Аверина Дарья Александровна
студент, Федеральное государственное бюджетное образовательное учреждение высшего образования «Восточно-Сибирский государственный университет технологий и управления», Россия, г. Улан-Удэ

Научный руководитель:
Евдокимова Инга Сергеевна
кандидат технических наук, доцент, Федеральное государственное бюджетное образовательное учреждение высшего образования «Восточно-Сибирский государственный университет технологий и управления», Россия, г. Улан-Удэ

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ RETRIEVAL-AUGMENTED GENERATION В ИНТЕЛЛЕКТУАЛЬНЫХ АССИСТЕНТАХ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ

Аннотация: В статье рассматривается технология Retrieval-Augmented Generation (RAG) как основа построения интеллектуальных ассистентов технической поддержки. Описывается принцип работы RAG-конвейера: векторизация документации, поиск релевантных фрагментов и генерация ответа локальной языковой моделью. Анализируется практический опыт реализации подобной системы для компании, занимающейся разработкой программного обеспечения. Рассматриваются особенности формирования векторной базы знаний, выбора модели эмбеддингов и настройки параметров поиска. Показано, что применение RAG-архитектуры позволяет существенно снизить нагрузку на

специалистов технической поддержки при сохранении высокой точности и актуальности ответов.

Ключевые слова: RAG, Retrieval-Augmented Generation, векторные базы данных, ChromaDB, эмбеддинги, языковые модели, техническая поддержка, интеллектуальный ассистент, LangChain, обработка естественного языка.

Введение

Стремительное развитие технологий обработки естественного языка открывает новые возможности для автоматизации процессов технической поддержки пользователей. Компании, предоставляющие программные продукты, ежедневно сталкиваются с большим потоком однотипных обращений: вопросы об установке, настройке, обновлении и лицензировании продуктов. При этом актуальная документация, как правило, уже существует на сайте разработчика, однако пользователи нередко испытывают трудности с её самостоятельным поиском и интерпретацией.

Традиционные чат-боты, основанные на заранее прописанных сценариях (rule-based подход), плохо справляются с вариативностью пользовательских запросов и требуют постоянного ручного обновления. В то же время общие языковые модели (LLM), такие как GPT-4 или Llama, хотя и обладают широкими возможностями, не имеют доступа к специфической документации конкретной компании и могут генерировать фактически неверные ответы — так называемые «галлюцинации».

Решением данной проблемы является архитектура Retrieval-Augmented Generation (RAG), позволяющая сочетать мощь языковых моделей с точностью поиска по актуальной документации. Цель настоящей работы — описать принципы построения RAG-системы для технической поддержки и

проанализировать практический опыт её реализации в рамках конкурсного проекта.

1. Теоретические основы технологии RAG

Retrieval-Augmented Generation (RAG) — это архитектурный паттерн в области искусственного интеллекта, при котором языковая модель перед генерацией ответа обращается к внешней базе знаний для извлечения релевантного контекста [1]. Данный подход был предложен исследователями Facebook AI в 2020 году и с тех пор получил широкое распространение в прикладных системах обработки языка.

Ключевое отличие RAG от чистой генерации (без извлечения) состоит в следующем: вместо того чтобы полагаться исключительно на знания, «запомненные» моделью в процессе обучения, система динамически извлекает актуальную информацию из внешнего хранилища. Это позволяет устранить проблему устаревания знаний и значительно снизить вероятность галлюцинаций [2].

RAG-конвейер включает два основных этапа:

— этап индексирования (offline): документы разбиваются на фрагменты (чанки), преобразуются в числовые векторы (эмбеддинги) и сохраняются в векторной базе данных;

— этап обработки запроса (online): входной вопрос также векторизуется, система выполняет поиск наиболее близких по смыслу фрагментов и передаёт их вместе с вопросом в языковую модель для генерации итогового ответа.

Метрикой близости при поиске наиболее часто выступает косинусное расстояние между векторами — числовая мера семантического сходства двух текстовых фрагментов. Формально косинусное сходство определяется как:

$$\cos(\theta) = (A \cdot B) / (||A|| \cdot ||B||),$$

где A и B — векторные представления двух фрагментов текста. Значение, близкое к 1, указывает на высокую семантическую схожесть, близкое к 0 — на несвязанность текстов.

2. Формирование векторной базы знаний

В разработанной системе источником знаний служит документация, размещённая на корпоративном сайте компании. Для её автоматического извлечения был реализован веб-краулер на основе библиотеки BeautifulSoup, который обходит все страницы раздела документации, извлекает текстовый контент и преобразует его в формат Markdown. Такой формат позволяет сохранить структуру материала (заголовки, списки, таблицы) при дальнейшей обработке.

После сбора документации весь массив текста разбивается на фрагменты фиксированного размера (чанкинг). Размер чанка является важным гиперпараметром системы: слишком маленькие фрагменты теряют контекст, слишком большие — снижают точность поиска и перегружают контекстное окно языковой модели. В реализованной системе был выбран размер чанка 500 токенов с перекрытием 50 токенов между соседними фрагментами для сохранения контекстной связности.

Для преобразования текстовых фрагментов в эмбеддинги использовалась модель nomic-embed-text, запускаемая локально через платформу Ollama. Данная модель генерирует векторы размерностью 768, хорошо подходит для работы с текстами на русском языке и не требует отправки данных во внешние сервисы.

Полученные векторы сохраняются в базе данных ChromaDB — специализированном хранилище, оптимизированном для эффективного поиска по векторным представлениям. ChromaDB поддерживает хранение как самих

векторов, так и исходных текстовых фрагментов и метаданных (URL источника, заголовок страницы), что позволяет не только находить релевантные фрагменты, но и предоставлять пользователю ссылки на первоисточник.

3. RAG-конвейер обработки пользовательских запросов

При поступлении вопроса от пользователя система выполняет следующую последовательность шагов, реализованную с помощью библиотеки LangChain:

1. Векторизация запроса. Текст вопроса преобразуется в числовой вектор той же моделью эмбеддингов, которая использовалась при индексировании базы знаний.
2. Поиск релевантных фрагментов. В ChromaDB выполняется поиск k наиболее близких по косинусному расстоянию фрагментов (в данной реализации k = 3). Полученные фрагменты формируют контекст для языковой модели.
3. Формирование промпта. Контекст и исходный вопрос объединяются в структурированный промпт, который инструктирует языковую модель отвечать строго на основе предоставленных материалов и не придумывать информацию.
4. Генерация ответа. Промпт передаётся локальной языковой модели Mistral 7B, запущенной через Ollama. Модель генерирует ответ на русском языке с опорой на контекст.
5. Возврат ответа. Система возвращает сгенерированный ответ вместе со ссылками на источники, использованные при поиске.

Особое внимание было уделено обработке случаев, когда документация не содержит ответа на вопрос пользователя. В таких ситуациях система формирует флаг эскалации и направляет запрос специалисту технической поддержки вместо генерации потенциально неточного ответа. Подобный механизм позволяет поддерживать высокое качество ответов и одновременно формировать базу для дальнейшего самообучения системы.

4. Практические результаты и оценка эффективности

Разработанная RAG-система была протестирована на репрезентативной выборке из 50 вопросов, типичных для технической поддержки компании. Тестовые запросы охватывали темы установки программного обеспечения, настройки параметров, устранения типичных ошибок и лицензирования продуктов.

По результатам тестирования, система корректно ответила на 82% запросов, то есть предоставила точный и содержательный ответ с корректными ссылками на источник. Для 11% запросов система справедливо инициировала эскалацию, признав отсутствие релевантной информации в базе знаний. Лишь в 7% случаев ответ оказался неполным или содержал незначительные неточности. Таким образом, точность системы (при исключении случаев эскалации) составила около 92%.

Важно отметить, что все вычисления выполняются локально на сервере компании: данные пользователей не передаются в облачные сервисы или внешние API. Это обеспечивает соответствие требованиям информационной безопасности и позволяет использовать систему для обработки конфиденциальной информации. Среднее время ответа составило 4–7 секунд в зависимости от длины генерируемого текста, что является приемлемым для формата текстового чата.

Заключение

Проведённое исследование показало, что технология RAG является эффективным инструментом для построения интеллектуальных ассистентов технической поддержки. Ключевыми преимуществами данного подхода являются: высокая точность ответов за счёт опоры на актуальную документацию; прозрачность ответов благодаря ссылкам на первоисточники; возможность полностью локального развёртывания без зависимости от внешних API; лёгкость обновления базы знаний без повторного обучения модели.

Практическая реализация системы в рамках конкурсного проекта подтвердила работоспособность архитектурных решений. Полученный опыт свидетельствует о перспективности применения RAG-подхода в корпоративных системах поддержки пользователей. Дальнейшие направления развития включают тонкую настройку (fine-tuning) модели эмбеддингов на специфических данных компании, внедрение механизмов оценки качества ответов и расширение базы знаний за счёт интеграции с CRM-системами.

Список литературы

1. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. — 2020. — Vol. 33. — P. 9459–9474.
2. Gao Y., Xiong Y., Gao X. et al. Retrieval-Augmented Generation for Large Language Models: A Survey // arXiv preprint arXiv:2312.10997. — 2023. — 45 p.
3. Karpukhin V., Oguz B., Min S. et al. Dense Passage Retrieval for Open-Domain Question Answering // Proceedings of EMNLP. — 2020. — P. 6769–6781.
4. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // Proceedings of EMNLP-IJCNLP. — 2019. — P. 3982–3992.
5. Trummer I. From BERT to GPT-3 Codex: Harnessing the Potential of Very Large Language Models for Data Management // Proceedings of the VLDB Endowment. — 2022. — Vol. 15, № 12. — P. 3770–3773.
6. Chroma: The AI-native open-source embedding database [Электронный ресурс]. — Режим доступа: https://www.trychroma.com/ (дата обращения: 10.04.2026).
7. LangChain Documentation [Электронный ресурс]. — Режим доступа: https://docs.langchain.com/ (дата обращения: 10.04.2026).