МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

РАЗРАБОТКАУМНОГО ПОМОЩНИКА NETUTOR

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Смыслова Антона Сергеевича

Научный руководитель:
доцент _____ Е. В. Кудрина
подпись, дата

Зав. кафедрой:
к. ф.-м. н., доцент _____ М. В. Огнева
подпись, дата

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. В условиях стремительного развития технологий искусственного интеллекта всё больше образовательных учреждений стремятся внедрять интеллектуальные системы поддержки, направленные на улучшение взаимодействия между абитуриентами, студентами и администрацией. Современная цифровая среда требует высокой скорости отклика, точности и персонализации информации, что делает автоматизацию коммуникации в образовательных организациях особенно актуальной. С каждым годом растёт количество обращений, связанных с поступлением, расписанием, академическими вопросами и стипендиями, в то время как традиционные формы поддержки — телефонные линии, электронная почта и очные консультации — часто оказываются перегружены и не способны обеспечить должный уровень сервиса.

Всё большую популярность приобретают интеллектуальные сервисы, основанные на больших языковых моделях (Large Language Models, LLM). В отличие от классических сценарных систем, такие решения способны понимать естественный язык, формулировать развёрнутые ответы и вести осмысленный диалог с пользователем. Однако одной из ключевых проблем остаётся обеспечение достоверности и актуальности информации, особенно в образовательной сфере, где ошибки в ответах приводят к недопониманию и снижению доверия к цифровым сервисам. Одним из наиболее перспективных подходов к решению этой задачи является использование архитектуры Retrieval-Augmented Generation (RAG), которая сочетает генеративные возможности LLM с механизмом семантического поиска по внутренним документам организации. Это позволяет сервису не просто формировать текстовый ответ, а опираться на реальные данные, обеспечивая его точность и обоснованность. Актуальность работы обусловлена необходимостью внедрения интеллектуальных технологий в образовательную среду, повышением требований к качеству цифровых сервисов и стремлением учебных заведений оптимизировать внутренние процессы за счёт

автоматизации.

Цель бакалаврской работы – разработать интеллектуального помощника NETutor, обеспечивающего автоматизированный сбор, обработку и предоставление актуальной информации из системы документооборота организации. Работа выполнена по программе «Стартап как диплом»; личная зона ответственности автора — проектирование, программная реализация и экспериментальное тестирование RAG-системы на основе большой языковой модели, на которой основывается функционирование интеллектуального помощника NETutor.

Поставленная цель определила **следующие задачи**:

1. рассмотреть проблемы сбора и обработки информации в организациях;
2. представить обзор существующих RAG-решений и методов обработки информации;
3. описать программные средства и технологии, применяемые при разработке интеллектуальных помощников на основе RAG;
4. разработать и реализовать архитектуру RAG-системы для интеллектуального помощника NETutor;
5. продемонстрировать работу RAG-системы интеллектуального помощника NETutor и оценить эффективность её работы.

Методологические основы разработки интеллектуальных помощников на основе технологии Retrieval-Augmented Generation представлены в работах Lewis, Gao, Vaswani, Devlin, Reimers, Gurevych, Wang, Malkov, Yashunin, Robertson, Nogueira и других.

Теоретическая значимость бакалаврской работы заключается в систематизации подходов к построению RAG-систем, сравнительном анализе существующих интеллектуальных помощников по совокупности технологических критериев, а также в обосновании выбора компонентов конвейера — модели эмбеддингов, векторной базы данных, механизмов гибридного поиска и кросс-энкодерного переранжирования — применительно к задаче обработки русскоязычной документации

организации.

Практическая значимость бакалаврской работы подтверждается разработкой унифицированной RAG-системы NETutor, готовой к полностью локальному (on-premise) развёртыванию без передачи конфиденциальных данных внешним сервисам и пригодной к тиражированию в образовательных, коммерческих и государственных учреждениях. Инновационный характер и практическая ценность решения подтверждены получением гран-при на международном конкурсе стартапов в Республике Беларусь весной 2026 года.

Структура и объём работы. Бакалаврская работа состоит из введения, 2 разделов, заключения, списка использованных источников и 12 приложений. Общий объём работы — 99 страниц, из них 58 страниц — основное содержание, включая 6 рисунков и 15 таблиц, список использованных источников информации — 27 наименований.

Первый раздел «Теоретическая часть» посвящён анализу предметной области, обзору существующих решений и обоснованию применяемого технологического стека. В разделе рассмотрены проблемы сбора и обработки информации в организациях: разрозненность корпоративных хранилищ, закрытость ERP-платформ класса «1С:Университет ПРОФ», преобладание неструктурированных данных (PDF, DOCX, отсканированные изображения), требующих оптического распознавания текста (OCR) и извлечения именованных сущностей (NER), а также правовые ограничения, накладываемые Федеральным законом № 152-ФЗ «О персональных данных». Показано, что ключевым структурным ограничением большинства организаций является отсутствие единого информационного контура, а архитектура RAG рассматривается как современный ответ на эти вызовы.

Проведён сравнительный анализ существующих RAG-решений: коммерческих продуктов Confluence AI и Notion AI, реализующих семантический поиск лишь частично; закрытых SaaS-платформ Guru и

Document360; а также внутренних университетских прототипов НИУ ВШЭ (на основе YandexGPT) и НИЯУ МИФИ («АссистенТИИКС»). Сопоставление по пяти критериям — наличие полноценного RAG-конвейера, поддержка локального развёртывания, гибкость стека, качество обработки русского языка и тиражируемость — выявило технологический разрыв: ни одно из решений не сочетает всех перечисленных свойств, что обосновывает позицию NETutor на рынке.

Каждый из критериев сопоставления обоснован отдельно. Возможность on-premise развёртывания принципиальна для учреждений, работающих с персональными данными студентов и сотрудников, ввиду требований Федерального закона № 152-ФЗ; гибкость архитектурного стека отражает способность оператора самостоятельно выбирать и заменять компоненты конвейера — модель эмбеддингов, векторную базу и языковую модель; качество обработки русскоязычного контента выделено отдельно, поскольку семантический поиск чувствителен к языку обучения модели эмбеддингов; тиражируемость определяет возможность развёртывания в произвольной сторонней организации без участия разработчика. Именно совокупность этих характеристик — полноценная RAG-архитектура, локальное развёртывание, настраиваемый стек, поддержка русского языка и готовность к тиражированию — отличает NETutor от рассмотренных аналогов.

Дополнительно проанализированы проблемы семантической неоднородности данных, когда документы, относящиеся к одной сущности, отличаются по стилю, терминологии и структуре, а также проблема несогласованности и быстрого устаревания сведений, одновременно хранящихся в нескольких системах и обновляемых с разной периодичностью. Показано, что без механизмов отслеживания версий и автоматического обновления индекс быстро теряет достоверность, а существующие подходы — ручная систематизация силами сотрудников, точечная интеграция через API и внедрение систем электронного документооборота (СЭД) — либо не масштабируются, либо не решают задачу интеллектуального поиска по

содержимому документов.

Рассмотрены методы сбора и обработки данных из разнородных источников: краулинг с обходом в ширину (BFS) и в глубину (DFS), парсинг DOM-дерева с помощью CSS-селекторов и XPath-выражений, оптическое распознавание текста (OCR) с предварительной бинаризацией и шумоподавлением, извлечение именованных сущностей (NER) на основе трансформерных моделей класса BERT, интеграция через API и последующая нормализация данных — унификация регистра, устранение дублирующих записей, приведение дат и числовых значений к единому формату — с дальнейшей векторизацией и помещением в единое семантическое пространство.

Описан применяемый стек технологий с обоснованием выбора и рассмотрением альтернатив (таблица сравнения восьми категорий): язык Python — за зрелость экосистемы NLP; векторная база данных Qdrant — за нативную фильтрацию по метаданным, наличие REST API и возможность локального развёртывания; библиотека Sentence Transformers — за локальное вычисление эмбеддингов без обращения к платным внешним API; контейнеризация на основе Docker — за воспроизводимость окружения. В качестве рассмотренных альтернатив проанализированы Pinecone, Weaviate и ChromaDB (для векторного хранилища), Flask и Django REST Framework (для HTTP-интерфейса), Scrapy и BeautifulSoup (для сбора данных). Вывод раздела состоит в том, что комбинация локально развёртываемых компонентов обеспечивает требуемое соответствие требованиям к защите персональных данных и качеству обработки русскоязычного контента.

Второй раздел «Практическая часть» посвящён проектированию, программной реализации и экспериментальной оценке RAG-конвейера, составляющего личную зону ответственности автора. Конвейер последовательно проходит три взаимосвязанных этапа — индексацию, поиск и генерацию. На этапе индексации исходные документы (расписание учебных групп, сведения о факультете и общая информация об университете)

подвергаются семантическому чанкингу, после чего каждый фрагмент преобразуется моделью эмбеддингов intfloat/multilingual-e5-base в плотный вектор размерностью 768 измерений и сохраняется в векторной базе данных Qdrant вместе с метаданными. Для ускорения поиска ближайших соседей Qdrant использует алгоритм HNSW (Hierarchical Navigable Small World), обеспечивающий логарифмическую сложность поиска.

Стратегия семантического чанкинга адаптирована под каждый тип данных. Для расписания каждое отдельное занятие преобразуется в самостоятельный фрагмент из полей записи (дисциплина, тип занятия, преподаватель, аудитория, временной слот), что обеспечивает высокую точность поиска по запросам вида «когда у группы 441 физика». Для документов факультета применяется рекурсивный обход структуры JSON по секциям и подсекциям, для общеуниверситетских данных каждый элемент массива с заголовком и содержимым образует отдельный чанк. Необходимость разбиения обусловлена ограничением модели эмбеддингов на длину входной последовательности (512 токенов) и тем, что вектор длинного документа усредняет разнородные смысловые темы, снижая точность поиска.

На этапе поиска запрос пользователя векторизуется той же моделью, после чего вычисляется косинусное сходство между вектором запроса q и вектором фрагмента d по формуле $\mathrm{sim}(q, d) = (q \cdot d) / (\|q\| \cdot \|d\|)$; значение метрики лежит в диапазоне от −1 до 1, и косинусная мера выбрана за устойчивость к различиям в длине фрагментов. Чисто семантический поиск дополнен лексическим усилением: для каждого кандидата вычисляется коэффициент $\mathrm{keyword\_boost} = 1{,}0 + (\mathrm{matches} / |\mathrm{query\_words}|) \cdot 0{,}3$ в диапазоне от 1,0 до 1,3, а итоговый балл релевантности формируется как взвешенная сумма $\mathrm{score} = 0{,}7 \cdot \mathrm{vector\_score} + 0{,}3 \cdot \mathrm{keyword\_boost}$. Веса 0,7 и 0,3 отражают приоритет семантической близости при сохранении чувствительности к точным лексическим совпадениям, что компенсирует слабую чувствительность эмбеддинговых моделей к редким терминам, аббревиатурам и именам собственным (например, «КНиИТ», «441»).

Дополнительно реализован кросс-энкодерный реранкер на основе модели cross-encoder/ms-marco-MiniLM-L-6-v2 (22 млн параметров, 6 слоёв Transformer), который, в отличие от би-энкодеров, совместно обрабатывает пары «запрос — фрагмент» единой Transformer-архитектурой и тем самым учитывает тонкие семантические зависимости между ними. Реранкер располагается после векторного поиска: Qdrant возвращает расширенный пул из 10 фрагментов, кросс-энкодер переранжирует их, и 5 наиболее релевантных передаются на генерацию. Для снижения накладных расходов реализовано кэширование результатов по стратегии LRU. Хранилище организовано в Qdrant в виде точек (point), объединяющих уникальный идентификатор, вектор и payload с текстом и метаданными; разграничение типов контента обеспечивается фильтрацией по полю type без создания отдельных коллекций, а поиск ближайших соседей выполняется алгоритмом HNSW.

На этапе генерации отобранные фрагменты вместе с управляющей prompt-инструкцией передаются локальной языковой модели семейства Qwen, запускаемой через фреймворк Ollama, что гарантирует полную локальность обработки и соответствие требованиям № 152-ФЗ. Prompt-инструкция строится на принципах контекстной привязанности (ответ формируется исключительно на основе предоставленных фрагментов), явного указания на недостаточность контекста, форматирования ответа и языкового соответствия. Структура запроса к модели включает три блока: системную инструкцию, контекст из топ-5 фрагментов и исходный запрос пользователя, что обеспечивает чёткое разграничение инструкций, данных и пользовательского ввода. Генерация выполняется в потоковом (streaming) режиме с ограничением длины ответа в 512 токенов.

В ходе разработки выполнен ряд оптимизаций, подтверждённых экспериментами. Реализован модуль ConditionalRewriter, распознающий неформальные и сленговые запросы по списку маркеров и регулярному выражению для аббревиатур и переформулирующий их в формальные

поисковые запросы; для этой задачи применена облегчённая модель, что позволило ускорить этап переформулировки в 2,6 раза без потери качества. Проведена оптимизация размера чанков: исходная коллекция из 905 фрагментов с аномальной неравномерностью длины (от 38 до 99 897 символов) преобразована иерархическим алгоритмом повторной нарезки в 1816 фрагментов целевого размера 500 символов с перекрытием 80 символов, что устранило выбросы и повысило качество поиска.

Оптимизация размера чанков дала измеримый эффект: после повторной иерархической нарезки (по границам параграфов, затем предложений и лишь в последнюю очередь по длине) медианная длина фрагмента выросла со 138 до 303 символов, максимальная сократилась с 99 897 до 574 символов, число фрагментов, превышающих 1000 символов, снизилось со 130 до нуля, а средний скоринг поиска повысился с 0,8260 до 0,8315 (без переформулировки) и с 0,8381 до 0,8400 (с переформулировкой). Разделение моделей переформулировки и генерации сократило время этапа переформулировки в 2,6 раза — с 8,1 до 3,1 секунды — без ухудшения качества, подтверждённого экспертной оценкой.

Экспериментально подобраны ключевые параметры конвейера. Для размера выборки реранкера проведён бенчмарк при K = 5, 10, 15 и 20: при K = 5 совпадение топ-5 с базовым векторным поиском составляет около 60 % (реранкер лишь переставляет уже найденные фрагменты), при K = 10 — около 25 % (реранкер выдвигает на первые позиции фрагменты, пропущенные векторным поиском), а дальнейшее увеличение K повышает время обработки без заметного прироста качества; в результате значение K = 10 признано оптимальным. Для условного пропуска реранкинга методом сеточного поиска подобраны пороги (средний скоринг ≥ 0,83 при разбросе ≤ 0,03), при которых около 30 % запросов пропускают реранкер без снижения метрики ROUGE-L (0,840), что экономит порядка 0,15 секунды на каждом третьем запросе. Аналогичный условный механизм применён к переформулировке: проверка наличия сленговых маркеров и аббревиатур

позволяет пропускать около 40 % уже формальных запросов.

Между модулем сбора данных и векторной базой используется стабильный промежуточный формат JSON: скрипт индексации последовательно читает файлы трёх типов, передаёт их в соответствующий модуль семантического чанкинга, векторизует содержимое и загружает готовые точки в Qdrant операцией upsert. Реализованы идемпотентность создания коллекции (повторный запуск индексатора не уничтожает накопленные данные) и дедупликация фрагментов по хешу содержимого, что исключает дублирование при инкрементальных обновлениях; устаревшие фрагменты помечаются специальным признаком и могут фильтроваться при поиске. Такая организация конвейера упрощает замену любого компонента без изменения остальных, поскольку взаимодействие модулей не зависит от их внутренней реализации.

Работа системы продемонстрирована на серии тестовых запросов, иллюстрирующих различные сценарии исполнения конвейера. Формальный запрос «распред базы данн 421» не потребовал переформулировки, а реранкер был пропущен вследствие высокого качества векторного ранжирования (средний скоринг около 0,85 при минимальном разбросе). Неформальный запрос «чё по общагам?» был распознан модулем ConditionalRewriter как сленговый и инициировал этап переформулировки, в ходе которого сгенерировано несколько формальных поисковых вариантов, после чего реранкер скорректировал порядок фрагментов. При ошибке формата ответа модели переформулировки система автоматически использует исходный запрос без изменений, обеспечивая отказоустойчивость конвейера: сбой одного компонента не прерывает обработку запроса в целом.

Производительность системы оценивалась по полному времени обработки запроса и времени появления первого токена на двух конфигурациях — тестовой (CPU) и расчётной (GPU). Доминирующим по времени этапом на CPU является генерация ответа (более 70 % общего времени), что объясняется запуском модели без GPU-ускорения; на

конфигурации с дискретным GPU время генерации сокращается до 5–10 секунд. Для оперативного мониторинга разработан веб-дашборд, отображающий временные характеристики каждого этапа, значения векторных скорингов, количество найденных фрагментов и сгенерированных токенов. Совокупный эффект реализованных решений: полное время ответа составляет 7–13 секунд на конфигурации с GPU, первый токен генерируется через 1–2 секунды, а средняя ROUGE-L достигает значения 0,840.

Полученные результаты и перспективы. В ходе выполнения работы был обоснован полностью локальный технологический стек и реализован RAG-конвейер, включающий модули семантического чанкинга и индексации, гибридного векторно-лексического поиска, условной переформулировки запросов, кросс-энкодерного переранжирования и потоковой генерации ответов. Все компоненты выполнены как независимые модули с чётко определёнными входными и выходными данными, что обеспечивает возможность замены модели эмбеддингов, векторной базы данных или языковой модели без изменения остальных частей системы.

Экспериментальное тестирование подтвердило эффективность принятых решений. Оптимизация размера чанков устранила аномально длинные фрагменты (максимальная длина сокращена с 99 897 до 574 символов) и обеспечила измеримое улучшение качества поиска. Условный механизм переформулировки запросов позволил экономить вычислительные ресурсы на формальных запросах без потери качества, а разделение моделей ускорило этап переформулировки в 2,6 раза. Подобранное значение размера выборки реранкера и пороги условного пропуска обеспечили баланс между точностью ранжирования и временем отклика. Итоговые показатели системы — полное время ответа 7–13 секунд на GPU, генерация первого токена через 1–2 секунды и средняя ROUGE-L на уровне 0,840 — подтверждают применимость решения для интерактивного диалогового сервиса.

Созданная система полностью унифицирована, что позволяет гибко развёртывать её не только в образовательных учреждениях, но и в

коммерческих или государственных организациях без привязки к конкретной ИТ-инфраструктуре. Дальнейшее развитие проекта предполагает регулярное расширение корпуса документов, оптимизацию инференса локальной языковой модели на CPU-архитектурах, а также переход на более крупные нейросетевые модели для повышения качества генерации без изменения базовой архитектуры системы. Таким образом, все поставленные задачи бакалаврской работы выполнены, цель достигнута.

Отдельные части бакалаврской работы были представлены на международном конкурсе стартапов в Республике Беларусь весной 2026 года, где проект интеллектуального помощника NETutor был удостоен гран-при.