**Министерство науки и высшего образования**

**Российской Федерации**

**ФЕДЕРАЛЬНОЕ Государственное Автономное**

**образовательное учреждение высшего образования**

**«новосибирский национальный исследовательский   
государственный университет»**

**ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНОЙ РОБОТОТЕХНИКИ**

Кафедра Интеллектуальных систем теплофизики ИИР

Направление подготовки 15.03.06 Мехатроника и робототехника

Направленность (профиль) Мехатроника и робототехника

**ОТЧЕТ**

**о прохождении производственной практики, преддипломной практики**

(указывается наименование практики)

**Обучающегося Сыренного Ильи Игоревича группы № 21930** **курса** **4**

**Тема задания**: Разработка интерактивного учебного пособия с ответами на естественном языке на основе Retrieval Augmented Generation

**Место прохождения пратики:** Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Новосибирский национальный исследовательский государственный университет». 630090, Новосибирская область, г. Новосибирск, ул. Пирогова, д. 1

**Сроки прохождения практики:** с 18.03.2025г. по 07.05.2025 г.

**Руководитель практики от НГУ** Галактионова Юлия Юрьевна, специалист УМОВОИИР /

(Ф.И.О. полностью, должность) (подпись)

**Руководитель ВКР**   Оглезнев Никита Сергеевич, сотрудник КафИСТИИР, ассистент /

(Ф.И.О. полностью, должность) (подпись)

**Оценка по итогам защиты отчета:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(неудовлетворительно, удовлетворительно, хорошо, отлично)

**Отчет заслушан на заседании кафедры** КафИСТИИР

(наименование кафедры)

**протокол \_\_\_\_\_\_\_\_\_от** «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_г.

Новосибирск 2025 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc197766977)

[1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ RETRIEVAL AUGMENTED GENERATION 6](#_Toc197766978)

[1.1 Ключевые этапы RAG пайплайна 7](#_Toc197766979)

[1.1.1 Chunking 7](#_Toc197766980)

[1.1.2 Retrieval 8](#_Toc197766981)

[2 ПРОДВИНУТЫЕ ПОДХОДЫ 9](#_Toc197766982)

[2.1 Chunking 9](#_Toc197766983)

[2.1.1 Семантический Chunking 9](#_Toc197766984)

[2.2 Query Rewriting 10](#_Toc197766985)

[2.2.1 HyDE 11](#_Toc197766986)

[2.3 Reranking (Two-Stage Retrieval) 11](#_Toc197766987)

[3 ТОЧКИ ОТКАЗА СИСТЕМ С RETRIEVAL AUGMENTED GENERATION 13](#_Toc197766988)

[4 ОЦЕНКА СИСТЕМ С RETRIEVAL AUGMENTED GENERATION 15](#_Toc197766989)

[4.1 QASPER 15](#_Toc197766990)

[4.2 RAGAS 16](#_Toc197766991)

[4.3 Основные метрики 16](#_Toc197766992)

[4.3.1 Faithfulness 16](#_Toc197766993)

[4.3.2 Response Relevancy 17](#_Toc197766994)

[4.3.3 Answer Correctness 17](#_Toc197766995)

[5 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 19](#_Toc197766996)

[5.1 Дизайн-документ 19](#_Toc197766997)

[5.2 Постановка задачи 19](#_Toc197766998)

[5.3 Архитектура серверной части 20](#_Toc197766999)

[5.4 Клиент 21](#_Toc197767000)

[5.5 Пилотный запуск 21](#_Toc197767001)

[5.6 Требования к работе системы 21](#_Toc197767002)

[5.6.1 Механизмы безопасности 22](#_Toc197767003)

[5.7 UML-диаграммы 22](#_Toc197767004)

[5.7.1 Диаграмма вариантов использования (Use Case) 22](#_Toc197767005)

[5.7.2 Диаграмма последовательностей (Main Sequence) 23](#_Toc197767006)

[5.7.3 Диаграмма компонентов сервиса 24](#_Toc197767007)

[5.7.4 Дополнение: Диаграмма активностей (User Activity) 26](#_Toc197767008)

[6 РЕАЛИЗАЦИЯ ПРОТОТИПА 27](#_Toc197767009)

[6.1 Общая структура системы 27](#_Toc197767010)

[6.2 Серверная часть 27](#_Toc197767011)

[6.3 Итоговая архитектура 28](#_Toc197767012)

[6.4 Клиентская часть 31](#_Toc197767013)

[6.5 Хранение данных 32](#_Toc197767014)

[6.6 Модуль Retrieval-Augmented-Generation 32](#_Toc197767015)

[7 ТЕСТИРОВАНИЕ 34](#_Toc197767016)

[7.1 Датасет 34](#_Toc197767017)

[7.1.1 FRAMES 34](#_Toc197767018)

[7.1.2 Подготовка датасета 35](#_Toc197767019)

[7.2 Конфигурации 35](#_Toc197767020)

[7.2.1 RAG 35](#_Toc197767021)

[7.2.2 Agentic RAG 36](#_Toc197767022)

[7.2.3 LLM 36](#_Toc197767023)

[7.3 Анализ результатов 36](#_Toc197767024)

[7.3.1 Сравнение метрик на разных конфигурациях 36](#_Toc197767025)

[7.3.2 Выводы по метрикам 36](#_Toc197767026)

[ЗАКЛЮЧЕНИЕ 38](#_Toc197767027)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#_Toc197767028)

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

LLM (Large Language Model, Большая языковая модель) – большая языковая модель, обученная на текстовых данных для генерации ответов на запросы пользователя.

Бенчмарк – набор данных для вычисления метрик работоспособности системы в раздичных сценариях.

Эмбеддинг-модель – модель, преобразующая текстовые данные в векторное представление.

Промпт – запрос в большую языковую модель.

ВВЕДЕНИЕ

В современном мире, где темпы технологического прогресса и количество научных публикаций растут с каждым днём, эффективное освоение научной литературы становится ключевым фактором успешной работы в научной и инженерной сферах. Особенно это актуально для специалистов в области информационных технологий, где новые исследования, методы и инструменты появляются с высокой частотой, а актуальность знаний быстро устаревает. Доступ к этим знаниям зачастую ограничивается не только объёмом информации, но и её представлением — подавляющее большинство современных научных статей публикуется на английском языке, насыщенном специализированной терминологией и сложными конструкциями.

Для многих студентов, аспирантов, научных сотрудников и практикующих инженеров это создаёт серьёзные барьеры. Даже обладая базовыми знаниями языка, они сталкиваются с трудностями при попытке интерпретировать содержание статьи, понять значение терминов в конкретном контексте, установить связи между различными понятиями и быстро найти необходимую информацию среди большого объёма текста. Всё это существенно снижает скорость и качество усвоения материала, замедляет научный процесс и усложняет вхождение в новые исследовательские направления.

Дополнительно, задачи, связанные с систематическим изучением больших массивов научной информации, требуют от исследователей значительных временных затрат на поиск, фильтрацию, чтение и осмысление данных. В условиях информационной перегрузки традиционные методы работы с текстами — ручной перевод, чтение и аннотирование — становятся всё менее эффективными. Это делает актуальной задачу создания интеллектуальных систем, способных автоматизировать ключевые этапы взаимодействия человека с научной информацией.

Одним из перспективных направлений решения этой задачи является использование технологий Retrieval-Augmented Generation (RAG) — гибридного подхода, сочетающего механизмы поиска релевантных фрагментов текста с возможностями генеративных языковых моделей. Такие системы позволяют не только находить необходимую информацию в больших корпусах документов, но и формировать на её основе осмысленные и понятные ответы, включая пояснения терминов, краткие аннотации, перевод и ссылки на первоисточники.

Целью данной выпускной квалификационной работы является разработка системы, использующей методы Retrieval-Augmented Generation для поддержки пользователей в процессе изучения научных статей. Система должна обеспечивать автоматический поиск и извлечение информации, пояснение терминов, перевод с английского языка, а также предоставление удобных ссылок на соответствующие источники. Предполагается, что подобный инструмент позволит существенно повысить эффективность научной деятельности, снизить языковые и когнитивные барьеры, а также сократить время на обработку и усвоение информации.

Для реализации поставленной цели были сформулированы следующие задачи:

1. Провести анализ существующих решений в области поддержки чтения научных текстов, выявить их сильные и слабые стороны.
2. Изучить современные архитектуры систем Retrieval-Augmented Generation, определить их применимость к поставленной задаче.
3. Разработать интуитивно понятный пользовательский интерфейс, обеспечивающий эффективное взаимодействие с системой.
4. Спроектировать и реализовать сервис, интегрирующий компоненты Retrieval-Augmented Generation, обеспечивающий потоковое взаимодействие с пользователем.
5. Предложить и реализовать алгоритм оценки качества системы с точки зрения полноты, точности и полезности предоставляемых ответов.
6. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ RETRIEVAL AUGMENTED GENERATION

С развитием технологий обработки естественного языка (Natural Language Processing, NLP) большие языковые модели (Large Language Models, LLMs) продемонстрировали выдающиеся результаты в различных задачах, включая генерацию связного текста, машинный перевод, вопросно-ответные системы и обобщение информации. Эти модели обладают способностью усваивать огромные объёмы знаний из обучающих данных, функционируя как неявная база знаний. Однако такая архитектура имеет существенные ограничения.

Во-первых, языковые модели обучаются на статических наборах данных и не могут обновлять свои знания без полного переобучения. Это делает их уязвимыми к устареванию информации. Во-вторых, в отсутствии внешнего источника верификации модели склонны к генерации некорректной или вымышленной информации — явлению, известному как "галлюцинации" (hallucinations). Эти проблемы особенно критичны в задачах, где требуется точность, достоверность и актуальность ответов, например, при работе с научными текстами.

Для преодоления этих ограничений был предложен подход Retrieval-Augmented Generation (RAG), сочетающий в себе мощь языковых моделей и возможности внешнего поиска по базе знаний. Основная идея RAG заключается в том, чтобы обогатить запрос пользователя дополнительным контекстом, полученным из релевантных источников, и только затем передать его на вход генеративной модели. Таким образом, RAG позволяет обеспечить более достоверные, обоснованные и контекстуально релевантные ответы.

На **Рисунке 1** представлена обобщённая архитектура пайплайна Retrieval-Augmented Generation. Он включает два ключевых этапа: **поиск релевантной информации (retrieve)** и **генерация ответа на её основе (generate)**. На первом этапе система извлекает из внешнего хранилища текстовых данных (например, базы научных статей) фрагменты, наиболее подходящие под запрос. Затем эти фрагменты вместе с оригинальным запросом подаются на вход генеративной модели, которая формирует ответ, используя как внутренние знания, так и предоставленный контекст.

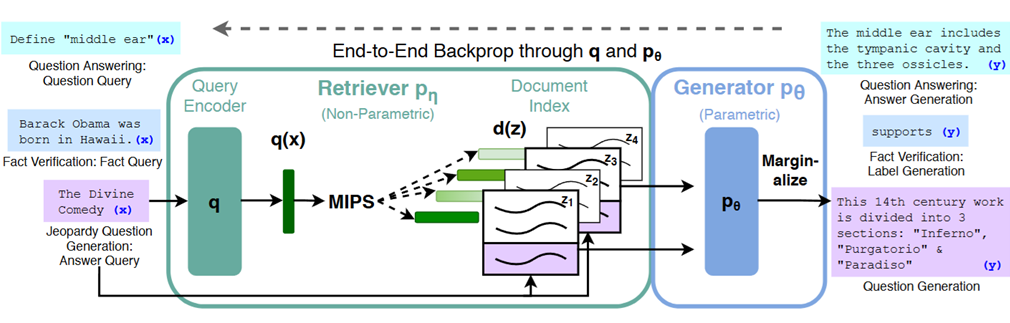
****

Рисунок 1 - Устройство пайплайнаRetrieval Augmented Generation

* 1. Ключевые этапы RAG пайплайна

Подход RAG состоит из ряда модульных этапов, каждый из которых играет важную роль в обеспечении качества итогового ответа. Ниже рассмотрим их подробнее.

* + 1. Chunking

Этап **Chunking** (разбиение текста на фрагменты) представляет собой начальную фазу подготовки корпуса для поиска. Исходные документы (например, PDF-файлы научных публикаций) разбиваются на небольшие логически связанные части — **чанки**. Основная задача — получить фрагменты, которые одновременно:

1. содержат достаточное количество информации для понимания контекста
2. не являются слишком длинными, чтобы их можно было эффективно обрабатывать языковой моделью
3. сохраняют логическую и смысловую связность

Существует два основных подхода к чанкингу:

1. **Эвристический подход**, основанный на структурных признаках текста. В качестве границ чанков используются элементы форматирования: абзацы, заголовки, списки, таблицы, а также знаки препинания. Такой подход прост в реализации и даёт хорошие результаты при наличии чётко структурированных документов.
2. Семантический подход, ориентированный на сохранение смысловой целостности фрагментов. Здесь применяются методы на основе векторных представлений текста: текст сначала представляется в виде последовательности эмбеддингов, после чего разбивается так, чтобы внутри чанков максимизировалась семантическая связность, а между ними — минимизировалась.

Выбор подхода зависит от специфики задач и характеристик документов: для технических текстов зачастую достаточно эвристик, в то время как для художественных или слабоформализованных текстов предпочтительнее семантический метод.

* + 1. Retrieval

После подготовки корпуса следующим этапом является Retrieval — поиск релевантных чанков по пользовательскому запросу. Цель этапа — выбрать множество доступных фрагментов до компактного и информативного набора, который будет использован генеративной моделью.

Существуют два основных класса retrieval-подходов:

1. Традиционные методы информационного поиска, такие как TF-IDF и BM25. Они работают с "мешками слов", определяя важность слов и их частотность. Эти методы быстры и интерпретируемы, но ограничены в способности учитывать семантику.
2. Методы на основе эмбеддингов. Здесь как запрос, так и текстовые фрагменты кодируются в векторном пространстве с помощью нейросетевых моделей (например, SBERT, BGE, GTE). Сходство между векторами измеряется с помощью косинусного расстояния или других метрик. Такой подход позволяет учитывать синонимы, контекст и грамматические зависимости.

Для повышения качества retrieval-а также применяются гибридные схемы, объединяющие преимущества обоих подходов. Например, можно сначала отобрать топ-100 кандидатов с помощью BM25, а затем провести их переоценку с использованием векторных моделей или LLM-based reranker'ов.

1. ПРОДВИНУТЫЕ ПОДХОДЫ

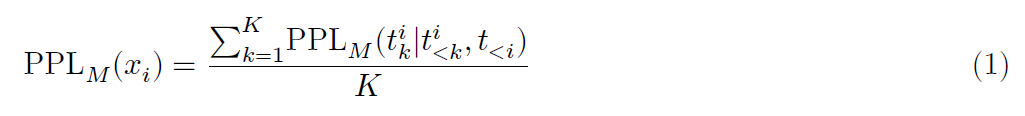
С момента появления Retrieval-Augmented Generation (RAG) было предложено множество подходов, направленных на повышение его эффективности. Эти улучшения затрагивают практически все стадии пайплайна — от предварительной обработки данных до ранжирования результатов. В данной главе рассмотрены наиболее значимые усовершенствования, сгруппированные по соответствующим этапам.

* 1. Chunking

Чанкинг — ключевой этап подготовки корпуса, на котором текст разбивается на отдельные фрагменты (чанки), пригодные для индексации и поиска. Главная задача на этом этапе — добиться оптимального баланса между объемом фрагмента и сохранением его смысловой целостности. Это особенно важно в доменно-специфичных системах, где плотность информации может значительно варьироваться.

Ключевые критерии качества чанков:

1. Семантическая целостность. Чанк должен представлять собой логически завершённую мысль, не теряющую смысла вне контекста. Нарушение границ предложений или смысловых блоков может привести к потере критической информации. Оптимально, если чанк охватывает не отдельные предложения, а их логически связанные группы.
2. Полнота информации. Каждый чанк должен содержать достаточное количество информации для ответа на типичный вопрос, относящийся к затронутой теме.
   * 1. Семантический Chunking
3. **Similarity Chunking. Данный метод использует эмбеддинг-модели для группировки связанных по смыслу предложений. Основан на косинусной близости между эмбеддингами предложений. Чанки разделяются по пороговому значению косинусной близости между соседними предложениями.**
4. Perplexity Chunking.На начальном этапе текст разделяется на набор предложений (x1, x2, ..., xn). Цель данного метода - сформировать набор чанков (X1, X2, ..., Xk), где каждый чанк представляет собой логически связанное объединение исходных предложений. Для объединения исходных предложений в чанки модель вычисляет перплексию (PPL) для каждого предложения на основе предшествующих предложений:



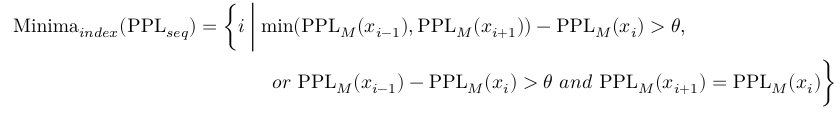
(1)

где K — общее количество токенов в ​, ​ — k-й токен в ​, а обозначает все токены, предшествующие ​. Для определения границ чанков алгоритм анализирует последовательность:



(2)

В поисках минимальных значений:



(3)

Смысл данной формулы заключается в следующем: если значения PPL по обе стороны точки выше, чем в самой точке, и разница хотя бы с одной стороны превышает заданный порог θ; либо разница между левой точкой и текущей больше θ, а значение справа равно значению в текущей точке. Эти минимумы рассматриваются как потенциальные границы фрагментов.

* 1. Query Rewriting

На этапе **переформулирования запроса** цель заключается в адаптации пользовательского запроса к структуре и стилю данных, хранящихся в индексе. Это особенно важно, если пользователь использует общие или неоднозначные формулировки, которые не соответствуют формулировкам в целевом корпусе.

* + 1. HyDE

Метод HyDE предполагает генерацию гипотетического документа на основе запроса. Языковая модель получает инструкцию в духе: “Напишите документ, который мог бы быть ответом на данный вопрос”. Созданный документ не обязательно достоверен, и может содержать фактические ошибки, так как его задача лишь имитировать релевантный текст. Ожидается, что в результате кодирования документа с помощью эмбеддинг-модели удалятся лишние (вымышленные) детали и в результате полученный вектор станет семантически ближе к релевантным фрагментам, чем вектор оригинального запроса пользователя.

* 1. Reranking (Two-Stage Retrieval)

После начального извлечения релевантных документов (на этапе Retrieval), часто применяются **двухступенчатые схемы**, где второй этап — **переранжирование** (Reranking) — уточняет порядок релевантности результатов. Рассмотрим два наиболее расрпостраненных подхода

1. Bi-Encoder. Запрос и каждый документ обрабатываются раздельно с помощью двух энкодеров. На выходе получаются векторы, между которыми вычисляется косинусное сходство. Это позволяет эффективно сравнивать множество документов с запросом, но иногда теряет контекстные связи между ними. На Рисунке 2 показано, как тексты преобразуются в векторы, объединяются в итоговые вектора (Pooling), а затем вычисляется их сходство.

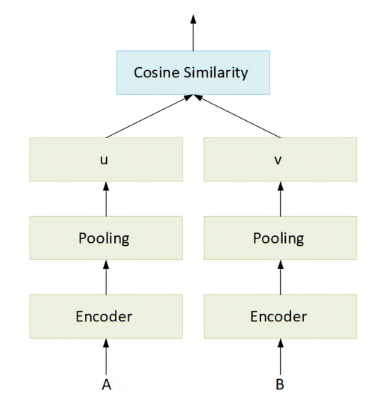


Рисунок 2 — Bi-Encoder

1. Cross-Encoder. В данном подходе запрос и документ объединяются в одну строку и подаются на вход языковой модели. Модель рассматривает их взаимодействие и возвращает итоговую оценку релевантности (similarity score) напрямую. Этот метод более ресурсоёмкий, но значительно точнее. На Рисунке 3 изображена последовательность преобразований. Similarity Score – выходной слой (например, полносвязный слой с сигмоидной функцией активации), который выдает оценку сходства – число в диапазоне от 0 до 1.

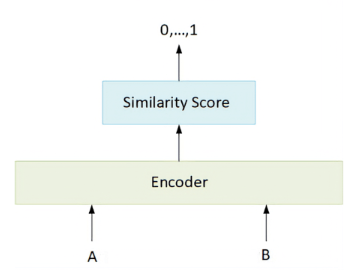


Рисунок 3 — Cross-Encoder

1. ТОЧКИ ОТКАЗА СИСТЕМ С RETRIEVAL AUGMENTED GENERATION

Системы Retrieval-Augmented Generation (RAG) обладают значительным потенциалом в задачах генерации ответов с опорой на внешние источники знаний. Однако при проектировании и эксплуатации подобных систем возникает ряд характерных точек отказа, способных существенно повлиять на качество выдачи. В статье “Seven Failure Points When Engineering a Retrieval Augmented Generation System” выделяются семь наиболее распространённых проблем, с которыми сталкиваются разработчики:

1. Отсутствие необходимого контента. Даже при корректной работе всех компонентов системы RAG может не удаваться извлечь релевантную информацию, если она отсутствует в базе знаний. Это может быть обусловлено неполнотой корпуса документов или ограничениями в процессах индексации.
2. Пропуск высокоранжированных результатов. Алгоритмы извлечения или ранжирования могут не отобрать релевантные фрагменты, даже если они присутствуют в индексе. Такое поведение часто связано с недостаточной точностью эмбеддинг-моделей, плохо подобранными метриками близости или ошибками в конфигурации поиска.
3. Неправильный контент. Извлечённые документы могут не соответствовать исходному запросу. Это приводит к генерации некорректных или вводящих в заблуждение ответов, особенно в случаях, когда модель генерации некритично воспринимает входные данные.
4. Неподходящий формат: Информация может быть представлена в формате, который затрудняет ее обработку или интеграцию в ответ, например, в виде изображений или таблиц.
5. Неправильная степень специфичности: Ответы, формируемые моделью, могут быть либо чрезмерно обобщёнными, либо излишне детализированными. Это снижает их полезность и не соответствует ожиданиям пользователя.
6. Неполные ответы: Система может предоставлять ответы, содержащие лишь часть необходимой информации, даже если полный ответ доступен в предоставленных документах.
7. Необходимость постоянной калибровки: Системы RAG подвержены деградации качества со временем и требуют регулярной переоценки ключевых компонентов — от индексируемого корпуса до настроек моделей и алгоритмов reranking. Особенно это критично при изменении пользовательских предпочтений или обновлении источников данных.
8. ОЦЕНКА СИСТЕМ С RETRIEVAL AUGMENTED GENERATION

Как было показано в предыдущей главе, при проектировании и внедрении систем с Retrieval-Augmented Generation (RAG) возникает множество потенциальных точек отказа. Для своевременного выявления этих уязвимостей и повышения качества системы необходимы надежные методы оценки. Поскольку RAG представляет собой комплексную архитектуру, тесно связанную с языковыми моделями, источниками знаний и пользовательскими сценариями, выбор и реализация подходящих стратегий оценки становятся критически важными

В обзорной статье “Evaluation of Retrieval-Augmented Generation: A Survey” выделяются три ключевых подхода к оценке RAG-систем:

1. **Оценка человеком** — субъективная, но точная форма оценки, применимая для оценки релевантности, достоверности и понятности ответов.
2. **Фреймворки для автоматической оценки** — обеспечивают воспроизводимость, масштабируемость и детализацию анализа.
3. **Использование специализированных датасетов**.

Кроме того, в статье представлена сводная таблица фреймворков и инструментов для оценки RAG-систем. Помимо RAGAS и MultiHop-RAG, о которых пойдет речь дальше, можно выделить датасет QASPER, как решение, подходящее по области вопросов.

* 1. QASPER

**QASPER** — это специализированный датасет, разработанный для задач вопросно-ответного поиска по научным публикациям в области обработки естественного языка. Он включает **5049** вопросов, сформулированных по **1585** NLP-статьям. Вопросы создавались экспертами, опираясь только на заголовок и аннотацию статьи, а затем на них отвечала другая группа исследователей, используя полный текст.

Особенность данного датасета состоит в том, что для ответа на вопросы используется только один документ. Несмотря на ценность QASPER, для RAG-систем, работающих с несколькими источниками, требуется расширение или использование других датасетов, ориентированных на **multi-hop** ответы.

* 1. RAGAS

RAGAS [6] — это фреймворк с открытым исходным кодом, разработанный специально для оценки качества работы RAG-систем. Он предоставляет как средства для **генерации тестовых запросов**, так и **набор метрик** для поэтапной оценки компонентов системы. Метрики для отдельных частей RAG позволяют декомпозировать оценку пайплайна на оценку его составляющих. С помощью таких метрик проще отслеживать уязвимые места системы, а также оценивать реакцию на изменение отдельных компонентов.

Методология RAGAS строится на подходе **LLM-as-a-Judge**, то есть большая языковая модель используется для генерации тестов и/или оценки ответов. Это позволяет автоматизировать процесс тестирования, хотя и создает определенные риски в задачах, требующих локализации.

Для задач на русском языке существует расширение фреймворка RAGAS, разработанное в России — GigaRAGAS. Особенность расширения состоит в том, что промпты для метрик и генерации датасета переведены и адаптированы под оценку RAG на русском языке.

* 1. Основные метрики

Для детальной оценки качества ответов в RAGAS (и других системах) применяются следующие метрики:

* + 1. Faithfulness

Метрика достоверности (Faithfulness [3]) измеряет, насколько фактически точен ответ относительно извлеченного контекста. Значения вариьируются от 0 до 1, где 1 означает, что утверждения в ответе подтверждаются контекстом.

Алгоритм расчета метрики:

1. Разбиение ответа на утверждения.
2. Каждое утверждение проходит проверку на релевантность к извлеченному контексту.
3. Вычисление показателя достоверности по формуле:

(1)

* + 1. Response Relevancy

Метрика релевантности ответа (Response Relevancy [3])оценивает, насколько ответ соответствует пользовательскому запросу. Высокий балл означает, что ответ полноценно отвечает на запрос, без избыточной или нерелевантной информации.

Алгоритм расчета метрики:

1. Генерация 3 искусственных вопросов на основе ответа системы.
2. Вычисление косинусного сходства между векторными представлениями запроса и каждого из вопросов.
3. Усреднение полученных значений.
   * 1. Answer Correctness

Метрика корректности ответа (Answer Correctness [3]) оценивает соответствие ответа эталонному. Балл варьируется от 0 до 1, где 1 означает полное совпадение.

Алгоритм расчета метрики:

1. Оценка фактической корректности: измеряет степень совпадения фактов между сгенерированным ответом и эталонным. Рассчитывается, как f1-score:

(2)

где TP – количество утверждений, присутствующих и в эталонном, и в сгенерированном ответах. FP – количество утверждений, которые есть в сгенерированном ответе, но отсутствуют в эталонном. FN – количество утверждений, которые есть в эталонном ответе, но отсутствуют в сгенерированном.

1. Оценка семантического сходства: рассчитывается на основе косинусной близости эмбеддингов сгенерированного и эталонного ответов.
2. Ответ: рассчитывается, как взвешенное среднее между фактическим соответствием и семантическим сходством
3. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

На этапе проектирования системы были активно использованы средства визуального моделирования, в частности UML-диаграммы (см. Приложение Б). Их применение позволило формализовать и структурировать требования к системе, а также визуализировать архитектуру системы. Это значительно упростило процесс анализа функциональности, выделения ключевых компонентов и их взаимодействий.

Дополнительно был разработан прототип пользовательского интерфейса, который стал важным инструментом в определении логики взаимодействия пользователя с системой. Благодаря прототипу удалось на раннем этапе выявить потенциальные узкие места в UX-дизайне и сформировать представление о сценариях использования системы.

* 1. Дизайн-документ

В качестве основы архитектурного проектирования использовался документ ML System Design, предложенный командой **Reliable ML** [1]. Он был адаптирован под специфику создаваемого приложения и стал основным ориентиром при проектировании клиент-серверной архитектуры. Документ помог систематизировать требования к масштабируемости, отказоустойчивости и безопасности, а также организовать структуру компонентов.

* 1. Постановка задачи

Целью разработки являлось создание прототипа системы с минимально необходимым функционалом — Minimum Viable Product (MVP), который в дальнейшем мог бы быть масштабирован и дополнен новыми функциями.

Разработка системы была разделена на несколько этапов:

1. Рразработка серверной части. Реализация API с поддержкой авторизации, загрузки пользовательских файлов и построения индексации. В этот же этап входила интеграция RAG-модуля для обработки пользовательских запросов.
2. Реализация клиентского приложения. Разработка интерфейса на основе решений ChatPDF и ChatUI с поддержкой диалога с моделью и возможностью загрузки PDF-документов.
3. **Интеграция и тестирование.** Проверка корректности взаимодействия между клиентом и сервером, функциональное и нагрузочное тестирование.
4. **Пилотный запуск.** Развертывание системы на ограниченной пользовательской выборке, сбор обратной связи, выявление точек роста и недоработок.
   1. Архитектура серверной части

Основные задачи — реализация методов для загрузки документов, авторизации и регистрации пользователей. На Рисунке 2 представлена схема взаимодействия компонентов сервера.

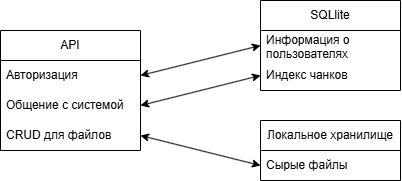


Рисунок 2 - Устройство пайплайна Retrieval Augmented Generation

1. Хранение информации о пользователях. Для хранения информации о пользователях я решил использовать SQLite с библиотекой SQLAlchemy. Такой выбор обусловлен необходимостью быстрого прототипирования и масштабируемости системы в будущем. SQLite идеально подходит для проекта на начальной стадии, так как не требует развертывания отдельного сервиса. Использование SQLAlchemy позволяет эффективно взаимодействовать с базой данных и упрощает расширение системы при необходимости. В будущем можно будет перейти на более сложное решение, если нагрузка на базу данных увеличится.
2. Хранение файлов. На этапе пилотной версии было принято решение использовать локальное файловое хранилище. Каждый докуменгт сохраняется в виде обычного файла на сервере. Такое решение позволяет сэкономить время на интеграцию с облачными хранилищами (например, S3), и подходит для пилотной версии с ограниченной нагрузкой.
3. Хранение индексированных чанков. Для реализации полнотекстового поиска по фрагментам текста я решил использовать модуль FTS5 для SQLite. Это решение подходит для требований проекта и позволяет реализовать полнотекстовый поиск без необходимости интегрировать более сложные системы на начальной стадии разработки. В будущем, если потребности в поиске будут увеличиваться, можно рассмотреть переход на более мощные решения.
   1. Клиент

В качестве основы для реализации клиентской части я решил использовать ChatUI [ссылка]. Он представляет собой современный чат-интерфейс для общения с LLM. Для поддержки просмотра PDF-файлов, необходимо встроить в приложение компонент для просмотра. Кроме того, необходима реализация клиентской логики для загрузки файлов.

* 1. Пилотный запуск

Пилотный запуск направлен на апробацию системы в условиях, приближенных к реальному использованию. Основной целью пилота является сбор обратной связи от пользователей, позволяющей выявить недочеты, узкие места в логике работы, а также сформулировать направления для дальнейшего развития продукта.

* 1. Требования к работе системы

С учетом ограниченного числа пользователей пилотной версии (до 10 человек), системные требования сформулированы с учетом запаса прочности:

Пропускная способность. Исходя из предположения об 1 запросе на пользователя каждые 15 секунд, расчетная нагрузка составит:

(1)

где RPS – количество запросов за секунду. Для обеспечения устойчивой работы системы установлена целевая пропускная способность в **1.5 RPS**. Это позволит учесть различные сценарии, такие как задержки в сети или повышение активности пользователей.

Задержка ответа. Ожидаемое время отклика API — не более **1 секунды**, что обеспечит комфортную работу с системой в режиме реального времени.

Вычислительные ресурсы:

GPU: для запуска эмбеддинг-модели deepvk/USER-bge-m3 (359 млн параметров) требуется ~1.44 ГБ VRAM. При пакетной обработке 32 фрагментов — до 4 ГБ VRAM.

CPU: для системы на 10 пользователей будет достаточно 2 виртуальных ядра на запрос, таким образом, минимальные требования для системы, с запасом: 4 ядра CPU.

RAM: достаточно 2–4 ГБ для хранения данных и работы API.

Дисковое пространство: для хранения файлов пользователей будет достаточно дискового пространства в 200мб на каждого пользователя. Кроме хранения загруженных файлов необходимо хранить еще и индексированные чанки для каждого документа, а также базу данных для информации о пользователях и другие вспомогательные файлы. Таким образом, для поддержания системы из 10 пользователей понадобится:

(2)

* + 1. Механизмы безопасности

Для обеспечения безопасности доступа к системе в рамках пилотного проекта используется метод предопределенного ключа API, который ограничивает доступ только для заранее определенной группы пользователей. Этот ключ передается в заголовке каждого запроса, и позволяет убедиться, что доступ к системе получают только авторизованные лица.

* 1. UML-диаграммы
     1. Диаграмма вариантов использования (Use Case)

Позволила выделить ключевые сценарии взаимодействия двух типов пользователей: **Пользователя** (задаёт вопросы, читает ответы, просматривает документы) и **Администратора** (кроме базового функционала, обладает правами для управления системой) (см. Рисунок 3).

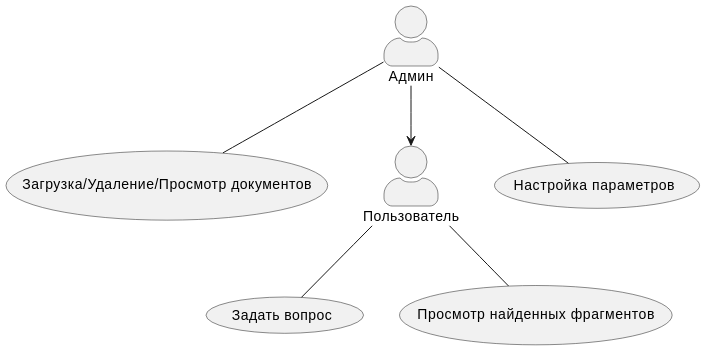


Рисунок 3 – Диаграмма вариантов использования

* + 1. Диаграмма последовательностей (Main Sequence)

Следующим шагом была разработка диаграммы последовательностей, которая иллюстрирует, как объекты системы взаимодействуют друг с другом для выполнения основного сценария работы. Эта диаграмма показывает последовательность шагов, включая отправку запроса, обработку его на сервере и возврат результатов пользователю. На основе диаграммы можно проследить, как данные перемещаются через систему и как различные компоненты обмениваются сообщениями. Этот этап помог уточнить логику взаимодействий между элементами сервиса.



* + 1. Диаграмма компонентов сервиса

Для проектирования архитектуры системы была создана диаграмма компонентов. Она представляет ключевые модули, такие как **RAG-модули**, **базы данных**, **API** и их взаимодействие. Эта диаграмма стала важным инструментом для понимания структуры системы, позволяя выделить основные компоненты и определить их связи. Кроме того, диаграмма помогла выявить протоколы взаимодействия компонентов, а также их зависимости друг от друга, что существенно облегчило разработку системы.



* + 1. Дополнение: Диаграмма активностей (User Activity)

В дополнение к созданным диаграммам также была составлена диаграмма активностей. Это анализ активностей пользователей, который помог понять, как именно пользователи взаимодействуют с системой в рамках различных сценариев.



1. РЕАЛИЗАЦИЯ ПРОТОТИПА

Завершив этап проектирования, в рамках которого была детально сформулирована архитектура системы, выбраны технологии и определены ключевые модули, логичным продолжением стало непосредственное воплощение этих проектных решений в виде программного прототипа. В данной главе приводится описание реализованной системы, охватывающее как серверную, так и клиентскую часть, структуру хранения данных, а также модуль Retrieval-Augmented Generation (RAG), играющий ключевую роль в обеспечении функциональности интеллектуального поиска и генерации ответов.

Реализация была выполнена с учетом предварительно сформированных требований и проектных ограничений, описанных в предыдущей главе. Основной упор был сделан на модульность, масштабируемость и асинхронность, что обеспечило высокую гибкость и производительность разрабатываемого программного обеспечения.

* 1. Общая структура системы

Разработанная система состоит из двух основных частей: клиентского интерфейса и серверного приложения. Между ними осуществляется обмен данными по протоколу HTTP, а также через потоковую передачу данных посредством Server-Sent Events (SSE). Центральным элементом архитектуры является модуль RAG, обеспечивающий возможность интеллектуального поиска и генерации ответов на естественном языке на основе пользовательских документов.

* 1. Серверная часть

Серверная часть реализована на языке Python с использованием асинхронного фреймворка FastAPI. Для работы с базой данных используется библиотека SQLAlchemy. Основные функции серверной части включают:

1. Обработку HTTP- и SSE-запросов от клиента.
2. Взаимодействие с RAG-системой для генерации ответов.
3. Аутентификация и авторизация пользователей на основе JWT-токенов.
4. Асинхронный доступ к хранимым файлам и метаданным.
5. Поддержка многопользовательского режима.

Для реализации потоковой генерации ответов используется однонаправленный протокол Server-Sent Events (SSE). Благодаря асинхронной архитектуре, сервер способен эффективно обрабатывать множество одновременных клиентов.

* 1. Итоговая архитектура

Для формализации архитектуры системы была выбрана C4-нотация. Она предлагает гибкий набор инструментов для проектирования программных систем, и включает в себя несколько уровней детализации:

1. Контекст (Context) – показывает внешние системы и пользователей.
2. Контейнеры (Containers) – архитектура приложения без глубокого погружения в техническую часть. Отображает основные логические блоки и используемые технологии.
3. Компоненты (Components) – раскрывает архитектуру отдельных контейнеров.
4. Code – Код – самый низкий уровень абстракции, чтобы показать классы и их связи.

Для проекта были использованы второй и третий уровни – “Контейнеры” и “Компоненты”, они позволили описать систему в достаточной детализации без избыточных подробностей о кодовой базе проекта.

На Рисунке 1 представлена диаграмма контейнеров, показывающая общее взаимодействие клиента, сервера, базы данных и RAG-пайплайна.



Рисунок 1 – диаграмма C4 (уровень контейнеров).

Следующий уровень детализации – компоненты серверной части (Рисунок 2). На нем я выделил ключевые модули:

1. API-сервис – точка входа, обрабатывает запросы от клиента, авторизует пользователей, управляет загрузкой файлов, и инициирует взаимодействие с RAG-пайплайном.
2. Сервис индексации RAG – обрабатывает загруженные документы, извлекает текст, и разделяет его на фрагменты для последующего поиска.
3. Сервис генерации RAG – принимает пользовательский запрос, извлекает релевантные фрагменты, генерирует финальный ответ с помощью большой языковой модели.



Рисунок 2 – диаграмма C4 третьего уровня (Components)

Наконец, я визуализировал более детализированную схему взаимодействия внутри RAG-пайплайна, куда включил этапы индексации и генерации вместе. (Рисунок 3).



Рисунок 3 - диаграмма C4 (уровень компонентов, RAG-пайплайн)

* 1. Клиентская часть

Клиент (Рисунок 4) реализован в виде одностраничного веб-приложения (SPA) с использованием библиотеки React. Основной функционал включает:

1. Интерактивный чат с возможностью ввода и отображения истории сообщений.
2. Поддержка потоковых ответов от сервера.
3. Регистрация и авторизация пользователей.
4. Загрузка и просмотр PDF-документов.
5. Навигация по загруженным файлам.



Рисунок 4 – Пользовательский интерфейс веб-приложения

* 1. Хранение данных

В качестве основной СУБД используется PostgreSQL. В базе хранятся:

1. Информация о пользователях
2. История запросов и ответов
3. Метаданные, связанные с загруженными файлами
4. Чанки и их векторные представления

Сами PDF-файлы сохраняются в локальном хранилище. Доступ к ним происходит асинхронно по запросу клиента.

* 1. Модуль Retrieval-Augmented-Generation

Модуль RAG – ключевая часть архитектуры моего приложения. Его работа включает два этапа: Индексация и Генерация (Рисунок 3). Первоначально я рассматривал классического подход с использованием поиска на основе BM25. Но несмотря на его простоту и эффективность в ряде задач, он имеет ограничение, связанное с работой с мультиязычными данными. В моей системе как запросы, так и чанки в базе данных могут быть представлены как на русском, так и на английском языках. BM25 оперирует на уровне отдельных слов, а значит не способен учитывать семантику.

Вместо этого я реализовал семантический поиск с использованием векторных представлений как для запросов, так и для чанков из базы данных. Для вычисления схожести используется косинусная мера. Это позволяет системе находить релевантные фрагменты даже с учетом различия в формулировках и языках.

1. ТЕСТИРОВАНИЕ

Для проверки корректности и надёжности функционала были разработаны модульные и интеграционные тесты с использованием библиотеки Pytest и асинхронного клиента из библиотеки HTTPX. Тестами покрыты основные компоненты:

1. Регистрация и авторизация пользователей.
2. Хранение и получение истории сообщений.
3. Работа с SSE, и получение потоковых ответов от системы.
4. Загрузка и парсинг PDF-файлов.
5. Интеграция с RAG-системой и корректная обработка запросов.

Тестирование проводится автоматически, что позволяет оперативно выявлять ошибки при изменении кода и вносить улучшения без снижения стабильности системы.

* 1. Датасет

Первоначально я рассматривал датасет QASPER для тестирования системы, однако в процессе текущей практики я выявил его ограничения: вопросы в датасете слишком общие, к тому же каждый вопрос связан только с одной статьей. Эти ограничения делают QASPER малопригодным для тестирования сложных retrieval-механизмов.

* + 1. FRAMES

В итоге я выбрал датасет FRAMES (Factuality, Retrieval, And reasoning MEasurement Set) [4]. Основные характеристики датасета:

1. 824 сложных вопросов, требующих информации из 2-15 статей из Wikipedia.
2. Широкий спект тем: история, спорт, наука, здоровье, животные и др.
3. Для каждого вопроса представлен эталонный ответ, а также список релевантных статей из Wikipedia.

Датасет был разработан и выпущен 24 января 2025 года исследователями из Google с целью создания стандартизированного набора задач для оценки работы Retrieval-Augmented-Generation систем. В статье [4] авторы демонстрируют, что даже передовые большие языковые модели, такие как Gemini-Pro-1.5, сталкиваются со значительными трудностями при ответах на вопросы из FRAMES. Это подтверждает, что добавление поискового механизма может существенно улучшить качество ответов.

* + 1. Подготовка датасета

Для тестирования я случайным образом отобрал из датасета 50 вопросов, и проиндексировал их статьи. Чтобы приблизить эксперимент к реальным условиям, в индекс также были добавлены нерелевантные статьи, содержащие информацию по другим вопросам. Этот шаг позволяет проверить, насколько эффективно система извлекает нужные данные и игнорирует лишнюю информацию. Итоговая выборка включает 330 статей и 50 вопросов.

* 1. Конфигурации

В данном разделе рассматриваются три конфигурации, использованные для оценки качества: стандартная Retrieval-Augmented Generation (RAG), версия Agentic RAG и LLM без поиска.

Во всех конфигурациях использовалась большая языковая модель openai/gpt-4o-mini [5] и эмбеддинг-модель deepvk/USER-bge-m3 [6].

* + 1. RAG

Стандартная конфигурация RAG включает два основных этапа:

1. Индексация. На данном этапе документы преобразуются в чанки с помощью алгоритма семантического деления [7]. Далее чанки индексируются с помощью алгоритма Okapi bm25 [8].
2. Инференс. На этапе инференса в систему приходит запрос пользователя, для которого необходимо найти релевантные чанки. Чанки извлекаются в два этапа [здесь бы еще про two-shot-retrieval]: сперва 10 чанков находятся с помощью алгоритма Okapi bm25 [8], далее среди отобранных чанков выбираются лучшие с помощью биэнкодер-ранжирования. Затем чанки подаются на вход генеративной модели для формирования финального ответа пользователю.
   * 1. Agentic RAG

Agentic RAG — это усовершенствованный подход к построению RAG. В отличие от традиционного RAG, где модель генерирует ответы на основе единственного шага извлечения данных, Agentic RAG позволяет большой языковой модели самостоятельно формулировать запросы, критически оценивать полученные результаты и при необходимости повторно обращаться к источникам для уточнения информации. Использует ту же систему индексации и поиска по базе знаний, что и стандартный RAG.

* + 1. LLM

Дополнительно была протестирована модель **LLM без поиска**, где генерация ответа происходит исключительно на основе входного вопроса, без извлечения контекста из базы данных.

* 1. Анализ результатов

Стоит отметить, что большие языковые модели склонны к более лояльной оценке своих текстов [9], поэтому для метрик LLM-as-Judge я использовал другую модель: google/gemini-pro [5].

* + 1. Сравнение метрик на разных конфигурациях

|  |  |  |  |
| --- | --- | --- | --- |
|  | RAG | Agentic-RAG | LLM |
| Faithfulness | 0.54 | 0.08 | 0.25 |
| Response Relevancy | 0.29 | 0.6 | - |
| Answer Correctness | 0.33 | 0.49 | 0.53 |
| BERTScore | 0.8 | 0.82 | 0.81 |
| Среднее время работы (с) | 6.5 | 29 | 2.37 |

* + 1. Выводы по метрикам

1. RAG требует доработки алгоритма чанкинга для улучшения качества поиска и общего качества извлекаемых чанков. Нужно исследовать эвристические подходы к формированию чанков и сравнить с текущим семантическим подходом.

Agentic-RAG показывает лучшие результаты по извлечению фрагментов благодаря множественным вызовам поиска по базе знаний. Однако необходимо улучшить формулировку промежуточных запросов для повышения их релевантности и точности, также переработать запрос для формирования финального ответа.

* 1. Пилотное тестирование

В рамках разработки проекта я также провел небольшое пилотное тестирование. Для этого я развернул приложение на сервере и открыл доступ к приложению через Интернет. Целью тестирования было получение обратной связи о работоспособности интерфейса, корректности генерации ответов и общего впечатления пользователей о работе приложения. Кроме того, было важно узнать, насколько потенциально полезным могло бы быть такое приложение в учебном процессе.

Я сформировал 6 вопросов и внес их в сервис Google Forms для автоматического сбора ответов пользователей. В результате пользователи получили в свободное использование приложение и анкеты для заполнения (см. Приложение 1).

В результате опроса я пришел к слудующим выводам:

1. Система корректно обрабатывала запросы, но в некоторых ситуациях давала слишком общие ответы.
2. Пользователи положительно оценили концепт дизайна интерфейса. Особо отметили возможность просмотра PDF-файлов прямо в приложении.

Результаты тестирования были мной учтены при доработке приложения.

ЗАКЛЮЧЕНИЕ

Разработано полнофункциональное клиент-серверное приложение, демонстрирующее принципы современной веб-разработки, стриминга с использованием LLM и взаимодействия с RAG-системами. Архитектура масштабируема и может быть расширена дополнительными модулями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // arXiv preprint arXiv:2005.11401. — 2020.

Seven Failure Points When Engineering a Retrieval Augmented Generation System // arXiv preprint arXiv:2401.05856. — 2024.

Query Rewriting for Retrieval-Augmented Large Language Models // arXiv preprint arXiv:2305.14283. — 2023.

Evaluation of Retrieval-Augmented Generation: A Survey // arXiv preprint arXiv:2405.07437. — 2024.

Re2G: Retrieve, Rerank, Generate // arXiv preprint arXiv:2207.06300. — 2022.

RAGAS: Automated Evaluation of Retrieval Augmented Generation // arXiv preprint arXiv:2309.15217. — 2023.

Searching for Best Practices in Retrieval-Augmented Generation // arXiv preprint arXiv:2407.01219. — 2024.

RAG and RAU: A Survey on Retrieval-Augmented Language Model in Natural Language Processing // arXiv preprint arXiv:2404.19543. — 2024.

Corrective Retrieval Augmented Generation // arXiv preprint arXiv:2401.15884. — 2024.

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models // arXiv preprint arXiv:2201.11903. — 2022.

Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection // arXiv preprint arXiv:2310.11511. — 2023.

Levels Of Text Splitting [Электронный ресурс]. — 2024. — Режим доступа: https://github.com/FullStackRetrieval-com/RetrievalTutorials/blob/main/tutorials/LevelsOfTextSplitting/5\_Levels\_Of\_Text\_Splitting.ipynb (дата обращения: 09.12.2024).

RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval // arXiv preprint arXiv:2401.18059. — 2024

A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers // arXiv preprint arXiv:2105.03011. — 2021

ML System Design Doc – Reliable ML [Электронный ресурс]. — 2025. — Режим доступа: https://github.com/IrinaGoloshchapova/ml\_system\_design\_doc\_ru/tree/main (дата обращения: 15.03.2025).

HuggingFace [Электронный ресурс]. — 2025. — Режим доступа: https://huggingface.co/deepvk/USER-bge-m3 (дата обращения: 15.03.2025).

BERTScore: Evaluating Text Generation with BERT // arXiv preprint arXiv:1904.09675. — 2019.

A Survey on LLM-as-a-Judge // arXiv preprint arXiv:2411.15594. — 2024.

RAGAS: Automated Evaluation of Retrieval Augmented Generation // arXiv preprint arXiv:2309.15217. — 2023.

Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation // arXiv preprint arXiv:2409.12941. — 2024.

VseGpt – провайдер для больших языковых моделей [Электронный ресурс]. — 2025. — Режим доступа: https://vsegpt.ru/Docs/Models (дата обращения: 26.02.2025).

HuggingFace [Электронный ресурс]. — 2025. — Режим доступа: https://huggingface.co/deepvk/USER-bge-m3 (дата обращения: 26.02.2025).

5 Levels Of Text Splitting [Электронный ресурс]. — 2025. — Режим доступа: https://github.com/FullStackRetrieval-com/RetrievalTutorials/blob/main/tutorials/LevelsOfTextSplitting/5\_Levels\_Of\_Text\_Splitting.ipynb (дата обращения: 26.02.2025).

Robertson S.E., Walker S., Jones S., Hancock-Beaulieu M., Gatford M. Okapi at TREC-3 // Proceedings of the Third Text REtrieval Conference (TREC 1994). Gaithersburg, USA, November 1994.

Pride and Prejudice: LLM Amplifies Self-Bias in Self-Refinement // arXiv preprint arXiv:2402.11436. — 2024.

OpenAPI specification for the OpenAI API [Электронный ресурс]. — 2025. — Режим доступа: https://github.com/openai/openai-openapi (дата обращения: 26.02.2025).

Open source codebase powering the HuggingChat app [Электронный ресурс]. — 2025. — Режим доступа: https://github.com/huggingface/chat-ui (дата обращения: 26.02.2025).