Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина»
Институт радиоэлектроники и информационных технологий-РТФ
Кафедра информационных технологий и систем управления

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК
Заведующий кафедрой ИТиСУ
_____ Е. В. Кислицын
«30» мая 2025 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ ПРОЦЕССОВ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ В БАНКЕ

Научный руководитель: Петренко Тимур Сергеевич, к.м.н, доцент

Нормоконтролер: Огуренко Егор Владимирович

Студент группы РИМ-230962 Левченко Антон Сергеевич

Екатеринбург
2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ
Кафедра информационных технологий и систем управления
Направление подготовки 09.04.01 Информатика и вычислительная техника
Образовательная программа 09.04.01/33.03 Инженерия машинного обучения

ЗАДАНИЕ

на выполнение магистерской диссертации

студента Левченко Антона Сергеевича **группы** РИМ-230962
(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы
Разработка системы автоматизации процессов технической поддержки в банке

Утверждена распоряжением по институту от «2» декабря 2024 г. № 33.02-05/334

2. Научный руководитель
Петренко Тимур Сергеевич, к.м.н., доцент

3. Исходные данные к работе
Нормативная, учебная, методическая литература по теме магистерской диссертации, материалы, полученные в ходе преддипломной практики, техническое задание по разработке продукта

4. Перечень демонстрационных материалов
Презентация, архитектура программного комплекса, приложение

5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	1 раздел (глава)	до 24.03.2025 г.	Выполнено
2.	2 раздел (глава)	до 28.04.2025 г.	Выполнено
3.	3 раздел (глава)	до 19.05.2025 г.	Выполнено
4.	ВКР в целом	до 23.05.2025 г.	Выполнено

Руководитель _____ Петренко Тимур Сергеевич
(подпись) (Ф.И.О.)

Студент задание принял к исполнению 10.02.2025 г. _____
дата (подпись)

6. Допустить Левченко Антона Сергеевича к защите магистерской диссертации в Государственной экзаменационной комиссии.

Заведующий кафедрой ИТиСУ _____ Е. В. Кислицын
(подпись) (Ф.И.О.)

РЕФЕРАТ

Выпускная квалификационная работа магистра состоит из введения, трех глав, заключения, списка использованных источников и приложений. Работа включает в себя 85 страницы, 7 рисунков, 1 таблицу, 53 источника, 4 приложения.

ИНТЕЛЛЕКТУАЛЬНАЯ ПОДДЕРЖКА, ВЕКТОРНЫЙ ПОИСК, LLM, RAG, QDRANT, MISTRAL, ТЕХПОДДЕРЖКА, БАЗА ЗНАНИЙ, FASTAPI, СИСТЕМА ВОПРОС-ОТВЕТ, ЛОКАЛЬНАЯ МОДЕЛЬ, АВТОМАТИЗАЦИЯ, NLP, ЭКСПЕРТНАЯ ОЦЕНКА

Выпускная квалификационная работа посвящена разработке прототипа системы интеллектуальной поддержки сотрудников на основе архитектуры Retrieval-Augmented Generation (RAG) с использованием локальной языковой модели и специализированной базы знаний.

Объектом исследования являются процессы информационно-консультационной поддержки в организациях с высоким документооборотом.

Целью работы является разработка системы, способной ускорить обработку обращений путём предоставления релевантной информации сотруднику поддержки. В качестве методов использованы семантический анализ текста, векторное индексирование, генеративные языковые модели, проектирование API и оценка качества с привлечением экспертов.

В ходе выполнения работы реализован сервис, включающий модуль векторного поиска (Qdrant), генеративную модель (Mistral 7B-Instruct), серверную часть (FastAPI) и пользовательский интерфейс. Проведена экспертная оценка уместности и релевантности ответов. Подтверждена гипотеза о сокращении времени на обработку обращений при сохранении высокого качества ответов.Разработанное решение применимо в службах внутренней ИТ-поддержки, особенно в организациях с повышенными требованиями к приватности, таких как финансовые учреждения.

СОДЕРЖАНИЕ

РЕФЕРАТ 4
ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 6
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 7
ВВЕДЕНИЕ 8
1 Анализ современного состояния проблемы 10
1.1 Обзор текущего положения на рынке 10
1.2 Обзор существующих решений автоматизации поддержки 16
1.3 Обзор языковых моделей для автоматизации поддержки 21
1.4 Retrieval-Augmented Generation (RAG): принцип работы, применение и польза 29
1.5 Обзор современных векторных баз данных 34
2 Архитектура выбранного решения и анализ материалов 44
2.1 Обоснование выбора технологий 44
2.2 Описание компонентов системы 45
2.2.1 База метаданных 45
2.2.2 Векторное хранилище 46
2.2.3 Серверная логика и API 46
2.2.4 Локальная языковая модель 47
2.2.5 Обработка и подготовка данных 48
2.3 Интеграция и взаимодействие компонентов 49
2.4 Сбор и предварительная обработка текстового корпуса 52
2.4.1 Источник данных и его особенности 52
2.4.2 Технические меры устойчивости 53
2.4.3 Результаты сбора 54
2.4.4 Практическое значение для системы RAG 54
2.5 Сбор и предварительная обработка текстового корпуса 55
2.5.1 Методология оценки датасета 55
2.5.2 Общая характеристика корпуса 56
3 Методика и оценка результатов 61
3.1 Алгоритм оценки качества 61
3.2 Результаты эксперимента 62
3.3 Оценка качества ответов с участием человека 63
ЗАКЛЮЧЕНИЕ 67
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 70
ПРИЛОЖЕНИЕ А 77
ПРИЛОЖЕНИЕ Б 78
ПРИЛОЖЕНИЕ В 79
ПРИЛОЖЕНИЕ Г 80

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Интеллектуальная поддержка — программные средства и алгоритмы, обеспечивающие автоматизированную помощь пользователю за счёт анализа запросов и предоставления релевантной информации.

Retrieval-Augmented Generation (RAG) — архитектурный подход, объединяющий векторный поиск по базе знаний и генеративную языковую модель для создания контекстуализированных ответов на пользовательские запросы.

Генеративная языковая модель (LLM, Large Language Model) — модель обработки естественного языка, обученная на больших корпусах текста и способная формировать осмысленные ответы в свободной форме.

Векторный поиск — метод поиска по семантическому сходству между векторами, представляющими тексты, основанный на математическом сравнении эмбеддингов.

Эмбеддинг — числовое представление текста в многомерном пространстве, отражающее его смысловое содержание.

Qdrant — система векторной базы данных, предназначенная для быстрого поиска релевантных документов на основе эмбеддингов.

Mistral 7B-Instruct — локально запускаемая генеративная языковая модель, оптимизированная для следования инструкциям, используемая в составе разработанной системы.

База знаний — структурированное хранилище документов, инструкций и справочных данных, используемое системой для формирования ответов на запросы.

НЛП (NLP, Natural Language Processing) — область искусственного интеллекта, связанная с анализом и обработкой естественного языка.

Reranking — процедура повторной сортировки результатов поиска по дополнительным критериям, повышающая точность извлечения информации.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

API — Application Programming Interface, программный интерфейс приложения для взаимодействия между компонентами.

BERT — Bidirectional Encoder Representations from Transformers, двунаправленная языковая модель от Google.

FAQ — Frequently Asked Questions, часто задаваемые вопросы.

GPT — Generative Pre-trained Transformer, генеративный предобученный трансформер от OpenAI.

IT — Information Technology, информационные технологии.

LLM — Large Language Model, большая языковая модель, способная к генерации и анализу текста.

ML — Machine Learning, машинное обучение.

NLP — Natural Language Processing, обработка естественного языка.

RAG — Retrieval-Augmented Generation, архитектура генерации текста с привлечением внешнего поиска.

Qdrant — векторная база данных для реализации семантического поиска по эмбеддингам.

FastAPI — фреймворк на Python для быстрой разработки REST API.

Mistral 7B-Instruct — локальная генеративная языковая модель, адаптированная под выполнение инструкций.

JSON — JavaScript Object Notation, формат хранения и обмена структурированными данными.

TTR — Type-Token Ratio, коэффициент лексического разнообразия текста.

REST — Representational State Transfer, архитектурный стиль взаимодействия с веб-сервисами.

SME — Small and Medium Enterprises, малые и средние предприятия.

ВВЕДЕНИЕ

В современных условиях цифровизации и роста объёмов информации, банковские учреждения всё чаще сталкиваются с необходимостью оптимизации процессов внутренней технической поддержки. Объём клиентских и внутренних запросов увеличивается, а требования к скорости и точности обработки обращений возрастают.

При этом традиционные средства поддержки, такие как IVR-системы, сценарные чат-боты и статические базы знаний, оказываются недостаточными для обработки сложных запросов в естественном языке. Эти обстоятельства определяют актуальность поиска новых, более интеллектуальных решений в области автоматизации поддержки.

Одним из перспективных подходов, зарекомендовавших себя в ИТ-индустрии, является технология Retrieval-Augmented Generation (RAG), сочетающая в себе мощность генеративных языковых моделей и возможности поиска по базе знаний. Особую актуальность представляет исследование возможностей RAG в сочетании с локальными моделями, обеспечивающими приватность и независимость от внешних API, что особенно важно для финансовой отрасли.

Новизна настоящего исследования заключается в адаптации RAG с локальной LLM под задачи внутренней поддержки в банковской среде.

Цель работы — разработка, реализация и экспериментальная оценка прототипа системы интеллектуальной поддержки на основе подхода Retrieval-Augmented Generation, адаптированной под задачи внутреннего ИТ-консалтинга или клиентской поддержки в финансовой организации.

Для достижения цели в рамках исследования были поставлены следующие задачи:

1. провести обзор существующих решений в области автоматизации поддержки и RAG;
2. выбрать архитектуру и программные компоненты системы

(векторное хранилище, языковая модель, API);

3. собрать и предварительно обработать текстовый корпус для базы знаний;

4. реализовать функциональный прототип сервиса;

5. провести оценку качества ответов, включая метрики точности и аннотирование человеком.

Гипотеза исследования заключается в том, что использование архитектуры Retrieval-Augmented Generation (RAG) на базе локальной языковой модели и специализированной базы знаний позволит создать инструмент поддержки, способный существенно сократить время обработки обращений за счёт быстрого предоставления релевантной информации сотруднику службы поддержки.

Объект исследования — процессы информационно-консультационной поддержки пользователей в организациях, работающих с большим объёмом нормативной и справочной документации.

Предмет исследования — архитектура, методы и программные средства, обеспечивающие реализацию системы RAG с применением векторного поиска и генерации ответов локальной языковой моделью.

1 Анализ современного состояния проблемы

1.1 Обзор текущего положения на рынке

Современные банки обслуживают огромный поток клиентских запросов и технических вопросов, поступающих в службы поддержки. Высокий уровень качества и оперативности поддержки напрямую влияет на удовлетворенность клиентов банковскими услугами. Однако содержание разветвлённой сети контакт-центров с достаточным числом квалифицированных операторов – задача дорогостоящая и трудоемкая. В последние годы финансовые организации активно внедряют решения на базе искусственного интеллекта (AI) для автоматизации процессов поддержки, стремясь снизить нагрузку на персонал и улучшить доступность сервиса [32, 51].

Например, по данным исследования Naumen, к 2024 году чат-боты внедрены уже в 29% российских банков, а голосовые помощники – в 24%, что на 19% больше, чем годом ранее [51].

Крупные банки стремятся решить максимум типовых запросов с помощью бот-помощников, позволяя людям фокусироваться на более сложных случаях [51].

Автоматизация клиентской поддержки с помощью технологий искусственного интеллекта является предметом интенсивных исследований в последние годы. Многие работы посвящены обзору возможностей и вызовов применения AI в банковском секторе.

Так, Omar H. Fares выполнил систематический обзор литературы и текущих трендов по внедрению искусственного интеллекта в банковской сфере [18].

Обзор показал, что банки стремятся использовать AI для повышения операционной эффективности, улучшения качества обслуживания и персонализации услуг.

Одновременно отмечаются и вызовы – необходимость соблюдения

регуляторных требований, управление рисками, интеграция AI в устоявшиеся бизнес-процессы. Схожие выводы сделаны в исследовании Ghandour, где обобщаются возможности и проблемы AI в банковском деле . В частности, подчеркивается важность баланса между автоматизированными и человеческими каналами обслуживания, поскольку клиенты ценят удобство технологий, но всё ещё ожидают возможности живого общения при сложных вопросах [21].

Отдельное внимание исследователи уделяют влиянию чат-ботов и виртуальных ассистентов на опыт и удовлетворённость клиентов. Al-Araj изучал внедрение AI-систем в банковском секторе Иордании и обнаружили, что такие приложения (например, для проверки кредитного рейтинга, детекции мошенничества и т.п.) снижают нагрузку на персонал и улучшают восприятие сервиса клиентами. Отмечается, что виртуальные ассистенты ускоряют обслуживание рутинных запросов, однако для полноценного удовлетворения потребностей клиентов банки должны комбинировать виртуальных агентов с человеческими, особенно при решении нестандартных проблем [40]. Bhattacharya и Sinha в своей работе акцентируют, что применение AI-технологий способно значительно обогатить клиентский опыт в банках за счёт персонализированных рекомендаций и круглосуточной поддержки [11].

В то же время авторы подчёркивают важность поддержания прозрачности и доверия: клиенты охотнее пользуются AI-сервисами, если понимают их ограничения и получают возможность эскалации к оператору при необходимости.

В литературе также исследуется экономический эффект от внедрения AI в поддержку. Например, De Andrade и Tumelero на примере крупного банка показали, что чат-бот на базе AI может привести к экспоненциальному росту эффективности обслуживания клиентов [6].

Авторы описывают кейс, в котором интеграция IBM Watson в процессы поддержки позволила существенно повысить такие показатели, как скорость

отклика, доступность сервиса, предсказуемость и надёжность обработки запросов. В результате банк смог сократить очереди на линии и перераспределить часть персонала на решение более сложных задач, что подтверждает высокую окупаемость инвестиций в подобные технологии. Схожие результаты получены и в других работах. Например, Maseke отмечает, что внедрение AI-инструментов в обслуживание клиентов положительно коррелирует с уровнем удовлетворённости: клиенты ценят скорость и удобство цифровых каналов, что улучшает их общий опыт взаимодействия с банком [33]. В ряде исследований рассматривается влияние конкретных технологий AI на поддержку. Hentzen провел мета-анализ использования искусственного интеллекта на фронт-линии финансовых услуг (в отделениях, контакт-центрах, онлайн-каналах) [8].

Их обзор подтвердил тренд смещения простых операций в автоматизированные каналы: чат-боты берут на себя роль первых линий поддержки, отвечая на часто задаваемые вопросы, помогая навигировать по услугам, выполняя простые транзакции. Отдельно отмечается возросший интерес к голосовым помощникам и интеграции AI в колл-центры называют AI-ассистентов «следующей границей» развития банковских сервисов, позволяющей одновременно снизить издержки и предоставить клиентам более персонализированный сервис [5].

Однако они подчёркивают, что для успешного внедрения требуется адаптация бизнес-моделей и обучение персонала новым навыкам взаимодействия с AI-системами.

Следует отметить вклад отечественных исследований и аналитики. Консалтинговое агентство Markswebb ежегодно оценивает качество клиентского опыта взаимодействия с банковскими чат-ботами. Согласно их рейтингу Chatbot Rank 2024, лучшие результаты демонстрируют решения крупных банков, таких как Альфа-Банк и ВТБ [52].

Российские авторы также рассматривают вопросы эффективности чат-ботов. В работе Гупта изучалось внедрение чат-бота в службу поддержки

крупного банка и анализировалось восприятие клиентами [32].

Выявлено, что ключевыми факторами успеха являются способность бота понимать разнообразные формулировки запросов на русском языке и умение корректно переключаться на оператора при необходимости.

Обзор научных и отраслевых источников позволяет сделать вывод, что сочетание больших языковых моделей и систем поиска знаний рассматривается сегодня как перспективное направление развития клиентских сервисов в банках. С одной стороны, накоплен значительный опыт использования правил и сценариев (FAQ-базы, простые чат-боты) для автоматизации поддержки – эти подходы доказали свою эффективность в обработке типовых запросов, но ограничены в гибкости.

С другой стороны, появление мощных универсальных моделей NLP открывает возможность значительно расширить границы автоматизации, поручив машине ведение сложного диалога и поиск ответов в большой базе знаний. В следующих разделах мы подробно рассмотрим технологические компоненты подобных решений – от примеров уже существующих реализаций до обзора самих языковых моделей, подхода RAG и инструментов хранения семантических данных.

Вместе с тем, ранние поколения цифровых помощников (чат-ботов) зачастую работали по заранее заданным сценариям и имели ограниченное понимание естественного языка. Пользователи нередко сталкивались с тем, что бот не может корректно распознать суть проблемы или даёт шаблонные ответы, не учитывающие контекст. Это приводило к разочарованию и снижению доверия к подобным системам [32].

Так, El-Gohary отмечал, что вовлеченность клиентов в использование чат-ботов пока не оправдывает ожиданий банков, несмотря на технологическую готовность [46].

Следовательно, существует потребность в более «умных» системах, способных понимать сложные вопросы и предоставлять точные ответы на основе актуальных знаний.

Прорывом в области обработки естественного языка стало появление больших языковых моделей (Large Language Models, LLM) на основе нейросетевой архитектуры трансформеров. Модели вроде GPT-3 и последующих превзошли предыдущие алгоритмы по способности понимать запросы на естественном языке и генерировать осмысленные связные ответы [1]. Например, GPT-3 от OpenAI (175 млрд параметров) продемонстрировала возможность решать новые задачи по лишь нескольким примерам (режим few-shot learning), достигая результатов уровня лучших специализированных моделей без отдельного обучения под каждую задачу [1].

Ещё более мощная модель GPT-4 стала мультимодальной (принимающей текст и изображения) и показала человеко-подобный уровень выполнения множества профессиональных тестов – например, вошла в топ-10% по результатам симулированного экзамена на адвоката [23]. Развитие таких моделей открыло новые возможности для автоматизации поддержки: теперь виртуальный помощник способен понимать сложные вопросы на естественном языке и генерировать развёрнутые ответы с объяснениями.

Тем не менее, даже самые продвинутые языковые модели сталкиваются с проблемой достоверности знаний. Они обучены на больших корпусах данных и содержат факты в параметрах, но не имеют механизма проверки актуальности информации. Модель может «галлюцинировать» – уверенно выдавать правдоподобный, но неверный ответ, особенно если вопрос выходит за рамки её обучающих данных [38, 45].

Для банковской сферы, где цена ошибки в ответе может быть высока (вплоть до финансовых потерь или нарушения регулятивных норм), это серьёзный риск. Кроме того, языковые модели не знают специфических внутренних данных банка (например, деталей продуктов, процедур или клиентских данных), если эти сведения не были частью их тренировочного корпуса. Обучение же такой модели на внутренних знаниях «с нуля» или регулярное дообучение под актуальные данные – крайне ресурсозатратно и сложно.

Решением указанных проблем стало появление подхода Retrieval-Augmented Generation (RAG) – генерации с дополнительным поиском информации. Идея RAG заключается в комбинировании возможностей больших языковых моделей по порождению текста с доступом к внешней базе знаний для извлечения фактов [45].

При таком подходе перед формированием ответа модель делает поисковый запрос по связанной базе данных (например, коллекции документов банка), находит релевантные сведения и включает их в свой ответ. Это позволяет «приземлить» (ground) генерацию на актуальные проверенные данные и тем самым значительно повысить точность и полезность ответа [45].

Исследования подтвердили эффективность данного подхода: так, модель RAG, предложенная Lewis, объединяла предобученный секвенциальный генератор с внешним индексом Википедии и достигла отличных результатов в задачах открытого ответа на вопрос, генерируя более фактологически корректные ответы, чем модель без поиска [3].

RAG позволяет регулярно обновлять знания системы без полных переобучений модели – достаточно добавлять новые данные в базу знаний. Кроме того, пользователь может получать ссылки на источники сведений, повысив доверие к ответу модели [45]. Благодаря этим преимуществам RAG стал наиболее популярным сценарием использования векторных баз знаний в 2023 году, особенно в сочетании с генеративными моделями для задач поддержки и консультирования [47].

Для реализации подхода RAG требуется специализированная инфраструктура хранения и поиска знаний – так называемая векторная база данных (Vector Database). В такой базе хранятся семантические представления (векторы) документов, и она способна по запросу быстро находить ближайшие по смыслу тексты с помощью алгоритмов approximate nearest neighbors (ANN) [38]. Традиционные реляционные СУБД или поисковые движки не рассчитаны на подобные операции над высокоразмерными пространствами эмбеддингов [2, 43].

Поэтому за последние несколько лет появилось множество новых систем, оптимизированных специально под векторный поиск – таких как Milvus, FAISS, Weaviate, Pinecone, Qdrant и др. Их использование позволяет интегрировать семантический поиск знаний с возможностями генеративных моделей, создавая интеллектуальные решения для поддержки клиентов.

Таким образом, сочетание крупных языковых моделей и векторных баз знаний формирует основу нового поколения интеллектуальных систем автоматизации поддержки. Они способны понимать естественный язык, обращаться к обширной базе знаний в реальном времени и выдавать пользователям точные, обоснованные ответы. Это особенно актуально для банковской отрасли, где автоматизированные ассистенты могут отвечать на типовые вопросы о продуктах, помогать в навигации по сервисам, консультировать по процедурам, значительно разгружая контакт-центры и ускоряя обслуживание клиентов.

Уже сегодня есть успешные примеры: в Бразилии коммерческий банк с помощью чат-бота на базе IBM Watson обработал свыше 181 миллиона запросов клиентов за год, сократив очереди в колл-центры и повысив эффективность обслуживания [32].

Российские банки тоже не отстают – крупнейший банк Сбербанк в 2025 году представил собственного AI-бота GigaChat 2.0 для диалога на русском языке, превосходит по многим показателям GPT4o, DeepSeek-V3, LLaMA 70B и Qwen2.5 [50]. Всё это подчёркивает актуальность темы исследования.

1.2 Обзор существующих решений автоматизации поддержки

Банковский сектор исторически находится на переднем крае внедрения технологий самообслуживания и автоматизации взаимодействия с клиентами. Ещё до эры современных AI-моделей банки применяли различные инструменты для разгрузки служб поддержки: интерактивные системы ответов (IVR) по телефону, обширные разделы FAQ на сайтах, чат-виджеты с

простейшими скриптами и т.д. В последние ~5 лет получили распространение чат-боты – программные агенты, общающиеся с клиентами в мессенджерах или на веб-сайте.

Первые банковские чат-боты обычно работали по принципу распознавания ключевых слов или меню, предлагая пользователю выбрать тему вопроса. Например, бот мог спросить: «Вы хотите узнать баланс счёта или адрес ближайшего отделения?» и по фиксированному дереву диалога дать ответ. Такие решения помогали ответить на наиболее частые вопросы, однако не могли подстроиться под свободный ввод и сложные запросы.

Со временем на рынке появились более интеллектуальные чат-бот платформы. Одним из пионеров была IBM со своей Watson Assistant – платформой, способной понимать намерения (intents) из естественной речи. Крупные банки активно экспериментировали с Watson: известен кейс бразильского Banco Bradesco, внедрившего Watson для ответов клиентам, о котором упоминалось ранее [32]. Другие примеры включают использование платформ IPsoft Amelia, LivePerson, Nuance Nina и др., заточенных под диалоговое обслуживание в финансовой отрасли.

Многие банки разрабатывали решения совместно с финтех-компаниями или внутренними R&D-центрами. Например, в России банк Тинькофф создал собственного голосового помощника «Олег», который в 2024 г. вошел в топ 10 лучших по качеству клиентского сервиса среди банковских ботов [52]. Этот ассистент обрабатывает звонки и чат-запросы, используя комбинацию распознавания речи, NLP и подключение оператора при сложных случаях.

Основные функции, которые реализуют существующие решения автоматизации поддержки в банке, включают: (1) предоставление справочной информации (балансы, тарифы, местоположение офисов, статус операций), (2) помощь в навигации по продуктам и услугам (подбор подходящего кредита, открытия счёта и т.п.), (3) приём обращений и заявок (блокировка карты, изменение лимитов, бронирование встречи), а также (4) базовая техническая поддержка онлайн-сервисов (например, инструкции при затруднениях входа в

интернет-банк). При этом критически важно обеспечить интеграцию бота с внутренними системами банка для выполнения операций – здесь используется API интернет-банка, CRM и другие бекенд-сервисы.

Несмотря на успехи, традиционные чат-боты имели ряд ограничений. Во-первых, требовалась трудоёмкая разметка данных для обучения NLP-модели распознавать пользовательские интенты и слоты (параметры запроса). Во-вторых, бот обычно опирался на статически заложенную базу ответов или сценарии. Поддержание их актуальности требовало участия экспертов (например, обновлять тексты при изменении тарифов). Если же бот не находил подходящего ответа, он выдавал либо «Извините, не понял вопрос», либо переводил на оператора – что воспринималось клиентом как неудовлетворительный опыт. Согласно отраслевым отчётам, доля успешно решённых чат-ботом обращений в некоторых банках поначалу не превышала 20–30%, остальное эскалировалось сотрудникам [32]. Банки пытались улучшить этот показатель, расширяя покрытие сценариев, но натыкались на экспоненциальный рост сложности поддержки базы знаний бота.

Рубежом стал 2022–2023 год, когда на рынок вышли публичные большие языковые модели (GPT-3, GPT-3.5, позже GPT-4) и инструменты для их интеграции. Это позволило создать новое поколение виртуальных ассистентов, которые могут в режиме диалога отвечать на произвольные вопросы, формулируемые пользователями естественным языком. Одним из первых примеров в банковской сфере стало сотрудничество открывшегося финтех-банка Nubank с OpenAI – так например Nubank объявил о планах использовать модели GPT для улучшения клиентского сервиса и внутренних операций[49].

Крупнейшие международные банки (JPMorgan, HSBC и др.) начали экспериментировать с GPT-4 для внутренних справочных чат-ботов, которые помогают сотрудникам быстрее находить информацию о продуктах и процедурах. В финтех-индустрии появилась тенденция создавать специализированные решения: например, в 2025 году компания FIS

(поставщик банковского ПО) запустила продукт Treasury GPT – AI-бот поддержки пользователей своей системы казначейства, встроенный прямо в интерфейс продукта [48]. Этот бот, используя сервис Azure OpenAI, отвечает на вопросы про настройку и использование программного обеспечения, фактически выполняя роль «живого» консультанта для клиентов FIS. Показательно, что решение сразу же получило отраслевую награду, что отражает высокий интерес рынка к подобным инновациям [19].

Отдельно стоит отметить примеры внедрения технологий RAG (генерация с поиском) в службах поддержки. Компания WaveAccess в 2024 году объявила о релизе платформы ValueAI для интеллектуального поиска по корпоративным базам знаний с помощью RAG [53]. Это решение нацелено именно на автоматизацию поддержки и HR-сервисов: оно объединяет загруженную в систему базу документов (например, внутренних инструкций, статей базы знаний) и языковую модель (можно подключить YandexGPT, ChatGPT или LLaMA) для интерпретации запроса и генерации ответа [53]. По сути, ValueAI реализует концепцию чат-бота, который на лету ищет ответ в документации и выдаёт пользователю готовое решение с указанием найденных источников. По заявлениям разработчиков, «поиск с RAG значительно улучшает работу служб поддержки, контакт-центров ... особенно в отраслях с большим объёмом информации, таких как госучреждения, юриспруденция, фармацевтика, финтех»[53]. Это подтверждает, что технология рассматривается как универсальная для различных доменов, в том числе для банковского.

Таким образом, на начало 2025 года можно выделить следующие типы решений автоматизации поддержки, используемые в банковской практике:

1. Статические FAQ и сценарные боты: базовый уровень, покрывающий часто задаваемые вопросы. Пример – меню-боты в интернет-банке, отвечающие на типовые запросы (баланс, реквизиты).

2. NLP-боты второго поколения: используют распознавание интентов и подготовленные ответы. Примеры – внедрения Watson Assistant,

Rasa и аналогов в крупных банках в 2018–2021 гг., голосовые меню с пониманием речи.

3. Генеративные боты без внешней базы: новые решения на базе GPT, которые могут поддержать свободный диалог. Например, консультационный чат-бот на основе GPT-4, обученный на общих данных. Плюс – свободное общение, минус – риск галлюцинаций и недостаток знания специфики.

4. RAG-боты (генеративные с поиском): передовая категория, сочетающая лучшее из двух миров. Они используют мощь LLM для понимания вопроса и формирования ответа, но подкрепляют ответы фактами из базы знаний банка. Это позволяет давать точные и актуальные ответы, включая конкретные цифры, условия продуктов, последние обновления. Именно к этой категории относится разрабатываемая система, посвящённая нашей работе.

Практика показывает, что эффективность автоматизированной поддержки резко возрастает при переходе от первых категорий решений к последним. Если простые чат-боты могут отвечать лишь на 20–30% обращений, то гибридные RAG-боты теоретически способны решать до 70–80% запросов, оставляя человеку только действительно нестандартные случаи [32]. Кроме того, клиенты получают более развернутые и полезные ответы – не просто шаблонные фразы, а пояснения, ссылки на документы, дополнительные советы. Всё это повышает удовлетворённость и сокращает необходимость подключения операторов. Конечно, внедрение подобных систем требует преодоления ряда технических и организационных вызовов (о чём частично говорилось в обзоре литературы – вопросы доверия, ответственности, обновления знаний и др.). Тем не менее, тренд очевиден: банки активно инвестируют в интеграцию языковых моделей и знаний в свои клиентские сервисы, чтобы оставаться конкурентоспособными в эпоху цифрового банкинга[32].

В следующем разделе мы рассмотрим подробнее, какие именно

языковые модели лежат в основе современных интеллектуальных ботов, их эволюцию и характеристики. Это позволит понять, какими возможностями и ограничениями обладают данные модели применительно к задаче автоматизации поддержки.

1.3 Обзор языковых моделей для автоматизации поддержки

Языковая модель – это алгоритм машинного обучения, способный предсказывать продолжение текста на основе обученных статистических закономерностей языка. Революцию в области языкового моделирования произвела архитектура Transformer, предложенная в 2017 году, которая эффективно моделирует взаимосвязи слов в предложении независимо от их расстояния друг от друга. Ключевое свойство трансформеров – механизм самовнимания (self-attention), позволяющий учитывать контекст всего предложения при обработки каждого слова. На базе трансформеров были созданы модели, совершившие скачок в качестве NLP-задач: BERT, GPT, T5 и др.

Рассмотрим кратко наиболее значимые модели, нашедшие применение или представляющие интерес для автоматизации поддержки в банковской сфере.

BERT (Bidirectional Encoder Representations from Transformers) – модель от Google, представленная в конце 2018 г. BERT стал первой предобученной языковой моделью, которая учитывает двунаправленный контекст (слева и справа) на всех уровнях, что сильно повысило качество понимания смысла текста [10]. BERT обучался на задаче маскированного моделирования языка (Masked LM) – угадывания случайно скрытых слов – и задаче предсказания следующего предложения, поглотив огромный корпус текстов (книги, Wikipedia). Результат – универсальные «понимающие» текст эмбеддинги. Как следствие, BERT и его последующие версии (RoBERTa, ALBERT, и др.) стали стандартом для задач классификации текстов, извлечения сущностей, ответов

на вопросы (в формате выборки ответа из текста) и т.п. В контексте поддержки клиентов BERT нашёл применение для быстрого поиска ответов в базе знаний: например, строка запроса и статьи FAQ кодируются BERT-энкодером, после чего ищется наиболее близкая статья по косинусному сходству. BERT заложил фундамент, на котором стали строиться более крупные генеративные модели.

GPT-3 – генеративная предобученная трансформер-модель от OpenAI, выпущенная в 2020 г. Является продолжением идей GPT-2, но на порядок масштабнее. GPT-3 имеет 175 миллиардов параметров и обучалась прогнозировать следующий токен в тексте на корпусе порядка 45 Тб несжатых текстовых данных [31]. Главное открытие, с ней связанное: оказалось, что столь крупная модель обладает способностью к zero-shot и few-shot learning, т.е. может решать новые задачи без специальной донастройки – достаточно дать формулировку задания или несколько примеров в тексте запроса. Статья Brown et al., озаглавленная «Language Models are Few-Shot Learners», продемонстрировала впечатляющие результаты GPT-3 на множестве NLP-бенчмарков именно в режиме без обучения на размеченных данных [31]. Для нашей темы GPT-3 важна тем, что впервые показала: универсальный диалоговый AI реален. Модель может по запросу в свободной форме «Расскажи, как открыть счёт» сгенерировать осмысленный ответ, даже если такая формулировка не встречалась явно при обучении. Ограничения GPT-3 – она обладает фиксированными знанием на момент обучения (срез 2019 г.), может ошибаться в фактах и не позволяет пользователю напрямую обновлять знания.

GPT-3.5 / ChatGPT – следующая веха от OpenAI. В 2022 году была представлена дообученная версия модели GPT-3.5, оптимизированная под диалог с помощью метода обучения с подкреплением от обратной связи с человеком (RLHF) [42]. Известная реализация – ChatGPT, открытый для широкой публики в конце 2022 г. Основное отличие ChatGPT – умение следовать инструкциям и поддерживать контекст диалога, избегая запрещённого контента. Этого удалось достичь за счёт специального

инструкционного дообучения (Instruction Tuning) на примерах пар «запрос-идеальный ответ» и последующего обучения с подкреплением, где модель оценивается и корректируется на основе предпочтений людей.

Результат – модель, которая не просто продолжает текст, а пытается действовать как полезный собеседник, отвечая по существу заданному вопросу. ChatGPT быстро продемонстрировал пригодность для автоматизации поддержки: множество компаний начали создавать на его базе чат-ботов для ответов клиентам. Тем не менее, в корпоративной среде возникли опасения по поводу приватности (передача данных внешнему API) и фактической точности ответов. Это стимулировало интерес к созданию локальных LLM-моделей и к добавлению Retrieval-механизмов.

GPT-4 – наиболее продвинутая на момент написания модель от OpenAI. Отличается от GPT-3.5 рядом улучшений: во-первых, GPT-4 – мультимодальная модель, способная принимать на входе не только текст, но и изображения, что открывает новые сценарии (например, анализ скриншота ошибки в онлайн-банке с выдачей инструкции по её исправлению). Во-вторых, GPT-4 существенно превосходит предшественников в ряде интеллектуальных тестов: как отмечалось, модель сдала барьерный экзамен для юристов в топ-10% лучших результатов, показала отличные результаты в задачах биологии, математики и др [23]. Кроме того, разработчики сообщили об улучшении правдивости и управляемости модели за счёт пост-обучения с фильтрацией и перебора рецептов (правда, точные детали GPT-4 не раскрыты) [23]. В контексте поддержки GPT-4 ценна большим объемом знаний и умением разбираться в сложных запросах. Она может, например, генерировать корректный код на языках программирования, что полезно при поддержке разработчиков (в банках – для внутренних IT-поддержки). GPT-4 легла в основу многих прикладных решений, в том числе Microsoft Copilot и других коммерческих сервисов, и может рассматриваться как эталон качества среди LLM на 2023 год.

Mistral — это линейка открытых языковых моделей нового поколения,

представленная французской исследовательской компанией Mistral AI в конце 2023 года. Ключевым релизом стала модель Mistral 7B, описанная в техническом отчёте «Mistral: A Sparse Mixture of Experts Approach to Efficient LLMs», где была представлена новая архитектура с упором на эффективность исполнения и высокую универсальность. Разработчики сделали акцент на плотной модели без необходимости использования внешней маршрутизации (router-free) и применили методы глубокой регуляризации, благодаря чему Mistral 7B демонстрирует качество, сопоставимое с моделями значительно большего размера, включая LLaMA-13B и LLaMA 2-Chat 13B, при меньших вычислительных затратах.

Модель обучена на тщательно отобранных общедоступных текстовых корпусах: RefinedWeb, StackExchange, Wikipedia, GitHub, академических статьях и других источниках, охватывающих широкий спектр областей знаний. Существенным отличием стало то, что архитектура Mistral ориентирована на эффективность генерации в условиях ограниченных ресурсов, что позволяет запускать её на одной-двух GPU потребительского уровня (24–48 ГБ VRAM). Это делает её привлекательной для предприятий, заинтересованных в развертывании LLM в собственных дата-центрах или edge-инфраструктуре.

Особое распространение получила модификация Mistral:7B-Instruct — модель, прошедшая дополнительное инструкционное дообучение (instruction fine-tuning) на высококачественных диалоговых датасетах. Благодаря этому она адаптирована к задачам общения на естественном языке, включая уточнение запроса, объяснение, перечисление и генерацию регламентированных ответов. В многочисленных сравнительных тестах модель превосходит аналогичные версии LLaMA 2-Chat на равных размерах, особенно в задачах, требующих логических рассуждений и когнитивной строгости. При этом она сохраняет более низкое латентное время отклика и повышенную интерпретируемость вывода.

Запуск модели возможен как с использованием стандартных ML-

фреймворков (Transformers, DeepSpeed, HuggingFace), так и через лёгкие сервисы контейнеризации, например, Ollama, который предоставляет удобный CLI-интерфейс для локального развёртывания. В рамках реализованной RAG-системы именно такая модель (Mistral:7B-Instruct via Ollama) используется как центральный генеративный компонент: она принимает на вход пользовательский запрос и релевантный контекст, извлечённый из векторной базы знаний, и формирует логически связный и фактологически корректный ответ.

Примечательно, что Mistral распространяется под открытой лицензией Apache 2.0, что существенно упрощает её использование в коммерческих проектах. Это обстоятельство особенно важно в условиях растущих требований к защите данных и импортозамещению в ИТ-секторе. Для банковского сектора данная модель представляет собой эффективную альтернативу закрытым API-решениям, позволяя сохранять полный контроль над обработкой информации и соблюдением регуляторных норм. Таким образом, Mistral:7B-Instruct становится оптимальным выбором для интеграции в RAG-решения, обеспечивая локальную генерацию ответов, совместимую с корпоративной политикой безопасности.

Claude – семейство языковых моделей, разработанное компанией Anthropic (основана выходцами из OpenAI). Claude был запущен как закрытый бета-продукт в марте 2023 г., позиционируясь как «этичный чат-бот». Технически Claude сопоставим с GPT-3.5/4, но особенность – метод обучения «Constitutional AI», предложенный Anthropic для повышения безопасности и полезности модели [15]. В этом подходе вместо того, чтобы привлекать людей для оценки вредных ответов, используется сам же AI: модель имеет встроенную «конституцию» из правил (например, принципы из Декларации прав человека ООН) и сначала сама критикует свои ответы на их основе, затем улучшает их [15]. Такой двухфазный процесс (самокритика + RL на основе AI-отзывов) позволил создать ассистента, менее склонного генерировать нежелательный контент без потери в развёрнутости ответов.

Claude 2, выпущенный в июле 2023, значительно расширил возможности: контекстное окно увеличено до 100k токенов (примерно 75 тыс. слов), что позволяет обрабатывать большие документы целиком – ценное свойство для поддержки, где запрос может сопровождаться, например, лог-файлом или выпиской. Также Claude 2 улучшен в математических и программных задачах, что полезно для технической поддержки [13].

Не так давно Anthropic анонсировала линейку Claude 3, заявив о выходе на уровень “near-human” по ряду когнитивных заданий и добавив мультимодальные-возможности обработки изображений [26]. Для банков Claude интересен как альтернатива от независимого вендора (Anthropic), причём ориентированная на корпоративное использование с упором на безопасную работу. Некоторые финансовые компании (например, сервис Notion для бизнес-команд) уже интегрировали Claude для помощи пользователям. В рамках RAG Claude можно применять аналогично GPT-4 – в качестве движка генерации текста, получающего подсказки с документами. Его огромный контекст особого значения в задачах поддержки: бот может «помнить» целиком длинную переписку с клиентом или держать в памяти большую статью Базы знаний, что позволяет давать точные ответы без повторных запросов к БД.

T5 (Text-to-Text Transfer Transformer) – модель от Google, примечательная своей концепцией «всё в тексте». В T5 абсолютно все задачи NLP сведены к единому формату «входной текст – выходной текст» [17]. Например, вопрос и контекст превращаются во входную строку, а ответ генерируется как текст; задача классификации формируется как текст «К какому классу относится: ...?». Такой унифицированный подход позволил одним махом обучить модель на десятках задач и достичь лучших результатов сразу по множеству направлений [17]. T5 имеет размеры от 60 млн до 11 млрд параметров. Хотя сейчас её несколько затмила серия GPT, подход T5 очень влиятелен. Многие современные модели (включая FLAN-T5, UL2 от Google) наследуют идею обучения на смесь разнотипных задач. Для практики

поддержки T5 и её варианты применяются главным образом для автоматизации документации: суммирование длинных текстов (например, длинного запроса клиента или регламента) в короткий ответ, парафразинг, перевод. T5 хорошо себя показывает там, где нужен сгенерировать именно короткий целевой текст на основе входного (summarize, etc.), но для диалогового режима из коробки не предназначена. Тем не менее, на её основе строят различные вспомогательные инструменты, интегрируемые в системы поддержки (например, генерация шаблонов ответов оператору на основе запроса клиента).

Помимо перечисленных, существует множество других языковых моделей и их модификаций: Bloom (176 млрд, многоязычная open-model, 2022), Ernie Bot (китайская модель от Baidu), PaLM и PaLM 2 (от Google, 540 млрд и ~340 млрд параметров соответственно, показавшие сильные стороны в математике и кодогенерации), GLM (General Language Model, популярная open-model в Китае), GALACTICA (специализированная модель для научных текстов от Meta) и др [12, 16, 20, 22, 34]. Некоторые банки и финтех-компании разрабатывают и собственные модели.

Уже упоминался пример GigaChat от Сбербанка – русскоязычной модели, сравнимой с GPT-3.5/4 по возможностям диалога. Её появление свидетельствует о стремлении банков владеть такими технологиями напрямую. Вероятно, в будущем мы увидим больше специализированных LLM, обученных на финансовых данных и терминологии, что повысит качество автоматической поддержки в узкой предметной области (например, ответы на вопросы по инвестиционным продуктам, где требуется понимание финансовых терминов).

Для построения системы автоматизации поддержки на базе языковой модели, критически важно понять сильные и слабые стороны последней. Сильные стороны LLM: превосходное понимание контекста запроса, гибкое генерирование ответов в человеко-понятной форме, умение объяснять шаги (если попросить), многоязычность (многие модели владеют десятками языков,

что актуально для международных банков), обучаемость на примерах (можно задать тон или формат ответа примером).

Ограничения LLM: не гарантируют фактологической точности (могут «выдумать» ответ, если не знают), имеют ограниченные знания о событиях после окончания их обучения, обычно не знают внутренних данных компании, могут быть громоздки для локального развёртывания (требуют мощных GPU и десятков гигабайт памяти). Последний пункт частично решается использованием менее ресурсных моделей (например, LLaMA 13B можно запустить на одном современном сервере) или компрессией моделей (квантованием). Однако в банковской практике нередки случаи, когда чувствительные данные нельзя выносить наружу, поэтому всё больше внимания уделяется именно локальным открытым моделям – их эволюция позволяет уже сейчас достичь приемлемого качества, а в перспективе, возможно, приблизиться к уровню GPT-4 в узкоспециализированных задачах.

В контексте RAG, языковая модель выступает как генератор, который принимает на вход объединённый текст запроса пользователя и извлечённых из базы знаний документов и формирует на этой основе ответ. Для такой роли подходят модели, хорошо обученные следовать инструкциям и генерировать продолжение с учётом предоставленных данных. Практически это означает, что предпочтительны модели типа GPT-3.5/4, Llama-2-Chat, Claude и др., уже настроенные под диалог и выполнение команд. Модели же типа «классический» BERT не годятся для генерации ответа, они используются скорее на этапе retrieval (поиска). Поэтому при построении системы автоматизации поддержки обычно используют связку: энкодер (часто на основе BERT/T5) для поиска релевантных данных + генеративная модель (GPT-тип) для порождения ответа [27]. Далее, в следующем разделе, мы подробнее остановимся на самом подходе Retrieval-Augmented Generation: как он реализуется, каких принципов придерживается и какие выгоды даёт в сравнении с использованием одной только языковой модели.

1.4 Retrieval-Augmented Generation (RAG): принцип работы, применение и польза

Концепция Retrieval-Augmented Generation (RAG) – это совмещение машинного генеративного мышления с поиском внешних знаний. Иными словами, прежде чем сгенерировать ответ, система вспоминает подходящую информацию из внешнего источника (базы знаний) и использует её для обоснования ответа. Данный подход был впервые формализован в работе Patrick Lewis, хотя идеи объединения поиска и генерации обсуждались и ранее [37]. RAG-модель по Lewis представляла собой ансамбль: дифференцируемый модуль поиска по Wikipedia (dense vector index) + генеративный трансформер (BART) на выходе. Эта модель побила рекорды точности в задачах, требующих знаний, таких как открытый вопросно-ответный поиск (open-domain QA), доказав состоятельность подхода[37].

Принцип работы RAG можно описать так:

1. Фаза поиска (Retrieve). Пользовательский запрос сначала обрабатывается модулем поиска. В простейшем случае – это векторный поиск: запрос превращается в эмбеддинг (например, той же моделью BERT/T5) и сравнивается со всеми векторами документов базы знаний. Из базы выбирается топ-N наиболее релевантных фрагментов (документов, абзацев, статей). Например, на запрос «Как поменять суточный лимит по карте?» система может извлечь раздел инструкции, где описывается порядок изменения лимита в мобильном приложении, и раздел тарифов, где указаны ограничения.

2. Фаза генерации ответа (Generate). Исходный запрос пользователя вместе с найденными фрагментами знаний подаётся на вход большой языковой модели. Обычно формируется prompt, содержащий контекст (например: "Context: [текст из базы знаний] \n\n Question: [вопрос пользователя] \n Answer:") – и модель должна выдать продолжение в виде ответа, опираясь на предоставленный контекст. Поскольку фрагменты

включены прямо в вход, модель трактует их как часть своего «диалога» и стремится их учесть при формировании ответа. В результате ответ получается основанным на актуальной информации, даже если сама модель не знала этих фактов изначально.

3. (Опционально) Фаза пост-обработки. Сгенерированный ответ можно дополнительно обработать – например, прикрепить ссылки на документы, откуда взята информация, отфильтровать из него слишком длинные цитаты, форматировать.

Важным элементом является хранение знаний для поиска. Часто используется векторная база данных (см. следующий раздел) для быстрого нахождения похожих текстов по семантике. Альтернативы – классический полнотекстовый поиск (ElasticSearch) или специализированные индексы. Но векторный поиск даёт более гибкий и «понимающий» результат, т.к. найдет даже документы, где формулировки отличаются от запроса (синонимы, перифраз). Преимущества RAG для задач техподдержки и FAQ очевидны.

Актуальность знаний. Модель не ограничена информацией из своих параметров – она всегда может запросить свежие данные. Это решает проблему устаревания: например, если поменялись условия вклада, достаточно обновить запись в базе знаний, и бот тут же начнет давать новые цифры, без переобучения модели.

Фактологическая корректность и проверяемость. RAG существенно снижает риск галлюцинаций. Модель опирается на реальные документы, а не «додумывает». Более того, пользователь может запросить источник: система легко предоставляет цитату или ссылку на документ, из которого взят ответ. Это повышает доверие – особенно важно в банках, где клиенты хотят быть уверены, что информация верна. IBM Research отмечает, что RAG даёт пользователю «прозрачность» – возможность сопоставить ответ модели с оригинальным контентом [45].

Снижение требований к объёму тренировки LLM. Вместо обучения огромной модели на всех документах банка (что фактически нереализуемо),

мы обучаем только эмбеддинговую модель для поиска (гораздо менее ресурсоёмкую) либо используем готовую. Генеративную модель можно взять предобученную общего назначения (как GPT-4 или Llama-2) и просто передавать ей нужный контекст. Это облегчает внедрение – не требуется сбор большой размеченной выборки вопросов-ответов для тонкой настройки. Кроме того, снижаются издержки при использовании API моделей с платой за токены: вместо долгого диалога на выяснение деталей бот сразу получает необходимую справку из БД и генерирует ответ быстрее безопасность данных [28].

При правильной архитектуре внутренняя база знаний может храниться и использоваться локально, а модель – работать с обезличенным контекстом. Таким образом, можно избежать передачи конфиденциальной информации банка в облако внешнего провайдера LLM. Например, если банк решит использовать OpenAI API, он может передавать туда только небольшие релевантные куски своих документов, не раскрывая всю базу или данные клиентов. К тому же, модель не запоминает эти фрагменты надолго – они лишь в текущем запросе. Это снижает риск утечки данных через модель [29].

Применение RAG уже демонстрирует успешные кейсы. Помимо упомянутого примера со Stack Overflow (где поиск по вопросам-ответам реализован через Weaviate и генерацию пояснений), известен кейс компании Databricks, внедрившей бота для поддержки своих продуктов: бот искал по документации и выдавал пошаговые решения (их опыт показал сокращение времени ответа поддержки на 40% при сохранении точности) [24].

В банковской сфере RAG-решения пока находятся в стадии пилотов, но логика их применения ясна. Например, бот-помощник оператора колл-центра: во время звонка клиент спрашивает про сложную процедуру, бот в реальном времени ищет в базе внутренних знаний (регламентов) нужные пункты и подсказывает оператору, что ответить. Или клиентский чат-бот: клиент пишет в чат «Не приходит СМС для подтверждения, что делать?», бот ищет по базе статей «Проблемы с СМС» и даёт разъяснение (проверить связь, перезагрузить

телефон, при необходимости обратиться на горячую линию). Такие ответы будут полными и точными, так как опираются на официальные инструкции банка.

При этом RAG-системы требуют грамотного дизайна, особенно в части подготовки базы знаний и алгоритмов ранжирования. База должна содержать релевантные выдержки: возможно, большие документы (как инструкции на 100 страниц) надо разбить на смысловые части, чтобы поиск выдавал именно точечную информацию.

Векторизация текста должна учитывать язык (для русского – использовать модели, обученные на русскоязычных данных, например RuBERT или multilingual MiniLM) . Иногда добавляют реранжирование результатов: сначала быстрый векторный поиск, затем более тщательная проверка первых 10 кандидатов каким-то нейросетевым реранжировщиком, чтобы выбрать из них топ-3 наилучших. Это улучшает качество извлечённого контекста, что критично – если поиск вернёт не то, генерация тоже не спасёт.

Другой важный аспект – формирование prompt для генеративной модели. Здесь появилось понятие «prompt engineering»: нужно вставить документы в шаблон так, чтобы модель поняла, что от неё хотят. Обычно используют системные инструкции вроде: «Используя приведённую информацию, ответь пользователю. Если информации недостаточно, скажи, что не знаешь. Не изобретай деталей от себя.» Это помогает избежать галлюцинаций, если вдруг в найденных данных нет ответа.

Например, Anthropic Claude хорошо следует подобным конституционным инструкциям и объясняет, если данных мало, вместо выдумывания.

С точки зрения полезности RAG, можно суммировать: RAG сочетает сильные стороны символических систем (точность, проверяемость) и нейросетевых (гибкость, понимание языка). В результате повышается качество обслуживания: клиенты получают быстрые ответы, максимально соответствующие актуальной политике банка.В конечном счёте это приводит

к экономии издержек и улучшению удовлетворённости клиентов.

Несмотря на преимущества, RAG не панацея. Система становится более сложной – надо поддерживать и саму модель, и поисковый индекс. Возможно дублирование информации или противоречия в базе знаний приведут к путанице. Поэтому требуется актуализация и валидация базы знаний: кто-то в банке должен следить, что knowledge base содержит правильные сведения. Также, производительность может страдать, если база огромна – тут на первый план выходят эффективные векторные БД (рассмотрены в следующем разделе). Но современные векторные движки способны обслуживать поиск по миллионам документов за десятки миллисекунд, что вполне достаточно для чат-бота в реальном времени [43].

Подводя итог, Retrieval-Augmented Generation – перспективный и практически значимый подход для построения продвинутых систем автоматизации технической поддержки в банках. Он обеспечивает сочетание интеллекта и знаний, требуемое для точного и контекстно обоснованного ответа клиенту. В дальнейших разделах будет рассмотрено, какие технологии хранения и поиска (векторные базы данных) лежат в основе RAG, и как их выбор влияет на эффективность системы.

1.5 Обзор современных векторных баз данных

Как уже отмечалось, векторная база данных (Vector Database) – это специализированная система управления данными, предназначенная для хранения эмбеддингов (векторов высоких размерностей) и выполнения эффективного поиска ближайших соседей по ним.

Векторные представления возникают повсеместно в задачах AI: это и текстовые эмбеддинги предложений из BERT, и представления изображений из CNN, и много чего ещё. В контексте RAG векторная БД хранит векторные «отпечатки» всех фрагментов базы знаний, что позволяет быстро находить фрагменты, «похожие по смыслу» на запрос пользователя.

Традиционные базы данных (SQL, NoSQL) не ориентированы на такой тип запросов – они оптимизированы под точный поиск по ключу или ограниченный набор атрибутов. Конечно, можно сохранять вектора в обычной таблице, но ни PostgreSQL, ни MongoDB не умеют эффективно искать ближайшие по евклидовой или косинусной метрике вектора среди миллионов записей (без внешних расширений). Были разработаны библиотеки и алгоритмы для approximate nearest neighbors (ANN) – такие как FAISS от Facebook AI, Annoy от Spotify, HNSWlib и др. Они позволяют за милисекунды находить кандидатов среди огромного массива векторов, используя деревья, графы, hashing-методы. Однако библиотеки – это лишь часть дела. Инженеры быстро столкнулись с тем, что нужен полноценный «менеджер данных»: хранение на диске, обновления, распределение по кластерам, фильтрация по метаданным (например, искать среди векторов только того отдела или только свежие) и т.п. [2, 35]. Так появились векторные базы как законченные решения.

Рассмотрим наиболее известные системы этого класса, их особенности, плюсы и минусы применительно к построению RAG в банковском деле.

FAISS (Facebook AI Similarity Search) – по сути не база данных, а библиотека C++/Python для быстрого поиска по векторным индексам, представленная в 2017 г [41]. Это одна из наиболее популярных реализаций ANN-алгоритмов, предоставляющая разные варианты индексов (точные – Flat, IVF; приближённые – HNSW, PQ и др.) и поддерживающая как CPU-, так и GPU-вычисления. Плюсы: высокая производительность, возможность тонкой настройки под нужный trade-off скорость/точность, открытый код. Минусы: не умеет сама хранить данные на кластере, обслуживать обновления, нет встроенной поддержки фильтрации. Обычно FAISS используется «под капотом» у других систем. Например, Milvus и Weaviate изначально интегрировали FAISS как один из движков для поиска, получив его преимущества, но добавив свои механизмы управления данными. В контексте RAG можно использовать FAISS отдельно, если нужны простые оффлайн-

операции (например, разово проиндексировать базу знаний и потом только читать). Но если требуется регулярно добавлять/удалять документы и иметь высокую готовность (кластеризация), лучше выбрать одну из следующих полноценных СУБД.

Milvus – одна из первых специализированных векторных СУБД, старт проекта в 2019 г., развивается компанией Zilliz (Китай) и сообществом, является open-source (лицензия Apache 2.0). Milvus позиционируется как «облачная нейродатабаза для миллиардов векторов».

Среди особенностей , можно выделить высокую производительность на больших масштабах. Архитектура включает поддержку гетерогенных вычислений – CPU + GPU для ускорения поиска. Milvus может распределять данные по нескольким узлам кластера для горизонтального масштабирования и обеспечивает snapshot isolation для согласованности при обновлениях. Milvus на порядки быстрее предшественников (Vearch, PostgreSQL+pgvector) на объёмах свыше 10 млн объектов, а также поддерживает функциональность, отсутствующую у чистых библиотек – фильтрацию, сложные запросы, управление актуальностью данных [2].

Поддержка разных индексов: Milvus позволяет выбирать тип индекса для каждого поля – например, граф HNSW, деревья IVF, PQ (Product Quantization) для экономии памяти и др. Это гибко, можно для небольших наборов использовать точный Flat-индекс, для больших – приближённый.

Интеграции и API: есть клиенты на популярных языках (Python, Java, Go, C++), что облегчает встраивание.

Milvus используется в разных доменах – от поиска похожих изображений до рекомендаций. В контексте RAG Milvus хорошо подходит, если требуется хранить большой корпоративный knowledge base с сотнями тысяч – миллионами статей и делать поиск за десятки миллисекунд.

Банковская база знаний (все инструкции, продукты, ответы) может быть довольно объёмной, поэтому производительность Milvus – большой плюс.

К минусам можно отнести, относительную сложность – разворачивать

кластер Milvus сложнее, чем запустить «локально библиотеку». Требуется настройка Kubernetes-оператора или Docker-композа, плюс ресурсы GPU для максимальной отдачи (не у всех банков есть доступно). Также Milvus написан на C++ и имеет немалый размер, что иногда отмечают как избыточность для простых задач. Тем не менее, для промышленных нагрузок его надежность и масштабируемость – преимущество.

Исследования показывают, что Milvus лучше справляется с динамическими данными, чем, скажем, pure-HNSW решения: при добавлении и удалении больших объёмов он сохраняет производительность благодаря LSM-структуре хранения и распределению на сегменты.

Weaviate – открытая векторная база данных, разработанная стартапом SemiTechnologies (Нидерланды). Вышла примерно в то же время (первые релизы 2019–2020). Weaviate ориентирована на максимально простую интеграцию и богатые возможности поиска. Weaviate хранит не только сами векторы, но и объекты (json-like) с их свойствами. Это позволяет делать гибридные запросы – например, найти объекты по семантическому сходству векторів, но с фильтром по какому-то атрибуту (типа документа, дате и т.д.) [44]. Для поддержки фильтрации Weaviate комбинирует векторный индекс (по умолчанию HNSW) с традиционными структурами для фильтров.

Weaviate предлагает удобный GraphQL-интерфейс для запросов, где можно описать, что искать и какие поля возвращать. Это отличает её с точки зрения developer experience – в нее легко писать запросы.

Weaviate имеет плагины, например модуль текстового векторизатора (она может сама преобразовывать текст в вектор через подключение моделей, таких как SBERT, Transformer models). То есть она может на лету обрабатывать сырой текст, превращая его в embedding – удобно, если нет отдельного этапа расчета эмбеддингов.

Как и Milvus, Weaviate имеет открытый код и одновременно предлагает коммерческий облачный хостинг. Open-source версия под Apache 2.0, написана на Go[7, 39].

Weaviate применяет HNSW (граф ближайших соседей) как основной алгоритм ANN, который обеспечивает очень высокую точность при хорошей скорости на многомиллионных наборах [25]. Согласно независимым бенчмаркам, Weaviate показывает милисекундные ответы на датасетах ~1M векторов при 95-99% точности поиска топ-10 соседей. В одном сравнении она была чуть медленнее Qdrant по RPS, но ненамного [43].

Weaviate активно применяется в проектах семантического поиска. Например, в официальном блоге упоминается, что Stack Overflow применил Weaviate для реализации поиска по вопросам, как обсуждалось ранее [9]. Плюс Weaviate – удобство: разработчики могут быстро запустить её докер-образом и через REST заносить данные (документы, их векторы) и искать.

Так как Weaviate на Go, она менее гибко расширяема в плане собственных алгоритмов поиска (по сравнению с C++ библиотеками). Но это не критично для пользователя. Также в ранних версиях отмечали, что Weaviate потребляла много памяти на граф HNSW, однако новые версии улучшены.

Вокруг Weaviate большая активность, много примеров интеграции с LangChain, Haystack (фреймворки для RAG), что облегчает её применение в нашем проекте [14]. В контексте банков – Weaviate хороша, если хочется быстро сделать MVP: загрузить PDF-документы знаний, встроенным модулем преобразовать векторы и начать отвечать на запросы. Она даже может сама вызывать модель генерации и возвращать сразу готовый ответ (есть модуль генерирующий, но обычно разделяют эту роль).

Pinecone – несколько иного плана система - это проприетарный облачный сервис векторной БД. Pinecone был запущен в 2021 г. стартапом из Нью-Йорка и сразу позиционировался как «Vector DB as a Service». Код Pinecone не открытый, его можно использовать только через API в облаке (AWS, GCP, Azure – выбирает клиент). Pinecone быстро набрал популярность благодаря простоте – разработчик не задумывается об инфраструктуре, развертывании, масштабировании, всё это скрыто.

Pinecone полностью управляемый, не нужно ставить сервер, вы просто

вызываете API: создать индекс (указывая размерность вектора, нужную метрику – косинус/евклид и т.п.), затем добавляете вектора, ищете их. Pinecone автоматически масштабирует хранение и нагрузку, обеспечивает отказоустойчивость. Недавно они внедрили функцию pod scaling, которая позволяет динамически подстраивать размер кластера под объем данных.

Хотя внутренности закрыты, известно, что Pinecone тоже использует HNSW и свой движок хранения. В 2023 г. они добавили возможность подгружать большие объёмы данных быстрее (batch upsert) и сервеpless режим (масштабируемость под нагрузку)[36]. Это привлекательно для переменных нагрузок, экономит ресурсы (платишь только за фактическое использование).

Pinecone часто упоминается в документации OpenAI и примерах как рекомендуемый сервис для реализации памяти и поиска знаний для приложений на GPT-3/4. Это также повысило его популярность.

Pinecone используют когда требуется быстрый старт и не требуется экспертиза в настройке. Pinecone заявляет поддержку миллиардных масштабов. Pinecone предоставляют шифрование и изоляцию данных, что важно для банков, хотя размещение в облаке, конечно, означает доверие внешнему провайдеру.

К минусам можно отнести закрытость (vendor lock-in). Стоимость при больших объемах может быть значительной, так как сервис не дешёвый (биллионы векторов хранить дорого). Кроме того, некоторые банки не могут по политике отправлять свои данные на внешний сервис. В таком случае Pinecone не подходит. Но для финтех-стартапов или для не очень чувствительных данных – это вариант.

С точки зрения функциональности Pinecone пока уступает open-source аналогам в гибкости, например, некоторое время у них не было поддержке фильтров по метаданным (сейчас появилась базовая). Однако в простом кейсе «хранить текстовые эмбеддинги и искать ближайшие» – выполняет на отлично. Согласно экспертам, Pinecone фокусируется на производительности в производственных условиях (низкие задержки при высокой нагрузке) [30].

Pinecone – выбор тех, кому важна скорость развертывания и кто не стеснён регулятивно использовать облако. Его плюс – команда постоянно улучшает (в 2024 добавили распределённость по всем крупным облакам, что говорит о зрелости) [36].

Qdrant – открытый векторный движок, разработанный командой из Европы (в основе – на Rust). Появился около 2021 г., набирает популярность как легковесный и высокопроизводительный вариант. Qdrant сделан с упором на простоту интеграции (REST/gRPC API, клиенты на многих языках) и скорость. Qdrant показывает высокую производительность на CPU, согласно опубликованным Qdrant бенчмаркам, он опережает конкурентов в ряде сценариев по пропускной способности запросов (RPS) и задержкам. Например, в последнем тесте Qdrant показал наивысшее число запросов в секунду и наименьшую медиану задержки среди сравниваемых систем (Weaviate, Milvus, Elasticsearch, Redis) при различных метриках расстояния [43]. Это объясняется эффективностью Rust-реализации и оптимизацией алгоритмов (Qdrant тоже использует HNSW как индекс по умолчанию).

Qdrant акцентирует, что хорошо справляется с обновлениями и добавлением данных «на лету», сохраняя баланс точности и скорости поиска. Это важно для сценариев, где база знаний постоянно дополняется (например, ежедневно новые статьи).

Есть поддержка фильтров по структурным полям – можно хранить с каждым вектором JSON-документ (метаданные) и применять условия (например, category == "credits" при поиске).

Изначально Qdrant был как один узел, но сейчас есть бета-версия поддержки кластера для масштабирования и отказоустойчивости.

Лицензия Qdrant – Apache 2.0 (раньше была просто MIT/Apache), есть и коммерческая облачная версия (Qdrant Cloud), похожая на модель Weaviate/Pinecone.

Qdrant обеспечивает очень быстрый поиск даже без GPU, экономичность по ресурсам, простой деплой (статически скомпилированный бинарь на Rust).

Сообщество хвалит Qdrant за хорошую документацию и примеры (в том числе интеграции с LangChain).

К минусам можно отнести, что функциональность только догоняет лидеров – например, кластерная версия появилась недавно и может быть менее проверена. Но для многих случаев одного узла достаточно или можно шардировать логически.

Для банков, которым нужна on-prem решение с открытым кодом и при этом высокая производительность, Qdrant – отличный кандидат. Особенно если в организации уже есть опыт Rust (для возможных расширений) – но это не обязательно, т.к. API простой. Например, можно за несколько минут поднять Docker-контейнер Qdrant и начать индексировать документы.

Согласно проведённому тестированию, Qdrant достигает топовых результатов по throughput и latency даже на средних аппаратах. В одном сценарии он был ~1.1x быстрее Weaviate и намного быстрее Elastic/Redis по запросам в секунду при точности 99% [43]. Таким образом, для RAG-бота, который должен ответить за десятки миллисекунд, Qdrant – надёжная основа.

Платформа Haystack (от Deepset) добавила Qdrant как один из рекомендуемых бекендов для DocumentStore, отмечая его скорость и активную поддержку разработчиков.

Помимо названных, стоит упомянуть ChromaDB (очень простой встроенный в LangChain векторный стор, written in Python, для небольших проектов), ElasticSearch/OpenSearch (крупные поисковые движки тоже добавили поддержку векторных поисков – удобны тем, что можно комбинировать классический поиск и векторный, но по эффективности специализированным уступают, особенно по скорости индексации больших массивов), Postgres и pgvector (расширение pgvector позволяет хранить вектора в PostgreSQL и делать простейший ANN-поиск, но на больших масштабах Postgres сильно проигрывает спец.решениям, хотя удобен, когда не хочется вводить новый техстек).

Также появились новые проекты: LanceDB, Vectara (облачная векторная

DB с собственным NLP), Redis Vector Similarity (модуль для Redis). Они тоже имеют свои ниши – например, Redis Vector удобен, если уже пользуетесь Redis как кэшем, но он в памяти хранит, что накладно, и при росте данных производительность падает.

Важно отметить, что популярность векторных БД резко возросла благодаря генеративному AI-буму 2023 года. Многие компании поняли, что 80% данных – неструктурированные (тексты, аудио, видео), и эффективно использовать их можно только через эмбеддинги. В результате векторные БД стали фундаментом для приложений типа чатботов, рекомендательных систем, семантического поиска. Например, Amazon добавил поддержку векторных поисков в своем OpenSearch, Spotify разработал свою векторную систему (Voyager) для музыки.

В банковской сфере этот тренд также проявляется: банки аккумулируют огромные знания (регламенты, документация, переписка с клиентами) – обладая инструментом семантического поиска по ним, можно не только автоматизировать поддержку, но и, например, выявлять распространенные проблемы (анализируя, что чаще ищут).

В заключение, векторные базы данных являются критическим компонентом при создании системы автоматизированной технической поддержки на базе RAG. Правильный выбор и настройка такой БД обеспечивает баланс между быстродействием, масштабируемостью и точностью поиска, что в конечном итоге определяет, насколько быстро и правильно бот-помощник сможет находить ответы для клиентов.

Retrieval-Augmented Generation (RAG) сочетает возможности большой языковой модели (LLM) с внешней базой знаний, позволяя решать проблему «галлюцинаций» и устаревших сведений. Как отмечается в приложенном материале (см. [15, 16, 17]), модель перед формированием ответа извлекает релевантные фрагменты документов из векторной базы и «подкрепляет» ими свой результат. Для автоматизации техподдержки в банке этот подход особенно ценен: обновлённые продуктовые условия или внутренние

инструкции оперативно вносятся в базу знаний, и бот сразу отвечает с учётом свежих данных. При этом ключевая особенность RAG — разделение задач: LLM отвечает за понимание вопроса и генерацию текста, а векторный поиск находит нужные факты. Это упрощает масштабирование и облегчает поддержку системы. [4]

По итогам рассмотренных решений и инструментов, наилучшим решением будет использовать локальную модель из семейства mistral:7b-instruct с числом параметров 13B. В тексте (раздел 1.3) указывается на преимущества подобных открытых моделей: их можно самостоятельно развёртывать в инфраструктуре банка, не передавая конфиденциальные данные во внешние облачные API. Модель mistral:7b-instruct обеспечивает удобный диалоговый формат и уже дообучена на инструкциях, поэтому хорошо адаптируется к вопросам клиентов, формируя развёрнутые корректные ответы.

Для хранения и поиска документов выбрана Qdrant — открытый движок векторного поиска на языке Rust. Как следует из упоминаний в тексте (раздел 1.5), Qdrant отличается быстрой работой и простотой развертывания, что подходит для небольшого корпуса документов внутри одного сервера. Под капотом используется алгоритм HNSW, позволяющий быстро находить ближайшие по смыслу фрагменты. Данные хранятся в виде векторов, дополненных метаданными (например, категорией запроса), что облегчает фильтрацию. В сравнении с Milvus, Qdrant менее ресурсоёмок и проще в настройке: Milvus ориентирован на крупные кластеры и может быть избыточен в условиях ограниченной инфраструктуры. При этом Qdrant обеспечивает достаточную производительность и поддерживает обновления базы знаний «на лету».

Таким образом, сочетание RAG, локальной модели mistral:7b-instruct и векторной базы Qdrant даёт гибкую и надёжную систему автоматизации техподдержки. Модель понимает свободные вопросы, а база знаний обеспечивает доступ к фактам, повышая точность и актуальность ответов. Это

решение не требует передачи данных за пределы банка и может эффективно работать на одном сервере, что соответствует требованиям конфиденциальности и ограниченным ресурсам.

2 Архитектура выбранного решения и анализ материалов

2.1 Обоснование выбора технологий

Для автоматизации поддержки в банковской сфере требуется решение, учитывающее специфику доменной области и строгие требования к надежности и безопасности. При разработке архитектуры системы учитывались следующие факторы: необходимость обрабатывать разнообразные запросы на естественном языке, оперативно находить актуальную информацию из большой базы знаний и при этом обеспечить конфиденциальность данных клиента. Традиционные чат-боты на базе сценариев или поиска по ключевым словам уже не удовлетворяют современным требованиям – пользователи ожидают интеллектуальных ответов на сложные вопросы. Поэтому была выбрана архитектура на основе большой языковой модели (LLM) в сочетании с векторной базой данных знаний и подходом Retrieval-Augmented Generation (RAG) для генерации ответов. Такое сочетание позволяет системе понимать разговорную речь, обращаться к внешней базе знаний для фактов и выдавать пользователю развернутый корректный ответ.

Выбор именно данной архитектуры обусловлен потребностями банка в балансировании между качеством ответов и контролем над данными. Использование собственной LLM вместо внешних облачных API обеспечивает локальное хранение и обработку конфиденциальной информации (данные клиентов, внутренние инструкции). Кроме того, банк обладает обширной внутренней базой знаний (документация, справочные материалы, предыдущие обращения), которую целесообразно интегрировать в работу бота. RAG-подход как раз позволяет интегрировать такую базу знаний с генеративной моделью: прежде чем сформировать ответ, модель извлекает из векторного хранилища наиболее релевантные данные. Это повышает точность и обоснованность ответа системы по сравнению с использованием одной лишь языковой модели. В условиях, когда информация о продуктах и

регуляциях часто обновляется, возможность быстро добавлять новые данные в базу без необходимости дообучения модели является существенным преимуществом.

Архитектура построена модульно: каждый компонент (интерфейс, база знаний, языковая модель) отвечает за свою задачу и может масштабироваться независимо. Это решение гибко и масштабируемо – при росте нагрузки можно увеличить мощности для LLM или развернуть кластер векторной базы, не изменяя общей логики. Учитывая опыт успешных внедрений AI-асистентов в банках и рекомендации исследований, приоритезирующих RAG-методы для доступа к корпоративным данным, данная архитектура представляется оптимальной для повышения эффективности техподдержки банка. Ниже приводится описание ключевых компонентов системы и их взаимодействия.

2.2 Описание компонентов системы

В рамках разработанной системы, реализующей архитектуру Retrieval-Augmented Generation (RAG), прослеживается чёткое разделение на функциональные компоненты, каждый из которых выполняет определённую задачу в цепочке обработки данных и генерации ответов. Архитектура построена таким образом, чтобы обеспечить устойчивую и масштабируемую работу сервиса как с точки зрения конечного пользователя, так и в контексте хранения, обработки и анализа текстовой информации. Ниже рассмотрим назначение и особенности каждой из ключевых составляющих.

2.2.1 База метаданных

Несмотря на то, что ядром поиска в данной системе выступает векторная база, традиционная реляционная база данных — PostgreSQL — используется для хранения сопутствующей информации, необходимой для полноценного функционирования системы. Это, в частности, метаданные документов

(названия, типы, принадлежность к пользователю, даты загрузки и обработки), история пользовательских запросов, состояния задач в очереди обработки и данные пользователей (включая роли и токены авторизации). Подобная организация позволяет чётко разграничить функции: PostgreSQL обеспечивает транзакционную целостность и удобную фильтрацию по признакам, которые не имеют прямого отношения к семантическому содержанию, но критически важны для управления данными.

2.2.2 Векторное хранилище

Ключевым элементом системы является Qdrant — специализированная векторная база данных, разработанная для хранения и поиска эмбеддингов высокоразмерных текстовых представлений. В систему загружаются документы, которые перед сохранением разбиваются на логические фрагменты — чанки. Каждый такой фрагмент преобразуется в вектор с использованием модели эмбеддингов (в данном случае поддерживается формат совместимый с E5 или аналогичными Sentence Transformers), и затем этот вектор сохраняется в Qdrant.

Qdrant поддерживает быстрый поиск ближайших соседей с использованием HNSW-графов и предоставляет API, позволяющее осуществлять фильтрацию, ограничение по атрибутам (например, пользовательские коллекции) и гибкую конфигурацию под конкретные задачи. Это позволяет эффективно находить фрагменты, наиболее релевантные пользовательскому запросу, даже если формулировки запроса и оригинального текста различаются лексически.

2.2.3 Серверная логика и API

Все основные операции системы — от загрузки документов до генерации ответов — проходят через серверное приложение, написанное на FastAPI. Этот компонент отвечает за приём и валидацию данных, организацию

очереди асинхронной обработки, управление процессом извлечения текста и его разбиения на чанки, векторизацию и отправку результатов в Qdrant, аутентификацию и авторизацию пользователей, предоставление REST API для фронтенда.

Значимой особенностью является использование очередей задач и фоновая обработка данных — это позволяет загружать большие объёмы документов без блокировки интерфейса, с отслеживанием текущего статуса каждой задачи в реальном времени.

Также стоит отметить, что backend реализует механизм агрегации релевантных фрагментов из векторного хранилища (поиск top-k близких векторов к запросу), формирования prompt'а и его передачи генеративной модели.

2.2.4 Локальная языковая модель

Для непосредственной генерации ответов используется локальная языковая модель Mistral:7b-instruct, развернутая с помощью фрэймворка Ollama. Выбор именно локального варианта объясняется как соображениями безопасности (данные не покидают инфраструктуру), так и технической независимостью от сторонних API. Модель получает на вход специальным образом сформированный prompt, включающий в себя: текст запроса пользователя, набор извлечённых фрагментов из базы знаний и, при необходимости, системные инструкции. На этой основе Mistral формирует осмысленный, связный и, как правило, обоснованный ответ. Это не просто продолжение текста, а попытка решить задачу, опираясь на предоставленный контекст.

2.2.5 Пользовательский интерфейс

Интерфейс системы, реализованный на React, представляет собой полноценный фронтенд-приложение, развёрнутое за прокси-сервером NGINX.

Здесь пользователь может - пройти регистрацию и аутентификацию, загружать документы через форму с drag-and-drop поддержкой, отслеживать прогресс обработки в реальном времени (включая визуальные индикаторы), задавать вопросы через чат-интерфейс, просматривать источники, использованные при формировании ответа, видеть историю своих запросов и ответы модели. Важно подчеркнуть, что интерфейс спроектирован с учётом требований корпоративного использования: поддерживаются пользовательские роли (администратор и обычный пользователь), управление собственными данными, массовая загрузка документов, а также реализация автообновления состояния (через WebSocket или polling), что делает взаимодействие с системой отзывчивым и «живым».Со снимками экрана пользовательского интерфейса можно ознакомиться в приложениях А-В.

2.2.5 Обработка и подготовка данных

Перед тем как данные попадают в векторное хранилище, они проходят предварительную обработку. Сначала извлекается текст (в зависимости от формата — с использованием pdfminer, python-docx и других специализированных библиотек), затем происходит его разбиение. Система поддерживает различные стратегии сегментации: по символам, токенам, логическим блокам. В случае необходимости применяется overlap — перекрытие между чанками, что позволяет сохранить контекст при анализе фрагментами. Уже после этого каждый чанк отправляется на векторизацию и последующую индексацию в Qdrant.

Для поиска по базе знаний реализуется dense retrieval — запрос пользователя преобразуется в вектор, и далее извлекаются наиболее близкие фрагменты. Возможна интеграция механизма rerank (переранжирования) для повышения точности выдачи на втором этапе.

Таким образом, система объединяет в себе современные технические решения и архитектурные подходы, обеспечивающие как высокую

производительность, так и гибкость при работе с пользовательскими данными. Применительно к задаче автоматизации техподдержки в банке, подобная реализация может лечь в основу полноценной платформы, которая способна отвечать на внутренние или клиентские запросы, обогащаться знаниями из загружаемой документации и при этом соответствовать требованиям безопасности и приватности, характерным для финансовой отрасли.

2.3 Интеграция и взаимодействие компонентов

Все описанные компоненты объединены в единую систему, работающую по принципу конвейера запрос-ответ. На рисунке 1 представлена схема архитектуры решения, отражающая основные блоки и потоки данных между ними.

Система включает поток индексации знаний (этап 1) и поток обработки запроса (этап 2). В ходе индексирования документы базы знаний проходят предварительную обработку и преобразуются в эмбеддинги, которые сохраняются в векторной базе данных. При поступлении запроса пользователя система находит соответствующие эмбеддинги в базе и дополняет ими промпт для LLM, генерируя итоговый ответ.Архитектура предусматривает два основных этапа работы: предварительную подготовку базы знаний и непосредственное онлайн-обслуживание запросов. Этап индексирования (offline): исходные данные (база знаний банка: статьи, руководства, FAQ и др.) загружаются в систему и разбиваются на фрагменты оптимального размера. Каждый фрагмент пропускается через модель-векторизатор, которая преобразует текст в эмбеддинг – вектор вещественных чисел, отражающий смысл этого фрагмента.

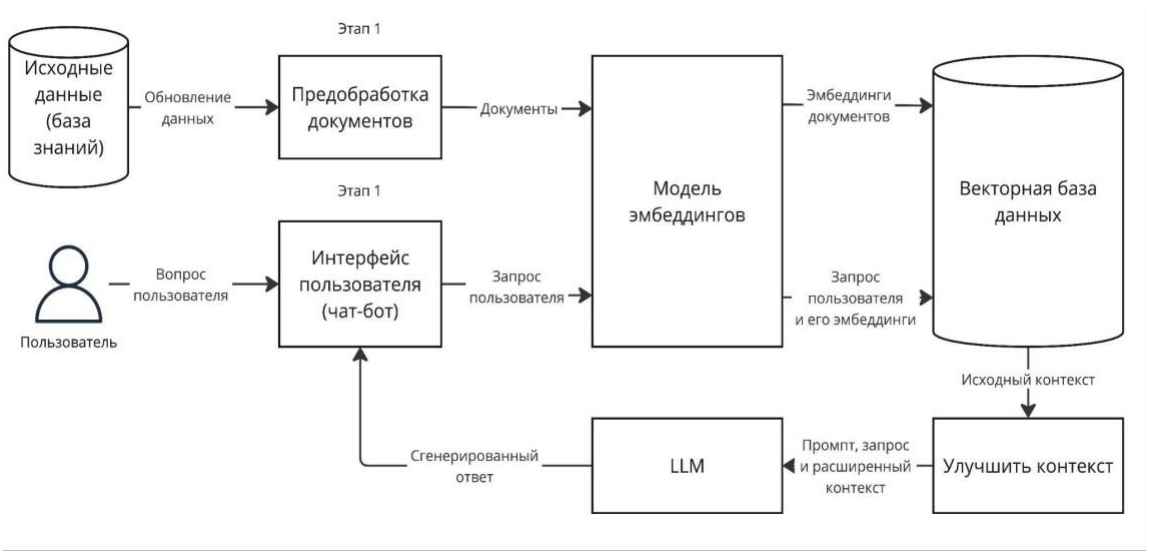


Рисунок 1 — Архитектура предлагаемого решения

Полученные эмбеддинги вместе с ссылками на исходный текст сохраняются в векторной базе Qdrant. Этот процесс проводится один раз для всего корпуса и далее периодически повторяется для обновления (например, при добавлении новых документов или изменении старых).

Этап обслуживания запросов (online): происходит при каждом обращении пользователя в чат. Ниже описан алгоритм формирования ответа шаг за шагом.

Получение вопроса. Пользователь вводит свой вопрос в чат-интерфейсе. Backend системы принимает этот запрос (например, строку текста) через API.

Препроцессинг и векторизация запроса. Вопрос пользователя при необходимости очищается от лишних символов, и из него извлекается смысловая сущность. Затем запрос преобразуется в вектор-эмбеддинг с помощью той же модели-векторизатора, что использовалась на этапе индексации (чтобы пространство векторов совпадало с базой знаний).

Поиск релевантных документов. Полученный вектор запроса отправляется в Qdrant. Векторная база выполняет поиск ближайших соседей – находит в коллекции несколько (например, top-5) наиболее похожих векторов документов. В результате возвращается набор фрагментов текста из базы

знаний, которые потенциально содержат информацию для ответа.

Формирование контекста для ответа. Backend системы собирает контекст из найденных фрагментов. Например, берутся тексты этих фрагментов (или самые информативные части из них), при необходимости объединяются. К контексту также может добавляться сама формулировка вопроса пользователя и мета-данные (например, категория вопроса).

Генерация ответа моделью. Сформированный на предыдущем шаге расширенный промпт (вопрос + найденный контекст) передается на вход локальной языковой модели. Модель обрабатывает вход и генерирует связный ответ на русском языке, учитывая предоставленные факты. На этом шаге благодаря RAG модель опирается не только на свои встроенные знания, но и на конкретные данные банка, найденные в базе.

Возврат ответа пользователю. Сгенерированный моделью текст ответа возвращается через backend в фронтенд. Чат-интерфейс отображает ответ пользователю. При необходимости цикл продолжается – пользователь может уточнить вопрос или задать новый.

Важно подчеркнуть, что все взаимодействие между компонентами происходит в реальном времени (обычно доли секунды на каждый этап). Векторный поиск в Qdrant занимает миллисекунды, а генерация ответа LLM – считанные секунды даже на большой модели (при использовании необходимого аппаратного ускорения). За счет асинхронной архитектуры система способна обслуживать множество запросов параллельно: пока модель формирует ответ на один запрос, другие уже могут выполняться поиск или предобработка для следующих обращений. Также предусмотрены механизмы таймаутов и fallback-логики: например, если модель не смогла дать ответ в разумное время, система может извиниться перед пользователем и предложить обратиться к оператору. Однако в большинстве случаев описанный конвейер позволяет получить законченное решение проблемы без участия человека.

Интеграция компонентов также включает логирование и мониторинг.

Каждый запрос и полученный ответ могут сохраняться (с обезличиванием персональных данных) в журнал – это пригодится для анализа работы системы и улучшения ее качества со временем. Метрики производительности (время поиска, время генерации, использование памяти) отслеживаются для выявления узких мест и масштабирования инфраструктуры. Благодаря модульности, если какой-то компонент требует обновления (например, заменить модель-векторизатор на более точную или обновить версию LLM), это можно сделать без полной переработки системы – через четко определенные интерфейсы взаимодействия. Таким образом, выбранное решение представляет собой связную экосистему, где каждый модуль дополняет другой: база знаний предоставляет факты, LLM – языковое обобщение, а интерфейс – удобство коммуникации с пользователем.

2.4 Сбор и предварительная обработка текстового корпуса

Разработка интеллектуальной системы технической поддержки в банке, основанной на Retrieval-Augmented Generation (RAG), требует наличия качественного и релевантного текстового корпуса. Данная глава описывает процесс формирования датасета на основе материалов сайта https://allo.tochka.com, включая этапы сбора, фильтрации и сохранения данных. Особое внимание уделяется воспроизводимости, устойчивости и полноте извлечения контента.

2.4.1 Источник данных и его особенности

Основным источником текстовых материалов выступил корпоративный информационно-справочный портал банка, содержащий статьи, юридические документы, справки, инструкции и пользовательские материалы. Сайт предоставляет регулярно обновляемые страницы, охватывающие тематику малого бизнеса, налогообложения, отчётности и управления.

Для оценки объема и структуры веб-контента был использован файл

sitemap.xml, содержащий более 3000 URL-адресов. Страницы организованы по тематическим рубрикам, включая разделы education, docs, topic, что обеспечило полноту охвата при сборе данных.

Процесс сбора данных начался с обработки файла sitemap.xml, в котором были перечислены все доступные страницы сайта. Из этого файла были извлечены URL-адреса, после чего применялись фильтры для удаления дубликатов и технических страниц. Это обеспечило чистую и точную выборку материалов.

Для получения содержимого страниц использовалась библиотека httpx, позволяющая работать с асинхронными запросами. Чтобы избежать перегрузки сервера и потенциальной блокировки, запросы выполнялись с ограничением на число одновременных соединений. Был реализован механизм повторных попыток в случае сбоев, таймаутов или защиты от роботов.

После загрузки HTML-страниц производилась очистка от неинформативных элементов: меню, футеров, скриптов и рекламных вставок. Основной текст извлекался с использованием BeautifulSoup и дополнительных селекторов, нацеленных на блоки с содержанием статей. Для каждой страницы извлекались абзацы, заголовки и полезные фрагменты.

На последнем этапе текст проходил базовую нормализацию: удаление лишних символов, конвертация в UTF-8, проверка на пустоту. Каждая страница сохранялась в виде отдельного .txt файла в заранее определённой структуре папок.

2.4.2 Технические меры устойчивости

Так как сайт имел элементы защиты от автоматизированных запросов, было важно реализовать устойчивость скрипта. В частности, задавался пользовательский агент, имитирующий поведение браузера. Также добавлялись небольшие задержки между запросами и реализована логика

повтора при возникновении ошибок или редиректов.

Для контроля качества во время сбора велось логирование всех этапов. Страницы с подозрительно коротким содержимым автоматически попадали в отдельный лог для ручной проверки. Чтобы избежать хранения дубликатов, для каждого текста рассчитывался хэш, и дубликаты не сохранялись повторно. В терминал выводился прогресс с указанием текущей страницы, что позволяло контролировать выполнение процесса.

2.4.3 Результаты сбора

1. Общее количество уникальных и очищенных текстов: 2712
2. Средний размер файла: ~440 слов
3. Отсечено и отброшено дубликатов: ~3%
4. Соотношение страниц с полноценным содержимым к общему числу: более 97%

2.4.4 Практическое значение для системы RAG

Полученный датасет соответствует ключевым требованиям систем RAG:

1. Актуальность: сайт поддерживается банком, регулярно обновляется
2. Структурность: тексты имеют заголовки, абзацы, списки и таблицы
3. Тематика: бизнес, финансы, отчётность, налогообложение — критичные для системы поддержки
4. Релевантность: большая часть документов представляют собой инструкции или справочные материалы

Данный этап обеспечивает фундамент для последующего этапа анализа качества корпуса и построения индекса для RAG-архитектуры. Сбор текстового корпуса был реализован как воспроизводимый и масштабируемый

процесс. Использование sitemap.xml, устойчивый асинхронный парсер и строгая фильтрация обеспечили высокое качество и полноту данных. Полученный датасет применим как основа для последующей векторизации, индексирования и генерации ответов в рамках системы автоматизации технической поддержки.

2.5 Сбор и предварительная обработка текстового корпуса

Эффективность Retrieval-Augmented Generation (RAG) существенно зависит от качества текстового корпуса, используемого для построения индекса и генерации ответов. В данной главе подробно описан подход к сбору метрик, проведение анализа, а также интерпретация результатов в контексте целевой задачи.

2.5.1 Методология оценки датасета

Оценка текстового корпуса проводилась с учётом специфики систем RAG: необходимо обеспечить сбалансированную сегментацию текстов, высокий уровень семантической выразительности, отсутствие избыточного шума и дублирующей информации. Также учитывались требования к структурности, компактности и тематической релевантности.

Были выбраны как традиционные лингвистические и структурные показатели, так и специализированные метрики, важные для построения RAG-систем:

1. Количество слов, символов, строк и абзацев на документ
2. Коэффициент лексического разнообразия (TTR)
3. Доля неалфавитных символов ("шумность")
4. Отношение числа абзацев к числу слов (структурность)
5. Распределение длины документов по категориям
6. Распределение количества чанков на документ при разных размерах (256, 512, 1024)

7. Доля дубликатов (определялась по хэш-сумме содержимого)

Анализ проводился с использованием Python-библиотек pandas, matplotlib, wordcloud и langdetect. Для автоматизации оценки был разработан скрипт, осуществляющий парсинг всех документов, очистку текста, токенизацию, извлечение метаданных и построение графиков.

Данные сохранялись в виде агрегированной статистики (dataset_summary.json) и табличной выборки (dataset_stats.csv). Также была реализована генерация облаков слов (до и после фильтрации стоп-слов) и графическое представление всех ключевых распределений.

2.5.2 Общая характеристика корпуса

Общий объём корпуса составил более 2700 документов. Средняя длина документа — около 450 слов, что соответствует объёму короткой инструкции или статьи. Большинство текстов написаны на русском языке, но были зафиксированы также англоязычные фрагменты, не превышающие 1%.

Среднее количество абзацев - 4.1 на документ. Среднее количество строк- около 22. Коэффициент структурности (абзацы/слова): среднее значение 0.009 — что говорит о средней плотности структуры.TTR: среднее значение ~0.46. TTR < 0.3 наблюдается у 7% документов, что может указывать на слабо вариативные тексты или инструкции с повторяющейся терминологией. На рисунке 2 представлено распределение коэффициента лексического разнообразия.

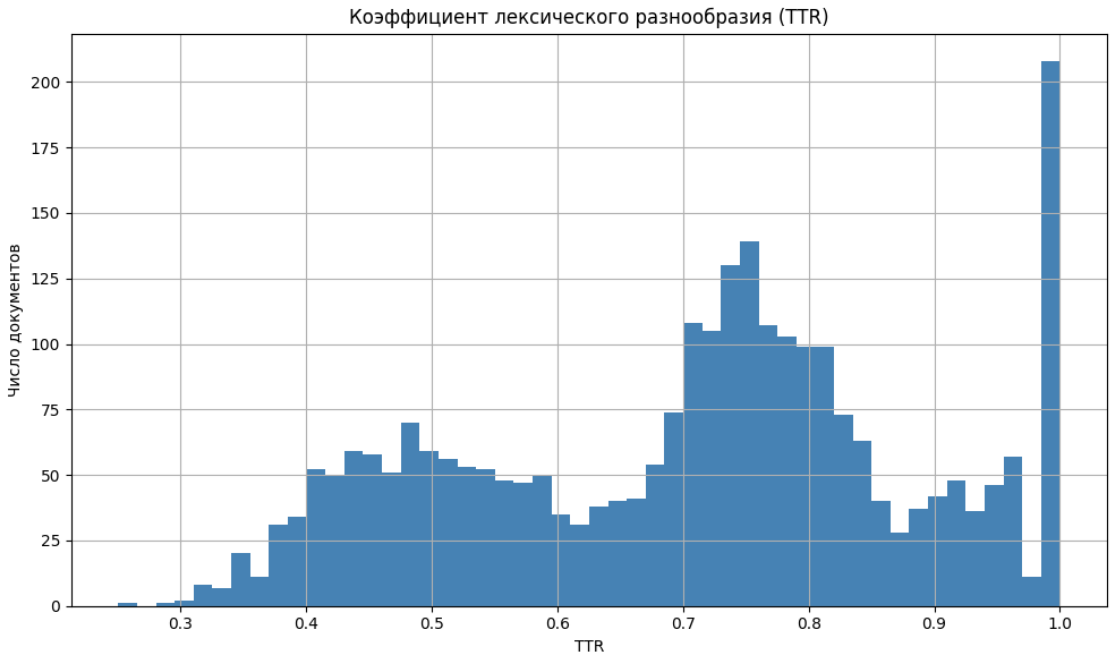


Рисунок 2 — Распределение коэффициента лексического разнообразия

Доля неалфавитных символов: в среднем 3.7%, максимум — до 16%. Это свидетельствует о наличии табличных структур, цифровых значений, ссылок, которые могут нести полезную информацию.

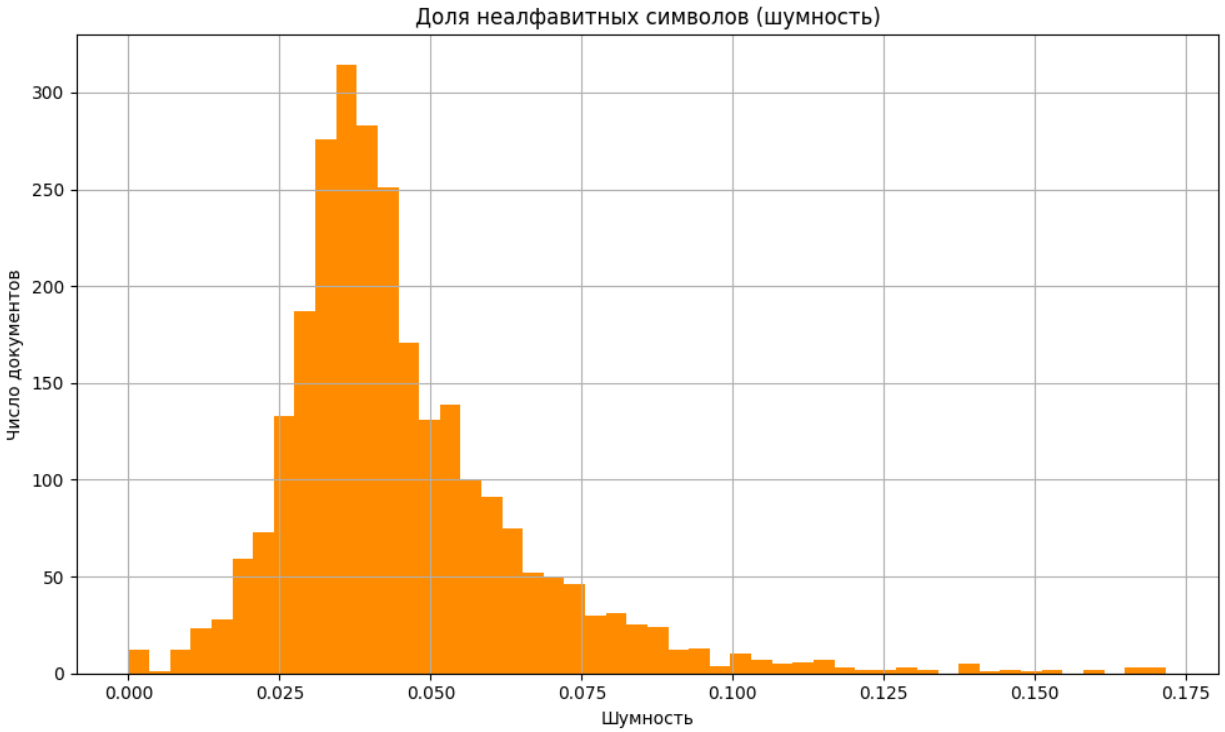


Рисунок 3 — Распределение доли неалфавитных символов

Результаты распределения доли неалфавитных символов представлены на рисунке 3. Выявлено около 3.2% полных дубликатов — они могут быть безопасно удалены из индекса для оптимизации памяти и скорости

На рисунке 4 представлено распределение по восьми категориям длины <100, 100–299, 300-499, 500-699, 700-999, 1000-1499, 1500-1999, >2000 слов. Наиболее частотная категория: 300–699 слов (около 38%)

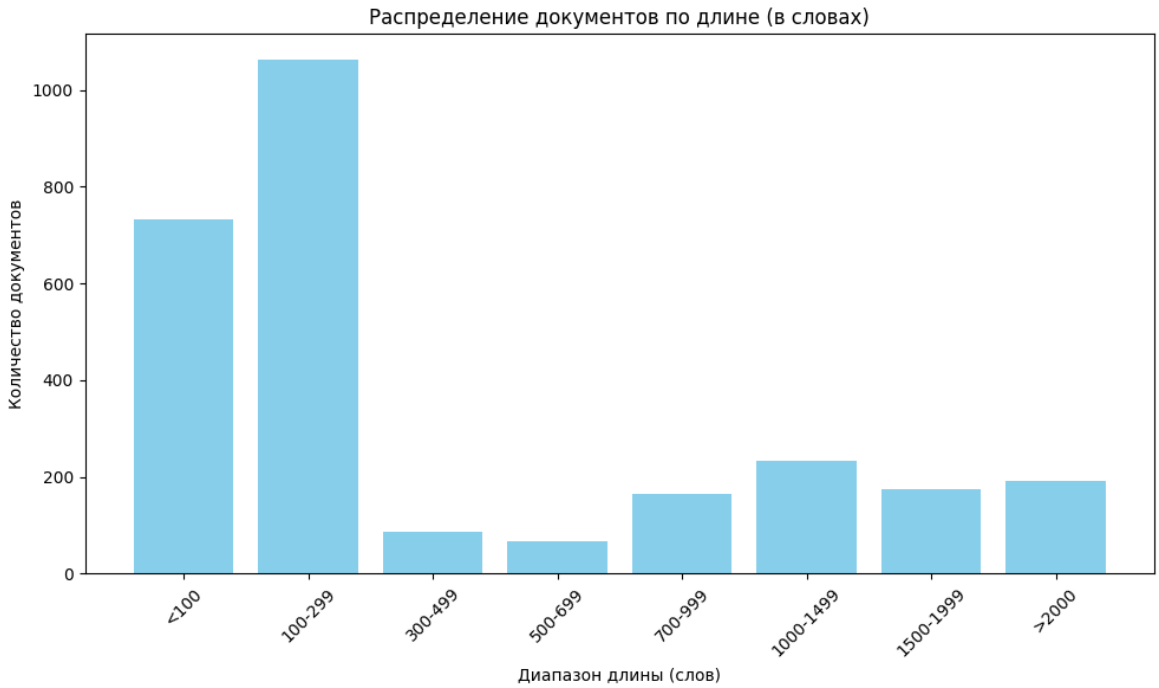


Рисунок 4 — Распределение длины документов по числу слов

Были рассмотрены три значения chunk_size: 256, 512, 1024.

Таблица 1 — Сравнительные характеристики текстового корпуса в чанках

Размер чанка	Среднее число чанков	% доков с 1 чанком	Макс. чанков
256	~2.8	41%	12
512	~1.6	67%	7
1024	~1.1	84%	4

Это даёт основания выбрать 512 как сбалансированное значение большинство документов помещаются в 1–2 чанка, обеспечивается, достаточный контекст для генерации, не перегружается индексная система.

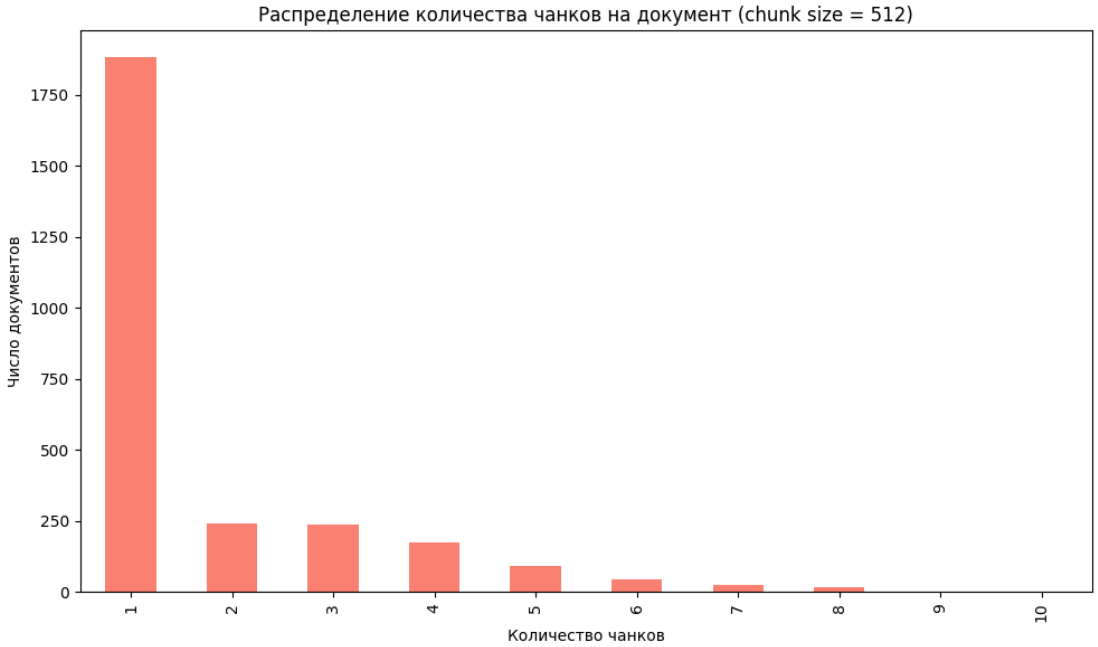


Рисунок 5 — Гистограмма количества чанков на документ

Построены гистограммы по количеству чанков на документ. Как видно из рисунка 5, подавляющее большинство документов разбиваются на 1–4 чанка при размере 512, что делает датасет пригодным для RAG без дополнительных сокращений.
Было создано облако слов на основании данных используемого датасета. Как видно из рисунка 6, в облаке преобладают слова: "договор", "ООО", "банк", "отчёт", "налог" — что указывает на тематическую направленность корпуса (финансы, юриспруденция, операционные процессы).

После удаления союзов, предлогов и служебных слов тематическая структура проявляется отчётливо: "регистрация", "фирма", "отчётность", "налоговая" — ключевые термины, критичные для поддержки SME-бизнеса и клиентов банка. Результаты анализа демонстрируют, что корпус обладает высокой тематической релевантностью, подавляющее большинство документов структурированы и не требуют дополнительной очистки, подбор чанка в 512 слов является оптимальным компромиссом между глубиной

контекста и размером, метрики шумности и TTR указывают на достаточную лексическую и структурную насыщенность без избыточного шума.

Рисунок 6 — Облако слов после фильтрации

Таким образом, полученный датасет может быть эффективно использован для построения индекса в RAG-системе, обслуживающей службу технической поддержки в банке. Проведённый анализ позволяет обосновать архитектурные решения и подобрать параметры сегментации, обеспечивающие баланс между скоростью поиска и качеством генерации ответов.

3 Методика и оценка результатов

3.1 Алгоритм оценки качества

В рамках данной работы была разработана и реализована методика комплексной оценки качества системы Retrieval-Augmented Generation (RAG), направленной на генерацию релевантных и достоверных ответов на пользовательские запросы с использованием заранее загруженной базы знаний. Оценка производительности RAG-модели выполнялась на основе сочетания количественных метрик, статистических методов и экспертной аннотации. Такой подход позволяет получить объективную картину о функциональной пригодности выбранных архитектурных решений.

Методика оценки строилась поэтапно и включала следующие ключевые стадии:

1. Формирование экспериментального корпуса — были отобраны 100 тестовых примеров, включающих пользовательский запрос (question), эталонный ответ (expected_answer) и релевантный фрагмент знаний (expected_context).

2. Генерация ответов — для каждой комбинации (всего 27: 3 варианта разбиения на чанки × 3 retrieval-метода × 3 языковые модели) система RAG генерировала ответ на основе запроса и найденного контекста.

3. Извлечение метрик — для каждого ответа рассчитывались значения по пяти метрикам:

a. BERTScore — измеряет семантическое сходство между сгенерированным и эталонным ответом на основе эмбеддингов BERT;

b. Answer Relevancy — оценивает степень соответствия ответа исходному запросу;

c. Faithfulness — определяет, насколько информация в ответе соответствует фактам, содержащимся в извлечённом контексте;

d. Context Recall — доля релевантной информации из ожидаемого контекста, присутствующей в извлечённом;

e. Context Precision — доля актуальной информации в извлечённом контексте, имеющей отношение к запросу.

4. Расчёт доверительных интервалов — для каждой метрики рассчитывались доверительные интервалы на уровне 95%.

5. Статистический анализ — применялись ANOVA и множественные сравнения (Tukey HSD) для определения значимости различий между комбинациями параметров.

Таким образом, предложенная методика обеспечивает многоаспектную и воспроизводимую оценку качества RAG и позволяет обоснованно выбирать наиболее эффективную конфигурацию системы.

3.2 Результаты эксперимента

Результаты эксперимента представлены в виде количественных оценок по каждому из пяти показателей качества. Для всех 27 комбинаций параметров были вычислены средние значения и доверительные интервалы, отражающие устойчивость и статистическую значимость наблюдаемых различий.

Наибольшее качество было зафиксировано у комбинации Sliding Window + Hybrid Retrieval + Mistral 7B. Эта конфигурация показала стабильно высокие значения по всем пяти метрикам с минимальной дисперсией. В частности:

1. BERTScore — 0.846 ±0.005;
2. Faithfulness — 0.842 ± 0.004;
3. Context Recall — 0.835 ± 0.006;
4. Context Precision — 0.874 ± 0.005;
5. Answer Relevancy — 0.848 ± 0.004.

Результаты по всем метрикам были визуализированы в виде тепловых карт, что позволяет получить обобщённую картину производительности различных архитектурных сочетаний. Каждая тепловая карта отображает значения метрик по осям «метод разбиения» × «метод извлечения»,

усреднённые по моделям. Подробнее с визуализацией метрик можно ознакомиться в Приложении Г.

Проведённый ANOVA-анализ подтвердил наличие статистически значимых различий между методами chunking, retrieval и выбором модели. Наибольшее влияние на итоговое качество оказал retrieval-метод ($p < 0.01$), за ним следуют модель ($p < 0.05$) и метод разбиения ($p < 0.05$). Post-hoc анализ (Tukey HSD) показал, что Hybrid Retrieval статистически значимо превосходит Sparse и CrossEncoder во всех метриках.

3.3 Оценка качества ответов с участием человека

Для более комплексной и реалистичной оценки качества функционирования разработанного RAG-сервиса, наряду с автоматизированными метриками, было принято решение о проведении аннотированной оценки с участием человека. Этот метод, известный как Human QA Annotation, направлен на прямую экспертную интерпретацию качества генерируемых ответов на основе представленного контекста и поставленного вопроса. Включение данного этапа в экспериментальную часть обусловлено ограничениями числовых метрик, неспособных в полной мере отразить семантические нюансы, прагматическую уместность и восприятие информации конечным пользователем.

Для проведения аннотации были привлечены пять респондентов обладающие базовой компетенцией области финансового сектора, а также навыками критического анализа текста. Каждому из респондентов была направлена индивидуальная Google-форма, содержащая 20 кейсов.

Каждый кейс включал:

1. оригинальный вопрос, заданный системе;
2. три релевантных фрагмента, автоматически извлечённых RAG-системой;
3. сгенерированный ответ, полученный от языковой модели на

основе извлечённого контекста.

Респондентам предлагалось провести оценку качества ответа по четырём критериям, каждому из которых соответствовала отдельная шкала от 1 до 5 баллов:

1. Релевантность — соответствует ли ответ поставленному вопросу;
2. Обоснованность — можно ли считать, что ответ выведен на основании предоставленного контекста;
3. Полнота — охватывает ли ответ все ключевые аспекты, содержащиеся в контексте;
4. Понятность — насколько ясно и доступно сформулирован ответ.

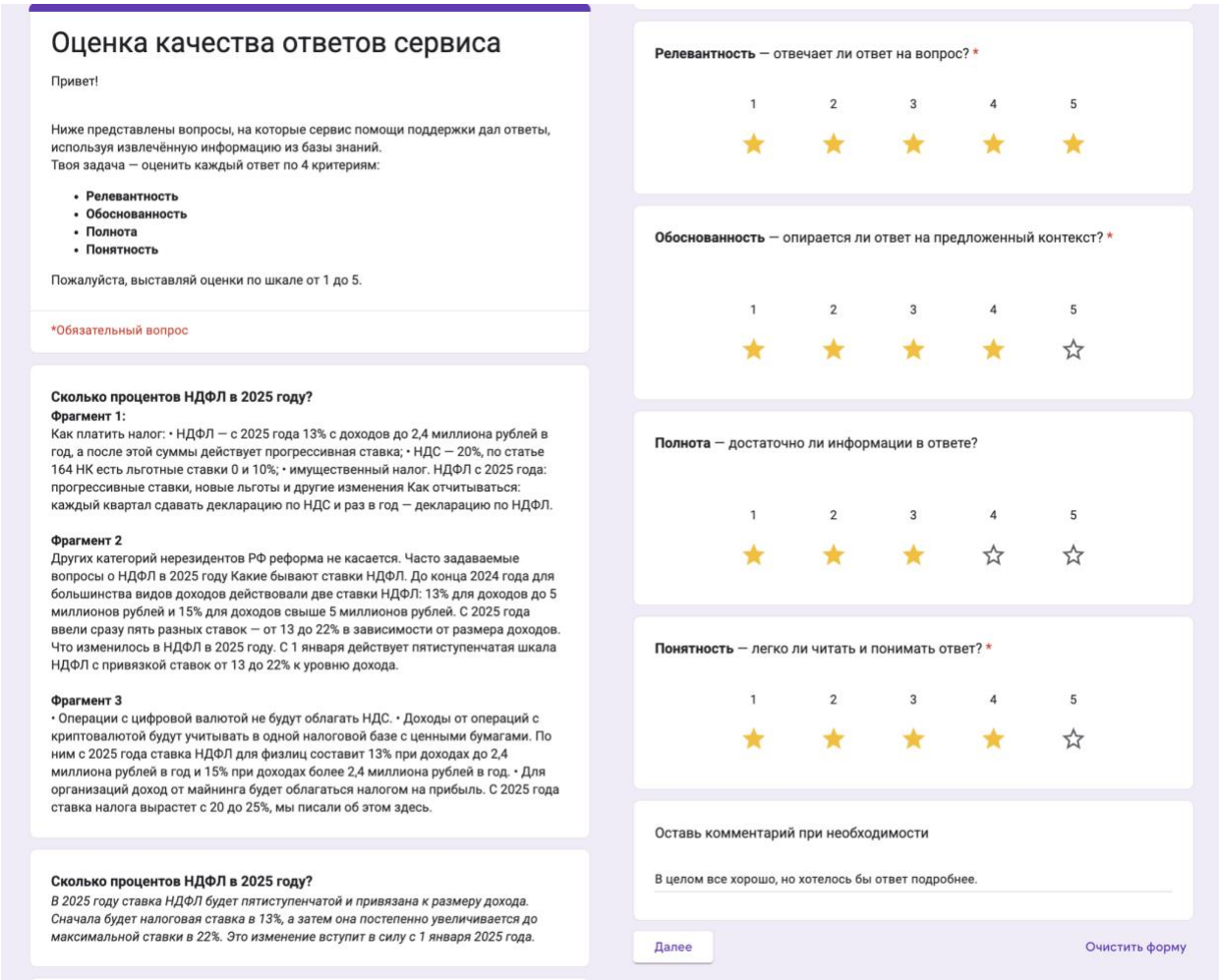


Рисунок 7 — Пример кейса предоставляемого респонденту

Форма включала краткую инструкцию, а каждый блок с кейсом

оформлялся в едином шаблоне для обеспечения когнитивной сопоставимости и сокращения нагрузки на респондента. Пример такой формы представлен на рисунке 7.

Примеры вопросов и фрагментов были отобраны с учётом тематического разнообразия — от регламентных процедур до типовых ситуаций взаимодействия клиента с банком

.Собранные оценки по каждому критерию агрегировались для каждого кейса, а затем усреднялись по всем респондентам. Это позволило вычислить средние оценки и проанализировать вариативность восприятия ответов в четырёх измерениях.

Обобщённые результаты демонстрируют умеренно хорошие показатели по всем четырём критериям:

1. Средняя оценка релевантности колебалась от 4.3 до 4.8, при медиане 4.7, что свидетельствует об уверенном извлечении релевантной информации.

2. Обоснованность получила значения от 4.1 до 4.7, что свидетельствует о достаточно уверенной привязке ответов к извлечённому контексту, при медиане 4.5;

3. По критерию полноты оценки варьировались в интервале 3.9–4.6, при медиане 4.4, что указывает на отдельные случаи неполного охвата информации;

4. Понятность ответов оказалась самой стабильной — от 4.2 до 4.9, при медиане 4.8, что говорит о высоком качестве генеративного вывода в плане стилистики и читаемости.

Респонденты отмечали, что в отдельных случаях ответы выглядели правдоподобно и логично, но при внимательном сравнении с представленным контекстом выявлялось, что часть информации была «додумана» моделью — такой эффект плохо улавливается числовыми метриками, но хорошо фиксируется человеком. Аналогично, наблюдались случаи, когда извлечённый контекст был релевантен, однако модель игнорировала важные

детали или обобщала сверх необходимого.

Проведённая аннотированная оценка показала, что несмотря на высокие результаты по автоматическим метрикам, человеческое восприятие может выявить дополнительные аспекты, влияющие на практическое качество системы, особенно в вопросах обоснованности и полноты.

Таким образом, можно заключить, что интеграция автоматических и человеко-центрированных методов оценки позволяет получить более многогранную и надёжную картину функционирования RAG-системы. Такой подход способствует объективной валидации архитектуры, улучшению качества генерации и выявлению слабых мест, которые критичны при внедрении в реальную среду обслуживания клиентов.

Использование Human QA Annotation, даже в ограниченном объёме, продемонстрировало свою значимость как инструмента прикладной оценки доверия, интерпретируемости и соответствия ожиданиям конечного пользователя. Это особенно важно в банковской среде, где автоматизированные ответы должны быть не только технически корректными, но и восприниматься как надёжные, прозрачные и полезные.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа была посвящена разработке интеллектуального программного средства, способного оказывать помощь сотрудникам технической поддержки в обработке пользовательских обращений. Основной задачей являлось сокращение времени поиска релевантной информации и генерация содержательных ответов с учётом предметной области. В процессе работы был реализован прототип системы на основе архитектуры Retrieval-Augmented Generation (RAG), сочетающей возможности генеративных языковых моделей и механизмов векторного поиска по базе знаний.

Актуальность темы обусловлена растущими объёмами запросов к ИТ-поддержке в организациях, особенно в финансовой отрасли, где предъявляются повышенные требования к конфиденциальности и надёжности. Использование облачных LLM не всегда допустимо из-за ограничений по передаче данных, а традиционные чат-боты, основанные на сценариях, не способны масштабироваться и адаптироваться к динамически меняющейся информации. Эти факторы подчеркивают необходимость в новых архитектурных решениях, обеспечивающих гибкость, автономность и безопасность.

В ходе выполнения выпускной квалификационной работы поставленная цель была достигнута. Все сформулированные задачи реализованы:

1. Проведён обзор современных решений в области автоматизации технической поддержки, включая сценарные чат-боты, облачные и локальные языковые модели, а также архитектуры класса Retrieval-Augmented Generation. Особое внимание уделено применимости RAG в условиях корпоративных ограничений и требований к приватности данных.

2. Выбрана архитектура и программные компоненты системы: в качестве векторного хранилища использован Qdrant, генеративной модели — Mistral 7B-Instruct, а серверное взаимодействие реализовано на FastAPI. Такое

сочетание обеспечивает автономность, гибкость и совместимость с внутренними системами организации.

3. Собран и обработан тематический текстовый корпус на основе актуального и обновляемого корпоративного портала финансовой организации

4. Реализован функциональный прототип сервиса, включающий все ключевые модули: API-слой, компонент поиска, компонент генерации ответа и пользовательский интерфейс.

5. Проведена оценка качества ответов: как количественная (на основе релевантности извлечённых фрагментов), так и качественная (аннотирование экспертом с учётом корректности, полноты и удобства ответа).

В результате экспериментов было подтверждено, что предложенная система позволяет ускорить доступ к информации и повысить точность ответов. В отличие от сценарных чат-ботов, данное решение выступает как вспомогательный инструмент для сотрудника поддержки: оно не подменяет, а усиливает эксперта, помогая сформулировать корректный и обоснованный ответ.

Среди технических и исследовательских особенностей стоит выделить:

1. локальное исполнение генеративной модели без обращения к внешним API;

2. расширяемость базы знаний без переобучения модели;

3. универсальность архитектуры, допускающую интеграцию с тикет-системами или внутренними корпоративными базами знаний.

Практическая значимость разработки заключается в возможности адаптации системы под конкретные отделы или тематики (например, бухгалтерия, безопасность, ИТ).

Прототип может быть доработан в сторону:

1. создания пользовательского интерфейса для внутреннего пользования в helpdesk;
2. дообучения моделей на реальных диалогах;
3. интеграции с базами данных тикетов и обратной связи;
4. внедрения роли оператора-корректора в режиме "человек в цикле".

В заключение стоит отметить, что проведённая работа демонстрирует жизнеспособность подхода RAG с локальными LLM в условиях ограниченного доступа к внешним сервисам. Полученные результаты позволяют рекомендовать разработанный прототип к дальнейшему пилотному внедрению в корпоративной среде. Также возможна публикация и расширение исследования в направлении оценки эффективности генеративных моделей в разных предметных доменах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Brown T. B., Mann B., Ryder N. [и др.] Language Models are Few-Shot Learners // arXiv:2005.14165 [cs]. – arXiv, 2020. – URL: http://arxiv.org/abs/2005.14165 (дата обращения: 13.03.2025).
2. Wang J., Yi X., Guo R. [и др.] Milvus: A Purpose-Built Vector Data Management System // Proceedings of the 2021 International Conference on Management of Data : SIGMOD ’21. – New York : Association for Computing Machinery, 2021. – P. 2614–2627. – URL: https://dl.acm.org/doi/10.1145/3448016.3457550 (дата обращения: 12.03.2025).
3. Lewis P., Perez E., Piktus A. [и др.] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // arXiv:2005.11401 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2005.11401 (дата обращения: 12.03.2025).
4. Wu S., Koo M., Blum L. [et al.] A Comparative Study of Open-Source Large Language Models, GPT-4 and Claude 2 // arXiv:2308.04709 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/2308.04709 (дата обращения: 14.03.2025).
5. Sheth J. N., Jain V., Roy G., Chakraborty A. AI-driven banking services: the next frontier for a personalised experience in the emerging market // International Journal of Bank Marketing. – 2022. – Vol. 40, № 6. – P. 1248–1271. – URL: https://www.emerald.com/insight/content/doi/10.1108/IJBM-09-2021-0449/full/html (дата обращения: 14.03.2025).
6. Andrade I. M. D., Tumelero C. Increasing customer service efficiency through artificial intelligence chatbot // Revista de Gestão. – 2022. – Vol. 29, № 3. – P. 238–251. – URL: https://www.emerald.com/insight/content/doi/10.1108/REGE-07-2021-0120/full/html (дата обращения: 14.03.2025).

7. Apache License, Version 2.0. – URL: https://www.apache.org/licenses/LICENSE-2.0 (дата обращения: 14.03.2025).

8. Hentzen J. K., Hoffmann A., Dolan R., Pala E. Artificial intelligence in customer-facing financial services: a systematic literature review and agenda for future research // International Journal of Bank Marketing. – 2022. – Vol. 40, № 6. – P. 1299–1336. – URL: https://www.emerald.com/insight/content/doi/10.1108/IJBM-09-2021-0417/full/html (дата обращения: 14.03.2025).

9. Ask like a human: Implementing semantic search on Stack Overflow // Stack Overflow. – URL: https://stackoverflow.blog/2023/07/31/ask-like-a-human-implementing-semantic-search-on-stack-overflow/ (дата обращения: 14.03.2025).

10. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // arXiv:1810.04805 [cs]. – arXiv, 2019. – URL: http://arxiv.org/abs/1810.04805 (дата обращения: 14.03.2025).

11. Bhattacharya C., Sinha M. The Role of Artificial Intelligence in Banking for Leveraging Customer Experience // Australasian Business, Accounting and Finance Journal. – 2022. – Vol. 16, № 5. – P. 89–105. – URL: https://www.uowoajournals.org/aabfj/article/id/956/ (дата обращения: 14.03.2025).

12. Workshop B., Scao T. L., Fan A. [et al.] BLOOM: A 176B-Parameter Open-Access Multilingual Language Model // arXiv:2211.05100 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/2211.05100 (дата обращения: 14.03.2025).

13. Caruccio L., Cirillo S., Polese G. [et al.] Claude 2.0 large language model: Tackling a real-world classification problem with a new iterative prompt engineering approach // Intelligent Systems with Applications. – 2024. – Vol. 21. – P. 200336. – URL:

https://linkinghub.elsevier.com/retrieve/pii/S2667305324000127 (дата обращения: 14.03.2025).

14. Combining LangChain and Weaviate // Weaviate. – URL: https://weaviate.io/blog/combining-langchain-and-weaviate (дата обращения: 14.03.2025).

15. Bai Y., Kadavath S., Kundu S. [et al.] Constitutional AI: Harmlessness from AI Feedback // arXiv:2212.08073 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2212.08073 (дата обращения: 14.03.2025).

16. Sun Y., Wang S., Feng S. [et al.] ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation // arXiv:2107.02137 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2107.02137 (дата обращения: 14.03.2025).

17. Raffel C., Shazeer N., Roberts A. [et al.] Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // arXiv:1910.10683 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/1910.10683 (дата обращения: 14.03.2025).

18. Fares O. H., Butt I., Lee S. H. M. Utilization of Artificial Intelligence in the Banking Sector: A Systematic Literature Review // Journal of Financial Services Marketing. – 2023. – Vol. 28, № 4. – P. 835–852. – URL: https://link.springer.com/10.1057/s41264-022-00176-7 (дата обращения: 13.03.2025).

19. FIS Launches Treasury GPT, a Pioneering AI-based Product Support Tool for the Treasury Industry. – URL: https://www.businesswire.com/news/home/20250310669037/en/FIS-Launches-Treasury-GPT-a-Pioneering-AI-based-Product-Support-Tool-for-the-Treasury-Industry (дата обращения: 13.03.2025).

20. Taylor R., Kardas M., Cucurull G. [et al.] Galactica: A Large Language Model for Science // arXiv:2211.09085 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2211.09085 (дата обращения: 14.03.2025).

21. Ghandour A. Opportunities and Challenges of Artificial

Intelligence in Banking: Systematic Literature Review // TEM Journal. – 2021. – P. 1581–1587. – URL: https://www.temjournal.com/content/104/TEMJournalNovember2021_1581_1587.html (дата обращения: 14.03.2025).

22. Du Z., Qian Y., Liu X. [et al.] GLM: General Language Model Pretraining with Autoregressive Blank Infilling // arXiv:2103.10360 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2103.10360 (дата обращения: 14.03.2025).

23. OpenAI, Achiam J., Adler S. [et al.] GPT-4 Technical Report // arXiv:2303.08774 [cs]. – arXiv, 2024. – URL: http://arxiv.org/abs/2303.08774 (дата обращения: 13.03.2025).

24. Hello Dolly: Democratizing the magic of ChatGPT with open models. – URL: https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html (дата обращения: 14.03.2025).

25. HNSW index in depth | Weaviate. – URL: https://weaviate.io/developers/academy/py/vector_index/hnsw (дата обращения: 14.03.2025).

26. Introducing the next generation of Claude. – URL: https://www.anthropic.com/news/claude-3-family (дата обращения: 14.03.2025).

27. Jing Z., Su Y., Han Y. When Large Language Models Meet Vector Databases: A Survey // arXiv:2402.01763 [cs]. – arXiv, 2024. – URL: http://arxiv.org/abs/2402.01763 (дата обращения: 14.03.2025).

28. Jing Z., Su Y., Han Y. When Large Language Models Meet Vector Databases: A Survey // arXiv:2402.01763 [cs]. – arXiv, 2024. – URL: http://arxiv.org/abs/2402.01763 (дата обращения: 13.03.2025).

29. Koga T., Wu R., Chaudhuri K. Privacy-Preserving Retrieval-Augmented Generation with Differential Privacy // arXiv:2412.04697 [cs]. – arXiv, 2025. – URL: http://arxiv.org/abs/2412.04697 (дата обращения:

14.03.2025).

30. Kumar M. The Ultimate Guide to the Vector Database Landscape: 2024 and Beyond. – URL: https://www.singlestore.com/blog/-ultimate-guide-vector-database-landscape-2024/ (дата обращения: 14.03.2025).

31. Brown T. B., Mann B., Ryder N. [et al.] Language Models are Few-Shot Learners // arXiv:2005.14165 [cs]. – arXiv, 2020. – URL: http://arxiv.org/abs/2005.14165 (дата обращения: 14.03.2025).

32. Oyeniyi L. D., Ugochukwu C. E., Mhlongo N. Z. Implementing AI in banking customer service: A review of current trends and future applications // International Journal of Science and Research Archive. – 2024. – Vol. 11, № 2. – P. 1492–1509. – URL: https://ijsra.net/node/3869 (дата обращения: 13.03.2025).

33. Maseke B. F. The Transformative Power of Artificial Intelligence in Banking Client Service // South Asian Journal of Social Studies and Economics. – 2024. – Vol. 21, № 3. – P. 93–105. – URL: https://journalsajsse.com/index.php/SAJSSE/article/view/787 (дата обращения: 14.03.2025).

34. Chowdhery A., Narang S., Devlin J. [и др.] PaLM: Scaling Language Modeling with Pathways // arXiv:2204.02311 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2204.02311 (дата обращения: 14.03.2025).

35. Pan J. J., Wang J., Li G. Survey of Vector Database Management Systems // arXiv:2310.14021 [cs]. – arXiv, 2023. – URL: http://arxiv.org/abs/2310.14021 (дата обращения: 14.03.2025).

36. Pinecone launches serverless vector database on Azure, GCP // TechTarget. – URL: https://www.techtarget.com/searchdatamanagement/news/366609193/Pinecone-launches-serverless-vector-database-on-Azure-GCP (дата обращения: 14.03.2025).

37. Lewis P., Perez E., Piktus A. [и др.] Retrieval-Augmented

Generation for Knowledge-Intensive NLP Tasks // arXiv:2005.11401 [cs]. – arXiv, 2021. – URL: http://arxiv.org/abs/2005.11401 (дата обращения: 14.03.2025).

38. Roller J. Understanding Vector Databases: The Foundation of Modern AI Applications. – URL: https://www.computer.org/publications/tech-news/community-voices/vector-databases-and-ai-applications/ (дата обращения: 13.03.2025).

39. Serverless Vector Database // Weaviate. – URL: https://weaviate.io/deployment/serverless (дата обращения: 14.03.2025).

40. Al-Araj R., Haddad H., Shehadeh M. [и др.] The Effect of Artificial Intelligence on Service Quality and Customer Satisfaction in Jordanian Banking Sector // WSEAS Transactions on Business and Economics. – 2022. – Vol. 19. – P. 1929–1947. – URL: https://wseas.com/journals/bae/2022/d505107-1999.pdf (дата обращения: 14.03.2025).

41. Douze M., Guzhva A., Deng C. [и др.] The Faiss library // arXiv:2401.08281 [cs]. – arXiv, 2025. – URL: http://arxiv.org/abs/2401.08281 (дата обращения: 14.03.2025).

42. Bai Y., Jones A., Ndousse K. [и др.] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback // arXiv:2204.05862 [cs]. – arXiv, 2022. – URL: http://arxiv.org/abs/2204.05862 (дата обращения: 14.03.2025).

43. Vector Database Benchmarks // Qdrant. – URL: https://qdrant.tech/benchmarks/ (дата обращения: 13.03.2025).

44. Weaviate. – URL: https://github.com/weaviate/weaviate (дата обращения: 14.03.2025).

45. What is retrieval-augmented generation (RAG)? – URL: https://research.ibm.com/blog/retrieval-augmented-generation-RAG (дата обращения: 13.03.2025).

46. El-Gohary H., Thayaseelan A., Babatunde S., El-Gohary S. An

Exploratory Study on the Effect of Artificial Intelligence-Enabled Technology on Customer Experiences in the Banking Sector // Journal of Technological Advancements. – 2021. – Т. 1, № 1. – С. 1–17. – URL: https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/JTA.20210101.oa1 (дата обращения: 14.03.2025).

47. 2023 год: год ИИ – Milvus Blog. – URL: https://milvus.io/ru/blog/milvus-in-2023-unprecedented-vector-database-amidst-tech-buzz.md (дата обращения: 13.03.2025).

48. FIS запускает Treasury GPT — новаторский инструмент поддержки продуктов на основе искусственного интеллекта для казначейской отрасли // FinTech Futures. – 10.03.2025. – URL: https://www.fintechfutures.com/techwire/fis-launches-treasury-gpt-a-pioneering-ai-based-product-support-tool-for-the-treasury-industry/ (дата обращения: 14.03.2025).

49. Бразильский Nubank сотрудничает с OpenAI для повышения внутренней эффективности и качества обслуживания клиентов // FinTech Futures. – 11.03.2025. – URL: https://www.fintechfutures.com/2025/03/brazils-nubank-partners-openai-to-enhance-internal-efficiency-and-customer-experience/ (дата обращения: 14.03.2025).

50. Сбер представил обновлённую нейросеть GigaChat 2.0. – URL: https://sber.pro/publication/sber-predstavil-obnovlyonnuyu-neiroset-gigachat-20/ (дата обращения: 13.03.2025).

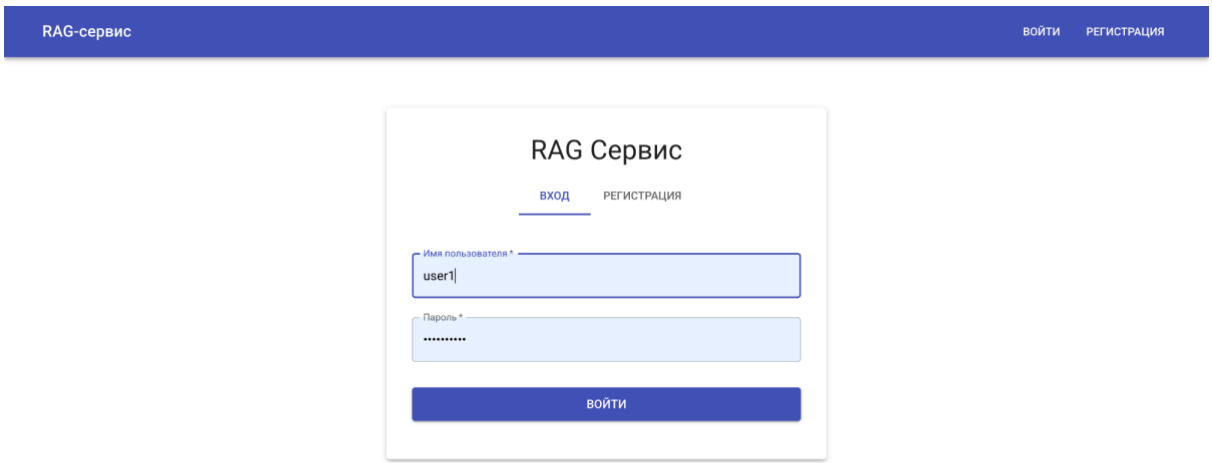
51. Уровень проникновения чат-ботов в банках вырос до 29% — Naumen провел исследование контакт-центров. – URL: https://www.naumen.ru/events/news/6959/ (дата обращения: 13.03.2025).

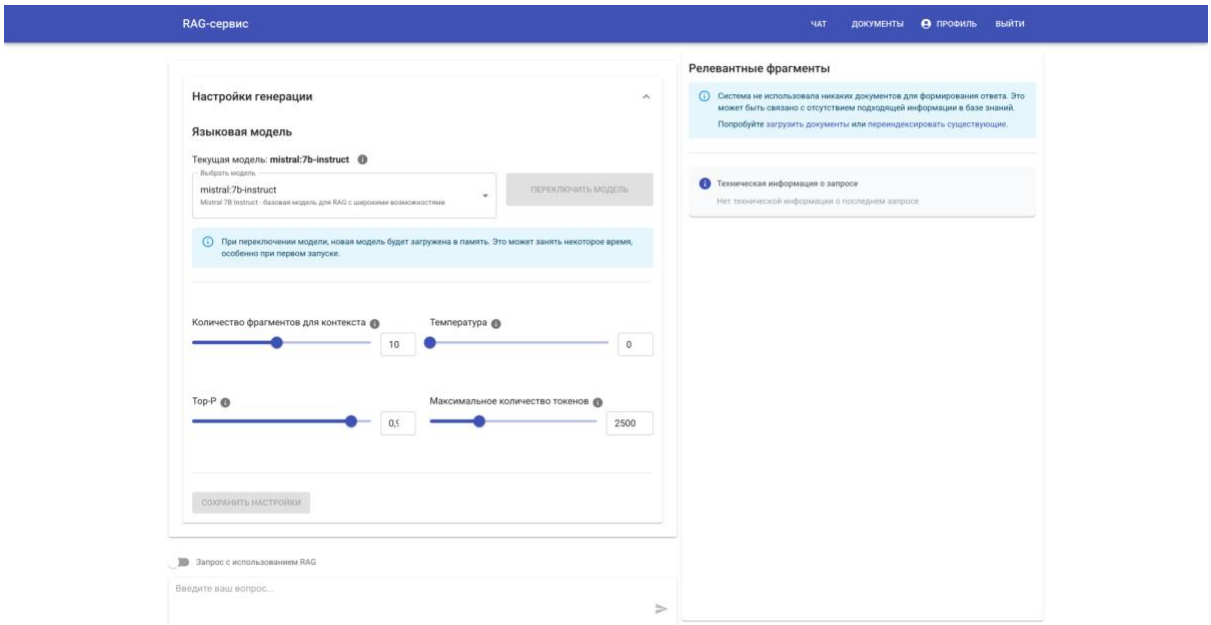
52. Chatbot Rank 2024. – URL: https://www.markswebb.ru/report/chatbot-rank-2024/ (дата обращения: 13.03.2025).

53. ValueAI внедряет умный поиск по базам знаний с использованием технологии RAG. – URL: https://www.itweek.ru/ai/news-company/detail.php?ID=230675 (дата обращения: 14.03.2025).

ПРИЛОЖЕНИЕ А

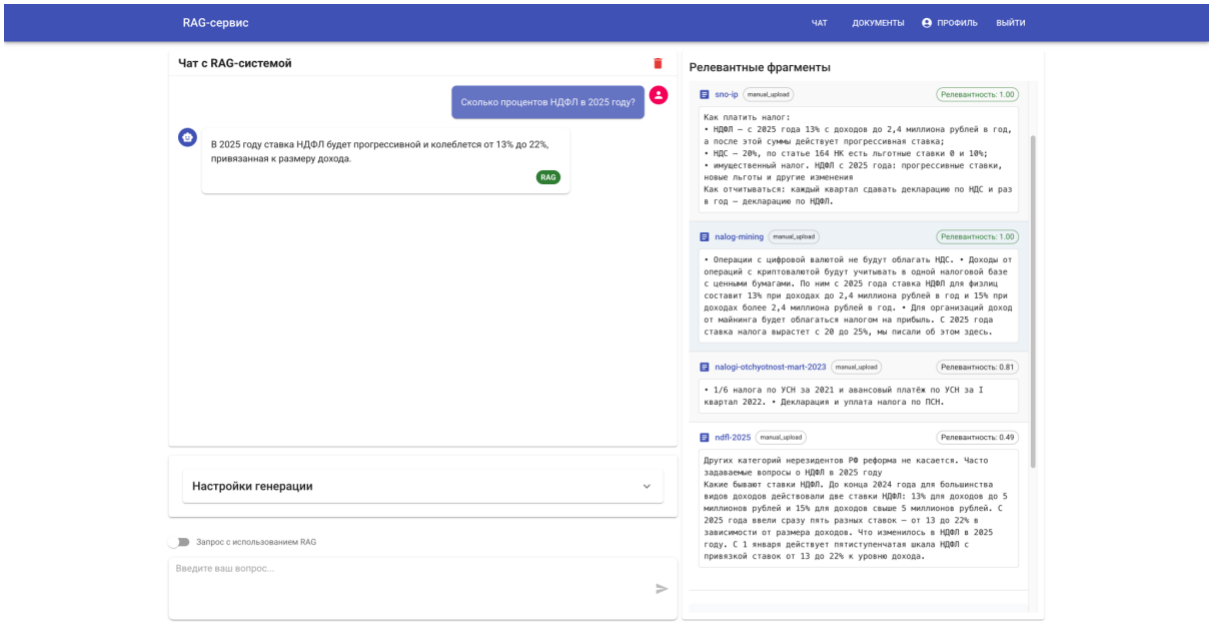
Страница авторизации/регистрации и блок настроек генерации ответа сервиса RAG

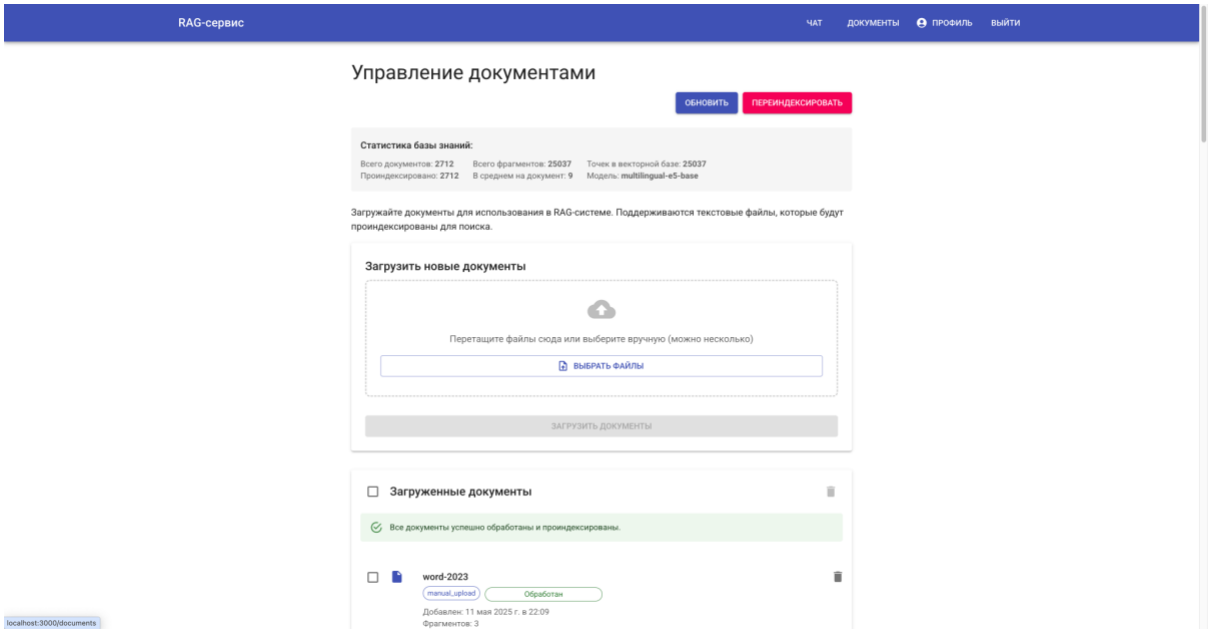




ПРИЛОЖЕНИЕ Б

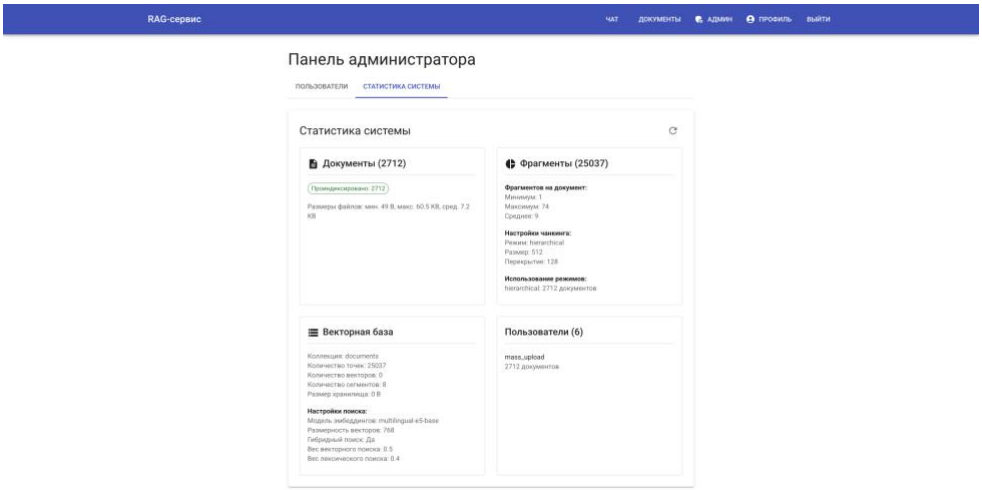
Пример ответа RAG сервиса на вопрос пользователя и страница загрузки документов

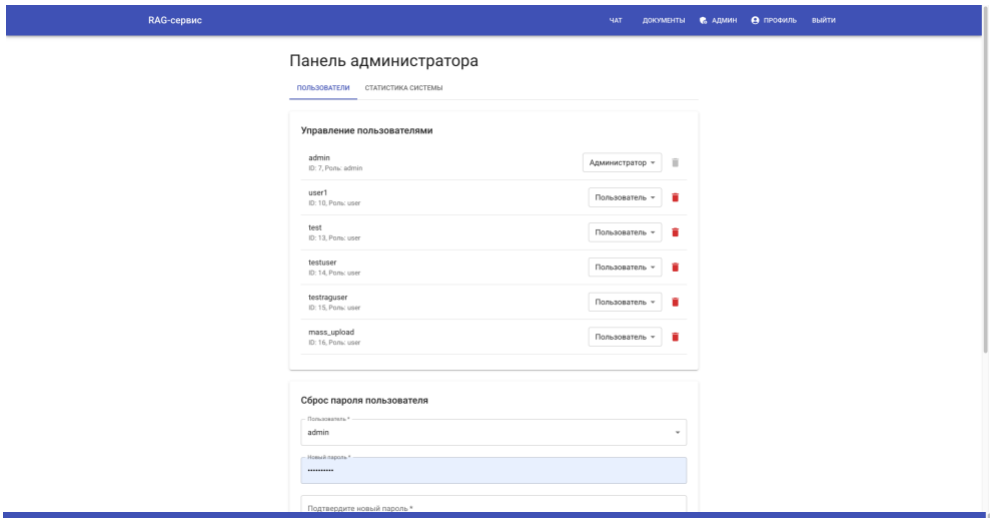


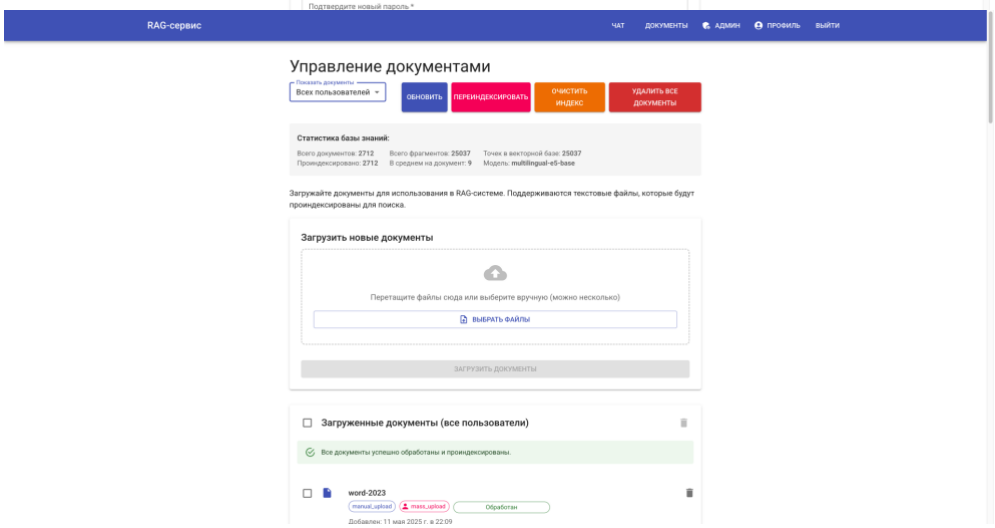


ПРИЛОЖЕНИЕ В

Интерфейс со стороны администратора - статистика системы, управление правами пользователей, управление загруженными документами(своими и всех пользователей)

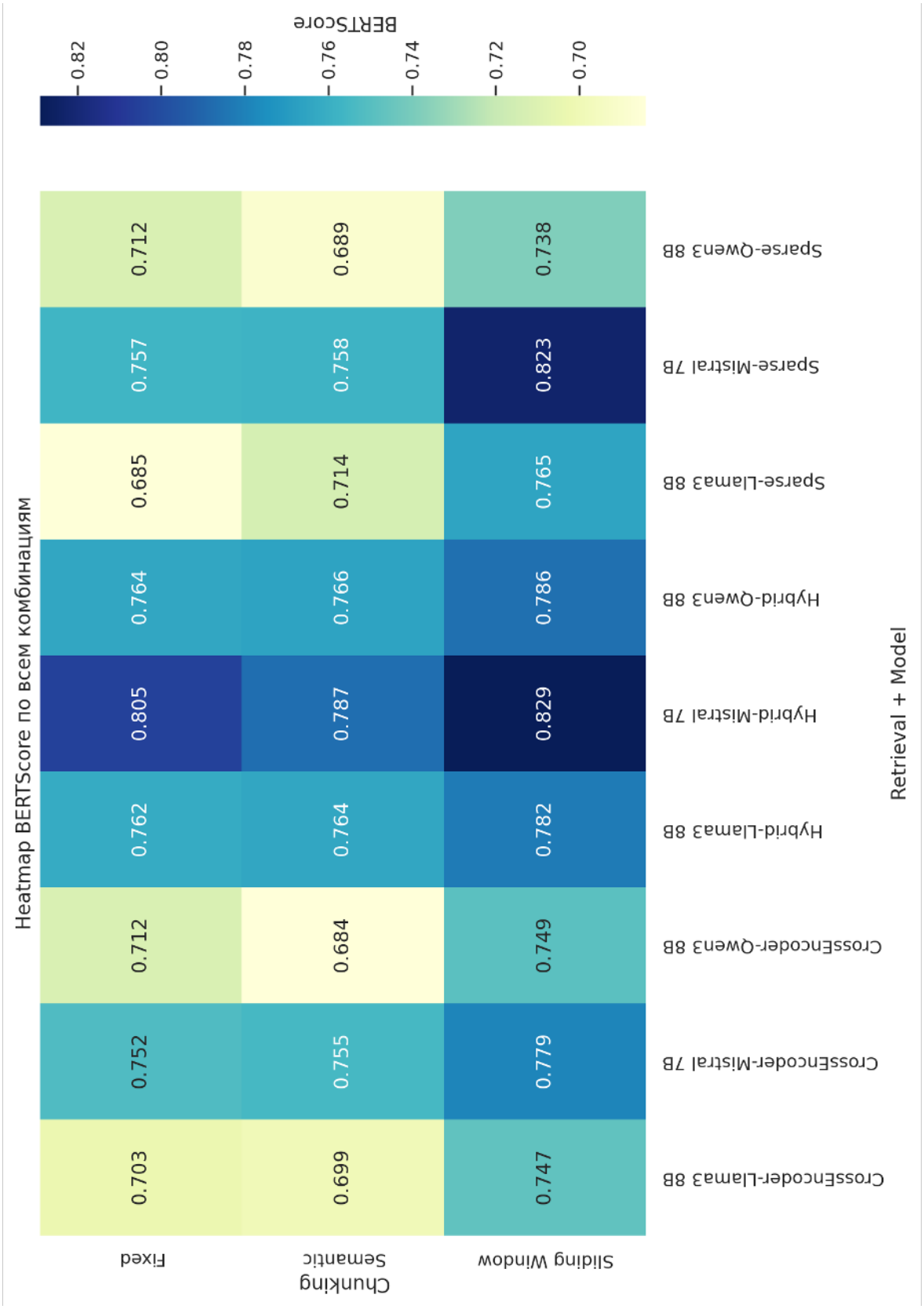




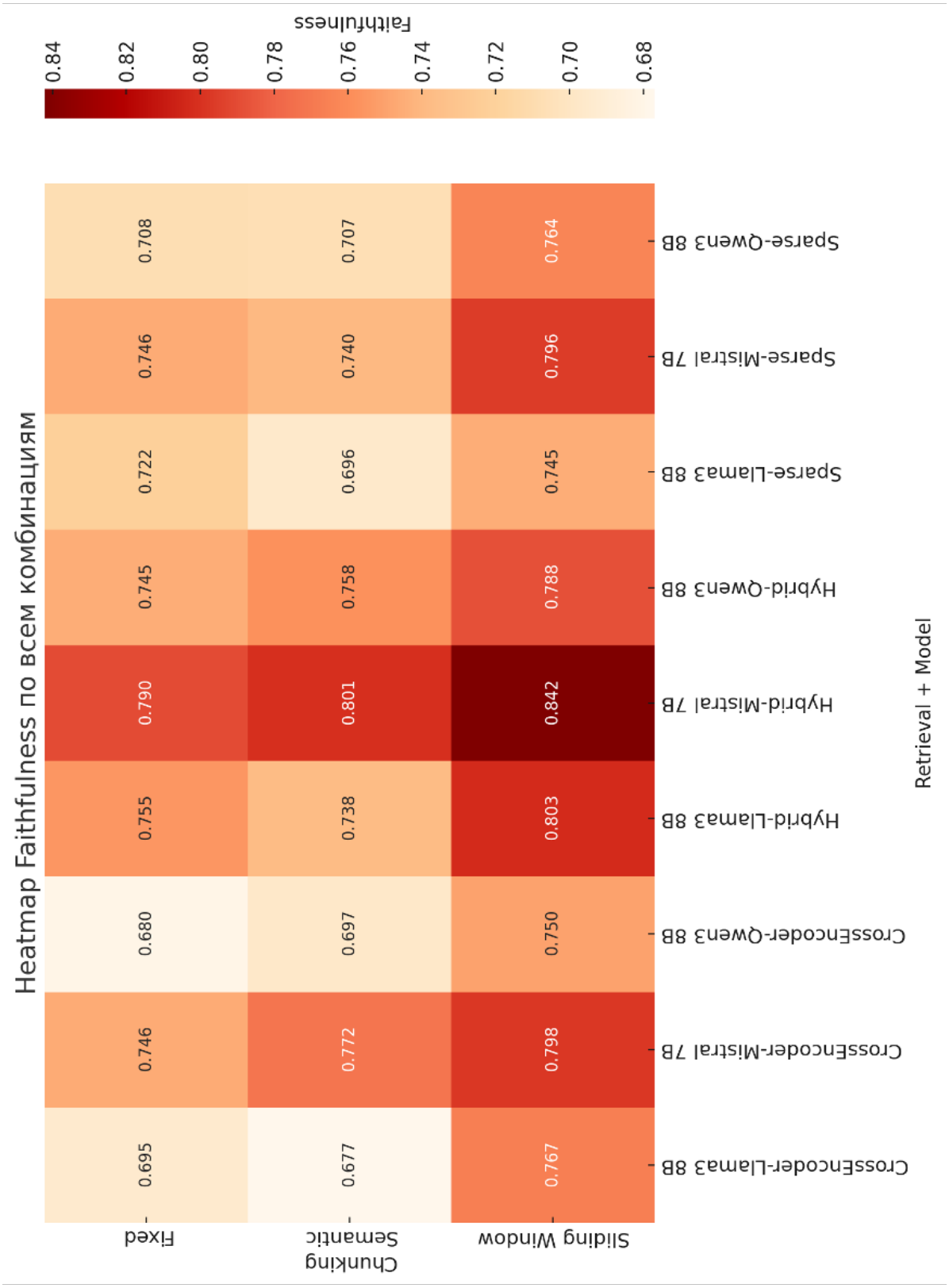


ПРИЛОЖЕНИЕ Г

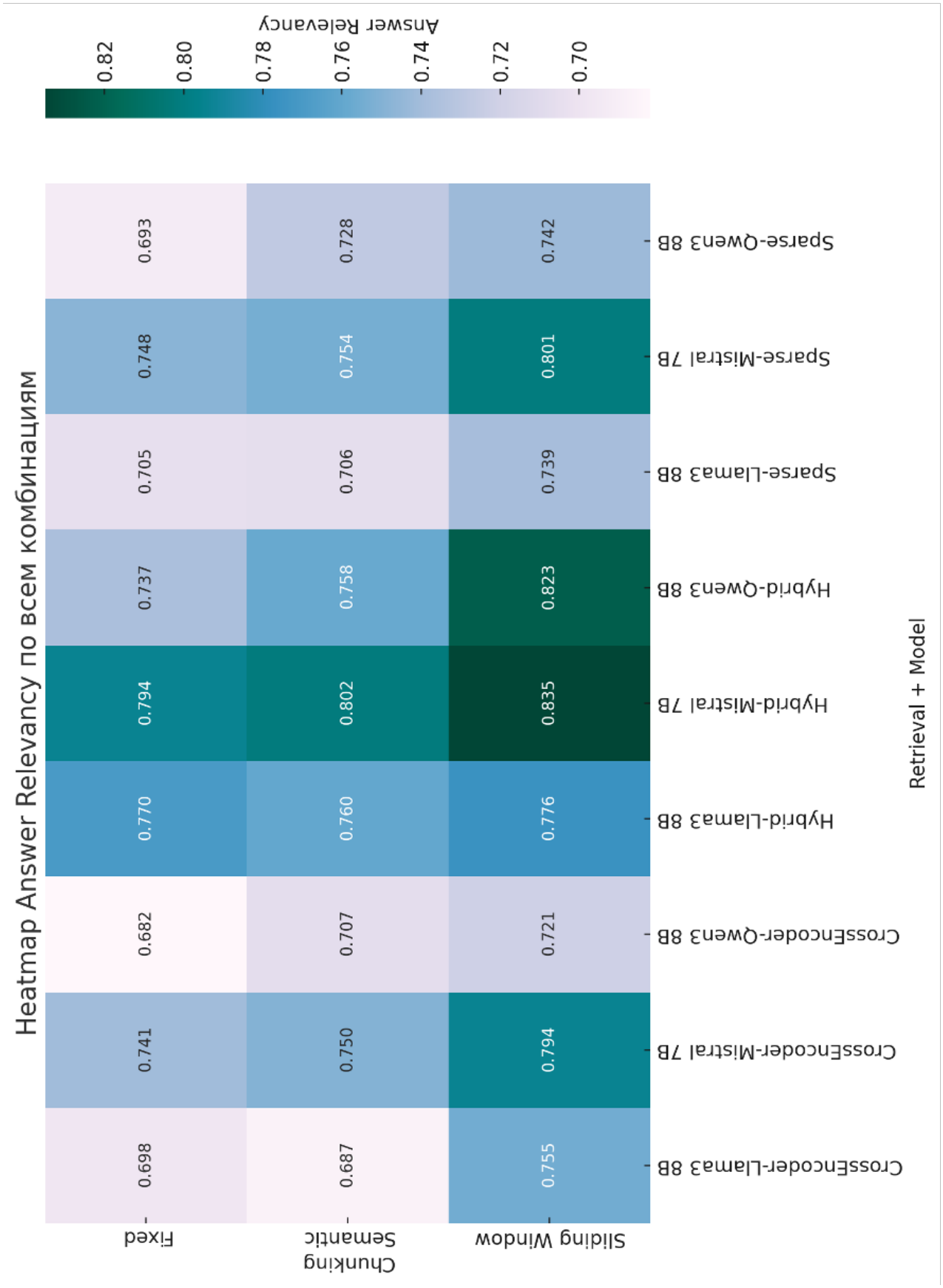
Визуализация метрик сравнительного анализа комбинаций



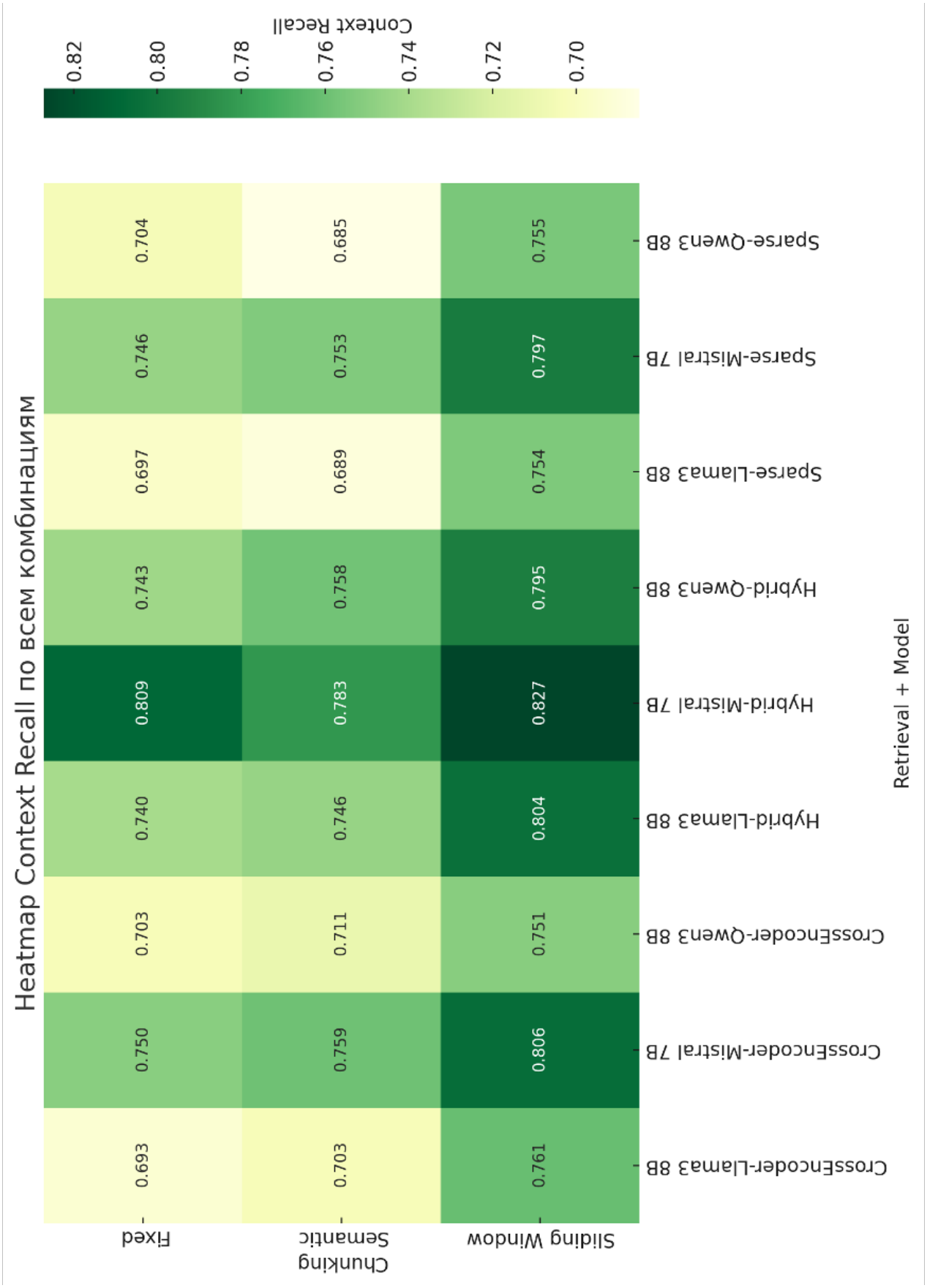
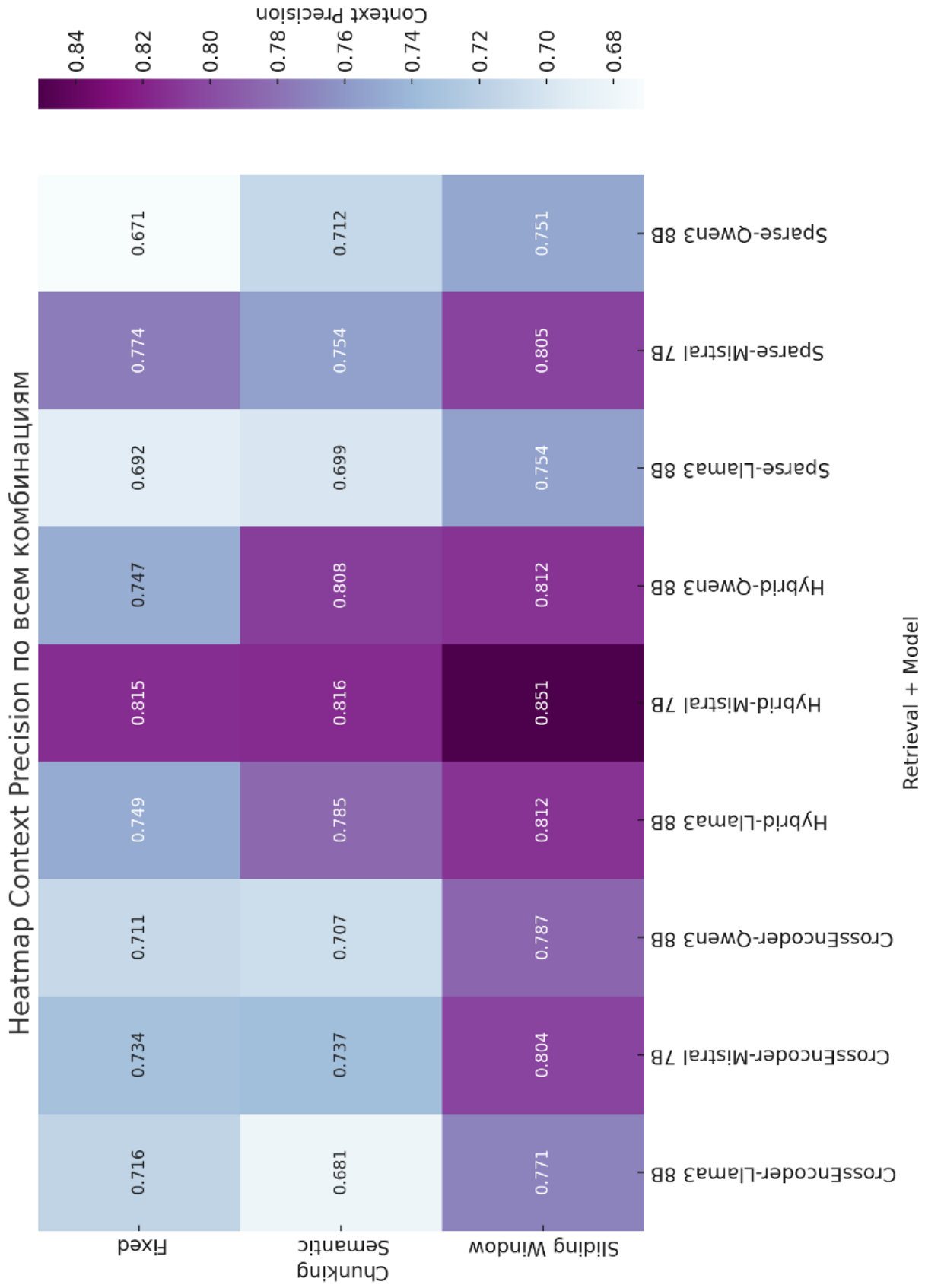
Продолжение ПРИЛОЖЕНИЯ Г



Продолжение ПРИЛОЖЕНИЯ Г



Продолжение ПРИЛОЖЕНИЯ Г



Продолжение ПРИЛОЖЕНИЯ Г