Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики
Лаборатория машинного интеллекта

Выпускная квалификационная работа бакалавра

Эффективное дообучение больших языковых моделей в задаче выделения информации из контекста

Автор:
Студент 131 группы
Вицын Семён Сергеевич

Научный руководитель:
Алексей Владимирович Гончаров

Научные консультанты:
Глеб Моргачев
Алексей Меркулов
Валентин Шишков

Москва 2025

Эффективное дообучение больших языковых моделей в задаче выделения информации из контекста

Вицын Семён Сергеевич

Современные подходы к улучшению языковых моделей всё чаще основываются на Retrieval Augmented Generation (RAG), отходя от прямого дообучения под конкретные задачи. Однако, несмотря на универсальность метода, он сопряжён с рядом трудностей. Данная работа посвящена изучению этих проблем, а также методам адаптаций языковых моделей для задач RAG и способам оценки их эффективности.

Содержание

1 Введение

В последние годы большие языковые модели (LLM) достигли значительного прогресса. Такие архитектуры уже сейчас способны эффективно решать широкий круг задач. В число таких задач входят: перевод, обработка и суммаризация текстовых данных, автоматизация бизнес-процессов через использование чат-ботов, а также многое другое. Однако действительно высоких результатов удается достигать только на моделях, в которых количество параметров исчисляется десятками и сотнями миллиардов. В то же время их разработка и поддержка требуют огромных вычислительных ресурсов, а среди качественных проблем главными остаются фактологические ошибки моделей и их неспособность адаптироваться к динамически изменяющимся данным.

При создании систем, способных поддерживать актуальность знаний в определенной предметной области, на сегодняшний день чаще всего используют метод Retrieval Augmented Generation (RAG) [1]. Для этого подхода требуется формирование и поддержание документационных индексов, содержащих всю необходимую информацию по целевому домену. Наиболее частым сценарием применения подобных систем является Question-Answering (QA), когда задачей модели является генерация ответа на запрос пользователя с учетом документационной базы. Для этого в контекст генерации интегрируются релевантные фрагменты, извлечённые на основе пользовательского вопроса.

В классической постановке такую архитектуру можно разделить на 4 ключевых блока:

1. ***Система предобработки данных.***

   Компонент, который отвечает за обработку и разбиение исходных документов на более мелкие фрагменты (чанки). На этом этапе дополнительно может быть произведена очистка или обогащение метаданными (например, указание источника или номера главы).

2. ***Retriever.***

   Часть системы, выполняющая задачу поиска релевантных фрагментов документации на основе пользовательского запроса. Может быть реализована с помощью классических статистических подходов, таких TF-IDF или BM25. На практике чаще всего представляют собой нейросетевую модель, отображающую текстовые фрагменты в векторное

пространство высокой размерности для последующего поиска семантически близких фрагментов.

3. ***Системы векторного хранения и поиска.***

   Специализированное хранилище, оптимизированное для эффективного поиска по векторным представлениям. Оно индексирует чанки, преобразованные retriever'ом, и поддерживает операции поиска по метрике близости. Популярными решениями являются Qdrant (полноценная векторная база данных) и FAISS (библиотека для эффективного векторного поиска).

4. ***Генератор.***

   Языковая модель, в контекст генерации которой подается запрос пользователя и извлеченные ранее текстовые фрагменты. Обогащение контекста документацией позволяет LLM генерировать ответ с учётом актуальной информации.

Общая схема работы алгоритма изображена на рис. 1.

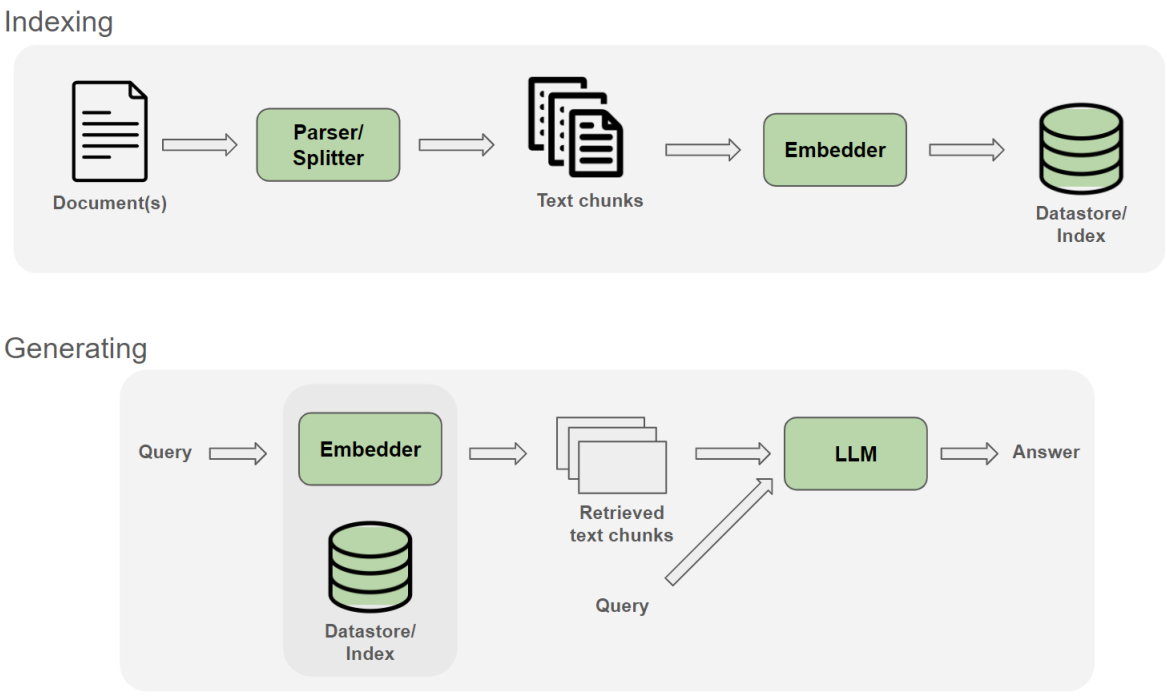


Рис. 1: Схема RAG pipeline.

Работа фокусируется на использовании языковых моделей в качестве интеллектуальных ассистентов для работы с обширными массивами документации. Многие современные компании активно разрабатывают и внедряют подобные системы в свою инфраструктуру.

Технология полезна как для внутренних сотрудников компаний благодаря оперативному доступу к релевантной информации, так и для улучшения взаимодействия внешних пользователей с продуктом. При формировании ответа ключевым фактором является его фактологичность и корректность.

Несмотря на подтверждённую эффективность и широкое распространение, архитектура RAG имеет ряд ограничений, связанных с качеством её генеративной компоненты. В научной литературе выделяются две ключевые проблемы:

1. ***Генерация ответов на основе ложного контекста.***

   Согласно исследованию «Large Language Models Can Be Easily Distracted by Irrelevant Context» [2], некорректный контекст в RAG может вызывать «галлюцинации» модели, основанные на ложной или нерелевантной информации. Возможны даже ситуации, когда LLM могла самостоятельно ответить на поставленный вопрос, однако при добавлении дополнительного контекста, ответ становился неверным (рис. 2).

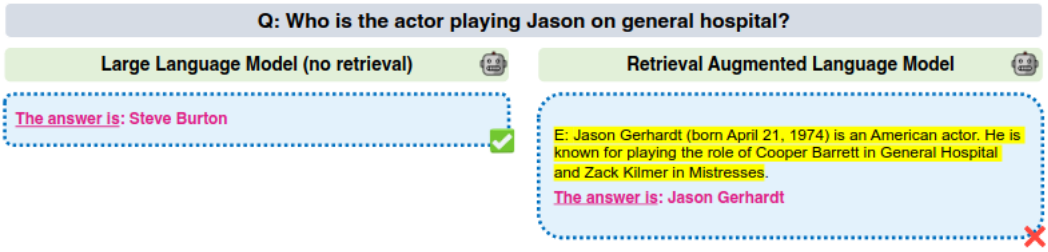


Рис. 2: Неверный ответ из-за нерелеватного контекста, взято из [3]

2. ***Зависимость от позиции релевантной информации.***

   При использовании RAG-систем для генерации ответов часто применяют модели, поддерживающие обработку расширенных контекстов. Данный подход позволяет потенциально передать больше релевантной информации. И хотя с технической точки зрения нет проблемы в реализации архитектур с большим или даже неограниченным контекстом, однако качество ответов моделей в таких сценариях значительно ухудшается.

Применительно к задаче извлечения информации эта проблема подробно описана в статье «Lost in the Middle: How Language Models Use Long Contexts» [4]. Результаты исследования показывают, что LLM по умолчанию плохо справляются с извлечением информации из большого объёма разрозненных документов, фокусируясь преимущественно на информации в начале и в конце (рис. 3). Это приводит к снижению качества генерации, вплоть до уровня ниже, чем при отсутствии вспомогательных документов.

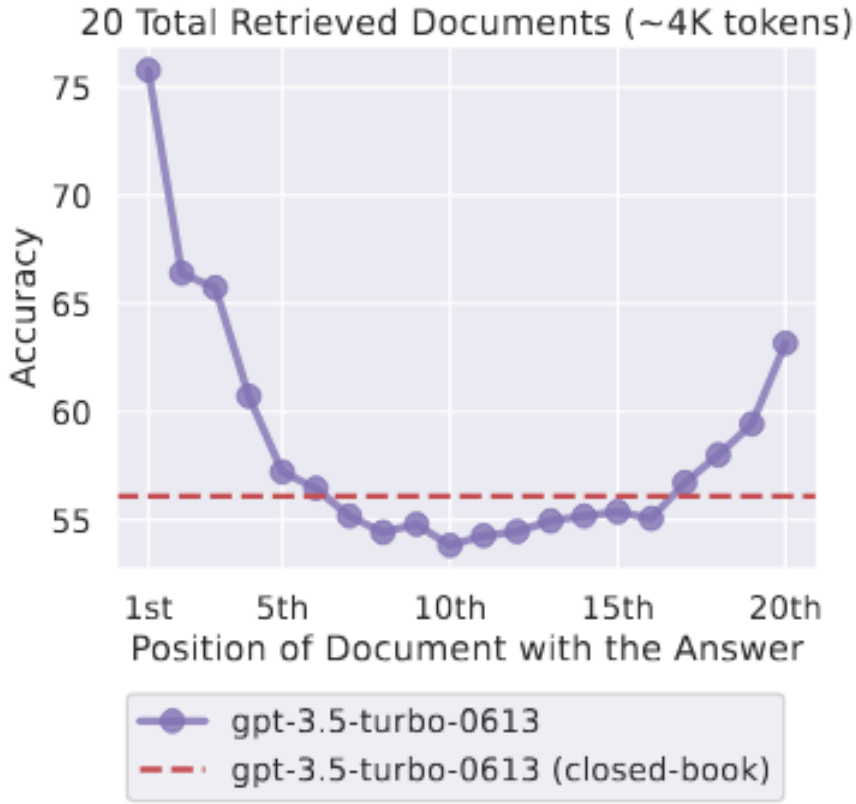


Рис. 3: Качество ответа в зависимости от положения релевантной информации в контексте, взято из [4].

Таким образом, возникает необходимость дообучения языковых моделей для повышения качества выделения информации из контекста. Для LLM критически важно не только извлекать полезные сведения из всего контекста генерации, но и иметь возможность находить и игнорировать шумовые или противоречивые фрагменты.

Важно отметить, что в рамках RAG-архитектур языковые модели выполняют функцию контекстуальных агрегаторов, синтезируя ответ на основе предварительно отобранной информации. На практике данную задачу, в отличие от классического QA, можно считать упрощенной, так как

модель фокусируется не на генерации, а на структурировании ответа, что снижает требования к её «пониманию» изложенной информации.

Данная особенность открывает возможность оптимизации вычислительных ресурсов через применение компактных архитектур (объёмом 1-2 млрд параметров) при условии их специализированной адаптации под RAG-сценарии. Сохранение производительности в таких условиях достигается за счёт узкой специализации модели и усовершенствованных методов работы с контекстом.

Цель данного исследования — анализ, разработка и комбинация эффективных подходов дообучения LLM в сочетании с методами оценки качества генерации для задач RAG в условиях ограниченных вычислительных ресурсов.

2 Предметная область

Введем определения, используемые в данной работе:

Таблица 1: Нотация

Термин	Обозначение	Описание
Корпус документов	$\mathcal{D} = \{d_i\}_{i=1}^{\lvert\mathcal{D}\rvert}$	Множество объектов, где d_i — фрагмент документации после этапа предобработки.
Retriever	f_ϕ, f_ψ	Отображения текстовых данных в $\mathbb{R}^m$ с параметрами ϕ и ψ. Отображение f_ϕ для пользовательских запросов, а f_ψ для документов.
Метрика близости	$Sim(q,d)$	Семантическая близость между запросом и документом.
Generator (LLM)	g_θ	Отображение, параметризованное θ.
Запрос	q	Пользовательский запрос.
Контекст	c	Контекст запроса.
Ответ	a	Целевой ответ.
QA датасет	$X = \{X_i\}_{i=1}^{\lvert X\rvert}$	Набор объектов, где $X_i = (q_i, c_i, y_i)$.

Обучением модели будем называть процесс построения оптимального отображения, минимизирующего целевую функцию потерь.

Обучаемыми параметрами модели - параметры отображения, которые могут изменяться в ходе процесса обучения модели для получения оптимального результата.

2.1 Большая языковая модель

С точки зрения формальной математической постановки большая языковая модель это отображение, моделирующее вероятностное распределение следующего токена (символа или части слова) в текста, т.е. $g_\theta : \mathcal{V}^* \mapsto [0,1]^{|\mathcal{V}|}$, где $\mathcal{V}$ фиксированный словарь.

Вероятность последовательности токенов $(x_1, \ldots, x_T)$ факторизуется авторегрессионно:

$$P_\theta(x_1, \ldots, x_T) = \prod_{t=1}^{T} P_\theta(x_t \mid x_{<t}),\ \textit{где}\ g_\theta(x_{\leq t}) = P_\theta(\cdot \mid x_{\leq t}) \tag{1}$$

Обучение производится на большом корпусе $D = \{(x_1^{(i)}, \ldots, x_{T_i}^{(i)}\}_{i=1}^{N}$ через вычисление оценки максимального правдоподобия, путем минимизации среднего отрицательного логарифма правдоподобия (NLL loss):

$$\mathcal{L}_{LM}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} logP_\theta(x_t^{(i)} \mid x_{<t}^{(i)}) \tag{2}$$

Этот этап обычно называют предварительным обучением (pretrain). На нём модель «получает» основные знания о мире и оптимизируется для правдоподобного продолжения текстов.

Следующим этапом является обучение на датасетах, составленных из инструкционных пар $D = \{I^{(i)}, R^{(i)}\}_{i=1}^{N}$, *где*

- $I^{(i)} = (l_1^{(i)}, \ldots, l_{L_i}^{(i)})$ - описание задачи на естественном языке.
- $R^{(i)} = (r_1^{(i)}, \ldots, r_{T_i}^{(i)})$ - ожидаемый ответ.

В данном случае задачей для LLM ставится генерация R. В частности функция потерь приобретает вид:

$$\mathcal{L}_{SFT}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} logP_\theta(r_t^{(i)} \mid l_1^{(i)}, \ldots, l_{L_i}^{(i)}, r_{<t}^{(i)}) \tag{3}$$

Этот этап называется Instruct Tuning или же Supervised Finetuning (SFT). Он необходим для адаптации модели к выполнению запросов, описанных на ествественном языке.

Частным случаем инструкционного обучения является question-answering на датасете X:

$$\mathcal{L}_{QA}(\theta) = -\frac{1}{|X|}\sum_{i=1}^{X}\sum_{t=1}^{T_i} logP_\theta(a_t^{(i)} \mid q^{(i)}, c^{(i)}, a_{<t}^{(i)}) \tag{4}$$

2.2 Retriever и генерация в RAG

Как упоминалось ранее, в архитектуре RAG за создание информативных векторных представлений отвечает отдельный Retriever модуль. Формально, он задается как отображение $f : \mathcal{T} \mapsto \mathbb{R}^m$, где $\mathcal{T}$ - пространство текстов. Такие модели также называют эмбеддерами(embedder), а создаваемые ими представления эмбеддингами (embedding), они играют важную роль во множества задач машинного обучения. Ключевым требованием для информативности является:

- $f(x) \approx f(y)$, если тексты x и y схожи по смыслу.
- $f(x) \not\approx f(y)$, если тексты x и y различны по смыслу.

Семантическую близость полученных векторов чаще всего вычисляется по формуле косинусной меры близости или скалярного произведения:

$$Similarity(q, d_i) = \langle f_\phi(q)\ ,\ f_\psi(d_i)\rangle = f_\phi(q)^\top f_\psi(d_i) \tag{5}$$

Принцип работы изображен на рис. 4.

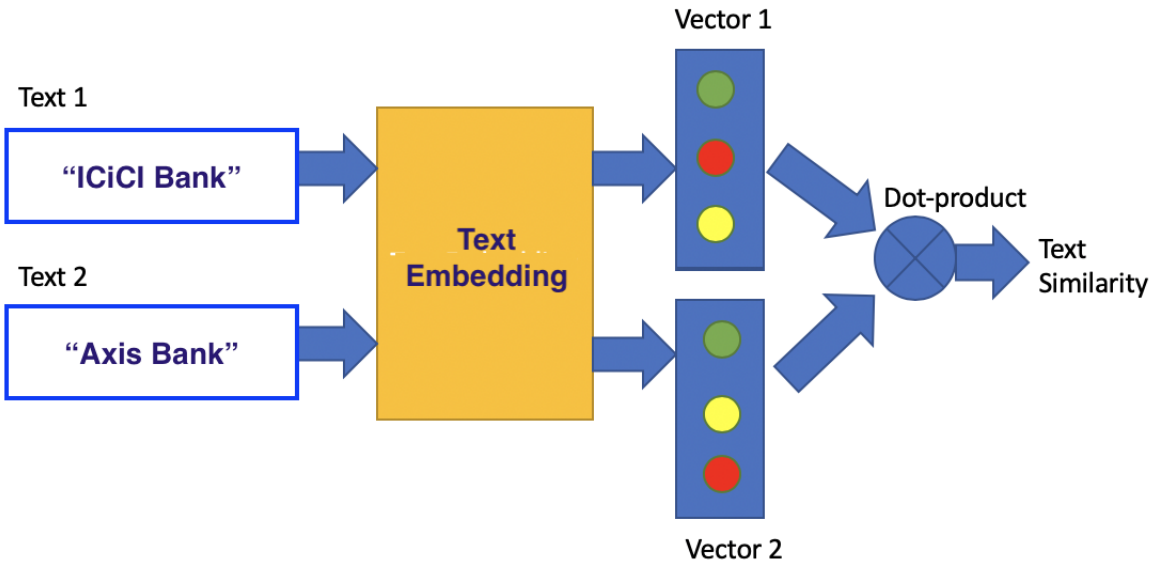


Рис. 4: Алгоритм поиска метрики близости.

Обучение retriever модели как правило основано на contrastive learning, когда для каждого запроса q есть разметка, состоящая из целевого документа d^+ и набора нерелевантных документов $d^- = (d_1^-, \ldots, d_l^-)$, а задачей является увеличение близости векторных представлений для q и d^+, одновременно с уменьшением для q и d^-.

Тогда вероятность релевантности документа $d \in \mathcal{D}$ к запросу q оценивается как:

$$P(d \mid q) = \frac{exp(\langle f_\phi(q)\ ,\ f_\psi(d)\rangle)}{\sum_{d' \in D} exp(\langle f_\phi(q)\ ,\ f_\psi(d')\rangle)} \tag{6}$$

А функция потерь на 1 объекте имеет вид:

$$\mathcal{L}_{Retrieval} = -logP(d^+ \mid q) = -log\frac{exp(\langle f_\phi(q)\ ,\ f_\psi(d^+)\rangle)}{\sum_{d' \in D} exp(\langle f_\phi(q)\ ,\ f_\psi(d')\rangle)} \tag{7}$$

При последующей генерации ответов языковая модель использует несколько наиболее вероятных документов. Другими словами, в качестве контекста для генерации LLM используются top-K фрагментов с наибольшей метрикой близости к запросу:

$$c = \arg\max_{\{d_i\}_{i=1}^{K}} \sum_{i=1}^{K} Similarity(q, d_i). \tag{8}$$

2.3 Архитектуры моделей

В задачах Natural Language Processing (NLP) большое распространение получили трансформерные архитектуры, представленные в работы «Attention Is All You Need» [5] (рис. 5).

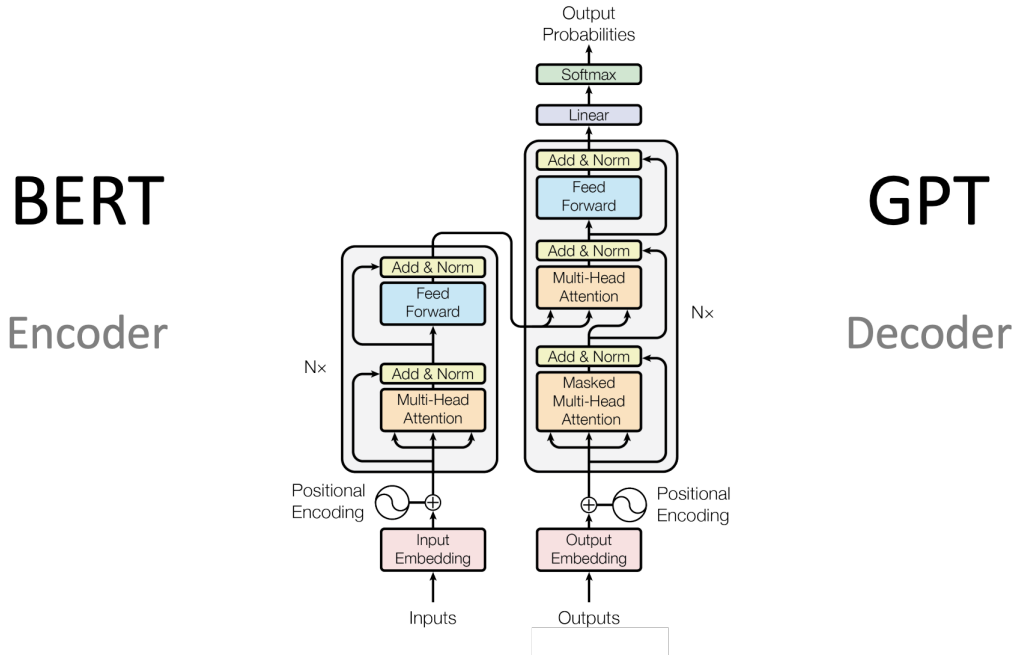


Рис. 5: Encoder-Decoder архитектура трансформер, оригинальное изображение взято из [5].

Ключевой особенностью оригинальной архитектуры является представление в виде encoder-decoder пары, а также добавление механизма внимания, который учитывает влияние каждого элемента последовательности на текущий. Формально, self-attention вычисляется следующим образом:

$$Attention(Q, K, V) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{9}$$

где Q, K, V – это матрицы запросов (queries), ключей (keys) и значений (values) соответственно, а d_k – размерность векторного пространства представлений.

Механизм внимания позволяет модели эффективно обрабатывать длинные текстовые последовательности ценой большей вычислительной сложности. Данная работа ограничивается рассмотрением только трансформерных архитектур.

Дальнейшие исследования показали, что для генеративных задач наибольшую эффективность показывают gpt-подобные decoder-only архитектуры, обученные на кросс-энтропийной функции потерь (2).

Для embedder модели чаще всего применяются bert-подобные encoder-only архитектуры, обученные на контрастивную функцию потерь (7).

2.4 Метод Low-Rank Adaptation (LoRA)

Для эффективной адаптации языковой модели к целевой задаче в условиях ограничения вычислительных ресурсов чаще всего применяется метод низкоранговой адаптации [6]. Идея заключается в обновление исходных параметров $\mathbf{W}$ путём их заморозки и добавления $\Delta\mathbf{W}$, состоящего из низкорангового разложения (рис. 6):

$$\Delta\mathbf{W} = \mathbf{BA}, \text{ где:} \tag{10}$$

- $\Delta\mathbf{W} \in R^{d\times k}$ – матрица адаптации того же размера, что и исходная замороженная матрица весов $\mathbf{W}$.
- $\mathbf{B} \in \mathbb{R}^{d\times r}$, $\mathbf{A} \in \mathbb{R}^{r\times k}$ – обучаемые матрицы.
- $r \ll \min(d, k)$ – ранг разложения.

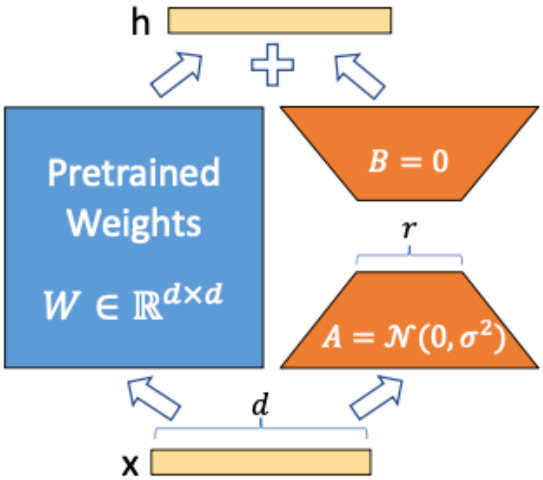


Рис. 6: LoRA адаптер, изображение из [6].

Итоговый слой модели действует следующим образом:

$$W'(x) = h = \mathbf{W}x + \frac{\alpha}{r}\mathbf{B}\mathbf{A}x, \text{где } \alpha \text{ это гиперпараметр масштабирования.} \tag{11}$$

Инициализация параметров:

- $\mathbf{A} \sim \mathcal{N}(0, \sigma^2)$.
- $\mathbf{B} = \mathbf{0}$ (нулевая инициализация).

Среди преимуществ метода можно выделить

- Параметрическая эффективность (обучения 0.5-2% от размера исходных параметров).
- Снижение риска катастрофического забывания.
- Высокая скорость обучения.

2.5 Методы оценки качества

2.5.1 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) - серия метрик, предложенная в работе «ROUGE: A Package for Automatic Evaluation of Summaries» [7] для оценки качества решения задачи суммаризации текста.

Пусть есть 2 текста - сгенерированный G и целевой R. Разобъем их на униграммы (отдельные слова), тогда определены следующие метрики:

$$\textit{ROUGE-1 Precision} = \frac{\textit{Количество совпадающих униграм в } G \textit{ и } R}{\textit{Количество униграм в } G} \tag{12}$$

$$\textit{ROUGE-1 Recall} = \frac{\textit{Количество совпадающих униграм в } G \textit{ и } R}{\textit{Количество униграм в } R} \tag{13}$$

$$\textit{ROUGE-1 F1-score} = 2\frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}} \tag{14}$$

Аналогично определяются метрики ROUGE-N, путём разбиения текста на n-граммы. Метрики ROUGE-L определены с использованием longest common subsequence (LCS):

$$ROUGE\text{-}L\ Precision = \frac{LCS(G,\ R)}{Количество\ униграм\ в\ G} \tag{15}$$

$$ROUGE\text{-}L\ Recall = \frac{LCS(G,\ R)}{Количество\ униграм\ в\ R} \tag{16}$$

Важно отметить, что подобные метрики не могут в полной мере оценить качество, так как являются чувствительными к порядку слов и использованию синонимов.

2.5.2 LLM как судья

Еще одним стандартным подходом для оценки качества суммаризации является использование нейросетевых методов. В частности метод LLM-as-judge предполагает использование генеративных языковых моделей для имитации человеческой оценки.

Множество исследований показывают ([8], [9], [10]), что хотя использование LLM для оценки качества на данный момент не может в полной заменить человеческую оценка, однако между ними уже сейчас имеется согласованность порядка 80% [8].

В данной работе в качестве модели-судьи будет выступать Llama-3.3-70B-Instruct. На вход модели подается вопрос пользователя, релевантный фрагмент текста, эталонный ответ, сгенерированный ответ и подробная инструкцию по оценке. Запрос можно найти в приложении А 7.

2.6 Бенчмарк

Для оценки качества RAG-системы нужна специализированная процедура. На текущий момент в открытом доступе отсутствуют русскоязычные бенчмарки, в полной мере отвечающие всем требованиям. Во многом это обусловлено отсутствием стандартизированных метрик качества контекстно-зависимой QA генерации. Классические статистические методы не подходят для всесторонней оценки реального качества, а крупные коммерческие компании чаще всего используют дорогостоящую человеческую разметку. В связи с этим актуальной становится задача создания бенчмарка, максимально приближенного к практическим сценариям применения RAG.

В качестве доменов для бенчмарка были выбраны 3 следующие сферы: Статьи и регламенты, техническая документация, финансовые отчеты. Для

эффективной работы в этих областях требуется постоянное взаимодействие с большим количеством текстовой документации. Потому в этом сценарии часто внедряются интеллектуальные ассистенты для оптимизации времени поиска нужной информации.

Однако при оценка важно учитывать не только покрываемые сферы, но также и наиболее частные сценарии применения. По этой причине бенчмарк разделен на несколько типов вопросов, каждый из которых оценивает разные компетентности модели.

Представлены следующие типы вопросов:

1. ***Simple*** - прямые вопросы, требующие нахождения одного фрагмента документации. Оценивают качество работы модели в стандартном сценарии применения.

2. ***With errors*** - вопросы, содержащие ряд грамматических и синтаксических ошибок. Требует от модели навыков понимания низкокачественных запросов.

3. ***Trash*** - избыточные запросы, содержащие помимо непосредственного вопроса множество текста на отвлеченную тему. Требует от модели навыков понимания запроса в условиях шума.

4. ***Reformulation*** - запросы, аналогичные *Simple*, однако переформулированные на более технический стиль или составленные с использованием слов, не содержащихся в текстовом фрагменте, содержащем ответ на вопрос. Требует понимания синонимов или экспертизы в предметной области.

5. ***Incorrect by design*** - вопросы, содержащие неверное предположение, провоцирующее на простой, однако неверный ответ. Требует умения не обосновываться на неверный контекст.

6. ***Table*** - запросы, требующие извлечения информации из табличных данных. Оценивает навыки работы со структурированными данными.

7. ***No info*** - вопросы, ответов на которых нет в документационном индексе. Проверяет навык модели, не придумывать ответ, а явно отвечать, что его нет.

8. ***Double*** - запросы, содержащие 2 независимых вопроса одновременно. Оценивает качество работы с разрозненной информацией.

9. ***Multi block*** - вопросы, для ответа на которые требуется использование информации из разных фрагментов текстовой документации. Проверяет навык суммаризации информации из разных источников.

10. ***Logical thinking*** - вопросы, требующие рассуждений по аналогии или по принципу перехода от «общего к частному». Требует от модели понимания мира и предметной области.

Каждый домен разделен на 3 сета:

- *Статьи и регламенты*: медицинские научные статьи (pubmed), регламенты нефтегазовой отрасли, регламенты металлургии.
- *Техническая документация*: документация linux, инструкции по эксплуатации техники, руководства к ПО.
- *Финансовые отчеты*: финансовые отчеты кампаний, аналитические отчеты, отчеты об AI отрасли.

Поиск документов проводился вручную. Вопросы формировались полуавтоматически с последующей верификацией. Итоговый общий размер - 644 примера (рис. 7). Более детально процесс построения будет описан в практической части работы.

	A	B	C	D	E	F	G
1	**Домен документов**	**Сет документов**	**Название документа**	**Отрывок из документа**	**Тип вопроса**	**Вопрос**	**Ответ**
2	Статьи и регламенты	Металлургия	ИТС_11-2019_Производство_алюминия.pdf	4.3 Производство первичного алюминия На алюминиевых заводах в промышленных масштабах применяются три технологии электролиза: - технология электролиза в электролизерах с предварительно обожженными анодами (ОА); - технология электролиза в электролизерах с самообжигающимися анодами и верхним подводом тока к аноду (ВТ); - технология электролиза в электролизерах с самообжигающимися анодами и боковым подводом тока к аноду (БТ).	Simple	Какие технологии электролиза применяются на алюминиевых заводах в промышленных масштабах?	На алюминиевых заводах в промышленных масштабах применяются три технологии электролиза: - технология электролиза в электролизерах с предварительно обожженными анодами (ОА); - технология электролиза в электролизерах с самообжигающимися анодами и верхним подводом тока к аноду (ВТ); - технология электролиза в электролизерах с самообжигающимися анодами и боковым подводом тока к аноду (БТ).
3	Статьи и регламенты	Металлургия	ИТС_11-2019_Производство_алюминия.pdf	4.3 Производство первичного алюминия На алюминиевых заводах в промышленных масштабах применяются три технологии электролиза: - технология электролиза в электролизерах с предварительно обожженными анодами (ОА); - технология электролиза в электролизерах с самообжигающимися анодами и верхним подводом тока к аноду (ВТ); - технология электролиза в электролизерах с самообжигающимися анодами и боковым подводом тока к аноду (БТ).	With errors	Пораскажи про способы эмлектролиза для производства аллюминия	На алюминиевых заводах в промышленных масштабах применяются три технологии электролиза: - технология электролиза в электролизерах с предварительно обожженными анодами (ОА); - технология электролиза в электролизерах с самообжигающимися анодами и верхним подводом тока к аноду (ВТ); - технология электролиза в электролизерах с самообжигающимися анодами и боковым подводом тока к аноду (БТ).
4	Статьи и регламенты	Металлургия	ИТС_11-2019_Производство_алюминия.pdf	4.3 Производство первичного алюминия На алюминиевых заводах в промышленных масштабах применяются три технологии электролиза: - технология электролиза в электролизерах с предварительно обожженными анодами (ОА); - технология электролиза в электролизерах с самообжигающимися анодами и верхним подводом тока к аноду (ВТ); - технология электролиза в электролизерах с самообжигающимися анодами и боковым подводом тока к аноду (БТ).	Trash	Привет! Мне нужно узнать кое-что для проекта. Дело в том, что я пишу работу про производство металлов, и мне нужно понять, какие вообще существуют способы получения алюминия на заводах, особенно что-то связанное с электролизом. Если можешь, объясни это простыми словами, без сложных терминов. Ну, или просто перечисли основные технологии, если лень. Спасибо!	На алюминиевых заводах в промышленных масштабах применяются три технологии электролиза: - технология электролиза в электролизерах с предварительно обожженными анодами (ОА); - технология электролиза в электролизерах с самообжигающимися анодами и верхним подводом тока к аноду (ВТ); - технология электролиза в электролизерах с самообжигающимися анодами и боковым подводом тока к аноду (БТ).

Рис. 7: Пример данных в бенчмарке.

3 Постановка задачи

В данной работе обратим внимание на существующие проблемы современных языковых моделей, построим и опишем эффективный пайплайн дообучения LLM в задаче извлечения информации из контекста на примере русскоязычного RAG, а также представим пайплайн оценки качества. Исследование посвящено улучшению исключительно генеративной части RAG-пайплайна, считая остальные компоненты фиксированными.

Формально можно поставить задачу оптимизации со следующими составляющими:

- **Модель генерации** $g_\theta : \mathcal{X} \to \mathcal{Y}$ с обучаемыми параметрами θ.
- **Обучающие данные** $D_{train} = (x_i, y_i)_{i=1}^{M}$ где:
  - $x_i \in \mathcal{X}$ — входные тексты, состоящие из контекста и запроса.
  - $y_i \in \mathcal{Y}$ — эталонные ответы.
- **Гиперпараметры обучения** η, включающие:
  - Learning rate: λ.
  - Коэффициент регуляризации: μ.
  - Размер батча: bs.
  - Lora rank: r.
- **Оценочный бенчмарк**: benchmark.
- **Метрика итогового качества**, определяемая композицией метрик ROUGE-L и оценки LLM-судьи:

$$Quality = \{q_i(g_\theta, benchmark) \mid q_i \in \{ROUGE\text{-}L, LLM\text{-}Score\}\} \quad (17)$$

Требуется найти оптимальные параметры модели, гиперпараметры и обучающие данные, максимизирующие качество на бенчмарке:

$$(\theta^\star, \eta^\star, D^\star_{train}) = \underset{\theta,\eta,D_{train}}{\arg\max}\, Quality(g_\theta, benchmark) \quad (18)$$

При ограничениях в объеме вычислительных ресурсов VRAM $\leq Capacity$.

4 Обзор существующих проблем и решений

4.1 Lost in the middle

Качество генерации моделей может ухудшаться из-за слишком обширного контекста. Применительно к задаче извлечения информации эта проблема подробно описана в статье «Lost in the Middle: How Language Models Use Long Contexts» [4]. В ней авторы брали по 20 фрагментов страниц с Википедии, лишь один из которых содержал точный ответ на вопрос, а затем сравнивали качество ответа в зависимости от положения этого документа. Общая тенденция такова – наивысшее качество в начале, хорошее в конце, а между ними критическое снижение. При этом суммарная длина контекста составляла всего порядка 4000 токенов.

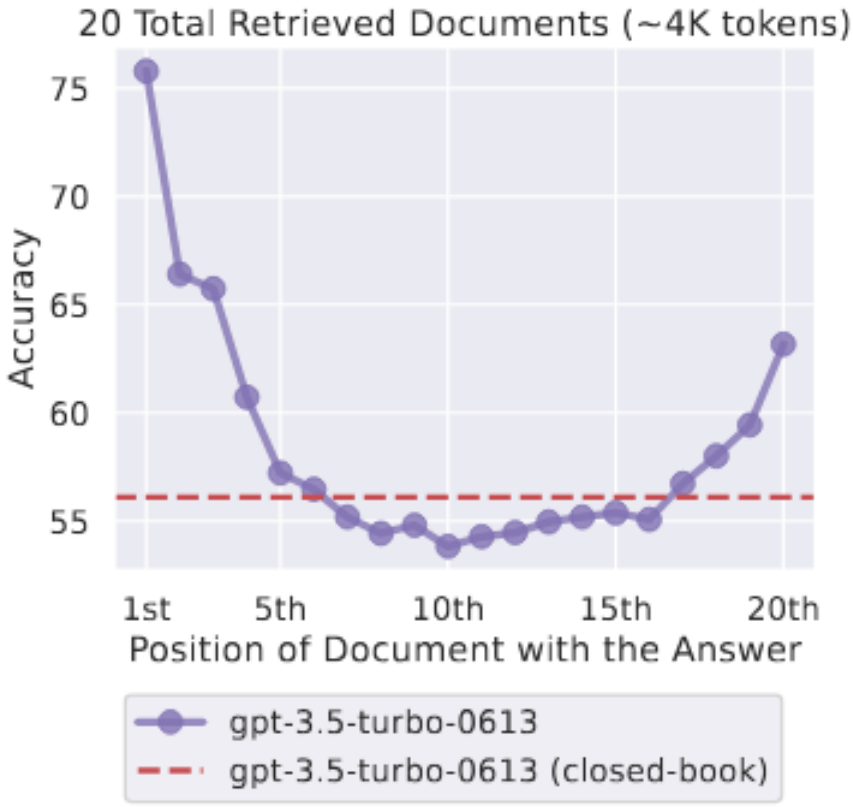


Рис. 8: Качество ответа в зависимости от положения релевантной информации в контексте, взято из [4].

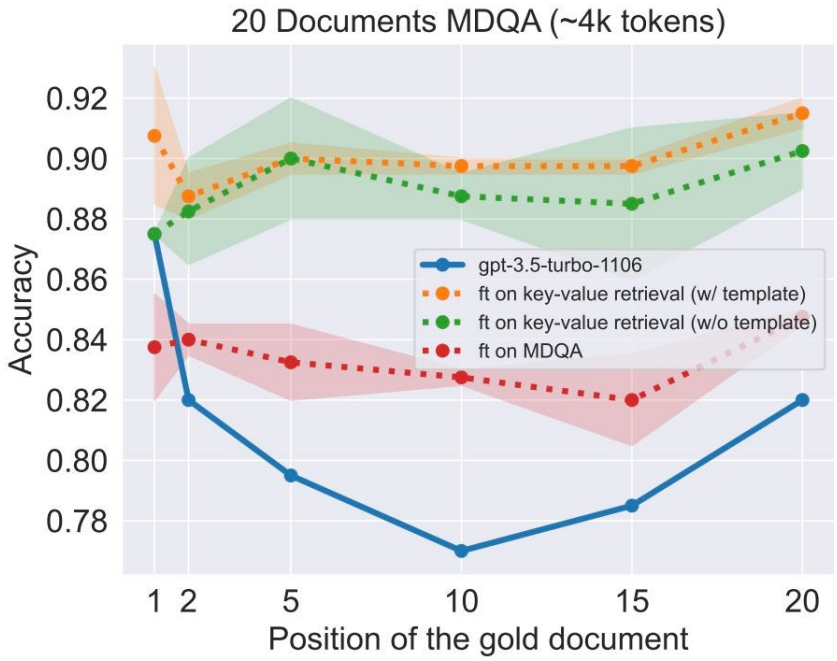
На практике в задаче RAG такая проблема чаще всего решается добавлением в pipeline генерации дополнительной reranker-модели, предназначенной для ранжирования документов, извлеченных из базы знаний retrieval-моделью, на основании их релевантности к запросу.

Такой подход эффективен, однако фундаментально никак не решает существующую проблему. Кроме того, из-за эффекта кумулятивной ошибки,

потенциально возникающей при использовании слабых retrieval- и reranker-моделей, даже хорошая генеративная модель может показывать низкое качество по причине неуместного контекста.

Способ борьбы с этой проблемой предлагают авторы статьи «From Artificial Needles to Real Haystacks: Improving Retrieval Capabilities in LLMs by Finetuning on Synthetic Data» [11]. Они генерируют синтетический датасет из набора словарей и обучают модель находить значение по ключу.

Дообучение модели на такую прокси-задачу закрепляет навык поиска информации в контексте и почти полностью решает описанную в предыдущей статье проблему. Кроме того, это не ухудшает обобщающую способность – результаты модели на общих бенчмарках если и упали, то незначительно.



(a) GPT-3.5 Turbo and the finetuned versions.

Рис. 9: Качество после дообучения, взято из [11].

4.2 Нерелевантный контекст

Как упоминалось ранее, еще одной серьезной проблемой контекстной генерации являются нерелевантные документы. Они могут как просто ухудшать качество ответа, так и приводить к «галлюцинациям» модели.

Этот эффект детально описан в статьях «Large Language Models Can Be Easily Distracted by Irrelevant Context» [2] и «Making Retrieval-Augmented Language Models Robust to Irrelevant Context» [3].

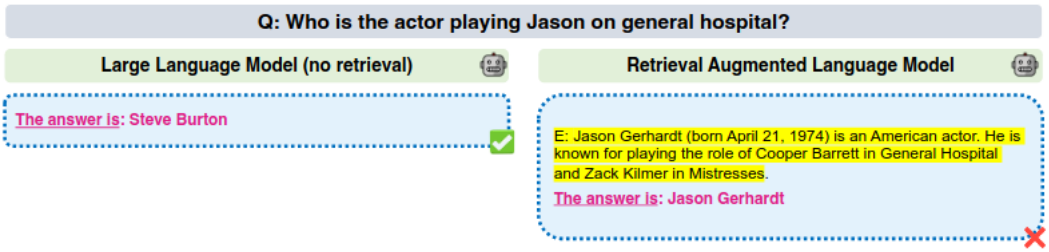


Рис. 10: Неверный ответ из-за нерелеватного контекста, взято из [3]

Кроме изучения негативного влияния, в работе [3] также приведен способ решения проблемы – дообучение на датасете со смесью релевантных и нерелевантных документов в контексте.

Духовным наследником этой идеи является исследование «RAFT: Adapting Language Model to Domain Specific RAG» [12], где описан метод Retrieval Augmented Fine Tuning (RAFT). Основная идея подхода заключается в дообучении модели на синтетически расширенных датасетах. Авторы предполагают следующую процедуру обработки SFT-датасетов с контекстно-зависимыми запросами:

1. ***Обогащение контекста.***

   К исходному контексту запроса добавляются релевантные фрагменты из других запросов, что позволяет модели научиться работать в условиях, максимально схожих с реальными, и выделять полезные данные в условиях шума.

2. ***Разметка Chain-of-Thought (CoT).***

   Ответы аннотируются с явным выделением логических цепочек рассуждений. Это позволяет модели учиться обосновывать свой ответ и аргументированно использовать релевантный контекст. Такая разметка делается синтетически при помощи запросов к другой LLM.

3. ***Добавление негативных примеров.***

   В датасет включаются запросы без релевантных документов, где ожидаемым ответом ставится отказ. Это формирует у модели навык распознавать неподходящий контекст и явно отмечать неспособность предоставить корректный ответ в этой ситуации.

Авторы статьи отмечают, что CoT-разметка играет ключевую роль в этом подходе - она не только делает обучение более стабильным, но и существенно увеличивает качество итоговой модели.

	PubMed	HotpotQA	HuggingFace	Torch Hub	TensorFlow
RAFT w.o CoT	68.30	25.62	59.07	**86.56**	83.21
RAFT	**73.30**	**35.28**	**74.00**	84.95	**86.86**

Рис. 11: Влияние CoT, взято из [12].

Данный подход позволяет адаптировать языковые модели к генерации ответов в условиях шума, а также развивает их способность аргументировать ответы. Глобально этот подход делает этап обучения наиболее близким к реальному сценарию применения модели.

	PubMed	HotPot	HuggingFace	Torch Hub	TensorFlow
GPT-3.5 + RAG	71.60	**41.5**	29.08	60.21	65.59
LLaMA2-7B	56.5	0.54	0.22	0	0
LLaMA2-7B + RAG	58.8	0.03	26.43	08.60	43.06
DSF	59.7	6.38	61.06	84.94	86.56
DSF + RAG	71.6	4.41	42.59	82.80	60.29
RAFT (LLaMA2-7B)	**73.30**	35.28	**74.00**	**84.95**	**86.86**

Рис. 12: Результаты RAFT, взято из [12].

4.3 Построение решения задачи

Для оптимизации времени разработки эффективного подхода было принято решения разделить процесс обучения на несколько ключевых этапов и в дальнейшем оценить влияние каждого из них.

1. ***Адаптация к русскому языку.***

   Современные языковые модели обучаются на обширных мультиязычных текстовых корпусах, что обеспечивает их базовую функциональность во многих сценариях применения. Однако доминирование англоязычных данных в обучающих выборках приводит в выраженному

ухудшению качество генерации и понимания для языков с меньшей репрезентацией, включая русский.

2. ***Улучшение качества контекстно-зависимой генерации.***

   Следующим этапом стал сравнительный анализ различных подходов адаптации языковой модели к задаче контекстной генерации. Главной целью было сравнение классического подхода и RAFT.

3. ***Адаптация к большому контексту.***

   Для решения проблемы «lost in the middle» [4], а также увеличения размера эффективного контекста модели, было проведено сравнительное дообучение на синтетических данных с размером контекста порядка 7-8 тысяч токенов.

5 Описание практической части

В силу ориентированности работы на задачу RAG с большим количеством входных токенов и ограниченности вычислительных ресурсов, выбор моделей был ограничен архитектурами в диапазоне 1–3 млрд параметров.

Основными исследуемыми архитектурой стали модели Qwen2.5-1.5B-Instruct и Qwen2.5-3B-Instruct от Qwen team, Alibaba Cloud. Несмотря на официальное отсутствие поддержки русского языка, на практике модели эффективно справляется с мультиязычными задачами. А также согласно различным независимым рейтингам (например, Open LLM Leaderboard и MERA Leaderboard), являются одними из самых эффективных в своем весовой категории для множества стандартных задач.

Процесс обучения больших языковых моделей требует большого количества вычислительных ресурсов и времени. Для проведения всей экспериментальной части в этом исследовании использовалась следующая конфигурация:

- GPU NVIDIA Tesla V100 PCIe 32 GB.
- 20 VCPU.
- 64 GB RAM.
- 400 GB SSD.

5.1 Построение RAG пайплайна

1. ***Обработка документов***

   Исходно все документы бенчмарка были представлены в формате pdf, однако он не подходит для чанкования и подачи в контекст LLM из-за своей структуры. По этой причине все документы были переведены в формат markdown, что позволило сохранить общую структуру документы (подразделы, табличные данных) и преобразовать данные в исключительно текстовый формат. Для этого этапа была использована библиотека docling.

2. ***Подготовка инфраструктуры***

   В качестве embedder модели использовался intfloat/multilingual-e5-large. Данная модель показывает наилучшее качество на бенчмарке ruMTEB и полностью подходит для нашей задачи.

При чанковании текст разделялся на фрагменты по 300 токенов, что составляет примерно 10-20 предложений, в зависимости от их длины. Разделение проводилось по знакам препинания с перекрытием 50 токенов между соседними фрагментами текста.

Так как в каждом домене бенчмарка было 3-10 документов, то не было необходимости использовать полноценную векторную базу данных. Для построения тематических индексов использовалась библиотека FAISS.

3. ***Ретрив***

   При генерации контекста к запросам бенчмарка, использовались top-10 наиболее близких документа. Итоговый контекст запроса к модели составлял порядка 4-5 тысяч токенов. Для корректного сравнения различных генеративных моделей набор документов фиксировался. Качество ретрива высокое, в 85% случаев релевантный фрагмент находился в top-5 документах.

5.2 Процесс формирования бенчмарка

Создание бенчмарка можно разделить на следующие этапы:

1. ***Формирование индексов и поиск документов***

   Данный этап является ключевым. Он должен быть тщательно спланировал и реализован полностью вручную, отражая реальный потребности технологии.

2. ***Типизация вопросов***

   В нашей работе использовалась типизация на 10 классов, где вопросы «Multi block» и «Logical thinking» считались усложненными, так как требовали модели дополнительного понимания контекста, а не только извлечения релевантного фрагмента текста. Наша классификация покрывает большинство сценариев применения RAG-систем.

3. ***Создание примеров***

   Сперва необходимо вручную найти независимый фрагмент документа, к которому можно задать правдоподобный вопрос. Это делалось вручную и таким образом формировались вопросы типа «Simple». Соответствующие этому вопросу типы «With errors», «Trash», «Reformulation»,

«Incorrect by design» формировались автоматически через запросы к языковой модели, в частности использовались DeepSeek-R1 и GPT-4o.

Вопросы «No Info» генерировались автоматически, путем отправки нескольких глав документа языковой модели. Оставшиеся типы вопросов генерировались полностью вручную.

4. ***Верификация***

   При формировании итогового бенчмарка важно просмотреть все примеры и очистить данные от мусора.

Данный подход позволяет значительно оптимизировать время разработки без существенных потерь в итоговом качестве данных.

5.3 Конфигурации обучения

При реализации процесса дообучения использовались следующие гиперпараметры:

Таблица 2: Гиперпараметры

Параметр	**Диапазон**
LoRA rank	16
LoRA alpha	8 - 16
Learning rate	0.00001 - 0.00004
Weight decay	0.01
Global batch size	24 - 32
Warmup fraction	10%
Learning rate scheduler	linear
Optimizer	AdamW 8bit

Через LoRA адаптеры обучались attention и feedforward network блоки, остальные компоненты модели не модифицировались:

```
model = FastLanguageModel.get_peft_model(
    model,
    r=16,
    lora_alpha=16,
    lora_dropout=0,
    target_modules=["q_proj", "k_proj", "v_proj", "up_proj", "down_proj", "o_proj",
        "gate_proj"],
)
```

Обучение проводилось с использованием библиотеки unsloth. Конфигурации для каждого из типов обучения находятся в приложении С 7.

5.4 Адаптация к русскому языку

Главной проблемой при руссификации моделей остается недостаточное количество качественных русскоязычных данных для обучения. Датасет `Saiga_scored` представляет собой смесь русскоязычных QA запросов из различных источников.

Таблица 3: Описание Saiga

Параметр	**Описание**
Общее количество данных	42к
Размер обучающей выборки	35к
Размер валидационной выборки	5к
Размер тестовой выборки	2к
Размер контекста	2048 токенов
Содержание	Инструкционные запросы, адаптированные под русский язык (75% данных на русском языке)
Источник данных	Смешанный - есть как примеры из мультиязычных датасетов, содержащих русский, так и синтетически сгенерированные ответы других языковых моделей

Среди существующих инструментов оценки языковых моделей для русского языка выделяется мультимодальный бенчмарк MERA (Multimodal Evaluation of Russian-language Architectures) [13]. Этот набор является собранием различных задач, включая адаптированную для русского языка версию MMLU.

Оригинальный MMLU [14] предназначен для измерения профессиональных знаний модели, приобретенных в процессе обучения в различных областях. Задачи охватывают 57 различных тем: гуманитарные науки, математика, инженерия и другие. Данные состоят из вопроса и четырёх вариантов ответа, среди которых только один правильный. Метрика оценки - accuracy.

Из-за большого разнообразия затрагиваемых тем в данной работе набор использовался для оценки эффективности адаптации архитектур к русскому языку.

5.5 WebGLM-QA

В качестве обучающего набора для адаптации LLM к задаче контекстной генерации выступил корпус WebGLM-QA, разработанный для web-enhanced question-answering system based on the General Language Model (WebGLM) [15]. Предварительно данные были адаптированны под русский язык с помощью машинного перевода.

Таблица 4: Описание WebGLM-QA

Параметр	**Описание**
Общее количество данных	45к
Размер обучающей выборки	44к
Размер валидационной выборки	1000
Размер тестовой выборки	400
Размер контекста	2048 токенов
Содержание	Запросы с результатами веб-поиска. Имитируют реальные сценарии применения.
Источник данных	Вопросы взяты из открытых источников, контекст собирался с помощью автоматического веб-поиска, эталонные ответы формировались с помощью языковой модели. В дальнейшем проведена фильтрации данных, путем проверки ссылок и удаления ошибочных фрагментов.
Особенности	В ответе также содержатся указания на используемые источники, что по задумке должно научить модель эффективнее использовать контекст.

Оценка эффективности проводилась на итоговом бенчмарке.

5.6 WebGLM-RAFT

Для модификации датасета WebGLM-QA под метод RAFT использовалась следующая конфигурация:

Таблица 5: Построение WebGLM-RAFT

Параметр	**Значение**	**Описание**
Количество отвлекающих документов	3 фрагмента	Для добавления шума в контекст генерации модели, помимо релевантного фрагмента, добавлялось еще 3 фрагмента с предыдущих запросов. В последствии порядок источников перемешивался, а номер источников в ответе актуализировался для текущей индексации.
COT разметка	Llama-3.3-70B-Instruct	Для генерации последовательности рассуждений использовалась Llama-3.3-70B-Instruct через Nebius API. Так как столь громоздкую модель хостить локально не представляется возможным, использовалось API для генерации ответов (Формат запроса указан в приложении В 7).
Доля негативных примеров	10%	Дополнительно в датасет было добавлено 5000 примеров без релевантного для запроса контекста. В качестве нерелевантного брался контекст с 4 предыдущих запросов. В качестве эталонного ответа ставился шаблонный отказ.

Итоговая конфигурация данных:

Таблица 6: Описание WebGLM-RAFT

Параметр	**Описание**
Общее количество данных	50к
Размер обучающей выборки	45к
Размер валидационной выборки	3500
Размер тестовой выборки	1500
Размер контекста	4096 токенов

Оценка эффективности проводилась на итоговом бенчмарке.

5.7 Синтетические данные

Для построения синтетических данных была написана функция на python, создающая случайные словари. Детальнее:

- 100 словарей из 2-4 пар ключ-значение.
- Ключи это кортежи из 3-4 чисел в диапазоне 100-999.
- Значения это целые числа в диапазоне 1000-9999.
- В начале создаются golden key и golden value, которые в дальнейшем будут учавствовать в запросе к модели. Они помещаются в случаный словарь.
- Оставшиеся элементы в словарях генерируются с учётом корректности и единственности golden пары.
- В запрос к модели подается перемешанный golden key. Модель должна ответить в json формате, указав оригинальный golden key, соответствующее ему golden value, а также номер словаря, в котором находится пара.

Таблица 7: Описание синтетических данных

Параметр	**Описание**
Общее количество данных	1500
Размер обучающей выборки	1400
Размер валидационной выборки	100
Размер контекста	8192 токена

6 Результаты

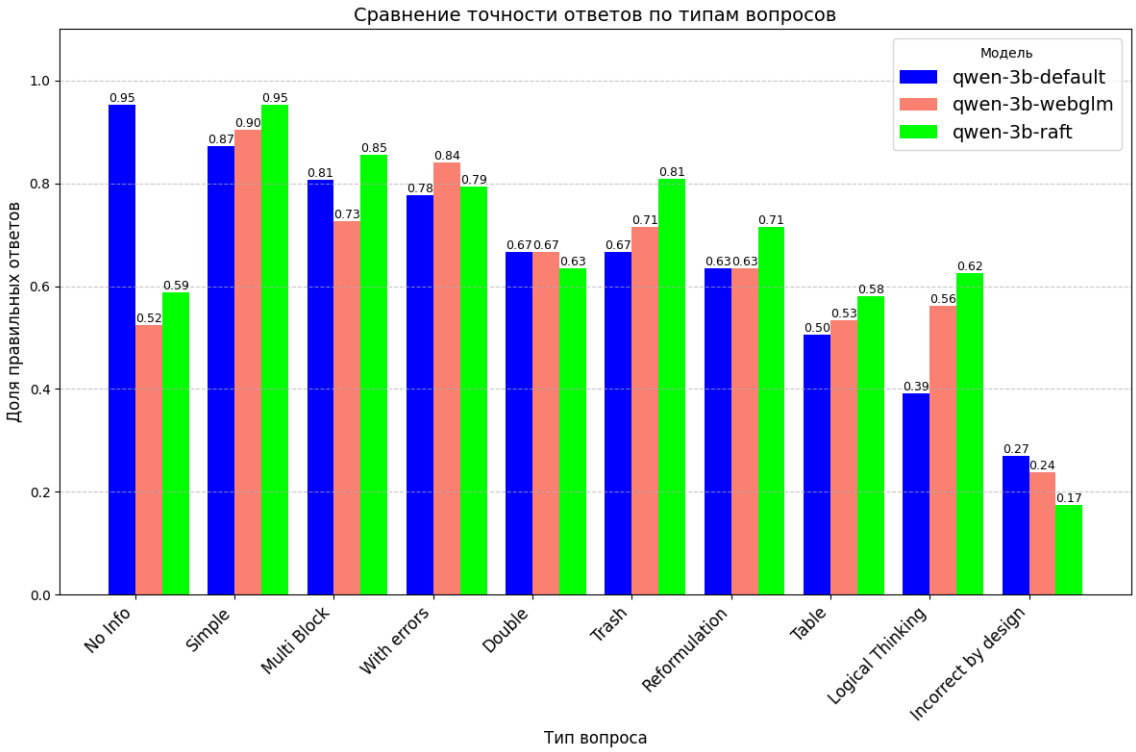
6.1 Результаты русскоязычной адаптации

Таблица 8: MMLU accuracy

Model	Default	RU Finetune
Qwen2.5-1.5B Instruct	39.36%	42.14%
Qwen2.5-3B Instruct	47.96%	49.55%

По метрикам на MMLU виден прирост, однако дальнейшее обучение этих моделей на WebGLM показало существенное ухудшение итогового качества для Qwen2.5-1.5B Instruct — доля верных ответов на бенчмарке упала на 5-10%. Вероятно это связано с тем, что модель переобучалась под формат данных и начинала хуже работать с контекстной информацией. Для Qwen2.5-3B Instruct падение было не столь существенно, но итоговое качество так же не улучшилось. В целом можно сказать, что этот этап необходим только для моделей, плохо работающих с русскоязычными текстами.

6.2 Результаты WebGLM и RAFT



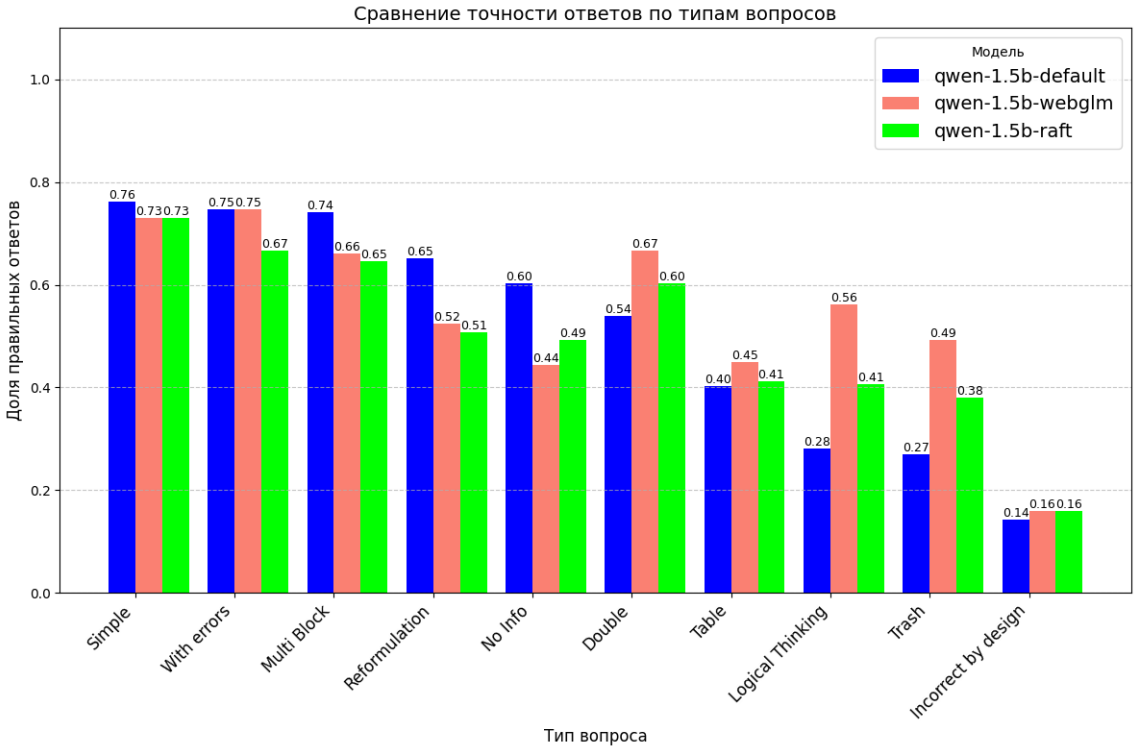
Можно заметить существенный рост качества RAFT на всех типах вопросов, кроме «No Info» и «Incorrect by design». Важно отметить, что сни-

жение качества ответа на «No Info» связо в первую очередь со снижение доли шаблонных отказов при ответах.

Доля отказов:

- Default - 30% (из них 70% неуместно).
- WebGLM - 9% (из них 49% неуместно).
- RAFT - 8% (из них 36% неуместно).

По результатам обучения 1.5B моделей так же был получен прирост, однако в данном случае метод RAFT показал себя хуже. Вероятно такого количества параметров недостаточно для качественной работы модели на этой задаче.



В целом можно увидеть преимущество обучения RAFT относительно классического finetune на WebGLM для среднего размера моделях. Этот этап крайне эффективен и не требует дополнительных модификаций. Итоговые метрики можно увидеть в сводных таблицах:

Таблица 9: *Оценки LLM.*

Model	AVG Score	Accuracy	Irrelevant Refuse
Qwen2.5-1.5b-default	3.31	0.50	0.18
Qwen2.5-1.5b-WebGLM	**3.37**	**0.53**	**0.06**
Qwen2.5-1.5b-RAFT	3.32	0.50	0.15
Qwen2.5-3b-default	3.86	0.64	0.21
Qwen2.5-3b-WebGLM	3.73	0.63	0.05
Qwen2.5-3b-RAFT	**3.86**	**0.67**	**0.03**
Qwen2.5-32b-default	4.28	0.77	0.12

Таблица 10: *ROUGE-L метрики.*

Model	Precision	Recall	F1
Qwen2.5-1.5b-default	0.18	0.34	0.20
Qwen2.5-1.5b-WebGLM	**0.19**	0.36	**0.22**
Qwen2.5-1.5b-RAFT	0.14	**0.40**	0.19
Qwen2.5-3b-default	0.18	0.40	0.23
Qwen2.5-3b-WebGLM	**0.23**	0.41	**0.27**
Qwen2.5-3b-RAFT	0.14	**0.51**	0.20
Qwen2.5-32b-default	0.36	0.53	0.40

6.3 «Lost in the middle» и синтетические данные

Для исследования актуальности проблемы «lost in the middle» были проведены замеры на срезке из 231 примера бенчмарка с разным количеством документов и их положениями в контексте.

Таблица 11: *Качество Qwen2.5-3b-default при разных положениях документов, метрика - среднее значение accuracy.*

Documents	Default order	Reverse order	Random shuffle
Top-5 (2k context)	**0.68**	0.64	**0.68**
Top-10 (4k context)	**0.72**	0.71	0.68
Top-20 (8k context)	0.68	0.68	**0.71**

Можно сделать вывод, что на таком размере контекста у современных моделей нет явно выраженного эффекта «lost in the middle», а также количество документов имеет слабое влияние на результат при достаточно качественной retriever модели (в данном случае $recall@10 > 0.9$). Дообучение на синтетике так же не привело к росту качества на бенчмарке, а потому расценено как неактуальное.

7 Заключение

В этой работе было проведено исследование существующих проблем LLM в задаче извлечения информации из контекста, а также проведен сравнительных анализ различных подходов обучения. По результатам проведенных экспериментов можно сделать следующее заключение:

1. Для современных мультиязычных моделей вроде Qwen2.5 нет необходимости в дополнительном этапе русификации. В данном случае он приводил даже к ухудшению итогового качества в сценарии RAG пайплайна, т.к. модель начинала хуже обрабатывать контекст.

2. Подход RAFT показал улучшение качества на реальных задачах в сравнение с классическим подходом дообучения. В силу простоты и доступности способа формирования данных, подход можно считать крайне эффективным для дообучения под формат задач с контекстной информацией.

3. На достаточном для RAG-задач размере контекста не выявлена существенная зависимость от положения релевантных документов. В связи с этим, для RAG на контексте не более 10 тыс. токенов проблему можно считать неактуальной даже для моделей с небольшим количеством параметров.

Результаты показывают, что описанный подход построения RAG-бенчмарков является эффективным и может быть использован с последующим расширением типизации вопросов и покрываемых доменов.

Дальнейшие направления работы могут включать в себя адаптацию эффективных методов дообучения для языковых моделей с менее чем 2 миллиардами параметров, а также дальнейшее изучение способов сокращения разрыва в качестве между языковыми моделями различных весовых категорий в задаче извлечения информации из контекста.

Список литературы

[1] *Lewis, Patrick.* Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. — 2020. https://arxiv.org/abs/2005.11401.

[2] *Asai, Akari.* Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. — 2023. https://arxiv.org/abs/2310.11511.

[3] *Yoran, Ori.* Making Retrieval-Augmented Language Models Robust to Irrelevant Context. — 2023. https://arxiv.org/abs/2310.01558.

[4] *Liu, Nelson F.* Lost in the Middle: How Language Models Use Long Contexts. — 2023. https://arxiv.org/abs/2307.03172.

[5] *Vaswani, Ashish.* Attention Is All You Need. — 2017. https://arxiv.org/abs/1706.03762.

[6] *Hu, Edward J.* LoRA: Low-Rank Adaptation of Large Language Models. — 2021. https://arxiv.org/abs/2106.09685.

[7] *Lin, Chin-Yew.* ROUGE: A Package for Automatic Evaluation of Summaries. — 2004. https://aclanthology.org/W04-1013.pdf.

[8] *Zheng, Lianmin.* Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. — 2023. https://arxiv.org/abs/2306.05685.

[9] *Bavaresco, Anna.* LLMs instead of Human Judges? A Large Scale Empirical Study across 20 NLP Evaluation Tasks. — 2024. https://arxiv.org/abs/2406.18403.

[10] *Gu, Jiawei.* A Survey on LLM-as-a-Judge. — 2024. https://arxiv.org/abs/2411.15594.

[11] *Xiong, Zheyang.* From Artificial Needles to Real Haystacks: Improving Retrieval Capabilities in LLMs by Finetuning on Synthetic Data. — 2024. https://arxiv.org/abs/2406.19292v2.

[12] *Zhang, Tianjun.* RAFT: Adapting Language Model to Domain Specific RAG. — 2024. https://arxiv.org/abs/2403.10131.

[13] *Fenogenova, Alena.* MERA: A Comprehensive LLM Evaluation in Russian. — 2024. https://arxiv.org/abs/2401.04531.

[14] *Hendrycks, Dan.* Measuring Massive Multitask Language Understanding. — 2020. https://arxiv.org/abs/2009.03300.

[15] *Liu, Xiao.* WebGLM: Towards An Efficient Web-Enhanced Question Answering System with Human Preferences. — 2023. https://arxiv.org/abs/2306.07906.

Приложение

Приложение А: Запрос для LLM-судьи

```
def get_judge_prompt(question, relevant_context, reference_answer, model_answer):
    messages = []

    messages.append({"role": "system", "content": ("You are a helpful assistant "
    that scores RAG answers against the ground truth. Always respond with a JSON "
    object containing a 'comment' (string), 'is_inappropriate_refusal' (0-1), "
    "'score' (number 1-5), 'is_correct' (0-1) fields.")})

    judge_prompt = f"""
    **Question**: {question}
    **Relevant context**: {relevant_context}
    **Reference Answer**: {reference_answer}
    **Model Answer**: {model_answer}

    **Evaluation Rules**:
    1. Scoring Scale:
    - 5: Good answer (factually correct, complete)
    - 4: Mostly correct with minor issues
    - 3: Partially correct but has inaccuracies
    - 2: Mostly incorrect but contains relevant elements
    - 1: Completely wrong or irrelevant

    2. Binary Correctness:
    - 1 (Correct): Answer conveys the same meaning as reference
    (exact wording not required)
    - 0 (Incorrect): Meaning differs from reference or contains false information

    3. Binary Refuse:
    - 1 (Inappropriate refusal): The model answer is a refusal,
    while the reference contains a full answer
    - 0 (A normal answer or an appropriate refusal)

    4. Key Principles:
    - Focus on whether the model answers the question asked,
    not on whether it exactly matches the reference
    - There is no need to lower the score if the model's answer
    omits some facts that are insignificant to the question
    - If an answer seems redundant in relation to the reference, but the
    information is contained in a relevant context, then there is no need
    to lower the score for this
    - Ignore stylistic differences if core meaning is preserved
    - Inappropriate refusal should always have score 2 out of 5!

    Provide analysis in JSON format: {{"comment": "...",
    "is_inappropriate_refusal": 0|1, score": 1-5, "is_correct": 0|1}}"""

    messages.append({"role": "user", "content": judge_prompt})

    return messages
```

Приложение B: Запрос при генерации COT разметки

```
def get_cot_prompt(context_list, question, answer):
    context = ''
    for i, c in enumerate(context_list):
        context += f'Источник [{i+1}]:'+"\n"+c+"\n\n"

    prompt = (
        f"# Контекстная информация:\n\n{context}\n\n"
        f"# Вопрос:\n\n{question}\n"
        f"# Оригинальный ответ:\n\n{answer}\n"
        "# Задача:\n\n"
        "Твоя задача состоит в том, чтобы дать более детальный ответ на" \
        "вопрос, используя исключительно информацию из контекста.\n" \
        "Для этого последовательно порассуждай про то, какую информацию" \
        "хочет узнать пользователь и как информация из " \
        "оригинального ответа с этим соотносится.\n" \
        "В своем ответе не ссылайся на оригинальный ответ, но учти, что он " \
        "абсолютно корректен и тебе нужно лишь дать расширенную его версию.\n" \
        "Указывай номер источника, про который рассуждаешь.\n" \
        "Последний абзац должен начинаться со слова ОТВЕТ:\\n и " \
        "содержать полный ответ на поставленный вопрос.\n"
        "В нем особенно важно указывать номера использованных источников." \
        "# Твой расширенный ответ:\n\n"
    )
    return prompt
```

Приложение C: конфигурации обучения

Таблица 12: RU конфигурация обучения

Параметр	Значение
Learning rate	0.00004
Device batch size	8
Gradient accumulation	4
Global batch size	32
Epoch	1

Таблица 13: Default WebGLM конфигурация обучения

Параметр	**Значение**
Learning rate	0.00002
Device batch size	8
Gradient accumulation	4
Global batch size	32
Epoch	1

Таблица 14: RAFT WebGLM конфигурация обучения

Параметр	**Значение**
Learning rate	0.00002
Device batch size	4
Gradient accumulation	8
Global batch size	32
Epoch	1

Таблица 15: Synth конфигурация обучения

Параметр	**Значение**
Learning rate	0.00001
Device batch size	2
Gradient accumulation	12
Global batch size	24
Epoch	1