УДК 004

# РАЗРАБОТКА ИИ АССИСТЕНТА С ПАРАДИГМОЙ RAG

Снегирев З.В. , студент гр. ПСб-251, I курс
Научный руководитель: Бабарыкин В.О. Ассистент кафедры ПИТ
Кузбасский государственный технический университет
имени Т.Ф. Горбачева, г. Кемерово

Большие языковые модели (англ. Large Language Models, LLM) представляют собой класс моделей искусственного интеллекта, обученных на огромных объемах открытых текстовых данных. В последние годы данные технологии стали одним из ключевых достижений XXI века, оказав существенное влияние на различные сферы человеческой деятельности, включая повседневную жизнь, бизнес-процессы и автоматизацию.

Несмотря на высокую эффективность, подобные модели обладают рядом ограничений. Поскольку обучение осуществляется преимущественно на открытых и общедоступных данных, модели могут воспроизводить неточную или устаревшую информацию. Более того, в процессе генерации ответов возможно возникновение так называемых «галлюцинаций», при которых модель формирует правдоподобные, но фактически неверные утверждения.

Особую значимость данная проблема приобретает в контексте использования искусственного интеллекта в бизнесе, где требуется работа с внутренними, конфиденциальными или специализированными данными, недоступными для публичного обучения. Передача таких данных сторонним поставщикам ИИ-решений зачастую невозможна по причинам безопасности и конфиденциальности.

В настоящее время существуют два основных подхода к адаптации языковых моделей под задачи работы с локальными данными: дообучение модели (fine-tuning) и генерация с дополненной выборкой (Retrieval-Augmented Generation, RAG).

Дообучение (fine-tuning) предполагает дополнительное обучение модели на специализированных наборах данных. Данный подход позволяет существенно повысить точность ответов в узкой предметной области, однако требует значительных вычислительных ресурсов и финансовых затрат, что ограничивает его доступность для малого и среднего бизнеса.

Альтернативным и более доступным подходом является генерация с дополненной выборкой (RAG). В рамках данного метода модель формирует ответы, опираясь не только на внутренние параметры, но и на внешние источники данных, полученные в процессе поиска, например, в корпоративных базах знаний или через интернет. Это позволяет обеспечить актуальность, точность и

релевантность генерируемой информации без необходимости полного дообучения модели.

Далее рассмотрим ключевые этапы работы RAG и их практическую реализацию.

1. Индексация. Это этап подготовки базы знаний, к которой будет обращаться модель. Исходные документы (.txt, .docx, .md, .pdf) сначала разбиваются на небольшие фрагменты - чанки. Затем каждый фрагмент преобразуется в векторное представление (embedding) с помощью специализированной модели. Полученные эмбединги сохраняются в векторной базе данных. Векторная базы данных — специализированная система поиска и хранения информации, которая организует данные в виде многомерных векторов. После этого база знаний готова к использованию.
2. Извлечение. На этом этапе система получает пользовательский запрос, преобразует его в вектор и сравнивает с уже проиндексированными фрагментами. Для этого используется та же модель эмбеддингов, что и на этапе подготовки данных. Затем выполняется поиск наиболее релевантных чанков в векторной базе данных на основе метрики сходства между вектором запроса и векторами сохраненных фрагментов.
3. Генерация. Финальный этап начинается с формирования промта для модели. В него передаются вопрос пользователя и релевантный контекст, найденный на предыдущем этапе. При этом важно ограничить модель использованием только предоставленных данных, без опоры на собственные знания. Например: «Если ответ есть в контексте - дай ответ на вопрос на основе контекста. Если ответа нет в контексте - скажи "Нет информации"». Такой подход снижает риск галлюцинаций и повышает точность ответа. После этого LLM, формирует итоговый ответ на основе переданного контекста. На выходе пользователь получает релевантную и обоснованную информацию.

Целью работы являлась разработка доступного и эффективного интерфейса взаимодействия с интеллектуальным ассистентом, функционирующим в рамках парадигмы Retrieval-Augmented Generation (RAG). Ключевым требованием выступала реализация системы на базе локальных open-source моделей, что обусловлено необходимостью обеспечения конфиденциальности данных и исключения их передачи сторонним корпоративным сервисам.

В качестве основного языка разработки был выбран Python, что обусловлено его статусом стандарта в области машинного обучения и обработки нейронных сетей, а также наличием зрелой экосистемы специализированных библиотек.

Для организации векторного поиска и хранения эмбеддингов была использована база данных Chroma. Данный инструмент с открытым исходным кодом оптимально подходит для задач среднего масштаба, обеспечивая баланс между производительностью и простотой интеграции.

Запуск и управление языковыми моделями на локальной машине реализованы с использованием платформы Ollama, предоставляющей унифицированный и бесплатный механизм работы с широким спектром открытых LLM.

В качестве основного фреймворка для построения логики приложения была выбрана библиотека LangChain, благодаря ее гибкости, нативной поддержке RAG-подхода и встроенной интеграции с локальными моделями, включая взаимодействие с Ollama.

Таким образом, итоговый технологический стек включает: Python, Chroma, Ollama и LangChain, что позволяет обеспечить автономность, безопасность и расширяемость разработанного решения.

Следующим этапом работы стала практическая реализация описанной архитектуры в виде CLI-приложения, обеспечивающего взаимодействия пользователя ассистентом. Выбор CLI-интерфейса обусловлен простотой его легковесностью, кроссплатформенностью и удобством интеграции в рабочие процессы.

Разработанное приложение реализует полный цикл обработки пользовательского запроса - от момента его получения до генерации ответа с учетом извлеченного контекста. При поступлении запроса выполняется его векторизация с использованием модели Qwen3-Embedding, выбор которой обусловлен сравнительно небольшим размером (около 4 ГБ) при высоких показателях качества в общедоступных бенчмарках. После преобразования запроса в эмбеддинг осуществляется поиск релевантных фрагментов (чанков) в векторной базе данных.

Извлеченный контекст передается в языковую модель, где используется для генерации итогового ответа. В качестве основной LLM была выбрана модель Qwen 3.5, что связано с ее оптимальным соотношением размера и производительности (порядка 6 ГБ при ~9 млрд параметров), а также высокой способностью следовать пользовательским инструкциям, что является критически важным для задач, основанных на парадигме RAG.

Особое внимание было уделено модульности архитектуры приложения. Каждый этап от подготовки данных, до поиска релевантных фрагментов и генерации ответа реализованы независимо, что обеспечивает их заменяемость и гибкость.

В качестве экспериментальных данных, были выбраны обучающие материалы по программированию. Такой выбор обусловлен их

структурированностью, разнообразием и достаточно четкая семантическая нагрузка, что делает их удобным тестовым набором.

Тестирование включало проверку качества подбора релевантного контекста, а также корректность и полноту генерируемых ответов. Особое внимание уделялось способности модели не полагаться на собственные знания и отказываться от ответа на вопрос если подходящий контекст не был найден.

Полученные результаты демонстрируют, что предложенный подход позволяет эффективно реализовать локального интеллектуального ассистента, способного работать с пользовательскими данными без необходимости обращения к внешним сервисам. Это подтверждает применимость выбранного стека технологий для создания автономных и безопасных решений в области обработки естественного языка.

Пример вывода программы:

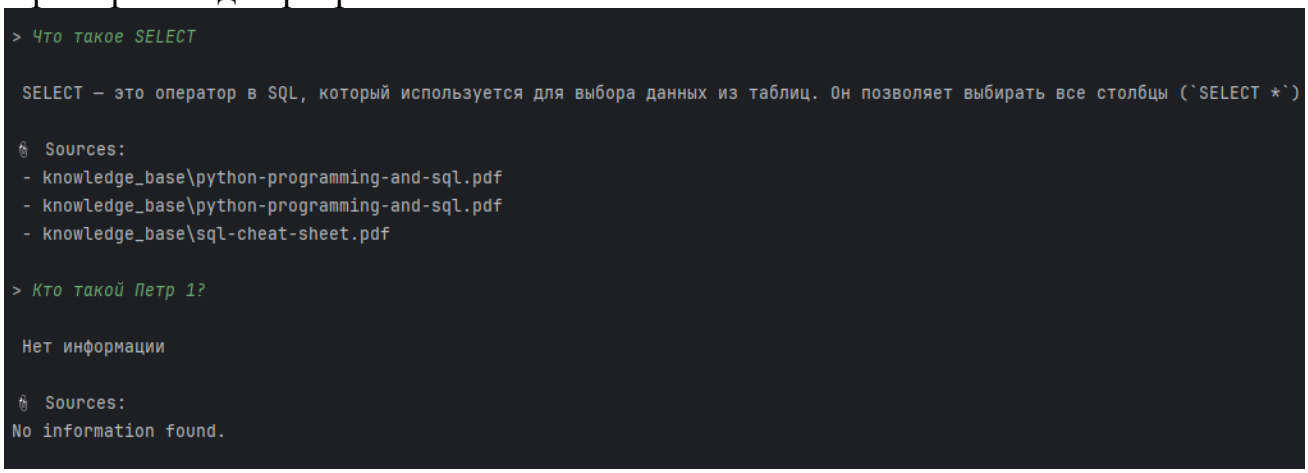


Рис. 1

Список литературы:

1. Сайт «LangChain Docs» [Электронный ресурс]. - Режим доступа: docs.langchain.com, свободный
2. Сайт «TryChroma» [Электронный ресурс]. - Режим доступа: docs.trychroma.com/docs/overview/introduction, свободный
3. Официальный сайт языка Python [Электорнный ресурс]. — Режим доступа: https://www.python.org, свободный