**Методы Retrieval-Augmented Generation (RAG): экспериментальное исследование в задачах генерации в NLP**

**1. Введение**

В настоящее время большие языковые модели демонстрируют впечатляющие возможности генерации текста, однако их знания ограничены обучающими данными и могут быть неактуальными или неполными. Для повышения качества ответов в задачах вопрос–ответ и других генеративных задачах NLP востребован подход **Retrieval-Augmented Generation (RAG)** – сочетание генеративной модели с системой поиска релевантной информации из внешней базы знаний. Это позволяет моделям добавлять в контекст необходимую фактологическую информацию и тем самым делать ответы более обоснованными и точными.

В условиях бурного развития LLM и чат-ботов актуально исследовать, как различные параметры RAG-пайплайна влияют на качество вывода. В этой работе рассматривается вопрос: *каким образом выбор параметров нарезки текстов (размер чанка, перекрытие), числа извлекаемых фрагментов (top\_k) и стратегии поиска (простая выборка, переписанный запрос, повторное ранжирование) отражается на качестве ответов?* Цель исследования – построить и протестировать пошаговый RAG-пайплайн на примере конкретной задачи, определить оптимальные настройки и продемонстрировать методику оценки качества ответов по метрикам достоверности (faithfulness), релевантности (relevancy) и семантического сходства. Мы суммируем результаты в виде среднего комбинированного показателя для удобства сравнения конфигураций.

**2. Методология**

В основе нашего подхода лежит стандартный RAG-пайплайн с тремя основными этапами: **нарезка документов на чанки и векторизация**, **индексация и поиск**, **генерация ответа с контекстом**, а также этап автоматической оценки.

* **Нарезка на чанки:** Большие тексты разбиваются на более мелкие сегменты (чанки), поскольку LLM и модели эмбеддингов ограничены по длине входного контекста. Мы используем параметры chunk\_size (размер чанка в словах) и chunk\_overlap (перекрытие между соседними чанками). Наличие перекрытия помогает не потерять информацию, пересекающую границы чанков. Эта техника широко применяется в RAG-подходах: «вся база знаний «нарезается» на небольшие куски текста, так называемые chunks (чанки)».
* **Векторизация (эмбеддинг):** Каждый чанк преобразуется в числовой вектор с помощью модели эмбеддингов. В нашем эксперименте используется модель **BAAI/bge-multilingual-gemma2** (через API Nebius AI), которая выдаёт эмбеддинги смыслового содержания текста.
* **Индексация и поиск:** Эмбеддинги всех чанков сохраняются в индекс **FAISS** (IndexFlatL2) для быстрого поиска ближайших векторов. FAISS – это библиотека для эффективного поиска и кластеризации векторов сходства, специально разработанная для таких задач (расширение «Facebook AI Similarity Search»). Поисковая система находит топ-K чанков с наибольшей близостью (минимальным евклидовым расстоянием) к вектору запроса.
* **Стратегии RAG:** Мы сравниваем три стратегии извлечения контекста:
  + **Simple RAG (простая стратегия):** напрямую ищем top\_k наиболее релевантных чанков по исходному запросу.
  + **Query Rewrite RAG (RAG с переписыванием запроса):** сначала LLM переформулирует исходный вопрос (для улучшения релевантности), а затем по новому запросу извлекаются top\_k чанков.
  + **Rerank RAG (повторное ранжирование, симуляция):** сначала извлекаются более широкое число фрагментов (top\_k \* multiplier), а затем выбираются лучшие top\_k. В реальных системах для этого может использоваться специальная модель ранжирования, мы же имитируем этот этап выборкой лучших результатов из увеличенного пула.
* **Генерация ответа:** Система формирует запрос к языковой модели, объединяя извлечённые чанки в контекст и исходный вопрос. Затем этот запрос передаётся в LLM (в нашем случае модель **DeepSeek-V3** через API Nebius AI) для генерации ответа. Для генерации используется низкая температура (0.1) и достаточный лимит токенов (до 400), чтобы получить полные структурированные ответы.
* **Оценка качества ответа:** Каждый сгенерированный ответ оценивается по трём метрикам:
  + *Достоверность (Faithfulness):* как фактически верен ответ относительно предоставленного контекста (содержит ли он только информацию из корпуса без фактических ошибок). Мы используем LLM-оценщик с соответствующим промптом, который возвращает оценку в диапазоне [0,1] (где 1 – полностью соответствующий фактам).
  + *Релевантность (Relevancy):* насколько сгенерированный ответ соответствует изначальному вопросу. Снова применяется LLM-оценщик, проверяющий, какой процент ответа релевантен вопросу.
  + *Семантическое сходство (Semantic Similarity):* косинусное сходство эмбеддингов сгенерированного ответа и эталонного. Косинусное сходство измеряет угол между векторами: 1 означает полное совпадение смыслов, 0 – полное различие. Мы вычисляем эмбеддинги обоих текстов и считаем косинус.  
    Баллы этих трёх метрик усредняются для получения интегрального показателя качества каждого ответа.

Таким образом, мы строим экспериментальную RAG-систему и последовательно проверяем, как меняются её выходные ответы и метрики при изменении параметров нарезки и стратегии извлечения. Все компоненты реализованы на Python с использованием библиотек openai (для доступа к API Nebius AI), numpy и faiss (поиск сходных векторов), pandas (анализ результатов) и scikit-learn (косинусное сходство).

**3. Эксперимент**

Для иллюстрации мы выбрали узкую предметную область – возобновляемые источники энергии. В качестве **базы знаний** сформировали пять текстовых документов, посвящённых разным технологиям: солнечной энергии, ветру, гидроэнергии, геотермальной и биомассе. Каждый документ – это обзорная заметка, содержащая ключевые факты. Например, документ о солнечной энергии описывает, что фотоэлектрические панели (PV) напрямую преобразуют солнечный свет в электричество, но их выработка зависит от времени суток и погоды. Документ о гидроэнергии объясняет, что крупные гидроэлектростанции надёжны и дают электроэнергию круглосуточно, но строительством плотин наносится ущерб экосистемам.

В качестве **запроса** мы задали системе: *«Compare the consistency and environmental impact of solar power versus hydropower.»* – сравните надёжность (устойчивость выработки) и экологические последствия солнечной и гидроэнергии. Эталонный «истинный» ответ на этот вопрос сформулирован вручную на основе только текста из корпуса: «Солнечная энергия непостоянна из-за погоды и смены дня/ночи, для стабильности требует накопителей (например, батарей). Гидроэнергия, наоборот, обычно надёжна, но строительство больших плотин существенно влияет на экосистемы и население, чего не происходит у солнечных панелей, где основной эффект – использование земли». Этот ответ использован как справочный для оценки Faithfulness и вычисления семантического сходства.

Реализация эксперимента в ноутбуке проходила в несколько этапов. Сначала настраивались параметры: тестируемые значения **размеров чанка** {150, 250} слов, **перекрытия** {30, 50} слов, **числа извлекаемых фрагментов** top\_k равных {3, 5}. Также задавались параметры генерации (температура=0.1, max токенов=400) и используемые модели Nebius (эмбеддинг: Gemma2, генерация и оценка: DeepSeek-V3). Для стратегии Rerank установлен коэффициент выборки RERANK\_MULTIPLIER=5 (то есть сначала берутся 5·top\_k фрагментов).

Затем для каждого сочетания параметров (chunk\_size, chunk\_overlap, top\_k) выполнялся следующий цикл:

1. **Обработка документов:** если поменялись параметры нарезки, всю базу знаний разрезали на чанки новой длины функцией chunk\_text(text, chunk\_size, chunk\_overlap). Функция создавала пересекающиеся части заданного размера. Далее полученные чанки переведены в эмбеддинги (пакетно для эффективности) с помощью модели Nebius. С этими векторами строился индекс FAISS (структура IndexFlatL2) для быстрого поиска ближайших векторов. При повторном тестировании таких же параметров использовались закэшированные эмбеддинги и индекс.
2. **Извлечение по стратегии:** в зависимости от стратегии формировался запрос к индексу:
   * *Simple RAG:* сразу отправляем изначальный вопрос, получаем top\_k самых близких чанков.
   * *Query Rewrite RAG:* сначала задаём LLM задачу переформулировать вопрос, по переформулированному запросу ищем top\_k чанков.
   * *Rerank RAG:* ищем top\_k \* RERANK\_MULTIPLIER фрагментов по исходному запросу, затем выбираем из них top\_k лучших по расстоянию (имитируя этап повторного ранжирования).
3. **Генерация ответа:** из полученных чанков формируем промпт, включающий тексты фрагментов и исходный вопрос. Этот промпт передаётся LLM-модели (DeepSeek-V3) для генерации ответа.
4. **Оценка:** сгенерированный ответ подаётся на оценку. Сначала используется LLM-оценщик с промптом Faithfulness, дающий оценку достоверности (насколько ответ базируется на фактографической информации из контекста). Затем с помощью промпта Relevancy оценивается, насколько ответ отвечает на сам вопрос. Параллельно вычисляется косинусное сходство между эмбеддингом ответа и эталонного ответа. Значения faithfulness и relevancy обычно в диапазоне [0,1], как и cosine\_similarity (в нашем контексте – [0,1] для неотрицательных эмбеддингов). Наконец, расчёт среднего балла как простое среднее трёх метрик.
5. **Сохранение результатов:** для каждой попытки сохраняются параметры (chunk\_size, chunk\_overlap, top\_k, стратегия), идентификаторы выбранных чанков, переформулированный вопрос (если был), сгенерированный ответ, три отдельные оценки и их среднее, а также время выполнения.

После завершения всех запусков результаты были собраны в структуру all\_results и представлены в виде DataFrame pandas для дальнейшего анализа.

**4. Результаты и обсуждение**

Эксперимент дал 24 набора результатов (3 стратегии × 8 комбинаций параметров). Ключевым оказалось сравнение усреднённых оценок. Наилучшее качество (макс. средний балл ≈ 0,8976) показала стратегия Rerank RAG (симуляция) при параметрах chunk\_size=150, overlap=30, top\_k=3. Второе место занял также Rerank с параметрами (250,50,5) (≈0,8972). В целом Rerank-метод превосходил Simple и Query Rewrite (см. таблицу топ-результатов). Например, Simple RAG при (250,30,3) дал средний балл ≈0,8953. Query Rewrite показал результат около 0,8944 на лучших конфигурациях. Разница между лучшими настройками была невелика (<0,5%), однако последовательно наибольшие значения давал Rerank. Отметим, что оценки dostovernosti и релевантности во всех запусках достигали максимума (0,9 и 1,0 соответственно) – модель стабильно считала ответы фактически верными и полностью отвечающими на вопрос. Главное различие в качестве лежало в семантическом сходстве к эталонному ответу: в лучшем случае оно было ≈0,793, в худшем – ≈0,778 (таблица результатов 10 лучших конфигураций содержит эти значения). Это говорит о том, что все варианты выдавали ответ, близкий по смыслу к референсу, но Rerank позволял немного лучше передать ключевые факты. Сгенерированные ответы подтвердили правильность основных наблюдений. Например, лучшая конфигурация (Rerank, 150,30,3) дала ответ вида:

“Solar power and hydropower differ significantly in consistency and environmental impact:

Consistency: Hydropower is highly reliable (continuous 24/7 power as long as water flows), whereas Solar power is inconsistent due to weather/daylight and requires storage (batteries) for stability.

Environmental Impact: Large hydro dams cause significant ecosystem and community disruption, while solar panels mainly impact land use.”

Ответ логично разбит на пункты Consistency и Impact, отражает основную информацию: гидроэнергия надёжна, но имеет экологические риски, в то время как солнечная энергия зависит от погоды и нуждается в аккумуляторах, но менее вредна для природы. Аналогичные ответы получили и другие конфигурации, лишь различающиеся формулировками. Например, при Query Rewrite RAG модель акцентировала переформулированный запрос, но итоговый ответ был сопоставим. Время выполнения разных стратегий также учтено. Симулированное повторное ранжирование требует дополнительного поиска, поэтому его итоги занимали чуть больше времени (~7–10 сек), в то время как Simple RAG работал немного быстрее (~5–7 сек). Разница не критична для небольшого корпуса, но при масштабировании может повлиять на задержку при ответе. Ошибки и ограничения. Стоит отметить, что все оценки были очень высокими – возможно, из-за небольшой и чёткой обучающей области LLM-оценщик не находил некорректностей. В реальных задачах метрики могут быть ниже. Также наши эксперименты использовали одну парадигму вопроса; потребуется проверка на большем числе вопросов и документов. Кроме того, фактическое переранжирование запросов производится отдельно обученной моделью, а мы лишь эмулировали его простой фильтрацией. В дальнейшем целесообразно реализовать полноценный этап ранжирования (например, с помощью BERT-сети или другого классификатора). Тем не менее, полученные результаты ясно показывают: использование RAG даёт точные ответы, а стратегия с дополнительной фильтрацией (Rerank) даёт небольшое преимущество по совпадению с эталоном.

**5. Стратегии внедрения**

Полученные результаты демонстрируют, что RAG-подход может быть эффективно применён на практике для систем ответов на вопросы и ассистентов. Несколько рекомендаций по интеграции RAG в прикладные задачи:

* **Выбор платформы и моделей:** В эксперименте использован API Nebius AI (модель DeepSeek-V3, Gemma2), однако аналогичные результаты можно получить с другими LLM и эмбеддинговыми моделями (например, OpenAI GPT-4 + текст-эмбеддинги GPT, или локальные аналоги). Главное – обеспечить возможность векторного поиска по базе знаний и генерации текста.
* **Подготовка базы знаний:** Для RAG важно иметь актуальную и релевантную коллекцию документов. Наш эксперимент показал, как нарезка на чанки и индексация через Faiss ускоряют поиск. На практике следует поддерживать базу знаний: добавлять новые документы, обновлять старые. Так как модель не дообучается, свежие данные быстро интегрируются в систему.
* **Настройки параметров:** на основании эксперимента можно рекомендовать умеренный размер чанка (≈150 слов) и не слишком большое top\_k (3–5). Если вычислительных ресурсов много, стратегия с повторным ранжированием оправдана для повышенной точности. В противном случае даже *Simple RAG* показал хороший результат. Переформулирование запроса (Query Rewrite) следует использовать, если входной запрос может быть неоднозначен – в нашем случае эффект был невелик.
* **Интеграция в приложения:** RAG-процесс можно встроить в сервисы через готовые библиотеки: генерация эмбеддингов и поиск может выполняться в фоновом режиме (векторные БД вроде Pinecone, Milvus, Elastic Vector Search и др.). Затем фронтенд-система передаёт запрос, а сервер собирает документы, генерирует ответ LLM и возвращает конечный текст. Особое внимание стоит уделять безопасности: проверять источники и избегать рисков «галлюцинаций». Наша методика оценки (LLM-eval) может использоваться и в рабочей системе для мониторинга качества.
* **Масштабирование и расширение:** Результаты показали устойчивость подхода на небольшой базе. В реальных применениях RAG часто применяется к гораздо более объёмным данным (корпоративные базы знаний, веб-документация и т. д.). Для таких случаев целесообразно: (a) использовать более мощные индексы (ELK/Weaviate + плотные векторы), (b) проводить стемминг и лемматизацию при поиске, (c) внедрять механизмы отбора пользователя (feedback loops), (d) кэшировать запросы и ответы. В дополнение можно применять RAG-цепочки (multi-hop retrieval) и специальные LLM-инструкции для улучшения релевантности ответов.

Таким образом, навыки, полученные в эксперименте, могут лечь в основу реального приложения. Например, для корпоративного чата-бота можно рекомендовать именно RAG с дополнением фактической базы, а из тестирования – использовать небольшие чанки (100–200 слов) и настраивать top\_k в зависимости от объёма. В итоге система ответов будет давать точные и актуальные ответы без дорогостоящего дообучения модели.

**6. Заключение**

В работе представлен подробный эксперимент по настройке и оценке RAG-системы для генерации ответов на вопросы. Мы построили пайплайн с разбиением корпуса на чанки, эмбеддингом, поиском через Faiss, генерацией ответов LLM и их оценкой по метрикам Faithfulness, Relevancy и семантическому сходству. В результате показано, что различные стратегии RAG дают схожие по качеству ответы, при этом повторное ранжирование демонстрирует небольшое преимущество. Сгенерированные ответы оказались высоко релевантными и достоверными по отношению к эталонному, что подтверждает работоспособность подхода.

Основной вклад работы – системная методика испытаний параметров RAG и автоматическая оценка качества ответов. Подобный подход может быть применён в учебных и практических задачах NLP: например, при создании ассистентов, Q&A-систем или образовательных платформ. В дальнейшем планируется масштабировать эксперимент (больший объём данных, разные домены, различные языки) и интегрировать полноценную модель повторного ранжирования для более глубокого анализа.

**Список литературы:**

1. Маскляров Д. *RAG (Retrieval Augmented Generation) — простое и понятное объяснение*. Habr.com, 2023. [habr.com](https://habr.com/ru/articles/779526/#:~:text=RAG%20,%D0%B1%D0%BE%D0%BB%D0%B5%D0%B5%20%D0%BF%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9%20%D0%B8%20%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BE%D1%82%D0%B2%D0%B5%D1%82)[habr.com](https://habr.com/ru/articles/779526/#:~:text=1,%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%20100%C2%A0%D0%B4%D0%BE%201000%C2%A0%D1%81%D0%BB%D0%BE%D0%B2).
2. *LLM Evaluation Metrics: The Ultimate LLM Evaluation Guide*. Confident AI, 2025 (статья блога). [confident-ai.com](https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation#:~:text=Faithfulness%20is%20a%20RAG%20metric,use%20for%20the%20faithfulness%20metric)[confident-ai.com](https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation#:~:text=Answer%20relevancy%20is%20a%20RAG,the%20total%20number%20of%20sentences).