Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

КУРСОВАЯ РАБОТА

Проект на тему "Разработка чат-бота для абитуриентов"

Выполнил студент группы 244, 2 курса,
Толмаков Арсений Александрович

Руководитель КР:
Андреева Дарья Александровна

Москва 2026

Содержание

1 Введение

Разработка чат-ботов для университета становится актуальной задачей в периоды пиковой нагрузки на службу поддержки университета. Большинство вопросов абитуриентов, студентов и остальных пользователей носит повторяющийся характер, и ответами на эти вопросы часто являются правила, требования и прочая информация, которая уже зафиксирована в нормативных документах НИУ ВШЭ.

Таким образом возникает возможность автоматизировать процесс взаимодействия со студентами и абитуриентами и снизить нагрузку на операторов службы поддержки: удобный чат-бот с понятным для обычного пользователя интерфейсом, и, что самое главное, с проверяемостью ответа, то есть пользователь будет получать ответ со ссылками на конкретный пункт.

Сам бот не будет являться полноценным собеседником – на него будут наложены ограничения для упрощения взаимодействия с пользователем, такие как: отсутствие диалогов на отвлеченные темы, отсутствие собственного мнения о сотрудниках университета и каких-либо мероприятий/событий, связанных с ним.

2 Цель и задачи работы

Цель работы

Разработать Telegram-бота для снижения нагрузки на службу поддержки НИУ ВШЭ в периоды повышенного спроса (периоды поступления и сессии). Бот будет представлять собой справочный сервис по документу ПОПАТКУС: поиск релевантной информации с указанием конкретного пункта и формирование пояснения по ней. В случае, если информация по запросу не найдена, бот будет давать контактную информацию службы поддержки.

Задачи работы

1. Подготовить заранее типовые запросы и сценарии
2. Спроектировать структуру работы бота: интерфейс, обработка запроса, модуль семантического поиска по документу и формирование ответа
3. Разработать механизм извлечения данных из ПОПАТКУС: чтение документа, его разбиение на фрагменты и их привязка к пунктам самого документа
4. Создать модуль LLM для разъяснения найденной информации и опорой в виде ссылки на источник
5. Добавить инструкцию для ошибочных запросов – разъяснение, предложения по исправлению и сообщение об отсутствии информации
6. Обеспечить фильтрацию ответа бота – отсутствие сбора личной информации пользователя, отсутствие диалога на отвлеченные темы, отсутствие какого-либо мнения о внутренних сотрудниках, соблюдение политик университета
7. Интеграция: связать Telegram-интерфейс, систему поиска информации по документу и генерацию ответа LLM в единую систему обработки запросов.
8. Тестирование работы бота: формирование 50 тестовых вопросов с заранее размеченными эталонными пунктами для рассчета метрик
9. Добавить возможность оставить отзыв на ответ бота для будущих исправлений

3 Актуальность

Основным аргументом в пользу практической актуальности создания чат-бота в Telegram является снижение нагрузки на операторов службы поддержки во время набора новых студентов и периоды сессии. Автоматизация ответов на типовые вопросы значительно снизит время ответа на них и упростит работу службы поддержки.

Если говорить о технической актуальности используемого метода, очень важно учитывать, что ответ на вопрос пользователя должен быть как можно более строгим и при его нахождении его можно будет проверить. Поэтому в данном проекте будет использована архитектура RAG (Retrieval-augmented generation), которая объединяет в себе извлечение релевантных фрагментов из текста документа и формирование ответа языковой моделью на их основе. Такой подход отличается от простой генерации ответа языковой модели тем, что снижает риск возникновения галлюцинаций и додуманной информации. Также повышается надежность работы бота при обновлении документа: нет необходимости в переобучении модели. Это позволит быстрее поддерживать актуальность информации в документе.

Наконец, такой подход позволит проводить измеримое тестирование ответа и сформулировать критерии качества, что поможет улучшить работу бота.

4 Обзор литературы

Рассмотрим существующие методы решения подобных задач.

Основными критериями при выборе метода являются качество понимания текста, написанного пользователем, и достоверность выданной ему информации – её проверяемость.

Один из классических способов поиска информации в документе – лексический поиск. То есть система сопоставляет слова из запроса с документом, с которым она работает, и ранжирует результаты. Самый распространненый метод лексического поиска – BM25 [1]. Его достоинства в простоте, скорости работы, отсутствии необходимости в обучении и понятном результате. Однако у этого метода есть главное ограничение – он опирается на совпадение слов, что значительно усложняет задачу в условиях общения с пользователем, который зачастую использует сокращения и разговорную речь.

Существуют способы улучшить лексический поиск: расширение запроса, учёт синонимов и нормализация текста, то есть подгонка под требуемый формат. Однако эти способы ресурсозатратны и требуют поддержки словарей, и при этом все равно не решают проблему полноценно: пользователь может описать свой запрос более свободным языком, чем этого ожидает программа.

В связи с этими проблемами возник новый способ решения задачи – методы, умеющие представлять текст семантически. Одним из таких методов стал Word2Vec, позволяющий отображать слова в векторном пространстве, где слова, похожие по смыслу, оказываются рядом [2]. Сам способ не решает задачу целиком, однако является важной методологической основой: он значительно расширяет возможности поиска информации по запросу.

Далее рассмотрим ещё один метод семантического поиска - sentence embedding. Довольно популярным подходом к этому методу является SBERT (Sentence Transformers) [3]. С его помощью мы можем получать векторные представления целых предложений и отрывков текста, а затем искать фрагменты, наиболее подходящие запросу пользователя. Также это помогает решить проблему сокращенных слов и разговорных формулировок. Помимо этого, вектора документа можно вычислить заранее, и при запросе пользователя вычислять только вектор запроса и сравнивать с уже заготовленными.

Но и у метода sentence-embedding существуют ограничения: во многих документах пункты построены однообразно, и при построении векторов может возникнуть ситуация, когда система выберет похожий, однако относящийся к другой теме пункт.

В качестве альтернативы существует еще один метод – extractive QA [4]. Система выбирает строгую цитату из документа по запросу. В данном методе снижается вероятность возникновения не относящейся к теме ответа, так как пользователь получит цитату. Но в нашей задаче требуется разъяснение найденной информации, поэтому этот метод используется чаще как подстраховка.

Наконец рассмотрим подробнее используемый нами метод Retrieval-augmented generation [5]. Его преимущество в том, что сам документ хранится отдельно от языковой модели, и она поясняет только ту информацию, что выдал релевантный поиск. Это снижает вероятность возникновения неподтвержденных ответов. При обновлении документа достаточно обновить набор фрагментов, пересчитать embeddings, и переиндексировать базу. На практике это сильно упрощает работу архитектуры.

Еще одним преимуществом, как было упомянуто ранее, является возможность

измеримо тестировать качество ответа: насколько часто верный ответ попадает в топы выдачи поиска, и насколько генерируемый ответ соответствует извлеченному фрагменту.

Таким образом, лучшим из обозреваемых решений является архитектура RAG, как объединение преимуществ семантического поиска и языковой модели.

5 Результаты первых экспериментов

В рамках данного проекта единственным источником информации для бота является документ ПОПАТКУС, что упростит его разработку и тестирование. Будем брать последнюю его версию, которая была принята 26.02.2025 и вступила в силу 01.09.2025.

Перед началом стоит описать базу данных, которая будет использоваться для проведения экспериментов – ChromaDB [6]. Она предоставляет удобную Python API для добавления документов и поиску ближайших соседей в векторном представлении. ChromaDB поддерживает локальный режим работы, что упрощает работу с ней.

В рамках эксперимента ChromaDB была использована для индексации фрагментов ПОПАТКУСа и затем для формирования top-k релевантных фрагментов, чтобы с их помощью генерировать ответ.

Начинать эксперимент необходимо с дробления документа на страницы, для того, чтобы в будущем ссылаться на них при ответе. Далее проведем очистку и нормализацию текста, удалим артефакты извлечения из формата PDF, и элементы, не несущие смысловой нагрузки.

После этих шагов можно начинать сегментацию - текст каждой страницы разбивается на фрагменты фиксированного размера. Каждому такому фрагменту присваиваются метаданные (уникальный идентификатор, тип информации, номер страницы), которые помогут при поиске и формулировке ответа.

Список литературы

[1] S. Robertson и H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond", *Foundations and Trends in Information Retrieval*, т. 3, № 4, с. 333—389, 2009. DOI: 10.1561/1500000019. url: https://www.staff.city.ac.uk/~sbrp622/papers/foundations_bm25_review.pdf.

[2] T. Mikolov, K. Chen, G. Corrado и J. Dean, "Efficient Estimation of Word Representations in Vector Space", *arXiv preprint arXiv:1301.3781*, 2013. url: https://arxiv.org/abs/1301.3781.

[3] N. Reimers и I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", *arXiv preprint arXiv:1908.10084*, 2019. url: https://arxiv.org/abs/1908.10084.

[4] P. Rajpurkar, J. Zhang, K. Lopyrev и P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text", в *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016, с. 2383—2392. DOI: 10.18653/v1/D16-1264. url: https://aclanthology.org/D16-1264.

[5] P. Lewis и др., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", в *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. url: https://arxiv.org/abs/2005.11401.

[6] Chroma, *Chroma: The AI-native open-source embedding database*, https://www.trychroma.com, Accessed: 2026-02-04, 2024.