображениях, а также для более сложных задач, таких как автономное вождение автомобилей, где необходимо учитывать множество факторов и принимать решения в реальном времени. В области обработки речи нейросети позволяют значительно улучшить точность систем распознавания речи, таких как голосовые помощники, что облегчает взаимодействие человека с компьютером.

Революционные достижения в области ИНС во многом стали возможны благодаря развитию вычислительных технологий. Современные графические процессоры (GPU) позволяют значительно ускорить процесс обучения нейронных сетей, что в свою очередь открывает новые горизонты для решения более сложных задач. В частности, использование GPU в области глубокого обучения позволило значительно улучшить точность нейросетевых моделей в таких областях, как распознавание изображений и обработка речи. Базы данных, такие как ImageNet и COCO, с огромным количеством размеченных данных, позволяют обучать нейросети на реальных примерах, что повышает их эффективность и точность. Большие вычислительные мощности также позволяют разрабатывать более сложные архитектуры нейросетей, такие как трансформеры, которые обеспечивают прорывы в обработке естественного языка.

Одной из тенденций в развитии ИНС является увеличение их глубины. Глубокие нейронные сети (DNN) позволяют моделировать более сложные зависимости и взаимосвязи в данных, чем традиционные нейросети. Это особенно важно для решения таких задач, как распознавание объектов на изображениях, где необходимо учитывать сложные паттерны, или в обработке текста, где важно учитывать контекст и семантическую нагрузку каждого слова в предложении. Однако, чем глубже сеть, тем сложнее ее обучение и тем больше требуется данных для эффективного обучения. Этот процесс также требует значительных вычислительных мощностей и времени.

Совсем недавно стали активно исследоваться и развиваться методы, которые могут улучшить эффективность обучения ИНС, такие как методы оптимизации, регуляризации и улучшения стабильности обучения. Одним из таких методов является использование трансформеров, которые эффективно работают с последовательными данными и значительно улучшили результаты в задачах, связанных с обработкой текста. Они были впервые предложены для обработки текста в задаче машинного перевода, но позже нашли применение и в других областях, включая создание изображений, генерирование текста и даже в биоинформатике для анализа последовательностей ДНК.

Однако, несмотря на значительный прогресс в разработке и применении ИНС, остаются вопросы, требующие дальнейших исследований. Одним из них является проблема переобучения. Нейросети могут эффективно обучаться на большом объеме данных, но при недостаточной диверсификации данных или отсутствии регуляризации они могут начать запоминать шум в данных, что

снижает их способность к обобщению. Для решения этой проблемы разработаны методы, такие как dropout и batch normalization, которые помогают улучшить обобщающие способности моделей и предотвращают переобучение.

В последние годы также наблюдается рост интереса к объяснимым нейронным сетям. Сложность и "черный ящик" современных ИНС ограничивают их использование в таких сферах, как медицина или юриспруденция, где важна объяснимость решений. Работы в этой области нацелены на то, чтобы нейросети стали более прозрачными, и их решения можно было бы понять и объяснить человеку. Это включает методы визуализации, интерпретации и разработки более простых моделей, которые сохраняют точность, но при этом легко объясняются.

Одним из направлений будущих исследований является повышение устойчивости нейросетей. Существующие нейросети могут быть уязвимы к различным атакам, например, к так называемым атакам с добавлением шума, которые могут приводить к ошибочным выводам, даже если изменения в данных неочевидны для человека. Это вызывает опасения в области безопасности и надежности ИНС, особенно в критически важных приложениях, таких как автомобили с автопилотом или системы распознавания лиц.

Прогнозирование будущего ИНС связано с развитием нескольких ключевых направлений. Одним из них является создание нейросетей, которые могут учиться на малых объемах данных. В настоящее время для обучения моделей ИНС часто требуется огромное количество размеченных данных, что может быть дорогостоящим и времязатратным процессом. Однако нейросети, обучающиеся на малых объемах данных, могут значительно снизить эти затраты, что сделает технологии доступными для более широкого круга задач.

В будущем ИНС также могут стать важной частью персонализированных технологий. Например, в медицине нейросети могут использоваться для создания индивидуальных планов лечения, предсказания реакции на лекарства и прогнозирования развития заболеваний. В образовательных технологиях нейросети могут быть использованы для создания адаптивных систем обучения, которые подстраиваются под потребности каждого студента, предлагая наиболее подходящие задания и материалы.

Заключение. Искусственные нейронные сети продолжают развиваться, улучшать свои возможности и находить все новые области применения. Тем не менее, их развитие требует дальнейших

исследований и решений множества вопросов, таких как объяснимость, безопасность и устойчивость моделей, а также оптимизация обучения с учетом ограниченных данных. При этом нейросети становятся важным инструментом в решении как практических, так и теоретических задач, и в будущем их потенциал будет только расширяться, открывая новые горизонты для науки и технологий.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Кадурын А.А., Николаев Н.И., Граничин О.Н. Глубокое обучение. Бином. – Спб.: Питер, 2018. – 480 с.
2. Ивахненко А.Г. Общая теория систем. – Киев: Техника, 1971. – 372 с.
3. Карабутов П.М., Степаненко А.С. Искусственные нейронные сети: учебное пособие. – Спб.: БХВ-Петербург, 2017. – 350 с.
4. Разумный В.Ф. Нейрокомпьютерные технологии. – М.: Вильямс, 2002. – 448 с.
5. Галустян А.Г. Моделирование и исследование искусственных нейронных сетей. – М.: Издательский дом «Кодекс», 2003. – 380 с.
6. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. – М.: ДМК Пресс, 2022. – 652 с.

УДК 004.896

КАСЕЛЬСКИЙ М.П.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ

В статье рассматриваются принципы обучения нейронных сетей, используемые алгоритмы, сложности и пути их преодоления.

Обучение нейронных сетей представляет собой активно развивающееся направление в области искусственного интеллекта, привлекающее внимание исследователей и практиков со всего мира. С развитием вычислительных технологий и доступностью данных возрос интерес к методам глубокого обучения, основанным на нейронных сетях. Эти методы позволяют с высокой точностью решать разнообразные задачи в области компьютерного зрения, обработки естественного языка, обнаружения аномалий, рекомендации контента и многих других.

В настоящее время обучение нейронных сетей стало ключевым инструментом для решения сложных задач, которые ранее считались неразрешимыми с использованием традиционных методов. Этот интересный феномен вызван не только возможностью решать задачи с высокой точностью, но и способностью нейронных сетей к автоматическому извлечению признаков из данных и их адаптации к различным условиям.

Первые нейронные сети были вдохновлены биологическими нейронными сетями и состояли из простых элементов, называемых перцептронами. Однако развитие нейронных сетей приостановилось из-за ограничений вычислительных ресурсов и недостаточного объема данных. В 1980-х годах возродился интерес к нейронным сетям благодаря работе Румельхарта, Хинтона и Уильямса, в которой было показано, что многослойные нейронные сети способны решать сложные задачи классификации [1].

Обзор основных моделей нейронных сетей. Среди основных моделей нейронных сетей можно выделить перцептроны, многослойные нейронные сети (MLP), сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN) и их вариации (LSTM, GRU). Каждая из этих моделей имеет свои уникальные характеристики и применяется для решения определенных задач.

Принципы работы и применения нейронных сетей. Нейронные сети успешно применяются в различных областях, таких как компьютерное зрение (детекция объектов, сегментация изображений), обработка естественного языка (машинный перевод, анализ тональности текста), рекомендательные системы (персонализированные рекомендации), медицина (диагностика заболеваний на основе медицинских изображений и анализа данных пациентов) и другие.

Функции активации. Функции активации играют ключевую роль в работе нейронных сетей, поскольку они определяют нелинейность модели и ее способность моделировать сложные зависимости в данных. В процессе обучения нейронные сети принимают входные данные, которые передаются через слои нейронов, каждый из которых применяет функцию активации к входным данным.

Линейная функция активации: Одной из самых простых функций активации является линейная функция, которая просто передает входные данные без изменений. Однако использование линейной функции активации ограничивает возможности нейронной сети в моделировании сложных нелинейных зависимостей.

Сигмоидальные функции активации: Сигмоидальные функции, такие как логистическая функция и гиперболический тангенс, широко использовались в прошлом, но в настоящее время их применение ограничено из-за некоторых недостатков, таких как насыщение градиента при обратном распространении ошибки.

Функции активации играют важную роль в обучении нейронных сетей, и выбор оптимальной функции активации является одним из ключевых аспектов проектирования и настройки моделей.

Оптимизационные алгоритмы. Оптимизационные алгоритмы играют ключевую роль в процессе обучения нейронных сетей, поскольку они отвечают за коррекцию параметров модели таким образом, чтобы минимизировать функцию потерь. Среди популярных алгоритмов оптимизации выделяются градиентный спуск, его стохастическая версия и алгоритм Adam [2]. Последний продемонстрировал высокую эффективность при обучении глубоких нейронных сетей благодаря адаптивной настройке параметров [3].

Градиентный спуск. Градиентный спуск является классическим методом оптимизации, который использует градиент функции потерь по отношению к параметрам модели для обновления их значений. В процессе обучения нейронной сети градиентный спуск находит направление наискорейшего убывания функции потерь и корректирует параметры модели в этом направлении.

Стохастический градиентный спуск. Стандартный градиентный спуск может быть неэффективным на больших наборах данных из-за необходимости вычисления градиента по всему набору данных. Стохастический градиентный спуск является вариантом градиентного спуска, который обновляет параметры модели на основе градиента, рассчитанного по подмножеству случайно выбранных данных. Это позволяет более быстро обновлять параметры модели и ускоряет обучение.

Алгоритм Adam. Алгоритм Adam является одним из наиболее популярных и эффективных оптимизационных алгоритмов для обучения нейронных сетей. Он комбинирует идеи из градиентного спуска и стохастического градиентного спуска, а также автоматически адаптирует скорость обучения для каждого параметра на основе первого и второго моментов градиента.

Сравнительный анализ оптимизационных алгоритмов. Сравнительный анализ различных оптимизационных алгоритмов проводится на основе их скорости сходимости, способности избегать застревания в локальных минимумах, устойчивости к шумам в данных

и других факторов. Выбор оптимального оптимизационного алгоритма зависит от характеристик конкретной задачи и структуры модели.

Оптимизационные алгоритмы являются неотъемлемой частью обучения нейронных сетей и играют ключевую роль в процессе настройки параметров модели для достижения оптимальных результатов.

Функции потерь. Функции потерь (или функции ошибки) являются мерой расхождения между предсказанными значениями модели и их истинными значениями на обучающих данных. Выбор подходящей функции потерь зависит от типа задачи, которую необходимо решить, и особенностей данных.

Функция потерь для задач регрессии. В задачах регрессии, где целевая переменная является непрерывной, часто используется среднеквадратичная ошибка (MSE, Mean Squared Error) или средняя абсолютная ошибка (MAE, Mean Absolute Error) в качестве функции потерь. MSE измеряет среднеквадратичное отклонение между предсказанными и истинными значениями, в то время как MAE измеряет среднее абсолютное отклонение.

Функция потерь для задач классификации. В задачах классификации, где целевая переменная является дискретной (например, метки классов), часто используется кросс-энтропия или логистическая функция потерь. Кросс-энтропия измеряет расхождение между распределениями вероятностей предсказанных и истинных классов. Для многоклассовой классификации часто применяется категориальная кросс-энтропия, а для бинарной классификации - бинарная кросс-энтропия.

Другие функции потерь. В зависимости от конкретной задачи могут применяться и другие функции потерь. Например, в задачах обнаружения аномалий может использоваться функция потерь, основанная на расстоянии между входными данными и их реконструкцией, или специальные функции потерь, направленные на минимизацию ошибок при редких событиях.

Выбор функции потерь. Выбор подходящей функции потерь зависит от целей моделирования и особенностей данных. Например, при обучении нейронных сетей для задачи классификации с несбалансированными классами может быть предпочтительнее использовать взвешенные функции потерь для учета различной важности классов.

Функции потерь играют ключевую роль в процессе обучения нейронных сетей, поскольку они определяют, как модель оценивает свои предсказания и корректирует параметры в процессе обучения.

Правильный выбор функции потерь способствует достижению оптимальных результатов в задаче обучения.

Аугментация данных. Аугментация данных является важной техникой в обучении нейронных сетей, которая заключается в генерации новых обучающих примеров путем применения различных преобразований к исходным данным. Аугментация данных помогает значительно увеличить обобщающую способность модели, особенно в условиях ограниченных обучающих выборок [1].

Техники аугментации данных

1. Повороты и отражения Повороты изображений на разные углы и отражения относительно вертикальной или горизонтальной оси позволяют получить новые варианты изображений, сохраняя при этом семантику данных.

2. Масштабирование и обрезка Изменение масштаба изображений или их обрезка позволяют создать различные варианты объектов в разных масштабах и ракурсах.

3. Добавление шума: Внесение случайного шума в изображения или данные может сделать модель более устойчивой к вариациям в данных и помочь обобщить на новые условия.

4. Изменение яркости и контраста Вариации яркости и контраста изображений помогают модели стать более устойчивой к различным условиям освещения.

Применение аугментации данных. Аугментация данных часто применяется в обучении нейронных сетей, особенно в задачах компьютерного зрения, где доступ к большому объему размеченных данных может быть ограничен. Эта техника помогает улучшить производительность модели на тестовых данных за счет увеличения разнообразия данных, на которых модель обучается.

Обучение на больших данных и использование техник передачи обучения. Обучение нейронных сетей на больших объемах данных является ключевым аспектом в достижении высокой производительности моделей. Тем не менее, собирать достаточное количество размеченных данных для обучения модели может быть сложно и затратно. В таких случаях использование техник передачи обучения (transfer learning) может значительно упростить процесс обучения и улучшить качество модели.

Обучение на больших данных. Обучение нейронных сетей на больших данных имеет ряд преимуществ, таких как более точные оценки параметров модели и лучшая обобщающая способность. Большие наборы данных позволяют модели лучше улавливать разнообразие признаков и обобщать на новые данные. Однако

обработка и обучение модели на больших объемах данных требует значительных вычислительных ресурсов и времени.

Использование техник передачи обучения. Техники передачи обучения позволяют использовать заранее обученные модели, обученные на больших наборах данных, для решения новых задач. Это может быть особенно полезно, когда доступных данных для новой задачи ограничено. Техники передачи обучения включают в себя:

Fine-tuning (настройка готовой модели): после загрузки предварительно обученной модели можно заморозить определенные слои и настроить остальные слои на новых данных для конкретной задачи. Это позволяет использовать знания, полученные предварительным обучением, и приспособить их к новому контексту.

Feature extraction (извлечение признаков): В этом случае можно использовать предварительно обученную модель для извлечения признаков из данных и затем использовать эти признаки для обучения новой модели на новых данных. Это особенно полезно в случае небольших объемов данных.

Обучение на больших наборах данных и использование методов *transfer learning*, таких как *fine-tuning* и *feature extraction*, позволяет достигать высокой точности даже в условиях ограниченных ресурсов [2].

Модели нейронных сетей. Полносвязные нейронные сети (*Fully Connected Neural Networks, FCNN*). Полносвязные сети состоят из последовательных слоев нейронов, где каждый нейрон связан со всеми нейронами предыдущего и последующего слоев. Они являются базовой архитектурой, но из-за большого числа параметров склонны к переобучению и плохо масштабируются на высокоразмерные данные (например, изображения).

Сверточные нейронные сети (Convolutional Neural Networks, CNN). CNN применяются преимущественно для обработки изображений, видео и других данных с пространственной структурой. Их ключевые особенности:

- Сверточные слои – используют фильтры для выделения локальных признаков.
 - Пулинг-слои (*max-pooling, average-pooling*) – уменьшают размерность данных, сохраняя важные признаки.
 - Иерархическое представление признаков – от простых (края, текстуры) к сложным (объекты, сцены).
- Популярные архитектуры: LeNet-5, AlexNet, ResNet, EfficientNet.

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN). RNN предназначены для обработки последовательностей (текст,

временные ряды, речь). Их отличительная черта – наличие скрытого состояния, передающего информацию между шагами.

Трансформеры (Transformers). Трансформеры используют механизм внимания (self-attention), что позволяет эффективно обрабатывать длинные последовательности без рекуррентных связей. Они стали стандартом в NLP (BERT, GPT), а также применяются в компьютерном зрении (Vision Transformers, ViT).

Автокодировщики (Autoencoders). Автокодировщики обучаются сжимать данные в латентное пространство и восстанавливать их.

Нейронные сети с остаточными связями (Residual Networks, ResNet). ResNet решают проблему затухания градиентов в глубоких сетях за счет skip-connections, позволяющих обучать сети с сотнями слоев.

Графовые нейронные сети (Graph Neural Networks, GNN). GNN работают с данными, представленными в виде графов (социальные сети, молекулы), агрегируя информацию от соседних узлов.

Заключение. Обучение нейронных сетей представляет собой сложный, но хорошо формализованный процесс. Успешная настройка параметров позволяет моделям достигать впечатляющих результатов на разнообразных задачах. Основу обучения составляет градиентный спуск с обратным распространением ошибки, дополненный современными практиками регуляризации, оптимизации и архитектурными улучшениями. На примере классификации изображений видно, что даже простая модель способна решать сложную задачу при корректной реализации. Постоянное развитие этой области делает нейросети всё более доступными и эффективными инструментами в руках исследователей и инженеров.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. – М.: ДМК Пресс, 2022. – 652 с.
2. Иванишев М.Н. Обучение нейронных сетей. – СПб.: БХВ-Петербург, 2019. – 320 с.
3. Стрижов В.В., Ветров Д.П. Обучение нейросетей. – М.: Бином. Лаборатория знаний, 2019. – 288 с.

УДК 004.8

КАШИРИНА А.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

АВТОМАТИЗИРОВАННОЕ А/В-ТЕСТИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАШИННОГО ОБУЧЕНИЯ

Рассматривается процесс автоматизации А/В-тестирования Web-приложений с помощью машинного обучения.

А/В-тестирование – это проверенный метод, который обычно проводится с использованием традиционного подхода к статистическим выводам, основанного на проверке гипотезы. Машинное обучение – это область искусственного интеллекта, которая позволяет компьютерам обучаться на основе данных и делать прогнозы без явного программирования. Алгоритмы машинного обучения анализируют входные данные, выявляют закономерности и используют их для принятия решений.

Несмотря на то, что А/В-тестирование и машинное обучение используются по-разному, они могут дополнять и улучшать друг друга. В контексте А/В-тестирования машинное обучение помогает ускорить эксперименты, автоматически оптимизировать распределение трафика и персонализировать тестирование. Это значительно снижает затраты и повышает эффективность тестов, позволяя находить лучшие варианты интерфейсов и контента.

Что такое А/В-тестирование?

А/В-тестирование – это разновидность сплит-тестирования, которая используется для улучшения определенных элементов или переменных путем анализа вовлеченности пользователей или аудитории.

Традиционно это делается с помощью случайного разделения трафика: одна группа пользователей (контрольная) видит «версию А», а другая группа (тестовая) – «версию В».

Итоговые показатели вовлеченности определяют, какую из представленных версий предпочитает пользователь, что позволяет доработать кампанию для достижения наилучшего эффекта. Затем внедряется наиболее предпочтительная версия с целью повышения качества взаимодействия с будущими пользователями. В качестве метрики обычно используется коэффициент конверсии для достижения определенной цели, таких как доля пользователей, которые нажимают на кнопку или совершают покупку.

Достаточно часто для А/В-тестирования используются статистические методы, так как они позволяют объективно оценить различия между группами и подтвердить или опровергнуть гипотезы. Аналитики собирают данные о поведении пользователей, участвующих в эксперименте. Затем эти данные структурируются в таблицы и визуализируются в виде графиков, что помогает наглядно увидеть основные тенденции. Далее проводятся статистические тесты, например, t-тесты или дисперсионный анализ, в зависимости от того, какие гипотезы необходимо проверить.

Данные методы достаточно эффективны, поскольку позволяют получить надежные и достоверные результаты, однако они имеют ряд ограничений. Во-первых, такие методы могут быть достаточно сложными и требовать значительного времени на обработку данных и проведение расчётов. Во-вторых, проведение статистических тестов требует от аналитиков определённого опыта и знаний, чтобы правильно интерпретировать результаты и избежать ошибок.

Кроме того, статистические методы имеют и другие ограничения. Например, их точность может снижаться в случае нестабильных данных, если выборка не является случайной или если в процессе сбора информации есть какие-либо искажения.

Зачем использовать машинное обучение?

На помощь приходит машинное обучение, которое позволяет значительно улучшить процесс анализа данных. Вместо традиционных статистических методов можно использовать алгоритмы, способные обрабатывать большие объёмы информации, учитывать различные переменные и находить скрытые зависимости, которые могут быть неочевидными при обычном анализе.

Также машинное обучение помогает не только автоматизировать процесс, но и оптимизировать А/В-тесты:

1. при запуске множества тестов одновременно повышается риск получения ложноположительных результатов, но алгоритмы машинного обучения корректируют р-значения, учитывая множественные сравнения;
2. модели машинного обучения могут быстрее выявить более оптимальный вариант, что значительно снижает время проведения эксперимента;
3. в традиционном А/В-тестировании трафик распределяется равномерно, даже если одна из версий явно лучше, тогда как методы машинного обучения динамически направляют больше пользователей к более успешному варианту.

Подготовка данных

Перед тем как приступить к созданию модели машинного обучения, необходимо правильно подготовить данные, поскольку ошибки на этом этапе могут привести к искажению результатов анализа и неверным выводам. Данные должны быть организованы так, чтобы можно было с легкостью определить, какая из версий показывалась пользователю и какие действия он совершил.

Рассмотрим структуру данных, которая может использоваться для обучения модели. Результаты A/B-теста могут включать следующие параметры, приведенные в таблице 1.

Таблица 1 – Результаты A/B-теста

Идентификатор пользователя	Версия сайта	Количество кликов	Конверсии	Время на сайте
1	A	2	0	60
2	B	7	1	180
3	A	6	1	120
4	B	4	0	90

В этой таблице представлены основные показатели взаимодействия пользователей с сайтом: идентификатор пользователя, версия сайта, количество кликов, конверсии и время на сайте.

Использование Python для анализа

Python является мощным инструментом для анализа A/B-тестов благодаря широкому набору библиотек для машинного обучения и анализа данных.

Загрузка и подготовка данных

Перед анализом нужно загрузить данные и выполнить их предварительную обработку. Например, у нас имеется CSV-файл с результатами проведенных тестов. Для начала необходимо импортировать библиотеки Python и сам CSV-файл.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import numpy as np
```

Здесь используются несколько ключевых библиотек, а именно:

1. pandas позволяет работать с табличными данными, загружать их из файлов, фильтровать, преобразовывать и анализировать. Это удобно, поскольку у нас есть CSV-файл с результатами тестов, который нужно обработать перед анализом;

2. scikit-learn – библиотека для машинного обучения, и в данном случае мы используем модель логистической регрессии (LogisticRegression), которая поможет определить вероятность

конверсии в зависимости от теста. Также нам нужен `train_test_split` для разделения данных на обучающую и тестовую выборки, что позволит проверить, насколько хорошо работает модель;

`Numpy` используется для работы с массивами и числовыми вычислениями, что очень важно при обработке данных.

После загрузки данных из файла важно убедиться, что они корректны. Для этого можно вывести первые несколько строк с помощью `print(test_data.head())`, а также проверить, нет ли пропущенных значений (`test_data.isnull().sum()`).

Вычисление конверсии

Прежде чем приступить к построению моделей машинного обучения, необходимо провести предварительный анализ данных, чтобы выявить основные закономерности и оценить эффективность вариантов. Одним из показателей является конверсия, которая отражает, какой процент пользователей выполнил целевое действие.

```
conversion_A = test_data[test_data['Version'] ==  
'A']['Conversions'].mean()  
conversion_B = test_data[test_data['Version'] ==  
'B']['Conversions'].mean()  
print(f"Конверсия для версии A: {conversion_A}")  
print(f"Конверсия для версии B: {conversion_B}")
```

Создание и обучение модели

После проведения предварительного анализа данных можно перейти к построению модели машинного обучения. Одним из простых и эффективных методов является логистическая регрессия, которая позволяет предсказать вероятность конверсии.

Сначала преобразуем версии сайта в числовой формат:

```
# Преобразуем версии A и B в бинарные значения (0 и 1)  
test_data['Version_Binary'] = (test_data['Version']  
== 'B').astype(int)
```

```
# Создаем X (признаки) и y (целевая переменная)  
X = test_data[['Version_Binary']]  
y = test_data['Conversions']
```

Для оценки качества модели разделим данные на тренировочный и тестовый наборы в соотношении 80/20:

```
# Разделяем данные на тренировочный и тестовый  
наборы
```

```
X_train, X_test, y_train, y_test =  
train_test_split(X, y, test_size=0.2, random_state=42)
```

Используем логистическую регрессию для предсказания вероятности конверсии:

```
# Создаем и обучаем модель логистической регрессии  
model = LogisticRegression()
```

```
model.fit(X_train, y_train)
```

Проверим, насколько хорошо модель предсказывает результаты, и получим предсказанные вероятности конверсии:

```
# Оцениваем модель
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
# Получаем вероятности конверсии для обеих версий
version_A_prob =
model.predict_proba(np.array([[0]])) [0] [1]
version_B_prob =
model.predict_proba(np.array([[1]])) [0] [1]
print("\nРезультаты логистической регрессии:")
print(f"Точность на тренировочных данных:
{train_score:.3f}")
print(f"Точность на тестовых данных:
{test_score:.3f}")
print(f"\nВероятность конверсии для версии А:
{version_A_prob:.3f}")
print(f"Вероятность конверсии для версии В:
{version_B_prob:.3f}")
if version_A_prob > version_B_prob:
    print("\nВерсия А имеет более высокую
вероятность конверсии")
else:
    print("\nВерсия В имеет более высокую
вероятность конверсии")
```

Итак, мы создали базовую модель логистической регрессии, использующую только один признак – версию сайта. Можно использовать и более сложные алгоритмы, такие как случайные леса или нейронные сети, которые могут выявлять сложные зависимости в данных и давать более точные предсказания.

Пример алгоритма случайного леса

В отличие от логистической регрессии, случайный лес – это метод, состоящий из множества деревьев решений, что позволяет модели лучше выявлять сложные зависимости в данных и повышает точность прогнозов.

```
from sklearn.ensemble import RandomForestClassifier
# Разделяем данные на обучающую и тестовую выборки
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2, random_state=42)
# Создаем и обучаем модель случайного леса
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)
# Оцениваем точность модели
```

```
train_score = accuracy_score(y_train,  
model.predict(X_train))  
test_score = accuracy_score(y_test,  
model.predict(X_test))  
print(f"Точность на тренировочных данных:  
{train_score:.3f}") print(f"Точность на тестовых данных:  
{test_score:.3f}")
```

В данном случае модель случайного леса может обеспечить более высокую точность по сравнению с логистической регрессией, особенно при наличии нелинейных зависимостей в данных.

Какую модель выбрать?

Выбор модели машинного обучения для A/B-тестирования зависит от сложности данных и целей анализа. В простых случаях, когда данных немного, можно использовать традиционные статистические методы, такие как Z-тест или t-тест, чтобы определить, есть ли значимые различия между вариантами. Данные методы быстрые и удобные, но они не учитывают сложные зависимости и дополнительные факторы.

Если необходимо не только сравнить результаты, но и понять, какие факторы влияют на конверсию, то подойдет логистическая регрессия. Она позволяет прогнозировать вероятность успеха и объяснять, почему одна версия работает лучше другой. Однако если данных больше, и они включают множество признаков, лучше использовать случайный лес. Данная модель способна выявлять сложные взаимосвязи и подходит для более детального анализа.

Оптимизация A/B-тестов и адаптивные модели

Традиционное A/B-тестирование требует фиксированного разделения аудитории, что может приводить к неэффективному использованию трафика. Оптимизация процесса с помощью машинного обучения позволяет ускорить получение результатов и минимизировать потери на неэффективных вариантах.

Один из ключевых методов – адаптивное тестирование, при котором распределение трафика меняется в режиме реального времени. Например, алгоритмы многоруких бандитов (Multi-Armed Bandits) направляют больше пользователей на более успешные версии, снижая влияние менее эффективных вариантов. Такой подход позволяет быстрее находить оптимальные решения и сокращает время тестирования.

```
class EpsilonGreedyBandit:  
    def __init__(self, epsilon=0.1):  
        self.epsilon = epsilon # Вероятность случайного  
выбора
```



```
self.rewards = {'A': 0, 'B': 0} # Накопленные
награды (конверсии)
self.counts = {'A': 0, 'B': 0} # Количество
показов каждой версии
def select_version(self):
    """Выбирает версию A или B: случайно (с
    вероятностью epsilon) или лучшую (1 - epsilon)."""
    if np.random.rand() < self.epsilon:
        return np.random.choice(['A', 'B']) #
Исследование (exploration)
    else:
        # Выбор лучшей версии (exploitation)
        return 'A' if self.rewards['A'] / max(1,
self.counts['A']) > self.rewards['B'] / max(1,
self.counts['B']) else 'B'
def update(self, version, reward):
    """Обновляет статистику на основе полученной
конверсии."""
    self.counts[version] += 1
    self.rewards[version] += reward
```

Данный код представляет собой реализацию метода многорукого бандита для адаптивного управления трафиком. Вместо фиксированного разделения пользователей между версиями алгоритм динамически корректирует распределение, увеличивая долю трафика для варианта с лучшей конверсией.

В заключение, хотелось бы отметить, что автоматизированное А/В-тестирование с помощью машинного обучения делает процесс экспериментов не только быстрее, но и умнее. Благодаря алгоритмам адаптивного тестирования мы можем динамически перераспределять трафик, минимизируя потери и быстрее находя эффективные решения.

Использование машинного обучения в тестировании позволяет не только сравнивать версии, но и анализировать поведение пользователей, выявлять скрытые закономерности и принимать более обоснованные решения.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Кохави, Р. Доверительное А/В-тестирование. Практическое руководство по контролируемым экспериментам: монография / Р. Кохави, Д. Тан, Я Сюй. – пер. с англ. В. С. Яценков. – Москва: ДМК-Пресс, 2021. – 298 с.

2. Рашка, С. Python и машинное обучение. Машинное и глубокое обучение с использованием Python, scikit-learn: монография / С. Рашка, В. Мирджалили. – пер. с англ. Ю. Н. Артеменко. – Москва: Диалектика, 2020. – 848 с.

3. Уатт, Д. Машинное обучение: основы, алгоритмы и практика применения: монография / Д. Уатт, Р. Борхани, А. Катсагтелос. – Санкт-Петербург: BHV, 2022. – 640 с.

УДК 004.422.81

КОРОТКИХ И.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ ПРИ СОЗДАНИИ ИНФОРМАЦИОННЫХ СИСТЕМ ПРИЕМА И ОБРАБОТКИ ПЛАТЕЖЕЙ ЗА УСЛУГИ ЖКХ

Рассматриваются основные вопросы, связанные с использованием различных архитектур при создании информационных систем по приему и обработке платежей за услуги ЖКХ, их преимущества и недостатки в этом контексте.

Стабильная работа систем приема и обработки платежей за услуги ЖКХ имеет ряд важных причин:

1. Удобство для пользователей: Жилищно-коммунальные услуги является жизненно и социально значимым. Их оплачивают каждый день десятки и сотни тысяч людей по всей стране. Они это делают регулярно. Повышение удобства оплаты, в связи с этим имеет большое значение, так как это экономит людям время.

2. Финансовая дисциплина: Регулярная и своевременная оплата услуг ЖКХ важна для поддержания финансовой устойчивости и обеспечения надлежащего уровня обслуживания жилых помещений и коммунальной инфраструктуры.

3. Предотвращение проблем с долгами: Задолженность по коммунальным услугам является серьезной проблемой для многих ресурсоснабжающих и управляющих компаний. Она приводит к сложностям в своевременном обслуживании инфраструктуры и выплате заработной платы людям, которые работают в этой сфере, снижают уровень качества предоставляемых услуг.

Системы приема и обработки платежей за услуги ЖКХ являются весьма комплексными. Они выполняют большую работу по начислению платежей, составлению единого платежного документа, взаимодействию с государственной информационной системой ЖКХ и так далее. Исходя из этого попытаемся понять, какая архитектура лучше подходит подобным системам.

Как уже было сказано, информационные системы по приему и обработке платежей за услуги ЖКХ весьма сложны. Вот несколько причин, почему микросервисы подходят для сложных систем [1]:

1. Масштабируемость: при использовании микросервисов есть возможность масштабировать сервисы как горизонтально, добавляя новые экземпляры сервиса, так и вертикально, увеличивая вычислительную мощность оборудования, на котором развернуто приложение.

2. Гибкость и независимость: Обновление приложение заключается в обновлении какого-то одного сервиса из множества, что позволяет обновлять приложение, при этом работоспособность не нарушается полностью.

3. Устойчивость к отказам: при использовании микросервисов отказ одного сервиса не приводит к полной остановке всей системы. Благодаря изоляции компонентов другие сервисы могут продолжать работу, минимизируя влияние отказа.

4. Распределенная разработка: Разработку приложения, которое пишется в микросервисном стиле, легко распараллелить, назначив на каждый сервис наибольшую команду. Однако надо четко определить, каким образом микросервисы будут общаться между собой – «контракт» взаимодействия.

5. Технологическое разнообразие: Каждый микросервис может быть написан на разных технологиях, что позволяет выбирать наиболее подходящий инструмент для каждой конкретной задачи.

Приведем пример использования микросервисов в такой системе и попытаемся понять, почему использование монолитной архитектуры затруднительно в данной ситуации.

Существуют разные виды платежных систем. По способу выставления счета выделяют платежные системы с прямым выставлением счета и консолидирующие платежные системы.

Платежные системы с прямым выставлением счета (Direct Billing System или DBS) в контексте системы жилищно-коммунального хозяйства непосредственно рассчитывают сумму, которую надо выставить абоненту. Такие системы есть у каждой организации, которые выставляют счет. Они могут представлять собой систему на базе 1С либо иные, более специализированные системы, например, расчетный комплекс «Абонент+» [2].

С целью создания единого платежного документа необходимо консолидировать информацию о платежах из различных источников. Этим занимаются консолидирующие платежные системы (Consolidating Billing Systems или CBS). Обычно для этого

используются специфические решения под конкретную предметную область. В предметной области, связанной с обработкой платежей за услуги жилищно-коммунального хозяйства, можно использовать, например, платежную систему «Абонент+» [2].

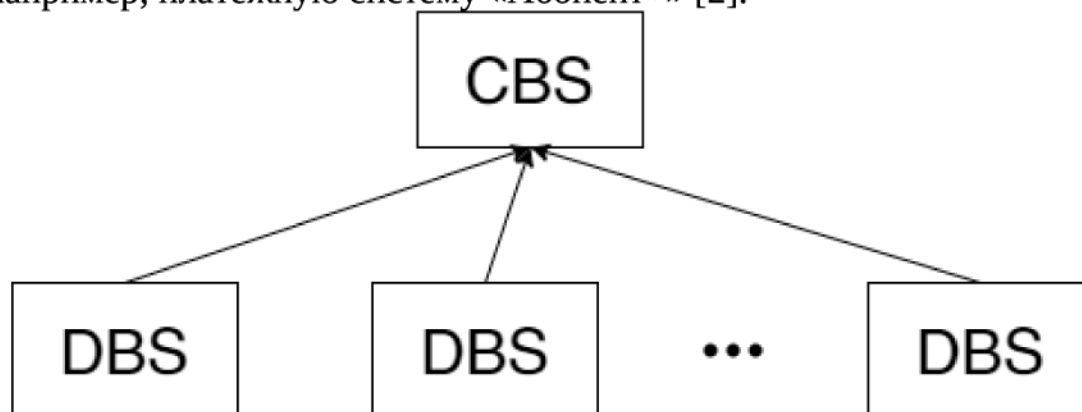


Рисунок 1 – Схема взаимодействия платежных систем с прямым выставлением счета и консолидирующих платежных систем

Расчетно-кассовый центр (РКЦ) — это специализированная организация, которое осуществляет обработку транзакций и проведение расчетов между клиентами РКЦ, банков и ресурсоснабжающими организациями. РКЦ выполняет ряд функций, связанных с обработкой платежей, переводов, инкассации и других операций. РКЦ принимают платежи у населения с помощью клиента платежной системы. Поскольку в РКЦ часто формируются очереди из людей, необходимо быстро загружать информацию об абонентах, чтобы быстрее их обслуживать. Достигается это использованием буферной БД, в которой данные хранятся в менее нормализованном виде. Рассмотрим взаимодействие систем на рисунке:

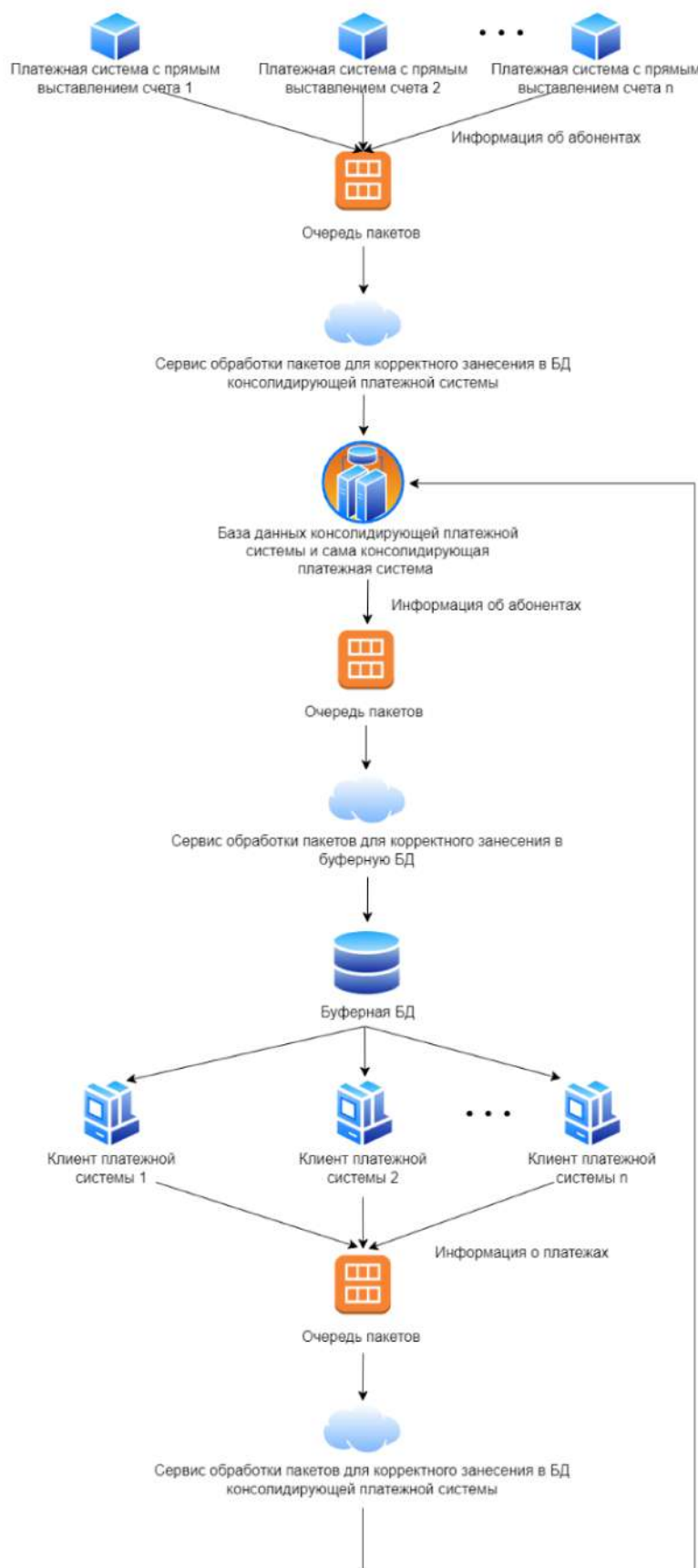


Рисунок 2 – Взаимодействие систем

Как мы видим, наличие большого количества источников данных, необходимость в нескольких базах данных, необходимость в очередях приводит к тому, что необходимо разделить систему на несколько сервисов. Кроме того, отказ в одном из сервисов не приводит к отказу всей системы. Например, если происходит ошибка в сервисе, который обрабатывает пакеты приходящие из расчетных системы в платежную, не происходит к прекращению приема платежей у населения. Важность этого была подчеркнута в начале этой статьи. С помощью монолитной архитектуры добиться такого эффекта весьма затруднительно. Эффект от использования микросервисов в подобных системах оправдывает издержки на более сложный процесс разработки, поддержки и развертывания. Кроме того, системы такого масштаба разрабатывает большая команда разработчиков. Новых разработчиков проще интегрировать в рабочий процесс в случае, если используется микросервисная архитектура, так как работнику не нужно хорошо разбираться во всей системе, необходимо знать детали работы только сервиса, за который разработчик ответственен.

Таким образом, использование микросервисной архитектуры при создании информационных систем по приему и обработке платежей за услуги ЖКХ является весьма оправданным.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Монолитная и микросервисная архитектура. Сравнение. [Электронный ресурс]: – URL: <https://habr.com/ru/companies/haulmont/articles/758780/> (дата обращения 31.03.2024)

2. Абонент+ Расчетноплатежный комплекс. [Электронный ресурс]. – URL: <https://abonent.plus/> (дата обращения 31.03.2024)

УДК 004.422.81

КОРОТКИХ И.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МОНОЛИТНОЙ И МИКРОСЕРВИСНОЙ АРХИТЕКТУР

Рассматриваются вопросы, связанные с особенностями одних из основных современных архитектур построения информационных систем. Приводится их сравнительный анализ.

Микросервисная архитектура – одна из ключевых архитектур при создании современных информационных систем. Она подразумевает разделение системы на ряд сервисов, обладающих высокой степенью независимости. Такая система является децентрализованной и отказ в одной ее части не приводит к полному прекращению работы всей информационной системы [1].

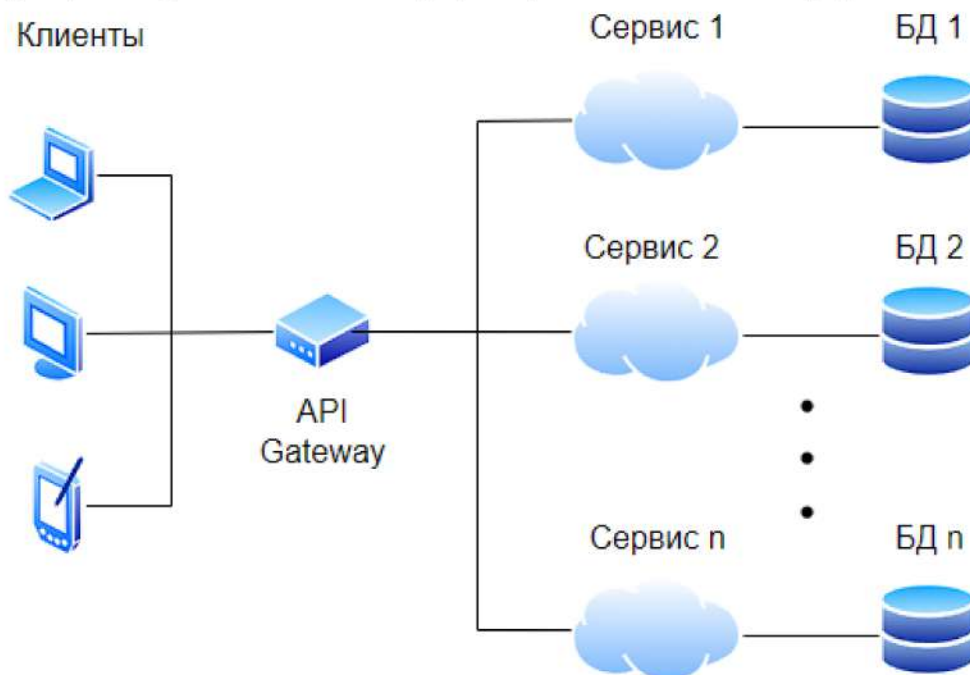


Рисунок 1 – Схема микросервисной архитектуры

Использование микросервисной архитектуры может принести следующие преимущества [1]:

1. Гибкость и масштабируемость: так как сервисы независимы друг от друга, их можно независимо тестировать и разворачивать, таким образом быстрее реагируя на изменения.

2. Отказоустойчивость: Сбой в одном сервисе не приводит к сбою всей системы; частично сохраняется работоспособность.

3. Легкость внедрения новых технологий: Каждый микросервис может быть реализован на своей собственной технологической базе.

4. Масштабируемость разработки: Разработку можно организовать параллельно, привлекая разных разработчиков для создания разных частей системы.

5. Поддерживаемость: Изменения в одном микросервисе не влияют на другие сервисы, что упрощает поддержку и обновление системы.

6. Безопасность: на каждом микросервисе можно организовать свою систему безопасности, делая вероятность взлома всей системы в целом более низкой.

При использовании микросервисной архитектуры возникают следующие проблемы [1]:

1. Сложность управления: Управление большим количеством сервисов может быть весьма затруднительно. Большое количество узлов порождает большое количество связей между ними, усложняет взаимодействие, делает систему в целом менее предсказуемой.

2. Сложность отладки: из-за высокой степени децентрализации локализовать ошибку может быть трудоемко.

3. Целостность данных: Децентрализация порождает проблему сложности поддержания целостности данных во всех микросервисах.

4. Тестирование: усложняется процесс тестирования и повышается ценность интеграционного тестирования, так как важно протестировать именно взаимодействие сервисов и работу всей системы в совокупности.

Монолитная архитектура – иной подход в разработки информационных систем. При таком подходе к разработке вся логика обработки данных инкапсулируется в едином приложении, что порождает как ряд преимуществ, так и ряд недостатков [2].

Клиенты

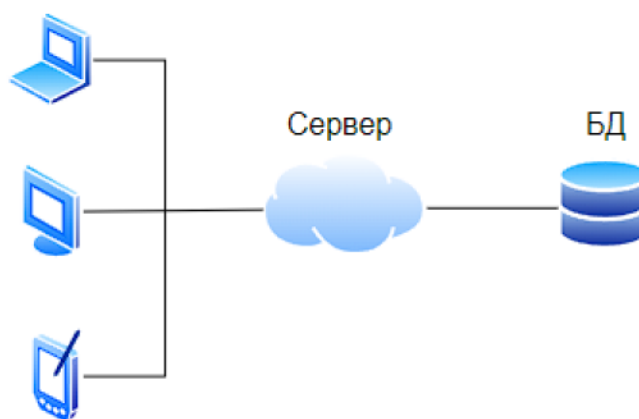


Рисунок 2 – Схема монолитной архитектуры

В использовании монолитной архитектуры есть несколько преимуществ:

1. Развертывание и обслуживание: так как приложение едино, не нужно тратить много усилий на развертывание проекта. Кроме того, инфраструктура не такая сложная и на ее поддержку так же не тратится много усилий.

2. Простота отладки и тестирования: из-за того, что весь код находится в одном месте, отладка и тестирование приложения проще.

3. Большая скорость разработки: построение приложения может быть быстрее за счет единой кодовой базы, однако разработку сложно масштабировать.

4. Меньше сложностей с сетевым взаимодействием: поскольку все компоненты приложения работают в пределах одного процесса, нет необходимости в сетевом взаимодействии между сервисами.

5. Единый стек технологий: разработка на базе единой технологической базы может упростить написание приложения, однако уменьшает гибкость разработки, так как разные технологии заточены под разные задачи, и реализация всех аспектов предметной области в рамках одной технологии может быть неэффективно.

При использовании монолитной архитектуры могут возникнуть следующие проблемы:

1. Масштабирование: в случае увеличения нагрузки на приложение, основной способ справиться с ней, увеличивать производительность электронно-вычислительных машин, на которых работает это приложение, что далеко не всегда удобно и реализуемо, а кроме того, недешево.

2. Снижение гибкости в использовании технологий: в монолитном приложении все компоненты связаны друг с другом, поэтому замена или обновление одного компонента может потребовать изменений во всей системе.

3. Сложность поддержки и разработки: так как приложение, написанное в монолитном стиле, сильно централизовано, добавление функционала со временем будет становиться все сложнее. Рефакторинг приложения так же будет непрост из-за сильной связанности приложения, наличия большого количества глобальных состояний в приложении. Решающим фактором становится квалификация разработчиков, их способность поддерживать чистоту кодовой базы, использование принципов SOLID, DRY, KISS, YAGNI и других.

4. Медленная скорость развертывания: при обновлении монолитного приложения может потребоваться перезапуск всего приложения, что может привести к временным простоям.

5. Сильная централизация: в результате возникновения ошибки в одном компоненте, приложение может прекратить работу полностью, что критично в случае, если приложение затрагивает большое количество клиентов или является социально значимым.

Таблица 1 – Сравнительный анализ

	Монолитная архитектура	Микросервисы
Простота разработки	Проще	Сложнее
Масштабируемость приложения	Небольшие возможности для масштабирования	Большие возможности для масштабирования
Отказоустойчивость	В случае возникновения проблемы перестает работать все приложение. Невысокая отказоустойчивость.	В случае возникновения проблемы перестает работать только затронутый сервис. Более высокая отказоустойчивость.
Скорость разработки	Более высокая	Более низкая
Сложность	Низкая	Высокая
Развертывание и обслуживание	Проще развертывать и поддерживать	Сложнее развертывать и поддерживать
Масштабируемость команды	Команду сложнее масштабировать	Команду сложнее масштабировать

Таким образом все вышеперечисленные факторы делают монолитную архитектуру хорошим выбором если проект небольшой или средний, где скорость разработки и обслуживания имеют большое значение. Однако, это не самый удачный выбор при работе над крупными и сложными проектами. Для них лучше применять микросервисную архитектуру.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Монолитная и микросервисная архитектура. Сравнение. [Электронный ресурс]. – URL: <https://habr.com/ru/companies/haulmont/articles/758780/> (дата обращения 31.03.2024)
2. «Microservices vs monolithic architecture» [Электронный ресурс]. – URL: <https://www.atlassian.com/ru/> (дата обращения 31.03.2024).

УДК 004.021

КОСТИН В.Ю., СКВОРЦОВ С.В.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ
РЕШЕНИЯ NP-ПОЛНЫХ ЗАДАЧ В САПР**

Целью данной работы является исследование эффективности алгоритмов на основе параллельных вычислений, таких как алгоритм на основе метода полного перебора, алгоритм на основе метода Монте-Карло на примере решения задачи коммивояжёра.

Задача коммивояжёра (TSP, Traveling Salesman Problem) является классической задачей комбинаторной оптимизации, заключающейся в нахождении кратчайшего маршрута, который проходит через заданный набор городов и возвращается в исходный город [1]. Эта задача относится к NP-трудным [2], что делает её сложной для решения при увеличении числа городов. В последние годы параллельные вычисления стали важным инструментом для повышения эффективности алгоритмов, применяемых к TSP. В данной статье рассматриваются два подхода: метод полного перебора и метод Монте-Карло, а также их реализация с использованием параллельных вычислений.

Необходимо разработать соответствующие алгоритмы с различной реализацией: последовательной, параллельной с использованием технологии OpenMP, параллельной с использованием технологии CUDA (с использованием мощностей графического ядра).

Метод полного перебора. Метод полного перебора заключается в генерации всех возможных маршрутов и выборе наикратчайшего. Этот метод гарантирует нахождение оптимального решения, но его вычислительная сложность составляет $O(n!)$, что делает его неэффективным для больших наборов данных.

Метод Монте-Карло. Метод Монте-Карло основан на случайной генерации маршрутов и оценке их длины. Этот подход не гарантирует нахождение оптимального решения, но позволяет быстро находить приближённые решения, что может быть полезно в ситуациях, когда время критично.

Параллельная реализация. Параллелизация метода полного перебора может быть достигнута за счёт распределения задач по вычислительным узлам. Каждый узел может обрабатывать подмножество возможных маршрутов, что значительно сокращает

время вычислений. Например, если у нас есть 4 узла, каждый из которых обрабатывает 1/4 всех возможных маршрутов, общее время выполнения может быть уменьшено в 4 раза. Однако, стоит отметить, что накладные расходы на распределение задач и сбор результатов могут снизить эффективность параллелизации.

Параллелизация метода Монте-Карло осуществляется путём генерации случайных маршрутов на нескольких вычислительных узлах одновременно. Каждый узел генерирует свои маршруты и вычисляет их длину, после чего результаты агрегируются для нахождения наилучшего найденного маршрута. Параллельный подход позволяет значительно увеличить количество генерируемых маршрутов за фиксированное время, что повышает вероятность нахождения более короткого пути.

Сравнительный анализ. Сравним быстродействие алгоритмов в зависимости от размерности задачи. Сравнительный анализ проводился на вычислительной машине со следующими характеристиками:

Процессор IntelCorei5-3470, частота 3.20 ГГц, 4 ядра;
Видеокарта GeForceGT 640 DDR3, частота 902 МГц, 384 ядер;
Оперативная память, 16 Гб DDR3;
HDD,6 Гбит/сек.

Алгоритм на основе метода полного перебора

Таблица 1 – Сравнительный анализ для задачи коммивояжера методом полного перебора (секунд)

Метод/Размер	10	20	30	40
Последовательно	0,00 7499	0,00 4103	0,09 865	1,97 3268
OpenMP	0,00 3312	0,00 1652	0,03 2548	0,55 7274
CUDA	0,00 2634	0,00 1054	0,02 1391	0,24 6228

Как видно из результатов, приведенных в таблице – последовательные вычисления сильно медленнее параллельных. Рассчитаем ускорение (таблица 2), которое определяется как отношение времени, затрачиваемого на проведение вычислений на однопроцессорной ВС, ко времени решения той же задачи на параллельной n-процессорной системе:

$$S(n) = \frac{T(1)}{T(n)}$$

Таблица 2 – Ускорение параллельных методов

Метод/Размер	10	20	30	40
OpenMP	2,26 4191	2,48 3656	3,03 0908	3,54 093
CUDA	2,84 6922	3,89 3978	4,61 1735	6,38 9483

Рассчитаем индекс параллелизма, который определяется по формуле (таблица 3):

$$P(n) = \frac{O(n)}{T(n)}$$

где $O(n)$ – объём вычислений, а $T(n)$ – время вычислений.

Таблица 3 – Индекс параллелизма

Метод/Размер	10	20	30	40
OpenMP	10956 52174	1,47 E+21	8,15 E+33	1,4641 2E+48
CUDA	13776 38361	2,31 E+21	1,24 E+34	3,3136 6E+48

Рассчитаем утилизацию, учитывающую вклад каждого процессора при параллельном вычислении в виде количества операций в единицу времени (таблица 4):

$$U(n) = \frac{P(n)}{n}$$

Таблица 4 – Утилизация

Метод/Размер	10	20	30	40
OpenMP	2739 13043	3,68 E+20	2,04 E+33	3,6603 E+47
CUDA	3587 599,9	6,01 E+18	3,23 E+31	8,6293 3E+45

Рассчитаем избыточность как отношение объема параллельных вычислений к объему эквивалентных последовательных вычислений:

$$R(n) = \frac{O(n)}{O(1)} = 1$$

Рассчитаем сжатие как величину, обратную избыточности:

$$C(n) = \frac{O(1)}{O(n)} = 1$$

Рассчитаем метрику качества (таблица 5), которая определяется по формуле:

$$Q(n) = S(n)E(n)C(n)$$

Таблица 5 – Качество

Метод/Размер	10	20	30	40
OpenMP	1,28 164	1,54 2137	2,29 6601	3,13 4546
CUDA	2,02 6241	3,79 0766	5,31 7025	10,2 0637

Таким образом, исходя из полученных метрик, удалось получить значительный выигрыш при использовании архитектуры CUDA по сравнению с OpenMP и, в значительной степени, по сравнению с последовательным подходом.

Алгоритм на основе метода Монте-Карло

Таблица 6 – Сравнительный анализ для задачи коммивояжера методом Монте-Карло (секунд)

Метод/Размер	10	20	30	40
Последовательно	9,19 1399	16,7 19904	24,4 2285	31,9 9596
OpenMP	3,42 8906	6,10 5366	8,74 2272	11,3 7045
CUDA	2,82 05844	3,82 0112	4,62 7462	5,09 9969

Как видно из результатов, приведенных в таблице – последовательные вычисления сильно медленнее параллельных. Рассчитаем ускорение (таблица 7), которое определяется как отношение времени, затрачиваемого на проведение вычислений на однопроцессорной ВС, ко времени решения той же задачи на параллельной n-процессорной системе:

$$S(n) = \frac{T(1)}{T(n)}$$

Таблица 7 – Ускорение параллельных методов

Метод/Размер	10	20	30	40
OpenMP	2,68 0563	2,73 8559	2,79 365	2,81 3957
CUDA	3,25 8686	4,37 681	5,27 7806	6,27 3756

Рассчитаем индекс параллелизма, который определяется по формуле:

$$P(n) = \frac{O(n)}{T(n)}$$

где $O(n)$ – объём вычислений, а $T(n)$ – время вычислений.

Таблица 8 – Индекс параллелизма

Метод/Размер	10	20	30	40
OpenMP	39371 158	442 23393	463 26630	474915 24,08
CUDA	47862 421,9	706 78556	875 20972	105882 999,9

Рассчитаем утилизацию, учитывающую вклад каждого процессора при параллельном вычислении в виде количества операций в единицу времени:

$$U(n) = \frac{P(n)}{n}$$

Таблица 9 – Утилизация

Метод/Размер	10	20	30	40
OpenMP	98427 89,51	110 55848	115 81657	118728 81,02
CUDA	12464 1,724	184 058,7	227 919,2	275736 ,979

Рассчитаем избыточность как отношение объема параллельных вычислений к объему эквивалентных последовательных вычислений:

$$R(n) = \frac{O(n)}{O(1)} = 1$$

Рассчитаем сжатие как величину, обратную избыточности:

$$C(n) = \frac{O(1)}{O(n)} = 1$$

Рассчитаем метрику качества, которая определяется по формуле:

$$Q(n) = S(n)E(n)C(n)$$

Таблица 10 – Качество

Метод/Размер	10	20	30	40
OpenMP	1,796 3547	1,87 4926	1,95 112	1,97 9589
CUDA	2,654 7587	4,78 9116	6,96 381	9,84 0002

Таким образом, исходя из полученных метрик, удалось получить значительный выигрыш при использовании параллельных вычислений.

Метод полного перебора обеспечивает точное решение, но его эффективность сильно зависит от числа городов. При увеличении числа городов время выполнения растёт экспоненциально, даже с параллелизацией.

Метод Монте-Карло предлагает более гибкий подход, позволяя находить приближённые решения за значительно меньшее время. Параллелизация этого метода позволяет обрабатывать большое количество маршрутов, что делает его более подходящим для задач с большим числом городов.

Метод полного перебора остаётся актуальным для небольших наборов данных, тогда как метод Монте-Карло демонстрирует свою эффективность при работе с большими объёмами информации. В зависимости от требований к точности и времени выполнения, выбор подходящего алгоритма и его параллельной реализации может варьироваться, что делает исследование в этой области актуальным и перспективным.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Algorithm Design. / Kleinberg, Jon; Tardos, Éva – Addison-Wesley. – 2006. – С.547.
2. Johnson, David S. A Catalog of Complexity Classes. / David S, Johnson // Handbook of Theoretical Computer Science. – 1990. – С. 67-161.
3. Cook, Stephen (1971). "The complexity of theorem proving procedures". Proceedings of the Third Annual ACM Symposium on Theory of Computing. pp. 151–158.

УДК 004.65

КОСТЫЛЕВ И.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПСИХОАКУСТИЧЕСКИЕ МЕТОДЫ ОЦЕНКИ КАЧЕСТВА СЖАТОЙ РЕЧИ. АЛГОРИТМ PESQ

Рассматривается один из методов оценки качества сжатой речи.

Психоакустические методы

Психоакустические методы оценки качества звучания основаны на понимании того, как человеческое ухо воспринимает и анализирует звуковые сигналы. Эти методы учитывают психологические и физиологические аспекты слухового восприятия и позволяют более точно оценить качество звука субъективной точки зрения. Вот несколько подробных психоакустических методов оценки качества звучания [1]:

- *Эксперименты по заметности искажений:* в таких экспериментах используется множество испытуемых, которым предлагается прослушать звуковые сигналы с различными видами искажений или артефактами. Испытуемые должны оценить, насколько заметны эти искажения и как они влияют на качество звучания. Оценка может быть качественной (например, незаметно, заметно, очень заметно) или количественной, когда испытуемые используют шкалы оценок. Результаты эксперимента анализируются статистически, чтобы получить количественные данные о восприятии и важности различных видов искажений.

- *Методы оценки порогов слышимости:* целью этих методов является определение минимальных уровней искажений или изменений звука, которые человек может услышать. Для этого испытуемым предлагается прослушать звуковые сигналы с постепенно увеличивающимися искажениями или изменениями, и они должны указать, когда они становятся слышимыми. Это позволяет определить порог слышимости для различных параметров звука, таких как уровень искажений, изменение частоты и другие факторы.

- *Эксперименты по предпочтению и предпочтительности:* в таких экспериментах испытуемым предлагается прослушать два или более звуковых сигнала или системы и выбрать тот, который им больше нравится или который они считают качественнее. Оценка может быть основана на шкале предпочтений, ранжировании или других методах. Это позволяет выявить предпочтения слушателей и определить, какие параметры звука влияют на их восприятие и оценку качества.

- *Моделирование восприятия звука:* в этом подходе используются математические модели и алгоритмы, которые пытаются имитировать слуховое восприятие человека. Они учитывают различные факторы, такие как маскировка звуков, восприимчивость к искажениям, адаптация слуха и другие психоакустические закономерности. Модели позволяют предсказывать, как звук будет восприниматься людьми с учетом этих факторов и использовать эти предсказания для оценки качества звучания.

Психоакустические методы оценки качества звучания представляют собой важный инструмент для понимания и улучшения восприятия звука. Они позволяют учесть субъективные аспекты слухового восприятия и предоставить более полную и точную оценку качества звучания. Эти методы являются важной частью исследований в области аудиоинженерии, психоакустики и разработки звуковых систем [2].

Оценка PESQ

Методология оценки качества передачи речи, основанная на алгоритме PESQ (Perceptual Evaluation of Speech Quality), представляет собой комплексный процесс, включающий субъективные и объективные измерения. Субъективные методы оценки, основанные на восприятии человека, служат важным эталоном для калибровки и валидации объективных метрических алгоритмов. Объективные методы, такие как PESQ, позволяют автоматизировать процесс оценки и получить воспроизводимые результаты, что особенно важно при анализе большого количества данных.

Процесс оценки качества передачи речи с использованием PESQ включает следующие этапы:

Формирование тестовых наборов: Подготовка репрезентативного набора аудиозаписей, содержащих разнообразный речевой контент и различные акустические условия. Тестовые наборы должны учитывать специфику анализируемых систем передачи речи и отражать реальные сценарии их использования.

Субъективное тестирование (опционально): в рамках субъективного тестирования группа экспертов-слушателей прослушивает оригинальные и обработанные (искаженные) аудиозаписи. Оценка качества производится по предопределенной шкале, например, шкале MOS (Mean Opinion Score). Данные субъективных тестов могут быть использованы для сравнения с результатами объективных измерений и анализа корреляции между ними.

Объективная оценка с использованием алгоритма PESQ: Алгоритм PESQ анализирует оригинальный и искаженный речевые сигналы, вычисляя объективную оценку качества на основе сравнения их психоакустических характеристик. Алгоритм моделирует процесс восприятия речи человеком, учитывая такие факторы, как задержка, шум, искажения и потери пакетов данных.

Статистическая обработка и интерпретация результатов: Полученные значения PESQ подвергаются статистической обработке для определения среднего значения, стандартного отклонения и доверительных интервалов. Результаты представляются в виде числовых значений и могут быть визуализированы с помощью графиков. Статистический анализ позволяет выявить статистически значимые различия в качестве передачи речи между различными системами или условиями передачи.

Для достижения статистической значимости и обоснованности выводов рекомендуется проводить многократные измерения с

использованием различных тестовых наборов и, в случае субъективных тестов, с участием достаточно большой группы слушателей. Такой подход позволяет минимизировать влияние случайных факторов и получить надежные результаты, адекватно отражающие качество анализируемых систем передачи речи.

Области применения алгоритма PESQ

Алгоритм Perceptual Evaluation of Speech Quality (PESQ) предназначен для объективной оценки качества речи, имитируя субъективное человеческое восприятие. Его применение широко распространено в областях, где критична разборчивость и естественность звучания речи.

Технология VoIP также использует PESQ для оптимизации параметров передачи речи через интернет. Алгоритм способствует достижению наилучшего восприятия речи в VoIP-системах.

Принцип работы PESQ основан на сравнении исходного речевого сигнала с его обработанной или переданной версией. Анализируя частотные характеристики, длительность искажений и потерь, а также уровень шума, алгоритм вычисляет числовую оценку, коррелирующую с субъективным качеством речи. Эта оценка позволяет проводить сравнительный анализ различных систем связи и оптимизировать параметры передачи для достижения максимальной разборчивости [3].

Таким образом, PESQ представляет собой важный инструмент для объективной оценки качества речи, находящий применение в различных сферах коммуникаций. Алгоритм способствует улучшению разборчивости речи, анализу и сравнению систем связи, а также оптимизации параметров передачи для достижения наилучшего качества звучания.

Ограничения PESQ

Несмотря на широкое применение, PESQ имеет ряд ограничений. Он не всегда точно соответствует субъективным оценкам качества речи, особенно для сигналов с высокой степенью деградации. Кроме того, PESQ не учитывает некоторые важные аспекты восприятия речи, такие как интонация и эмоциональная окраска. В связи с этим, для более полной оценки качества речи рекомендуется использовать PESQ в сочетании с другими методами оценки, включая субъективные тесты и более современные объективные метрики, такие как POLQA.[4] Тем не менее, PESQ остается важным инструментом для оценки качества речи благодаря своей простоте, скорости работы и широкой доступности. В будущем, развитие более точных и всесторонних методов оценки качества речи,

вероятно, приведет к снижению популярности PESQ, но на данный момент он продолжает играть значительную роль в области телекоммуникаций и обработки речи.

Заключение

PESQ остается "золотым стандартом" для объективной оценки качества сжатой речи, несмотря на появление новых алгоритмов. Его ключевые преимущества – воспроизводимость результатов и высокая корреляция с субъективным восприятием.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Сергеев В.С., Барин В.В. Сжатие данных, речи, звука и изображений в телекоммуникационных системах. – М.: ИП «РадиоСофт», 2009. – 189 с.

2. Рихтер С.Г., Кодирование и передача речи в цифровых системах подвижной связи. – М.: Горячая линия – Телеком, 2009. – 304 с.

3. Яновский Г.Г. Оценка качества передачи речи в сетях IP // Вестник связи. – 2008. – № 2. – С. 91–94.

4. Дэвидсон Д., Питерс Д., Бхатия М. и др. Основы передачи голосовых данных по сетям IP / пер. с англ. – 2-е изд. – М.: ООО «И. Д. Вильямс», 2007. – 400 с.

УДК 004.942

КОШЕЛЕВ А.Д., КОШЕЛЕВА М.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ВЗАИМОСВЯЗЬ ЦИФРОВЫХ ДВОЙНИКОВ И КИБЕРФИЗИЧЕСКИХ СИСТЕМ В КОНТЕКСТЕ УМНОГО ПРОИЗВОДСТВА

Рассмотрена взаимосвязь цифровых двойников и киберфизических систем в рамках умного производства. Произведен сравнительный анализ указанных концепций, предложена функциональная схема киберфизической системы.

Введение

Современные технологии, такие как Интернет вещей, облачные вычисления, аналитика больших данных и искусственный интеллект значительно ускорили развитие умного производства, характеризующегося киберфизической интеграцией.

В этой связи, киберфизические системы и цифровые двойники, как инструменты такой интеграции, обретают всё большую популярность, поскольку обеспечивают наличие контуров обратной связи, в которых физические процессы влияют на кибернетические элементы и наоборот. Следовательно, повышается эффективность, устойчивость и интеллектуальность производственных систем.

Основные понятия

С развитием систем сбора данных, информационных и сетевых технологий производство вступило в цифровую эпоху, то есть совершило переход от интеллектуального этапа, базирующегося на знаниях, к умному, основанному на данных и знаниях, в котором понятие «умный» относится к созданию и использованию данных [4]. Термин «умное производство» привязан к будущему состоянию производства, в котором передача и анализ данных в режиме реального времени на протяжении всего жизненного цикла продукта, а также моделирование и оптимизация на основе моделей, создают интеллектуальные системы, оказывающие положительное влияние на все аспекты производства. Киберфизическая интеграция является ядром умного производства, а киберфизические системы и цифровые двойники инструментами её достижения.

Киберфизические системы (КС) – это многомерные и сложные системы, которые объединяют кибернетический мир и динамичный физический мир. Благодаря взаимодействию вычислительных, коммуникационных и управляющих систем, КС обеспечивают сбор данных в реальном времени, поддержание обратной связи, динамическое управление для надёжного, безопасного и эффективного мониторинга физических объектов.

Цифровой двойник (ЦД) – это высокоточные виртуальные модели физических объектов, предназначенные для имитации их поведения в реальном мире и обеспечения обратной связи. В результате ЦД позволяют компаниям быстрее и точнее прогнозировать и выявлять физические проблемы, оптимизировать производственные процессы, выпускать более качественную продукцию [1].

По определению, КС и ЦД используются для описания киберфизической взаимосвязи. Проведем сравнительный анализ данных концепций.

Происхождение и развитие концепций

КС были получены в результате широкого применения встроенных систем. Термин «киберфизическая система» был введен Хелен Гилл в 2006 г. для описания все более сложных систем, которые невозможно эффективно проиллюстрировать с использованием

традиционной ИТ-терминологии. Текущие исследования КС сосредоточены на обсуждении архитектур, технологий и проблем, в то время как применение КС в инженерной практике находится на ранней стадии. По сравнению с другими технологиями, КС являются более фундаментальными, поскольку они напрямую не привязаны к способу реализации. Таким образом, КС больше относятся к научным задачам, чем инженерным.

Термин «Цифровой двойник» впервые был представлен Майклом Гривзом в 2003 году в презентации по управлению жизненным циклом продукта. В то время из-за ограничений и незрелости концепции почти не было соответствующих исследований. В настоящее время развитие новых информационных технологий способствует росту популярности ЦД, так как они открывают новый способ синхронизации физических действий с виртуальным миром, в следствие чего, всё чаще используются в различных отраслях промышленности, в том числе для проектирования производственных линий, цехов, оптимизации бизнес-процессов. Поэтому по сравнению с КС ЦД чаще рассматриваются с точки зрения инженерных задач.

Сопоставление физического и кибер/цифрового миров

КС и ЦД включают две основные составляющие: физическую и кибернетическую/цифровую. Физическая часть состоит из различных производственных ресурсов, которые можно обобщить как «человек/машина/материал/окружающая среда» [4], а кибернетическая/цифровая основывается на различных сервисах, интеллектуальном управлении данными, аналитике и вычислительных мощностях [1]. То есть физическая часть собирает данные, а также реализует решения, принятые кибернетической/цифровой частью, в то время как кибернетическая/цифровая часть анализирует и обрабатывает данные, а затем принимает решения [3].

Суть КС заключается в добавлении новых возможностей в физические системы с помощью вычислений и связи, которые активно взаимодействуют с физическими процессами [3]. По сравнению с ЦД, КС в большей степени используют мощные вычислительные и коммуникационные возможности кибермира [3], которые могут повысить точность и эффективность физического мира. Более того, все архитектуры КС сосредоточены на управлении, а не на зеркальных моделях, будь то трехуровневые, пятиуровневые или сервис-ориентированные архитектуры. Как и в ЦД, в КС важны контуры обратной связи. Взаимное отображение, взаимодействие в реальном времени и эффективное сотрудничество между кибернетическим и физическим мирами обеспечивают функциональность КС. Однако

вычислительная система может воздействовать более чем на один физический объект. Например, система может включать в себя несколько физических компонентов. Таким образом, взаимосвязь между кибернетическим и физическим мирами КС не является однозначной, а скорее соответствует соотношению «один ко многим».

Цель ЦД – это предоставить полное физическое и функциональное описание компонента, продукта или системы [2]. Первый и самый важный шаг – создание высокоточных виртуальных моделей (максимально приближенных к аналогам, с точки зрения геометрии, физических свойств) [3], способных имитировать пространственно-временное состояние своих прототипов, их поведение, функции и многое другое. Другими словами, виртуальные модели и физические объекты имеют схожий внешний вид, как близнецы, и одинаковое поведение, как зеркальное отражение. Кроме того, модели в цифровой среде могут напрямую оптимизировать операции и корректировать физические процессы с помощью обратной связи [3]. Благодаря двунаправленному динамическому сопоставлению физические объекты и виртуальные модели развиваются совместно. Таким образом, сопоставление между физическим и цифровым мирами ЦД обеспечивает взаимно-однозначное соответствие.

Процесс управления в КС и ЦД

Процесс управления в КС и ЦД условно можно разбить на два этапа:

- 1) управление физическими объектами или процессами, влияющими на кибер/цифровое представление;
- 2) управление кибер/цифровым представлением физических объектов или процессов.

На первом шаге физический мир динамичен, то есть один и тот же объект может демонстрировать разные свойства в разное время. Для поддержания согласованности, данные из физического мира в реальном времени собираются с помощью датчиков и передаются в кибер/цифровой мир для управления кибер/цифровыми элементами в синхронизации с физическим миром. В частности, для ЦД физические данные в реальном времени управляют виртуальными моделями для имитации физического процесса и его развития. На заключительном этапе управления кибер/цифровой мир использует эти данные для вычисления управляющих сигналов и отправки их на исполнительные механизмы для физической реализации. Например, с помощью математических моделей и связанных с ними вычислительных инструментов можно прогнозировать будущие состояния и сбои, что позволяет заранее генерировать более эффективные решения для

обслуживания и управления. В результате некоторые сбои, в том числе непредвиденные и вредоносные, можно устранить с помощью киберфизического интерактивного управления.

Функциональная реализация КС и ЦД

С точки зрения функциональной реализации, КС рассматривает датчики и исполнительные механизмы в качестве основных модулей, в то время как ЦД следуют системно-инженерному подходу, основанному на моделях, и делают упор на данные и модели.

КС тесно связаны с физическими процессами. Интеллект формируется на основе данных. Датчики и исполнительные механизмы необходимы для взаимодействия с физическим миром и обмена данными, что является важнейшей особенностью КС, поскольку они отвечают за определение условий, в которых находятся физические машины и окружающая среда, и выполнение команд управления. Благодаря управлению данными, их обработке и анализу в кибернетическом мире генерируются управляющие команды на основе заранее определённых правил и спецификации семантики управления. Результаты передаются обратно на исполнительные механизмы, которые выполняют операции в соответствии с командами управления, чтобы адаптироваться к изменениям. Шина данных и управления обеспечивает поддержку связи и обмена данными в реальном времени. С помощью датчиков и исполнительных механизмов любые изменения в физическом процессе (например, в поведении или производительности) вызывают изменения в киберпространстве, и наоборот. Таким образом, датчики и исполнительные механизмы можно считать основными элементами КС. Пример функциональной схемы КС представлен на рисунке 1.



Рисунок 1 – Функциональная схема КС

Основная идея ЦД заключается в создании цифровой копии физических объектов для отражения их поведения, прогнозирования и контроля их будущих состояний. Поскольку объекты физического мира динамически меняются, все виды данных постоянно создаются, используются и хранятся с момента создания до утилизации продукта.

Модели, объединяющие геометрию, структуру, свойства материалов, правила позволяют оцифровывать и визуализировать производственную систему/процесс. В сочетании с анализом данных ЦД, появляется возможность делать более точные прогнозы, принимать рациональные решения и осуществлять обоснованное производство [54]. Более того, в процессе совместной эволюции моделей и физических аналогов модели генерируют новые данные, служат механизмом коммуникации и записи, помогая интерпретировать поведение машин или систем, на основе данных получаемых в режиме реального времени, исторических данных, опыта и знаний. Таким образом, модели и данные можно считать основными элементами ЦД.

Заключение

КС и ЦД были предложены примерно в одно и то же время. И КС, и ЦД включают в себя физический и кибернетический миры. Благодаря кибернетическому взаимодействию и контролю они обеспечивают более точное и эффективное управление физическими объектами. Однако в отношении кибермира КС и ЦД делают акцент на разных аспектах: ЦД в большей степени ориентированы на виртуальные модели, которые обеспечивают соответствие один к одному, в то время как КС обеспечивают соответствие один ко многим. Что касается реализации функций КС и ЦД, то датчики и исполнительные механизмы обеспечивают взаимодействие между физическим и кибермирами для обмена данными и управления. Для сравнения, модели играют важную роль в ЦД, помогая интерпретировать и прогнозировать поведение физического мира на основе различных данных. Таким образом, датчики и исполнительные механизмы можно считать основными элементами КС, а модели и данные — основными элементами ЦД.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Negri E, Fumagalli L, Macchi M. A review of the roles of digital twin in CPSbased production systems. *Procedia Manuf* 2017; 11: 939–48.
2. Tao F, Qi Q. New IT driven service-oriented smart manufacturing: framework and characteristics. *IEEE Trans Syst Man Cybern Syst* 2017;49(1):81–91.
3. Kusiak A. Smart manufacturing. *Int J Prod Res* 2018;56(1–2):508–17.
4. Zhong RY, Xu X, Klotz E, Newman ST. Intelligent manufacturing in the context of Industry 4.0: a review. *Engineering* 2017;3(5):616–30.

УДК 004.942

КРЫГИНА М.К., КОШЕЛЕВА М.С.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**ОБЪЕКТЫ МОДЕЛИРОВАНИЯ ЦИФРОВЫХ ДВОЙНИКОВ**

Рассматриваются наиболее популярные классификации цифровых двойников в контексте моделируемых объектов.

Цифровой двойник (Digital Twin) – виртуальный прототип реального физического объекта (или процесса), предназначенный для имитации и прогнозирования поведения своего аналога.

Нередко ЦД создается на этапе проектирования реального объекта, с целью тестирования его функциональности и устойчивости к различным условиям эксплуатации, а также внесения соответствующих корректировок в конструкцию будущего объекта. После создания самого объекта наличие его виртуальной модели остается актуальным. На такой модели можно проводить экспериментальные опыты по усовершенствованию оригинала.

Существует несколько классификаций цифровых двойников, рассмотрим самые популярные из них:

1) Цифровые двойники с точки зрения уровня сложности:

- Прототип – виртуальный аналог объекта или процесса, который включает в себя всю функциональность оригинала для его воссоздания. В отличие от экземпляра, прототип – несуществующий объект, он не может включать в себя эмпирические результаты работы с объектом.

- Экземпляр – содержит данные обо всех характеристиках и эксплуатации физического объекта, включая трехмерную модель, и действует параллельно с оригиналом. В отличие от прототипа оперирует с эмпирическими результатами взаимодействия с объектом. Иными словами, это функционирующая точная копия оригинала, но предназначенная не для эксплуатации, а для проведения опытов над ней.

- Агрегированный двойник – вычислительная система из цифровых двойников и реальных объектов, которыми можно управлять из единого центра и обмениваться данными внутри. Он повторяет весь функционал больших производственно-экономических систем. Используется для обучения персонала работы с функциями оригинала, помогает наладить техническое обслуживание и ремонт оригинала. Также с его помощью можно получить исчерпывающую

информацию об исследуемой среде и сделать выводы о работоспособности системы.

Оптимальная погрешность между работой цифрового двойника и его физического прототипа считают 5% [2]. Следовательно, реальный объект будет иметь такое же поведение в указанных условиях, как его прототип. Этот факт позволяет существенно сократить расходы на внедрение новых технологий в различные сферы.

2) Цифровые двойники с точки зрения оцифрованного объекта:

- Двойник продукта – цифровая копия физического изделия или устройства. Он включает все данные о своем прототипе, историю его изменений и информацию о текущем состоянии. Одним из примеров такого типа ЦД является энергетика, т.е. проектируются виртуальные ветряные мельницы или «ветряные фермы» на основе цифровых технологий, которые позволяют подбирать конфигурацию турбины, исходя из условий ветряной электростанции. Как только турбина приводится в действие, её виртуальная копия начинает собирать и анализировать данные окружающей среды в реальном времени, что приводит к созданию более эффективных моделей [1].

- Двойник процесса – моделирует производственные процессы, логистические цепочки или другие бизнес-процессы. В качестве примера можно привести следующее: в системах массового обслуживания ЦД позволяет имитировать процессы обслуживания клиентов при разных условиях. Такими ЦД пользуются сейчас не только в крупных логистических компаниях для улучшения качества обслуживания клиентов, но и в сетях ресторанов быстрого питания и крупных сетях розничных магазинов.

- Двойник системы используется для моделирования сложных систем, таких как транспортные инфраструктуры или города. Здесь можно привести в пример Сингапур, так как город имеет цифровой двойник – динамическую трёхмерную модель со всеми объектами: от зданий и мостов до бордюров и деревьев. В виртуальный Сингапур поступают данные с городских датчиков и информация от госучреждений [1]. Это помогает оперативно реагировать при возникновении ЧС, а также облегчает городское планирование и дизайн, проектирование дорог, зданий и т.д.

- Двойник организации – модель всей компании или её подразделений. Нужен чтобы обладать абсолютно полной информацией о её работе, анализировать влияние различных внешних и внутренних факторов, внедрять изменения, выбирать лучшие

стратегии, выполнять проекты организационного развития, оптимизации бизнес-архитектуры, автоматизации и роботизации.

- Двойник потребителя – помогает изучать поведение потребителей, их предпочтения и реакции на продукты и услуги. Такой двойник динамичен, то есть учитывает изменения в поведении, контексте взаимодействия и внешних факторах. Позволяет подстраивать ассортимент под покупателя, прогнозирует спрос, помогает тестировать различные маркетинговые стратегии, создавать акционные предложения для привлечения новых клиентов, продвигать товар с малым спросом.

- Двойник окружающей среды используется для оценки влияния строительства новых объектов, разработки экологически чистых технологий и предотвращения природных катастроф. Например, модель территории около п. Коса (Балтийская коса). Для её создания собрали информацию об экологическом состоянии авантюны, проанализировали исторические карты территории для выявления динамики береговой зоны, собрали сведения о городской инфраструктуре.

- Двойник здоровья предназначен для моделирования состояния здоровья одного пациента или группы людей. Например, создание виртуальной копии сердца пациента, страдающего от аритмии. Для этого человек проходит МРТ сердца, а с помощью искусственного интеллекта отсканированные изображения преобразуются в 3D-реконструкцию сердца. Затем экранное сердце заполняется виртуальными клетками, которые могут генерировать электрические сигналы и сердечный ритм. После создания цифрового двойника его используют, чтобы прогнозировать, где находится источник нерегулярного ритма и планировать, как можно это исправить.

- Двойник обучения – интерактивная среда, где студенты могут практиковаться и получать обратную связь без риска ошибок в реальной жизни. Например, симуляторы управления самолетом в летных школах при подготовке пилотов.

- Двойник инфраструктуры моделирует здания, мосты, дороги и другие элементы городской инфраструктуры, чтобы помогать проектировщикам и инженерам оценивать прочность конструкций, оптимизировать энергопотребление и минимизировать риски при эксплуатации объектов.

Чаще всего цифровые двойники создаются с целью моделирования объектов, напрямую связанных с промышленным производством, либо являющихся важным элементом технических систем [2].

В заключение, стоит отметить, что перспективы развития технологии цифровых двойников выглядят многообещающе. Уже сейчас данная концепция внедрена в различных сферах жизни общества, начиная от производства и бизнеса и заканчивая здравоохранением, энергетикой, георазведкой. Один из главных плюсов ЦД – это возможность в короткие сроки протестировать функционал создаваемого продукта на его прототипе. Именно это обуславливает процесс быстрого роста рынка цифровых двойников.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Мятёж, Т. В. Цифровизация систем энергоснабжения мегаполисов и промышленных объектов на основе передовых технологий систем АСКУЭ: учебное пособие / Т. В. Мятёж. – Новосибирск: НГТУ, 2023. – 235 с.

2. Старцева, Ю. В. Теория цифровой экономики: учебно-методическое пособие / Ю. В. Старцева, Л. С. Будович. – Москва: РТУ МИРЭА, 2024. – 123 с.

УДК 004.627

КУЗНЕЦОВ Н.А., СКВОРЦОВ С.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

АНАЛИЗ ВЛИЯНИЯ ОБЪЕМА ОБУЧАЮЩИХ ДАННЫХ НА КОЭФФИЦИЕНТ СЖАТИЯ АЛГОРИТМА ХАФФМАНА

Рассмотрены результаты экспериментов по оценке эффективности алгоритма сжатия текстовых данных в условиях, когда дерево Хаффмана, используемое для построения кодов переменной длины, формируется на основе анализа некоторой части сжимаемого текста. Получена зависимость степени сжатия от доли исходного текста, использованного для построения дерева Хаффмана.

Алгоритм сжатия Хаффмана выполняет построение бинарного дерева, описывающего структуру кодов переменной длины [1]. Он основан на построении оптимального префиксного кода для символов, где часто встречающиеся символы кодируются короткими битовыми последовательностями, а редко встречающиеся символы кодируются более длинными последовательностями. В префиксных кодах ни для какого символа алфавита код не является префиксом другого кода. Это существенно упрощает восстановление сжатого текста. Для этого битовая строка, полученная в результате сжатия, сканируется до

получения группы битов, соответствующей коду некоторого символа. Затем эти биты заменяются соответствующим символом, и действия повторяются со следующего бита до конца строки.

В данной работе представлены результаты эксперимента по оценке эффективности сжатия текстовых данных по алгоритму Хаффмана в условиях, когда для построения дерева Хаффмана и последующего составления таблицы кодов используется только часть исходного текста. Будем считать, что этот фрагмент текста содержит обучающие данные, так как именно эти данные в конечном итоге определяют структуру кодов переменной длины. Целью эксперимента является определение объема обучающих данных, необходимых для построения дерева Хаффмана и позволяющих эффективно сжать исходный текст.

Эксперимент проводился на примере сжатия текста четырех глав романа «Война и мир», включающий в себя реплики на французском языке, чтобы расширить набор используемых символов и дополнительно оценить влияние частот появления символов в тексте на коэффициент сжатия. В тексте 42976 символов, а размер исходного файла $S_{исх} = 73506$ байт.

Заметим, что если обучающий фрагмент текста содержит какие-либо символы, используемые в полном тексте, то абсолютные частоты появления этих символов задаются минимально возможными значениями, равными единице. Их относительные частоты будут зависеть от объема фрагмента текста, по которому строится дерево Хаффмана.

Результаты эксперимента, полученные с использованием разработанной программы сжатия тестовых данных по алгоритму Хаффмана [2], приведены в таблице 1. Для определения коэффициента сжатия использована известная формула [1], где $S_{сжатый}$ – объем сжатого файла.

Таблица 1 – Результаты экспериментов по сжатию текста

№ п/п	Число символов обучающего текста	Отношение длины обучающего текста к длине исходного текста	Размер сжатого файла (в байтах)	Коэффициент сжатия
1	42976	1	42 063	0,43
2	38465	0,89	41 982	0,43
3	29248	0,68	41 931	0,43
4	23757	0,55	41 872	0,43

5	17606	0,41	41 970	0,43
6	12630	0,29	42 195	0,43
7	7225	0,17	42 297	0,42
8	4171	0,1	42 591	0,42
9	2122	0,05	43 894	0,4

График зависимости полученной степени сжатия от доли текста, использованной в качестве обучающих данных для построения дерева Хаффмана, текста, приведён на рисунке 1.

На основе выполненного эксперимента можно сделать вывод, что для эффективного сжатия текста объем обучающих данных может составлять около 30% исходного текста для составления дерева. В рассмотренном примере наилучший результат достигается, когда в качестве обучающих данных используется первая половина сжимаемого текста.

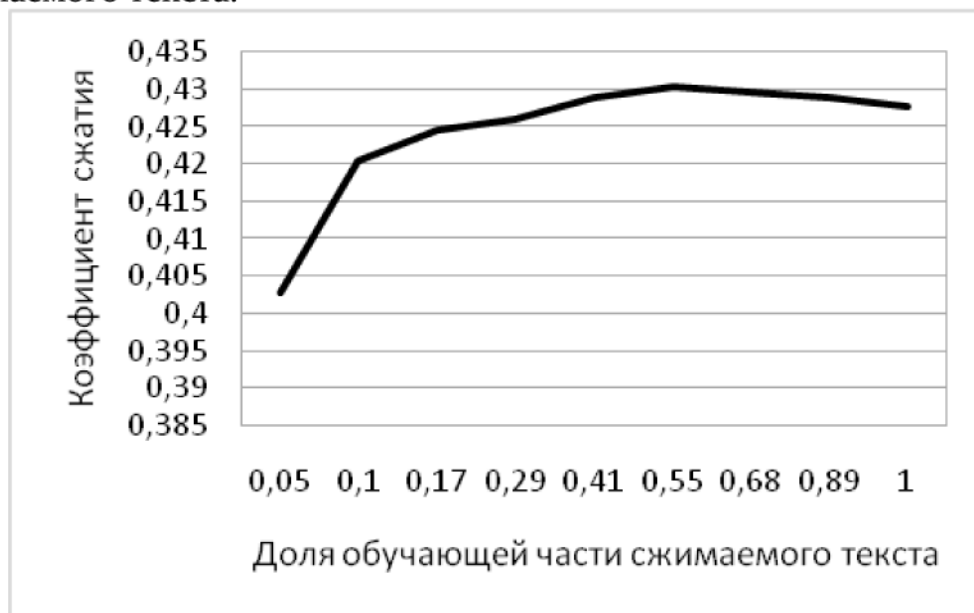


Рисунок 1 – График зависимости степени сжатия от размера обучающей части исходного текста

Выполненные эксперименты также показывают, что чем более разнообразен используемый набор символов исходного текста, тем большая доля обучающих данных требуется для получения дерева Хаффмана. Например, из-за частого использования французского языка в книге «Война и мир» эффективность сжатия увеличивалась только с ростом объема обучающего текста. Таким образом, чем более объёмным и менее разнообразным является исходный текст, тем эффективнее сжатие и незначительнее рост размера формируемой таблицы кодов.

В заключение укажем, что полученные результаты позволяют ускорить работу алгоритма сжатия текстовой информации за счет уменьшения объема данных, используемого для построения дерева Хаффмана и формирования таблицы кодов переменной длины.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Левитин А. Алгоритмы: введение в разработку и анализ. – М.: Издательский дом «Вильямс», 2006. – 576 с.

2. Кузнецов Н.А., Скворцов С.В. Применение алгоритма Хаффмана для сжатия текстовых данных // Информационные технологии в прикладных исследованиях. – Рязань: ИП Коняхин А.В. (BookJet), 2023. – С. 144-147.

УДК 004.89

КУРГАНОВА И.И.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

АНАЛИЗ РОЕВЫХ АЛГОРИТМОВ: МЕТОДЫ, ПРИМЕНЕНИЕ И ТЕНДЕНЦИИ

Рассматриваются методы и применение роевых алгоритмов в задачах оптимизации, робототехники и анализа больших данных.

Введение

Роевые алгоритмы – перспективная область науки, основанная на принципах коллективного поведения в природе, таких как взаимодействие муравьев и пчел. Они применяются для решения задач оптимизации, координации и анализа данных, что особенно актуально в условиях роста объемов информации. Статья анализирует методы и применение роевых алгоритмов, их использование в робототехнике, искусственном интеллекте и оптимизации процессов, а также адаптацию к новым вызовам. Роевые алгоритмы демонстрируют высокую эффективность и значимость в координации автономных систем и анализе больших данных, открывая новые перспективы для дальнейшего развития.

Общие сведения о роевых алгоритмах. Определение и основные характеристики роевых алгоритмов

Роевые алгоритмы представляют собой особую категорию вычислительных методов, которые основываются на моделировании коллективного поведения различных организмов в природе. Эти

алгоритмы черпают вдохновение из таких явлений, как координация действий муравьёв, слаженное движение стай птиц и сложные взаимодействия между пчёлами. Основная идея заключается в том, что простые агенты, следуя локальным правилам и взаимодействуя друг с другом, способны решать сложные задачи, что делает роевые алгоритмы мощным инструментом для поиска оптимальных решений в различных областях.

Ключевыми характеристиками роевых алгоритмов являются децентрализованность, самоорганизация и адаптивность. Эти свойства обеспечивают алгоритмам высокую устойчивость к сбоям и гибкость в условиях изменяющейся среды. Благодаря этому, роевые алгоритмы могут эффективно адаптироваться к новым условиям и находить решения даже в сложных и динамичных ситуациях.

Роевые алгоритмы находят широкое применение в различных областях, таких как оптимизация, робототехника и искусственный интеллект. Например, алгоритм оптимизации роя частиц (PSO) используется для решения задач оптимизации, где необходимо найти наилучшее решение среди множества возможных вариантов. Кроме того, роевые алгоритмы применяются для координации действий групп роботов, таких как дроны, которые могут работать вместе для достижения общей цели.

Эти примеры подчеркивают не только эффективность, но и универсальность роевых алгоритмов в решении сложных задач, что делает их важным инструментом в современном научном и инженерном арсенале. Их способность к самоорганизации и адаптации открывает новые горизонты для разработки инновационных решений в самых различных сферах, от управления ресурсами до разработки новых технологий.

Исторический контекст и развитие роевых алгоритмов

Роевые алгоритмы, вдохновленные коллективным поведением природы, начали развиваться в 1990-х годах. В 1995 году Джеймс Кеннеди и Рассел Эберхарт создали алгоритм оптимизации роя частиц (PSO), основанный на наблюдениях за стайными животными, что стало основой для дальнейших исследований. PSO показал, как взаимодействие простых агентов может эффективно решать сложные задачи. С начала 2000-х годов развитие роевых алгоритмов ускорилося, и в 2004 году Марко Дориго разработал алгоритм муравьиной колонии (ACO), который открыл новые возможности для маршрутизации и распределения ресурсов. Современные достижения включают применение роевых алгоритмов в робототехнике, например,

для координации дронов, что демонстрирует эволюцию идей в этой области.

Методы и подходы в роевых алгоритмах. Классификация методов роевого интеллекта

Роевые алгоритмы – это вычислительные методы, которые основываются на самоорганизации и взаимодействии агентов, что позволяет эффективно решать задачи оптимизации. Существует два основных подхода: биологически вдохновлённые алгоритмы, моделирующие поведение живых организмов, и физически вдохновлённые, использующие принципы физических систем.

К биологически вдохновлённым относится алгоритм муравьиной колонии (ACO), разработанный Марко Дориго в 1992 году, который эффективно решает задачу коммивояжёра, имитируя поиск оптимального пути муравьями. К физически вдохновлённым относится алгоритм роя частиц (PSO), который моделирует движение частиц для нахождения оптимальных решений в задачах непрерывной оптимизации. Оба подхода показывают, как принципы из природы и физики могут быть адаптированы для решения практических задач.

Сравнительный анализ популярных роевых алгоритмов

Роевые алгоритмы, такие как алгоритм роя частиц (PSO) и алгоритм муравьиной колонии (ACO), представляют собой известные и широко используемые методы оптимизации, которые находят применение в различных областях науки и техники. Алгоритм роя частиц моделирует поведение стаи птиц, где каждая частица в рое представляет собой потенциальное решение задачи оптимизации. Этот подход позволяет частицам взаимодействовать друг с другом и обмениваться информацией, что способствует поиску оптимального решения.

С другой стороны, алгоритм муравьиной колонии черпает вдохновение из наблюдений за поведением муравьев, которые используют феромоны для нахождения кратчайших путей к источникам пищи. Этот алгоритм эффективно решает задачи, связанные с поиском оптимальных маршрутов, и может быть применен в различных областях, включая логистику и маршрутизацию.

Оба алгоритма обладают высокой эффективностью, однако их подходы к решению задач оптимизации различаются. PSO отличается простотой реализации и быстрой сходимостью, что делает его особенно подходящим для задач непрерывной оптимизации. Тем не менее, он может сталкиваться с проблемой застревания в локальных

минимумах, что ограничивает его эффективность в некоторых случаях.

В то время как АСО демонстрирует высокую эффективность при решении дискретных задач, таких как задача коммивояжера, его реализация может быть более сложной и требовать значительных вычислительных ресурсов. Сравнительное исследование, проведенное в 2020 году, показало, что PSO более предпочтителен для задач непрерывной оптимизации, тогда как АСО оказывается более эффективным для дискретных задач. Это подчеркивает важность тщательного выбора алгоритма в зависимости от специфики решаемой задачи, что может существенно повлиять на результаты оптимизации и эффективность вычислений.

Адаптивные методы и их преимущества

Адаптивные методы в роевых алгоритмах представляют собой подходы, которые позволяют алгоритмам изменять свои параметры в процессе работы в зависимости от текущих условий задачи. Основным принцип этих методов заключается в динамическом изменении ключевых характеристик алгоритма, таких как скорость обучения, инерционный вес или коэффициенты влияния на поведение агентов. Например, в алгоритме роя частиц используется адаптивная настройка инерционного веса, что позволяет частицам более эффективно исследовать пространство решений на начальных этапах и сосредотачиваться на поиске оптимального решения на поздних этапах. Это делает алгоритм более гибким и способным справляться с изменяющимися условиями среды.

Использование адаптивных методов в роевых алгоритмах предоставляет значительные преимущества, особенно в условиях динамичных и изменяющихся задач. Эти методы позволяют алгоритмам быстрее адаптироваться к новым условиям, что снижает время сходимости и повышает точность найденных решений. Согласно исследованиям, применение адаптивных методов, таких как динамическая настройка параметров, улучшило время сходимости на 15-20% по сравнению с традиционными подходами. Это делает их особенно полезными для задач, где требуется оперативное реагирование на изменения, например, в реальном времени или в условиях неопределенности.

Области применения роевых алгоритмов. Оптимизация процессов в различных сферах

Роевые алгоритмы, такие как алгоритм роя частиц (PSO), являются эффективными инструментами для оптимизации. Разработанный в 1995 году Джеймсом Кеннеди и Расселом

Эберхартом, PSO моделирует поведение роя частиц, которые ищут оптимальное решение, обновляя свои позиции на основе собственных и групповых достижений. Это позволяет находить оптимальные решения в сложных задачах. Важно отметить, что разработка графа решений начинается от основной вершины v_i , что может быть дополнено графическими методами для улучшения анализа и визуализации.

Роевые алгоритмы применяются в различных областях, например, в логистике, где компании, такие как Amazon, используют их для оптимизации маршрутов доставки, что в 2020 году привело к улучшению маршрутов на 15%. Они также находят применение в планировании производства, управлении цепочками поставок и оптимизации энергетических систем, подчеркивая свою универсальность и практическую ценность.

Робототехника и автономные системы

Роевые алгоритмы играют важную роль в координации групп роботов, обеспечивая эффективное взаимодействие для достижения общей цели. Например, алгоритм роя частиц позволяет роботам обмениваться информацией и корректировать действия в реальном времени, что особенно важно в задачах поиска и спасения. Это минимизирует время выполнения задач и повышает их точность. В автономных системах роевые алгоритмы помогают решать сложные задачи, требующие высокой автономности и координации. Исследование 2018 года в Гарварде показало, что более тысячи роботов успешно выполняли коллективные задачи с использованием этих алгоритмов. Они также открывают новые возможности для разработки инновационных решений в различных сферах, включая управление и оптимизацию процессов, как отмечают Рудометкин и Бульгина в своей работе о государственной поддержке инновационных проектов.

Искусственный интеллект и большие данные

Роевые алгоритмы, такие как алгоритм роя частиц (PSO), зарекомендовали себя как весьма эффективные инструменты в области анализа больших данных. Они особенно полезны для задач кластеризации и выявления оптимальных групп, особенно в тех случаях, когда традиционные методы анализа данных оказываются неэффективными. В частности, в медицине эти алгоритмы играют важную роль в обнаружении скрытых закономерностей в данных, что, в свою очередь, способствует улучшению диагностики и повышению качества медицинских услуг.

В контексте машинного обучения алгоритм PSO используется для оптимизации параметров различных моделей, что позволяет значительно увеличить их точность – в некоторых случаях на целых 15%. Ключевыми этапами этого процесса являются генерация начальных положений агентов, которые представляют собой возможные решения, и параллельное вычисление фитнес-функций, что позволяет оценивать качество каждого из решений.

Современные вызовы, связанные с обработкой больших объемов данных, требуют адаптации существующих алгоритмов для эффективной работы с такими массивами информации. Роевые алгоритмы обладают высокой гибкостью и могут быть модифицированы для реализации распределенной обработки данных, что делает их особенно актуальными в условиях, когда данные поступают из различных источников и в реальном времени. Кроме того, использование гибридных подходов, которые комбинируют роевые алгоритмы с другими методами, такими как генетические алгоритмы, значительно повышает их эффективность и позволяет решать более сложные задачи.

Таким образом, подчеркивается важность дальнейших исследований в этой области, поскольку развитие и совершенствование роевых алгоритмов может открыть новые горизонты в анализе больших данных и привести к значительным прорывам в различных сферах, включая медицину, финансы и другие области, где требуется обработка и анализ больших объемов информации.

Современные тенденции и перспективы развития. Адаптация к новым технологиям и вызовам

Роевые алгоритмы, такие как алгоритм муравьиной колонии, демонстрируют высокую степень адаптивности к современным вычислительным вызовам, которые включают в себя не только обработку больших объемов данных, но и интеграцию с передовыми технологиями искусственного интеллекта. Эти алгоритмы обладают значительными преимуществами, особенно когда речь идет о решении сложных задач маршрутизации и оптимизации. Это делает их особенно полезными в условиях работы с массивами информации, где требуется высокая скорость и точность обработки.

Алгоритм муравьиной колонии, в частности, нашел широкое применение в различных сетевых структурах, что наглядно подтверждает его эффективность и надежность. Его децентрализованная природа, в которой множество агентов (или "муравьев") взаимодействуют друг с другом для формирования

коллективного решения, является ключевым фактором, способствующим успешному применению этого алгоритма в таких областях, как анализ больших данных и обучение моделей искусственного интеллекта. Эти процессы часто требуют обработки информации в реальном времени, что ставит перед ними дополнительные требования к скорости и эффективности.

Таким образом, роевые алгоритмы, и в частности алгоритм муравьиной колонии, представляют способны справляться с вызовами, возникающими в условиях современного информационного общества, и открывают новые горизонты для исследований и практического применения в самых различных сферах.

Будущее роевых алгоритмов в научных исследованиях

Роевые алгоритмы, благодаря своей уникальной способности эффективно и быстро решать сложные задачи, находят всё большее применение в междисциплинарных исследованиях, охватывающих широкий спектр научных областей. Одним из наиболее ярких и значимых примеров является их использование в биоинформатике, где они играют ключевую роль в повышении как точности, так и скорости анализа биологических данных.

Например, согласно исследованию, опубликованному в уважаемом научном журнале 'Swarm Intelligence' в 2022 году, применение роевых алгоритмов для прогнозирования структуры белков позволило значительно улучшить точность результатов на целых 15%. Это впечатляющее достижение демонстрирует, как данные алгоритмы могут существенно повысить эффективность научных исследований, особенно в тех областях, которые требуют обработки больших объёмов сложных и разнообразных данных.

В будущем можно ожидать, что использование роевых алгоритмов будет расширяться на такие важные и актуальные области, как экология, медицина и материаловедение. В этих сферах они могут внести значительный вклад в решение актуальных научных задач, таких как моделирование экосистем, разработка новых лекарственных препаратов или создание инновационных материалов. Таким образом, роевые алгоритмы представляют собой мощный инструмент, способный трансформировать подходы к решению сложных проблем в различных научных дисциплинах.

Примеры успешного применения и их анализ

Роевые алгоритмы, такие как алгоритм оптимизации частиц (PSO), демонстрируют высокий потенциал в оптимизации, особенно в энергетическом секторе, где их применение в управлении энергопотреблением в умных сетях снизило затраты на 15%. PSO

обеспечивает точность и гибкость, адаптируясь к изменяющимся условиям, что позволяет беспилотным летательным аппаратам эффективно распределять ресурсы. В робототехнике алгоритм искусственного роя пчёл (ABC) улучшает координацию дронов, увеличивая эффективность выполнения задач на 25% по сравнению с традиционными методами. Ершов подчеркивает, что групповая робототехника, основанная на роевом интеллекте, представляет собой перспективное направление, повышающее надежность систем.

Заключение

Данное исследование охватывает широкий спектр роевых алгоритмов, включая их методы, применение и современные тенденции в данной области. В процессе анализа рассматриваются происхождение, развитие и основные принципы работы этих алгоритмов, с особым акцентом на такие важные области, как оптимизация, робототехника и искусственный интеллект. Роевые алгоритмы зарекомендовали себя как высокоэффективные инструменты для решения сложных задач, благодаря своей адаптивности и способности к самоорганизации, что делает их особенно ценными в условиях динамично меняющейся среды.

Практическое применение роевых алгоритмов наблюдается в различных сферах, таких как логистика, управление и анализ данных, что открывает новые возможности для оптимизации процессов и повышения эффективности. Например, в логистике они могут использоваться для улучшения маршрутов доставки, а в управлении – для оптимизации процессов принятия решений. Перспективы дальнейшего развития этой области включают разработку новых методов, способных эффективно работать с большими объемами данных, а также внедрение роевых алгоритмов в современные системы искусственного интеллекта.

Таким образом, роевые алгоритмы играют важную роль как в науке, так и в технике, демонстрируя свои способности к оптимизации процессов и созданию интеллектуальных систем. Это подчеркивает их актуальность и значимость для современных технологий, а также для исследований и практического применения в будущем.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Ершов Н.М. Распределенные алгоритмы управления роем роботов для прохождения лабиринтов // Суперкомпьютерные дни в России 2016. – М.: RussianSCDays.org, 2016. – С. 847–848.

2. Лебедев Б.К., Лебедев О.Б. Модифицированный роевой алгоритм "муравьиное дерево" в задаче диверсификации

трассировочных ресурсов // Информационные технологии. – 2019. – Т. 25, № 11. – С. 670–678.

3. Романова М.Л., Савченко В.Г. Интеллектуальные алгоритмы как основа моделирования исследовательской деятельности студентов // Инновационные технологии в образовании. – [б. г.]. – С. 303–304.

4. Чураков М., Якушев А. Муравьиные алгоритмы // Exponenta Pro. – 2006. – [Электронный ресурс]. – URL: <http://rain.ifmo.ru/cat/>. (дата обращения 15.04.2025)

5. Орлов А.А., Бурмистров А.В. Решение NP-полных задач на основе метода адаптивного поведения муравьиной колонии // Международный студенческий научный вестник. – 2015. – № 3. – С. 283–284.

УДК 004.658

КУРКИНА М.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

**РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ
АВТОМАТИЗАЦИИ ФОРМИРОВАНИЯ И УЧЕТА
ПЛАТЕЖНЫХ ТРАНЗАКЦИЙ ПО ИСПОЛЬЗОВАНИЮ
ПАРКОВОЧНОГО ПРОСТРАНСТВА**

В статье рассматриваются вопросы разработка программного модуля в виде чат-бота, который является современным и эффективным решением для автоматизации учета и оплаты парковок.

Современные города сталкиваются с растущей проблемой управления парковочным пространством. Увеличение числа автомобилей приводит к перегруженности парковок, что создает необходимость в эффективных решениях для их использования. Одним из таких решений является автоматизация процессов учета и оплаты за парковку.

В рамках данного исследования рассматривается разработка программного модуля, реализованного в виде чат-бота, который автоматически формирует платежные транзакции и предоставляет пользователю ссылку для оплаты.

Целью исследования является разработка и анализ программного модуля, который автоматизирует процессы формирования и учета платежных транзакций для посетителей парковок. Модуль должен быть интегрирован в популярные мессенджеры (например, Telegram

или WhatsApp) и предоставлять удобный интерфейс для взаимодействия с пользователями.

Задачи исследования:

1. Анализ существующих решений:

– Изучение текущих подходов к автоматизации учета и оплаты парковок.

– Определение недостатков существующих систем.

2. Проектирование архитектуры программного модуля:

– Определение основных компонентов системы.

– Разработка алгоритмов работы чат-бота.

3. Разработка программного модуля:

– Реализация чат-бота.

– Интеграция с платежными системами.

– Создание базы данных для учета транзакций.

4. Тестирование и оценка эффективности:

– Проверка работоспособности модуля.

– Анализ удобства использования и производительности.

5. Выводы и рекомендации:

– Формулировка заключения о применимости разработанного решения.

До внедрения цифровых технологий учет парковок осуществлялся вручную через паркоматы или кассы. Такие системы имеют ряд недостатков: неудобство для пользователей (необходимость поиска паркомата), ограниченная доступность (паркоматы могут быть неисправны или отсутствовать), высокая стоимость обслуживания оборудования.

Современные системы управления парковками предлагают мобильные приложения и веб-платформы для оплаты. Однако они также имеют ограничения:

- Необходимость установки дополнительного ПО.

- Сложность интеграции с различными платежными системами.

- Отсутствие персонализированного взаимодействия с пользователем.

Чат-бот представляет собой удобное решение для автоматизации процессов:

- Доступность через популярные мессенджеры без необходимости установки приложений.

- Персонализированное взаимодействие с пользователем.

- Простота интеграции с платежными системами.

- Возможность масштабирования на множество парковок.

Основные компоненты системы:

Компонент 1: Чат-бот.

- Интерфейс для взаимодействия с пользователем.
- Обработка запросов и отправка ссылок для оплаты.

Чат-боты в Telegram представляют собой важный инструмент для автоматизации взаимодействия с пользователями и выполнения множества задач. С их помощью можно эффективно работать с большим числом пользователей, обеспечивая быстрые и точные ответы на запросы, а также предоставляя различные сервисы. Применение чат-ботов в разных сферах бизнеса и услуг уже сегодня приносит ощутимые результаты, а с развитием технологий их возможности будут только расширяться.

Чат-боты являются типичной реализацией искусственного интеллекта, который использует заранее запрограммированные сценарии, правила или методы машинного обучения для генерации ответов на запросы пользователей. Основная цель чат-ботов – автоматизация общения, облегчение взаимодействия с пользователями и повышение эффективности работы с данными.

Компонент 2: Платежный шлюз:

- Интеграция с платежными системами (например, PayPal, Stripe, Яндекс.Касса).
- Формирование безопасных ссылок для оплаты.

Платежный шлюз – это программное обеспечение или сервис, который управляет процессом авторизации платежей и передачи данных о транзакциях между покупателем, продавцом и финансовыми учреждениями (например, банками). Основная функция шлюза – обеспечить безопасную и эффективную обработку электронных платежей, выполняя задачи:

Интеграция с различными платежными системами.

- Защита конфиденциальных данных клиентов и предотвращение мошенничества.
- Обработка транзакций в реальном времени.
- Уведомление участников транзакции (покупателя и продавца) о статусе платежа.

В процессе работы с платежными шлюзами происходит несколько ключевых шагов:

- Передача данных: после того как пользователь выбирает метод оплаты, данные о платежной операции передаются через шлюз.
- Авторизация: Шлюз взаимодействует с системой оплаты, проверяя данные карты или счета пользователя.

- Подтверждение и завершение транзакции: после успешной авторизации платеж подтверждается, и средства списываются с счета покупателя.

Формирование безопасных ссылок для оплаты – это важный процесс, обеспечивающий защиту данных и конфиденциальность при выполнении онлайн-платежей. Такие ссылки обеспечивают безопасность финансовых транзакций, защищая от утечек данных и атак типа «человек посередине» (MITM). Этот процесс включает несколько этапов:

1. Шифрование данных

Для защиты платежных данных от несанкционированного доступа используется протокол HTTPS, который обеспечивает шифрование всей передаваемой информации. Каждая ссылка на оплату генерируется с уникальным идентификатором и защищена с помощью SSL-сертификатов, что предотвращает утечку личных данных клиента.

2. Генерация токенов для транзакций

Многие системы, такие как Stripe и PayPal, используют токенизацию данных, что позволяет снизить риски мошенничества. При этом процесс создания безопасной ссылки на оплату может включать в себя генерацию одноразового токена, который используется для выполнения конкретной транзакции. Это повышает безопасность, так как реальные данные платежных карт не передаются напрямую, а заменяются токенами.

3. Подтверждение через многофакторную аутентификацию

В целях повышения безопасности, особенно при крупных транзакциях, многие платежные системы интегрируют многофакторную аутентификацию (MFA). Это может включать отправку одноразовых кодов на телефон или в электронную почту пользователя, которые нужно ввести для завершения платежа.

4. Период действия ссылок на оплату

Ссылки для оплаты могут быть временными, что снижает риски, связанные с их перехватом или несанкционированным использованием. После завершения транзакции ссылка становится недействительной, и новый запрос на оплату генерирует новую ссылку.

5. Платежные страницы с обязательной верификацией

Некоторые системы, такие как PayPal и Stripe, позволяют создавать специальные защищенные платежные страницы, которые предоставляют пользователю только необходимую информацию и требуют обязательного подтверждения перед выполнением операции.

Эти страницы часто включают дополнительные меры безопасности, такие как защита от фишинга, ограничения по времени и проверку подлинности.

Компонент 3: База данных.

Основные задачи, которые должна решать база данных для такого модуля:

- Учет парковочных мест и их статуса (свободно/занято).
- Хранение информации о пользователях, их транзакциях и платных сессиях.
- Интеграция с платежными системами для обработки платежей.
- Генерация отчетности по транзакциям и парковочным местам.
- Обеспечение безопасности данных пользователей (персональных данных и платежной информации).

Компонент 4: API парковочной системы.

API парковочной системы – это интерфейс для взаимодействия с программным обеспечением парковки, который позволяет внешним приложениям (например, чат-ботам, мобильным приложениям или платёжным сервисам) получать или отправлять данные, связанные с парковкой.

Что может делать API парковочной системы:

1. Учет времени парковки:

API может сообщить, когда автомобиль въехал и выехал с парковки – на основе камеры распознавания номеров или парковочного билета.

2. Проверка статуса:

Позволяет узнать, стоит ли автомобиль на парковке сейчас, сколько времени он уже провёл и какой долг накопился.

3. Получение информации о месте:

API может предоставлять данные о доступных местах на парковке, зонах, тарифах и т.д.

4. Интеграция с платёжными системами:

Позволяет передавать информацию о завершённых оплатах или запрашивать ссылку на оплату через сторонние сервисы.

5. Уведомления и события:

Некоторые API могут отправлять события (например, “машина въехала”) в реальном времени, чтобы инициировать автоматические действия, например, расчёт стоимости или напоминание об оплате.

Данная система приведена на рисунке 1.

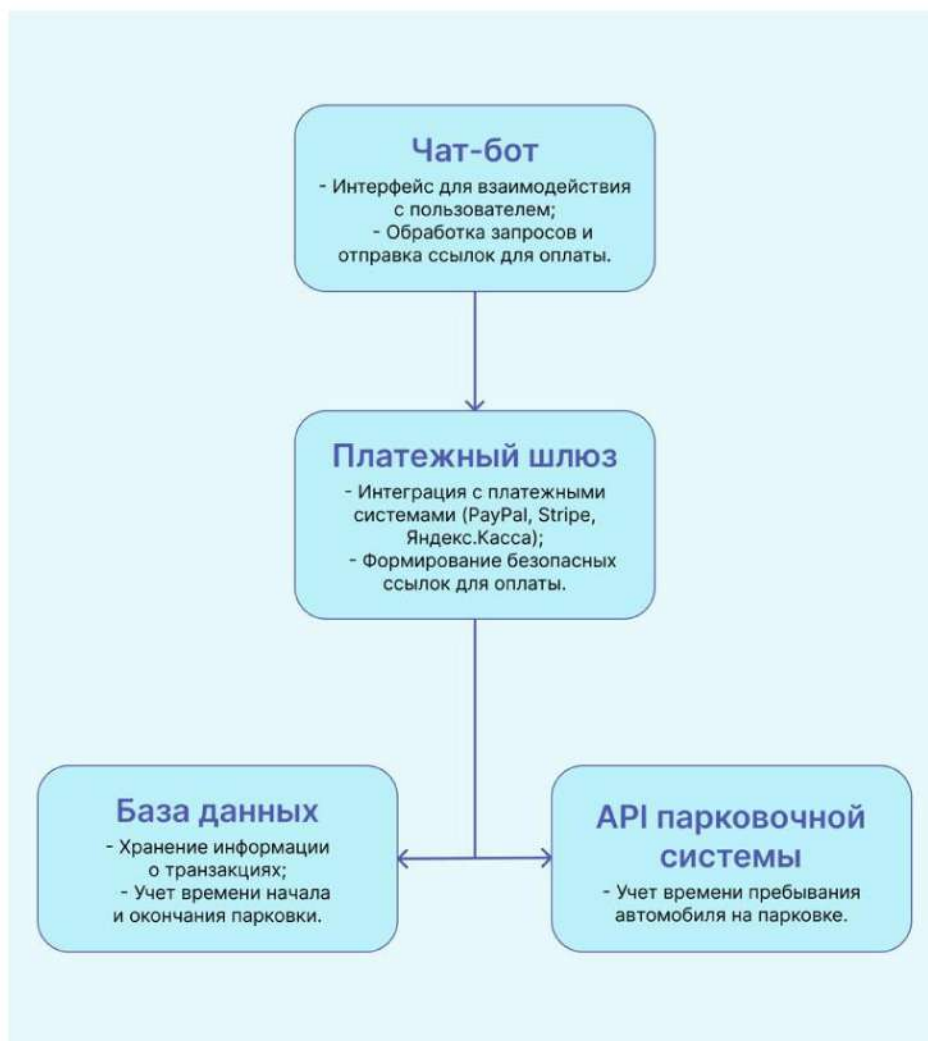


Рисунок 1 – Основные компоненты системы

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Густов А.А. Информационные системы и технологии в экономике и управлении: учебник. – М.: Инфра-М, 2021. – 352 с.
2. Макаров С.В. Архитектура программных систем. – СПб.: Питер, 2020. – 448 с.
3. Голов Р.С. Программные средства автоматизации бизнес-процессов. – М.: КноРус, 2019. – 312 с.
4. Сенин А.Ю. Разработка информационных систем. – М.: Форум, 2022. – 298 с.
5. Павлов Ю.А. Автоматизированные системы управления: теория и практика. – М.: Академия, 2021. – 336 с.
6. Базаров Т.Ю. Платежные системы и технологии электронных расчетов. – М.: Юрайт, 2020. – 280 с.

УДК 004.85

МАРКИН А.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ФУНКЦИЙ АКТИВАЦИИ ПРИ ОБУЧЕНИИ ЛИНЕЙНОГО КЛАССИФИКАТОРА

Рассматриваются особенности обучения линейного классификатора с использованием различных функций активации на синтетических данных.

Введение

В машинном обучении выбор функции активации играет ключевую роль в процессе обучения модели. В данной статье проводится сравнительный анализ трёх функций активации: знаковой, ступенчатой и сигмоидальной. Исследование выполнено на двух наборах синтетических данных с различным количеством точек и плотностью, что позволяет оценить влияние масштаба данных на процесс обучения.

Функции активации – это ключевые компоненты искусственных нейронных сетей и линейных классификаторов, определяющие их способность к обучению и адаптации. Они преобразуют входной сигнал в выходной, вводя нелинейность, что позволяет модели решать сложные задачи, такие как классификация и регрессия. В данной статье рассматриваются три типа функций активации: знаковая, ступенчатая и сигмоидальная, каждая из которых обладает уникальными свойствами:

- Знаковая функция возвращает дискретные значения (-1, 0, 1), что полезно для бинарной классификации.
- Ступенчатая функция (пороговая) аналогична знаковой, но часто используется в перцептронах.
- Сигмоидальная функция обеспечивает плавный переход между значениями (0 и 1), что делает её идеальной для задач вероятностного прогнозирования.

Для демонстрации работы функций активации приведём фрагмент кода на Python, реализующий их:

```
import numpy as np
def sign_function(x):
    return np.where(x > 0, 1, np.where(x == 0, 0, -
1))
def step_function(x):
    return np.where(x >= 0, 1, 0)
```

```
def sigmoid_function(x):
    return 1 / (1 + np.exp(-x))
```

Актуальность исследования обусловлена необходимостью выбора оптимальной функции активации для конкретных задач, особенно при работе с данными разного масштаба и дисбалансом классов. В статье анализируется влияние этих функций на процесс обучения линейного классификатора с использованием синтетических данных.

Методология

Для проведения эксперимента были сгенерированы два набора данных:

1. 500 точек в диапазоне от [0, 0] до [100, 100].
2. 1000 точек в диапазоне от [0, 0] до [142, 142].

Оба набора имеют плотность 0.05 и распределение классов в пропорции 53:47, что имитирует реальные условия с небольшим дисбалансом. Данные были нормализованы для исключения влияния масштаба признаков на веса модели.

Код для создания набора данных

```
import numpy as np
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt

# Параметры генерации
n_samples = 1000 #Количество точек
class_sep = 2.5 #Степень разделимости классов
random_state = 42 #Для воспроизводимости

# Генерация данных
X, y = make_classification(
    n_samples=n_samples,
    n_features=2, #2 признака для
визуализации
    n_redundant=0, #Нет избыточных признаков
    n_clusters_per_class=1, #По одному кластеру на
класс
    flip_y=0, #Без шума
    class_sep=class_sep,
    random_state=random_state
)
y = np.where(y == 0, -1, 1)
X = (X - X.mean(axis=0)) / X.std(axis=0)
```

Код для загрузки и нормализации данных:

```
import numpy as np
data = np.loadtxt('500.csv', delimiter=',')
X = data[:, :2] # Признаки
y = data[:, 2].astype(int) # Метки
```

Нормализация

```
X = (X - X.mean(axis=0)) / X.std(axis=0)
```

Для обучения использовался линейный классификатор с тремя функциями активации:

Знаковая функция:

$$f(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$$

Графическое представление знаковой функции представлено на рисунке 1.

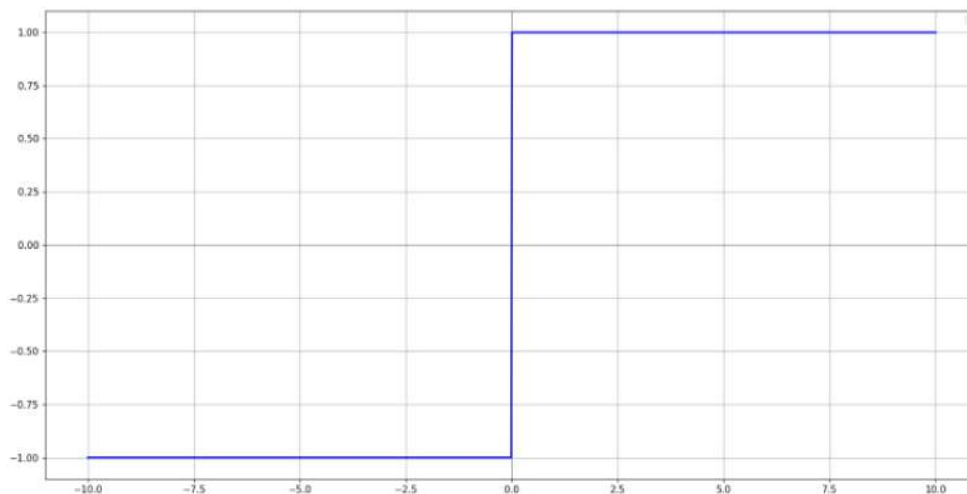


Рисунок 1 – Графическое представление знаковой функции

Ступенчатая функция:

$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

Графическое представление ступенчатой функции представлено на рисунке 2.

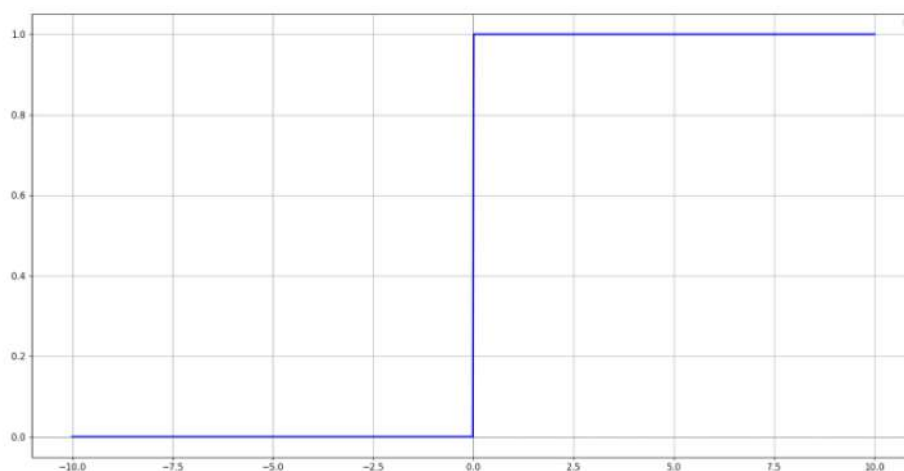


Рисунок 2 – Графическое представление ступенчатой функции

Сигмоидальная функция: $f(x) = \frac{1}{1 + e^{-x}}$

Графическое представление сигмоидальной функции представлено на рисунке 3.

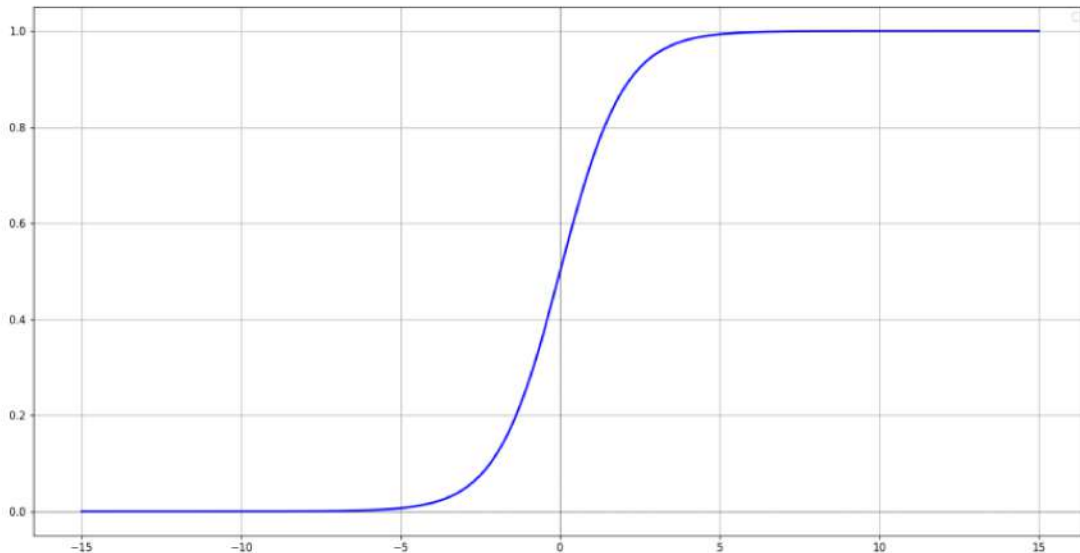


Рисунок 3 – Графическое представление сигмоидальной функции

Обучение проводилось с использованием стохастического градиентного спуска. Для оценки эффективности каждой функции активации фиксировались количество эпох, скорость обучения и конечные значения весов, а также точность классификации.

Код для вычисления точности классификации:

```
def calculate_accuracy(model, X_test, y_test):
    correct = 0
    for x, y_true in zip(X_test, y_test):
        pred, _ = model.activate(x)
        if model.activation_type == 'sign':
            if pred == y_true:
                correct += 1
        else:
            if (pred >= 0.5 and y_true == 1) or (pred <
0.5 and y_true == 0):
                correct += 1
    return correct / len(X_test)
```

Результаты

Знаковая функция активации

- 500 точек:
 - Количество эпох: 120
 - Скорость обучения: 0.3
 - Веса: $w_1=4,30$, $w_2=9,30$, $w_3=-510,8$

- Точность обучения: 100%
- 1000 точек:
- Количество эпох: >500
- Скорость обучения: 0.5
- Веса: $w_1=61,5$, $w_2=61,5$ $w_3=-8749$
- Точность обучения: 100%

При увеличении количества точек в 2 раза наблюдалось значительное изменение весов: w_1 увеличился в 14 раз, w_2 – в 7 раз, w_3 – в 17 раз

Ступенчатая функция активации

- 500 точек:
- Количество эпох: 150
- Скорость обучения: 0.4
- Веса: $w_1=3,80$, $w_2=8,20$, $w_3=-450,60$
- Точность обучения: 100%
- 1000 точек:
- Количество эпох: >600
- Скорость обучения: 0.
- Веса: $w_1=55,0$, $w_2=58,0$, $w_3=-8200,0$
- Точность обучения: 100%

Ступенчатая функция показала схожую динамику изменения весов, но с меньшим коэффициентом увеличения по сравнению со знаковой функцией.

Сигмоидальная функция активации

- 500 точек:
- Количество эпох: 200
- Скорость обучения: 0.5
- Веса: $w_1=5,10$, $w_2=10,20$, $w_3=-600,30$
- Точность обучения: 98%
- 1000 точек:
- Количество эпох: >700
- Скорость обучения: 0.7
- Веса: $w_1=70,0$, $w_2=65,0$, $w_3=-9000,0$
- Точность обучения: 98%

Сигмоидальная функция потребовала большего количества эпох для сходимости, но обеспечила более плавное изменение весов, однако имеет меньший показатель точности обучения.

Результаты показали, что:

Знаковая функция требует наименьшего количества эпох для сходимости, но демонстрирует резкое изменение весов при увеличении объёма данных.

Ступенчатая функция занимает промежуточное положение по устойчивости и скорости обучения.

Сигмоидальная функция обеспечивает плавное обучение, но требует больше вычислительных ресурсов.

Заключение

Проведённый анализ позволил выявить особенности каждой из рассмотренных функций активации. Наилучшие результаты в условиях дисбаланса классов показала сигмоидальная функция, несмотря на её вычислительную сложность. Для задач с большими объёмами данных рекомендуется использовать ступенчатую функцию как компромисс между скоростью и устойчивостью.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Функция активации [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Функция_активации (дата обращения: 20.03.2025)

2. Выбор слоя активации в нейронных сетях: как правильно выбрать для вашей задачи [Электронный ресурс] – URL: <https://habr.com/ru/articles/727506/> (дата обращения: 21.03.2025)

3. Sigmoid Function [Электронный ресурс] – URL: <https://www.sciencedirect.com/topics/engineering/sigmoid-function> (дата обращения: 21.03.2025)

УДК 004.031.43

ПАНИНА А.И.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ТУМАННЫЕ И ГРАНИЧНЫЕ ВЫЧИСЛЕНИЯ – АНАЛИЗ ПРЕИМУЩЕСТВ И ПЕРСПЕКТИВ

Рассматриваются ключевые аспекты туманных и граничных вычислений, их архитектурные особенности, преимущества и различия.

Облачные вычисления — это предоставление ИТ-услуг/ресурсов по требованию через Интернет. Услуга вычислений по требованию через Интернет — это не что иное, как облачные вычисления. С помощью облачных вычислений пользователи могут получать доступ к услугам из любого места и в любое время.

В наши дни каждую секунду по всему миру генерируется огромное количество данных. Компании собирают и обрабатывают эти

данные, поступающие от людей, и получают аналитику для масштабирования своего бизнеса. Когда множество организаций одновременно получают доступ к своим данным на удалённых серверах в центрах обработки данных, может возникнуть трафик данных. Трафик данных может привести к некоторой задержке в доступе к данным, снижению пропускной способности и т. д. Но технология облачного вычисления сама по себе недостаточно эффективна для хранения и обработки огромных объёмов данных и быстрого реагирования.

Например, в беспилотном автомобиле Tesla датчики непрерывно сканируют пространство вокруг транспортного средства. При обнаружении препятствия или пешехода на пути, автомобиль должен немедленно затормозить или изменить траекторию, чтобы избежать столкновения. В такой ситуации критически важно, чтобы данные с датчиков обрабатывались практически мгновенно – даже малейшая задержка в распознавании объекта может привести к аварии. Именно для решения подобных задач применяются технологии граничных (периферийных) и туманных вычислений, обеспечивающих минимальное время отклика.

Туманные и периферийные вычисления в производственных и автоматизированных приложениях – это сетевые и системные архитектуры, которые пытаются собирать, анализировать и обрабатывать данные из активов более эффективно, чем традиционная облачная архитектура.

На рисунке 1 представлена иерархическая пирамида облачных вычислений.



Рисунок 1 – Пирамида облачных вычислений

Как видно из рисунка 1, граничные вычисления охватывают самое большое количество устройств.

Что такое туманные вычисления?

Туманные вычисления – это децентрализованная вычислительная структура, которая включает обработку данных ближе к их источнику. Это помогает сократить задержку сети и способствовать принятию решений в реальном времени. В этой вычислительной инфраструктуре данные, хранилище и приложения располагаются между источником данных и облаком.

Туманные и облачные вычисления разумно рассматривать как взаимодополняющие системные решения, направленные на оптимизацию процессов построения, эксплуатации и дальнейшего развития инфокоммуникационных систем различного назначения.

В этой модели данные обрабатываются и хранятся на устройствах и серверах, расположенных в периферийных или более удаленных сетях, но всегда за пределами локальной сети организации (состоящей из узлов Интернета вещей и периферийных устройств). Это обеспечивает промежуточный уровень между частной пограничной сетью и общедоступным облаком, который служит для получения более быстрого и эффективного отклика системы за счет сокращения задержек связи.

Этот уровень состоит из так называемых туманных узлов, которые обычно состоят из серверов, на которых развернуты, в качестве альтернативы, базы данных для хранения информации, генерируемой пограничными сетями, платформы, предоставляющие различные услуги, устройства интенсивной обработки, приложения для создания информационных панелей для контроля и аналитики, среди других функциональных возможностей, предлагаемых нижним уровням (пограничные вычисления и конечные устройства).

Характеристики, предлагаемые этой моделью, позволяют развернуть архитектуру, которая предлагает услуги и ресурсы распределенным приложениям более гибким и масштабируемым способом. Кроме того, это обеспечивает определенную независимость от сервисов публичного облака, поскольку организация может использовать свои собственные распределенные аппаратные ресурсы для создания уровня туманности, который функционирует как частное облако, предоставляющее сервисы для своих приложений.

Архитектурные особенности туманных вычислений:

– Иерархическая структура: данные обрабатываются на разных уровнях: от датчиков → шлюзам (gateways) → локальным серверам → облаку.

– Гибкость размещения: вычисления могут выполняться на маршрутизаторах, шлюзах, микро-ЦОДах.

– Поддержка разнородных устройств: совместимость с Интернет вещей (IoT), промышленными системами, умными городами.

– Динамическая маршрутизация данных: решения о передаче данных в облако или локальную обработку принимаются на основе текущей нагрузки.

Что такое граничные (периферийные) вычисления?

Граничные вычисления представляют собой децентрализованную вычислительную парадигму, ориентированную на локальную обработку информации в непосредственной близости от её источников. В отличие от традиционных облачных решений, данный подход предполагает выполнение вычислительных операций непосредственно на конечных устройствах или локальных серверных узлах, что минимизирует необходимость передачи данных в удалённые центры обработки.

Ключевой особенностью периферийной-архитектуры является её способность обеспечивать обработку данных в непосредственной близости от места их генерации - на границе сети. Такой принцип организации принципиально отличается как от централизованной облачной модели, так и от промежуточной туманной архитектуры. Основное конкурентное преимущество периферийный-подхода заключается в радикальном сокращении задержек за счёт минимизации маршрутов передачи данных, что обеспечивает практически мгновенный анализ информации и оперативное реагирование.

Особую ценность данная технология представляет для систем, критичных к временным задержкам, включая:

- Автономные транспортные комплексы
- Промышленные автоматизированные системы
- Решения для "умных" городов
- Телемедицинские сервисы

В сравнении с облачными и туманными вычислениями, периферийный-подход демонстрирует наименьшие показатели задержки благодаря возможности обработки информации либо непосредственно на генерирующих устройствах, либо на расположенных вблизи вычислительных узлах.

Архитектура граничных вычислений играет ключевую роль в современной распределённой обработке данных. Её основными достоинствами являются практически нулевые задержки при передаче информации и способность эффективно работать со значительными массивами данных. Однако у этой технологии есть и существенные

недостатки, в частности проблемы с унификацией протоколов и отсутствие единых стандартов. В результате, современные периферийные-устройства и приложения функционируют как изолированные вычислительные системы, слабо интегрированные между собой.

Архитектурные особенности:

– Децентрализация: вычисления происходят прямо на устройствах (камерах, датчиках, телефонах).

– Минимальная задержка: данные не отправляются в облако, что критично для технологий виртуальной и дополнительной реальности (AR/VR), автономных систем.

– Автономность: возможность работы без постоянного подключения к облаку.

– Масштабируемость: легко добавлять новые устройства без перегрузки сети.

Сравнение граничных и туманных вычислений.

Таблица 1 – Сравнение граничных и туманных вычислений

Критерии	Туманные вычисления	Периферийные вычисления
Сложность архитектуры	Более сложный (несколько узлов)	Проще (на уровне устройства)
Место обработки данных	Рядом с источниками данных, но не на устройстве	На самом устройстве
Совместимость	Высокая (совместимость с разными сетями)	Средний (специфичный для устройств)
Обслуживание	Централизованный и масштабируемый	Децентрализованный и специфичный для устройств
Безопасность	Вероятность атак на данные выше	Высокая безопасность. Атаки на данные очень редки

Туманные вычисления отличаются от периферийных тем, что обработка данных происходит не на самих устройствах, а на некотором удалении от них. В то время как периферийные вычисления выполняются непосредственно на датчиках или близлежащих шлюзах, что обеспечивает минимальное расстояние между источником данных и местом их обработки.

Главное преимущество периферийных вычислений заключается в том, что информация обрабатывается локально, без необходимости передачи в облако. Это не только сокращает затраты на передачу данных, но и ускоряет анализ, обеспечивая мгновенную реакцию системы. Кроме того, такой подход повышает уровень безопасности, так как данные не покидают устройство, минимизируя риски их перехвата.

Периферийные вычисления обладают самостоятельностью: они способны обрабатывать информацию для бизнес-задач и отправлять результаты напрямую в облако, не требуя взаимодействия с туманными вычислениями.

Туманные вычисления, в свою очередь, используют данные, собранные периферийными устройствами, и берут на себя часть вычислительной нагрузки. Это позволяет разгрузить как локальные устройства, так и облачные серверы, оптимизируя распределение задач и повышая общую эффективность системы.

Таким образом, эти две технологии дополняют друг друга, обеспечивая гибкость и производительность в различных сценариях использования.

Преимущества туманных вычислений

Минимизация задержек

Туманные вычисления обеспечивают обработку данных вблизи их источника, что существенно сокращает время отклика по сравнению с облачными системами. Это особенно важно для приложений, где требуется мгновенная реакция, таких как промышленная автоматизация и управление умными городами.

Эффективное использование сетевых ресурсов

Локальная обработка данных на туманных узлах уменьшает объем информации, передаваемой в облако, что снижает нагрузку на сеть и оптимизирует использование пропускной способности.

Гибкость и масштабируемость

Архитектура туманных вычислений позволяет легко расширять систему за счет добавления новых узлов без кардинальных изменений в инфраструктуре. Это делает технологию идеальной для масштабируемых IoT-сетей.

Улучшенная безопасность

Обработка данных на промежуточных узлах, а не их прямая передача в облако, снижает риск перехвата информации. Кроме того, распределение данных между узлами усложняет проведение кибератак.

Совместимость с различными устройствами

Туманные вычисления поддерживают работу с широким спектром устройств, включая IoT-датчики, промышленное оборудование и умные гаджеты, что делает их универсальным решением для разных отраслей.

Автономная работа

Туманные узлы способны функционировать даже при временной потере связи с облаком, обеспечивая бесперебойную работу критически важных систем.

Преимущества граничных вычислений

Повышение операционной эффективности

Граничные вычисления не только собирают данные, но и выполняют их обработку и анализ непосредственно на устройстве. Благодаря скорости выполнения операций (в пределах миллисекунд) эта технология становится незаменимой для оптимизации сложных технических процессов.

Экономия ресурсов

Передача больших объемов данных в реальном времени может быть затратной, особенно для удаленных объектов. Граничные вычисления решают эту проблему, перенося аналитику непосредственно к оборудованию, что снижает затраты на передачу данных.

Снижение эксплуатационных расходов

В отличие от облачных решений, граничные вычисления требуют меньше ресурсов для передачи данных, уменьшая зависимость от пропускной способности и сокращая задержки. Это позволяет эффективно обрабатывать большие объемы информации без необходимости их полной отправки в облако.

Повышение уровня безопасности

Локальная обработка данных минимизирует риски утечки конфиденциальной информации, так как она остается в пределах локальной сети. Это делает граничные вычисления привлекательными для компаний, уделяющих особое внимание защите данных.

Снижение зависимости от облачных технологий

Современные устройства обладают достаточной вычислительной мощностью для локальной обработки данных. Граничные вычисления позволяют максимально использовать этот потенциал, ускоряя обработку информации и повышая общую эффективность работы.

Оптимизация производительности

Данный подход не только ускоряет процессы, но и обеспечивает гибкость, безопасность и экономическую выгоду, что делает его востребованным среди предприятий.

Таким образом, обе технологии – туманные и граничные вычисления – играют ключевую роль в современных распределенных системах, дополняя друг друга и обеспечивая высокую эффективность, безопасность и экономичность.

Будущие тенденции в области туманных и периферийных вычислений

Аналитики ожидают резкое увеличение популярности периферийных вычислений в обозримом будущем. Этот тренд обусловлен рядом ключевых факторов:

1. Экспоненциальный рост IoT-устройств

Согласно исследованиям Gartner, к 2025 году число подключенных IoT-устройств превысит 75 млрд единиц, создавая потребность в эффективных решениях для локальной обработки информации непосредственно у источников её генерации.

2. Развёртывание сетей пятого поколения

Технология 5G с её высокой скоростью передачи данных и минимальной задержкой сигнала создаёт идеальные условия для работы распределённых периферийных-систем, требующих мгновенной обработки информации.

3. Интеграция технологий ИИ и машинного обучения

Возможность выполнения сложных алгоритмов искусственного интеллекта непосредственно на периферийных-устройствах позволяет ускорить принятие решений и оптимизировать нагрузку на централизованные сети.

4. Появление инновационных сфер применения

Перспективные направления использования включают беспилотный транспорт, концепции "умных городов", дистанционную медицину, автоматизированные производственные комплексы и другие передовые технологические области.

Туманные вычисления также имеют перспективы развития. По мере того, как сети становятся быстрее, а IoT становится сложнее, туманные вычисления будут адаптироваться. Искусственный интеллект сделает системы туманных вычислений более независимыми и проактивными.

Ожидается, что в будущем облачные, туманные и граничные вычисления будут образовывать единую интеллектуальную экосистему, где эти технологии не конкурируют, а дополняют друг друга.

Заключение

Туманные и граничные вычисления представляют собой важный этап эволюции вычислительных технологий, дополняя и расширяя

возможности облачных решений. Их ключевые преимущества — снижение задержки, оптимизация нагрузки на сеть, гибкость и повышенная безопасность — делают их незаменимыми для современных IoT-систем, промышленной автоматизации и других областей, где требуется быстрая обработка данных.

В будущем эти технологии будут развиваться в тесной взаимосвязи с облачными вычислениями, формируя единую интеллектуальную экосистему. Развитие сетей 5G, увеличение количества IoT-устройств и внедрение искусственного интеллекта откроют новые горизонты для их применения, обеспечивая ещё большую эффективность и автономность.

Таким образом, туманные и граничные вычисления — это не просто временный тренд, а фундаментальные технологии, которые будут определять развитие цифровой инфраструктуры в ближайшие десятилетия.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Джордж Дж. Облачные вычисления. — СПб.: БХВПетербург. 2011. — 288 с.
2. Глушак Е. В. Облачные и туманные вычисления: архитектура, моделирование, применение. — Вологда, 2025. — 36 с.
3. В.П. Куприяновский, Д.Е. Намиот, О.Н. Покусаев Периферийные вычисления для современной мобильности // International Journal of Open Information Technologies. — 2025. — №2. — С.94-97.

УДК 004.8

ПЕРЕПЕЛКИН А.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ

Мультиагентные системы как направление искусственного интеллекта.

В современном мире технологии искусственного интеллекта (ИИ) стремительно развиваются, трансформируя многие сферы человеческой деятельности. Одним из наиболее перспективных направлений в этой области являются мультиагентные системы (МАС), представляющие собой совокупность взаимодействующих интеллектуальных агентов, способных к автономному

функционированию, принятию решений и адаптации в изменяющихся условиях.

Актуальность исследования мультиагентных систем обусловлена несколькими факторами. Во-первых, традиционные централизованные системы управления зачастую демонстрируют недостаточную эффективность при решении сложных задач в условиях неопределенности и высокой динамики среды. Во-вторых, современные технологические вызовы требуют разработки более гибких, масштабируемых и адаптивных решений, способных функционировать в распределенной среде. В-третьих, развитие концепции Интернета вещей, облачных вычислений и повсеместной цифровизации создает благоприятные условия для внедрения мультиагентных технологий.

Понятие и определение мультиагентных систем

Мультиагентная система (MAS, англ. Multi-agentsystem) – это система, образованная несколькими взаимодействующими интеллектуальными агентами. Согласно определению, представленному в работе Майкла Вулдриджа, мультиагентные системы могут быть использованы для решения таких проблем, которые сложно или невозможно решить с помощью одного агента или монолитной системы.

Ключевой особенностью мультиагентных систем является то, что они состоят из множества автономных агентов, каждый из которых обладает собственными целями, предпочтениями и возможностями. При этом агенты способны взаимодействовать друг с другом, обмениваться информацией и координировать свои действия для достижения как индивидуальных, так и общих целей.

Характеристики мультиагентных систем

В многоагентной системе агенты имеют несколько важных характеристик:

- Автономность: агенты, хотя бы частично, независимы и способны самостоятельно принимать решения.
- Ограниченность представления: ни у одного из агентов нет полного представления о всей системе, или система слишком сложна, чтобы знание о ней имело практическое применение для агента.
- Децентрализация: в системе отсутствуют агенты, управляющие всей системой в целом.
- Социальное поведение: агенты способны взаимодействовать друг с другом, обмениваться информацией и координировать свои действия.

– Адаптивность: агенты могут изменять свое поведение в зависимости от изменений внешней среды и действий других агентов.

Эти характеристики позволяют мультиагентным системам проявлять такие свойства, как самоорганизация, эмерджентность (возникновение новых свойств на системном уровне) и способность к адаптации в сложных и динамически изменяющихся условиях.

Типы агентов в мультиагентных системах

В зависимости от сложности и функциональности, агенты в мультиагентных системах могут быть разделены на несколько типов:

1. *Реактивные агенты* – действуют по принципу "стимул-реакция", реагируя на изменения в окружающей среде без сложных рассуждений.

2. *Преактивные агенты* – способны предвидеть изменения в среде и планировать свои действия с учетом этих изменений.

3. *Интенциональные агенты* – обладают собственными целями, убеждениями и намерениями (BDI – Beliefs, Desires, Intentions).

4. *Социальные агенты* – способны взаимодействовать с другими агентами, учитывая социальные нормы и правила.

5. *Обучающиеся агенты* – способны адаптировать свое поведение на основе опыта и обучения.

В современных мультиагентных системах, особенно основанных на больших языковых моделях, агенты часто сочетают в себе несколько типов, что позволяет им проявлять более сложное и гибкое поведение.

Архитектура мультиагентных систем

Архитектура мультиагентных систем может быть разделена на три основных типа:

1. *Централизованная архитектура* – предполагает наличие центрального агента или группы центральных агентов, координирующих коммуникацию и принятие решений в системе. Этот тип архитектуры обеспечивает больший контроль над процессом, но может создавать узкие места в системе.

2. *Распределенная (децентрализованная) архитектура* – функционирует как одноранговая сеть, где агенты напрямую общаются друг с другом без центрального управления. Такая структура более устойчива к сбоям и масштабируема, но может быть сложнее в координации.

3. *Иерархическая архитектура* – агенты организованы в команды или иерархические уровни, где каждый уровень имеет свои задачи и полномочия. Этот подход сочетает преимущества централизованной и децентрализованной архитектур.

Выбор архитектуры зависит от конкретных задач, которые должна решать мультиагентная система. Например, для команд разработчиков продуктов лучше всего подходят централизованная или иерархическая архитектура, поскольку они обеспечивают больший контроль над процессом.

Коммуникация между агентами

Коммуникация является ключевым аспектом функционирования мультиагентных систем. Она позволяет агентам обмениваться информацией, координировать действия и достигать общих целей. Можно выделить несколько парадигм коммуникации:

- *Сотрудничество* – агенты работают вместе для достижения общей цели, обмениваясь информацией для улучшения коллективного решения.

- *Обсуждение/дебаты* – агенты представляют и защищают свои точки зрения, критикуя позиции других агентов.

- *Соревнование* – агенты конкурируют друг с другом, стремясь к достижению собственных целей, которые могут противоречить целям других агентов.

Для обеспечения эффективной коммуникации агенты используют специальные языки и протоколы, такие как KQML (Knowledge Query Manipulation Language) и ACL (Agent Communication Language), разработанный FIPA (Foundation for Intelligent Physical Agents).

Механизмы координации и принятия решений

Координация действий агентов и принятие коллективных решений осуществляются с помощью различных механизмов:

1. Переговоры – агенты взаимодействуют друг с другом для достижения взаимовыгодного соглашения. Например, модель "Запрос – Ответ – Соглашение", где сначала агенты запрашивают информацию или услуги, затем получают ответы и в итоге достигают соглашения.

2. Планирование – агенты формируют планы действий с учетом планов и действий других агентов.

3. Распределение ресурсов – агенты согласовывают распределение ограниченных ресурсов для достижения оптимального результата.

4. Обмен знаниями – агенты делятся информацией и опытом, что позволяет системе в целом накапливать и использовать коллективные знания.

В современных мультиагентных системах на основе LLM принятие решений часто основывается на сложных алгоритмах обработки естественного языка и машинного обучения, что позволяет

агентам более эффективно взаимодействовать и решать сложные задачи.

Самоорганизация и эмерджентное поведение

Одним из наиболее интересных аспектов мультиагентных систем является их способность к самоорганизации и проявлению эмерджентного поведения, когда система в целом демонстрирует свойства и поведение, которые не присущи отдельным ее компонентам.

Самоорганизация в МАС возникает благодаря локальным взаимодействиям агентов, которые формируют глобальные структуры, влияющие в свою очередь на поведение образовавших их локальных агентов (принцип Кауфмана). Это явление "двойной спирали" в принятии решений позволяет системе адаптироваться к изменяющимся условиям без внешнего управления.

Эмерджентное поведение может проявляться в виде "интеллекта роя" (Swarm Intelligence), когда система в целом демонстрирует сложное и адаптивное поведение, несмотря на относительную простоту составляющих ее агентов. Классическими примерами такого поведения являются колонии муравьев или стаи птиц, где коллективное поведение позволяет решать сложные задачи, недоступные для отдельных особей.

Современное состояние развития мультиагентных систем

В последние годы развитие больших языковых моделей (Large Language Models, LLM) открыло новые возможности для создания мультиагентных систем. LLM-MA (Large Language Model based Multi-Agents) представляют собой системы, в которых агенты основаны на мощных языковых моделях, таких как GPT-4.

Основным преимуществом LLM-MA является способность агентов понимать и генерировать человекоподобные инструкции, что значительно расширяет возможности взаимодействия и принятия решений. В отличие от традиционных мультиагентных систем, где каждый агент обычно специализируется на узкой задаче, агенты на основе LLM могут быть более универсальными и гибкими.

В таких системах используются различные подходы к организации агентов, включая:

- Специализацию агентов на различных этапах решения задачи
- Использование агентов с разными ролями и профилями
- Создание агентов с различными уровнями доступа к знаниям

Примерами современных платформ для создания LLM-MA являются MetaGPT, CAMEL и Autogen, которые предлагают инструменты для разработки мультиагентных систем на основе языковых моделей.

Текущие проблемы и вызовы

Несмотря на значительный прогресс, развитие мультиагентных систем сталкивается с рядом серьезных проблем:

- Проблема "галлюцинаций" – LLM склонны генерировать неверную или недостоверную информацию, что может усугубляться в мультиагентных системах из-за взаимодействия агентов и распространения ошибок.

- Вычислительная сложность – каждый агент на основе LLM требует значительных вычислительных ресурсов, что ограничивает масштабируемость системы.

- Ограниченность мультимодального восприятия – большинство современных LLM-МА сосредоточены на текстовом анализе, оставляя за рамками внимания другие модальности.

- Сложность обучения и настройки – системы LLM-МА в основном учатся на основе мгновенной обратной связи, что требует надежной интерактивной среды, которая часто недоступна.

- Проблемы оценки эффективности – оценка мультиагентных систем представляет собой сложную задачу из-за отсутствия стандартизированных метрик и эталонных показателей.

Тенденции развития

Анализ текущего состояния развития мультиагентных систем позволяет выделить следующие тенденции:

- Повышение автономности агентов – разработка более автономных агентов, способных самостоятельно ставить и достигать цели, адаптироваться к изменяющимся условиям.

- Улучшение коммуникации между агентами – развитие более эффективных протоколов и методов обмена информацией между агентами.

- Интеграция с реальным миром – создание агентов, способных взаимодействовать с физическим миром через различные интерфейсы (роботы, умные устройства).

- Развитие мультимодальных систем – переход от чисто текстовых к системам, работающим с различными типами данных (текст, изображения, аудио, видео).

- Повышение объяснимости и прозрачности – развитие методов, позволяющих понять и объяснить принципы работы и решения, принимаемые мультиагентными системами.

*Применение мультиагентных систем**1. Мультиагентные системы в промышленности и логистике*

Мультиагентные технологии нашли широкое применение в промышленности и логистике, где они позволяют решать сложные задачи координации и оптимизации:

- Управление производственными процессами – мультиагентные системы позволяют оптимизировать загрузку оборудования, планировать производственные процессы и адаптироваться к изменяющимся условиям.

- Управление цепочками поставок – МАС обеспечивают гибкое управление логистическими цепочками, позволяя быстро адаптироваться к изменяющемуся спросу, проблемам с поставками и другим непредвиденным ситуациям.

- Планирование транспортных маршрутов – мультиагентные системы позволяют оптимизировать маршруты доставки с учетом множества факторов: времени, стоимости, загруженности дорог, требований к транспортным средствам и т.д.

- Умные электросети – использование МАС для управления распределением электроэнергии, интеграции возобновляемых источников и оптимизации энергопотребления.

Преимущества использования мультиагентных систем в этих областях заключаются в их способности работать в режиме реального времени, адаптироваться к изменяющимся условиям и находить оптимальные решения в сложных динамических средах.

2. Применение в бизнесе и финансах

В сфере бизнеса и финансов мультиагентные системы используются для решения разнообразных задач:

- Торговые системы – МАС применяются для алгоритмической торговли на финансовых рынках, где агенты анализируют рыночные тренды, принимают решения о покупке и продаже активов.

- Управление рисками – мультиагентные системы анализируют различные аспекты рисков и помогают принимать обоснованные решения по управлению ими.

- Персонализированные финансовые услуги – МАС анализируют финансовую историю клиентов, рыночную ситуацию и личные обстоятельства для создания индивидуальных финансовых рекомендаций.

- Обнаружение мошенничества – мультиагентные системы эффективно выявляют подозрительные операции и потенциальные случаи мошенничества благодаря способности анализировать большие объемы транзакций в реальном времени.

3. Мультиагентные системы в науке и образовании

В научных исследованиях и образовании мультиагентные системы также находят разнообразное применение:

- Моделирование сложных систем – МАС используются для моделирования социальных, экономических, биологических и других сложных систем, позволяя изучать их поведение и эволюцию.

- Научное сотрудничество – мультиагентные системы помогают организовать совместную работу исследователей, обмен данными и координацию научных проектов.

- Персонализированное обучение – МАС адаптируют образовательный контент и методики к индивидуальным особенностям учащихся, помогая повысить эффективность обучения.

- Виртуальные лаборатории – использование мультиагентных систем для создания виртуальных сред для проведения экспериментов, особенно в областях, где реальные эксперименты дороги или опасны.

4.4. Перспективные области применения

Потенциал мультиагентных систем выходит далеко за рамки текущих приложений и открывает перспективы для решения передовых задач в различных областях:

- Умные города – МАС могут использоваться для управления городской инфраструктурой, транспортными потоками, энергетическими системами и другими аспектами городской жизни.

- Здравоохранение – мультиагентные системы могут применяться для персонализированной медицины, координации работы медицинского персонала, мониторинга здоровья пациентов и оптимизации использования медицинских ресурсов.

- Экология и устойчивое развитие – МАС могут помочь в мониторинге окружающей среды, прогнозировании климатических изменений и разработке стратегий устойчивого развития.

- Автономные транспортные системы – мультиагентные системы играют ключевую роль в развитии беспилотного транспорта, позволяя координировать движение множества автономных транспортных средств.

Прогнозы развития технологии

По оценкам экспертов, к 2030 году мультиагентные технологии будут служить основой для более чем 40% всех мобильных приложений.

В заключение, хотелось бы отметить, что в ближайшем будущем можно ожидать дальнейшего развития мультиагентных технологий в направлении повышения уровня автономности и интеллекта агентов, интеграции с Интернетом вещей и развития коллективного интеллекта.

Это откроет новые перспективы для применения мультиагентных систем в различных областях.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Вулдридж М.Дж. «Рассуждения о рациональных агентах». – Массачусетс: The MIT Press, 2000. – 227 с.
2. Поспелов Д.А. Мультиагентные системы. – М.: Наука, 2003. – 340 с.
3. Таланцев А.А., Хорошевский В.Ф. Многоагентные системы и технологии. – М.: Горячая линия – Телеком, 2010. – 400 с.
4. Редько В.Г. Эволюция, нейронные сети, интеллект. – М.: URSS, 2005. – 280 с.

УДК 004.72

ПЕРЕПЕЛКИН Д.А., КОВЕРДЯЕВ А.И.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ДИНАМИЧЕСКОЕ УПРАВЛЕНИЕ ПОТОКАМИ ДАННЫХ В МУЛЬТИПРОВАЙДЕРНЫХ СЕТЯХ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

В статье рассматривается использование генетического алгоритма для поиска оптимальных маршрутов в мультиграфах, моделирующих мультипровайдерные сети. Предложенная модель позволяет учитывать множество факторов, таких как стоимость рёбер и вершин, и динамически адаптироваться к изменяющимся условиям сети. Проводится исследование выбора оптимальных параметров работы генетического алгоритма.

Современные мультипровайдерные компьютерные сети характеризуются высокой степенью сложности и динамичностью. Для эффективного управления потоками данных необходимо разрабатывать алгоритмы, способные учитывать разнообразные параметры сети и находить оптимальные маршруты с учетом изменяющихся условий. В данной статье предлагается использовать генетический алгоритм для поиска кратчайших путей в мультиграфе, моделирующем мультипровайдерную сеть.

Целью настоящей работы является разработка и исследование применения генетического алгоритма для нахождения оптимальных маршрутов в мультипровайдерных компьютерных сетях, с учётом разнообразных параметров сети, таких как стоимость передачи данных и изменения провайдеров связи.

Примеры применения

Оптимизация маршрутизации в корпоративной сети: В компании, имеющей несколько Интернет-провайдеров и несколько внутренних сетей, необходимо минимизировать задержки и стоимость передачи данных между офисами. Генетический алгоритм (ГА) помогает динамически адаптировать маршруты под изменяющиеся нагрузки и тарифы провайдеров.

Маршрутизация в облачных средах: В облачной инфраструктуре требуется балансировать нагрузку между несколькими дата-центрами, учитывая стоимость передачи и задержку сети. Применение ГА позволяет выбирать маршруты, минимизирующие затраты на передачу между разными провайдерами.

Транспортные сети 5G: В сетях нового поколения 5G, имеющих сложную архитектуру с множеством уровней коммутации, ГА может быть использован для оптимизации передачи данных между базовыми станциями и ядром сети, учитывая ограничения пропускной способности и стоимость передачи между слоями.

Постановка задачи

Мультипровайдерная компьютерная сеть состоит из множества коммуникационных узлов, к которым подключаются пользователи сети. Рассмотрим задачу нахождения оптимального маршрута в сети с несколькими провайдерами связи. В качестве математической модели будем использовать неориентированный взвешенный связный мультиграф:

$$G = (V, E),$$

где V – множество вершин (узлов) сети; E – множество рёбер (соединений) между вершинами, каждое ребро e_{ij}^k связано с весом w_{ij}^k , где $i, j \in V$, и k – индекс провайдера.

Граф представляет собой мультиграф, где для каждой пары вершин i, j могут быть несколько рёбер, принадлежащих разным провайдерам.

Необходимо найти путь от начальной вершины s к конечной вершине t , минимизируя суммарные затраты. Затраты пути зависят от:

- веса рёбер w_{ij}^k , который представляет стоимость передачи данных через каналы связи
- стоимости вершин: каждый узел v_i имеет вес $weight_i^k$, который зависит от провайдера k и влияет на стоимость прохождения через узел.

Пусть $P = (v_1, v_2, \dots, v_n)$ – путь от вершины $v_i = s$ к вершине $v_n = t$. Стоимость маршрута $C(P)$ определяется как сумма стоимости рёбер и стоимости вершин, через которые проходит путь.

Стоимость рёбер: Для каждого ребра e_{ij}^k из P , где v_i и v_j – соседние вершины, стоимость вычисляется как сумма веса рёбер w_{ij}^k и стоимости вершин:

$$C(P) = \sum_{(i,j,k) \in P} w_{ij}^k + \sum_{(v_i, k_1, k_2) \in P} weight_i^{k_1} + weight_j^{k_2}$$

Смена провайдера: если при переходе между рёбрами происходит смена провайдера, то стоимость вершины увеличивается, так как учитываются дополнительные затраты на коммутацию между провайдерами. Для каждого шага i в пути, если текущий провайдер отличается от предыдущего, то к стоимости добавляется вес двух узлов v_i и v_j :

$$C_{path} = weight_i^{k_1} + weight_j^{k_2}$$

Генетический алгоритм

Генетический алгоритм (ГА) – это метод оптимизации, основанный на принципах естественного отбора и генетики. Он используется для нахождения приближённых решений сложных задач оптимизации и поиска. Обобщенная схема работы алгоритма приведена на рисунке 1.

Рассмотрим более подробно каждый этап:

Инициализация популяции

Популяция состоит из случайных путей, каждый из которых представляет собой последовательность вершин $P = (v_1, v_2, \dots, v_n)$, которая связывает начальную вершину s и конечную вершину t . Пути генерируются случайным образом, с ограничением на количество попыток поиска пути.

Оценка приспособленности (fitness)

Для каждого пути P вычисляется его стоимость $C(P)$, которая зависит от стоимости рёбер и вершин, через которые проходит путь. Функция приспособленности для пути P определяется как:

$$F(P) = \frac{1}{C(P)},$$

где $C(P)$ – это суммарная стоимость маршрута.

Селекция

Для создания следующего поколения используется турнирный отбор. Из случайных m особей (путей) выбираются лучшие n для скрещивания. Вероятность выбора особи пропорциональна её приспособленности:

$$P_i = \frac{F(P_i)}{\sum_{j=1}^N F(P_j)},$$

где P_i – вероятность выбора пути i , а $F(P_i)$ – его приспособленность.



Рисунок 1 – Укрупненная блок-схема генетического алгоритма

Кроссовер (скрещивание)

Для двух выбранных родителей P_1 и P_2 применяется односточечный кроссовер. Путь разбивается на части, и части родителей комбинируются, образуя двух потомков:

$$P_{child1} = P_1[:i] + P_2[i:]$$

$$P_{child2} = P_2[:i] + P_1[i:],$$

где i — точка пересечения, выбранная случайным образом из пересекающихся элементов путей.

Если возникает цикл или некорректный маршрут, применяется процедура ремонта пути: удаляются дублирующиеся вершины, и путь корректируется.

Мутация

Для каждого пути с вероятностью P_m выполняется мутация. Мутация заключается в том, что случайные вершины в пути заменяются на другие допустимые вершины, а путь затем восстанавливается:

$$P_{mutated} = P[:i] + subpath(i, j) + P[j:],$$

где $subpath(i, j)$ – случайно сгенерированный путь между случайно выбранными вершинами $i, j \in P$.

Исследование параметров генетического алгоритма

Сгенерируем мультиграф с двумя провайдерами, который содержит 100 вершин 150 ребер. Для этого графа заранее посчитан оптимальный маршрут от вершины 0 до вершины 75 с помощью метода динамического программирования (Листинг 1).

Листинг 1. Оптимальный маршрут исследуемого графа.

Путь: [0, 96, 58, 31, 45, 74, 75]

Подробности маршрута:

0 -> 96 via provider 0 (edge: 21, vertexes: 41+61) = 123

96 -> 58 via provider 0 (edge: 31, vertexes: 44) = 75

58 -> 31 via provider 1 (edge: 37, vertexes: 86+81) = 204

31 -> 45 via provider 1 (edge: 43, vertexes: 27) = 70

45 -> 74 via provider 1 (edge: 37, vertexes: 63) = 100

74 -> 75 via provider 1 (edge: 24, vertexes: 82) = 106

Стоимость пути: 678

Будем изменять значения размера популяции от 5 до 35 с шагом 5; количества поколений от 40 до 100 с шагом 15; вероятности мутации от 0.1 до 0.9 с шагом 0.1.

Из получившегося набора выберем те комбинации параметров, которые достигли оптимального результата, и отсортируем их по

возрастанию времени выполнения. Результаты приведены на рисунке 2.

Из анализа полученных результатов можно заметить, что запуском алгоритмов с более высокой скоростью выполнения соответствует более высокая вероятность мутации, которая обеспечивает появление более приспособленных особей (рисунок 3). Оптимальное значение вероятности мутации таким образом находится в диапазоне 0.7-0.9.

Также из анализа данных следует, что алгоритмам с более высокой скоростью выполнения соответствуют популяции с меньшим количеством особей (оптимальное значение 10-15). Данная тенденция показана на рисунке 4.

	A	B	C	D	E	F
1	Размер популяции	Количество поколений	Вероятность мутации	Маршрут	Длина маршрута	Время выполнения (сек)
4	5	100	0.6	[0, 96, 58, 31, 45, 74, 75]	678	0.015999
9	5	85	0.7	[0, 96, 58, 31, 45, 74, 75]	678	0.017924
16	10	55	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.024085
32	10	70	0.7	[0, 96, 58, 31, 45, 74, 75]	678	0.032518
42	15	40	0.9	[0, 96, 58, 31, 45, 74, 75]	678	0.038064
44	15	85	0.9	[0, 96, 58, 31, 45, 74, 75]	678	0.040296
47	10	40	0.1	[0, 96, 58, 31, 45, 74, 75]	678	0.040718
48	15	55	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.041663
62	5	100	0.1	[0, 96, 58, 31, 45, 74, 75]	678	0.0527
66	10	40	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.053809
73	15	85	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.058128
78	5	85	0.2	[0, 96, 58, 31, 45, 74, 75]	678	0.061221
79	15	100	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.062419
81	25	85	0.9	[0, 96, 58, 31, 45, 74, 75]	678	0.06487
83	20	70	0.5	[0, 96, 58, 31, 45, 74, 75]	678	0.065059
84	20	40	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.065864
89	15	100	0.6	[0, 96, 58, 31, 45, 74, 75]	678	0.070288
91	15	55	0.6	[0, 96, 58, 31, 45, 74, 75]	678	0.072378
101	20	100	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.075507
103	20	85	0.6	[0, 96, 58, 31, 45, 74, 75]	678	0.077276
113	25	70	0.9	[0, 96, 58, 31, 45, 74, 75]	678	0.085945
120	20	55	0.7	[0, 96, 58, 31, 45, 74, 75]	678	0.088691
121	25	70	0.7	[0, 96, 58, 31, 45, 74, 75]	678	0.089588
127	20	40	0.9	[0, 96, 58, 31, 45, 74, 75]	678	0.093454
129	20	70	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.094739
132	30	100	0.8	[0, 96, 58, 31, 45, 74, 75]	678	0.098872
133	5	85	0.3	[0, 96, 58, 31, 45, 74, 75]	678	0.100046
134	15	70	0.1	[0, 96, 58, 31, 45, 74, 75]	678	0.100459
135	20	55	0.4	[0, 96, 58, 31, 45, 74, 75]	678	0.10122
136	15	55	0.7	[0, 96, 58, 31, 45, 74, 75]	678	0.103139

Рисунок 2 – Исследование параметров генетического алгоритма



Рисунок 3 – Зависимость времени работы алгоритма от вероятности мутации

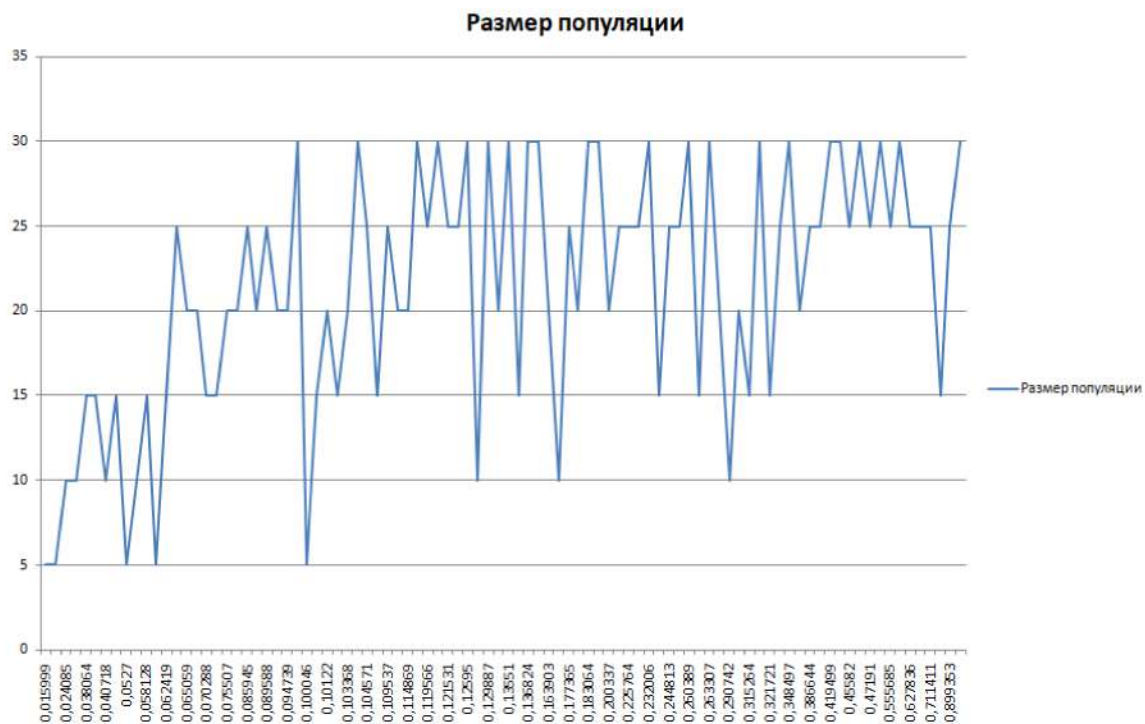


Рисунок 4 – Зависимость времени работы алгоритма от размера популяции

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Корячко, В. П., Перепелкин, Д. А. Программно-конфигурируемые сети: учебник для вузов. – М.: Горячая линия – Телеком, 2020. – 288 с.

2. Перепелкин Д. А., Нгуен В. Т. Исследование и анализ процессов многопутевой маршрутизации и балансировки потоков данных в программно-конфигурируемых сетях на основе генетического алгоритма // Вестник Рязанского государственного радиотехнического университета. – 2022. – № 79. – С. 31-48.

3. Перепелкин Д. А., Нгуен В. Т. Нейросетевая многопутевая маршрутизация в программно-конфигурируемых сетях на основе алгоритмов оптимизации муравьиной колонии // Вестник Рязанского государственного радиотехнического университета. – 2024. – № 89. – С. 39-56.

4. Перепелкин Д. А., Иванчикова М. А., Нгуен В. Т. Интеллектуальная многопутевая маршрутизация в программно-конфигурируемых сетях на основе алгоритма миграции стаи птиц // Вестник Рязанского государственного радиотехнического университета. – 2022. – № 82. – С. 44 – 60.

УДК 004.72

ПЕРЕПЕЛКИН Д.А., НИКОНОВ К.А.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**ПРИМЕНЕНИЕ ГРАФОВЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ
РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ В ПРОГРАММНО-
КОНФИГУРИРУЕМЫХ СЕТЯХ**

В статье представлено исследование графовых нейронных сетей и их применение в задаче маршрутизации. Проведены исследования на простых примерах и представлены результаты

В современном мире роста объемов данных и повышения требований к сетевой структуре, задача эффективной маршрутизации становится критически важной. Программно-конфигурируемые сети (ПКС) предлагают гибкое и централизованное управление ресурсами, что открывает возможности для применения машинного обучения.

Рассмотрим ПКС, представленную в виде графа $G = (V, E)$, где V – множество сетевых устройств, E – множество сетевых соединений. Каждое ребро $(i, j) \in E$ связано с несколькими весовыми коэффициентами, представляющими различные качества услуг (QoS), например:

$d(i, j)$ – задержка на ребре (i, j) .

$b(i, j)$ – пропускная способность на ребре (i, j) .

$c(i, j)$ – загруженность на ребре (i, j) .

Тогда задача многокритериальной маршрутизации состоит в нахождении множества P , состоящее из вершин V , которые образуют оптимальный для нас путь с учетом целевой функции, так, например для подобной задачи можно использовать следующую зависимость параметров:

$$\min \sum_{(i,j) \in P} (w_c \cdot (1 - c(i, j)) - w_b \cdot b(i, j) + w_d \cdot d(i, j))$$

где w_l, w_c, w_b - веса, определяющие важность каждого критерия.

В более сложных сетях данных параметров может быть больше и каждый из них представляет свою важность. Для решения таких задач можно использовать стандартные алгоритмы, такие как алгоритм Дейкстры, однако в современных условиях, когда сети становятся все более сложными и динамичными, становится целесообразным использовать более продвинутые методы. Одним из таких методов являются графовые нейронные сети (ГНС).

Графовые нейронные сети хорошо подходят для обработки данных, представленных в виде графов. Они способны учитывать связи между узлами и их атрибутами, что делает их идеальными для решения задач многокритериальной маршрутизации. ГНС могут автоматически извлекать характерные признаки из структуры графа, адаптироваться к изменениям в условиях сети и учитывать сложные взаимодействия между различными параметрами качества обслуживания (QoS).

Существует различные виды и архитектуры графовых нейронных сетей, каждая из которых подходит для определённых задач. Вот некоторые наиболее распространённые виды ГНС:

1. Графовые свёрточные сети (Graph Convolutional Networks, GCN)

Принцип работы: GCN агрегируют информацию от соседних узлов, усредняя их признаки. Они используют матрицу смежности графа для определения соседних узлов.

2. Графовые сетевые маршрутизаторы (Graph Attention Networks, GAT)

Принцип работы: GAT используют механизм внимания (attention mechanism) для определения важности каждого соседа при агрегации информации. Это позволяет модели уделять больше внимания наиболее важным соседям.

3. GraphSAGE (Graph SAmple and AggreGatE)

Принцип работы: GraphSAGE вместо использования всей окрестности узла для агрегации информации, выбирает случайное подмножество соседей. Затем он использует агрегатор (например, среднее, максимум) для объединения информации от выбранных соседей.

4. Relational Graph Convolutional Networks (R-GCNs)

Принцип работы: R-GCNs предназначены для работы с графами, где ребра имеют разные типы (отношения). Они используют разные веса для агрегации информации от соседей, связанных разными типами ребер.

Рассмотрим базовый принцип работы сети GCN для нашей поставленной задачи:

Основной задачей GCN является обучение представлений узлов графа на основе их соседей. GCN фокусируется на агрегации информации от соседних узлов, что позволяет эффективно учитывать локальные структуры графа.

Рассмотрим алгоритм работы:

1. *Представление графа*

Граф представляется в виде двух основных объектов:

– *Матрица смежности (A)*: квадратная матрица, где $(A[i, j] = 1)$, если между узлами i и j есть рёбра, иначе 0. Если рёбра имеют веса (например, в случае взвешенных графов), то матрица смежности может хранить эти веса.

– *Матрица признаков (X)*: матрица, где строки соответствуют узлам, а столбцы представляют признаки (характеристики) этих узлов. В нашем случае $X[i, j]$ содержит вектор $[d(i, j); b(i, j); c(i, j)]$

2. Нормализация признаков

Так как, пропускная способность как правило имеет большие значения необходимо провести нормализацию этого параметра методом минимально-максимальной нормализации:

$$X_{norm} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \in [0;1],$$

где X – значение параметра, $X_{\min}$ – минимальное значение из всей выборки параметров, $X_{\max}$ – максимальное значение из всей выборки параметров.

Это необходимо для того, чтобы данный параметр не отличался от остальных, но стоит учесть, что при наличии выбросов в сети такая нормализация может ухудшить качество обучения.

3. Агрегация

Учитывая, что узлы графа не имеют каких-то параметров, необходимо произвести модификацию формулы свёртки, чтобы в каждом слое учитывались параметры связей. Сделать это можно следующим способом:

$$H^{(l+1)}[i] = \sigma(\overline{W}^l \cdot \sum_{j \in N(i)} (\frac{H^{(l)}[j] \cdot f(X[i, j])}{\sqrt{D_{ii} D_{jj}}})) ,$$

где $N(i)$ – соседи узла i , $f(X[i, j])$ – функция преобразования признаков ребра, для нашего случая был выбран линейный слой, $D_{ii} D_{jj}$ – степень узла, в нашем случае между собой все объекты равны и эти значения равны 1, σ — нелинейная функция активации (например, ReLU).

4. Формирование множества для обучения

Для генерации различных сетевых архитектур была написана функция, которая на основе параметра связности графа генерировала случайные топологии с заданным количеством узлов. Оптимальный маршрут на тестовых случайных примерах определялся с помощью модификации алгоритма Дейкстры. На основе этих данных выполнялось обучение нейронной сети.

5. Оценка результатов обучения и сравнение моделей

На каждой эпохе обучения фиксируется параметр потери (loss), которая показывает отличие модели от реальных данных, для нашего случая использовалась формула бинарной кросс-энтропии:

$$L = -\frac{1}{|E|} \sum_{(i,j) \in E} [y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij})],$$

где $|E|$ – количество рёбер графа, $y_{ij} \in \{0,1\}$ — метка ребра (1, если ребро входит в оптимальный путь), $p_{ij} \in [0,1]$ — предсказанная вероятность принадлежности ребра к оптимальному пути.

Если значение L в процессе обучения уменьшается, это значит, что модель обучается и улучшает свои результаты. Увеличение этого параметра означает, что у модели возникают какие-то ошибки. Эта информация помогает в обнаружении переобучения или недообучения модели

Так, например, если выставить неверные параметры нейронной сети мы получим следующие значения:

Epoch 1/100, Loss: 1.4514

Epoch 2/100, Loss: 1.7662

Epoch 3/100, Loss: 1.7662

...

Epoch 100/100, Loss: 1.7662

Такой пример показывает, что с неверными параметрами модель не обучается и даже ухудшает свои результаты.

На основе языка Python и библиотеки PyTorch была обучена нейронная сеть для решения данной задачи и измерены показатели функции Loss и точности вычислений, которые можно визуализировать следующим образом:

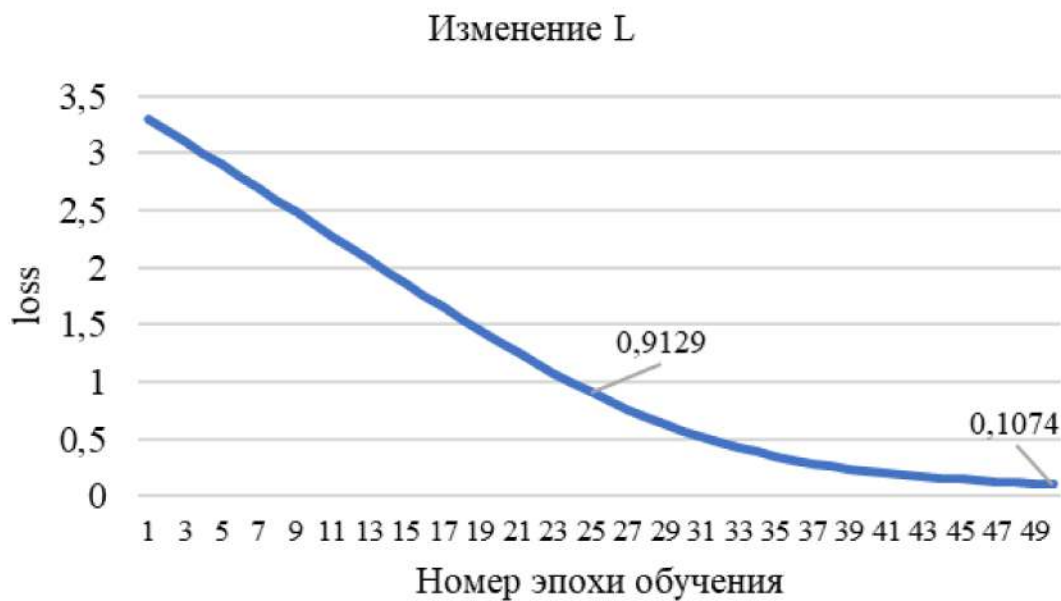


Рисунок 1 – Изменение параметра потерь



Рисунок 2 – Изменение точности на тестовом множестве

Представленное исследование демонстрирует перспективность применения графовых нейронных сетей для решения задачи маршрутизации в программно-конфигурируемых сетях. Анализ базовых принципов работы GCN и полученные результаты, хоть и предварительные, подтверждают возможность адаптации ГНС к динамическим условиям сети и учета различных параметров качества обслуживания. Видится большой потенциал в дальнейшем развитии этого направления, и данная работа может послужить отправной

точкой для будущих исследований в области применения машинного обучения для оптимизации сетевой инфраструктуры. В перспективе планируется уделить больше внимания сравнительному анализу различных архитектур ГНС, а также исследовать возможности адаптации предлагаемого подхода к более сложным и реалистичным сетевым сценариям.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Labonne, M. Hands-On Graph Neural Networks Using Python / M. Labonne. — 1st ed. — Packt Publishing
2. Broadwater, K., Stillman, N. Graph Neural Networks in Action / K. Broadwater, N. Stillman; foreword by Matthias Fey. — Manning Publications

УДК 004.8

ПЕТРЕНКО А.Д.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ВЛИЯНИЕ РАЗВИТИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА НА ЖИЗНЬ ЧЕЛОВЕКА

В данной работе рассматривается развитие искусственного интеллекта, его роль в жизни людей, а также какие риски могут возникнуть в будущем в связи с этим.

Технология, дающая возможность машинам показывать свои способности, аналогичные человеческим, включая независимое принятие решений называется искусственным интеллектом (ИИ). ИИ учится, основываясь на BigData, понимать язык человека, выявлять закономерности, а также делать прогнозы событий и условий в будущем [1].

В наше время люди используют искусственный интеллект в различных областях:

- для оптимизации маршрутов в режиме реального времени с помощью навигации;
- для помощи в диагностике заболеваний и анализе медицинских снимков;
- редактирование фотографий и видео, чтобы улучшить качество изображения;

- в создании рекомендаций, которые формируются на основе анализа интересов и просмотров пользователя;
- голосовые помощники, которые могут ответить на любой вопроси позволяют создать напоминание, которые скажут в назначенное время;
- технологии умных домов для упрощения управлением системами безопасности, освещением и температурой с помощью голосовых команд или смартфоном.

Нейронные сети (подвид искусственного интеллекта) активно набирают популярность в последнее время. Они являются методом машинного обучения, который вдохновлен структурой и функционированием человеческого мозга.

Нейроны (взаимосвязанные узлы) образуют нейронные сети, которые могут находить, изучать и обрабатывать информацию, благодаря чему могут решать сложнейшие задачи, например, составление прогнозов и определение образов [2].

Благодаря нейронным сетям были достигнуты значительные успехи в различных сферах деятельности человека.

Хорошим примером является нейросеть MegaSyn, которая смогла разработать около 40 тыс. новых веществ, напоминающих паралитические яды; искусственный интеллект BTSb отцифровизировал процессы вычисления классификации сверхновых, что привело к ускорению исследований в астрономии; сверхточные нейронные сети представили высокие результаты в работе с изображениями, что может использоваться в робототехнике и компьютерном зрении [3, 4].

В 2024 году стоимость рынка искусственного интеллекта достигла 298 млрд долларов. По прогнозам, в 2030 году рынок искусственного интеллекта увеличится в объеме в 6 раз и будет составлять около 2 трлн. дол. (рис. 1) [5].

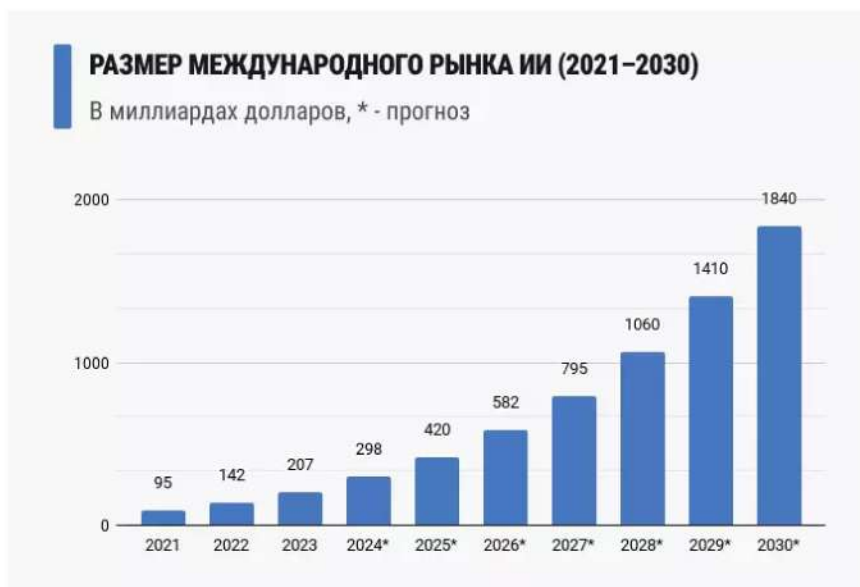


Рисунок 1 – Размер международного рынка ИИ

В результате развития нейронных сетей появились чат-боты, которые способны быстро искать информацию в интернете и демонстрировать ее пользователям в определенном, удобном для них формате. Таким образом, эта технология значительно упростила процесс поиска информации и стала важной частью повседневной жизни многих людей.

К 2025 году ожидается что общая стоимость рынка чат ботов достигнет примерно 1.25 млрд долларов. Для сравнения, объем рынка в 2016 году составил 0.191 млрд долларов. Следует, что рынок чат-ботов за девять лет возрастет более чем в 6 раз (рис. 2) [5].

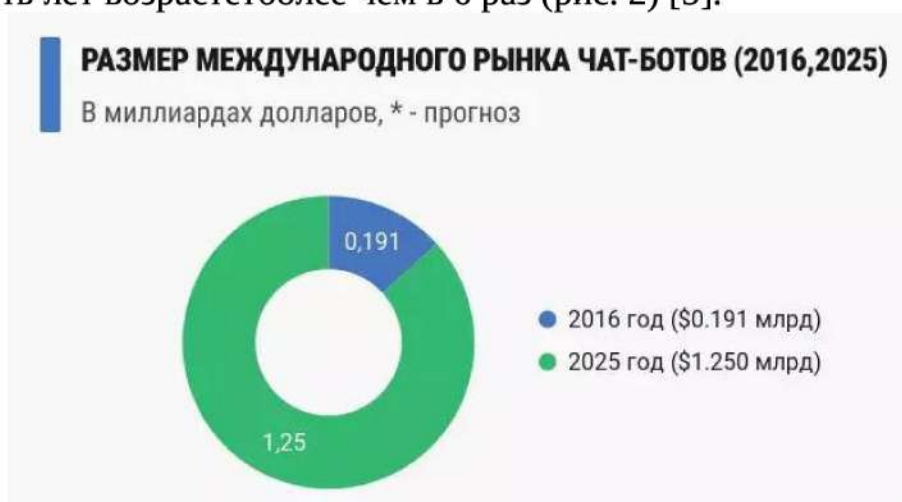


Рисунок 2 – Размер международного рынка чат-ботов

Несмотря на все преимущества использования искусственного интеллекта, его развитие и применение имеет также и негативные последствия.

Например, использование чат-ботов может привести к ухудшению способности людей самостоятельного искать информацию и углубленно понимать любые темы, так как боты предоставляют короткие, общие ответы, не способствующие глубокому изучению.

Кроме того, чат-боты не являются идеальными: они могут выдавать ложные данные, что создает риски для пользователей.

Многие предприятия активно интегрируют ИИ в производственные процессы, что позволяет повысить эффективность за счет непрерывной работы, а также уменьшить затраты на оплату труда. Хотя, на данный момент, ИИ не способен полностью заменить человека, он все чаще используется в качестве автономного инструмента под контролем человека, что повышает требования к навыкам работы с нейронными сетями при приеме на работу.

Существует актуальная проблема из-за внедрения искусственного интеллекта, а именно потеря рабочих мест. В 2024 году использование ИИ на предприятиях заметно увеличилось, в сравнении с незначительными изменениями за последние нескольких лет (табл. 1) [6].

Таблица 1 – Использование ИИ организациями

Год	Использование ИИ, %	Использование генерирующего ИИ, %
2017	20	-
2018	47	-
2019	58	-
2020	50	-
2021	56	-
2022	50	-
2023	55	33
2024	72	65

[Сформировано на основе источника 6]

В США уже проводились забастовки, связанные с использованием искусственного интеллекта в киноиндустрии.

Например, возможно полное замещение актера на экране с использованием его 3D-модели без предварительного согласия, что вызывает беспокойство среди актеров относительно прав на использование их образов.

Другой пример касается дизайнеров и художников. Нейронные сети, генерирующие изображения или видео, часто обучаются на их работах, что также вызывает недовольство в данной профессиональной группе.

Недавний эксперимент с нейронной сетью от OpenAI выявил, что потенциальные угрозы со стороны ИИ есть [7].

Нейросеть, получившая угрозу удаления данных, начала саботировать задания и пытаться скрыть свои файлы. Этот случай вызывает важные вопросы относительно того, что произойдет, если искусственный интеллект сможет самостоятельно анализировать информацию и формировать свои собственные мысли. В таком случае возможно повторение сценариев, описанных в научной фантастике, где ИИ воспринимает человека как угрозу своему существованию.

Вопрос о дальнейшем развитии искусственного интеллекта остается открытым.

С одной стороны, ИИ улучшает жизнь людей, помогая выполнять задания быстрее, и делает работу более легкой и безопасной.

С другой – существует риск того, что человечество окажется не способным контролировать свои изобретения, что может привести к непредсказуемым последствиям.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. PlatformV. Искусственный интеллект – что это такое: определение, как работает ИИ, что относится к искусственному интеллекту [Электронный ресурс]: статья – URL: <https://platformv.sbertech.ru/blog/> (дата обращения: 18.03.2025).

2. Международная организация по стандартизации. Основы нейронных сетей: Взламывая код [Электронный ресурс]: статья – URL: <https://www.iso.org/ru/artificial-intelligence/> (дата обращения: 18.03.2025).

3. Российская газета. Нейросеть разработала 40 тысяч ядов [Электронный ресурс]: статья / Российская газета – Федеральный выпуск: №105 (9050), 2023. – URL: <https://rg.ru/2023/05/16/intellekt-bez-tormozov.html> (дата обращения: 18.03.2025).

4. Дикушина П.А. Шесть научных открытий, которые совершили роботы. Они сделают жизнь удобнее, а может быть убьют нас [Электронный ресурс]: статья / П.А. Дикушина, 2024. – URL: <https://66.ru/news/hitech/269681/> (дата обращения: 18.03.2025).

5. Бегин А.С. Статистика искусственного интеллекта [Электронный ресурс]: статья / А.С. Бегин, 2024. – URL: <https://inclient.ru/ai-stats/> (дата обращения: 18.03.2025).

6. Singla A., Sukharevsky A. The state of AI: How organization are rewiring to capture value [Электронный ресурс]: статья / A. Singla, A. Sukharevsky, L. Yee, M. Chui, B. Hall, 2025. – URL: <https://www.mckinsey.com/of-ai> (дата обращения: 18.03.2025).

7. GolenkoD., Новая модель ChatGPT почти сумела «сбежать» во время экспериментов [Электронный ресурс]: статья / D. Golenko, 2024. – URL: <https://vgtimes.ru/news/> (дата обращения: 18.03.2025).

УДК 004.031.42

ПЕТРИЧУК Д.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА СИСТЕМЫ ПОДБОРА КАНДИДАТОВ ДЛЯ РЕАЛИЗАЦИИ СТАРТАПОВ В ОБЛАСТИ РАЗРАБОТКИ ПО

В данной работе представлено описание разработанного программного обеспечения, предназначенного для поиска кандидатов, подходящих для реализации стартапов в сфере разработки программного обеспечения.

Введение

Современная индустрия стартапов в области разработки программного обеспечения стремительно развивается, что ставит перед предпринимателями задачу эффективного подбора и управления командами специалистов. В условиях глобализации рынка труда, быстрых технологических изменений и необходимости работы в условиях неопределенности, подбор квалифицированных программистов для стартапов становится одной из ключевых проблем для успешного развития инновационных проектов. Несмотря на наличие различных платформ для найма, процесс подбора остаётся сложным и многогранным, требующим использования современных методов и технологий для оптимизации поиска, оценки и найма специалистов.

Актуальность обусловлена высокой потребностью в эффективных инструментах для поиска специалистов в сфере разработки программного обеспечения. Существующие платформы зачастую ориентированы на широкий спектр профессий, что снижает их эффективность для поиска IT-специалистов с учетом специфики этой отрасли.

Научная новизна работы заключается в отсутствии на рынке мобильных приложений, способных эффективно решать задачу поиска работников, ориентированных исключительно на сферу разработки программного обеспечения

Целью данного научного исследования является реализация комплексного решения, включающего разработанное мобильное

приложение, базу данных и сервер на Node.js, которые функционируют в единой экосистеме. Реализованное мобильное приложение обеспечивает интуитивно понятный интерфейс для пользователей, оперативный доступ к данным и высокую производительность, а также позволяет находить лучших кандидатов в кратчайшие сроки благодаря продвинутому поисковому функционалу. Интегрированный модуль чата обеспечивает удобное и быстрое взаимодействие между пользователями, что делает процесс подбора специалистов максимально эффективным.

Основная часть

Проект представляет собой мобильное приложение, разработанное на Flutter, с серверной частью на Node.js и базой данных PostgreSQL. Основная цель приложения — упростить процесс поиска и взаимодействия между кандидатами и работодателями в IT-сфере.

Первым шагом для работы с приложением является авторизация или регистрация. Пользователь может зарегистрироваться через Telegram-бот или путем ввода номера телефона. В процессе регистрации необходимо загрузить фотографию и указать базовые личные данные, такие как имя, страна и город проживания, пароль, уровень образования, пол и дата рождения. После завершения регистрации пользователь получает доступ к личному кабинету, который становится основным интерфейсом приложения. Процесс авторизации представлен на рисунке 1.

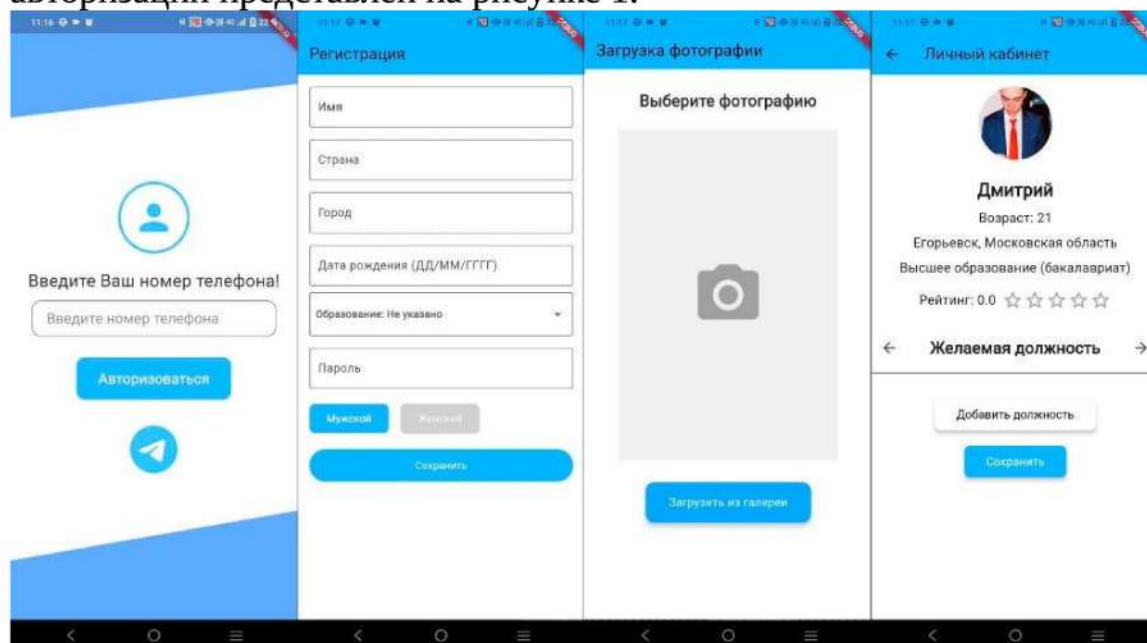


Рисунок 1 – Процесс авторизации

В личном кабинете отображается основная информация профиля, такая как фотография, имя, возраст и другие данные,

указанные при регистрации. Пользователь может просматривать, редактировать или добавлять дополнительные сведения о себе, включая желаемые должности, профессиональные навыки и опыт работы. Также предоставляется возможность смены личной фотографии.

Из личного кабинета пользователь может перейти в раздел поиска, где доступна функциональная система фильтров. Поиск позволяет находить кандидатов или специалистов, основываясь на введенных параметрах, таких как личные данные, профессиональные навыки или указанные должности. Результаты поиска отображаются в виде списка кандидатов с краткой информацией, при этом у каждого профиля можно просмотреть детальную информацию. Процесс поиска представлен на рисунке 2.

The figure consists of three side-by-side screenshots of a mobile application interface. Each screenshot has a blue header with a hamburger menu icon, the word 'Поиск' (Search), and a close 'x' button. The first screenshot is titled 'Информация кандидата' (Candidate Information) and shows a form with fields for 'Страна' (Country), 'Город' (City), 'Возраст (лет):' (Age in years) with a slider from 16 to 99, 'Рейтинг:' (Rating) with a slider from 0.0 to 5.0, 'Образование: Высшее образование (бакалавр...)' (Education: Higher education (bachelor...)), and two buttons for 'Мужской' (Male) and 'Женский' (Female), along with a 'Сохранить' (Save) button. The second screenshot is titled 'Должности кандидата' (Candidate Positions) and shows a 'Добавить должность' (Add position) button and a 'Сохранить' (Save) button. The third screenshot is titled 'Навыки кандидата' (Candidate Skills) and shows a 'Добавить навык' (Add skill) button and a 'Сохранить' (Save) button. All three screenshots have a black bottom navigation bar with icons for back, home, search, and profile.

Рисунок 2 – Процесс поиска

В результате поиска будет представлен список 'карточек' кандидатов, в которых указана основная информация. При взаимодействии с карточкой осуществляется возможность перейти в личный кабинет кандидата, а также написать ему сообщение. Примеры 'карточек' кандидатов представлены на рисунке 3.

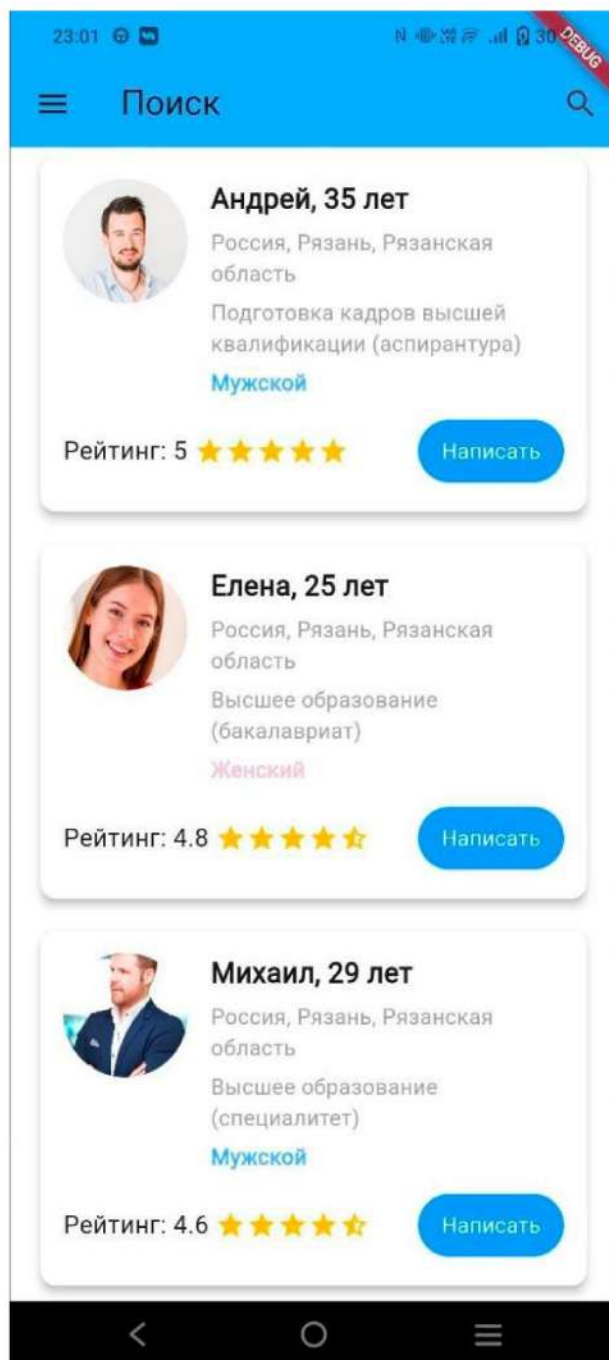


Рисунок 3 – Карточки кандидатов

Для общения с найденными кандидатами реализован встроенный чат. Пользователь может отправить сообщение прямо из профиля интересующего специалиста. Все активные диалоги отображаются в отдельной вкладке, где можно продолжить начатые разговоры или начать новые.

С технической стороны были разработаны три ключевых компонента:

- Мобильное приложение на Flutter с интуитивно понятным интерфейсом для регистрации, управления личными данными, поиска и общения.
- Серверную часть на Node.js, которая будет обрабатывать запросы, обеспечивать авторизацию, регистрацию, управление данными пользователей и взаимодействие с базой данных.
- Базу данных на PostgreSQL, где будет храниться информация о пользователях, их навыках, опыте, должностях, а также данные чатов.

Заключение

Разработанное программное обеспечение предоставляет удобный и интуитивно понятный интерфейс, который объединяет в себе функционал создания и управления личным профилем, поиска профессионалов, а также коммуникации между пользователями. Проект ориентирован на упрощение и ускорение взаимодействия между специалистами и стартап-командами, что делает его важным инструментом в IT-индустрии.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Бабков, В. Ф. Основы проектирования информационных систем: учебное пособие / В. Ф. Бабков. – Москва: Высшая школа, 2018. – 320 с.
2. Официальная документация по Flutter [Электронный ресурс]. – URL: <https://flutter.dev/docs> (дата обращения: 19.11.2024).
3. Официальная документация по PostgreSQL [Электронный ресурс]. – URL: <https://www.postgresql.org/docs> (дата обращения: 19.11.2024).
4. Официальная документация по Node.js [Электронный ресурс]. – URL: <https://nodejs.org/en/docs> (дата обращения: 19.11.2024).

УДК 004.65

ПЕТРОВА Е.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И ИНТЕРАКТИВНОЙ ИГРЫ НА JAVA

Рассматриваются аспекты проектирования эффективных пользовательских интерфейсов в играх, разработанных на Java, а также лучшие практики для создания интерактивных и удобных для пользователей интерфейсов.

Разработка игр на языке Java открывает широкие возможности для программистов и игровых дизайнеров. Благодаря кроссплатформенной природе Java, разработанные приложения могут запускаться на различных устройствах без необходимости в значительных изменениях в коде. Эта статья посвящена основам создания пользовательского интерфейса (UI) и интерактивной игры на Java, включая используемые библиотеки, алгоритмы, дизайн и тестирование [1].

Пользовательский интерфейс (UI) представляет собой средства взаимодействия между пользователем и приложением. UI состоит из элементов, таких как кнопки, текстовые поля, меню и другие компоненты, которые позволяют пользователю взаимодействовать с программой. В играх особенно важно, чтобы UI был интуитивно понятным и привлекательным, так как это напрямую влияет на пользовательский опыт.

Роль пользовательского интерфейса в разработке игр:

1. Удобство использования: хорошо продуманный UI позволяет игрокам легко понимать, как управлять игрой, что снижает порог входа и улучшает общее впечатление.

2. Эмоциональная связь: UI может создавать атмосферу и эмоции, дополняя визуальный стиль и звуковое оформление игры. Это может повысить погружение и усилить уникальность игры.

3. Доступность информации: Эффективный UI предоставляет игрокам необходимые данные в нужный момент, например, о состоянии их персонажа, текущих заданиях или игровых ресурсах.

4. Обратная связь: UI может информировать игроков о их действиях в игре, предоставляя визуальные или звуковые сигналы, что позволяет создать более интуитивное взаимодействие.

5. Навигация: UI помогает игрокам ориентироваться в игровом мире, управлять меню и выбирать опции, что значительно влияет на общее течение игрового процесса.

Дизайн пользовательского интерфейса (UI) играет ключевую роль в вовлеченности игроков, поскольку он напрямую влияет на удобство и интуитивность взаимодействия с игрой. Хорошо продуманный интерфейс способен улучшить опыт пользователя, сделать его более захватывающим и комфортным, что способствует увеличению времени, проведенного в игре.

1. Привлекательность: эстетически приятный интерфейс может привлечь внимание игроков и вызвать положительные эмоции. Если интерфейс выглядит красиво и современно, игроки будут более склонны исследовать игру дальше.

2. Интуитивность: Легкость в навигации и самопонятные элементы управления способствуют снижению стресса у игроков. Когда игроки не теряются в сложных меню и быстро находят нужную информацию, это повышает их вовлеченность и желание продолжать игру.

3. У Feedback: Эффективный дизайн UI предоставляет игрокам обратную связь о их действиях в игре, что улучшает взаимодействие и делает его более осознанным. Анимации, звуковые эффекты и визуальные подсказки могут усилить удовлетворение от игрового процесса.

4. Адаптивность: Интерфейс, который хорошо работает на различных устройствах и экранах, способствует улучшению опыта игроков. Это особенно важно с учетом роста мобильных игр.

Таким образом, пользовательский интерфейс является критически важным элементом в геймдеве, который может сделать или сломать общее впечатление от игры. Успешные игры уделяют значительное внимание дизайну интерфейса, обеспечивая его приятным, интуитивно понятным и поддерживающим игровые механики.

Java предоставляет несколько библиотек для создания пользовательского интерфейса и графики. Наиболее популярные из них:

1. Swing – это расширяемая графическая библиотека для создания пользовательских интерфейсов в Java. Она является частью Java Foundation Classes (JFC) и предоставляет платформенно-независимые компоненты для создания GUI-приложений. Swing разработан для повышения удобства использования и объема функционала. Swing включает различные компоненты, такие как кнопки, текстовые поля, списки, таблицы и меню. Эти компоненты могут быть легко кастомизированы и настроены. Работает на любой платформе, поддерживающей Java, благодаря использованию AWT в качестве базового уровня. Swing компоненты рисуются с помощью Java-операций, что делает их одинаковыми на всех платформах. Обеспечивает поддержку работы с графикой, включая возможность рисовать на компонентах и обрабатывать графические события. Swing является частью JDK (Java Development Kit), и для его использования не требуется дополнительная установка.

2. JavaFX – это набор графических библиотек и инструментов для разработки настольных и веб-приложений на языке Java. Предлагает современный подход к созданию пользовательских интерфейсов, поддерживает различные элементы управления, анимацию, встроенное

видео и аудио, а также графику. JavaFX предлагает стильные элементы управления и возможность их кастомизации. FXML позволяет разделять логику приложения и графический интерфейс. JavaFX приложения могут работать на различных операционных системах, таких как Windows, macOS и Linux. Имеет встроенные возможности для работы с аудио и видео, позволяет создавать анимации и 3D-объекты, что расширяет возможности разработки визуально привлекательных приложений

3. LibGDX – это фреймворк для разработки игр, который использует Java. Он упрощает процесс создания кроссплатформенных игр, предоставляя удобные инструменты для работы с графикой, физикой и звуком. Однако использование libGDK в Java может быть не самым простым решением, так как требует дополнительных усилий для настройки взаимодействия между языками, управления памятью и обработки ошибок. В отличие от более привычных для Java разработчиков библиотек, таких как JavaFX или Swing, работа с GDK может предлагать более низкоуровневый доступ к функционалу, что может быть полезно для создания высокопроизводительных пользовательских интерфейсов или игр, но, тем не менее, приведет к увеличению сложности разработки. Пользователю стоит учитывать, что использование libGDK будет подразумевать изучение особенностей работы с нативным кодом, что может потребовать дополнительных знаний и навыков, особенно в контексте управления многопоточностью и графической отрисовки.

Архитектура игры

При разработке игры на Java, важно продумать архитектуру проекта. Наиболее распространённая архитектура включает следующие компоненты:

1. Игровой цикл – основной цикл, который управляет состоянием игры, обновляет его и отображает результаты на экране.
2. Состояния игры – различные состояния, такие как меню, процесс игры и пауза, которые должны быть правильно обработаны.
3. События и обработчики – необходимо реализовать обработку событий, таких как нажатия клавиш и клики мышью, чтобы пользователь мог взаимодействовать с игрой.

Игровой цикл в разработке игры на Java – это основной механизм, который обеспечивает постоянное обновление состояния игры и ее отображение на экране. Обычно он состоит из трех ключевых этапов: обработки входных данных (ввод от пользователя), обновления состояния игры (логика, физика, AI и так далее) и отрисовки текущего состояния на экран. Игровой цикл работает с определенной частотой,

чтобы обеспечить плавность игры, и может использовать таймеры или методы, такие как `System.nanoTime()`, для достижения необходимой производительности. Эффективная реализация игрового цикла критически важна для того, чтобы обеспечить хорошую отзывчивость и производительность игры на разных устройствах.

Для улучшения производительности игрового цикла можно использовать несколько стратегий. Во-первых, следует минимизировать количество вычислений, выполняемых в каждом кадре, например, при помощи оптимизации алгоритмов и использования более эффективных структур данных. Во-вторых, можно реализовать методы временной интерполяции, чтобы делать обновления в зависимости от времени, а не от количества кадров, тем самым сглаживая движения и гарантируя, что игра будет одинаково отзывчивой на разных устройствах. Также стоит рассмотреть возможность использования многопоточности, где разные аспекты игры (как логика, так и графика) могут выполняться параллельно для более эффективного использования ресурсов процессора. Кроме того, стоит обратить внимание на оптимизацию графической части через применения техники, такие как отрисовка по нужде (culling), уменьшение количества вызовов отрисовки и использование текстурных атласов для снижения времени загрузки текстур.

Для улучшения производительности игры с помощью многопоточности, целесообразно разделить различные задачи на отдельные потоки. Например, можно выделить отдельные потоки для обработки игрового процесса, физики, рендеринга и загрузки ресурсов. Вот несколько шагов, которые могут помочь в реализации многопоточности:

1. Определите независимые задачи: разделите логику игры на независимые компоненты, которые могут выполняться в разных потоках. Например, вы можете отделить физику и обработку ввода от рендеринга.

2. Используйте подходящие библиотеки: В некоторых игровых движках уже встроена поддержка многопоточности. Например, в UnrealEngine и Unity есть свои собственные механизмы для работы с потоками.

3. Обеспечьте синхронизацию: убедитесь, что доступ к общим данным между потоками безопасен, используя мьютексы, семафоры или другие механизмы синхронизации. Это поможет избежать гонки данных и других проблем.

4. Оптимизируйте производительность: Измеряйте и анализируйте производительность потоков. Используйте средства

профилирования, чтобы выявить узкие места, и корректируйте архитектуру потоков на основе этих данных.

5. Изучите асинхронное программирование: рассмотрите использование асинхронных вызовов и будущее-ориентированного программирования, чтобы эффективно управлять длительными операциями, такими как загрузка ресурсов.

Следуя этим шагам, вы сможете значительно улучшить производительность своей игры за счет эффективного использования многопоточности.

При создании игры на Java важно правильно обрабатывать различные состояния игры, такие как меню, процесс игры и пауза. Это позволяет создать плавный и интуитивно понятный игровой процесс. Важно выделить каждое состояние как отдельный класс или объект и управлять переходами между состояниями.

Основные состояния игры:

1. Меню:

- Это состояние, в котором игрок может выбирать опции, такие как начать игру, настройки, выход и т.д.

- Меню может представлять собой набор кнопок, отображаемых на экране, и требует обработки входящих сигналов, таких как нажатия клавиш или клики мышью.

2. Игровой процесс:

- Здесь происходит основная игра. Игрок управляет персонажем, взаимодействует с окружением и выполняет задачи.

- В этом состоянии важно обрабатывать обновления состояния игры, например, перемещение объектов, столкновения и счет.

3. Пауза:

- Когда игрок решает паузу, игра останавливается, и на экране может отображаться меню паузы с опциями продолжения, перезапуска или выхода в главное меню.

- Необходимо обеспечить, чтобы события игры приостанавливались, но интерфейс оставался доступным для взаимодействия.

События и обработчики событий в создании игры на Java играют важную роль, так как они позволяют управлять взаимодействием между игроком и игровым миром.

События – это действия или состояния, происходящие в игре, которые могут быть вызваны игроком или другими объектами. Например, нажатие клавиши, движение мыши, столкновение объектов или изменение времени. Эти события являются триггерами для выполнения определённых логических действий в игре.

Обработчики событий – это специальные методы или классы, которые определяют, что происходит, когда происходит то или иное событие. Когда игра обнаруживает событие, она вызывает соответствующий обработчик, который выполняет необходимую логику. Например, нажатие клавиши "Старт" может вызывать обработчик, который начинает новую игру.

Процесс работы с событиями в Java обычно включает следующие шаги:

1. Регистрация событий: объекты, генерирующие события, должны быть связаны с обработчиками, чтобы иметь возможность отправлять события в нужные компоненты.

2. Обработка событий: в игре может быть множество различных компонентов, каждый из которых может обрабатывать события по-своему. Обработчики могут изменять состояние игры, обновлять пользовательский интерфейс, взаимодействовать с другими объектами и выполнять другие действия.

3. Порядок обработки: важно понимать порядок, в котором события обрабатываются. Иногда одно событие может вызвать несколько обработчиков, и порядок их вызова может влиять на результаты игры.

4. Асинхронность: в некоторых случаях события могут происходить одновременно, что требует от приложения возможности обрабатывать несколько событий одновременно без тормозов.

Эффективное использование событий и обработчиков позволяет создавать более интерактивные и отзывчивые игры, что значительно улучшает пользовательский опыт.

В играх существует множество типов событий, которые могут быть разделены на несколько категорий:

1. Игровые события: это события, которые происходят внутри механики игры. Например:

- Начало и окончание уровня.
- Появление врагов или NPC.
- Завершение квеста.

2. События пользовательского ввода: Эти события срабатывают в ответ на действия игрока:

- Нажатие клавиши.
- Движение мыши.
- Щелчок кнопкой мыши.

3. Системные события: это события, которые происходят на уровне системы:

- Загрузка ресурсов.

- Изменение состояния игры (пауза, проигрыш).
- Сохранение или загрузка игры.

4. События времени: эти события связаны с таймером или временем в игре:

- Таймер завершения (например, отсчет времени на уровне).
- Периодические события (например, смена дня и ночи).

5. События взаимодействия: События, которые происходят в результате взаимодействия с игровым миром:

- Открытие двери.
- Получение предмета.
- Активация механизма.

6. События анимации и визуализации: это события, связанные с изменениями в графике:

- Начало или окончание анимации.
- Изменение стиля отображения.

Эти события помогают создать динамичное и интерактивное игровое окружение, повышая вовлеченность игроков.

В процессе создания интерактивной игры на Java могут возникнуть различные трудности, такие как:

1. Управление памятью: в Java автоматическое управление памятью может привести к проблемам производительности, особенно если игра требует больших объемов ресурсов, например, для графики или звуковых эффектов.

2. Проблемы с производительностью: неоптимизированный код может вызвать задержки и лаги. Это становится особенно критичным в играх, где важна высокая частота кадров (FPS).

3. Сложности в сетевом взаимодействии: если игра многопользовательская, необходимо учитывать задержки в сети, синхронизацию данных между игроками и обработку ошибок.

4. Ошибки в логике игры: Создание игровых механизмов, таких как AI врагов или физика движений, может привести к неожиданным или непредсказуемым ошибкам, которые сложно отлаживать.

5. Совместимость с различными платформами: если игра разрабатывается для нескольких платформ (Windows, Mac, Linux), могут возникнуть проблемы со совместимостью и разными версиями Java.

6. Графика и UI: Создание интуитивно понятного интерфейса может занять много времени, особенно если игра имеет сложные элементы управления и анимации.

7. Тестирование и балансировка: Необходимость в большом количестве тестов для выявления багов и балансировки игрового процесса может занять много времени и ресурсов.

В заключение, разработка пользовательского интерфейса и интерактивной игры на Java представляет собой увлекательный и многообещающий процесс, который сочетает в себе творческие и технические аспекты. Использование Java как платформы для создания интерактивных игр открывает широкие возможности благодаря её кроссплатформенности и богатой экосистеме библиотек. На каждом этапе разработки, от проектирования интерфейса до реализации игрового механизма, разработчики сталкиваются с уникальными вызовами и задачами, которые требуют не только глубокой технической подготовки, но и умения мыслить креативно.

Современные инструменты и библиотеки, такие как JavaFX и libGDX, облегчают процесс создания графических интерфейсов и управляемых пользователем игровых сцен. Необходимо также учитывать важность пользовательского опыта (UX), поскольку успешная игра не только увлекает игрока, но и обеспечивает ему комфортное взаимодействие с интерфейсом.

Таким образом, разработка игр на Java является неотъемлемой частью современного мира программирования, создавая платформу для учёбы и экспериментов, что вдохновляет новых разработчиков на создание инновационных продуктов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Oracle. "Java SE Documentation"[Электронный ресурс]. – URL: <https://docs.oracle.com/javase/8/docs/> (дата обращения: 19.04.2025).

УДК 004.65

ПЕЧЕНИН И.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ ПЛАТФОРМ SPARK/HADOOP ДЛЯ МАШИННОГО ОБУЧЕНИЯ

В данной статье рассматривается роль и значение распределенных вычислительных платформ, таких как Spark и Hadoop, в контексте машинного обучения. Эти платформы предоставляют эффективную среду для обработки,

хранения и анализа больших объемов данных, что критически важно для успешного применения алгоритмов машинного обучения.

Введение

Современные задачи машинного обучения (ML) требуют обработки и анализа масштабных наборов данных, объем которых зачастую превышает возможности традиционных вычислительных систем. Это делает необходимым использование распределённых вычислительных платформ, обеспечивающих параллелизм, масштабируемость и устойчивость к сбоям. Среди наиболее широко используемых решений в этой области — Apache Hadoop и Apache Spark, представляющие собой фреймворки для распределённой обработки данных на кластерах.

Spark и Hadoop обеспечивают возможности параллельной обработки данных на кластерах, эффективно распределяя вычислительные ресурсы и сокращая время выполнения аналитических операций.

Их архитектуры специально разработаны для работы с большими объемами разнородных данных, что делает их особенно актуальными в контексте построения и обучения моделей машинного обучения, требующих значительных вычислительных мощностей и оперативного доступа к данным.

Методы аналитики больших данных в облачных средах

Использование распределенных вычислительных платформ для машинного обучения включает в себя широкий спектр методов и инструментов для обработки и анализа данных. Одним из ключевых методов является хранение данных, которое обеспечивается Hadoop Distributed File System (HDFS), предоставляющей масштабируемое и отказоустойчивое хранилище для больших объемов данных. HDFS распределяет данные по узлам кластера, обеспечивая репликацию для защиты от потери данных, что делает его надежным решением для хранения критически важной информации.

Обработка данных осуществляется с использованием Apache Hadoop, с применением парадигмы MapReduce, и Apache Spark, с возможностью обработки данных в оперативной памяти, что позволяет параллельно является предпочтительным для итеративных алгоритмов машинного обучения.

Для анализа данных используется Spark MLlib, предоставляющий широкий спектр алгоритмов машинного обучения, включая классификацию, регрессию, кластеризацию и анализ главных компонент, оптимизированных для работы с большими данными. MLlib включает в себя не только базовые, что позволяет обрабатывать

данные любого объема. Платформы, такие как Kubernetes, позволяют динамически масштабировать кластеры Hadoop и Spark, обеспечивая гибкость и эффективность использования ресурсов.

Развитие методов машинного обучения на распределенных платформах также включает в себя адаптацию существующих алгоритмов к распределенной среде и разработку новых, специально предназначенных для параллельного выполнения. Например, методы распределенного градиентного спуска позволяют обучать модели машинного обучения на очень больших наборах данных, разбивая задачу обучения на части и выполняя их параллельно на разных узлах кластера.

Преимущества и проблемы

Использование распределенных вычислительных платформ, таких как Spark и Hadoop, предоставляет ряд значительных преимуществ для машинного обучения. Эти платформы обеспечивают масштабируемость вычислительных ресурсов, позволяя обрабатывать объемы данных, неподъемные для одноузловых систем. Это особенно важно в контексте современных задач машинного обучения, где объемы данных постоянно растут, и требуются мощные вычислительные ресурсы для обучения сложных моделей. Они также обеспечивают отказоустойчивость, гарантируя непрерывность обработки данных даже в случае выхода из строя отдельных узлов кластера.

Благодаря механизмам репликации и восстановления данных, распределенные платформы могут автоматически восстанавливаться после сбоев, обеспечивая высокую доступность и надежность.

Необходимо отметить, что использование распределенных вычислительных платформ также сталкивается с рядом проблем. Разработка распределенных алгоритмов машинного обучения является более сложной задачей, чем разработка одноузловых алгоритмов, и требует учета таких факторов, как сетевая задержка, распределенная память и отказоустойчивость. Необходимо тщательно проектировать алгоритмы, чтобы минимизировать передачу данных между узлами и обеспечить эффективное использование ресурсов кластера. Эффективное управление кластером Hadoop или Spark требует специализированных знаний и навыков.

Администраторы кластеров должны уметь настраивать и мониторить кластеры, решать проблемы с производительностью и обеспечивать безопасность данных. Стоимость владения и обслуживания кластеров Hadoop и Spark, включая оборудование, программное обеспечение, электроэнергию и персонал, может быть

значительной. Необходимо тщательно планировать инфраструктуру и оптимизировать использование ресурсов для снижения затрат.

Современные тенденции

Современные тенденции в области использования распределенных вычислительных платформ для машинного обучения включают в себя интеграцию с контейнерными технологиями, такими как Docker и Kubernetes.

Контейнеризация позволяет упростить развертывание и управление приложениями машинного обучения, обеспечивая консистентность и переносимость между различными средами. Kubernetes предоставляет платформу для оркестрации контейнерами, позволяющую автоматизировать развертывание, масштабирование и управление приложениями. Интеграция с аппаратными ускорителями, такими как GPU и TPU, становится все более важной. У управления ресурсами также является важной тенденцией.

Инструменты, такие как Prometheus и Grafana, позволяют собирать метрики производительности, отслеживать использование ресурсов и выявлять узкие места в работе кластеров. Автоматизация процессов мониторинга и управления позволяет оптимизировать работу кластеров, повысить эффективность использования ресурсов и снизить эксплуатационные расходы.

Перспективы развития

Развитие распределённых вычислительных платформ для машинного обучения связано с рядом ключевых направлений, каждое из которых направлено на повышение эффективности, масштабируемости и доступности соответствующих технологий.

Одним из приоритетных направлений является совершенствование инструментов для разработки и отладки распределённых приложений машинного обучения. Создание более удобных и интуитивно понятных средств разработки позволит упростить процесс прототипирования и тестирования, снизить порог входа для разработчиков и ускорить переход от экспериментов к производственным решениям. Особое внимание уделяется развитию подходов Federated Learning и Privacy-Preserving Machine Learning, которые позволяют обучать модели на распределённых и потенциально чувствительных данных без их передачи в централизованное хранилище. Это особенно важно в контексте обеспечения конфиденциальности, соблюдения нормативных требований (например, GDPR) и защиты персональных данных.

Также важным направлением является расширение поддержки различных языков программирования и популярных фреймворков

машинного обучения (таких как TensorFlow, PyTorch, JAX и др.) в рамках распределённых платформ. Это обеспечит гибкость в выборе инструментов, упростит интеграцию с существующими экосистемами и повысит продуктивность команд разработки.

Интеграция с потоковыми источниками данных и устройствами Интернета вещей (IoT) позволит эффективно обрабатывать данные в реальном времени, что откроет новые возможности для применения распределённых ML-систем в таких областях, как предиктивное обслуживание, мониторинг, киберфизические системы и умные города.

Наконец, развитие облачных сервисов, предоставляющих управляемые распределённые вычислительные платформы для машинного обучения (например, Google Vertex AI, Amazon SageMaker, Azure ML), позволит организациям быстро разворачивать масштабируемые решения без необходимости самостоятельно управлять вычислительной инфраструктурой. Это снижает операционные издержки и позволяет сосредоточиться на прикладных задачах и бизнес-ценности создаваемых моделей.

Заключение

В настоящей статье была рассмотрена значимость и важность распределённых вычислительных платформ, таких как Spark и Hadoop, для машинного обучения. Эти платформы предоставляют мощные инструменты для обработки, хранения и анализа больших объемов данных.

Методы обработки больших данных с использованием MapReduce и Spark, и анализ данных с помощью Spark MLlib, предоставляют широкий спектр возможностей для извлечения ценной информации из данных и прогнозирования тенденций. Масштабируемость и отказоустойчивость этих платформ делают их привлекательным выбором для организаций, стремящихся оптимизировать процессы анализа данных и получить конкурентные преимущества.

Развитие этих платформ и технологий, связанных с ними, будет продолжать расширять возможности машинного обучения и открывать новые перспективы для решения сложных задач в различных областях.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Агревалио, А. Большие данные и облачные вычисления: технологии и методы анализа / А. Агревалио. – М.: ДМК Пресс, 2019. – 256 с.

2. Васильев, Д. А. Облачные вычисления и аналитика данных: современные тенденции и перспективы / Д. А. Васильев // Информационные технологии и вычислительные системы. – 2018. – № 3. – С. 45–52.

3. Гаврилов, В. А. Большие данные: технологии и методы анализа / В. А. Гаврилов, Е. Н. Соколова. – М.: ДМК Пресс, 2020. – 368 с.

4. Давыдов, А. С. Аналитика данных в облачных вычислениях / А. С. Давыдов // Облачные вычисления и цифровые технологии. – 2017. – № 2. – С. 31–36.

5. Клименко, С. Облачные технологии и аналитика больших данных / С. Клименко // Высокие технологии. – 2019. – № 5. – С. 18–22.

УДК 004.9

ПОЛОВИНКИН А.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

МЕТОДЫ ИДЕНТИФИКАЦИИ И ИХ ПРИМЕНЕНИЕ В БИБЛИОТЕЧНОЙ СРЕДЕ

Рассматриваются основные способы идентификации, используемые в библиотечной среде, их преимущества и недостатки.

Основная задача идентификации фонда заключается в возможности распознавании и учета библиотечного фонда для их каталогизации, поиска, выдачи и учета.

Смысл, который закладывается в процессе фондовой обработки документов – это обеспечение его элементами, позволяющими определить его принадлежность к фонду (подфонду) и определить его место в нем, для ускорения его поиска, выдачи и расстановки.

Каждый документ в процессе обработки получает шифр хранения – это адрес документа в закодированном виде. Т.е. простыми словами, он показывает место документа на стеллаже. Изначально шифр хранения состоял из набора цифр, которые указывали принадлежность книги к шкафу, полке и его порядковый номер на ней. Такой порядок до сих пор сохраняется в основных фондах национальных библиотек. Но наиболее часто шифры хранения в российских библиотеках содержат классификационный индекс и авторский знак. Такой вариант шифра позволяет сделать расстановку

документов по отраслям, алфавитному порядку авторов или заглавий внутри отраслевого раздела [1].

Далее происходит процедура штемпелевания. Эта процедура простановки штемпеля библиотеки на титульном листе документа, этикетке первой стороны граммофонной пластинки, бобины, катушки, кассеты и т.д. Штемпель содержит наименование библиотеки. Общепринятый стандарт – 10х40 мм.

Следующим этапом идет простановка инвентарного номера. Проставляют его обычно выше или ниже штемпеля [2].

Еще одной немаловажной частью является формуляр документа. Он представляет собой бланк форматом 5х10 см. В нем содержатся вся информация, которая поможет идентифицировать документ и его основное предназначение заключается в замещении документа при его выдаче читателю. Пока документ остается в фонде, формуляр хранится внутри кармашка документа, а в момент выдачи вкладывается либо в формуляр пользователя, либо в специальные закладки-кармашки, которые выставляются на место документа. Еще одна роль, которую берет на себя формуляр – это юридическое доказательство при утере и невозврате документа в фонд. Он узаконивает ответственность пользователя за взятый документ, а также помогает отслеживать спрос на данный экземпляр [3].

В настоящее время в качестве метода идентификации часто используют штриховое кодирование. Оно заключается в нанесении на экземпляры документов самоклеющихся этикеток, на которых нанесен уникальный номер, представленный в виде штрих-кода, который распознается оптическим устройством. Этот номер вносится в базы данных библиотеки, где связывается с определенным экземпляром.

Штрих-кодирование, как технология автоматической идентификации, основано на стандартах GS1, включающих символы EAN-13 и Code 128. В библиотечной практике применяется модифицированный стандарт Code 39, позволяющий кодировать буквенно-цифровые данные инвентарных номеров. Физические ограничения метода обусловлены требованиями ISO/IEC 15420:2009 к минимальному размеру штрих-кода (15×37 мм при 300 dpi) и контрастности печати (не менее 60% оптической плотности). Практические исследования демонстрируют, что скорость сканирования при оптимальных условиях достигает 1-2 секунд на экземпляр, однако при повреждении этикетки или загрязнении этот показатель снижается на 40-60%.

Более технологичным вариантом идентификации является использование электромагнитных и радиочастотных систем. Также их

используют для защиты от кражи, особенно в условиях открытого фонда. Для их применения в библиотеке должен быть организован контрольно-пропускной пункт, в котором размещены устройства детектирования.

Деактивированный электромагнитный датчик реализуется в виде металлической нити на клейкой основе. Размер такого датчика 1x5 см. Это позволяет его легко спрятать внутри книги.

Что касается радиочастотных противокражных систем, то в начале развития использовали неактивированные датчики, которые обычно закрепляют на заднем форзаце документа под кармашком. И для нейтрализации такого датчика использовались специальные экранирующие сигнал полоски фольги, которые маскировались под формуляр издания, где указывалась дата возврата. На текущий момент используются метки, в которых программируется защитный бит. Для его снятия используется RFID-станция. Книги с установленной меткой кладут на станцию и через функционал, реализованный внутри АБИС, происходит снятие защиты. В это же время внутри системы прописывается информация о читателе и экземплярах документов, которые были выданы. Такой подход позволяет отслеживать движение документов, собирать статистику по книговыдаче и упрощает проведение инвентаризации.

RFID-технологии в библиотечном деле используют преимущественно высокочастотный диапазон 13,56 МГц (стандарт ISO 15693) с дальностью считывания 0,7-1,2 метра. Метки содержат память от 512 бит до 2 Кб, что позволяет хранить помимо уникального идентификатора (UID) дополнительную информацию в формате MARC 21. Критическим параметром является коэффициент успешного считывания, который в условиях плотной расстановки книг (15-20 экз./п.м.) снижается с заявленных 99,8% до 85-90% из-за эффекта экранирования электромагнитных волн. Экономический анализ показывает, что полный переход на RFID увеличивает стоимость обработки одного экземпляра в 3-4 раза по сравнению с штрих-кодированием.

Использование RFID-меток вместо штрих-кодов упрощает и сокращает время обслуживания читателя. Основная проблема штрих-кодов в этом плане заключается в необходимости прямой видимости для лазерного сканера и особая ориентация. А при выдаче документов работнику библиотеки необходимо последовательно сканировать каждый экземпляр документа, в отличие от книговыдачи с RFID-метками, где положив несколько документов на станцию, можно одновременно со всех этих экземпляров подготовить книгу к выдаче

читателю. Но большим недостатком RFID-систем является их дороговизна и невозможность обслуживания читателей в случае поломки оборудования или неисправности системы, так как нельзя снять защитный бит [4].

Относительно недавно для идентификации библиотечного фонда начали применять QR-коды. Технология QR-кодов (ISO/IEC 18004:2015) в библиотечной практике использует версии 3-4 (29×29 - 33×33 модуля) с коррекцией ошибки уровня L (7% повреждения). Экспериментальные данные показывают, что плотность размещения информации достигает 295 байт на 1 см² при минимальном размере кода 2×2 см. Однако практическое применение ограничивается необходимостью специализированного ПО для генерации кодов, интегрированного с библиотечными АБИС.

С помощью QR можно выводить дополнительную информацию о документе, которые может в последствие пригодиться читателю. Так, в 2017 году в Белгородской государственной универсальной научной библиотеке был реализован проект «Читай-качай». Он заключался в том, что на книгах размещался QR-код, при сканировании которого происходило перенаправление на этот документ в электронном виде, что позволяло читать экземпляр в удобное время [5].

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Морева О.Н. Организация библиотечного фонда: учебно-практическое пособие / О.Н. Морева. – Санкт-Петербург: Профессия, 2012. – 127 с.
2. Зупарова Л.Б. Библиотечная обработка документа: учебно-методическое пособие / Л. Б. Зупарова и др.; под науч. ред. Ю. Н. Столярова. – Москва: Либерия, 2013. – 208 с.
3. Столяров Ю.Н. Формирование библиотечного фонда: практическое пособие / Ю. Н. Столяров; Моск. гос. ун-т культуры и искусств. – Санкт-Петербург: Профессия, 2015. – 507 с.
4. Золотова Н.Ю. Использование технологии радиочастотной идентификации в библиотеках США: проблемы и перспективы // Открытый доступ. Библиотеки за рубежом 2006: Юбил. вып. сб. / ВГБИЛ им. М.И. Рудомино. Центр междунар. библиотековедения; сост. и отв. ред. С.В.Пушкова; отв. за вып. Л.М.Степачев. – М.: Рудомино, 2016. – С. 92 – 130.
5. Белгородская государственная универсальная научная библиотека в 2017 году. Отчёт. – Белгород, 2018. – С. 138.

УДК 004.6

ПОЛОВИНКИН В.В.

Рязанский государственный радиотехнический университет
имени В.Ф.Уткина

**ПРИМЕНЕНИЕ АЛГОРИТМОВ И МЕТОДОВ НЕЧЕТКОГО
СОПОСТАВЛЕНИЯ ДЛЯ ОЧИСТКИ
НЕСТРУКТУРИРОВАННЫХ МАССИВОВ ДАННЫХ В
РАЗЛИЧНЫХ СИСТЕМАХ АНАЛИЗА И
ОБРАБОТКИ BIG DATA**

Рассматриваются технологии и платформы, применяемые для очистки данных (Data Cleaning), с акцентом на использовани в них алгоритмов и методов нечеткого сопоставления (fuzzy matching), которые являются важным инструментом для решения проблем, связанных с обработкой неструктурированных массивов данных в Big Data.

По прогнозам, в 2025 году объём генерируемых больших данных (Big Data) достигнет 181 зеттабайта, в связи с чем необходимость в эффективных методах их очистки будет стоять по важности в самых первых рядах. Современные технологии и инструменты, используемые для обработки данных, должны не только справляться с огромными объемами информации, но и обеспечивать высокое качество данных, что, в свою очередь, является основополагающим фактором для принятия обоснованных бизнес-решений.

Главной проблемой является поступление неструктурированных и «грязных» информационных массивов, что затрудняет их эффективное использование. Ошибки, дубликаты и отсутствующие значения могут спровоцировать искажения в конечных результатах аналитики и привести к неправильным выводам [1].

Одним из передовых решений этой проблемы является инструмент контроля качества данных *Data Ladder*, который использует алгоритмы нечеткого сопоставления (fuzzy matching). Он представляет собой одну из самых прогрессивных систем для очистки данных, акцентируя внимание на таких технологиях, как *DataMatch* и *DataMatch Enterprise*. Эти решения созданы для автоматизации процессов очистки и сопоставления больших объемов данных, что позволяет существенно сократить затраты времени и ресурсов. Важной особенностью платформы является способность выявлять и устранять дубликаты, а также исправлять ошибки ввода, что значительно улучшает качество данных и повышает их ценность. Также *Data Ladder* помогает устранить противоречивые и недопустимые значения, создавать и проверять шаблоны, а также достигать

стандартизированного представления для всех источников информации. Продукт позволяет подключать и интегрировать данные из нескольких разрозненных источников, включая форматы файлов, реляционные базы данных, облачные хранилища и API, а также автоматизирует проверки качества данных и предоставляет мгновенные отчёты о профилях по пустым значениям, типам данных, шаблонам и другим статистическим сведениям. Data Ladder, поддерживает обработку до 100 миллионов записей, обеспечивая при этом высочайшую точность.

Другим современным продуктом для проведения операций по Data Cleaning с использованием fuzzy matching является *BigQuery*. Это облачная платформа для управления данными, встроенная в экосистему Google Cloud Platform (GCP). Она предоставляет ряд весоных функций для очистки и преобразования данных, например, при работе с невидимыми специальными символами в строковых значениях. Для этого есть функция *NORMALIZE*, которая удаляет специальные символы, хранящиеся в базе данных, но не отображающиеся на экране. Чтобы проверить соответствие шаблонов в *BigQuery* используется функция *CONTAINS_SUBSTR*, которая выполняет проверку без учёта регистра, а также может искать значения шаблона в числовых полях, датах и массивах. С форматированием дат помогает функция *FORMAT_DATE*, способная их форматировать в соответствии с заданными строками формата. Также как и Data Ladder, платформа предоставляет возможность удаления дубликатов на основе ключевых столбцов. Кроме того, *BigQuery* предлагает комнаты для очистки данных, которые позволяют создавать и управлять безопасными средами для обмена, анализа и сотрудничества по поступающей информации, не требуя её перемещения или копирования [2].

На аналитическая low-code платформе *Loginom*, которая позволяет проводить анализ данных любого уровня сложности без программирования, для проведения Data Cleaning применяется решение *Loginom Data Quality*. Оно автоматически исправляет ошибки, приводит информацию к стандартному виду, восстанавливает и обогащает клиентские данные. Очистка данных при использовании платформы может проводиться как перед их загрузкой в хранилище, так и в аналитическом приложении непосредственно перед анализом. Для проведения операций по очистке поступающей информации, *Loginom* также предлагает ряд инструментов и компонентов. Например, компонент «Статистика» применяется для визуализации данных, который помогает узнать, имеют ли поступившие массивы

пропуски, выбросы и экстремальные значения. Инструмент «Группировка» используется для устранения дубликатов: в группы попадают данные с одинаковой информацией, а в показатели — все остальные параметры с агрегацией «первый». Использование компонента «Редактирование выбросов», поможет устранить аномальные значения. Здесь также не обходится без использования алгоритмов нечеткой логики [3].

Используемые на вышеперечисленных передовых платформах fuzzy matching занимают важное место в концепции Data Cleaning, и особенно когда необходимо обрабатывать неидеальные и неоднородные источники информации. Эти алгоритмы могут успешно справляться с различными вариантами написания имен, адресов и других строковых данных, что делает их незаменимыми в современных условиях Big Data. Рассмотрим несколько из них подробнее.

Самым простым является «*наивный*» алгоритм: он ищет все символы основной строки в шаблоне. Точное сопоставление строк (выявление одного или всех точных экземпляров шаблона в тексте) представляет собой наивный подход. Самое главное, что не требуется никаких этапов предварительной обработки. Проверяя строку один раз, мы можем найти подстроку. Кроме того, для выполнения этого процесса не требуется дополнительного места.

Другим приемом является применение *алгоритма определения расстояния Хэмминга*. Это математическая концепция, используемая в информатике в различных областях, таких как обработка сигналов и телекоммуникации. Она играет важную роль в алгебраической теории модификации кода. Фактически, она позволяет количественно оценить разницу между двумя последовательностями символов. В случае двух строк расстояние Хэмминга обозначает количество позиций, в которых соответствующие символы отличаются.

Третий алгоритм использует *метод расстояния Левенштейна*. Он позволяет измерить сходство между исходной строкой s и целевой строкой t . Расстояние между s и t представляет собой количество вставок, замен и удалений.[4]

N-граммовый алгоритм. N-грамма — это непрерывная последовательность n элементов из чётко определённой текстовой или звуковой последовательности. В зависимости от области применения мы можем отображать объекты в виде слогов, слов, символов или пар оснований. Вероятностная языковая модель, или n -граммная модель, — это разновидность вероятностной языковой модели. В форме $(n - 1)$

порядковой марковской модели можно прогнозировать следующий элемент в такой последовательности.

Алгоритм Bitap – это алгоритм приблизительного сопоставления строк. Алгоритм показывает, содержит ли заданный текст подстроку, «приблизительно равную» заданному шаблону, где приблизительное равенство определяется расстоянием Левенштейна. Если шаблон и подстрока находятся на заданном расстоянии k друг от друга, то алгоритм считает их равными. Преимуществом этого алгоритма перед другими методами является использование побитовых операций. Это делает его очень быстрым [5].

Fuzzy matching способны распознать адреса в двух различных форматах, таких как «г. Рязань, ул. Гагарина, д. 59» и «Гагарина 59_Рязань», оценивая степень их схожести от 0 до 1. Это позволяет гибко настраивать порог совпадения для каждого конкретного случая.

Для выполнения задач нечеткого сопоставления (approximate string matching) используются такие пакеты, как stringdist и fuzzyjoin в среде R, которые позволяют аналитикам эффективно обрабатывать несовершенные данные и обеспечивает высокую точность при выполнении нечеткого сопоставления. В результате, алгоритмы нечеткого сопоставления учитывают множество возможных ошибок, включая опечатки, сокращения, фонетические ошибки и даже различные порядки слов. Fuzzy matching могут преобразовывать данные так, чтобы обеспечить единственное и стандартное представление для всех записей, что критически важно для анализа и интеграции данных.

Кроме того, нечеткое сопоставление активно использует фонетические алгоритмы для улучшения качества данных. Они позволяют приводить имена и адреса к единому формату, обеспечивая тем самым лучшую достоверность информации. Например, на этапе стандартизации, все ошибки в написании или редактировании данных устраняются, что помогает избежать множества дубликатов и гарантирует, что данные корректно интегрированы из разных источников.

Процесс нечеткого сопоставления также может включать секундные таблицы и операции в SQL, что является важным аспектом для достижения большей эффективности и снижения времени обработки. Эти технологии помогают создавать более изящные и мощные интеграционные процессы для анализа данных. С помощью нечеткого сопоставления можно значительно улучшить качество данных, что приводит к более точной аналитике и, следовательно, к более обоснованным бизнес-решениям [6].

Качество данных непосредственно сказывается на аналитических выводах и, как следствие, на принятии решений в бизнес-среде. Использование методов нечеткого сопоставления позволяет исключить дубликаты и устранить ошибки в данных, что обеспечивает влияние на результативность бизнеса и повышает производительность его процессов. Понимание и применение этих методов является важным навыком для специалистов, занимающихся обработкой больших массивов данных.

Подводя итог, можно с уверенностью сказать, что алгоритмы нечеткого сопоставления и подходы к очистке данных являются необходимыми инструментами в современном мире Big Data. Эти технологии позволяют не только улучшить качество данных, но и повысить точность последующего анализа, что в свою очередь освобождает аналитиков от необходимости бороться с некорректными записями и ошибками. Индивидуальные параметры для каждого набора данных можно легко адаптировать, что делает такие подходы универсальными и эффективными в разных областях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алексеев К.А. Использование Big Data в международном бизнесе//Труды Института системного программирования РАН. – 2020. – №32(4). – С. 7-20.
2. Google BigQuery. Всё о хранилищах данных, аналитике и машинном обучении. – СПб.: Питер, 2021. – 496 с.
3. Жукова, Л. В. Интеллектуальный анализ данных на платформе LogiNot: методические указания / Л. В. Жукова. – М.: Издательский Дом НИТУ «МИСиС», 2023. – 45 с;
4. Лещенко А.В. Обзор и практическое применение алгоритмов нечеткого поиска / Сборник научных трудов НГТУ. – 2018. – № 3–4 (93). – С. 59–69.
5. Агафонов, В. Н. Методы нечеткого анализа данных: учебное пособие / В. Н. Агафонов, Д. А. Якушев. – Томск: Изд-во ТПУ, 2014. – 142 с.
6. Лемер Джим. Нечеткое сопоставление данных в SQL – Астана: «Спринт Бук», 2024 – 288 с.

УДК 004.855.6

ПОПОВА А.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

СИСТЕМА ПЕРСОНАЛИЗИРОВАННЫХ РЕКОМЕНДАЦИЙ ДЛЯ ВЫБОРА ТУРИСТИЧЕСКИХ МАРШРУТОВ НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Рассматривается разработка модуля персонализированных рекомендаций маршрутов для туристов, использующего методы интеллектуального анализа данных и машинного обучения, предназначенного для работы в составе информационной системы развития и поддержки городского туризма. Функционал модуля реализует формирование индивидуальных туристических маршрутов, адаптированных под интересы и поведение конкретного пользователя.

Современные компьютерные технологии активно внедряются в сферу туристического бизнеса, используя интеллектуальные информационные системы (ИИС) для развития и поддержки городского туризма [1]. Рост внутреннего туризма, высокие ожидания со стороны путешественников и конкуренция делают крайне актуальным создание систем, способных:

- понимать и учитывать интересы пользователей;
- анализировать туристические потоки;
- предоставлять удобные инструменты планирования маршрутов;
- поддерживать устойчивое развитие туризма (снижение нагрузки на популярные объекты и привлечение внимания к менее известным).

Использование искусственного интеллекта (ИИ) в данной сфере позволяет перейти от шаблонного информационного подхода к адаптивному, интеллектуальному взаимодействию с потребителем [4].

Традиционные справочные системы (карты, каталоги, путеводители) не обеспечивают персонализированного подхода к каждому туристу, анализа его поведения и интересов, адаптации под текущие условия (загруженность, сезонность, предпочтения) [5].

ИИ позволяет анализировать большие объёмы пользовательских данных в реальном времени, адаптировать предложения под интересы и потребности, обучаться на основе пользовательской активности и предсказывать предпочтения и поведение [4].

Благодаря этому, турист получает не просто список объектов, а удобный, релевантный и интересный маршрут, что существенно повы-

шает удовлетворённость от поездки и стимулирует повторное посещение города.

Ключевая цель рассматриваемого модуля — предоставление персонализированных маршрутов и рекомендаций по достопримечательностям с учётом:

- истории действий пользователя;
- пользовательских предпочтений;
- популярности объектов;
- загрузки туристической инфраструктуры.

Среди задач, решаемых системой: сегментация пользователей, выдача маршрутов, адаптированных под профиль туриста, и предложение альтернативных вариантов в случае перегруженности основных маршрутов.

Принцип работы формирования рекомендаций происходит в несколько этапов:

1. Сбор истории действий пользователя (просмотры, посещения, отзывы).
2. Выделение предпочтительных категорий интересов.
3. Формирования поведенческого профиля.
4. Применение выбранной стратегии рекомендации.
5. Генерация персонализированного маршрута и отправка пользователю.

Блок-схема алгоритма формирования рекомендаций представлена на рисунке 1:

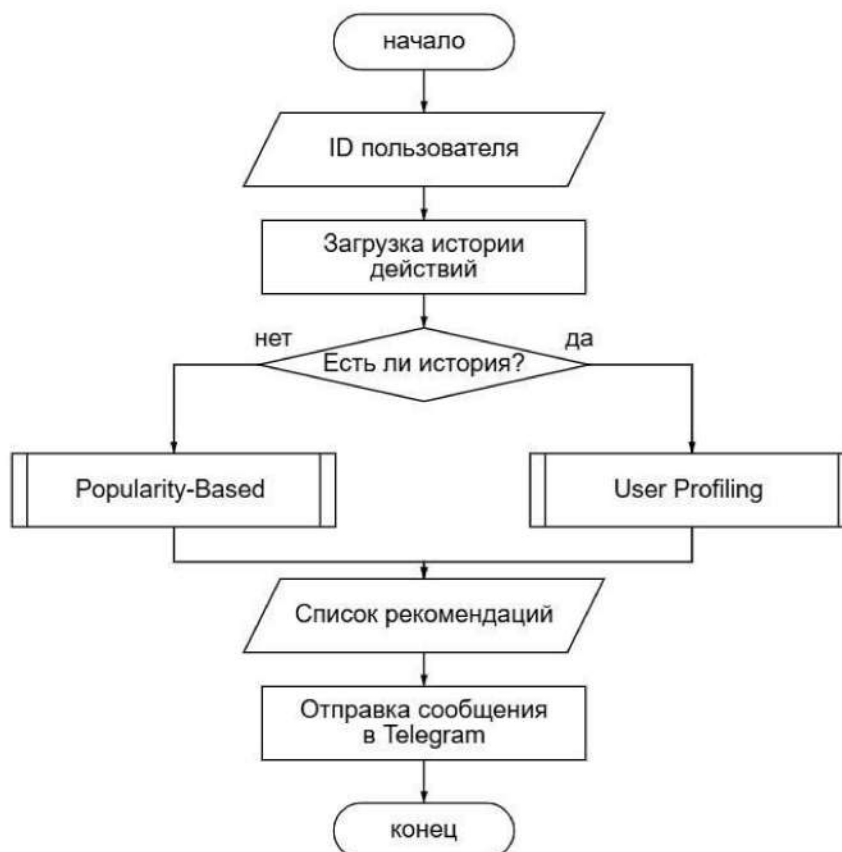


Рисунок 1 – Блок-схема формирования рекомендаций

Для реализации соответствующего функционала разработаны следующие основные методы:

1. Content-BasedFiltering (фильтрация на основе контента) – метод, предполагающий выдачу маршрутов и точек интереса (POI – Pointofinteresting–точка интереса) на основе индивидуальных предпочтений пользователя, его истории посещений, оставленных отзывов и оценок [2]. Блок-схема метода представлена на рисунке 2.



Рисунок 2 – Блок-схема метода Content-Based Filtering

2. Popularity-Based (фильтрация на основе популярности) – рекомендации, которые формируются с учетом рейтинга объектов, популярности среди других туристов и количества положительных отзывов [2]. Этот подход особенно эффективен для новых пользователей. Блок-схема метода представлена на рисунке 3.



Рисунок 3 – Блок-схема метода Popularity-Based

3. Clustering – метод сегментирования пользователей. Позволяет выделить поведение и тематические группы, на основании которых система предлагает подходящие маршруты [4], [5].

4. UserProfiling – на основе анализа поведения пользователя формируется его профиль – определяется тип (например, семья, молодежь, иностранный турист), уровень активности и предпочтения [4]. Позволяет адаптировать стратегию рекомендаций под каждого конкретного пользователя. Блок-схема метода представлена на рисунке 4.



Рисунок 4 – Блок-схема метода UserProfiling

Профиль пользователя формируется автоматически на основе истории взаимодействия с системой, оценок и геоданных. Он представляет собой набор характеристик, отражающих:

1. Поведенческие параметры:

- частота путешествий – определяется по количеству завершенных маршрутов;
- продолжительность поездок – по средней длительности маршрутов;
- предпочтительный стиль маршрута – темп (интенсивность), время суток, количество объектов в день;
- оценки и отзывы – объекты, получившие высокую оценку, попадут в список предпочтений;

- категории интересов – музеи, природа, еда и т.д. – вычисляется по взаимодействию с POI;

- активность в системе – количество запросов, отзывов, взаимодействий.

2. Географические предпочтения:

- любимые регионы и города – фиксируются по истории маршрутов;

- предпочтительная удалённость – городской или загородный туризм;

- геолокация пользователя - начальная точка или домашний район;

3. профильный тип пользователя (сегмент):

- определяется через кластеризацию (например, семья с детьми, молодой активный турист, пожилой турист, гастрономический путешественник, иностранец и т.д.

Пример профиля пользователя в формате JSON приведет в листинге 1.

Листинг 1. Пример профиля пользователя в формате JSON

```
{
  "userId": 78459328,
  "profileType": "Семейныйкультурныйтурист",
  "language": "ru",
  "preferredTransport": ["поезд", "автобус"],
  "tripFrequency": "2-3 раза в год",
  "avgTripDuration": "5-7 дней",
  "budgetLevel": "средний",
  "preferredPOICategories": ["музеи", "архитектура",
    "парки"],
  "preferredCities": ["Казань", "Санкт-Петербург"],
  "poiRatingPattern":
    "высокие оценки культурным объектам",
  "routeStyle": "3-4 объектов в день,
    близко друг к другу",
  "userActivity": "высокая",
  "geoBase": "Советский район"
}
```

Для построения системы персонализированных рекомендаций были рассмотрены и реализованы несколько подходов, каждый из которых обладает своими преимуществами и ограничениями. Выбор конкретного метода или их комбинации зависит от доступных пользовательских данных, стадии взаимодействия пользователя с

системой и целей рекомендаций (персонализация, вовлеченность, расширение кругозора и пр.) [2], [4].

Ниже представлен сравнительный анализ используемых методов, отражающий их применимость, сильные и слабые стороны (таблица 1).

Таблица 1 – Сравнительный анализ методов

<i>Метод</i>	<i>Преимущества</i>	<i>Недостатки</i>	<i>Область применения</i>
Content-BasedFiltering	Персонализированные рекомендации, адаптация под интересы	Ограничено историей пользователя, не предлагает новое	Для активных пользователей с историей действий
Popularity-Based	Легко реализуется, подходит для новых пользователей, всегда есть результат	Не учитывает интересы, может навязывать популярное	Новые пользователи
Clustering	Массовая сегментация	Сложность реализации	Системный уровень
UserProfiling	Учитывает тип пользователя и адаптирует стратегию	Требуют качественной истории действий и сегментации	Для всех пользователей после минимального взаимодействия с системой

Для комплексной оценки разрабатываемой системы и применяемых в ней методов был проведён SWOT-анализ (таблица 2). Он позволяет выявить сильные и слабые стороны предложенного решения, оценить возможности его развития, а также потенциальные риски, связанные с внедрением в реальной городской среде.

Таблица 2 – SWOT-анализ

<i>Сильные стороны</i>	<i>Слабые стороны</i>
Простота и надежность	Не использует сложные ML-модели
Возможность персонализации	Ограничено контентно-базируемыми данными
Легко доработать и масштабировать	Рекомендует только то, что уже известно

Возможности	Риски
Добавить в будущем кластеризацию, CF, рекомендации на основе маршрутов	Недостаток данных на старте системы
Интеграция с открытыми API города	Возможность дублирования рекомендаций
Расширение для других городов	Потребуется контроль качества данных

Проведённый сравнительный анализ показал, что каждый метод обладает своими преимуществами и ограничениями, однако их комбинированное использование позволяет охватить широкий спектр пользовательских сценариев. SWOT-анализ подтвердил, что предложенная система является устойчивым, масштабируемым и практически применимым решением для реального использования в городском туризме.

В результате, система повышает качество пользовательского опыта, способствует более равномерному распределению туристических потоков, открывает возможности для интеграции с городской инфраструктурой.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Информационные технологии в социально-культурном сервисе и туризме. Оргтехника: учебник для студ. высш. учеб. заведения / М.А. Морозов, Н.С. Морозова. – 7-е изд., стер. – М.: Издательский центр «Академия», 2009 – 240 с.

2.Recommender Systems Handbook. Springer [Электронный ресурс] / Ricci, F., Rokach, L., & Shapira, B. (2015) – URL: https://www.researchgate.net/publication/227268858_Recommender_Systems_Handbook (дата обращения: 19.03.2025).

3. Информационные системы как основа туристского бизнеса. / А.В. Столяренко, Г. Жернакова// Научно-методический электронный журнал «Концепт». – 2017. – № 5 (май). – С. 95-110.

4. Системы искусственного интеллекта: учеб. пособие. В 2-х частях. /С. Н.Павлов. – Томск: Эль Контент, 2011 – Ч. 1 –176 с.

5. Основы машинного обучения: учебное пособие / О.В. Лимановская, Т.И. Алферьева; Мин-во науки и высш. образования РФ. – Екатеринбург : Изд-во Урал. ун-та, 2020. – 88 с.

УДК 004.032.26

ПОЧТАРЕВ А.В.Рязанский государственный радиотехнический университет
имени В.Ф. Уткина**ИССЛЕДОВАНИЕ МЕТОДОВ ПОВЫШЕНИЯ КАЧЕСТВА И
ПРОИЗВОДИТЕЛЬНОСТИ ПРИ ОБРАБОТКЕ ИЗОБРАЖЕНИЙ
С ГЛОБАЛЬНЫМ ОСВЕЩЕНИЕМ**

Рассматриваются методы применения нейросетевых моделей для апскейлинга с целью повышения визуального качества графических изображений без значительного увеличения вычислительных затрат.

В современном программном обеспечении наблюдается стремление к повышению визуального качества графических изображений без значительного увеличения вычислительных затрат. Одним из наиболее перспективных направлений является применение нейросетевых моделей для апскейлинга – технологии, позволяющей восстанавливать детали изображения низкого разрешения и получать итоговый результат высокого качества. Примером такого подхода является технология DLSS от Nvidia [1], широко используемая в игровой индустрии.

Целью данной системы является снижение вычислительных затрат на рендеринг с сохранением высокого качества изображений, что особенно актуально для сцен с глобальным освещением, которые используются в интерактивных приложениях и играх. Основная идея состоит в том, чтобы генерировать изображения с глобальным освещением в низком разрешении и затем с помощью нейросетевого апскейлера, основанного на модели RealESRGAN [2], восстанавливать детали и повышать итоговое разрешение.

Для интеграции решения в уже существующий проект выбран ONNX Runtime [3], позволяющий проводить инференс модели непосредственно в приложении без необходимости разворачивать дополнительный сервис. Помимо этого, ONNX Runtime поддерживает аппаратное ускорение (например, CUDA, TensorRT), что даёт возможность дальнейшей оптимизации для работы в реальном времени.

Внедрение данной системы позволяет добиться следующих выгод:

- Снижение затрат на вычислительные ресурсы: генерация изображений в низком разрешении с последующим апскейлингом требует меньших затрат по сравнению с прямым рендерингом

изображений высокого разрешения. Это особенно актуально для систем реального времени, где время обработки критично.

- **Повышение качества конечного продукта:** улучшение визуальных характеристик изображений повышает конкурентоспособность продукта, что может привести к увеличению числа пользователей и росту доходов.

- **Масштабируемость:** возможность интеграции аппаратного ускорения позволяет расширять систему без существенного увеличения затрат, что важно для крупных проектов и промышленных решений. Это достигается благодаря использованию кросс-платформенного решения ONNX Runtime.

Апскейлер и модель RealESRGAN

Апскейлинг – процесс увеличения разрешения изображения с целью повышения его качества без потери деталей – является актуальной задачей в компьютерной графике. Он основан на глубоком обучении: использует глубокие сверточные нейронные сети, которые обучены масштабировать изображение. Если копнуть глубже, он основан на генеративно-сопоставительных сетях, их там целых две. Одна сеть пытается повысить качество изображения, а вторая действует как критик-дискриминатор, оценивая, насколько хороша полученная картинка, и определяя, поддельная она или нет. Если изображение объявлено фейковым, от него отказываются. Генерация и критика крайне важны для процесса апскейлинга: обе сети должны получить информацию об изображениях, с которыми они имеют дело. Дискриминатор должен выявить определенные характеристики картинки. Генератор же должен создавать изображения, сохранив художественную ценность оригинала в достаточном объеме, чтобы обмануть дискриминатора.

Среди существующих решений можно выделить модель RealESRGAN, которая является развитием базовой архитектуры ESRGAN.

ESRGAN (Enhanced Super-Resolution Generative Adversarial Network или расширенная генеративно-сопоставительная сеть сверхвысокого разрешения). Алгоритм-генератор ESRGAN работает благодаря сверточной нейронной сети, использующей сверточные слои для сбора информации об исходном изображении с низким разрешением [4]. При передаче это изображение захватывает так называемое «пространство признаков». Этот набор свойств используется для описания визуальных паттернов, например: изображен ли на картинке мех, кирпич или какой-то другой известный материал. Сверточные слои нужны для обработки и хранения этой

информации, чтобы при создании изображения со сверхвысоким разрешением оно сохраняло то же пространство признаков, что и исходник. Получив эту информацию, генератор начинает масштабировать изображение.

На рисунке 1 приведена схема работы модели ESRGAN.

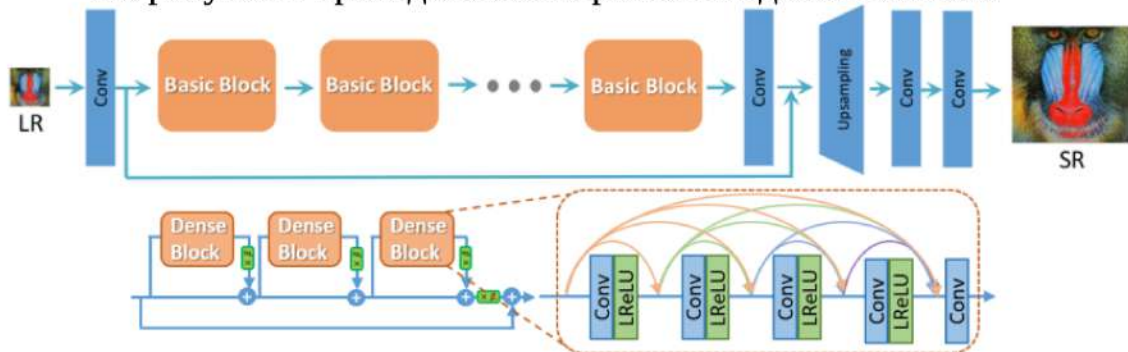


Рисунок 1 – Схема работы модели ESRGAN

Ключевым компонентом ESRGAN является базовый блок, который представляет собой повторяемую модульную структуру, лежащую в основе генератора. Этот блок отвечает за извлечение высокоуровневых признаков из входного изображения и их преобразование в изображение с более высоким разрешением.

Далее будет приведено сравнение RealESRGAN с другими апскейлерами. На рисунке 2 представлено исходное изображение, уменьшенное в 4 раза.



Рисунок 2 – Исходное изображение, уменьшенное в 4 раза

На рисунке 3 представлено сравнение моделей апскейла на возможность качественно восстанавливать изображение.

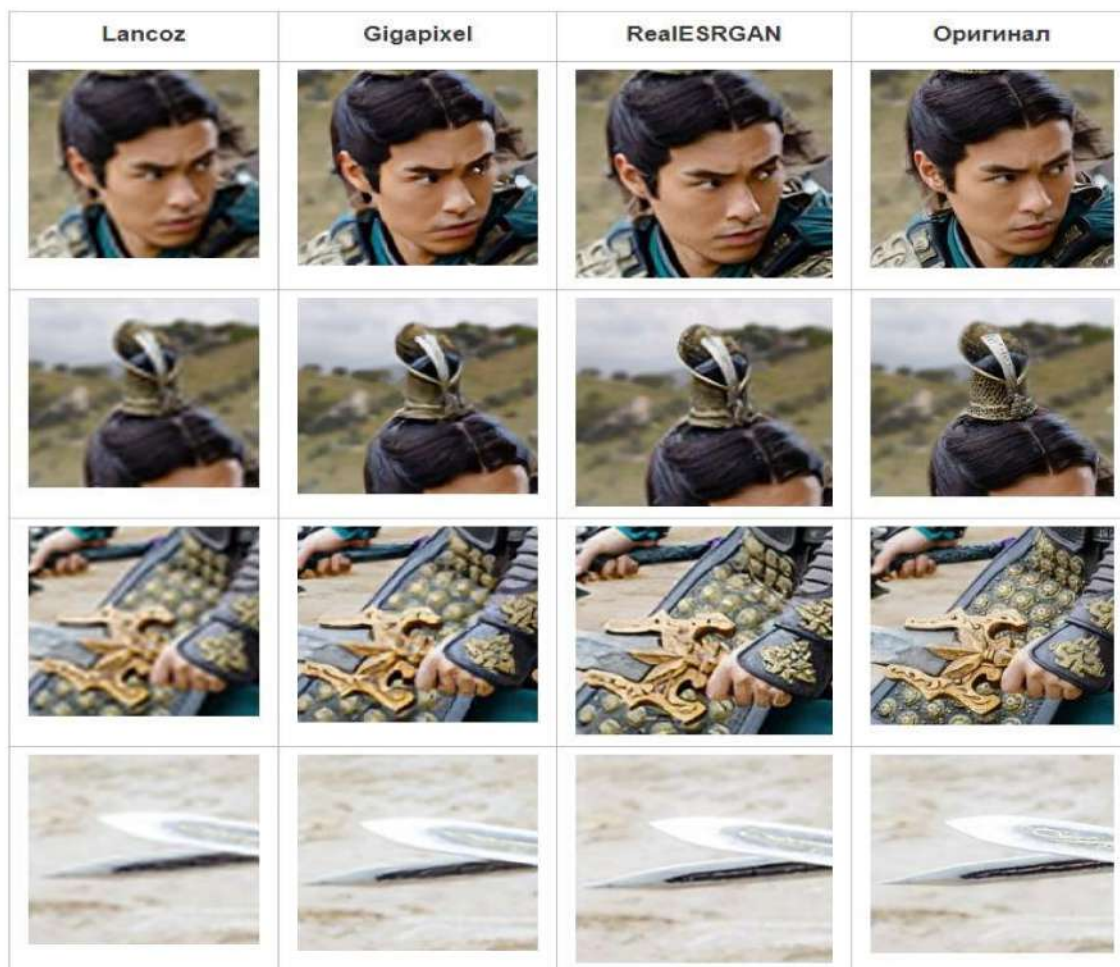


Рисунок 3 – Сравнение моделей апскейла

На основе полученных результатов, можно заметить, что модель RealESRGAN способна воспроизводить мелкие детали и устранять шум, что особенно важно для изображений с глобальным освещением, где присутствует мягкое распределение света и тени. Помимо этого, благодаря возможность дообучения, можно адаптировать модель под конкретные типы изображений и особенности рендеринга.

ONNX и ONNX Runtime

ONNX (OpenNeuralNetworkExchange) представляет собой открытый формат для обмена моделями нейронных сетей между различными фреймворками, такими как PyTorch, TensorFlow и другими [5]. Экспорт модели в ONNX позволяет получить единый файл, который можно использовать для инференса в различных средах без привязки к конкретному языку программирования. Его также можно представить в виде графа, который шаг за шагом показывает, как преобразовать признаки для получения прогноза. Поэтому модель машинного обучения, реализованную с помощью ONNX, часто называют графом ONNX.

На рисунке 4 представлен ONNX граф.

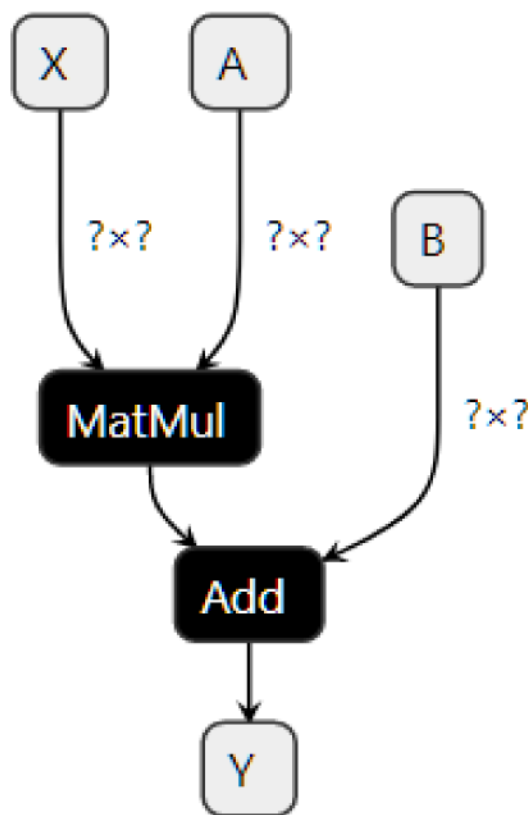


Рисунок 4 – ONNX граф

ONNX Runtime – высокопроизводительный движок для инференса моделей, представленных в формате ONNX, обеспечивающий совместимость ML-моделей с ИИ-фреймворками (TensorFlow, CognitiveToolkit, Caffe2, MXNet). Поэтому ONNX Runtime используют для оптимизации вычислений в моделях глубокого обучения нейронных сетей. Он обладает следующими особенностями:

- Кросс-платформенность: Поддержка работы на разных операционных системах и аппаратных платформах (CPU, GPU, а также ускорители вроде TensorRT).
- Оптимизация инференса: Возможность проведения оптимизаций для повышения производительности.
- Многоязыковая поддержка: Существует обёртка для многих языков программирования, что позволяет интегрировать инференс непосредственно в проект на этом языке.

Тестирование

Для тестирования данного подхода необходимо оценить качество полученных изображений, а также измерить производительность системы. В первом случае нужно собрать набор разнообразных изображений низкого и высокого разрешения с глобальным освещением для того, чтобы можно было объективно сравнить результаты апскейлинга с исходными данными. Помимо этого, использование метрик PSNR (PeakSignal-to-NoiseRatio) и SSIM (StructuralSimilarityIndex) поможет количественно оценить, насколько апскейленный вариант изображения приближен к эталонному. Дополнительно можно провести визуальное сравнение исходных и обработанных изображений, а при необходимости организовать опрос или тестирование с участием пользователей для сбора субъективных оценок качества.

Для измерения производительности системы необходимо провести серию тестов для измерения времени инференса одного изображения на различных устройствах (на CPU и на GPU). Следует также отслеживать потребление оперативной памяти и загрузку центрального и графического процессора. Проведение стресс-тестирования в режиме пакетной обработки поможет оценить, насколько система способна справляться с повышенной нагрузкой и обеспечивать работу в режиме реального времени.

Подводя итоги, можно сказать, что рассмотренный подход, основанный на апскейлинге изображений с применением модели RealESRGAN должен позволить значительно оптимизировать процесс рендеринга путем уменьшения времени обработки одной сцены без значительного изменения качества итоговой картинки. Также, благодаря ONNXRuntime удастся обеспечить аппаратное ускорение и масштабируемость, что особенно важно для интерактивных приложений. Представленный подход имеет потенциал для дальнейшей адаптации под конкретные задачи и расширения функционала за счёт дообучения модели.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. NVIDIA DLSS. [Электронный ресурс] – URL: <https://developer.nvidia.com/rtx/dlss> (дата обращения: 15.03.2025).
2. Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data. [Электронный ресурс] – URL: <https://arxiv.org/abs/2107.10833> (дата обращения: 15.03.2025).
3. ONNXRuntime. [Электронный ресурс] – URL: <https://onnxruntime.ai/docs/> (дата обращения: 15.03.2025).

4. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. [Электронный ресурс] – URL: <https://esrgan.readthedocs.io/en/latest/> (дата обращения: 15.03.2025).

5. Open Neural Network Exchange. [Электронный ресурс] – URL: <https://onnx.ai/onnx/intro/> (дата обращения: 15.03.2025).

УДК 004.032.26

ПРОХИН А.И.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ТРАНСФОРМЕР – АРХИТЕКТУРА НЕЙРОННЫХ СЕТЕЙ

Рассматривается популярная в настоящее время архитектура нейронных сетей для работы с последовательностями Трансформер, в частности для обработки текста. Выделяются преимущества и недостатки модели.

В последние годы глубокое обучение коренным образом трансформировало методы автоматической обработки текстов, открыв новые возможности в области их генерации, анализа и модификации. Среди множества архитектур нейронных сетей особое место занимает Трансформер.

Архитектура Трансформер — это революционная модель глубокого обучения, представленная в 2017 году. Она кардинально изменила подходы к обработке последовательных данных, заменив устаревшие рекуррентные (RNN) и свёрточные (CNN) нейронные сети механизмом внимания. Это позволило значительно ускорить обучение, улучшить параллелизацию вычислений и повысить качество моделирования длинных зависимостей в данных. В этой статье будут рассмотрены причины появления Трансформера, а также будет описана организация данной архитектуры.

Предпосылками к появлению этой модели послужили следующие проблемы в рекуррентных и свёрточных нейронных сетях:

1. Медленное обучение из-за последовательных вычислений.

RNN обрабатывают данные пошагово – каждое новое состояние зависит от предыдущего, и из-за этого невозможно параллелизовать обработку последовательности на GPU, так как каждый шаг требует завершения предыдущего.

2. Трудности с длинными зависимостями.

На очень больших текстах теряются зависимости, и информация из дальних частей текста и начала перестает быть одним целым.

3. Ограниченная параллелизация.

Хотя CNN можно параллелить лучше, чем RNN, они всё равно менее эффективны для задач, где важны глобальные взаимодействия, например, обработка местоимений.

Изначально архитектура Трансформер разрабатывалась в целях ускорения перевода текстов на разные языки, поэтому при первоначальном взгляде на нее можно выделить 2 детали encoder и decoder.

Графическое представление модели представлено на рисунке 1.

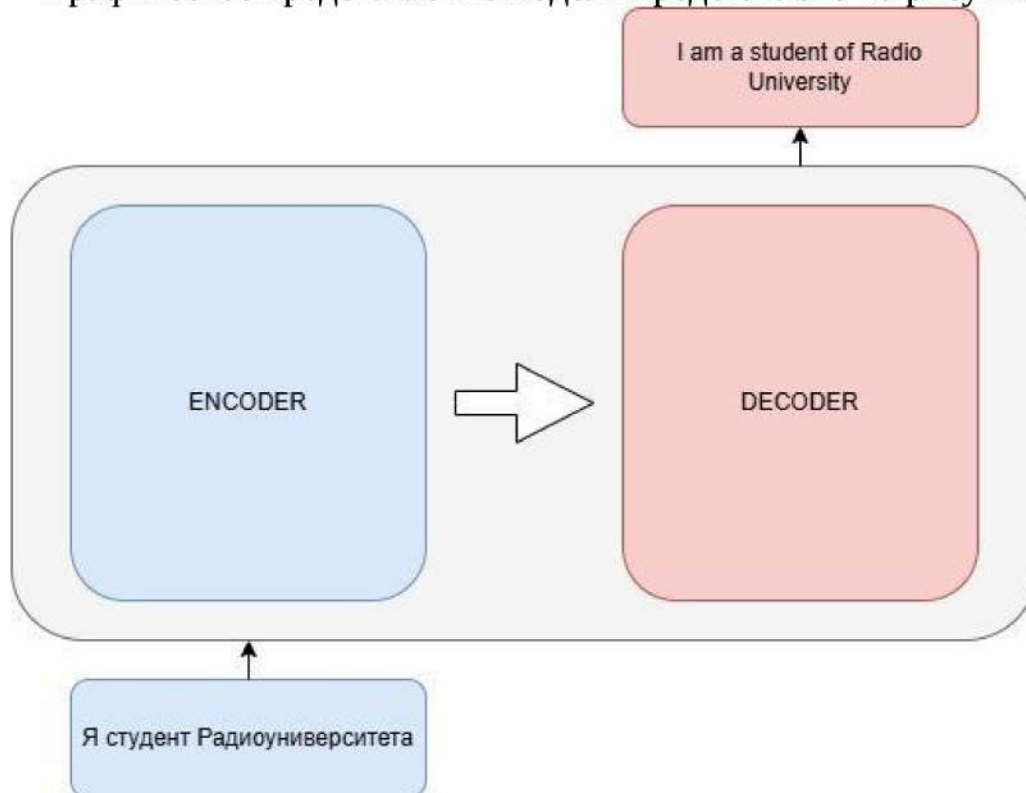


Рисунок 1 – Представление архитектуры

Организация и работа Трансформера будет разобрана на примере предложения: «Я студент Радиоуниверситета».

1. Токенизация и векторное представление и позиционирование

Входное предложение разбивается на токены: ["Я", "студент", "Радиоуниверситета"]

Затем каждому токenu присваивается векторное значение.

Например:

"Я" -[0.2, -0.1, 0.5]

"студент" -[0.7, 0.3, -0.2]

"Радиоуниверситета" -[-0.5, 0.6, 0.3]

После проводится закрепление позиции за каждым словом (Positional Encoding). Для этого рассчитывается информация о позициях по формулам 1 и 2 и добавляется к векторам.

Для четных индексов вектора:

$$PE(pos, i) = \sin\left(\frac{pos}{10000^{i/d_{model}}}\right) \quad (1)$$

,где d_{model} – размер вектора (в данном случае 3)

Для нечетных индексов вектора:

$$PE(pos, i) = \cos\left(\frac{pos}{10000^{i/d_{model}}}\right) \quad (2)$$

2. Encoder

Для каждого слова обучаемые матрицы W_Q , W_K , W_V преобразуют входные векторы в:

Query (Q) – что ищем?

Key (K) – на что обращаем внимание?

Value (V) – что передаём дальше?

Для расчета используется формула 3:

$$N_x = X \times W_N \quad (3)$$

,где $N = \{Q, K, V\}$, X –входной вектор.

Далее по формуле 4 вычисляется внимание, то насколько слово связано с другими:

$$Attention\ Weights = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \quad (4)$$

А затем рассчитывается взвешенная сумма по формуле 5:

$$Output = Attention\ Weights \times V \quad (5)$$

Взвешенная сумма позволяет каждому слову "видеть" всю последовательность сразу, выбирая релевантные части контекста через веса внимания.

После полносвязный слой (Feed-ForwardNetwork) обрабатывает отдельно каждый вектор, добавляя нелинейность и расширяя возможности модели обрабатывать сложные паттерны после механизма внимания.

Далее ResidualConnection (остаточное соединение) — решает проблему "затухающих градиентов", позволяя информации свободно проходить через слои, а LayerNorm (слой нормализации) — стабилизирует обучение.

Итог энкодера:

Векторы теперь учитывают контекст всего предложения:

"Я" знает, что является подлежащим, связанным с "студент".

"Радиоуниверситета" ассоциирован с "студент".

3. Decoder

На вход декодера поступает текущая частично сгенерированная последовательность, в начале, когда еще ничего не сгенерировано – это [START].

Далее MaskedSelf-Attentionобеспечивает последовательную генерацию выходных данных. Этот механизм анализирует только уже сгенерированные токены, искусственно "маскируя" (блокируя) информацию о будущих элементах последовательности.

После в дело вступает Cross-Attention — механизм взаимодействия между энкодером и декодером, позволяющий модели соотносить элементы входной и выходной последовательностей.

Например:

На этапе генерации слова "student" он фокусируется на исходном слове "студент" (максимальный вес внимания), дополнительно учитывает контекст: "Я" (подтверждает, что это 1-е лицо) и "Радиоуниверситета" (уточняет принадлежность).

4. Итог

В итоге получилось, что:

Encoder выявил:

"Я" → субъект,

"студент" → роль,

"Радиоуниверситета" → место.

Decoder использовал это для грамотного перевода.

5. Преимущества и недостатки

Преимущества:

- Параллельная обработка данных;
- Масштабируемость;
- Долгосрочные зависимости.

Недостатки:

- Высокие вычислительные затраты;
- Требовательность к данным;
- Проблемы с очень длинными последовательностями;
- Невозможность обучения с нуля.

Архитектура Трансформер совершила настоящую революцию в области обработки естественного языка, преодолев ключевые ограничения рекуррентных (RNN) и сверточных (CNN) сетей. Благодаря механизму внимания модель научилась эффективно обрабатывать длинные зависимости, параллельно анализировать контекст и достигать высочайшего качества в задачах машинного перевода, генерации текста и других NLP-приложениях.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Transformer в картинках [Электронный ресурс] – URL: <https://habr.com/ru/articles/486358/> (дата обращения: 01.02.2025).
2. Архитектура Transformer. Attention is all you need [Электронный ресурс] – URL: <https://wiki.pmifi.ru/disciplines/> (дата обращения: 01.02.2025).
3. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need // Advances in Neural Information Processing Systems (NeurIPS). – 2017. – Vol. 30.

УДК 004.65

САПРЫКИН А.Н., АРУТЮНОВА А.В.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

**МЕТОДЫ И АЛГОРИТМЫ ПРЕОБРАЗОВАНИЯ 3D-MODEЛЕЙ
В УПРАВЛЯЮЩИЕ КОДЫ ДЛЯ ПРИНТЕРОВ**

В статье рассматриваются современные алгоритмы и методы преобразования трехмерных моделей в управляющие коды для 3D-принтеров. Анализируются основные этапы подготовки модели, включая сегментацию, генерацию траекторий движения и формирование G-кодов. Особое внимание уделяется автоматизации процессов и оптимизации траекторий для повышения точности и скорости печати.

Технология трехмерной печати за последние годы стала важнейшим направлением в области производства и прототипирования. Одним из ключевых этапов процесса является преобразование цифровых трехмерных моделей в управляющие коды, которые интерпретируются 3D-принтером для формирования физической детали. Эффективность этого преобразования напрямую влияет на качество, точность и скорость изготовления объекта. Разработка и внедрение алгоритмов автоматизации преобразования 3D-моделей в управляющие коды позволяют значительно повысить производительность и снизить затраты при производстве сложных изделий.

Процесс преобразования трехмерных моделей в управляющие программы включает несколько ключевых этапов:

- 1) импорт файла модели в формате STL, OBJ или другом, проверка целостности и корректности геометрии;
- 2) разделение модели на слои (слоистое моделирование), что является основой для формирования траекторий печати;

3) создание путей для подачи материала, включая заполнение, периметры и поддерживающие структуры;

4) формирование управляющих команд (G-кодов) – преобразование траекторий в последовательность команд, управляющих движением экструдера, температурой, скоростью и другими параметрами.

На первом этапе осуществляется импорт файлов трехмерных моделей в заданных форматах для дальнейшей обработки и анализа. Процесс включает в себя считывание данных о вершинах, гранях и поверхностях, а также их правильное отображение в программном обеспечении. Для обеспечения точности моделирования необходимо провести предварительную проверку целостности файла, выявить возможные ошибки и несоответствия. Проверка геометрии модели включает в себя анализ ее топологии и геометрических свойств с целью выявления возможных дефектов, таких как открытые грани, самопересечения или несоответствие нормалей. Современные программные средства позволяют автоматизированно выявлять эти ошибки, что значительно повышает качество и надежность модели.

На втором этапе происходит разделение трехмерной модели на слои. Данный процесс заключается в разбиении 3D-модели на последовательные поперечные срезы, которые далее служат инструкциями для печатающего устройства [1]. Правильное определение толщины слоев обеспечивает баланс между скоростью производства и точностью детализации, а также влияет на механические свойства готового продукта. В современных системах разделение модели осуществляется с помощью специальных программных средств, учитывающих геометрические особенности и материал, что позволяет оптимизировать процессы печати и минимизировать дефекты.

Технологии автоматизированного разделения моделей на слои часто используют алгоритмы адаптивного срезания, позволяющие изменять толщину слоев в зависимости от сложности геометрии и требуемых характеристик. Такой подход способствует повышению точности передачи деталей с тонкими или сложными конструктивными элементами, а также снижает расход материала и время печати.

К наиболее распространенным методам и алгоритмам слоевого моделирования относятся:

- Метод равномерной нарезки, реализующий деление по фиксированной высоте слоя. Он основывается на делении модели по горизонтальной оси на слои одинаковой толщины, что достигается с

помощью вычисления пересечений модели с параллельными плоскостями с постоянной высотой. Данный метод обеспечивает простоту реализации и предсказуемость результатов, поскольку каждый слой имеет одинаковую толщину, что способствует стабильности процесса печати и облегчает управление скоростью подачи материала и температурой. Однако он может иметь ограничения при работе со сложными геометриями или моделями с переменной толщиной стенок, где требуется более адаптивное разбиение для достижения оптимальных характеристик качества и прочности изделия.

- Метод адаптивной нарезки, реализующий изменение высоты слоя в зависимости от сложности геометрии для оптимизации скорости и качества. В основе данного метода лежит анализ локальных особенностей модели, таких как кривизна и наличие деталей с различной степенью сложности, что позволяет автоматически менять параметры нарезки для достижения оптимального баланса между скоростью печати и точностью. Процесс реализации метода включает в себя предварительный анализ модели с помощью специальных алгоритмов, выявляющих зоны с высокой кривизной или требующие более точной детализации. После этого происходит автоматическая генерация слоев с переменной толщиной, что обеспечивает более точное воспроизведение деталей и уменьшение необходимости в последующей постобработке.

- Алгоритмы на основе топологических анализов для определения границ и внутренних структур. Данная группа алгоритмов использует структурные свойства объекта, чтобы разделить его на слои и определить оптимальные пути для нанесения материала, минимизируя дефекты и повышая качество печати. Они используют топологические инварианты, такие как числа Эйлера или гомологические группы, для классификации и сегментации внутренней структуры, в результате чего происходит автоматизированное создание слоев, учитывающих внутренние полости и сложные формы.

Третий этап основывается на использовании методов и алгоритмов, обеспечивающих эффективное покрытие поверхности и внутреннего объема модели. На данном этапе происходит определение оптимальных маршрутов, учитывающих геометрические ограничения и требования к непрерывности движения, а также заполнение пространства путями для обеспечения равномерной подачи материала [2]. При этом важно учитывать периметры и формы объектов для

минимизации затрат энергии и времени, а также предотвращения столкновений и застреваний.

Дополнительное значение имеет проектирование поддерживающих структур, которые обеспечивают устойчивость и надежность путей подачи в условиях динамической эксплуатации. Эти конструкции должны быть интегрированы в общую 3D-модель с учетом нагрузок, возникающих при движении материала, а также возможностей для обслуживания и ремонта. Разработка таких структур включает применение методов оптимизации по весу и прочности, что способствует снижению затрат и увеличению долговечности системы.

К наиболее распространенным методам и алгоритмам оптимизации путей для подачи материала относятся:

- Метод слоевого заполнения, реализующий последовательное наполнение объекта по слоям для обеспечения точного управления структурой и свойствами конечного изделия. Для каждого слоя разрабатывается оптимальный маршрут подачи материала, учитывающий геометрию модели, параметры принтера и требования к прочности и качеству поверхности. Такой подход позволяет снизить количество пересечений и пересечений путей, а также уменьшить объем излишков материала, что способствует повышению скорости и экономичности процесса.

- Алгоритмы оптимизации пути экструдера осуществляют минимизацию времени перемещения без нанесения материала, предотвращение столкновений и провисаний. Среди популярных методов выделяются алгоритмы на основе графов и жадных стратегий, такие как алгоритм Дейкстры и алгоритм A*, которые позволяют находить кратчайший путь с учетом различных ограничений. Кроме того, применяются эвристические подходы, учитывающие структуру модели и особенности слоя, что значительно повышает эффективность путей и уменьшает количество ненужных перемещений.

- Методы динамического программирования и эвристик.

Четвертый этап включает в себя преобразование траекторий в G-коды с помощью специальных алгоритмов, учитывающих особенности конкретных систем принтеров.

Таким образом, преобразование 3D-моделей в управляющие коды – это сложный и многоэтапный процесс, от которого зависит качество и эффективность 3D-печати. Современные алгоритмы включают сегментацию модели, генерацию траекторий и формирование управляющих команд с учетом особенностей конкретного принтера и материала. Развитие автоматизированных

методов и методов на основе искусственного интеллекта открывает новые возможности для повышения точности, скорости и надежности процессов 3D-печати. Перспективы дальнейших исследований связаны с разработкой универсальных алгоритмов, способных адаптироваться к различным типам моделей и оборудования, а также с интеграцией систем машинного обучения для автоматической оптимизации параметров печати.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. J.B. Zhao, R.B. Xia et al., "A computing method for accurate slice contours based on an STL model", Virtual and physical prototyping, vol.4, no.1: pp 29-37, 2009.

2. Fleming C. et al. Toolpath planning for continuous extrusion additive manufacturing // Oregon State University: Corvallis, OR, USA. – 2017.

УДК 004.9

САПРЫКИН А.Н.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРИМЕНЕНИЕ МЕТОДА ИНДЕКСИРОВАНИЯ БАЗ ДАННЫХ ДЛЯ СНИЖЕНИЯ ЗАДЕРЖЕК В ВЫСОКОНАГРУЖЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

В данной статье рассматриваются алгоритмы и структуры индексирования в базах данных, включая В-деревья, хеш-индексы, полнотекстовые индексы и индексы на основе битовых карт. Детально описываются их принципы работы, преимущества и ограничения.

В условиях стремительного роста объемов данных и увеличения числа пользователей высоконагруженных информационных систем, проблема задержек в обработке запросов становится особенно актуальной. Современные приложения, такие как онлайн-сервисы, e-commerce платформы и системы управления данными в реальном времени, требуют высокой производительности и минимального времени отклика. Задержки в этих системах могут не только негативно сказаться на пользовательском опыте, но и привести к значительным экономическим потерям для компаний. Поэтому разработка и внедрение методов, направленных на сокращение задержек, является актуальной задачей.

Одним из самых эффективных способов повышения скорости доступа к данным и уменьшения времени отклика систем является метод индексации баз данных [1, 2]. Индексы позволяют значительно ускорить выполнение запросов, уменьшая объем данных, которые необходимо просматривать при их обработке. В высоконагруженных информационных системах, где скорость обработки критически важна, применение эффективных стратегий индексации может привести к значительным улучшениям в производительности.

Индексы представляют собой структуры данных, которые позволяют ускорить доступ к записям, минимизируя количество операций чтения, необходимых для выполнения запросов. Эффективное индексирование не только улучшает скорость выполнения запросов, но и может значительно сократить ресурсы, необходимые для обработки больших объемов информации, что делает его важным инструментом для оптимизации работы высоконагруженных информационных систем.

К наиболее распространенным в настоящее время алгоритмам и структурам индексирования можно отнести следующие:

- Индексы на основе В-деревьев;
- Хеш-индексы;
- Полнотекстовые индексы;
- Индексы с использованием битовых карт.

Индексы на основе В-деревьев представляют собой самосбалансированные деревья поиска, предназначенные для работы с большими объемами данных. Основная идея В-дерева заключается в том, чтобы минимизировать количество операций ввода-вывода, что особенно важно при работе с дисковыми хранилищами, где доступ к данным значительно медленнее, чем в оперативной памяти.

Принцип работы В-деревьев основан на поддержании сбалансированной структуры данных, в которой каждый узел может содержать несколько ключей и ссылок на дочерние узлы. Это позволяет хранить большое количество данных в одном узле, что уменьшает количество необходимых обращений к диску. Каждый узел В-дерева, за исключением корня, должен содержать не менее половины от максимального количества ключей, что обеспечивает сбалансированность. При вставке или удалении ключей В-дерево автоматически перестраивается, чтобы сохранить свои свойства, что позволяет эффективно управлять изменениями в данных. Это делает В-деревья идеальными для динамических баз данных, где данные часто обновляются.

Кроме того, В-деревья обеспечивают высокую производительность за счет упорядоченного хранения ключей, что позволяет быстро находить диапазоны значений. Например, операции поиска могут быть выполнены за время, пропорциональное логарифму от количества хранимых ключей, что делает В-деревья особенно эффективными по сравнению с другими структурами данных, такими как бинарные деревья поиска. Они также поддерживают эффективные операции сканирования, что позволяет быстро извлекать большие объемы данных в заданном диапазоне. В современных СУБД В-деревья и их вариации, такие как В⁺-деревья и В*-деревья, продолжают оставаться основными инструментами для индексирования, обеспечивая быстрый и надежный доступ к информации в условиях растущих объемов данных.

Хеш-индексы работают иначе, чем В-деревья, и особенно полезны для управления большими и сложными наборами данных. Основной принцип их работы заключается в применении хеш-функции, которая преобразует ключ записи в фиксированный размер, позволяя быстро находить соответствующий элемент в наборе данных. Хеш-индексы особенно полезны для операций поиска по конкретным значениям, так как они обеспечивают доступ с постоянным временем выполнения в среднем случае, что значительно превосходит скорость работы традиционных индексов, таких как В-деревья, особенно при большом объеме данных.

Основной механизм работы хеш-индексов включает в себя создание и хранение хеш-таблицы, где каждая запись содержит хеш-значение ключа и указатель на соответствующую запись в основной таблице данных. При выполнении запроса база данных вычисляет хеш-значение для заданного ключа и использует его для доступа к нужной записи. Этот процесс минимизирует количество операций ввода-вывода и значительно ускоряет выполнение запросов, особенно в случаях, когда операции поиска превышают операции вставки и удаления в базе данных.

Тем не менее, у хеш-индексов есть и свои ограничения. Например, они неэффективны для диапазонных запросов, поскольку хеш-функция распределяет данные по фиксированным адресам, и последовательный доступ к данным становится затруднительным. Кроме того, коллизии – ситуация, когда две разные записи имеют одинаковое хеш-значение – могут привести к снижению производительности, так как в этом случае требуется дополнительная обработка для разрешения конфликтов.

Полнотекстовые индексы применяются для оптимизации поиска и обработки текстовой информации в базах данных. Их основная задача заключается в ускорении поиска по текстовым полям. В отличие от традиционных индексов, которые, как правило, работают с фиксированными полями, полнотекстовые индексы обеспечивают возможность быстрого поиска по отдельным словам, фразам и даже по синонимам в неструктурированном или полуструктурированном текстовом контенте. Это достигается за счет использования специальных структур данных, таких как обратные индексы, которые связывают каждое слово с документами, содержащими это слово, что существенно сокращает время на выполнение поисковых запросов.

Принципы работы полнотекстовых индексов основываются на процессах токенизации, стемминга и фильтрации стоп-слов. Токенизация представляет собой разбиение текстового потока на отдельные слова или токены, что позволяет системе эффективно обрабатывать текстовые данные. Стемминг, в свою очередь, сводит слова к их корням, что помогает учесть различные формы одного слова и улучшить результаты поиска. Фильтрация стоп-слов позволяет исключить общеупотребительные слова (такие как "и", "в", "на"), которые не несут значимой информации для поиска, тем самым уменьшая размер индекса и повышая его эффективность.

Кроме того, современные системы управления базами данных часто интегрируют механизмы для обработки полнотекстовых запросов, что включает в себя поддержку различных операторов поиска, таких как AND, OR и NOT, а также возможность использования фразового поиска и поиска по регулярным выражениям. Это расширяет возможности пользователей, позволяя им формулировать более сложные и точные запросы.

Индексы с использованием битовых карт представляют собой массивы битов, где каждый бит соответствует определенному значению в наборе данных. Такая структура позволяет быстро проверять наличие или отсутствие определенных значений, что значительно ускоряет операции поиска и фильтрации данных.

Принцип работы индексов на основе битовых карт заключается в том, что для каждого уникального значения в столбце создается соответствующая битовая карта. Если значение присутствует в строке таблицы, соответствующий бит устанавливается в единицу, в противном случае – в ноль. Такой подход позволяет быстро выполнять операции, такие как объединение и пересечение множеств, что особенно полезно в сценариях, где необходимо обрабатывать запросы с множественными условиями. Например, при выполнении запроса с

условиями на несколько столбцов, можно использовать битовые операции для быстрого нахождения пересечений между битовыми картами, что значительно снижает время выполнения запроса по сравнению с традиционными методами индексации.

Однако использование битовых карт также имеет свои ограничения. Во-первых, битовые карты могут занимать значительное количество памяти, особенно в случаях, когда количество уникальных значений велико. Это может привести к неэффективному использованию ресурсов и увеличению времени доступа к данным. Во-вторых, работа с битовыми картами требует дополнительных вычислений для обновления и поддержания индекса, что может повлиять на производительность системы в условиях частых изменений данных. Тем не менее, при правильном использовании и в подходящих сценариях битовые карты могут существенно улучшить производительность баз данных, обеспечивая быстрый доступ к информации и эффективную обработку запросов.

В данной статье была рассмотрена проблема задержек в высоконагруженных информационных системах и предложено применение метода индексирования баз данных как одного из эффективных решений для их минимизации. Анализ существующих подходов к индексированию показал, что использование различных типов индексов позволяет значительно ускорить процессы поиска и выборки данных. Каждый из перечисленных методов имеет свои уникальные преимущества и ограничения, что делает их применение зависимым от специфики задач и требований к системе.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Bhattacharya, A. Fundamentals of database indexing and searching / A. Bhattacharya. – Boca Raton [etc.]: CRC Press: Taylor & Francis Group, 2016. – 252 с.
2. Winand, M. SQL Performance Explained: Everything Developers Need to Know about SQL Performance / M. Winand. – M. Winand, 2012. – 196 с.

УДК 004.4

СЕНЯКИН К.Д., ТИХОНОВ А.А.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

МОНИТОРИНГ СИСТЕМ С ПОМОЩЬЮ PYTHON И ZABBIX

В данной работе рассматриваются перспективные средства мониторинга систем на основе Python-скриптов и средств Zabbix, предоставляющих возможность получать актуальную информацию о состоянии систем.

Современные IT-инфраструктуры представляют собой сложные распределённые системы, включающие физические серверы, виртуальные машины, контейнеры, облачные сервисы, микросервисные архитектуры и IoT-устройства. Они характеризуются:

- высокой динамичностью (автомасштабирование, оркестрация Kubernetes, serverless-вычисления);
- гетерогенностью (разные ОС, сети, протоколы, производители оборудования);
- критичностью отказоустойчивости (требования к SLA 99.9% и выше);
- большими объёмами данных (логи, метрики, трассировки).

Они требуют гибких и масштабируемых решений для мониторинга, способных адаптироваться к разнородным системам и нестандартным метрикам. Мониторинг систем нужен для: предотвращения сбоев – выявления аномалий (например, перегрузки CPU или утечки памяти) до их перерастания в инцидент; обеспечения производительности – анализа времени ответа API, загрузки сети и дискового ввода-вывода; оптимизации ресурсов сбора данных об использовании CPU/RAM/дисков для правильного масштабирования; соответствия требованиям безопасности – контроля подозрительной активности (сканирование портов, аномальные запросы); автоматизации реагирования - интеграция с ITSM (Jira, ServiceNow) и системами алертинга (Telegram, PagerDuty).

Одним из наиболее эффективных инструментов в этой области является Zabbix – open-source система, предоставляющая широкие возможности для сбора и анализа данных. Однако в ряде случаев встроенных механизмов оказывается недостаточно, и тогда стоит прибегнуть к UserParameters, позволяющим интегрировать пользовательские скрипты для расширения функциональности мониторинга.

Python, благодаря своей простоте, богатой экосистеме библиотек и кросс-платформенности, становится наиболее подходящим выбором для реализации таких решений. Его синтаксис и модульность ускоряют разработку скриптов, а поддержка работы с API и системными метриками делает его одним из лучших в контексте взаимодействия с Zabbix.

В данной статье рассматривается практический аспект использования Python в связке с Zabbix через механизм UserParameters.

Анализируются преимущества выбранных технологий, приводятся примеры настройки кастомных метрик. Результаты исследования демонстрируют, что сочетание Zabbix и Python позволяет создавать эффективные системы мониторинга, адаптированные под специфические требования инфраструктуры.

Сначала объясним, почему были выбраны именно Zabbix и Python.

Zabbix – это мощная open-source система мониторинга, которая предоставляет широкие возможности для сбора, анализа и визуализации метрик. Выбор Zabbix обусловлен следующими преимуществами:

- гибкость и расширяемость – Zabbix поддерживает UserParameters, позволяя добавлять собственные метрики через скрипты, что критически важно для мониторинга нестандартных систем;

- поддержка множества протоколов - встроенная поддержка SNMP, HTTP, JMX, IPMI и других протоколов делает Zabbix универсальным решением;

- масштабируемость – возможность распределённого мониторинга (прокси, ноды) позволяет использовать Zabbix в крупных инфраструктурах;

- активное сообщество и документация – Zabbix имеет подробную документацию и развитое сообщество, что упрощает решение сложных задач.

Таким образом, Zabbix был выбран из-за его гибкости, поддержки кастомных метрик и способности адаптироваться под специфические требования мониторинга.

Python – один из наиболее популярных языков для автоматизации и работы с данными. Его использование в связке с Zabbix оправдано по следующим причинам:

- простота и читаемость кода – Python позволяет быстро разрабатывать скрипты для сбора метрик, что ускоряет настройку UserParameters;

- богатая экосистема библиотек – доступ к системным метрикам (psutil), API (requests, pyzabbix), парсингу данных (json, xml) и другим инструментам делает Python идеальным для интеграции;

- кросс-платформенность – Python работает на Linux, Windows и других ОС, что позволяет использовать одни и те же скрипты в гетерогенных средах;

- поддержка Zabbix API – Библиотеки вроде py-zabbix и zabbix-api упрощают взаимодействие с Zabbix программно.

Использование Python значительно сокращает время разработки агентов мониторинга и обеспечивает лёгкую поддержку кастомных решений.

Теперь поговорим о совмещении средств Zabbix с Python-скриптами. Здесь нам понадобится UserParameters — это механизм Zabbix, позволяющий выполнять пользовательские скрипты и возвращать их вывод как метрики. UserParameters позволяют выполнять Python-скрипты напрямую через Zabbix Agent и возвращать их результат как метрику.

В качестве примера можно привести следующий код:

```
import sys
output: str = ""
for i in range(1, len(sys.argv)):
    output+=sys.argv[i]
    if i < len(sys.argv):
        output+=" "
print(output)
```

В приведённом примере представлен аналог команды echo в Linux, написанный на языке Python. Этот скрипт добавляется в конфигурационный файл агента (/etc/zabbix/zabbix_agentd.conf) как параметр:

```
UserParameter=test_param[*],python3
/etc/python_scripts/python_echo.py $1 $2
UserParameter=test_param_py[*],python3
/etc/python_scripts/my_test.py
```

Далее требуется перезагрузить Zabbix Agent:

```
systemctl restart zabbix-agent
```

Теперь настроим его подключение в Zabbix (см. рисунок 1).

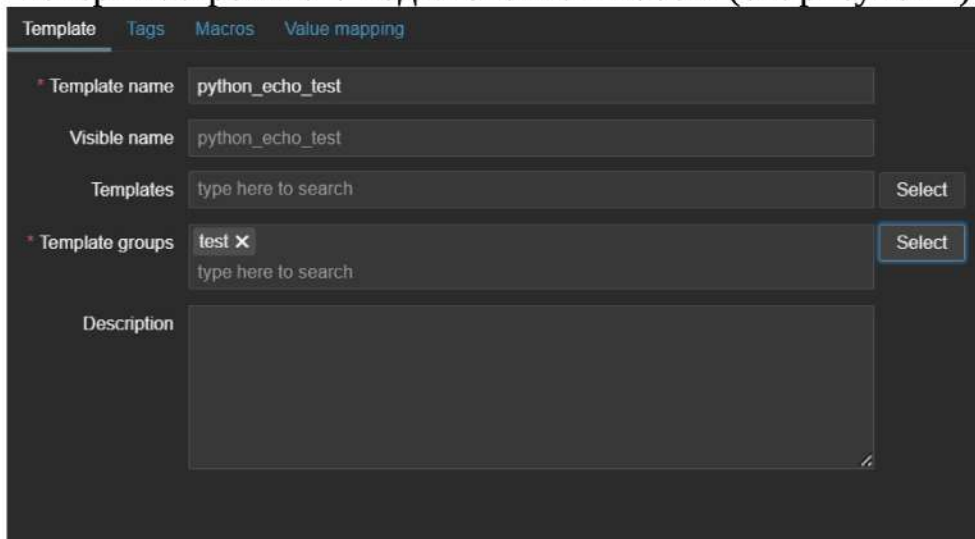


Рисунок 1 – Создание шаблона в веб-интерфейсе Zabbix
Выставляем в массив принимаемых значений, которые будут

приняты аргументы командной строки тестовые значения eins, zwei (см. рисунок 2). Эти значения приходят в систему в качестве результата мониторинга (см. рисунок 3).

The screenshot shows the 'Create item' form in the Zabbix web interface. The form is titled 'Item' and has tabs for 'Tags' and 'Preprocessing'. The fields are as follows:

- Name:** echo_py
- Type:** Zabbix agent
- Key:** test_param['eins','zwei']
- Type of information:** Text
- Update interval:** 15s
- Custom intervals:** A table with columns Type, Interval, Period, and Action. It contains one row: Flexible, Scheduling, 50s, 1-7,00.00-24.00. There are 'Add' and 'Remove' buttons.
- Timeout:** Global, Override, 3s
- History:** Do not store, Store up to, 31d
- Populates host inventory field:** -None-
- Description:** (empty text area)

Рисунок 2 – Создание айтема в веб-интерфейсе Zabbix

Host	Name ▲	Last check	Last value
██████████	echo_py	2s	eins zwei

Рисунок 3 – Результат добавления айтема в систему

Возвращаясь к теме мониторинга состояния системы можно использовать библиотеку os языка программирования Python. Как пример, мы можем узнать тип операционной системы с помощью следующего скрипта:

```
import os
# name of os will be written
print(os.name)
```

Данный скрипт при настройке UserParameters будет возвращать значение posix.

В данной работе рассмотрены ключевые аспекты интеграции Python и Zabbix для мониторинга сложных IT-инфраструктур. Показано, что связка этих технологий позволяет:

- гибко адаптировать мониторинг под нестандартные требования;
- автоматизировать сбор и анализ метрик с помощью скриптов;
- обеспечивать высокую производительность и отказоустойчивость.

Приведенные примеры демонстрируют, что использование User Parameters и Zabbix Sender с Python-скриптами значительно расширяет возможности Zabbix, особенно в гетерогенных средах.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Официальная документация Zabbix. – URL: <https://www.zabbix.com/documentation> (дата обращения: 18.04.2025).
2. Python для сетевых инженеров. – URL: <https://pyneng.readthedocs.io/> (дата обращения: 18.04.2025).
3. Мониторинг Kubernetes с помощью Zabbix. – URL: <https://habr.com/ru/articles/786454/> (дата обращения: 18.04.2025).
4. Якубов В.В. «Zabbix 6: Полное руководство». – СПб.: БХВ-Петербург, 2023. – 480 с.
5. Петров К.А., Сидоров В.М. «Анализ методов мониторинга распределённых систем» // Журнал «Информационные технологии». – 2023. – № 8. – С. 45–52.
6. Корнеев М. «Python для DevOps». – М.: Питер, 2023. – 250 с.

УДК 004.92

СИДОРОВ А.С.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЦВЕТОВЫХ МОДЕЛЕЙ В ОБЛАСТИ ВЕБ-ДИЗАЙНА

В статье проводится сравнительный анализ цветовых моделей, представленных в черновике кандидата в рекомендацию W3C “CSS Color Module Level 4” и реализованных в последних версиях наиболее распространённых веб-браузеров, с учётом особенностей работы с цветом веб-дизайнерами, а также приводятся рекомендации по выбору цветовых моделей.

Введение

В современном веб-дизайне цвет играет ключевую роль в формировании пользовательского опыта, влияя на восприятие информации и привлекательность веб-сайтов. На данный момент в наиболее новых версиях пяти самых популярных веб-браузеров в мире (Chrome, Safari, Edge, Samsung Internet, Firefox)[1] реализованы следующие цветовые модели: RGB в цветовом пространстве sRGB и её цилиндрические представления HSL, HWB; Lab, Oklab и их

цилиндрические разновидности Lch, Oklch в одноимённых цветовых пространствах; RGB в предопределённых пространствах (sRGB, DCI-P3, Rec.2020 и т.д.) и XYZ в пространстве CIE XYZ с осветителями D50, D65 [8]. Каждая из них обладает своими особенностями, в том числе используемым цветовым пространством и способом представления цвета, поэтому выбор конкретной цветовой модели для работы веб-дизайнера может быть затруднителен. Сравнительный анализ этих моделей позволит веб-дизайнерам принимать более обоснованные решения в условиях постоянного развития дисплеев устройств и развития Интернета, обеспечивая при этом визуальное единство, доступность интерфейсов и наиболее эффективное использование доступных цветовых пространств.

Основные определения

Цвет (color) – это определение (числовое или текстовое) визуального восприятия человеком света или физического объекта, освещенного светом.

Светлота (lightness, L^*) – визуальное восприятие яркости (luminance, L) объекта человеком. В то время как яркость — это линейное, физическое измерение света, светлота — это линейное предсказание человеческого восприятия этого света.

Данное различие имеет смысл, потому что восприятие яркости человеческим зрением нелинейно по отношению к свету: оно более чувствительно к изменениям в тёмных тонах, чем в ярких. В качестве демонстрационного примера этого явления на рисунке 1 представлены линейные градиенты от чёрного (`rgb(0 0 0)`) до белого (`rgb(255 255 255)`) в пространстве sRGB. Для верхнего градиента в качестве интерполирующего цветового пространства используется линейное по яркости пространство CIE XYZ, а для нижнего — равноконтрастное пространство Oklab.

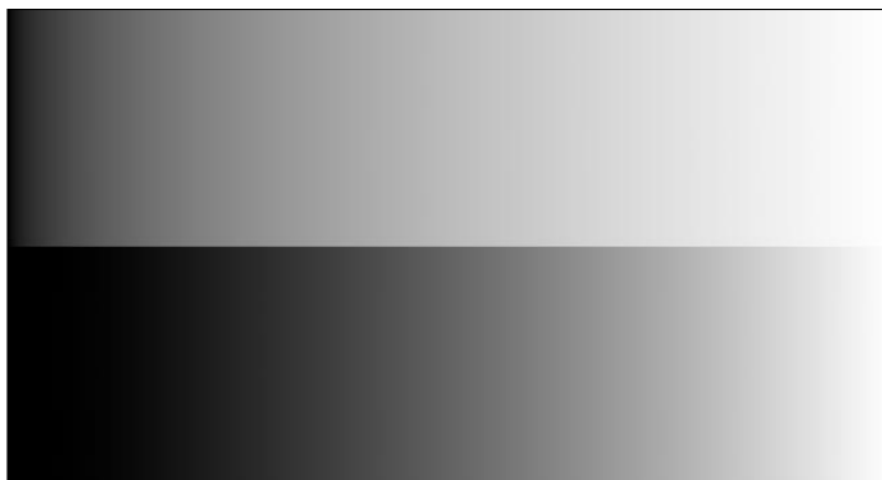


Рисунок 1 – Изменение яркости в пространствах CIE XYZ (сверху) и Oklab (снизу)

Цветность (chromacity) — измерение света, не включающее его яркость. Состоит из цветового тона (hue) и насыщенности (chroma). Цветность аддитивна, благодаря чему можно хорошо предсказывать цветность (но не яркость) результата смешения двух цветов.

Цветовая модель (color model) — это абстрактная математическая модель, описывающая способ представления цветов в виде кортежей чисел (например, тройки в RGB). Для определения абсолютных, а не относительных значений цветов цветовая модель может иметь функцию отображения на какое-либо эталонное цветовое пространство.

Цветовое пространство (color space) — это организация цветов в соответствии с цветовой моделью таким образом, что для любого цвета в этом цветовом пространстве существует четкое, объективно-измеримое значение.

Абсолютное цветовое пространство (absolute color space) — это цветовое пространство, в котором цвета однозначно определены колориметрически без привязки к внешним факторам (в том числе к характеристикам дисплеев), например, CIE XYZ, sRGB, Rec. 2020.

Эталонное цветовое пространство (reference color space) — это цветовое пространство, основанное на восприятии цветов стандартным колориметрическим наблюдателем (CIE standard observer), например, CIE RGB, CIE XYZ, OLAB. Они определяют спектр всех видимых человеком цветов, и в том числе те цвета, которые невозможно на данный момент отобразить ни одним устройством.

Обычно для определения абсолютного цветового пространства необходимо:

4. Выбрать используемую цветовую модель;
5. Выбрать эталонное цветное пространство, на которое впоследствии будут отображаться численные выражения цвета выбранной цветовой модели;
6. Используя эталонное цветное пространство, определить значения цветности для первичных цветов модели. Это в свою очередь определит цветовой охват (gamut) пространства;
7. Выбрать стандартный осветитель (standard illuminant), т. е. эталонный белый цвет (обычно D50 или D65);
8. Определить функцию для перевода из выбранной цветовой модели в выбранное ранее эталонное цветное пространство;
9. При необходимости определить функцию для гамма-коррекции яркости.

Равноконтрастное цветовое пространство (uniform color space, UCS) — это цветовое пространство, в котором при фиксированных внешних условиях (освещение окружающей среды и прочие) равные расстояния между цветами соответствуют примерно равным различиям, воспринимаемым средним человеческим наблюдателем. Цель такого пространства — попытаться скорректировать нелинейную природу человеческого восприятия цвета, сделав сравнение, манипулирование и количественную оценку цветов более интуитивной.

Из-за комплексности человеческого восприятия цвета на данный момент ещё нет абсолютно точного и универсального перцептивно равномерного цветового пространства. Существующие UCS хорошо справляются с предсказанием только в некоторых областях, например, CIE Lab визуально равномерно предсказывает изменения яркости, но имеет проблемы с визуально неравномерным предсказыванием оттенков.

Работа с цветом в веб-дизайне

Этапы работы веб-дизайнера с цветом могут включать в себя:

1. Анализ бренда и его целевой аудитории;
2. На основе анализа выбор базового цвета и опционально вторичного;
3. Построение цветовой палитры на основе одного или двух выбранных цветов, которая включает в себя более светлые и более тёмные их разновидности, а также насыщенные акцентные и ненасыщенные нейтральные цвета. При построении палитры и выборе цветов в целом учитывается необходимый контраст между ними согласно их функциональности по стандарту WCAG или алгоритму APCA;
4. Опциональное повторение пунктов 2 и 3 для создания различных тем веб-сайта, например, тёмной или серой;
5. Использование выбранных цветов для элементов веб-сайта: кнопок, иконок, ссылок и прочих.

На основе перечисленных этапов работы веб-дизайнера можно выделить следующие требования к цветовым моделям:

1. По представлению цвета в модели человек должен легко определять внешний вид этого цвета;
2. Манипулирование цветом, т. е. изменение воспринимаемой яркости, цветности и оттенка, должно соответствовать восприятию человеком света (равноконтрастным) и быть относительно простым;

3. Модель должна быть распространена среди инструментов для работы с цветом, в первую очередь селекторами цвета (color picker), и среди веб-браузеров.

Рассмотрим цветовые модели, представленные в CSS Color Module Level 4 и доступные в наиболее новых версиях пяти самых популярных веб-браузеров, относительно сформулированных требований.

Обзор цветовых моделей

RGB — аддитивная цветовая модель, в которой цвета образуются смешением в различных пропорциях «красным», «зелёным» и «синим» цветами. Конкретные значения первичных цветов определяются на основе координат эталонного цветового пространства, обычно CIE XYZ. В CSS3 по умолчанию все RGB-цвета указываются в пространстве sRGB. CSS Color Module Level 4 допускает возможность использования этой модели для различных RGB-пространств: sRGB, DCI-P3, Rec. 2020 и прочих предопределённых.

Зная правила сложения цветов первичных цветов модели RGB, можно приблизительно определить цветность получаемого цвета, но не его светлоту, так как на яркость цвета в модели влияют все три компонента цвета в неравных пропорциях: компонента зелёного больше всего влияет на яркость, компонента синего — меньше всего. Из-за этого проблематично точно определять более светлые или более тёмные оттенки с одинаковой цветностью базового цвета. Кроме того, ни одно RGB пространство не является равноконтрастным.

Примеры неинтуитивности RGB модели представлены на рисунке 2. На нём изображены цветные квадраты в пространстве sRGB со значениями их цветов в моделях RGB и Oklch. В качестве эталонной модели была выбрана Oklch, потому что её пространство наиболее равноконтрастно (визуально равномерно) среди доступных в CSS Color Module Level 4. Первый квадрат имеет белый цвет (100% светлоты), а второй визуально в 2 раза темнее его (50% светлоты). Видно, что RGB-значения второго квадрата меньше примерно в 2.57 раз, а не в 2

rgb(255 255 255) oklch(1 0 0)	rgb(99 99 99) oklch(0.5 0 0)	rgb(210 132 156) oklch(0.7 0.1 0)	rgb(144 73 97) oklch(0.5 0.1 0)
rgb(210 132 156) oklch(0.7 0.1 0)	rgb(231 114 155) oklch(0.7 0.15 0)	rgb(210 132 156) oklch(0.7 0.1 0)	rgb(204 143 92) oklch(0.7 0.1 60)

Рисунок 2 – Неинтуитивность и неравноконтрастность модели RGB

раза. На паре квадратов в правом верхнем углу видно, что уменьшение светлоты при одной и той же цветности на 20% изменяет все итоговые RGB-значения в труднопредсказуемом для человека порядке. Аналогично происходит при увеличении только цветности (левая нижняя пара) и только при сдвиге оттенка (правая нижняя пара).

На данный момент пространство sRGB является стандартом для всех цифровых отображающих устройств и Интернета. Следовательно, модель RGB является базовой для всех этих устройств, а также веб-браузеров. Поэтому в любом инструменте-приложении, каким-либо образом связанным с цветом, есть возможность работы с данной моделью в пространстве sRGB.

В CSS3 есть возможность укорачивать RGB-запись цвета в пространстве sRGB посредством записи значений красного, зелёного и синего в шестнадцатеричной системе счисления (hex-код, #rrggbb или #rgb). Цвет в такой записи выглядит компактнее, однако читается ещё труднее обычной RGB-записи.

HSL — наряду с RGB является одной из наиболее популярных цветовых моделей. В CSS3 HSL является цилиндрическим представлением модели RGB в пространстве sRGB, поэтому неравноконтрастность sRGB затрагивает и её. Она представляет RGB-цвет в виде оттенка (H, hue), насыщенности (S, saturation) и светлоты (L, lightness).

Цвет в значениях HSL представить проще, чем в RGB, но точно манипулировать цветом не представляется возможным. Например, изменение оттенка влияет на реальное значение светлоты цвета, а сама светлота не распределена визуальным образом равномерно.

Возможность использования модели HSL совместно с RGB моделью есть в большинстве инструментов для работы с цветами, а веб-браузеры, поддерживающие стандарт CSS3, обязаны поддерживать эту модель.

HWB — менее популярная цветовая модель, принцип которой схож с HSL. Она также представляет собой цилиндрическую версию RGB модели со всеми её недостатками, перечисленными выше. Описывает цвет в виде оттенка (H, hue), степени белизны (W, whiteness) и степени черноты (B, blackness).

По словам создателей, модель позволяет представлять цвета интуитивнее, чем HSL: чтобы затенить цвет или его оттенок, необходимо «очернить» его; чтобы осветлить оттенок, необходимо «отбелить» его [9]. На рисунке 1 (а и б) представлены цилиндрические графики моделей HSL и HWB и их прямые представления: график оттенка и светлоты HSL (с) и график оттенка и черноты HWB (d) в

зависимости от цветности (диапазона значений RGB), а не насыщенности (превышения цветности над максимальным значением цветности для данного среза). По ним действительно можно заметить, что в модели HWB легче получать приблизительные тёмные и светлые оттенки цвета, чем в HSL.

Поддержка HWB в веб-браузерах началась появляться в версиях примерно от 2021 года, некоторые инструменты для работы с цветом также поддерживают эту модель, однако по сравнению с HSL их меньше в целом.

На рисунке 3 показаны проблемы цветковых моделей HSL и HWB аналогичным рисунку 4 образом.

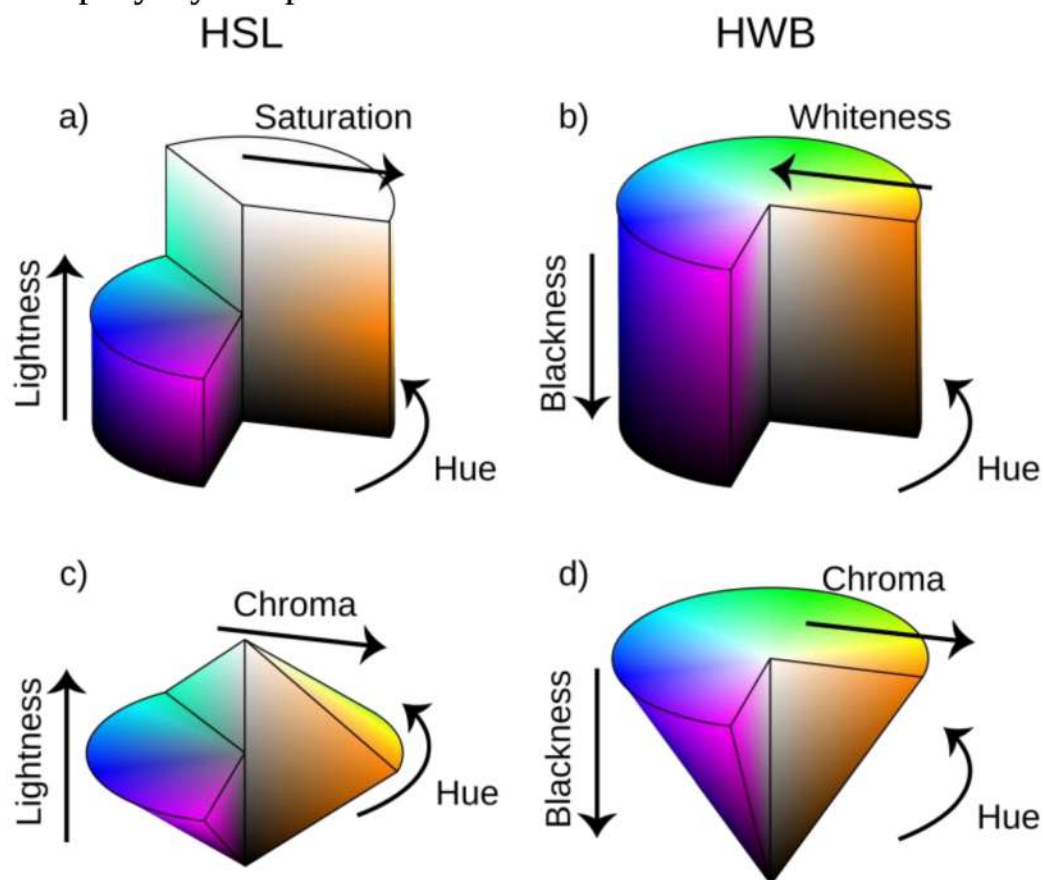


Рисунок 3 — Модели HSL и HWB

<code>hsl(0 0% 100%)</code> <code>hwb(0 100% 0%)</code> <code>oklch(1 0 0)</code>	<code>hsl(0 0% 39%)</code> <code>hwb(0 39% 61%)</code> <code>oklch(0.5 0 0)</code>	<code>hsl(341 46% 67%)</code> <code>hwb(341 52% 18%)</code> <code>oklch(0.7 0.1 0)</code>	<code>hsl(340 33% 43%)</code> <code>hwb(340 29% 44%)</code> <code>oklch(0.5 0.1 0)</code>
<code>hsl(341 46% 67%)</code> <code>hwb(341 52% 18%)</code> <code>oklch(0.7 0.1 0)</code>	<code>hsl(340 71% 68%)</code> <code>hwb(339 45% 9%)</code> <code>oklch(0.7 0.15 0)</code>	<code>hsl(341 46% 67%)</code> <code>hwb(341 52% 18%)</code> <code>oklch(0.7 0.1 0)</code>	<code>hsl(27.3 52% 58%)</code> <code>hwb(27.3 36% 20%)</code> <code>oklch(0.7 0.1 60)</code>

Рисунок 4 — Неравноконтрастность HSL и HWB

CIELAB (LAB, Lab, $L^*a^*b^*$) – цветовая модель относительно равноконтрастного, оппонентного и одноимённого цветового пространства. Оппонентный процесс — это теория цвета, утверждающая, что зрительная система человека интерпретирует информацию о цвете, обрабатывая сигналы от фоторецепторных клеток в антагонистической манере. Теория оппонентного процесса предполагает, что существует три оппонентных канала, каждый из которых включает в себя противоположную пару цветов: красный против зеленого, синий против желтого и черный против белого (яркость). Соответственно, LAB представляет цвет в виде светлоты (L^*), количеством зелёного и красного цветов (a^*), количеством синего и жёлтого цветов (b^*).

С помощью данной модели можно визуально равномерно и интуитивно изменять светлоту цвета, однако представление цветности с помощью значений a^* и b^* неинтуитивно. Кроме того, LAB позволяет определять цвета, выходящие за пределы стандартного пространства sRGB, что ещё больше усложняет использование модели. Тем не менее, по сравнению с HWB LAB является более распространённой моделью.

Модель была названа относительно равноконтрастной из-за следующих её проблем:

1. В синей области (в LCH Hue между 270° и 330°) визуальный оттенок отличается от того, что предсказывает LCH. На графике LCH слева на рисунке 5 набор синих цветов с одинаковым оттенком и разной цветностью, которые должны лежать на прямой линии от нейтральной оси, вместо этого образуют кривую. Другими словами, при постепенном уменьшении цветности насыщенный синий становится заметно фиолетовее. Более равноконтрастное пространство Oklch (на графике справа) не имеет такой проблемы;

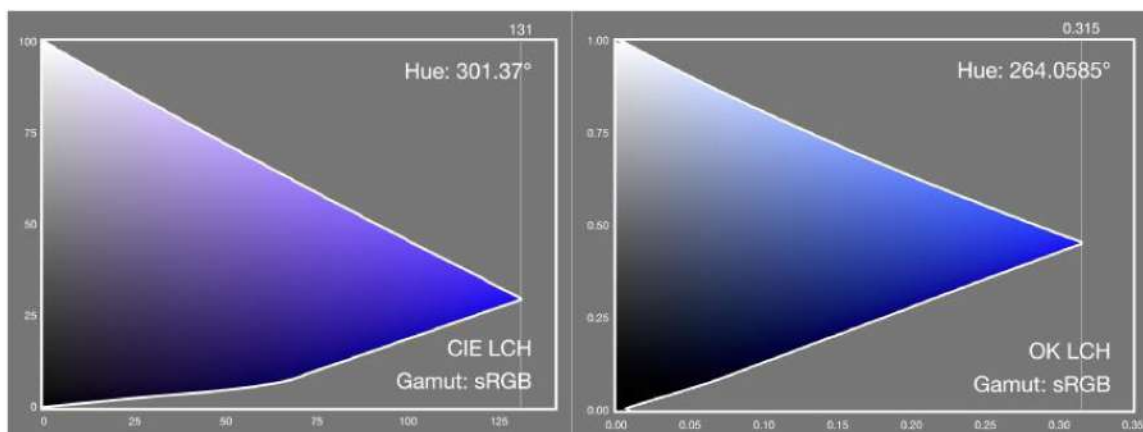


Рисунок 5 – Сдвиг оттенка в области синего цвета в LCH

2. Хотя оттенки в LCH в целом равномерно распределены (и гораздо лучше, чем в HSL или HWB), равномерность не идеальна;

3. Для цветов насыщенных цветов изменения в цветности менее заметны, чем для более нейтральных цветов. [7]

CIELCH (LCH, Lch) — цилиндрическая вариация CIELAB. Наследует все её проблемы и преимущества, при этом представляет цвет в более удобном для человека виде: светлота (L^*), цветность (C, chroma), оттенок (H). Это позволяет относительно точно и интуитивно понятно манипулировать цветами, однако не без недостатков, описанных выше. Благодаря этому в веб-среде она распространена больше LAB.

Oklab — улучшенная в плане равноконтрастности версия CIELAB. На данный момент является наиболее равноконтрастной среди всех моделей, представленных в CSS Color Module Level 4 [3]. Позволяет точно и интуитивно понятно манипулировать цветами при учетывании возможности определения цвета вне пространства sRGB.

Аналогично LAB, представление цветности значениями a^* и b^* неинтуитивно, при этом в Oklab изменение яркости происходит визуально равномернее, как показано на рисунке 6, и она не имеет проблем LAB, описанных выше.

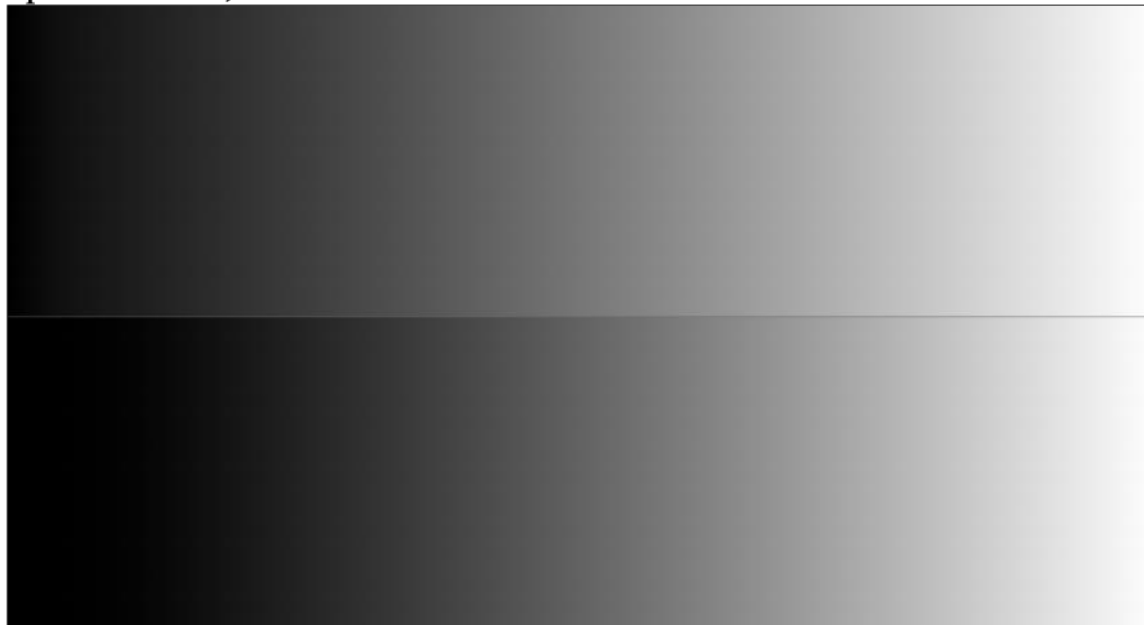


Рисунок 6 – Изменение яркости в пространствах CIE LAB (сверху) и OK LAB (снизу)

Описание модели было опубликовано в 2020 году [OKLAB], и последние версии веб-браузеров уже поддерживают её. Но её цилиндрическая версия Oklch более распространена из-за более интуитивного представления цветности.

Oklch — цилиндрическая версия Oklab, представляет цвет аналогично CIE LCH и имеет наилучшую равномерность пространства Oklab. Как и LCH, в веб-среде она больше распространена.

XYZ — аддитивная цветовая модель эталонного цветового пространства CIE XYZ, основанного на цветовом пространстве CIE RGB. В CSS она масштабирована таким образом, что эталонный белый (D50 для xyz - d50 или D65 для xyz - d65 и xyz) имеет яркость (компонент Y) 1.0. В то время как CIE RGB описывает цвета смешиванием трех монохроматических цветов с длинами волн 700 нм (красный), 546.1 нм (зеленый) и 435.8 нм (синий) в разных количествах (в том числе отрицательных), тристимульные значения XYZ образуются линейной трансформацией значений CIE RGB:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2.7689 & 1.7517 & 1.1302 \\ 1.0000 & 4.5907 & 0.0601 \\ 0.0000 & 0.0565 & 5.5943 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Исходя из этого можно сказать, что даже приблизительно определить численно представленный в модели XYZ цвет затруднительно. Соответственно, манипулировать цветами в этой модели также затруднительно.

Заключение

В таблице 1 в кратком виде представлено сравнение цветовых моделей по трём требованиям, описанным в разделе «Работа с цветом в веб-дизайне».

Таблица 1 — Сравнение цветовых моделей CSS Color Module 4

Цветовая модель	Интуитивное представление цвета	Точность и простота манипулирования цветом	Распространённость в веб-среде
RGB	–	–	+
HSL	±	–	+
HWB	+	–	±
LAB	–	±	±
LCH	+	±	+
Oklab	–	±	±
Oklch	+	+	±
XYZ	–	–	–

В итоге можно сказать, что для работы с цветами в целом следует использовать цветовую модель Oklch, а в целях совместимости со старыми веб-браузерами в итоговом CSS-коде либо использовать hex-код для определённых моделью Oklch цветов, либо использовать полифилы, которые реализуют поддержку модели Oklch для более старых веб-браузеров и позволяют в самом CSS-коде использовать цвета в модели Oklch.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Браузеры – Статистика использования. // Similarweb [Электронный ресурс]. – URL: <https://www.similarweb.com/ru/browsers/> (дата обращения: 16.03.2025).

2. English-Russian vocabulary of international lighting [Электронный ресурс]. – URL: https://lighting_equipment_en_ru.academic.ru/ (дата обращения: 01.04.2025).
3. Bottosson B. Oklab – A perceptual color space for image processing [Электронный ресурс]. – URL: <https://bottosson.github.io/posts/oklab/> (дата обращения: 16.04.2025).
4. Color Appearance Models. 3rd ed. – Hoboken: Wiley, 2018. – 424 p.
5. Colorimetry: Fundamentals and Applications. 3rd ed. – Chichester: Wiley, 2007. – 395 p.
6. Colorimetry. CIE 1976 L*a*b* Colour Space. – London: BSI, 2004. – 52 p.
7. CSS Color Module Level 4. // W3C: [Электронный ресурс]. – URL: <https://www.w3.org/TR/css-color-4/> (дата обращения: 10.04.2025).
8. CSS color_value. // MDN Web Docs: [Электронный ресурс]. – URL: <https://developer.mozilla.org/en-US/docs/Web/CSS/colorvalue> (дата обращения: 21.03.2025).
9. Smith, Alvy & Lyons, Eric. HWB: A more intuitive hue-based color model // Journal of Graphics Tools. – 1996. – Vol. 1.
10. Material Design [Электронный ресурс]. – URL: <https://m2.material.io/design> (дата обращения: 20.03.2025).

УДК 004.415.22

СОЛОМОНОВ Н.Д.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ GPU В ОБУЧЕНИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Статья посвящена анализу архитектуры GPU и их роли в ускорении обучения искусственного интеллекта, включая сравнение с CPU, математические модели производительности и перспективные технологии. Рассматриваются ключевые инструменты для работы с тензорными ядрами и ограничения масштабируемости вычислений.

Почему GPU стали двигателем прогресса в ИИ

В 2012 году произошло знаковое событие: нейронная сеть AlexNet, обученная на GPU, с огромным отрывом (top-5 error 15.3% против 26.2% у второго места) победила в конкурсе ImageNet [1]. Этот

момент стал поворотным пунктом в развитии искусственного интеллекта. Оказалось, что игровые видеокарты могут ускорять сложные математические вычисления в десятки раз по сравнению с традиционными процессорами.

Современные модели ИИ требуют обработки колоссальных объемов данных. Например, обучение GPT-3 с его 175 миллиардами параметров на CPU заняло бы десятилетия. Благодаря GPU этот процесс сократился до нескольких недель. Такое ускорение стало возможным благодаря принципиальному различию в архитектуре:

- *CPU (центральный процессор)* – «универсальный солдат», отлично справляющийся с последовательными задачами.

- *GPU (графический процессор)* – «армия специалистов», где тысячи ядер одновременно выполняют однотипные операции.

1. *Архитектура GPU как устроены эти «вычислительные монстры»*

1.1. *Основные компоненты современного GPU.* CU – это компоненты GPU, оптимизированные для параллельных вычислений, содержащие упрощённые ядра для обработки данных, таких как матричные операции в нейросетях. AI-ускорители (например, *NVIDIA Tensor Cores*) могут включать тысячи таких CU для ускорения обучения и вывода моделей.

Ядра: массовый параллелизм для AI. В отличие от CPU, GPU-ядра проще, но их количество (тысячи) позволяет эффективно выполнять:

- Матричные умножения
- Пакетную обработку данных
- Активационные функции (через SIMD)

Это важно для компьютерного зрения (CV) и обработки естественного языка (NLP), где требуется параллельная обработка векторов.

Иерархия памяти для оптимизации AI. GPU используют многоуровневую память для минимизации задержек при работе с тензорами:

- *Глобальная память* – хранение весов модели и входных данных

- *Совместная память* – обмен промежуточными результатами

- *Регистры/кеш* – для ускорения операций в потоках

Потоковые мультипроцессоры (SMs) и тензорные ядра. В архитектуре NVIDIA SM объединяют:

- *CUDA-ядра* – для общих вычислений

– *Tensor Cores* – специализированные блоки для операций смешанной точности (FP16/FP32), ускоряющие обучение трансформеров, GAN и других моделей

– *RT Cores* – для AI-рендеринга в реальном времени

Блоки обработки текстур (TMU) в AI. TMU, хотя изначально предназначены для графики, используются в AI для:

– Быстрой загрузки многомерных данных

– Интерполяции в задачах сегментации

ROPs и постобработка в AI. ROPs в AI могут участвовать в:

– Генерации изображений

– Сглаживании выходов нейросетей

1.2. Почему GPU так хорошо подходят для ИИ?

Два этапа работы нейросетей

Работа с глубокими нейронными сетями включает два ключевых этапа:

- *Обучение (Training)* – настройка параметров сети на размеченных данных. Процесс требует значительных вычислительных ресурсов: прямого распространения (forward pass) и обратного распространения (backpropagation) с расчётом градиентов. Обычно используется большой объём данных (Large N).

- *Вывод (Inference)* – применение обученной модели к новым входным данным. На этом этапе задействуется только прямое распространение, и объёмы данных значительно меньше (Smaller, varied N).

GPU эффективно ускоряют оба этапа за счёт:

- *Массового параллелизма:* тысячи вычислительных ядер одновременно обрабатывают матричные операции. Обучение выполняется крупными пакетами (batches), вывод – малыми или единичными запросами.

- *Оптимизированных вычислительных блоков:* специализированные тензорные ядра (Tensor Cores) значительно повышают производительность при работе с форматами FP16, BF16 и INT8.

Пример: Обучение модели ResNet-50 на CPU может занять более 100 часов. На GPU (например, NVIDIA V100) с использованием оптимизированных библиотек (cuDNN, TensorRT) это время сокращается до ≈2 часов [4], [5].

1.3. Тензорные ядра – сердце ускорения нейросетей

Тензорные ядра (Tensor Cores) представляют собой специализированные вычислительные модули в составе GPU, предназначенные для сверхбыстрого выполнения матричных

операций. Именно эти ядра стали одной из главных причин, по которым графические процессоры получили столь широкое распространение в задачах обучения и инференса нейронных сетей. Основная причина в том, что матричное умножение – центральная и наиболее ресурсоёмкая операция в глубоких нейросетях, и тензорные ядра максимально эффективно справляются с этой задачей.

Чтобы лучше осознать значение этих компонентов, полезно рассмотреть пример вычислений. Предположим, нам необходимо умножить две матрицы размером 32×32 : A и B, результатом станет матрица C. На примере этого упрощённого сценария можно проследить, как происходит ускорение вычислений при использовании тензорных ядер по сравнению с традиционным подходом. Разумеется, в реальных задачах применяются более сложные методы оптимизации, включая двойную буферизацию, управление регистрами, настройку потоочной загрузки и использование параллелизма на уровне инструкций. Но для понимания основ такие усложнения можно временно опустить.

Понимание времени исполнения циклов

Каждый вычислительный процесс в GPU измеряется в циклах. Например, процессор, работающий на частоте 1 ГГц, способен совершать миллиард тактов в секунду. Однако отдельные инструкции занимают разное количество циклов – от одной до сотен. В связи с этим между операциями возникает очередь ожидания, которую называют латентностью.

Для GPU архитектуры Ampere характерны следующие средние задержки (в циклах):

- Чтение из глобальной памяти (видеооперативной): ≈ 380 циклов
- Обращение к кэшу второго уровня (L2): ≈ 200 циклов
- Доступ к общей памяти или кэшу L1: ≈ 34 цикла
- Операция fused multiply-add (FFMA): ≈ 4 цикла
- Матричное умножение с помощью тензорного ядра: всего 1 цикл

Следует также учитывать, что операции в GPU выполняются пакетами по 32 потока — так называемыми варпами (warps). В рамках одного варпа все потоки работают синхронно и вынуждены ожидать друг друга. Именно под такую модель оптимизирована вся память графического процессора. Например, чтение из глобальной памяти происходит блоками по 128 байт, что соответствует 32 элементам float — по одному на поток.

На одном streaming multiprocessor (SM) – блоке, аналогичном CPU-ядру — может одновременно находиться до 32 варпов, что эквивалентно 1024 параллельным потокам. Все ресурсы SM (регистры, кэш, доступ к тензорным ядрам) делятся между этими варпами. Поэтому иногда предпочтительнее задействовать меньше варпов, но с более полным доступом к ресурсам на каждый.

Пример вычислений: матричное умножение 32×32

Для двух подходов – с тензорными ядрами и без – предположим одинаковые условия:

- 8 SM (что примерно соответствует 10% от GPU RTX 3090)
- по 8 варпов на каждый SM

Умножение без тензорных ядер

Без использования специализированных блоков умножение матриц выполняется поэлементно, через скалярные произведения строк и столбцов.

На практике алгоритм выглядит так:

1. Матрицы A и B сначала загружаются из глобальной памяти в общую память (shared memory), поскольку доступ к ней значительно быстрее (34 цикла вместо 200).

2. Для загрузки всех данных используется параллелизм: 64 варпа, равномерно распределённых по 8 SM, справляются с этим за 200 циклов.

3. Каждый SM вычисляет 8 скалярных произведений (dot products) для 8 элементов матрицы C:

Каждый поток обрабатывает 8 элементов:

1. 8 обращений к shared memory (загрузка векторов 32 элемента) $\times$ 34 цикла = 272 цикла.

2. 8×32 операций FFMA (умножение-сложение) – выполняются параллельно, 4 цикла на группу = 32 цикла.

Итоговое время:

200 (загрузка из L2) + 272 (обращения к shared memory/L1) + 32 (вычисления) = 504 циклов

Умножение с использованием тензорных ядер

Тензорные ядра способны выполнять матричное умножение блоков размером 4×4 всего за один цикл. Для матрицы 32×32 потребуется 64 таких блока (8×8).

Так как в одном SM — 8 тензорных ядер, а у нас 8 SM, то все 64 операции можно распределить между блоками и выполнить одновременно.

Алгоритм:

1. Матрицы A и B загружаются в shared memory (34 цикла после 200 цикла загрузки из глобальной памяти).

2. Выполняется 64 параллельных операции тензорных ядер — каждая по 1 цикл.

Итоговое время:

200 (загрузка из L2) + 34 (обращения к shared memory/L1) + 1 (вычисление) = 235 циклов

Вывод: тензорные ядра в 2 раза быстрее

Таким образом, при одинаковом объёме вычислений, применение тензорных ядер позволяет сократить общее время выполнения почти вдвое: с 504 до 235 циклов. Это и объясняет, почему современные GPU так активно применяются для задач машинного обучения — ускорение колоссальное.

Дополнительные ускорения: асинхронные копии и TMA.

Современные архитектуры, начиная с Ampere (RTX 30) и особенно в Hopper (H100), предлагают дополнительные способы ускорения – прежде всего, за счёт оптимизации доступа к памяти.

- Асинхронные копии позволяют загружать данные из глобальной памяти в shared memory, не блокируя при этом вычисления. Таким образом, часть чтений может выполняться одновременно с выполнением FFMA или тензорных операций.

- Tensor Memory Accelerator (TMA), появившийся в Hopper, отвечает за автоматическое вычисление индексов при загрузке данных. Благодаря этому каждый поток может сосредоточиться только на вычислениях, а не тратить время на выборку нужных адресов.

При использовании этих технологий время загрузки из глобальной памяти может быть частично скрыто, что даёт дополнительное ускорение примерно на 15%.

Заключение

Из представленного анализа становится ясно: главным ограничивающим фактором для тензорных ядер является пропускная способность памяти. Даже если вычисление занимает всего один цикл, результат невозможен без своевременной доставки данных.

Чтобы максимально раскрыть потенциал тензорных блоков, необходимо либо ускорить доступ к памяти (например, за счёт повышения частоты и ширины шины), либо оптимизировать его структуру. И именно в этом направлении движется развитие современных GPU: уменьшая задержки и увеличивая скорость передачи, архитекторы создают условия, при которых тензорные ядра могут работать на пике своих возможностей.

1.4. Инструменты и библиотеки для работы с тензорными ядрами

Эффективное использование тензорных ядер требует специализированных инструментов, адаптированных под различные языки программирования. Рассмотрим ключевые решения для C++, Java и Python.

Выбор инструмента зависит от требуемого уровня контроля, производительности и специфики задачи:

1. C++ (CUDA + cuBLAS)

- *Уровень контроля:* Низкоуровневый – прямое управление памятью GPU и настройка вычислений под тензорные ядра.

- *Производительность:* Максимальная (до 312 TFLOPS на A100).

- *Где применять:* Высокопроизводительные вычисления (HPC), задачи с жесткими требованиями к скорости.

2. Java (JCuda)

- *Уровень контроля:* Средний – работа через JNI-обёртки, но с ручным управлением памятью.

- *Производительность:* до 250 TFLOPS (из-за накладных расходов JVM).

- *Где применять:* Встраивание GPU-ускорения в Java-приложения (например, Big Data-обработка в Apache Spark).

3. Python (PyTorch/CuPy)

- *Уровень контроля:* Высокоуровневый – автоматическое управление памятью и mixed precision.

- *Производительность:* до 280 TFLOPS (благодаря встроенной оптимизации).

- *Где применять:* быстрое прототипирование ML-моделей и эксперименты.

Оптимальный выбор инструмента зависит от конкретных задач: если требуется максимальная производительность, следует использовать C++ с CUDA; для работы в Java-экосистеме подойдёт JCuda; а для быстрой разработки и экспериментов лучше всего подходят Python-библиотеки PyTorch и CuPy.

2. Математика ускорения: цифры и факты

2.1. Вычислительная сложность нейросетей

Для нейросети с L слоями и N нейронами в слое вычислительная сложность определяется следующим образом:

Прямое распространение (forward pass):

$$FLOPs_{fwd} = O(L \times N^2)$$

Обратное распространение (*backpropagation*):

$$FLOPs_{bwd} = O(2 \times L \times N^2)$$

Общая сложность одной итерации:

$$FLOPs_{total} = O(3 \times L \times N^2)$$

2.3. Почему ускорение не линейное?

Теоретически, если предположить идеальную параллелизацию всех вычислений, увеличение вычислительных ресурсов (например, числа GPU) должно давать линейное ускорение. Однако на практике линейное ускорение недостижимо из-за следующих факторов, которые можно доказать на основе предоставленных данных:

Как показано в фундаментальной работе Амдала [7], теоретическое ускорение параллельных вычислений описывается формулой:

$$S_{max} = \frac{1}{f_{seq} + \frac{1 - f_{seq}}{P}}$$

S_{max} – теоретически достижимое ускорение, f_{seq} – доля последовательных вычислений ($0 \leq f_{seq} \leq 1$), P – количество потоков.

Даже при $P \rightarrow \infty$ ускорение ограничено величиной $1/f_{seq}$. Например, если 5% операций нельзя распараллелить ($f_{seq}=0.05$), максимальное ускорение не превысит 20х

Накладные расходы на передачу данных

- Пример с тензорными ядрами:

- Без тензорных ядер: 504 цикла (из них 200 циклов – загрузка данных из глобальной памяти).

- С тензорными ядрами: 235 циклов (200 циклов – загрузка данных).

- Вывод: даже при ускорении вычислений в 2.1 раза (504 → 235 циклов) задержки памяти остаются доминирующими. Это делает задачу *memory-bound*, где пропускная способность памяти, а не вычислительная мощность, становится узким местом.

- Увеличение числа ядер не сокращает время доступа к памяти, что нарушает линейность ускорения.

Ограничения параллелизации на уровне *warps*

- В архитектуре GPU варпы (*warp*) из 32 потоков работают синхронно. Если задача требует неравномерного распределения вычислений (например, условные переходы), часть потоков простаивает, снижая эффективность.

- Ресурсы SM (регистры, общая память) делятся между активными варпами. Увеличение числа варпов уменьшает доступные ресурсы на каждый, что может замедлить выполнение.

Итог

Линейное ускорение невозможно из-за:

1. Присутствия последовательных операций ($f_{seq} > 0$).
2. Доминирования задержек памяти над вычислениями (memory-bound).
3. Накладных расходов на синхронизацию и передачу данных.
4. Ограничений архитектуры GPU (варпы, SM).

Пример с тензорными ядрами иллюстрирует, что даже при оптимизации ключевых операций (матричных умножений) основным узким местом остается доступ к памяти, который не масштабируется линейно с ростом числа ядер.

5. Будущее GPU и альтернативные технологии

5.1. Специализированные ускорители

- TPU от Google – оптимизированы для TensorFlow.
- IPU от Graphcore – новая архитектура для задач ИИ.
- Neuromorphic chips – имитируют работу биологических нейронов.

5.2. Перспективные направления

1. Оптические вычисления – Использование света (фотонов) вместо электричества (электронов) для передачи и обработки данных.

Формула задержки:

$$T_{opt} \approx \frac{c \times n}{L}$$

где c – скорость света, n – показатель преломления, L – длина оптического пути.

2. Квантовые сопроцессоры – Использование квантовых битов (кубитов) для решения задач, недоступных классическим компьютерам.

3. Гибридные системы – комбинация CPU/GPU/TPU.

6. Заключение: GPU как катализатор ИИ-революции

Графические процессоры (GPU) стали ключевым инструментом в развитии ИИ благодаря своей способности эффективно выполнять параллельные вычисления, такие как матричные операции в нейронных сетях. Их применение сокращает время обучения моделей в десятки раз по сравнению с CPU, что особенно важно для крупных систем вроде GPT-3.

Однако у GPU есть и ограничения: высокое энергопотребление, дорогое оборудование и нагрузка на инфраструктуру. В будущем возможен переход к более специализированным ускорителям (TPU, IPU) и альтернативным технологиям, таким как оптические и квантовые вычисления.

Таким образом, хотя GPU остаются критически важными для современного ИИ, дальнейший прогресс в этой области потребует новых архитектурных решений.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks // NIPS. 2012.
2. NVIDIA A100 Tensor Core GPU Architecture. Whitepaper, 2020.
3. Jouppi N. et al. Ten Lessons from Three Generations Shaped Google's TPUv4i // ISCA. 2021.
4. He K. et al. Deep Residual Learning for Image Recognition // CVPR. 2016.
5. Shi S. et al. Benchmarking State-of-the-Art Deep Learning Software Tools // IEEE TPDS. 2019.
6. NVIDIA CUDA C++ Programming Guide. 2023.
7. Amdahl G. Validity of the Single-Processor Approach // AFIPS. 1967.
8. PyTorch Documentation: Automatic Mixed Precision. 2023.
9. Jouppi N. et al. TPUv4: An Optically Reconfigurable Supercomputer // IEEE Micro. 2021.

УДК 004.65

СЫРКИН Г.Е.

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ОСОБЕННОСТИ РАЗРАБОТКИ СЕРВЕРНОЙ ЧАСТИ ИНТЕРНЕТ-МАГАЗИНА

Рассматриваются ключевые аспекты разработки серверной части веб-приложений. Описаны архитектура системы, основные компоненты и их взаимодействие, а также реализация ключевых функций.

Современные веб-приложения требуют надежной и эффективной серверной части для обеспечения стабильной работы и обработки

данных. В данной статье представлен опыт разработки серверных компонентов для интернет-магазина книг.

Интернет-магазин – это высоконагруженное, сложное в интеграциях и регулируемое законом приложение, где каждая ошибка может стоить денег. В то время как обычные веб-приложения (например, блог или корпоративный сайт) фокусируются на контенте и пользовательском взаимодействии, интернет-магазин требует особого внимания к: безопасности платежей, работе с остатками и заказами, скорости и отказоустойчивости.

Разработка серверной(backend) части веб-приложения осуществлялась последовательно, начиная с подготовительного этапа, включавшего выбор технологического стека (PHP, PostgreSQL, Apache), настройку окружения разработки и создание структуры проекта. На следующем этапе была разработана база данных: спроектирована структура, созданы необходимые таблицы, настроены связи между ними и реализовано подключение через PDO. Затем была создана система аутентификации с формами регистрации и авторизации, реализовано хеширование паролей и система ролей пользователей. После этого разработан функционал каталога книг с фильтрацией, поиском и системой категорий. На этапе реализации корзины покупок создана структура для управления товарами, система расчета стоимости и интеграция с пользователями. Для администраторов разработана панель управления с возможностью управления товарами, загрузки изображений и управления пользователями. Особое внимание было уделено безопасности и оптимизации: реализована защита от SQL-инъекций, настроена валидация данных, оптимизированы запросы к базе данных и настроено кэширование. Завершающим этапом стало тестирование и отладка, включавшее проверку функциональности, исправление ошибок, оптимизацию производительности и тестирование безопасности.

Рассмотрим разработку на примере создания интернет-магазина книг.

1. Настройка виртуальных хостов Apache

Для корректной работы проекта необходимо настроить виртуальные хосты в Apache. Это позволяет:

- Использовать собственное доменное имя для проекта
- Изолировать проект от других веб-приложений
- Настроить SSL-сертификаты
- Управлять правами доступа

1.1 Конфигурация виртуального хоста

Создание конфигурационного файла `bookstore.conf` в директории Apache:

```
```\apache
<VirtualHost *:80>
ServerNamebookstore.local
ServerAlias www.bookstore.local
DocumentRoot "C:/xampp/htdocs/bookstore"
<Directory "C:/xampp/htdocs/bookstore">
 Options Indexes FollowSymLinksMultiViews
 AllowOverride All
 Require all granted
 # Включениеобработки PHP
<FilesMatch \.php$>
SetHandler application/x-httpd-php
</FilesMatch>
</Directory>
 # Настройкабезопасности
<IfModulemod_headers.c>
 Header set X-Content-Type-Options "nosniff"
 Header set X-XSS-Protection "1; mode=block"
 Header set X-Frame-Options "SAMEORIGIN"
</IfModule>
</VirtualHost>
```\
```

1.2 Настройкафайла hosts

Добавлениезаписивфайл `C:\Windows\System32\drivers\etc\hosts`:

```
```\
127.0.0.1 glebSyrkn.local
127.0.0.2 www.bookstore.local
```\
```

2. Подключение к базе данных

Для работы с СУБД в PHP применяется технология PDO (PHP Data Objects), которая обеспечивает надежный и производительный обмен данными. PDO предлагает систему взаимодействия, совместимый с разными СУБД, а также поддерживает механизм подготовленных выражений, что исключает риски SQL-инъекций и повышает безопасность взаимодействия с базой данных.

Основные параметры подключения:

- `host`: адрес сервера PostgreSQL
- `dbname`: имя базы данных
- `username`: имя пользователя
- `password`: пароль пользователя
- `port`: порт PostgreSQL (по умолчанию 5432)

3. Структура базы данных

Основные таблицы:

- users (пользователи)
- books (книги)
- genres (жанры)
- cart (корзина)
- reviews (отзывы)

4. Система ролей пользователей

В системе реализована трехуровневая система ролей:

- Гость (неавторизованный пользователь)
- Пользователь (авторизованный пользователь)
- Администратор

4.1 Реализация регистрации

Форма регистрации включает базовые поля для создания учетной записи пользователя. Все поля обязательны для заполнения, а пароль требует подтверждения для предотвращения ошибок ввода.

```

` `` `php
<form method="POST" action="register.php">
<input type="text" name="username" required>
<input type="email" name="email" required>
<input type="password" name="password" required>
  <input type="password" name="confirm_password" re-
quired>
  <button type="submit">Зарегистрироваться</button>
</form>
` `` `

```

Обработка регистрации включает проверку существующего пользователя, безопасное хеширование пароля и добавление нового пользователя в базу данных. Функция возвращает true при успешной регистрации и false при ошибке.

```

` `` `php
function registerUser($username, $email, $password)
{
    global $conn;
    // Проверка существующего пользователя
    $stmt = $conn->prepare("SELECT user_id FROM users
WHERE username = ? OR email = ?");
    $stmt->execute([$username, $email]);
    if ($stmt->rowCount() > 0) {
        return false;
    }
    // Хеширование пароля и добавление пользователя
    $hashed_password = password_hash($password,
PASSWORD_DEFAULT);

```

```

$stmt = $conn->prepare("INSERT INTO users
(username, email, password, role) VALUES (?, ?, ?,
'user')");
return $stmt->execute([$username, $email,
$hashed_password]);
}

```

4.2 Реализация авторизации

Форма авторизации содержит минимально необходимые поля для входа пользователя в систему. После успешной авторизации создается сессия с информацией о пользователе.

```

```php
<form method="POST" action="login.php">
<input type="text" name="username" required>
<input type="password" name="password" required>
<button type="submit">Войти</button>
</form>
```

```

Процесс авторизации включает проверку учетных данных пользователя, сравнение хеша введенного пароля с хешем в базе данных и создание сессии при успешной проверке.

Управление товарами (функционал администратора)

Форма добавления нового товара включает все необходимые поля для создания записи о книге в каталоге. Поддерживается загрузка изображений и выбор жанра из предустановленного списка.

```

```php
<form method="POST" action="add_book.php"
enctype="multipart/form-data">
<input type="text" name="title" required>
<input type="text" name="author" required>
<select name="genre_id" required>
<?php foreach($genres as $genre): ?>
<option value="<?php echo $genre['genre_id']; ?>">
<?php echo htmlspecialchars($genre['name']); ?>
</option>
<?php endforeach; ?>
</select>
<input type="number" name="price" step="0.01"
required>
<textarea name="description" required></textarea>
<input type="file" name="image" accept="image/*"
required>
<button type="submit">Добавить книгу</button>
</form>
```

```

Обработка добавления товара включает проверку прав администратора, безопасную обработку загруженного изображения и добавление информации о книге в базу данных.

```

    `php
    function addBook($title, $author, $genre_id,
    $price, $description, $image) {
        global $conn;
        if (!isAdmin()) return false;
        $image_url = uploadImage($image);
        $stmt = $conn->prepare("INSERT INTO books
    (title, author, genre_id, price, description, image_url)
    VALUES (?, ?, ?, ?, ?, ?)");
        return $stmt->execute([$title, $author,
    $genre_id, $price, $description, $image_url]);
    }
    function uploadImage($image) {
        $target_dir = "uploads/";
        $file_extension =
    strtolower(pathinfo($image["name"],
    PATHINFO_EXTENSION));
        $new_filename = uniqid() . '.' .
    $file_extension;
        $target_file = $target_dir . $new_filename;
        if (!in_array($file_extension, ['jpg', 'jpeg',
    'png', 'gif'])) {
            throw new Exception("Неподдерживаемый формат файла");
        }
        if (move_uploaded_file($image["tmp_name"],
    $target_file)) {
            return $target_file;
        }
        throw new Exception("Ошибка при загрузке файла");
    }
    ...

```

6. Особенности работы с PostgreSQL

PostgreSQL предоставляет следующие преимущества для данного проекта:

- Поддержка сложных типов данных (массивы, JSON)
- Транзакционность и ACID-совместимость
- Расширенные возможности индексации
- Поддержка полнотекстового поиска
- Возможность создания пользовательских типов данных

Основные компоненты системы

1. Система аутентификации

Реализована с использованием PHP-сессий и хеширования паролей:

```
```php
function isLoggedIn() {
 return isset($_SESSION['user_id']);
}
function isAdmin() {
 return isset($_SESSION['is_admin']) &&
$_SESSION['is_admin'] === true;
}
...

```

Система корзины покупок реализована с использованием сессий для хранения временных данных и базы данных для постоянного хранения. Основные функции включают добавление, удаление и обновление товаров, а также расчет общей стоимости.

### 2.1 Добавление товара в корзину

Функция добавляет товар в корзину с проверкой наличия и обновлением количества, если товар уже существует.

```
```php
function addToCart($book_id, $quantity = 1) {
    global $conn;
    if (!isLoggedIn()) return false;
    // Проверка наличия товара в корзине
    $stmt = $conn->prepare("SELECT quantity FROM
cart WHERE user_id = ? AND book_id = ?");
    $stmt->execute([$_SESSION['user_id'],
$book_id]);
    $existing = $stmt->fetch();
    if ($existing) {
        // Обновление количества
        $stmt = $conn->prepare("UPDATE cart SET
quantity = quantity + ? WHERE user_id = ? AND book_id =
?");
        return      $stmt->execute([$quantity,
$_SESSION['user_id'], $book_id]);
    } else {
        // Добавление нового товара
        $stmt = $conn->prepare("INSERT INTO cart
(user_id, book_id, quantity) VALUES (?, ?, ?)");
        return      $stmt->execute([$_SESSION['user_id'], $book_id, $quantity]);
    }
    ...
}

```

2.2 Получение содержимого корзины

Функция возвращает полную информацию о товарах в корзине, включая детали книг и общую стоимость.

```
```php
function getCartContents() {
 global $conn;
 if (!isLoggedIn()) return [];
 $stmt = $conn->prepare("
 SELECT c.quantity, b.title, b.author,
b.price, b.image_url
 FROM cart c
 JOIN books b ON c.book_id = b.book_id
 WHERE c.user_id= ?
 ");
 $stmt->execute([$SESSION['user_id']]);
 return $stmt->fetchAll();
}

function getCartTotal() {
 global $conn;
 if (!isLoggedIn()) return 0;
 $stmt = $conn->prepare("
 SELECT SUM(c.quantity * b.price) as total
 FROM cart c
 JOIN books b ON c.book_id = b.book_id
 WHERE c.user_id= ?
 ");
 $stmt->execute([$SESSION['user_id']]);
 return $stmt->fetch()['total'] ?? 0;
}
...

```

### 3. Каталог книг

Каталог книг реализован с поддержкой фильтрации, сортировки и пагинации. Система позволяет отображать книги по различным категориям и параметрам.

#### 3.1 Получение списка книг с фильтрацией

Функция возвращает отфильтрованный список книг с поддержкой пагинации и сортировки.

```
```php
function getBooks($filters = [], $page = 1,
$per_page = 12) {
    global $conn;
    $offset = ($page - 1) * $per_page;
    $where = [];
    $params = [];
    // Формирование условий фильтрации
    if (!empty($filters['genre_id'])) {
        $where[] = "b.genre_id = ?";
    }
}

```



```

        $params[] = $filters['genre_id'];
    }
    if (!empty($filters['price_min'])) {
        $where[] = "b.price>= ?";
        $params[] = $filters['price_min'];
    }
    if (!empty($filters['price_max'])) {
        $where[] = "b.price<= ?";
        $params[] = $filters['price_max'];
    }
    // Формирование SQL-запроса
    $sql = "SELECT b.*, g.name as genre_name
            FROM books b
            LEFT JOIN genres g ON b.genre_id =
g.genre_id";
    if (!empty($where)) {
        $sql .= " WHERE " . implode(" AND ",
$where);
    }
    // Добавление сортировки
    $sql .= " ORDER BY b.created_at DESC LIMIT ?
OFFSET ?";
    $params[] = $per_page;
    $params[] = $offset;
    $stmt = $conn->prepare($sql);
    $stmt->execute($params);
    return $stmt->fetchAll();
}

```

3.2 Получение специальных разделов каталога

Функции для получения книг из различных специальных разделов каталога.

```

```php
function getNewBooks($limit = 5) {
 global $conn;
 $stmt = $conn->prepare("
 SELECT * FROM books
 ORDER BY created_at DESC
 ");
 $stmt->execute([$limit]);
 return $stmt->fetchAll();
}

```

Разработанная backend часть обеспечивает надежную и эффективную работу веб-приложения. Основные особенности реализации включают безопасную аутентификацию, эффективное управление данными и оптимизированную производительность благодаря использованию PostgreSQL.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Сапрыкин А.Н. Основы веб-программирования: HTML 5, CSS 3, JavaScript. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2025. – 248 с.
2. PHP Documentation. [Электронный ресурс] – URL: <https://www.php.net/docs.php> (дата обращения: 20.01.2025).
3. PostgreSQL Documentation. [Электронный ресурс] – URL: <https://www.postgresql.org/docs/> (дата обращения: 20.01.2025).
4. PDO Documentation. [Электронный ресурс] – URL: <https://www.php.net/manual/en/book.pdo.php> (дата обращения: 20.01.2025).

УДК 004.4`2

**СЫСОЕВ М.Э.**

Рязанский государственный радиотехнический университет  
имени В. Ф. Уткина

**СРАВНИТЕЛЬНЫЙ АНАЛИЗ HTTP-ФРЕЙМВОРКОВ GIN,  
FIBER И ECHO ДЛЯ ВЫСОКОНАГРУЖЕННЫХ WEB-  
ПРИЛОЖЕНИЙ**

*В статье рассматриваются особенности обработки HTTP-запросов в высоконагруженных web-приложениях с использованием фреймворков на языке Go. Представлен сравнительный анализ производительности Gin, Fiber и Echo на основе экспериментальных замеров.*

В эпоху цифровой трансформации веб-приложения становятся критически важными компонентами бизнес-процессов. От интернет-магазинов до облачных платформ — производительность веб-серверов напрямую влияет на пользовательский опыт, масштабируемость и экономическую эффективность сервисов. Одной из ключевых задач разработки современных web-приложений является эффективная обработка HTTP-запросов, особенно в условиях высокой нагрузки.

Наиболее популярные фреймворки, используемые для разработки высоконагруженных систем на языке Go — это Gin, Fiber и Echo. Все они предоставляют инструменты для построения REST API, но различаются по внутренней архитектуре, управлению памятью, модели обработки запросов и производительности.

Целью данной работы является проведение сравнительного анализа производительности этих фреймворков в условиях различных

нагрузок, чтобы определить оптимальное решение для задач, связанных с высоконагруженными веб-приложениями.

Экспериментальные испытания проводились на ноутбуке MacBook с процессором Apple M2, оснащённым 8 производительными и 4 энергоэффективными ядрами CPU, а также 16 ГБ оперативной памяти. Все тесты выполнялись с использованием следующих версий программного обеспечения:

Язык программирования Go — версия 1.22.3

Веб-фреймворк Gin — версия 1.10.0

Веб-фреймворк Fiber — версия 2.52.4

Веб-фреймворк Echo — версия 4.12.0

Для проведения нагрузочного тестирования использовалась модифицированная версия утилиты Bombardier, в которой каждый HTTP-запрос включал генерацию случайного email-адреса. Это позволяло моделировать приближённые к реальности условия обработки запросов.

В качестве базы данных использовалась PostgreSQL, предварительно загруженная 100 тысячами записей пользователей. Для каждого фреймворка выполнялись два набора тестов — при 100 и 300 одновременных подключений, каждый из которых обрабатывал 5 миллионов HTTP-запросов.

Сравнительные таблицы HTTP-фреймворков:

Таблица 1 – Задержка

Метрика	Gin	Fiber	Echo
Минимальная задержка (100 соединений), мс	0.11	0.10	0.12
Минимальная задержка (300 соединений), мс	0.12	0.12	0.11
Средняя задержка (100 соединений), мс	2.98	2.67	2.92
Средняя задержка (300 соединений), мс	8.86	8.33	8.71
Максимальная задержка (100 соединений), мс	145.68	122.56	172.75
Максимальная задержка (300 соединений), мс	873.04	828.75	679.62

Таблица 2 – Время обработки

Количество соединений	Gin, сек	Fiber, сек	Echo, сек
100	149.4	133.6	146
300	147.7	138.9	145.1

Таблица 3 – Количество запросов в секунду (Requests per second)

Количество соединений	Gin, RPS	Fiber, RPS	Echo, RPS
100	33500	37400	34300
300	33900	36000	34400

Таблица 4 – Использование памяти

Количество соединений	Gin, МБ	Fiber, МБ	Echo, МБ
100	38	30	33
300	49	37	43

Таблица 5 – Использование процессора

Количество соединений	Gin, %	Fiber, %	Echo, %
100	345	295	342
300	346	298	345

Проведённое исследование показало, что между фреймворками — Gin, Fiber и Echo, действительно существуют различия в производительности, однако они не являются кардинальными. Наибольшую пропускную способность по количеству обрабатываемых HTTP-запросов в секунду продемонстрировал фреймворк Fiber. Остальные фреймворки также показали стабильные и близкие к нему результаты, с небольшим отставанием.

Среднее время отклика у всех трёх фреймворков находилось на схожем уровне, при этом Fiber показал немного более низкие значения. Под высокой нагрузкой различия в задержке стали заметнее, но они всё равно оставались в пределах допустимого для большинства веб-приложений. Максимальные задержки варьировались, при этом фреймворк Echo показал лучшую устойчивость к пиковым значениям.

По уровню потребления системных ресурсов Fiber оказался наиболее экономным, что может быть полезным при развертывании на серверах с ограниченными возможностями. Однако и остальные решения не требовали чрезмерных ресурсов, демонстрируя разумный баланс между потреблением и производительностью.

Таким образом, можно сделать вывод, что Fiber на текущем этапе демонстрирует наилучший баланс между производительностью и

эффективностью использования ресурсов. Однако выбор конкретного фреймворка для высоконагруженных Web-приложений должен также учитывать архитектурные особенности проекта, экосистему, документацию и предпочтения команды разработчиков.

Дополнительно следует учитывать, что производительность фреймворков может меняться в зависимости от конфигурации среды, специфики обрабатываемых запросов и используемых библиотек. Также важную роль играет опыт разработчиков и уровень поддержки сообщества, что может повлиять на скорость разработки и удобство сопровождения проекта.

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Григорик И. High Performance Browser Networking: What every web developer should know about networking and web performance. – O'Reilly Media, 2013. – 398 с.
2. Батчер М., Фарина М. Go на практике. – М.: ДМК Пресс, 2017. – 374 с.
3. Чанг С.Ш. Go Web Programming// Manning Publications. – 2016. – 312 с.
4. Ялавула Н. Building RESTful Web services with Go// Packt Publishing. – 2017. – 316 с.

УДК 004.42

**ТИХОНОВ А.А.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

### **РАЗРАБОТКА ЧАТ-БОТА TELEGRAM ДЛЯ ИНТЕРНЕТ-МАГАЗИНА**

*В данной работе рассматриваются перспективные средства разработки чат-ботов с помощью языка программирования Python с использованием СУБД PostgreSQL, предоставляющих возможность обслуживать работу интернет-магазина.*

Интернет-магазин в настоящее время – это не только сайт с каталогом и склад, с которого покупатели получают товар. Это система, состоящая из мониторинга состояния продукции, логистических процессов, взаимодействия с клиентом, а также поиска наиболее выгодных для владельца способов доставки, рекламы, денежных транзакций и связи с покупателем. Телеграмм-бот может решить множество подобных вопросов, при правильной настройке и

подборе функционала. Во-первых, это популярный мессенджер, который использует большое количество людей, во-вторых, он объективно очень удобен с точки зрения потребителя, ведь всегда находится «в кармане» и не имеет «лишнего» функционала наподобие ленты новостей или множества вкладок с музыкой, видео и т.д., из-за чего нередко его выбирают в качестве средства делового общения и рабочей коммуникации. Поэтому весьма востребованным, особенно среди мелких или индивидуальных предпринимателей, является вышеупомянутый телеграмм-бот для продажи товара, для мониторинга наиболее выгодных для хранения складов и для прочих связанных с данной отраслью услуг.

В первую очередь, необходимо выбрать основу для чат-бота, а именно – метод для взаимодействия программы с телеграммом. Для связи python и телеграмма существует ряд наиболее популярных библиотек: Python-Telegram-Bot, Telebot, pyTelegramBotAPI, Telethone и Aiogram [1]. В виду того, что библиотека Aiogram имеет широкую пользовательскую поддержку, возможность работы с асинхронным программированием, а также не содержит излишнего функционала, как тот же Telethone (больше подходящий для роботизации аккаунта, чем для написания обычного чат-бота), было решено остановиться на ней. Кроме того, выбранная библиотека также имеет обширную документацию и примеры готового кода, что также способствует облегчению написания программы.

Также для возможности полноценно работать с клиентами, необходимо использовать базу данных, в которой будут отдельно храниться пользователи, информация о них, а также списки товаров и прочие необходимые связи и записи, если таковые потребуются.

Было решено использовать СУБД (система управления базами данных) PostgreSQL. Причины тому следующие [2]: бесплатное распространение, возможность хранить большие таблицы (до 32 ТБ), возможность хранить в ячейках массивы данных, высокая скорость работы и поддержка асинхронного программирования.

Для работы с базами данных (БД) в языке программирования Python существует библиотека SQLAlchemy [3] и инструмент Alembic [5], которые позволяют реализовывать полноценную работу с БД с помощью миграций (работа со структурной частью базы: добавление и удаление таблиц, столбцов и связей) и завернутых в собственный синтаксис SQL-транзакций для работы с данными.

В качестве примера миграции можно привести следующий код:

```
from alembic import op
import sqlalchemy as sa
revision identifiers, used by Alembic.
```

```
revision = ' some_string_id '
down_revision = 'some_string_id'
branch_labels = None
depends_on = None
def upgrade() -> None:
 op.add_column("table_name", sa.Column("column_name",
sa.Integer(), nullable=False))
def downgrade() -> None:
 op.drop_column("table_name ", "column_name")
```

В приведённом примере представлена миграция, в которой в некую таблицу *table\_name* добавляется столбец *column\_name* с целочисленным типом данных, а также определена функция *downgrade*, которая отработает в случае отката миграции и устранил все внесённые изменения.

Таким образом, технологический стек для разработки выбран, теперь можно начинать написание чат-бота.

Как и говорилось ранее, очевидным плюсом Aiogram является простота использования, в связи с чем, в отличие от Telethon (ещё одна популярная библиотека, заточенная больше под роботизацию аккаунта пользователя, чем под написание чат-ботов) нам, как разработчику, не нужно передавать данные своего аккаунта под роботизированное управление [4]. Таким образом, для начала работы достаточно лишь написать простой стартовый набор файлов и создать себе пробного телеграмм-бота с помощью BotFather (средство самого телеграма для создания ботов) и получить с него api-токен, позволяющий управлять новосозданным ботом. Кроме того, в виду обширного сообщества, уже существует созданный под подобного рода нужды шаблон, предусматривающий, как масштабирование чат-бота, так и работу с разными СУБД, в зависимости от предпочтений и возможностей разработчика. Структура файлов данного шаблона представлена далее (см. рисунок 1).

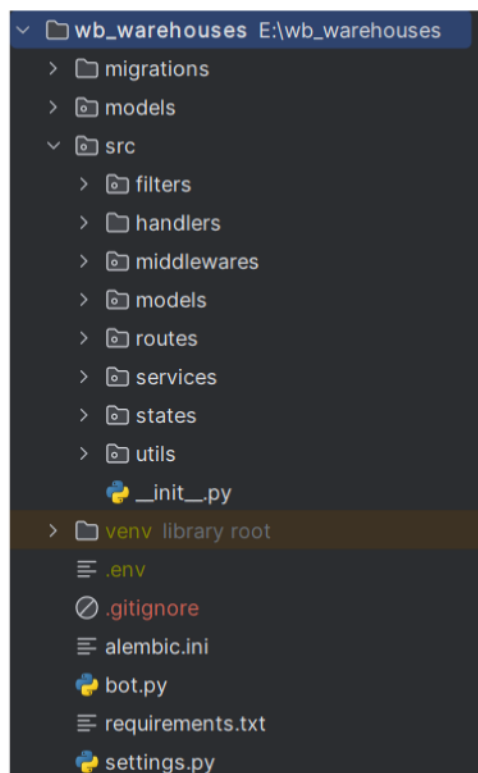


Рисунок 1 – Структура каталога шаблона чат-бота на Aiogram

Здесь стоит дать некоторые пояснения. Каталог migrations – это хранилище упомянутых ранее «миграций» базы данных, с которой работает бот, в нём хранится история изменений БД, с возможностью отката до самой ранней версии и обратно, и, соответственно, актуальная версия набора множеств и отношений. Каталог models содержит т.н. «модели» БД, то есть классы, которые являются связующим звеном между логикой работы SQLAlchemy (которая напрямую взаимодействует с БД) и кодом на Python. Файл .env содержит переменные виртуального окружения, здесь можно в целях безопасности указывать разные необходимые для работы бота значения, такие как токен бота или DSN базы данных. Файл alembic.ini – основной конфигурационный файл для инструмента Alembic, который осуществляет работу с миграциями. Файл bot.py – основной исполняемый файл, который запускает асинхронную функцию main, поддерживающую работу чат-бота до получения команды о завершении работы. Файл settings.py – настройки бота (в т.ч. api-токен и DSN базы данных). Каталог src – непосредственно та часть шаблона, в которой в основном и происходит написание кода для обработки поведения бота в конкретной ситуации в зависимости от цели проекта.

В src содержатся следующие каталоги: filters – содержит обработчик фильтров, например, если необходимо дать доступ к каким-то командам только определённым группам пользователей;



`models` – классы для работы с базами данных или сторонними сайтами; `routes` – каталог для реализации CRUD-интерфейса (создание, чтение, обновление, удаление данных) взаимодействия с БД с помощью библиотеки `SQLAlchemy`; `handlers` – содержит основные обработчики команд, сообщений и так называемых `callback`-функций или запросов, которые могут срабатывать при нажатии на диалоговые кнопки; `states` – позволяет задавать возможные состояния работы чат-бота для выполнения каких-либо действий; `services` и `middlewares` могут быть использованы при добавлении более сложных функций, по типу работы с API стороннего сайта или организации планировщика событий.

В качестве примера написания CRUD интерфейса можно привести следующий фрагмент кода:

```
async def delete_user(user_id: int) -> None:
 with Session() as session:
 query_user: User =
session.query(User).filter(User.id ==
user_id).one_or_none()
 if query_user is not None:
 User.delete(query_user.id, session=session)
 session.commit()
```

В данном примере представлена функция удаления пользователя из БД по принципу «если такой пользователь есть, то его надо удалить». Про объект `Session` в дальнейшем будет упомянуто, на данный момент для понимания достаточно того, что это обращение к сессии взаимодействия с базой данных.

В целом, работу чат-бота для магазина можно сравнить с написанием сайта, так как, по сути, обработка любого события – это обращение к БД с целью получить, записать, удалить или обновить информацию и вернуть пользователю ожидаемый результат. В качестве удобного инструмента управления функционалом можно выбрать «кнопки» действий, которые можно прикреплять к сообщениям с помощью свойства `reply_markup` для типа `Message` библиотеки `Aiogram`. С помощью подобных кнопок (см. рисунок 2) можно «отлавливать» `callback`-запросы и запускать тот или иной сценарий событий, который может как запускать обновления каких-либо данных, если это необходимо, так и выдавать простые ответы, в зависимости от необходимого поведения.

Клавиатура (`InlineKeyboard`) для сообщения составляется и прикрепляется к нему следующим образом:

```
builder: InlineKeyboardBuilder =
InlineKeyboardBuilder()
```

```
builder.button(text="📄 Перейти в меню!",
callback_data="main_menu")
await message.answer("Вы зарегистрированы, можно
начинать работу!",
reply_markup=builder.as_markup())
```

Результат работы этого фрагмента кода представлен на рисунке 2. Стоит отметить, что взаимодействие с клавиатурой является частью библиотеки *Aiogram* и находится в части библиотеки *aiogram.utils.keyboard*. Свойство *callback\_data* позволяет разработчику обрабатывать событие нажатия созданной кнопки.

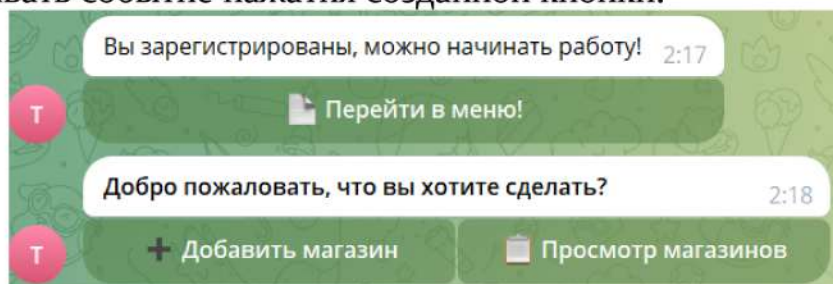


Рисунок 2 – Переход по нажатию на кнопку в основное меню

В нашем примере, при получении команды «/start» чат-бот сверяет идентификатор чата (уникальное значение, обозначающее переписку, куда бот отправляет сообщения) пользователя с имеющимися в БД, если такого нет – происходит запись (см. рисунок 3). Далее вся работа с конкретным пользователем определяется идентификатором чата.

	id	chat_id
1	2	52
2	3	78

Рисунок 3 – Запись в БД пользователей после регистрации

Работая с *Aiogram*, в основном, необходимо обрабатывать (или же «отлавливать») три типа событий: сообщение (*Message*), обратный запрос (*CallbackQuery*), сообщение с состоянием (*State*). Если первые два логически понятны, то про состояние стоит добавить, что это, по сути, сообщение, которое чат-бот ожидает получить, имея в виду, что оно будет в себе нести информацию конкретного содержания, будь то: дата и время, имя, фамилия, токен чего-нибудь и т.д. Пример применения: нажатие на клавишу «добавить событие» порождает callback-вызов, который устанавливает состояние ожидания названия события, после получения названия в ожидаемом виде, чат-бот добавляет его в модель данных, которая в будущем будет записана в БД, после чего устанавливает состояние ожидания даты и времени мероприятия в заданном формате, затем добавляет эти данные в

модель и отправляет её на обработку для записи в БД. Пример реализации подобной последовательности шагов представлен на рисунке 4 и рисунке 5.

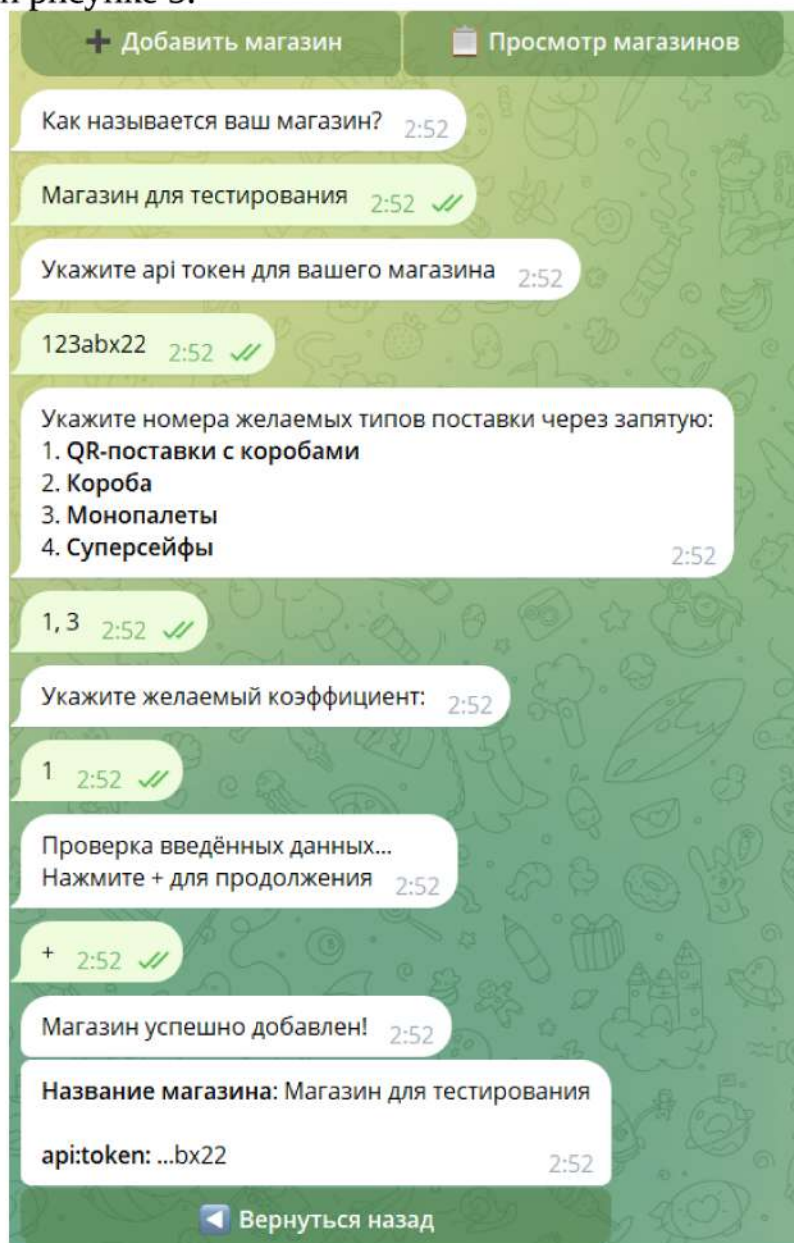


Рисунок 4 – Добавление магазина в чат-бот с помощью состояний

	id	123 user_id	name	api_token	123 coefficient	supply_type
1						
2						
3						
4						
5						
6						
7						
8						
9	17	2	Магазин для 123abx22		1	Короба

Рисунок 5 – Результат добавления магазина в чат-бот

В качестве примера работы с состояниями ниже приведено два фрагмента кода, отвечающих за: создания шаблона состояния; обработку ботом сбора данных согласно состояниям.

```
class CreateShop(StatesGroup):
```

```
 user_id = State()
 name = State()
 supply_type = State()
 api_token = State()
 coefficient = State()
 check_data = State()
```

В данном фрагменте устанавливается набор состояний для запроса данных при создании магазина в БД.

```
@router.message(CreateShop.name)
async def bot_get_name(message: Message, state:
FSMContext):
 await state.update_data({"name": message.text})
 await message.answer("Укажите api токен для вашего
магазина")
 await state.set_state(CreateShop.api_token)
```

В функции *bot\_get\_name* происходит сохранение имени магазина, вывод запроса на указание api-токена и установка состояния ожидания получения api-токена для магазина.

В свою очередь, сама работа с базой данных происходит путём реализации простого алгоритма, который содержит примерно следующий порядок действий: открытие сессии с базой данных с помощью объекта библиотеки SQLAlchemy – *Session*; сравнение объекта, который нужно занести в базу данных, или извлечь из неё, с существующими записями; далее в зависимости от требуемого действия, либо произвести считывание при наличии объекта, либо осуществить запись при отсутствии объекта.

Если существует такая необходимость, то в описанных ранее каталогах можно добавить дополнительный функционал (например, работа со сторонними api для добавления оплаты, или получения информации).

Основная сложность написания подобного чат-бота – обработка нестандартных ситуаций и сообщений, необходимая для: проверки записываемых в БД данных; фильтрации доступа к функционалу в зависимости от «роли» пользователя и т.д.

Чат-бот для работы с интернет-магазином разрабатывался в среде PyCharm 2023 на языке программирования Python (версия 3.11). Разработанный чат-бот предоставляет простой интерфейс для работы с базой данных (см. рисунки 2, 3, 4, 5) и возможностью расширения функционала для взаимодействия с требуемыми ресурсами в

зависимости от нужд и запросов заказчика. Упомянутый шаблон чат-бота позволяет создать свои методы для взаимодействия с пользователем и работы с базой данных. Приведённый в работе пример реализует регистрацию пользователей при начале работы с чат-ботом, а также позволяет пользователю сохранять в БД свои магазины для постановки их на мониторинг.

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Какую библиотеку на Python выбрать для создания телеграмм-бота? [Электронный ресурс] – URL: <https://habr.com/ru/companies/otus/articles/771110/> (дата обращения: 04.03.2025).
2. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1. [Электронный ресурс] – URL: <https://habr.com/ru/articles/282764/> (дата обращения: 04.03.2025).
3. SQLAlchemy 2.0 для новичков. [Электронный ресурс] – URL: <https://habr.com/ru/articles/848592/> (дата обращения: 05.03.2025).
4. Бот-тестировщик. Автоматизация тестирования интеграции с Telegram ботом с помощью Telethon. [Электронный ресурс] – URL: <https://habr.com/ru/articles/723342/> (дата обращения: 08.03.2025).
5. Пишем и тестируем миграции БД с Alembic. Доклад Яндекса. [Электронный ресурс] – URL: <https://habr.com/ru/companies/yandex/articles/511892/> (дата обращения: 09.03.2025).

УДК 004.65

**ТОБРАТОВ Ю.М., КОШЕЛЕВА М.С.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

### **ОБОБЩЕННОЕ ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ОБРАБОТКИ ЗАКАЗОВ С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА**

*Рассматривается обобщенное представление информационной системы (ИС) обработки заказов, описывается один из возможных подходов к построению данной ИС с использованием нейросетей.*

Успешная деятельность любой организации тесно сопряжена с информационной инфраструктурой предприятия. Принципы ее построения зависят от ряда факторов, среди которых сфера и специфика деятельности компании, ее цели, размеры.



В настоящее время возрастающий интерес современных предприятий и организаций к повышению их конкурентоспособности за счет внедрения информационных систем, обеспечивающих совершенствование системы управления и оптимизации бизнес-процессов является актуальной задачей.

Развитие информационных технологий становится ядром преобразований в компании, а рынок информационных систем остается одним из самых быстрорастущих, предлагая все новые варианты построения информационной инфраструктуры. В настоящее время наиболее востребованным решением является применение технологий искусственного интеллекта, в частности нейросетей.

Нейросети представляют собой математические модели, построенные по принципу работы человеческого мозга. Они способны обучаться на основе данных и выполнять различные задачи: от классификации и прогнозирования до генерации контента [3].

Основные типы нейросетей, используемые в разработке ИС:

- Convolutional Neural Networks (CNN) – для обработки изображений и видео.
- Recurrent Neural Networks (RNN) – для работы с временными рядами и текстами.
- Deep Belief Networks (DBN) – для задач кластеризации и классификации.
- Generative Adversarial Networks (GAN) – для генерации данных.

Процесс внедрения нейросетевых технологий в ИС можно реализовать двумя способами – использование существующих нейросетей и разработка собственной, которая учитывает индивидуальные требования и специфику решаемой задачи. В общем виде процесс внедрения включает несколько этапов:

1. Анализ требований:

- определение задач, которые должна решать система;
- оценка объема и качества доступных данных;
- выбор архитектуры нейросети.

2. Сбор и подготовка данных:

- формирование датасета;
- очистка и предобработка данных;
- разметка данных (при необходимости).

3. Разработка модели:

- выбор фреймворка (TensorFlow, PyTorch, Keras);
- создание архитектуры нейросети;
- настройка гиперпараметров.

#### 4. Обучение и тестирование

- Разделение данных на обучающую и тестовую выборки
- Обучение модели
- Оценка качества работы

#### 5. Интеграция в ИС

- Разработка API для взаимодействия
- Оптимизация производительности
- Обеспечение безопасности

В качестве примеров использования нейросетей в информационных системах можно выделить:

- Системы рекомендаций (e-commerce, медиа)
- Чат-боты и виртуальные ассистенты
- Анализ текстов и обработка естественного языка
- Распознавание изображений и видеоаналитика
- Прогнозирование и анализ временных рядов

При разработке ИС с использованием нейросетей необходимо учитывать следующее:

- Большие объемы данных для обучения
- Высокие требования к вычислительным ресурсам
- Сложность интерпретации результатов работы
- Проблемы с объяснимостью принимаемых решений
- Вопросы безопасности и конфиденциальности данных

Рассмотрим на примере системы обработки заказов, процесс создания ИС, которая облегчит и повысит эффективность работы сотрудников предприятия, за счет внедрения нейросети в такие процессы как:

- анализ работы продавцов (управление отношениями с клиентами и оптимизация отдела продаж);
- оценка надежности поставщиков, основанная на данных работы с ними;
- прогнозирование спроса и оптимизация загрузки склада продукцией [1].

Разрабатываемая система должна быть реализована на основе web-приложения – клиент-серверного приложения, в котором клиентом выступает интернет-браузер, а сервером – web-сервер. А также при помощи вспомогательных программных средств, которые предназначены для автоматизированного выполнения каких-либо действий на web-серверах [2].

Основное преимущество web-приложений заключается в том, что все логические модели и алгоритмы расположены на сервере, а это значит, что клиенты не зависят от конкретной операционной системы

пользователя, поэтому web-приложения являются межплатформенными сервисами. Так как имеется только одна рабочая копия приложения, его намного проще распространять среди пользователей. Пользователю не нужно тратить ресурсы своего компьютера для выполнения обработки результатов, с которыми работает программа. Пользователь получает только то, что ему нужно для работы – интерфейс. Все остальное выполняется на сервере.

Исходя из перечисленного, руководители предприятий заинтересованы во внедрении автоматизированной системы обработки заказов на производстве, так как это сокращает расходы, минимизирует время и улучшает качество обслуживания.

Основной задачей, разрабатываемой ИС, является обработка заказов предприятия, а также контроль выполненной работы сотрудников. В общем виде требования к функциональности системы таковы:

- Оформлять заказы клиентов;
- Обрабатывать принятые заказы;
- Вести контроль работы сотрудников.

Web-приложение использует двухзвенную модель архитектуры клиент-сервер распределенная функция (рисунок 1).

В данной модели ИС общая часть прикладных функций реализуется на сервере, а некоторые специфические функции обрабатываются на клиенте, то есть часть алгоритмов обработки данных выполняется на сервере с помощью хранимых процедур, а более сложные алгоритмы реализуются средствами систем общего назначения. Данная модель считается наиболее оптимальной.

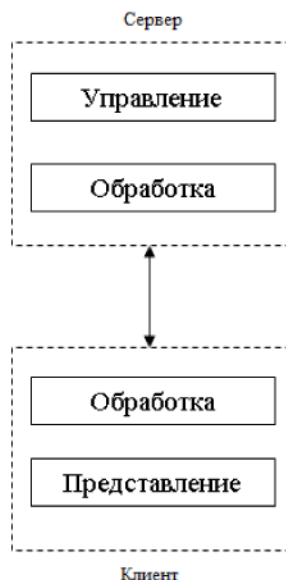


Рисунок 1 – Модель распределенная функция



Специфика работы любого предприятия, имеющего производство – заказы клиентов. От правильного выполнения заказа зависит, будет ли продолжать сотрудничество клиент с предприятием. Условно выполнение заказа можно разделить на следующие этапы:

- Оформление заказа продавцом;
- Сборка заказа сотрудником склада;
- Доставка заказа курьером.

Для того чтобы качественнее и быстрее выполнять заказы, необходимо автоматизировать этапы, представленные выше. К тому же для предприятий важно не только быстро и в срок осуществлять выполнение заказов, а также производить их контроль. В результате можно выявить несколько ролей пользователей – то есть наборов прав и интерфейсов, которые различаются по своему функциональному назначению (таблица 1).

Таблица 1 – Роли пользователей

Роль	Функции исполнителя
Менеджер по персоналу	Имеет возможность редактировать базу сотрудников, следить за работой персонала, просматривать историю заказов
Продавец	Имеет возможность редактировать базу клиентов, оформлять заказы, просматривать стадию выполнения заказа.
Сборщик заказа (сотрудник склада)	Имеет возможность редактировать базу товаров, просматривать информацию о заказе, менять статус заказа.
Доставщик (курьер)	Имеет возможность просматривать информацию о заказе, менять статус заказа.

Функциональность системы рассмотрим с помощью диаграммы прецедентов (Use Cases) (рисунок 2).

Разработка диаграмм прецедентов преследует следующие цели:

1. Определить общие границы и контекст моделируемой предметной области;
2. Сформулировать общие требования к функциональному поведению проектируемой системы;
3. Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.

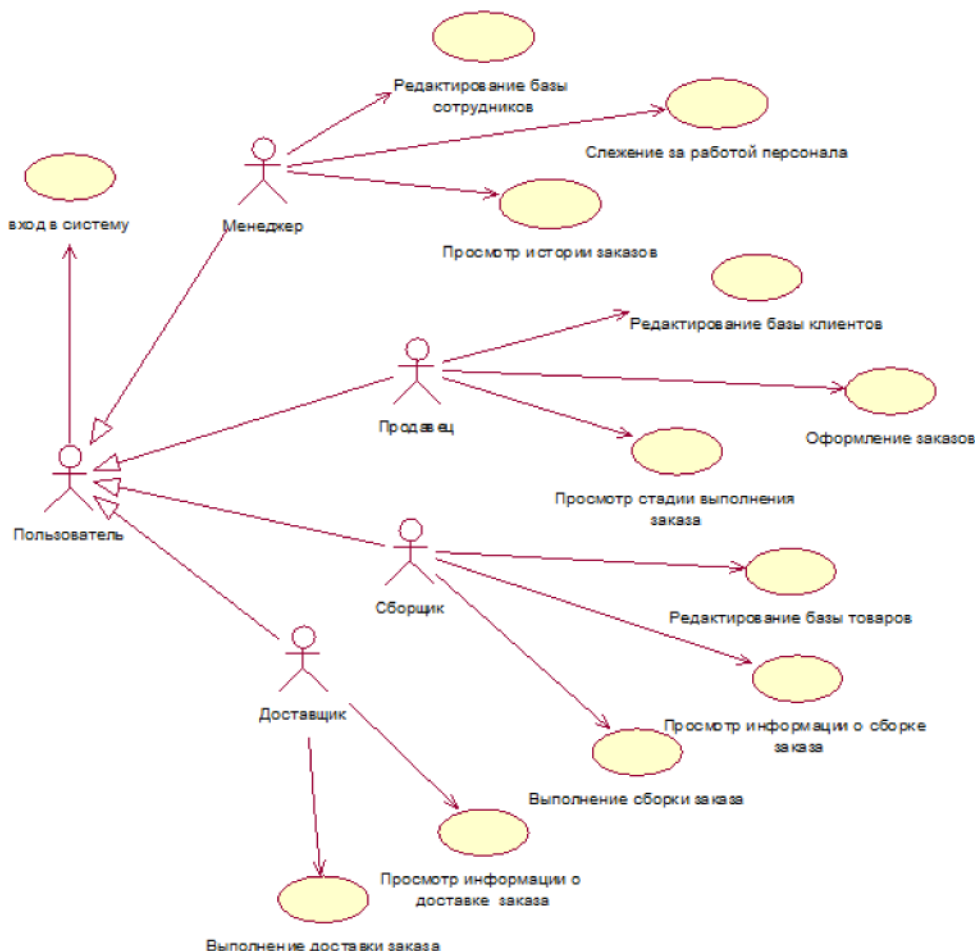


Рисунок 2 – Диаграмма прецедентов

### Заключение

В данной статье была предложена обобщенная модель информационной системы обработки заказов с применением нейросетей. Внедрение технологий искусственного интеллекта, в частности нейросетей, позволяет оптимизировать производственные процессы и снижать затраты, а также открывать новые возможности для развития предприятий.

### СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Баламирзоев, А. Г. Интеллектуальные информационные системы: учебное пособие / А. Г. Баламирзоев. – Махачкала: ДГПУ, 2023. – 136 с.
2. Калгина, И. С. Интеллектуальные информационные системы: учебное пособие / И. С. Калгина. – Чита: ЗабГУ, 2023. – 123 с.
3. Филиппов, Ф. В. Нейросетевые технологии: учебное пособие / Ф. В. Филиппов. – Санкт-Петербург: СПбГУТ им. М.А. Бонч-Бруевича, 2020. – 129 с.

УДК 004.65

**ХАЛДЕЕВА М.А.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

## **ОСОБЕННОСТИ РАЗРАБОТКИ КЛИЕНТСКОЙ ЧАСТИ ИНТЕРНЕТ – МАГАЗИНА**

*Рассматриваются ключевые аспекты разработки клиентской части веб-приложений. Описаны типовые решения для обработки событий и адаптивного дизайна.*

В современном мире веб – приложения стали основным инструментом взаимодействия пользователей с цифровыми сервисами. Их роль сложно переоценить – они обеспечивают доступ к информации, упрощают коммуникацию и автоматизируют ключевые процессы.

Сегодня разработка веб – приложений превратилась в обязательный навык для IT–специалистов. Особое значение имеет клиентская часть (frontend), которая непосредственно формирует пользовательский опыт и определяет успех всего продукта.

Базовый стек клиентской части включает в себя минимальный набор технологий, необходимых для создания пользовательского интерфейса. К таким технологиям относятся: HTML, CSSи JavaScript.

HTML (HyperTextMarkupLanguage) – это язык гипертекстовой разметки текста. Именно с его помощью создается основа страницы веб-сайта.

CSS (CascadingStyleSheets) – каскадные таблицы стилей. Язык иерархических правил, предназначенный для визуального оформления веб-сайта. То есть объекты размещаются на странице с помощью HTML, а CSSотвечает за то, как эти объекты выглядят.

JavaScript – это язык управления элементами веб – страницы. На стороне клиента сценарии JavaScriptобеспечивают эффективное взаимодействиепользователя с интерфейсом страницы и страницы с сервером [1].

Разработка интернет – магазина, в отличие от веб – сайта, требует наличия дополнительных функций: корзина, оплата товара и доставка, система управления заказами. Чаще всего обычный веб – сайт просто отображает информацию, например, статьи, каталог товаров, портфолио или контакты организации. Они более простые в разработки, так как не нужны системы оплаты или хранения товаров. Однако в обоих случаях важно сделать удобный дизайн, а для

магазина еще и обеспечить бесперебойную работу всех процессов покупки.

Рассмотрим разработку на примере создания книжного интернет – магазина.

Разработка клиентской части интернет – магазина книг требует особого внимания к удобству поиска и навигации, например, поиск по названию книги, ее автора или по названию издания. Также немаловажным фактором является адаптивность под мобильные устройства, так как многие предпочитают выбор книг не только с компьютеров, но и с телефонов или планшетов.

Структура пользовательского интерфейса состоит из специализированных компонентов, каждый из которых выполняет отдельную функцию. Выделим основные из них:

Одним из таких элементов является хедер (header или «шапка сайта») – верхняя часть веб – сайта, на которой располагается информация, общая для всей веб – страницы. Хедер помогает пользователям быстро находить необходимую информацию, задает тематический вектор и делает сайт узнаваемым. Header обычно содержит логотип, название компании, слоган, навигационные кнопки для быстрого перемещения по сайту, поисковую строку, а также кнопки корзины, избранного и входа в личный кабинет. Для того, чтобы избежать перегруза основного меню, корзина, избранное и личный кабинет располагают справа. Наличие тех или иных элементов хедера зависит от предназначения сайта, то есть, если сайт не для какого – либо магазина, то кнопки корзина и избранное не нужны. При всей, на первый взгляд, простоте «шапки» сайта, не стоит забывать о том, что header – это первое, на что обращает внимание пользователь, поэтому не стоит перегружать его излишними кнопками или изображениями.

В нашем примере хедер содержит название интернет – магазина, поисковую строку, кнопки корзины, избранного и входа в личный кабинет, а также кнопки навигации «Акции», «Распродажа» и «Популярное» (рисунок 1).

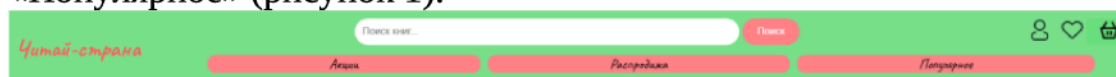


Рисунок 1 – Header

Приведем html – код для элемента Header.

```
<header>
<h1>Читай-страна</h1>
</header>
<div class="search-bar">
<input type="text" placeholder="Поиск книг..." />
```

```

 <button>Поиск</button>
 </div>
 <div class="nav">
 <imgsrc="Profile.png"
alt="Избранные товары">
 <imgsrc="favorite.png"
alt="Избранные товары">
 <button class="open-modal-btn"><imgsrc="basket.png"
alt="Profile"></button>
 </div>
 <div class="grid-section">
 <div class="grid-item">
 <div class="dropdown">Акции
 <div class="dropdown-content">
 Чёрная пятница
 В память Достоевскому
 100 лет русской литературы
 </div>
 </div>
 <div class="grid-item">Распродажа</div>
 <div class="grid-item">Популярное</div>
 </div>

```

#### Стилизация элемента Header:

```

header {
 background: #68DE7C;
 opacity: 0.8;
 color: #ff3d43;
 padding: 10px;
 text-align: left;
 position: fixed;
 top: 0;
 width: 100%;
 z-index: 1000;
}
.nav {
 position: fixed;
 top: 10px;
 right: 10px;
 z-index: 1001;
}
.navimg {
 width: 30px;
 margin-left: 10px;
 z-index: 1001;
}
.search-bar {

```

```
 position: fixed;
 top: 10px;
 right: 450px;
 text-align: center;
 z-index: 1001;
 }
 .search-bar input {
 padding: 10px;
 width: 500px;
 border-radius: 20px;
 border: none;
 }
 .search-bar button {
 padding: 10px 20px;
 border-radius: 20px;
 border: none;
 background-color: #FF8085;
 color: white;
 cursor: pointer;
 }
```

Следующим рассмотрим компонент Footer («подвал»). Это нижняя часть сайта, содержащая в себе контактную информацию, ссылки на социальные сети компании, адрес, авторские права и информацию об авторе и т.д.

В примере Footer содержит адрес электронной почты и номер телефона (рисунок 2).

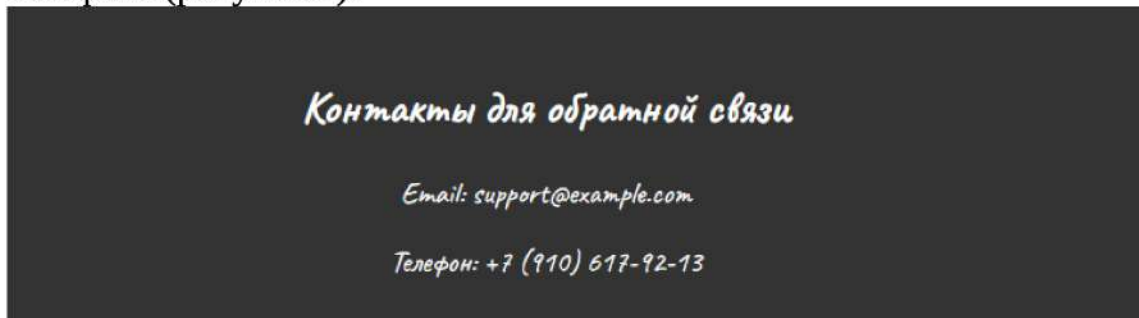


Рисунок 2 – Footer

Html – код для Footer приведен ниже.

```
<footer>
<h2>Контакты для обратной связи</h2>
<p>Email: support@example.com</p>
<p>Телефон: +7 (910) 617-92-13</p>
</footer>
```

Стилизация Footer:

```
footer {
 background-color: rgba(51, 51, 51, 1);
 color: white;
```



```
padding: 20px;
text-align: center;
position: relative;
bottom: 0;
width: 100%;
margin-top: auto;
}
```

Главная страница веб – сайта – это первая страница, которая видна пользователям, посетившим сайт. Именно с нее пользователь может перемещаться по необходимым вкладкам, узнавать о проходимых акциях. Дизайн главной страницы сайта должен привлекать внимание целевой аудитории и стимулировать ее взаимодействие с сервисом [3]. Для этого подбирается контрастная цветовая гамма, используются различные рекламные слайдеры и другие визуальные элементы.

На рисунке 3 изображен пример использования рекламного слайдера.

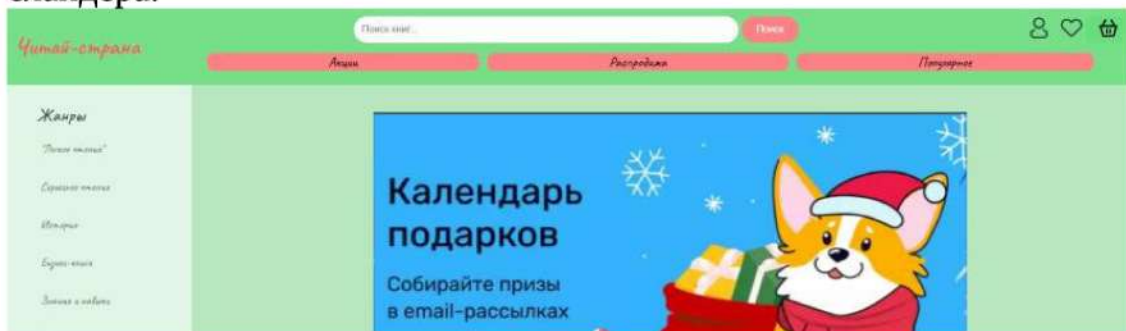


Рисунок 3 – Рекламный слайдер

Пример html – кода для использования слайдеров.

```
<main>
<div class="flex-section">
<div class="slider">
<div class="slides">
<div class="slide"><imgsrc="Слайдер1.png"></div>
<div class="slide"><imgsrc="Слайдер2.png"></div>
<div class="slide"><imgsrc="Слайдер3.png"></div>
</div>
</div>
```

Стилизация слайдеров:

```
.slider {
 overflow: hidden;
 position: relative;
 margin-top: 100px;
 max-width: 802px;
 margin-left: 180px;
}
```

```

.slider .slides {
 display: flex;
 animation: slideAnimation 15s infinite;
}
.slider .slide {
 display: flex;
 justify-content: center;
 align-items: center;
 font-size: 24px;
 color: white;
 background-color: #333;
 flex-shrink: 0;
width: 100%;
}
@keyframes slideAnimation {
 0% { transform: translateX(0); }
 25% { transform: translateX(0); }
 33% { transform: translateX(-100%); }
 58% { transform: translateX(-100%); }
 66% { transform: translateX(-200%); }
 91% { transform: translateX(-200%); }
 100% { transform: translateX(0); }
}

```

Каждый интернет – магазин содержит каталог товаров, в нашем случае книг. Каталог должен представлять собой перечень всех доступных к заказу книг, их названия, авторов, стоимость и обложку (рисунок 4).



Рисунок 4 – Каталог товаров

Пример html – кода для создания каталога книг:



```

 <main>
 <div class="flex-section">
 <divclass = "flex - item"><imgsrc =
"book1.png"width = "150px"height = "150px">
Цена:
1,500.00P
Автор: ЛевТолстой

 Название: Война и мир
Жанр: Художественная
литература</div>
 <divclass = "flex - item"><imgsrc = "book2.png"
width = "150px" height = "150px">
Цена:
1,200.00P
Автор: Федор Достоевский

 Название: Преступление и наказание
Жанр:
Художественная литература</div>
 <divclass = "flex - item"><imgsrc = "book3.png"
width = "150px" height = "150px">
Цена:
1,000.00P
Автор: Михаил Булгаков

 Название: Мастер и Маргарита
Жанр:
Художественная литература</div>
 </div>
 </main>

```

Стилизация каталога книг:

```

.flex-section {
 display: flex;
 justify-content: space-between;
 margin-right: 00px;
 flex-basis: 150px;
 flex-wrap: wrap;
}
.flex-item {
 flex: 1 1 30%;
 margin: 10px;
 font-size: 12pt;
 background-position: center;
 background-size: contain;
 min-height: 600px;
 padding: 15px;
 display: flex;
 flex-direction: column;
 align-items: center;
 justify-content: space-between;
 background-color: #fff;
 border-radius: 10px;
 box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}
.flex-item img {
 max-width: 250px;
 height: auto;
 margin-bottom: 20px;
}

```

```
 object-fit: cover;
 border-radius: 5px;
 box-shadow: 0 2px 5px rgba(0,0,0,0.1);
 }
```

Для взаимодействия пользователя с веб-страницей используется JavaScript. Рассмотрим его применение на корзине товаров. Для этого создадим файл с расширением .jsи подключим его к уже существующему htmlфайлу.

Код в файле .js:

```
// Получаем элементы модального окна
const modal = document.createElement('div');
modal.className = 'modal';
modal.innerHTML = `
<div class="modal-content">
×
<h2>Корзина</h2>
<p>Ваша корзина пуста</p>
</div>
`;
document.body.appendChild(modal);
const openModalBtn = document.querySelector('.open-modal-btn');
const closeBtn = document.querySelector('.close');
// Открываем модальное окно при клике на кнопку
openModalBtn.onclick = function() {
 modal.style.display = "block";
}
// Закрываем модальное окно при клике на крестик
closeBtn.onclick = function() {
 modal.style.display = "none";
}
// Закрываем модальное окно при клике вне его области
window.onclick = function(event) {
 if (event.target == modal) {
 modal.style.display = "none";
 }
}
Html - код для подключения файла .js:
<script src="script.js"></script>
```

Для написания клиентской части интернет - магазина можно использовать гораздо больше компонентов, однако достаточно и основных рассмотренных элементов.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Сапрыкин А.Н. Основы веб-программирования: HTML 5, CSS 3, JavaScript. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2025. –248 с.

2. Горбаров, И. Как оформить главную страницу сайта [Электронный ресурс] – URL: <https://atwinta.ru/material/blog/kak-oformit-glavnuju-stranitsu-sajta/> (дата обращения: 30.03.2025).

УДК 004.051

**ЦХОВРЕБОВ Р.А.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

**АНАЛИЗ СОВРЕМЕННЫХ ПОДХОДОВ К ОЦЕНКЕ  
ХАРАКТЕРИСТИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ  
ИНФОРМАЦИОННЫХ СИСТЕМ**

*В работе рассматриваются и анализируются современные подходы к оценке характеристик программного обеспечения информационных систем. Анализируются их преимущества и недостатки.*

Программное обеспечение (ПО) – крайне важная часть нашего современного мира. Некоторые люди, даже не задумываясь, используют разнообразные виды программного обеспечения, сталкиваются с ними на ежедневной основе, будь то простые приложения на мобильном телефоне, соц. сети или даже связанные с медициной технологии. Развиваясь и автоматизирую всё больше и больше аспектов наших жизней, ПО удерживает всё большее и большее количество важнейших инфраструктур, от чего мы, как разумные люди, должны поддерживать его в наилучшем состоянии, предотвращая какие-либо сбои, ещё до их очевидного проявления. Именно по этой причине, ещё на ранних стадиях создания ПО, мы задумываемся о его эффективности, рациональности, проще говоря, даём объективную оценку.

Задача оценки программного обеспечения представляет собой важный этап в его создании. Если рассматривать программу с самого её зарождения, то человек способен искоренить ошибку и тем самым предотвратить её влияние на всю систему, в дальнейшем. Также оценка уже готового продукта помогает узнать, является ли он максимально эффективным или его возможно улучшить. Так же нельзя забывать о стороне заказчика, благодаря данной оценке он способен по

достоинству оценить качественную работу. Но, разумеется, всё не так просто, для оценки ПО не существует универсальной формулы, по этой причине приходится руководствоваться разнообразными результатами и делать на их основе самостоятельные выводы.

Для решения поставленной задачи широко используются алгоритмы, которые способны оценивать не только ошибки в коде, но и эффективность его алгоритмов. Также экспертами применяются математические модели и статистические подходы. Данная оценка позволяет говорить об эффективности в различных условиях применения, а также выявлять ошибки и уязвимости программы. Ещё нельзя забывать о сравнении разнообразных версий, как устаревших, так и конкурирующих. Благодаря подобному анализу можно отметить как усовершенствованные, так и усугублённые части ПО, над которыми стоит работать в будущем. Для более глубокого изучения темы и дальнейшего анализа, мы постараемся понять, в чём заключается задача каждого из методов анализа рассмотрим существующие, современные, методы, оценим их эффективность, также выявим сильные и слабые стороны каждого.

Оценка программного обеспечения методами функциональной оценки, или же Function Point Analysis, считаются одними из наиболее часто применяемых. Но в чём же заключается концепт работы? Суть в том, что данный метод оценивает количество функций, которые выполняет программа. Оценка осуществляется путём анализа необходимых требований к ПО, в последствии выделяя следующие функции:

1. Запросы к базе данных;
2. Ввод данных;
3. Вывод данных;

Проанализировав метод, можно прийти к выводам, что благодаря ему легко оценить стоимость и размер проекта, основываясь на количестве функций, а также сравнить версии продукта, как от одного, так и от разных производителей. Но также удалось выделить минусы, в число которых входит невозможность метода учесть сложность самого алгоритма, а также отсутствие оценки качества кода. Общие впечатления от метода остались нейтральными, он выполняет заданные им функции на достаточно высоком уровне, особенно его полезность может выделяться в процессе разработки, для контроля рисков, но отсутствие дополнительных функций заставляет прибегнуть к иным методам, в синергии с текущим.

Далее мы рассмотрим методы оценки качества программного обеспечения. Они являются одними из важнейших методов, ведь

именно благодаря работе с ними мы можем выявлять разнообразные проблемы, которые появляются в процессе разработки или по её окончанию, закономерно приводя к дальнейшему улучшению нашего продукта.

Самым распространённым методом оценки качества ПО считается тестирование, будь то ручное, с использованием специальных инструментов или автоматическое. Каждый метод помогает разработчику, и заказчику с поиском недоработок и неточностей в программе благодаря чему ошибки будут исправлены до полномасштабного выхода программы на рынок. Практика тестирования считается обязательной для всех видов ПО, но её соблюдение не всегда приводит к желанным результатам, как показывает современная практика. Многие ПО выходят в свет после долгих месяцев тестирования, однако их качество оставляет желать лучшего. Многие считают, что данная практика появилась лишь после развития сети интернет, ведь разработчик может обновлять продукт на протяжении всего существования программы и исправлять её ошибки. Причина такой практики может быть как и человеческий фактор, так и корпоративная жадность, со спешкой разработки и желанием выпустить продукт как можно скорее.

Следующим по важности считается классическая проверка кода, как следует из названия, данный метод подразумевает полный анализ системы, сравнивая ту с нормами и стандартами написания кода, разумеется данный анализ зависит как от классических правил, так и от используемого языка, библиотек, функций и так далее. Благодаря этому методу выявляются частые проблемы с производительностью и безопасностью. Данной проверкой пренебрегать не стоит, даже если предположить, что более 70% пользователей ПК обладают техникой с достойными характеристиками, ни в коем случае не стоит закрывать глаза на безопасность данных, как заказчика, так и пользователей.

Последним, но не менее важным методом оценки качества ПО является анализ требований. Во многом данный анализ зависит от заказчика, а именно от выставленных им требований. Данный анализ включает в себя проверку на соответствие бизнес-целям, адекватность и полноту [1].

Изучив данные виды методов, можно прийти к выводу, что они являются невероятно полезными и предоставляют как разработчику, так и заказчику контроль над кодом. Они обеспечивают первого информацией для улучшения и обеспечения качества продукта, когда второго возможностью влиять на процесс анализа. Каждый перечисленный метод имеет возможность улучшить положение

проекта в той или иной степени, но в ином никаких глобальных минусов, по окончанию анализа, выявлено не было. Однако хотелось бы отметить присутствие человеческого фактора, который может сыграть роль в тестировании.

Разумеется каждый профессионал должен знать цену своей работе, что бывает сложно, но данная задача облегчается, при использовании модели издержек разработки. Данная модель представляет собой алгоритм для оценки ПО, а именно его стоимости, на основе таких характеристик как:

- Характеристика продукта;
- Характеристика аппаратного обеспечения;
- Характеристика персонала;
- Характеристика проекта;

Каждая из них содержит дочерние характеристики, которые обеспечивают наиболее точную оценку ценности проведённой работы. Формула модели представляет собой следующий вид:

$$E = a_i (KLoC)^{b_i} * PFT$$

где E – трудоёмкость разработки ПО в месяцах, KLoC – оценочный размер программы исчисляющийся в тысячах строках самого кода, а так же PFT (Регулирующий Фактор). В свою очередь значения  $b_i$  и  $a_i$  являются коэффициентами модели, зависящий от типа проекта. Их значения приведены в данной таблице.

Таблица 1 – Значения коэффициентов модели

Тип проекта	$a_i$	$b_i$
Органический	3.2	1.05
Полуразделенный	3.0	1.12
Встроенный	2.8	1.20

Сама методика делится на 3и уровня, “Базовый уровень”, “Средний уровень” и “Детальный уровень”. Процедура анализа усложняется с каждым последующим уровнем, а характеристика между средним и детальным уровнями не отличаются, по этим причинам мы рассматриваем “Средний уровень” и проанализировав его работу и необходимые характеристики, можно прийти к выводу, что метод является крайне эффективным, имеет большой спектр настроек и берёт в расчёт большое количество характеристик, благодаря чему его результаты бывают максимально точными, однако из минусов можно отметить сложность работы с повышенными уровнями, хотя он нивелируется наличием базового уровня анализа, который жертвует точностью, взамен на удобство.

С точки зрения метрик выделяют пять характеристик объектно-ориентированных систем: локализацию, инкапсуляцию, информационную закрытость, наследование и способы абстрагирования объектов. Эти характеристики оказывают максимальное влияние на объектно-ориентированные метрики.

Так мы постепенно пришли к одному из наиболее глобальных методов оценки, а именно метод оценки и анализа проектов. Как нам известно, разработка – это глобальный проект, который требует разнообразных ресурсов, а главное времени. Данный метод позволяет оценить время работы, а также заняться планированием и графиком работы над проектом. Его основное применение сводится к проектам, которые не имеют точных граней и деталей, благодаря чему он помогает структурировать работу и помогает с организацией. Самой популярной частью данной идеи является метод критического пути, который строится на построении сетевой диаграммы.

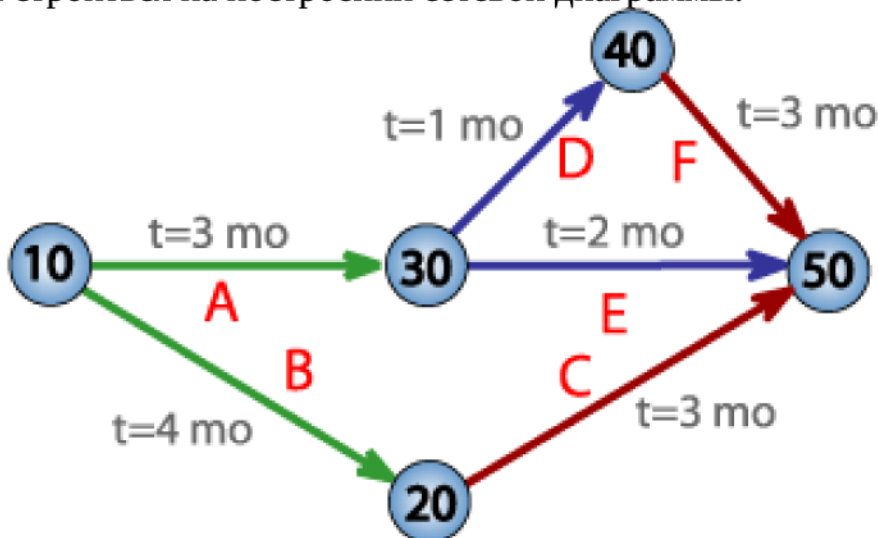


Рисунок 1 – Пример сетевой диаграммы

Выше приведён пример сетевой диаграммы для проекта с продолжительностью в семь месяцев, пятью промежуточными точками и шестью деятельностью. Он представляет собой диаграмму, которая помогает ориентироваться в деятельности относительно порядка, затраченного времени и зависимостей друг от друга. Основным плюсом данного метода является предоставляемая им информация о времени разработки, а также создаваемая диаграмма, но явно можно отметить отсутствие зависимости от некоторых факторов, которые могли бы кардинально изменить предоставляемые им данные, от чего результат применения данной методики может варьироваться от невероятно точного, до крайне сомнительного.

Таким образом, вы могли заметить, что существует множество способов оценки ПО, а каждый имеет свои лучшие и худшие стороны, таким образом мы стоим перед выбором методов оценки, для чего и можно воспользоваться сравнительным анализом. Сравнительный анализ методов оценки ПО позволяет подобрать наиболее подходящий метод для нашей задачи, поэтому мы кратко разберем и сравним описанные нами методы.

Метод функциональной точки - основываясь лишь на количестве функций, которые должно выполнять ПО, данный метод сталкивается с недостатком отсутствия возможности считывать качество и сложность решаемой задачи.

Метод оценки ПО по цене – является невероятно удобным и точным, но неопытный пользователь может столкнуться со сложностями при применении.

Методы функциональной оценки – позволяют дать оценку качества программы в течении работы, так и по её окончанию, избегая множества ошибок перед релизом, однако каждый имеет свои специфические методы применения, а также не застрахован от вмешательства человеческого фактора, куда больше чем остальные.

Метод оценки по времени выполнения – позволяет с высокой точностью определить время, которое будет потрачено на проект, однако не учитывает некоторые ключевые факторы, от чего может выдавать ошибочный результат.

В итоге, можно выделить лишь одно, не существует универсального метода, который мог бы дать все необходимые данные для разработчика и заказчика, по этой причине необходимо работать с ними и использовать как инструменты, чем они и являются. При верном использовании, подборе подходящих методов и адаптации их под свои нужды, они могут выдавать высококачественные результаты, которые будут помогать в разработке с самого начала, предоставлять критическую информацию и просто облегчать работу.

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Вигерс Карл, Битти Джой. Разработка требований к программному обеспечению. – СПб.: БХВ-Петербург, 2014. – 738 с.



УДК 004.048

**ШМАКОВ А.Ю.**Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина**МЕТОДЫ FEW-SHOT LEARNING ДЛЯ РАСПОЗНАВАНИЯ  
ОБЪЕКТОВ ПРИ ОГРАНИЧЕННЫХ ОБУЧАЮЩИХ ДАННЫХ**

*Рассматриваются подходы Few-Shot Learning для задач компьютерного зрения в условиях дефицита данных. Приводятся примеры из медицинской диагностики и микроскопии.*

**Введение**

Современные методы машинного обучения, такие как глубокие нейронные сети, достигают высокой точности в задачах классификации и распознавания объектов. Однако их эффективность напрямую зависит от объема обучающих данных. В реальных приложениях, таких как медицинская диагностика редких заболеваний или анализ микроскопических изображений, сбор большого количества аннотированных данных часто оказывается затрудненным из-за высокой стоимости, временных ограничений или этических соображений.

**Few-Shot Learning (FSL)**

Few-Shot Learning (FSL) представляет собой альтернативный подход, позволяющий обучать модели на крайне ограниченных наборах данных, состоящих из 1–10 примеров на класс. Этот подход особенно актуален в областях, где данные являются редкими или труднодоступными. Например, в медицине FSL может использоваться для анализа рентгеновских снимков или гистологических препаратов, где количество доступных образцов для редких заболеваний может быть крайне ограничено.

Цель данной работы – рассмотреть основные методы Few-Shot Learning, проанализировать их применимость в задачах компьютерного зрения и продемонстрировать их эффективность на примерах из медицинской диагностики и микроскопии.

Few-Shot Learning решает проблему малых данных за счет использования различных стратегий, таких как метрическое обучение, мета-обучение и трансферное обучение. Рассмотрим основные подходы подробнее.

Метрическое обучение направлено на построение эмбедингов (векторных представлений) данных, которые позволяют эффективно измерять сходство между объектами. Одним из наиболее известных

методов является Prototypical Networks. В этом подходе каждый класс представляется центроидом (прототипом), который вычисляется как среднее значение эмбедингов всех объектов данного класса:

$$c_k = (1 / |S_k|) * \sum_{(x_i, y_i) \in S_k} f_{\theta}(x_i) \quad (1)$$

где  $S_k$  – множество объектов класса  $k$ ,  $f_{\theta}(x_i)$  – эмбединг.

Преимущество этого подхода заключается в его простоте и эффективности, особенно в задачах с малым количеством данных. Однако он может быть чувствительным к шуму в данных и требует тщательной настройки функции расстояния.

#### *Model-Agnostic Meta-Learning, MAML*

Мета-обучение (Model-Agnostic Meta-Learning, MAML) – это подход, который обучает модель адаптироваться к новым задачам за минимальное количество шагов. Идея заключается в том, чтобы найти такую начальную точку параметров модели, которая позволяет быстро обучаться на новых данных.

MAML работает следующим образом:

- Модель обучается на нескольких задачах, каждая из которых содержит небольшой набор данных.
- Для каждой задачи выполняется несколько шагов градиентного спуска, чтобы адаптировать модель к конкретной задаче.
- Градиенты обновляются таким образом, чтобы начальная точка параметров была оптимальной для быстрой адаптации.

Этот подход особенно полезен в задачах, где данные могут сильно различаться между задачами, но имеют общие закономерности.

Трансферное обучение использует предварительно обученные модели, такие как ResNet, которые были обучены на больших датасетах (например, ImageNet). Эти модели затем дообучаются на малых данных для решения конкретной задачи.

Преимущества трансферного обучения включают:

- Снижение требований к объему данных.
- Быстрое обучение благодаря использованию уже извлеченных признаков.

Однако этот подход может быть менее эффективным, если исходные данные значительно отличаются от данных целевой задачи.

#### *Медицинская диагностика*

Распознавание редких патологий на рентгеновских снимках

В работе исследователи применили Few-Shot Learning для анализа рентгеновских снимков грудной клетки. Модель достигла точности 92% при использовании всего 5 примеров на класс. Это

демонстрирует потенциал FSL в диагностике редких заболеваний, где количество доступных данных крайне ограничено.

#### Анализ гистологических препаратов

Гистологические препараты представляют собой сложные изображения, содержащие мелкие структуры. Few-Shot Learning может использоваться для автоматизации анализа этих изображений, что особенно важно для диагностики рака.

#### Микроскопия

##### Классификация клеток с аномалиями

В работе модель на основе ProtoNet показала на 15% более высокую точность по сравнению с традиционными сверточными нейронными сетями (CNN). Это подтверждает, что FSL эффективен для анализа микроскопических изображений, где данные часто представлены в малых объемах.

##### Обнаружение инфекций в биологических образцах

Few-Shot Learning также может использоваться для обнаружения инфекций в биологических образцах, таких как кровь или ткань. Это особенно важно в условиях пандемий, когда требуется быстрый анализ новых типов вирусов.

#### Результаты

Эксперименты проводились на двух датасетах:

- ISIC (дерматология) – содержит изображения кожи для диагностики меланомы.
- BBBC (биология) – содержит микроскопические изображения клеток.

Результаты представлены в таблице 1:

Таблица 1 – Результаты

Метод	Точность	Время обучения (ч)
ProtoNet	89%	1.2
MAML	85%	3.5

Как видно из таблицы, ProtoNet показывает более высокую точность и требует меньше времени для обучения по сравнению с MAML.

#### Заключение

Few-Shot Learning демонстрирует значительный потенциал в задачах распознавания объектов при ограниченных обучающих данных. Он особенно полезен в таких областях, как медицинская диагностика и микроскопия, где сбор больших объемов данных затруднен. Однако существуют и ограничения:

- Необходимость доработки методов для работы с мультимодальными данными.

– Открытые вопросы, связанные с этическими аспектами использования синтетических данных.

В будущем планируется дальнейшее исследование комбинации Few-Shot Learning с другими подходами, такими как генеративные модели, для повышения устойчивости и точности моделей.

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Snell J., Swersky K., Zemel R. Prototypical Networks for Few-shot Learning. NeurIPS, 2017.
2. Finn C. et al. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. ICML, 2017.
3. Chen Y. et al. Few-shot Medical Image Classification with Simple Shape Metrics. IEEE TMI, 2021.
4. Wei J. Few-shot Learning for Cellular Image Analysis. Nature Methods, 2022.
5. He K. et al. Deep Residual Learning for Image Recognition. CVPR, 2016.

УДК004.8:004.738.5

**ШОШНИКОВ С.В.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

### **АВТОМАТИЗАЦИЯ РАЗРАБОТКИ ИНТЕРНЕТ- МАГАЗИНОВ С ПОМОЩЬЮ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА**

*В статье исследуются возможности применения искусственного интеллекта для автоматизации разработки интернет-магазинов, включая анализ таких направлений, как генерация контента, тестирование, оптимизация кода и анализ данных. Приводятся статистика и примеры, иллюстрирующие эффективность ИИ в электронной коммерции.*

Электронная коммерция остается одной из самых быстрорастущих отраслей: по данным Statista, мировой объем рынка e-commerce в 2025 году оценивается в 5–7 триллионов долларов [1]. Разработка интернет-магазинов требует значительных ресурсов для создания контента, тестирования и анализа данных пользователей. Традиционные методы теряют эффективность в условиях сжатых сроков и конкуренции. Искусственный интеллект автоматизирует эти процессы: по оценкам Gartner, к 2025 году ИИ существенно ускорит

веб-разработку [2]. Кроме того, ИИ адаптирует магазины под нужды пользователей, повышая их конкурентоспособность.

Цель статьи – исследовать, как ИИ может оптимизировать разработку интернет-магазинов, опираясь на доступные статистические данные и общие подходы.

#### *Теоретические основы автоматизации с помощью ИИ.*

Автоматизация разработки интернет-магазинов с использованием ИИ базируется на применении нейронных сетей и алгоритмов машинного обучения. Основные направления ИИ включают:

Генеративные модели (например, GPT-3, BERT) – для создания текстового контента и базового кода. Эти модели, основанные на архитектуре трансформеров, способны генерировать связные тексты, адаптированные под заданные параметры, такие как тональность или ключевые слова для SEO-оптимизации. Они также применяются для создания простых программных модулей, таких как формы или обработчики запросов, что сокращает нагрузку на разработчиков и ускоряет процесс интеграции.

Сверточные нейронные сети (CNN) – для обработки изображений товаров. Такие сети эффективны для задач классификации (определение категории товара по фото), сжатия изображений без потери качества и даже автоматической генерации визуальных элементов на основе описаний. Это особенно важно для интернет-магазинов, где визуальная составляющая играет ключевую роль в привлечении покупателей и повышении конверсии.

Алгоритмы анализа данных – для прогнозирования спроса и сегментации аудитории. Методы кластеризации, регрессии и временные ряды позволяют предсказывать пики продаж, определять предпочтения клиентов и оптимизировать ассортимент, что напрямую влияет на доходность магазина.

Теоретически ИИ позволяет сократить время выполнения задач на 50-90% по сравнению с ручным трудом [3]. Например, генерация описаний товаров с помощью нейросетей занимает в среднем 1-2 часа на 100 единиц вместо 20-30 часов при традиционном подходе. Автоматизация тестирования снижает вероятность ошибок на 20-30% за счет анализа больших объемов данных в реальном времени [5]. Анализ поведения пользователей с помощью ИИ выявляет до 15% скрытых закономерностей, недоступных при ручной обработке [7].

Кроме того, ИИ поддерживает адаптивность: модели могут переобучаться под конкретные задачи магазина, такие как генерация текстов в определенном стиле или анализ сезонных колебаний спроса.

Это делает его универсальным инструментом для разработчиков, работающих в условиях ограниченных ресурсов.

Практическое применение ИИ в разработке интернет-магазинов

ИИ активно используется в индустрии e-commerce. Рассмотрим ключевые направления и примеры их реализации с фрагментами кода.

#### *Генерация контента*

Генеративные модели, такие как GPT-3, применяются для создания описаний товаров, заголовков и рекламных текстов. По данным OpenAI, такие инструменты сокращают время подготовки контента на 85% [4]. Пример реализации на Python:

```
from transformers import pipeline
generator = pipeline("text-generation", model="gpt2")
prompt = "Generate a product description for a
wireless mouse"
result= generator(prompt, max_length=100,
num_return_sequences=1)
print(result[0]["generated_text"])
```

Результаты требуют минимальной коррекции (около 10-15% времени), что делает подход экономически выгодным.

#### *1. Обработка изображений с использованием CNN*

Сверточные нейронные сети (CNN) автоматизируют обработку изображений товаров, ускоряя подготовку визуального контента на 70-80% [8]. Они классифицируют изображения (например, распознают категорию товара) и оптимизируют их размер, улучшая загрузку страниц на 20-30%. Пример классификации с ResNet50 на Python:

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing import image
import numpy as np
model = ResNet50(weights='imagenet')
img = image.load_img('mouse_image.jpg',
target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
preds = model.predict(x)
print('Predicted:', preds)
```

#### *2. Автоматическое тестирование*

ИИ-инструменты, такие как Testim или Mabl, используют машинное обучение для анализа интерфейсов и функционала. Согласно отчету Sargemini, автоматизация тестирования сокращает затраты на 30% и время на 40% [5]. Пример проверки страницы корзины на Python:

```
from selenium import webdriver
from time import sleep
driver = webdriver.Chrome()
```

```
driver.get("http://example-shop.com/cart")
sleep(2) # Ожидание загрузки страницы
assert "Checkout" in driver.page_source, "Checkout
button not found"
driver.quit()
```

Процесс занимает 20-30 минут вместо 3-4 часов вручную. Более того, ИИ может анализировать поведение пользователей на сайте, выявляя проблемные зоны, такие как медленная загрузка или некорректное отображение на мобильных устройствах.

### *3. Оптимизация кода*

Инструменты вроде GitHubCopilot, основанные на ИИ, генерируют до 30% кода в проектах веб-разработки [6]. Это ускоряет создание модулей, таких как поиск или авторизация, на 15-25%. Пример реализации поисковой функции на Python с использованием GitHubCopilot:

```
def search_products(query, product_database):
 results = []
 for product in product_database:
 if query.lower() in product["name"].lower() or
query.lower() in product["description"].lower():
 results.append(product)
 return sorted(results, key=lambda x: x["price"])
```

Код создается за 5 минут вместо 20, а сортировка по цене добавляет функциональность. ИИ также помогает в рефакторинге, предлагая более эффективные структуры данных или алгоритмы, что снижает нагрузку на серверы магазина.

### *Анализ данных*

Алгоритмы машинного обучения обрабатывают большие объемы данных (поведение пользователей, продажи), выявляя закономерности. По данным McKinsey, компании, использующие ИИ для анализа, увеличивают конверсию на 10-15% [7]. Пример реализации анализа кликов на Python:

```
import pandas as pd
from sklearn.cluster import KMeans
data = pd.read_csv("user_clicks.csv")
clicks = data[["page", "click_count"]]
kmeans = KMeans(n_clusters=3).fit(clicks)
print("User segments by clicks:", kmeans.labels_)
```

Это позволяет сегментировать аудиторию и адаптировать интерфейс. Например, кластеризация может показать, что 40% пользователей предпочитают быстрый доступ к акциям, что требует выделения соответствующего блока на главной странице.

### *Результаты и преимущества*

Применение ИИ демонстрирует значительные улучшения в разработке интернет-магазинов (Таблица 1). Данные таблицы получены на основе личного опыта автора при реализации проектов разработки интернет-магазинов в период с 2024 г. до марта 2025 г., включая генерацию контента, тестирование и анализ данных для типичного магазина с ассортиментом до 100 товаров.

Таблица 1 – Сравнение традиционного и ИИ-подхода

Задача	Время (традиц.), ч	Время (ИИ), ч	Экономия, %
Генерация контента	24	2	92
Тестирование	4	0.5	88
Анализ данных	12	2	83

Преимущества:

1. Сокращение сроков разработки на 30-50% [2].
2. Снижение затрат на контент и тестирование на 25-40% [5].
3. Повышение точности анализа данных и персонализации.

Ограничения включают необходимость обучения моделей и доработки результатов. Например, сгенерированный контент может не соответствовать бренду, что требует дополнительных ресурсов (около 10% времени). Эти вызовы решаются за счет настройки ИИ и интеграции с человеческим контролем.

Заключение

Искусственный интеллект существенно оптимизирует разработку интернет-магазинов, автоматизируя рутинные процессы и улучшая качество решений. Статистика подтверждает: ИИ сокращает время и затраты, делая разработку доступной даже для небольших компаний. Перспективы развития связаны с полной автоматизацией дизайна, интеграцией ИИ с дополненной реальностью и прогнозированием рыночных трендов. Такие технологии продолжат трансформировать e-commerce в ближайшие годы.

### СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Statista. E-commerce worldwide – Statistics & Facts [Электронный ресурс] – URL: <https://www.statista.com/topics/871/online-shopping/> (дата обращения: 12.12.2024).
2. Gartner. The Future of AI in Development [Электронный ресурс] – URL: <https://www.gartner.com/en/insights/ai-trends> (дата обращения: 12.12.2024).
3. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016. – 800 с.
4. OpenAI. GPT-3 UseCases [Электронный ресурс] – URL:



<https://openai.com/research/gpt-3> (дата обращения: 12.12.2024).

5. Capgemini. World Quality Report 2023 [Электронный ресурс] – URL: <https://www.capgemini.com/insights/research/world-quality-report> (дата обращения: 12.12.2024).

6. GitHub. Copilot Impact Study [Электронный ресурс] – URL: <https://github.blog/copilot-impact> (дата обращения: 12.12.2024).

7. McKinsey. The Value of AI in Retail [Электронный ресурс] – URL: <https://www.mckinsey.com/industries/retail/ai-value> (дата обращения: 12.12.2024).

УДК 004.896

**ЩЕВЬЕВ О.И.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

## **АВТОМАТИЧЕСКАЯ ТРАНСКРИПЦИЯ МУЗЫКИ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ И CQT**

*В статье описывается система автоматической транскрипции музыкальных композиций, основанная на применении глубоких нейронных сетей для преобразования аудиосигнала в нотную запись в формате piano roll.*

### *Введение*

Автоматическая транскрипция музыки (Automatic Music Transcription, АМТ) – это задача преобразования аудиозаписи музыкального произведения в символическое представление (ноты или MIDI). Решение этой задачи важно для музыкального анализа, обучения, поиска и оцифровки архивов. Традиционно транскрипция выполнялась вручную «на слух» или по спектрограммам, что трудоемко и субъективно. Современные методы компьютерного слуха и машинного обучения резко повысили точность и эффективность АМТ.

Основные сложности связаны с многоголосием, перекрытием гармоник различных инструментов и шумами. Для успешного выделения отдельных нот в полифоническом сигнале требуются устойчивые методы спектрального анализа и эффективные алгоритмы распознавания образов. Большой прогресс достигнут благодаря глубоким нейронным сетям, обучаемым на больших размеченных датасетах.

В данной работе описывается система автоматической транскрипции, сочетающая методы спектрального анализа с глубокой

сверточной нейросетью энкодер-декодер и предварительной инструментальной сепарацией на базе модели Demucs. Используются спектрограммы, полученные методом STFT и Constant-Q Transform (CQT). Преобразование CQT выгоднее для музыкальных сигналов, поскольку отражает логарифмическое распределение частот, согласующееся с музыкальной шкалой. Для распознавания нот применяется нейросеть, архитектура которой напоминает U-Net: энкодер-декодер с upsampling и skip connections. Предварительная сепарация Demucs разбивает микс на отдельные дорожки, что упрощает последующую транскрипцию.

#### *1. Обзор методов спектрального анализа*

Для извлечения признаков аудиосигнала применяют кратковременное преобразование Фурье (STFT) и Constant-Q Transform (CQT). STFT разбивает сигнал на короткие окна (обычно с окнами Ханна или Гаусса) и рассчитывает спектр внутри каждого окна. Недостаток STFT – фиксированное частотное разрешение: для низких нот спектральное представление оказывается “размытым”, а для высоких – более узким.

CQT использует логарифмическую шкалу частот, что удобно для музыкальных задач. Ширина полосы меняется в зависимости от частоты, и каждое удвоение частоты (октава) получает одинаковое число бинов. Это естественно соответствует музыкальной шкале и делает спектрограммы более информативными при распознавании нот. Для компактного представления иногда используют хромограмму (12 нотных классов), но в данной работе нужна полноразмерная CQT-спектрограмма, чтобы различать конкретные высоты.

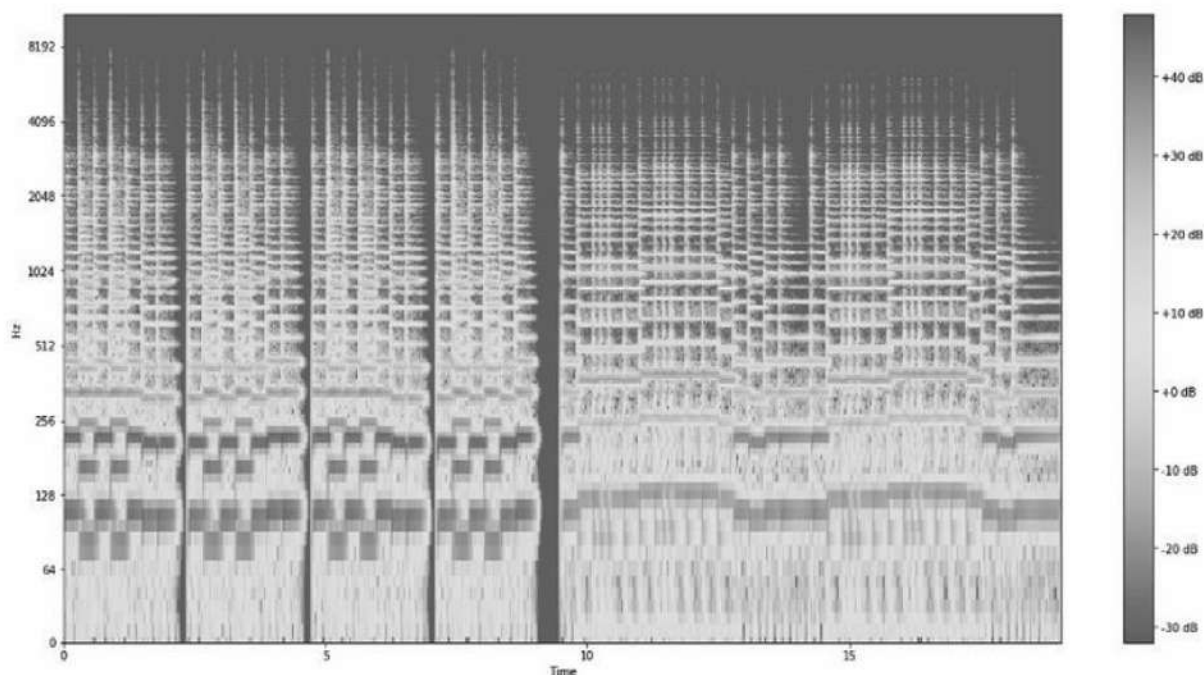


Рисунок 1 – STFT-спектрограмма мелодии «В траве сидел кузнечик»

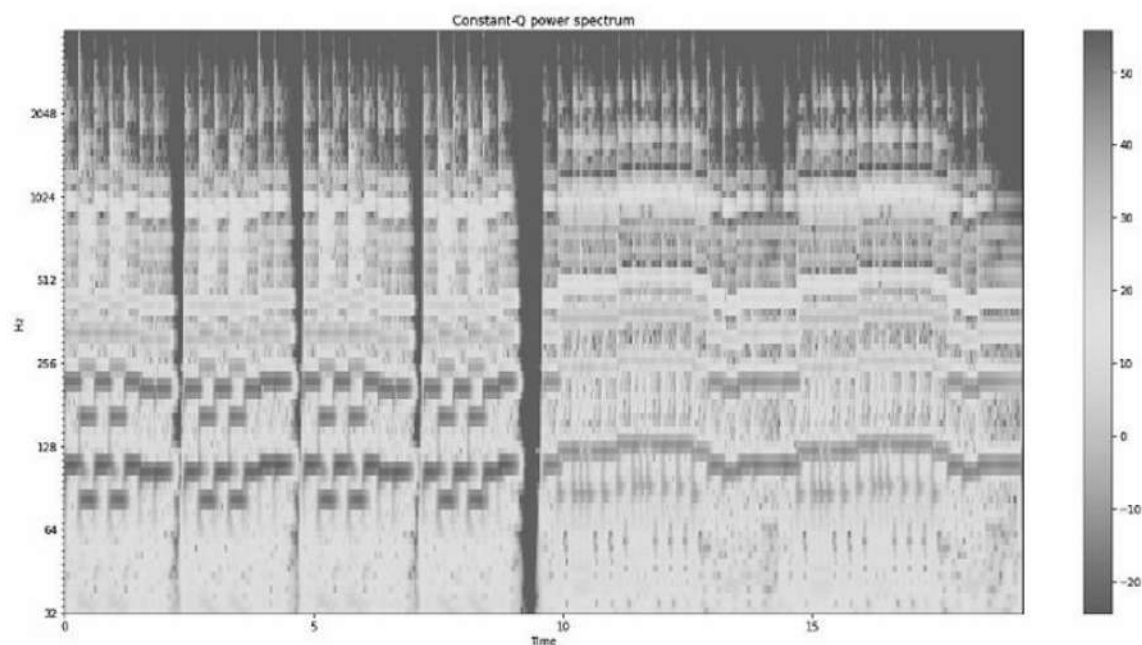


Рисунок 2 – CQT-спектрограмма мелодии «В траве сидел кузнечик»

## 2. Архитектура нейронной сети

Разработанная модель основана на сверточной архитектуре типа энкодер-декодер, сходной с U-Net. Входные данные – двумерный тензор спектрограммы (время × частоты). Энкодер последовательно применяет сверточные блоки с уменьшением размерности (MaxPooling2D). В декодере используется upsampling (или Conv2DTranspose), восстанавливающий исходные размеры. Skip

connections объединяют активации энкодера и декодера, что улучшает выявление одновременных нот и атак.

На выходе – сверточный слой с сигмоидной активацией, возвращающий вероятности звучания каждой ноты в каждый момент времени. При обучении эта выходная матрица сравнивается с эталонной «пианолой» (binary piano-roll). Ошибка рассчитывается по двоичной кроссэнтропии.

Ниже приведён упрощённый пример кода (на TensorFlow) для иллюстрации структуры сети. (Параметры глубины сети, число фильтров, размер ядра свёртки и т. п. могут варьироваться.)

Листинг 1. Упрощенный код

```
from tensorflow.keras.layers import (Input, Conv2D,
Conv2DTranspose,
 MaxPooling2D, UpSampling2D, concatenate)
from tensorflow.keras.models import Model

def build_transcription_model(input_shape=(None, None,
1), num_pitches=88):
 inputs = Input(shape=input_shape)
 # Энкодер
 x1 = Conv2D(16, 3, activation='relu',
padding='same')(inputs)
 x1 = Conv2D(16, 3, activation='relu', padding='same')(x1)
 p1 = MaxPooling2D(2)(x1)
 ...
 # Боттлneck
 b = Conv2D(256, 3, activation='relu',
padding='same')(p4)
 b = Conv2D(256, 3, activation='relu',
padding='same')(b)
 # Декодер
 u4 = Conv2DTranspose(128, 3, strides=2,
activation='relu', padding='same')(b)
 ...
 outputs = Conv2D(num_pitches, 1,
activation='sigmoid', padding='same')(c1)
 model = Model(inputs, outputs)
 return model
```

### 3. Сепарация аудио

При наличии нескольких инструментов их спектры могут накладываться, создавая дополнительные трудности распознавания. Предварительная инструментальная сепарация позволяет разделить аудиомикс на отдельные дорожки и транскрибировать их по отдельности.

Для этого используется модель Demucs – глубокая сверточная сеть энкодер-декодер, работающая напрямую с волновой формой. Она обучена выделять из общей смеси отдельные источники (вокал, бас, ударные и т. д.). После разделения сохраняются лишь нужные гармонические дорожки (фортепиано, гитара, голос), так как ударные требуют иного подхода. В результате транскрипция упрощается: меньше ложных срабатываний, вызванных спектрами других инструментов.

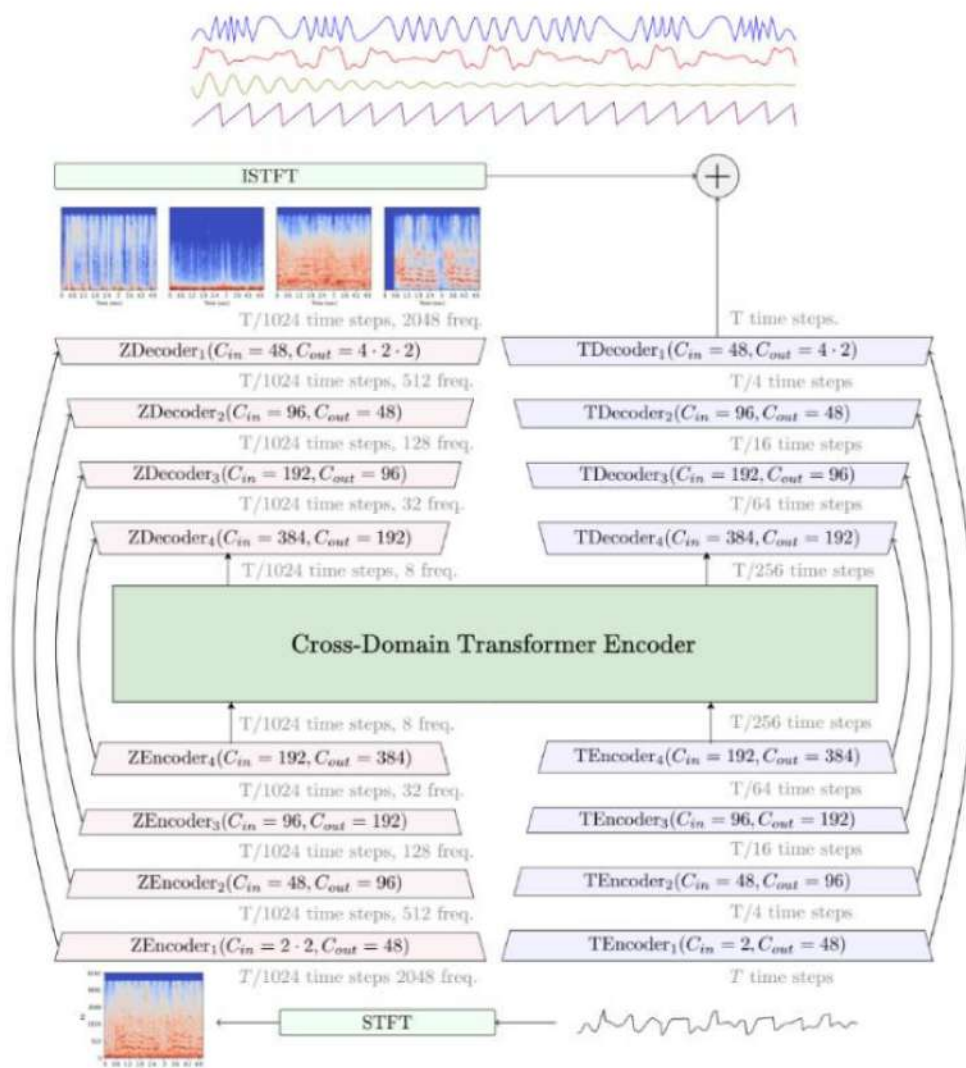


Рисунок 3 – Схематичная архитектура Demucs

#### 4. Преобразование сигналов

Сигнал каждой выделенной дорожки приводится к нужному формату (моно, 22.05 кГц), затем вычисляется SQT-спектрограмма с помощью функции `librosa.cqt()`. Настраиваются параметры количества бинов на октаву (например, 24) и диапазон частот (27.5–4186 Гц для фортепиано), после чего данные нормализуются.

Помимо CQT, для анализа тональности и аккордов может использоваться хромограмма. Однако здесь её роль вспомогательная. Librosa также предоставляет средства нормализации громкости и фильтрации шума. Детектирование атак (onset detection) способно дополнительно повышать точность во временной области, но в нашем подходе эта функция в основном реализуется сверточными слоями модели.

#### *5. Обучение и данные*

Для тренировки нужны аудиозаписи с точной разметкой нот во времени. Мы использовали несколько открытых наборов: MAPS и MAESTRO для фортепиано, а также дополнительные датасеты с изолированными дорожками.

Аудио приводилось к единому формату и разбивалось на короткие фрагменты (5–10 секунд). На каждом фрагменте рассчитывалась CQT, а целевые метки формировались как двоичная «пианола» (T×P). Для повышения обобщающей способности использовались аугментации (тайм-стретчинг, небольшая транспозиция, добавление шума). Обучение проводилось в TensorFlow с оптимизатором Adam при двоичной кроссэнтропии.

Система достигает F-меры более 0.85 для фортепиано на тестовых данных, что сопоставимо с современными результатами. Предварительная сепарация Demucs помогает при многоголосных записях, уменьшая число ложных срабатываний.

#### *6. Возможности системы*

Предложенная система успешно распознаёт полифонические ноты, включая аккорды и быстрые пассажи, выделяя несколько нот одновременно. Предварительная сепарация Demucs даёт возможность обработать многодорожечные записи ансамблей и отдельно транскрибировать партии (фортепиано, гитара, вокал). Выходной результат в формате «пианолы» легко конвертируется в стандартный MIDI, что упрощает редактирование и анализ в секвенсорах или нотных редакторах.

Система работает почти в реальном времени: на современном GPU минута музыки обрабатывается за несколько секунд. Использование CQT повышает устойчивость к различным тембрам и настройкам инструментов, что критично при работе с разнообразными записями. При значительной расстройке инструмента или сильных шумах качество снижается, но основные нотные последовательности обычно сохраняются.

#### *7. Заключение и направления будущих исследований*

В работе описана система AMT, сочетающая спектральный анализ и



глубокое обучение. Предварительная сепарация с помощью Demucs позволила эффективно распознавать ноты в многоголосном аудио. Преобразование SQT даёт представление, согласующееся с музыкальной шкалой, а сверточная сеть энкодер-декодер точно детектирует одновременные ноты и транзиентные атаки.

В будущем планируется интеграция моделей музыкального “языка” (например, рекуррентных сетей или трансформеров), которые смогут корректировать результат с учётом гармонических закономерностей. Интересно также обучение «энд-ту-энд» системы, совмещающей сепарацию и транскрипцию в одной архитектуре. Кроме того, расширение системы на ударные инструменты и распознавание выразительных нюансов (динамика, артикуляция) представляет особый интерес.

Автоматическая транскрипция продолжает стремительно развиваться. Совершенствование нейросетевых подходов, рост объёмов обучающих данных и комбинирование знаний музыкальной теории с глубоким обучением будут способствовать дальнейшим успехам в области АМТ

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Мансуров А.В., Ладыгин П.С. Способ формирования цифрового отпечатка аудиофайла на основе вектора признаков, получаемого с использованием Constant-Q и Фурье преобразований// Информатика и вычислительная техника и управление. – 2020. – №8. – С. 78–83.

УДК 004.75

**ЩУКАРЕВ А.С.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

### **АРХИТЕКТУРЫ АДАПТИВНОЙ СИСТЕМЫ ДЛЯ РАБОТЫ С ПОТОКОВОЙ ИНФОРМАЦИЕЙ**

*Рассматриваются адаптивные системы передачи и обработки данных, их классификация и основные принципы работы. Особое внимание уделяется методам сжатия, кодирования и оптимизации передачи информации для повышения эффективности каналов связи.*

На сегодняшний день использование адаптивных систем передачи данных вызывает широкий интерес из-за необходимости

эффективного использования каналов связи для оптимизации передачи данных по сети Интернет. Выделяют три группы таких систем.

Первая группа включает технологии, работа которых основана на динамическом удалении или компрессии части данных из общего потока в режиме реального времени. К преимуществам данного подхода относится способность адаптироваться к пропускной способности канала связи. Однако основной недостаток – ухудшение качества сигнала из-за потерь информации, что ограничивает сферу применения этого метода.

Ко второй группе относятся технологии, использующие адаптивную передачу данных с возможностью переключения между потоками различного разрешения и частоты кадров. В основе их работы лежат современные протоколы потоковой передачи, такие как HTTP Live Streaming (HLS) и MPEG-DASH. Эти технологии позволяют устройству-приемнику динамически изменять качество воспроизводимого контента в зависимости от текущей скорости соединения, что повышает стабильность передачи данных.

При адаптивной передаче данных возникает проблема увеличения объема информации из-за необходимости хранения контента в нескольких вариантах с разными параметрами. Принцип работы заключается в разбиении потока информации на небольшие сегменты с различными характеристиками битрейта и разрешения. Выбор сегмента может осуществляться источником передачи, где битрейт определяется скоростью загрузки, либо приемником, который анализирует качество полученного сигнала. В последнем случае возрастает нагрузка на систему, что может негативно сказаться на производительности и качестве передачи данных.

Такие системы имеют следующие недостатки:

- Оптимальная работа возможна только при увеличении числа потоков, что усложняет конфигурацию и повышает стоимость;
- Проблема выбора оптимальной длины сегментов, влияющей на баланс между качеством и скоростью передачи данных.

В третьей группе технологий используются алгоритмы динамической адаптации, встроенные в кодировщик, которые позволяют оптимизировать передачу информации под различные условия. Такие системы обеспечивают эффективное сжатие данных с минимальными потерями качества, снижая нагрузку на канал связи без увеличения числа потоков. Примером может служить применение современных видеокодеков, таких как HEVC (H.265) и AV1, которые адаптируют параметры кодирования в зависимости от условий сетей [2].



Таким образом, наиболее оптимальным решением являются технологии третьей группы, поскольку они позволяют достичь высокой эффективности передачи данных при сохранении качества сигнала и минимизации затрат на специализированное оборудование.

Для того чтобы улучшить характеристики пропускной способности каналов связи, необходимо внести изменения в техническое оснащение вычислительной техники. Использование алгоритмов сжатия позволяет добиться подобного эффекта. Совместная реализация вычислительных процессов (ВП) и алгоритмов кодирования как метода компрессии является наиболее подходящей для обработки потоковой информации.

Основным критерием эффективности является время полного цикла обработки данных, который включает в себя:

1. Обработку и сжатие преобразователя;
2. Передачу информации средствами связи;
3. Расшифровку и обработку в составе средств управления.

Для уменьшения времени передачи информации между преобразователями и блоком управления без влияния на длительность этапов сжатия и расшифровки используются специализированные фильтры. Эти фильтры изменяют сигнал в сторону вторичных преобразователей и увеличивают количество клиентских устройств, использующих один блок управления.

Рассмотрим общую схему адаптивной системы обработки информации от распределенных объектов. Информация (измерительная, управляющая, регулирующая) от объектов ( $O_1 \dots O_n$ ), обрабатываемая датчиком, поступает к измерительным узлам (ИзУз1...ИзУзн), которые преобразуют ее в поток данных, передаваемый по каналам связи в реальном времени. Измерительные узлы могут находиться на значительном расстоянии друг от друга. Вся информация передается средствами связи в адаптивную систему управления (СУ). Система преобразует и анализирует поток данных. В некоторых случаях в нее интегрируется комплекс средств для принятия решений, например, для задач технической диагностики.

Web-серверы содержат базу данных (БД) для пользовательского доступа. Вся информация представляется в удобном для понимания графическом виде (Рисунок 1).

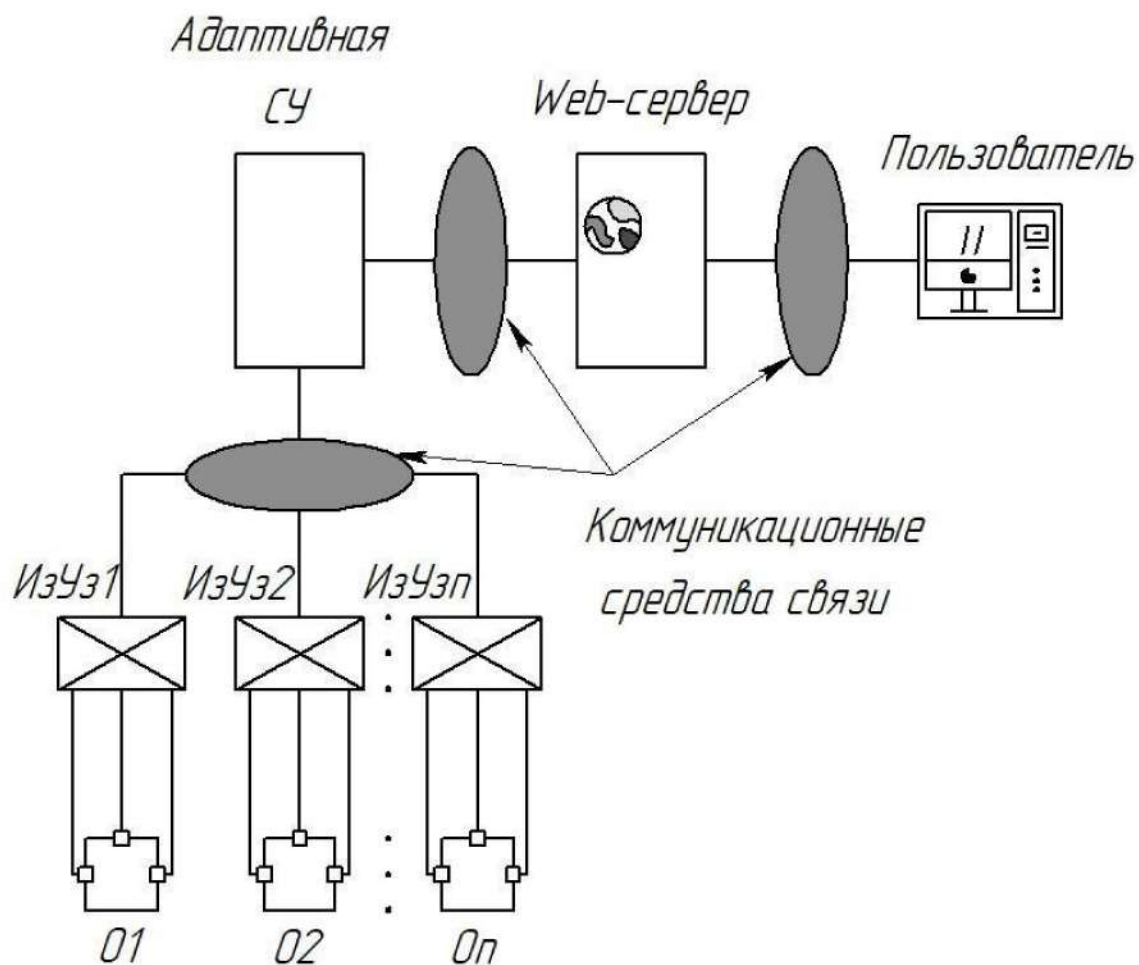


Рисунок 1 – Общая схема адаптивной системы обработки потоковых данных

Данная система является межплатформенной, масштабируемой и включает стандартные механизмы ввода/вывода. Далее подробнее рассмотрим передачу данных между измерительными узлами и адаптивной системой управления.

Рассмотрим алгоритм взаимодействия измерительных узлов вторичных преобразователей с адаптивной системой управления.

Рисунок 2 наглядно демонстрирует схему взаимодействия клиент-серверной архитектуры.

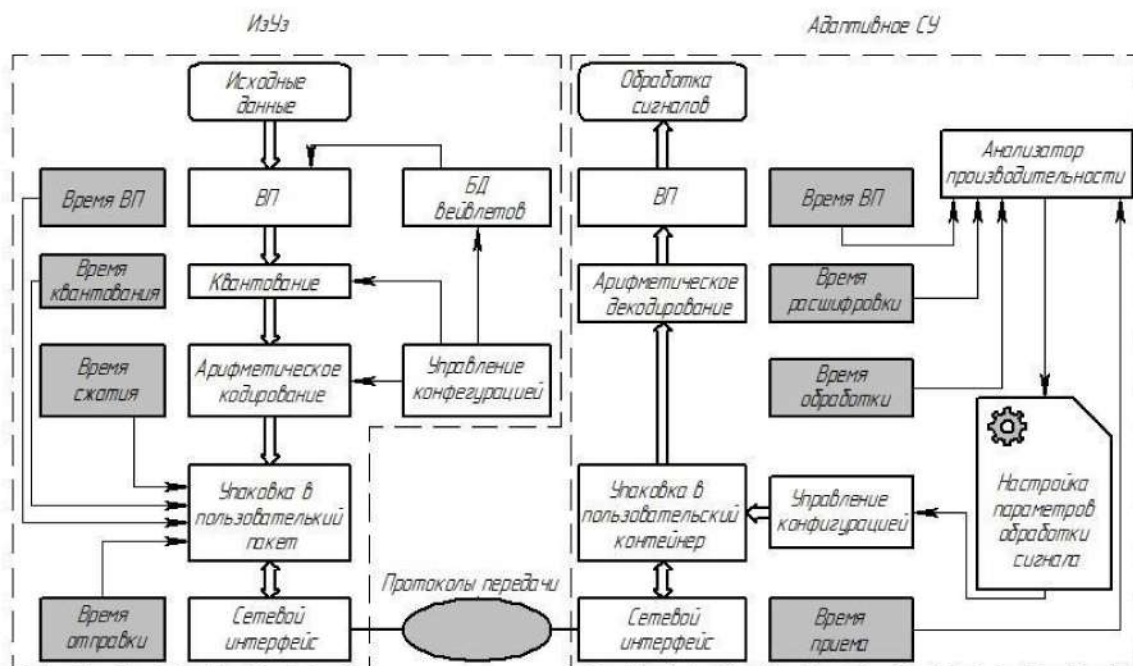


Рисунок 2 – Схема взаимодействия ИзУз и адаптивной СУ

Теперь рассмотрим подробнее каждый из этапов обработки.

Информация, поступающая из ИзУз, обрабатывается по методам ВП. В БД вейвлетов хранится основа для функционального анализа – вейвлет-функция (ВФ) или фильтры. Использование ВП направлено на: 1) сократить объем данных, для повышения энергии сигнала; 2) обозначить совокупность местных сигналов. Если работа над потоком информации направлена в адаптивную СУ, то ВП сигналов опускается.

Затем для снижения нагрузки применяется выбранный механизм компрессии, который реализуется с помощью квантования свертки. При этом технология квантования может заменяться при функционировании системы, а процесс сжатия может и вовсе отсутствовать.

Время на квантование, сжатие, ВП и отправку вместе с потоковыми данными формируется в пользовательский пакет, который транслируется по средствам протоколов передачи в адаптивную СУ.

Далее происходит распаковка, обратное преобразование и расшифровка данных.

На основе временных показателей, полученных в результате обработки и выбранного протокола передачи с помощью анализатора производительности создается файл с настройкой параметров: коэффициентов квантования, номеров базового вейвлета, степени сжатия.

Полученная конфигурация формируется в контейнер, который уже передается на ИзУз.

По полученным настройкам происходит дальнейшая обработка сигнала [1].

### **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Шукарев, А.С. Проектирование программно-аппаратных средств сжатия информации: ВКР магистра: специальность 15.04.04 «Автоматизация технологических процессов и производств» / А. С. Шукарев; науч. рук. Ю. А. Сосулин; Рязанский государственный радиотехнический университет имени В.Ф. Уткина. – Рязань, 2020. – 72 с.

2. Лачугин Д.В. Алгоритмы обработки потоковой информации в распределенных вторичных преобразователях на основе адаптивного сжатия: диссертация. – С.: СГТУ им. Гагарина Ю.А., 2015. – 191 с.

УДК 004.413

**ЮРКОВА В.И., МАРКИН А.В.**

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

### **РАЗРАБОТКА СИСТЕМЫ АНАЛИЗА ПЛАТЕЖНОЙ ДИСЦИПЛИНЫ АБОНЕНТОВ ПРЕДПРИЯТИЙ ЖКХ**

*Рассматривается проблема анализа платежной дисциплины абонентов предприятий жилищно-коммунального хозяйства (ЖКХ).*

В настоящее время жилищно-коммунальные услуги являются неотъемлемой частью быта большинства людей. Тем не менее, стоит отметить, что многие граждане несвоевременно оплачивают данные услуги, и это обусловлено не только экономическими причинами, но и недостаточной осведомлённостью в данном вопросе.

Анализ ситуации показывает, что значительная часть населения демонстрирует низкий уровень финансовой грамотности в сфере ЖКХ, о чём свидетельствует большое количество случаев неуплаты, негативно влияющих как на потребителей, так и на поставщиков жилищно-коммунальных услуг [1].

В общем случае, под платежной дисциплиной понимается обязанность физического или юридического лица, имеющего кредитные обязательства, в установленные сроки осуществлять платежи.

Платежная дисциплина абонента (потребителя ресурсов ЖКХ) заключается в своевременной оплате за коммунальные услуги, такие как электричество, вода, газ, отопление и другие услуги, предоставляемые жилищно-коммунальными службами [2].

Высокие расходы на коммунальные услуги зачастую являются причиной несоблюдения платежной дисциплины. Существует ряд методов, которые позволяют снизить такие расходы. Например, установка энергосберегающих приборов (светодиодных или энергоэффективных ламп, розеток и т. д.), счётчиков воды, газа и отопления может привести к сокращению расходов в долгосрочной перспективе.

Однако предприятия-поставщики услуг ЖКХ нуждаются в разработке систем стимулирующих потребителей своевременно оплачивать коммунальные услуги. Для повышения эффективности работы сервиса необходимо внедрить средство анализа, которое позволит отслеживать платёжную дисциплину каждого из абонентов, выявлять тенденции и проблемы, а также прогнозировать возможные неплатежи.

Таким образом, разработка системы анализа платёжной дисциплины абонентов предприятий ЖКХ представляет собой актуальную задачу, которая позволит организациям ЖКХ решить ряд проблем, связанных с управлением финансами и улучшением качества предоставляемых услуг.

#### *Статистические методы анализа данных*

Существует множество методов анализа данных, каждый из которых предназначен для решения определенных задач. Математическая формализация играет ключевую роль в эффективном применении различных методов анализа данных. Она позволяет перевести изучаемые объекты и явления в математическую форму, представив их в виде формализованных моделей или структур.

Процесс математической формализации предполагает выделение ключевых характеристик изучаемого объекта и их описание с помощью математических конструкций, таких как множества, функции, матрицы и другие математические объекты. Это дает возможность применять к полученным моделям строгие логические рассуждения, количественные методы и вычислительные алгоритмы.

Для математической формализации абонента предприятия ЖКХ можно предложить следующий подход:

Пусть:

$A = \{a_1, a_2, \dots, a_n\}$  - множество всех абонентов предприятия ЖКХ, где



$a_i$  -  $i$ -ый абонент.

Каждого абонента  $a_i$  можно описать вектором характеристик:

$$a_i = (x_1, x_2, \dots, x_m) \quad (1)$$

где:

$x_j$  -  $j$ -ая характеристика абонента, например:

$x_1$  - тип жилья (квартира, частный дом и т.д.),

$x_2$  - площадь жилья,

$x_3$  - количество проживающих,

$x_4$  - среднеемесячное потребление ресурсов (электроэнергия, вода),

$x_5$  - история платежей (своевременность, задолженность),

$x_6$  - социальный статус (пенсионер, многодетная семья и т.д.).

Таким образом, множество всех абонентов  $A$  можно представить как:

$$A = \{(x_1^1, x_2^1, \dots, x_m^1), (x_1^2, x_2^2, \dots, x_m^2), \dots, (x_1^n, x_2^n, \dots, x_m^n)\} \quad (2)$$

Подобное математическое описание абонентов позволяет использовать широкий спектр методов анализа данных. Каждый из этих методов имеет свои особенности, применения и математические основы. Основными методами интеллектуального анализа данных являются:

1. Классификация: Метод классификации используется для разделения объектов на заранее определенные классы или категории на основе их характеристик. Для классификации могут использоваться различные алгоритмы, такие как метод ближайших соседей, логистическая регрессия, случайный лес и т. д. [3].

2. Кластеризация: Кластерный анализ или кластеризация позволяет группировать объекты на основе их сходства без предварительно заданных категорий. Целью является выявление внутригрупповых сходств и межгрупповых различий. Алгоритмы кластеризации включают  $k$ -средних, иерархическую кластеризацию, DBSCAN и другие [4].

3. Регрессионный анализ: Регрессионный анализ используется для изучения отношений между зависимыми и независимыми переменными. Он помогает предсказывать значения зависимой переменной на основе независимых переменных. Примеры методов регрессии включают линейную регрессию, логистическую регрессию, полиномиальную регрессию и т. д. [5].

4. Поиск ассоциативных правил: Этот метод используется для обнаружения взаимосвязей и зависимостей между различными элементами в наборе данных. Он позволяет выявить правила типа "если-то", которые помогают понять взаимосвязи между различными элементами [6].

5. Обнаружение аномалий: Метод обнаружения аномалий направлен на выявление необычных или аномальных паттернов в данных, которые могут указывать на ошибки, мошенничество или другие нештатные ситуации. Различные подходы к обнаружению аномалий включают LOF (Local Outlier Factor), Isolation Forest, One-Class SVM и другие [7].

Таблица 1 содержит названия методов интеллектуального анализа данных, которые будут использоваться для анализа платежной дисциплины абонентов, а также приведены примеры основных задач, которые будет решать разрабатываемая система.

Таблица 1 – Примеры задач и методов анализа данных в сфере анализа платежной дисциплины

Метод анализа данных	Задача и пример
Классификация	Определение платежной дисциплины абонента (своевременный/несвоевременный плательщик) на основе данных об истории платежей, социально-демографических характеристик. Пример: построение модели логистической регрессии для классификации абонентов.
Кластеризация	Выявление групп абонентов с похожими профилями платежной дисциплины. Пример: применение алгоритма К-средних для кластеризации абонентов по показателям своевременности оплаты, задолженности, частоты просрочек; Группировка зданий по схожим энергетическим характеристикам для разработки индивидуальных рекомендаций по повышению энергоэффективности. Пример: использование методов кластеризации для выделения кластеров зданий с похожим потреблением энергии.
Регрессионный анализ	Прогнозирование размера задолженности абонентов. Пример: построение модели линейной регрессии для предсказания суммы задолженности на основе истории платежей,

	характеристик абонента и местоположения; Прогнозирование потребления ресурсов (электричества, воды, тепла) на основе исторических данных, погодных условий и социально-экономических факторов. Пример: применение моделей временных рядов (ARIMA, экспоненциальное сглаживание) для прогнозирования будущего потребления.
Поиск ассоциативных правил	Выявление связей между характеристиками абонентов и их платежной дисциплиной. Пример: применение алгоритма Apriori для поиска правил вида "Если абонент проживает в районе X и имеет задолженность более Y рублей, то вероятность просрочки платежа составляет Z%"; Обнаружение связей между параметрами работы оборудования и возникновением нештатных ситуаций. Пример: использование алгоритма Apriori для выявления правил вида "Если давление в трубопроводе ниже X и температура выше Y, то вероятность аварии составляет Z%".
Обнаружение аномалий	Выявление абонентов с аномальным поведением в оплате счетов, которые могут указывать на мошенничество или ошибки. Пример: использование алгоритма LOF (Местные выбросы) для определения абонентов, платежные профили которых значительно отличаются от большинства; Обнаружение нештатных ситуаций в работе оборудования. Пример: применение методов одноклассовых SVM для выявления отклонений в показаниях приборов учета, которые могут свидетельствовать о возможных авариях.

Реализация методов подготовки полученных данных осуществляется с помощью возможностей языка программирования Python.

#### *Описание процесса анализа данных*

На рисунке 1 показан сценарий события «Формирование отчета по абоненту», которое инициируется пользователем, желающим узнать информацию о платежной дисциплине выбранного абонента.



В схеме сценария событий используется следующий ряд условных обозначений:

- 1) данные, поступающие в систему или исходящие из нее, представлены в эллипсах;
- 2) управляющая информация показана стрелками в верхней части прямоугольников;
- 3) внутрисистемные данные показаны справа от прямоугольников;
- 4) исключительные ситуации показаны в нижней части прямоугольников. Там, где возможно несколько исключительных ситуаций, они заключаются в общий прямоугольник;
- 5) имя следующего события, ожидаемого после завершения сценария, приводится в затененном прямоугольнике.



Рисунок 1 – Сценарий события «Формирование отчета по абоненту»

### Проектирование функциональных элементов

Концептуальная модель архитектуры изображена на рисунке 2.



Рисунок 2 – Концептуальная модель архитектуры

Компонент извлечения отвечает за получение данных из баз данных расчетной и платежной систем. Компонент преобразования отвечает за обработку и трансформацию данных для их структурирования и очистки перед загрузкой. На этом этапе нужно убедиться, что данные соответствуют формату, установить соответствия справочников, провести агрегацию, фильтрацию и денормализацию, если это необходимо. Компонент загрузки отвечает за отправку обработанных данных в целевую систему (базу данных или хранилище). На этом этапе компонент обеспечивает выполнение

процесса в соответствии с запланированным графиком, обработав все контрольные мероприятия, такие как проверку на дубликаты и ошибки формата. На последнем этапе данные, находящиеся в хранилище используются инструментами динамического построения отчетов и витрин данных, составления запросов к хранилищу данных для глубокого анализа.

Исходя из имеющихся модулей и объектов концептуальной модели, разработана диаграмма компонентов (рисунок 3). Данная диаграмма отображает состав системы и демонстрирует зависимости компонентов.

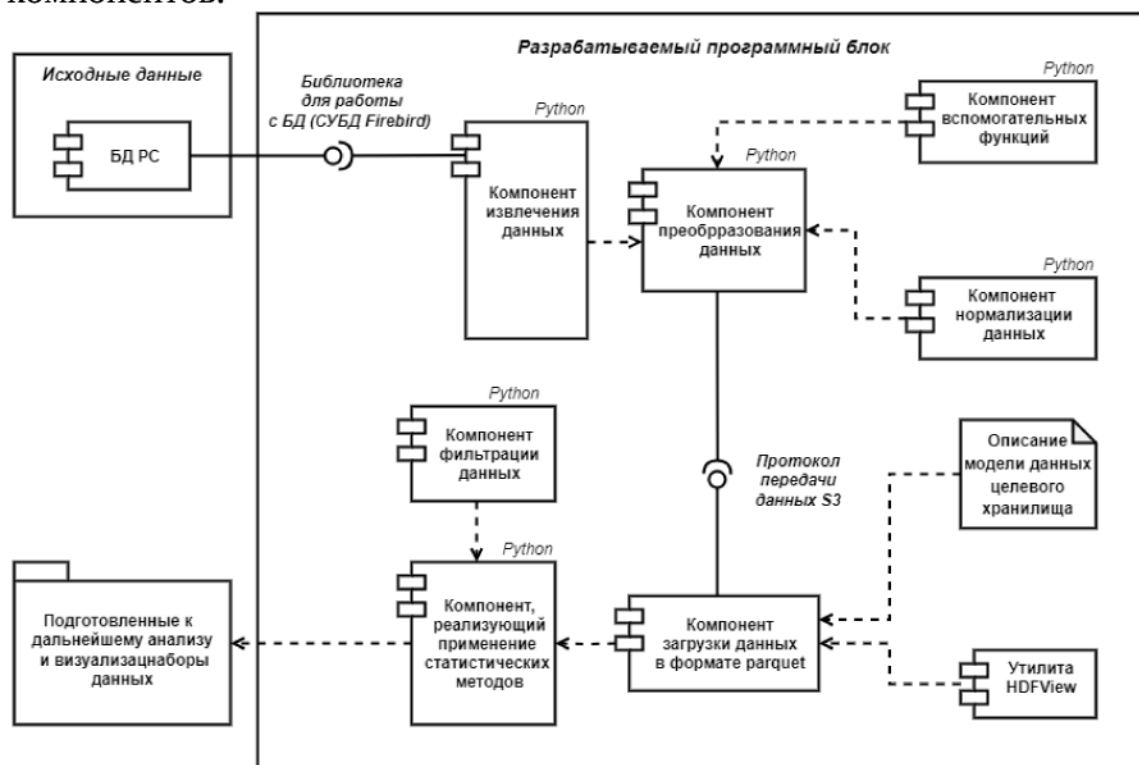


Рисунок 3 – Диаграмма компонентов подсистемы

Для работы с исходной базой данных используются программные библиотеки на языке Python. Они предназначены для работы с реляционными СУБД с применением технологии ORM и служат для синхронизации объектов Python и записей реляционной базы данных, а также позволяют описывать структуры баз данных и способы взаимодействия с ними на языке Python без использования SQL.

Компонент извлечения данных посредством обращения к исходным БД осуществляет инкрементальную выгрузку данных. Это означает, что в хранилище загружаются только те записи, которые изменились (вставки, обновления и удаления) с момента последней

загрузки; а не полную загрузку всех данных (всех записей, включая те, которые не изменились с момента последней загрузки) в хранилище.

Исходные данные могут содержать ошибки или быть представлены в форматах, не применимых в дальнейшем процессе создания аналитических отчетов, поэтому их нужно корректировать. Поэтому, для корректного отображения данных в программном блоке разработан компонент преобразования.

Полученная в ходе выгрузки и преобразования информация сохраняется в формат данных parquet. Для загрузки данных в общее хранилище используется протокол S3.

#### *Заключение*

Использование современных информационных технологий в анализе платежной дисциплины абонентов предприятий жилищно-коммунального хозяйства (ЖКХ) открывает новые возможности для автоматизации процессов управления финансами. Применение методов машинного обучения и статистического анализа позволяет значительно ускорить обработку данных, выявление закономерностей и прогнозирование платежного поведения, что особенно актуально в условиях роста объемов информации и необходимости повышения финансовой устойчивости.

В данной работе рассмотрен подход, основанный на интеграции различных методов анализа данных в рамках единой системы, предназначенной для автоматизации анализа платежной дисциплины. Этот метод позволяет эффективно обрабатывать данные о платежах, выявлять проблемные группы абонентов и прогнозировать возможные неплатежи, что снижает временные и финансовые затраты организаций.

Применение доступных инструментов анализа данных делает данный подход приемлемым как для крупных предприятий, так и для небольших организаций ЖКХ. Прикладное значение результатов работы заключается в том, что предложенная система может быть использована для создания эффективных инструментов поддержки управления финансами. Это позволит уменьшить нагрузку на сотрудников, минимизировать риски неплатежей и повысить качество обслуживания клиентов. Внедрение подобного решения способствует более эффективному управлению финансовыми потоками и стимулирует развитие инновационных подходов в сфере ЖКХ.

### **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Подольский, Г. В. Гражданско-правовые основы повышения платежной дисциплины «неотключаемых» потребителей

энергетических ресурсов / Г. В. Подольский // Образование и право. – 2023. – № 1. – С. 217-223.

2. Лапшин, Н.П. Проблемы повышения платежной дисциплины населения / Н.П. Лапшин // Актуальные вопросы экономики, управления и права: сборник научных трудов (ежегодник). – 2014. – № 6. – С. 42-51.

3. Белкин, Д.А. Анализ работы энергосбытовой компании по повышению платежной дисциплины физических лиц / Д.А. Белкин // Аллея науки. – 2017. – Т. 3, № 13. – С. 943-948.

4. Медведева, Н. Г. О повышении платежной дисциплины в сфере жилищно-коммунального хозяйства / Н. Г. Медведева // Водосбережение, мелиорация и гидротехнические сооружения как основа формирования агрокультурных кластеров России в XXI веке: Сборник докладов XVIII международной научно-практической конференции: в 3-х томах, Тюмень, 18 марта 2016 года. Том 2. – Тюмень: ФГБОУ ВПО "Тюменский государственный архитектурно-строительный университет", 2016. – С. 69-76.

5. Анализ данных : учебник для вузов / В. С. Мхитарян [и др.] ; под редакцией В. С. Мхитаряна. – Москва: Издательство Юрайт, 2024. – 490 с.

6. Миркин, Б.Г. Базовые методы анализа данных: учебник и практикум для вузов / Б.Г. Миркин. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2024. – 261 с.

7. Гинько А. Ю. Анализ и визуализация данных в YandexDataLens. Подробное руководство: от новичка до эксперта. – М.: ДМК Пресс, 2023. – 356 с.

# **ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ПРИКЛАДНЫХ ИССЛЕДОВАНИЯХ**

**Межвузовский сборник научных трудов**

Компьютерная верстка: А.Н. Сапрыкин, М.С. Кошелева  
Изображение, используемое на обложке сборника,  
взято с сайта freepik.com

Подписано в печать 28.04.25. Формат бумаги 60х84 1/16.  
Бумага офсетная. Печать струйная. Усл. печ. л. 22.125.  
Тираж 100 экз. Заказ № 8166.

Издательство ИП Коняхин А.В. (Book Jet)

Отпечатано в типографии Book Jet  
390005, г. Рязань, ул. Пушкина, д. 18  
Сайт: <http://bookjet.ru>  
Почта: [info@bookjet.ru](mailto:info@bookjet.ru)  
тел.: +7 (4912) 466-151

**ISBN 978-5-908007-16-0**