МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА RAG-СИСТЕМЫ ДЛЯ ПОИСКА ИНФОРМАЦИИ В ВИДЕОЛЕКЦИЯХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Фединой Ангелины Ильиничны

Научный руководитель
доцент, к. ф.-м. н. _____ Б. А. Филиппов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

Выпускная квалификационная работа посвящена разработке RAG-системы для поиска информации в видеолекциях и формирования кратких ответов на вопросы пользователя по найденным фрагментам транскрипций. Актуальность работы обусловлена тем, что видеолекции являются распространённым форматом образовательных материалов, однако они не всегда удобны для повторного обращения к отдельным частям содержания. Пользователь может помнить, что в лекции объяснялось определённое понятие, рассматривался пример или разбиралась ошибка, но не знать, в каком именно месте записи находится нужный фрагмент. В результате поиск информации часто превращается в просмотр значительной части видеоматериала, что требует времени и снижает удобство работы с учебным курсом.

Одним из способов решения данной задачи является преобразование видеолекций в текстовый вид и последующий поиск по полученным транскрипциям. Однако такой подход имеет ряд особенностей. Транскрипции, полученные с помощью автоматического распознавания речи, могут содержать ошибки, пропуски, неточную пунктуацию и искажения специальных терминов. Кроме того, устная речь отличается от письменного текста: лектор может повторять мысль, использовать неполные предложения, переходить от примера к определению или возвращаться к ранее сказанному. Поэтому нужная пользователю информация может быть распределена по нескольким фрагментам, а запрос пользователя не всегда совпадает с формулировками, использованными в лекции.

Целью работы являлась разработка и оценка RAG-системы, позволяющей искать релевантные фрагменты в транскрипциях видеолекций и формировать по ним ответ на вопрос пользователя.

Для достижения поставленной цели были выполнены следующие задачи:

- подготовлены данные: извлечены аудио из видеолекций, выполнена транскрипция и разбиение текста на фрагменты;
- спроектирована структура хранения лекций, фрагментов транскрипций, полнотекстовых и векторных представлений;
- реализован полнотекстовый, семантический и гибридный поиск;
- реализован RAG-компонент для формирования ответа по найденному контексту;
- разработана серверная часть и пользовательский интерфейс;
- проведена оценка качества поиска и генерации ответов;

— проанализированы основные ошибки системы.

Разработанная система предназначена не только для поиска фрагментов, содержащих совпадения с запросом, но и для формирования связного ответа по материалам лекций. Это отличает её от обычной поисковой системы, которая возвращает только список результатов. В данной работе поиск используется как предварительный этап: сначала система находит фрагменты транскрипций, которые могут содержать ответ, затем передаёт их языковой модели в качестве контекста, после чего модель формирует краткое объяснение. Такой подход позволяет использовать набор видеолекций как базу знаний для вопросно-ответной системы.

Практическая реализация системы начинается с подготовки данных. Исходные материалы представлены видеолекциями, поэтому первым этапом стало извлечение аудиодорожек из видеофайлов, после чего они передавались на автоматическую транскрипцию.

Для автоматического распознавания речи использовалась модель whisper-large-v3-turbo в реализации `mlx-whisper`. Данная реализация позволяет выполнять транскрипцию локально и эффективно использовать ресурсы устройств с Apple Silicon. При запуске модели явно указывался русский язык, поскольку видеолекции были записаны на русском. Результат распознавания сохранялся в формате JSON. В нём содержался не только текст, но и временные метки начала и конца отдельных сегментов. Наличие временных меток важно для всей дальнейшей работы системы, так как найденный фрагмент можно связать с конкретным местом видеозаписи.

После получения транскрипции выполнялось разбиение текста на отдельные фрагменты. Поиск по полной транскрипции лекции был бы неэффективен, так как одна лекция может содержать несколько тем, а большой фрагмент будет включать много лишней информации. В то же время слишком короткие фрагменты могут оказаться недостаточно информативными: в одном фрагменте может находиться только начало объяснения, а в следующем — пример или уточнение. Поэтому в работе был выбран подход, при котором последовательные сегменты транскрипции объединяются в более крупные фрагменты с учётом временных меток и границ предложений.

В реализации использовались ограничения на длительность фрагмента. Минимальная длительность составляла 20 секунд, максимальная — 60 секунд.

Сегменты объединялись до тех пор, пока фрагмент не достигал минимальной длительности и не завершался на границе предложения, либо пока его длительность не превышала максимально допустимое значение. В результате каждый фрагмент содержал текст, время начала и время окончания. Такой способ разбиения позволил сохранить достаточный контекст внутри каждого фрагмента и одновременно избежать передачи языковой модели слишком длинных и разнородных частей лекции.

Полученные фрагменты сохранялись в базу данных. Для хранения данных была спроектирована структура, включающая таблицы `videos` и `video_chunks`. Таблица `videos` содержит сведения об исходных видеолекциях: название лекции, название курса, имя файла и ссылку на источник при наличии. Таблица `video_chunks` хранит отдельные фрагменты транскрипций. Каждый фрагмент связан с соответствующей лекцией через внешний ключ `video_id`.

Для каждого фрагмента сохраняются текстовое содержимое, временные метки, полнотекстовое и векторное представления. Поля `start_time` и `end_time` позволяют определить положение фрагмента внутри видеозаписи. Поле `tsv` предназначено для полнотекстового поиска, а поле `embedding` — для семантического поиска. Размерность векторного представления составляет 1024, так как такая размерность используется выбранной моделью построения эмбеддингов.

В качестве системы управления базой данных использовался PostgreSQL с расширением `pgvector`. PostgreSQL предоставляет средства хранения структурированных данных и встроенный полнотекстовый поиск, а `pgvector` добавляет возможность хранить векторные представления и выполнять поиск ближайших векторов. Благодаря этому оба типа поиска реализованы в рамках одной базы данных. Это упрощает архитектуру системы, так как не требуется отдельно разворачивать специализированное векторное хранилище.

Поисковый компонент системы включает три варианта поиска: полнотекстовый, семантический и гибридный. Полнотекстовый поиск реализован средствами PostgreSQL с использованием типов `tsvector` и `tsquery`. Для каждого фрагмента транскрипции формируется полнотекстовое представление, которое сохраняется в поле `tsv`. Поскольку материалы представлены на русском языке, используется языковая конфигурация `russian`. Она позволяет учитывать особенности русского языка и сопоставлять разные формы одного слова.

Для ускорения полнотекстового поиска по полю `tsv` создаётся GIN-индекс. При обработке запроса используется функция `websearch_to_tsquery`, которая преобразует пользовательский запрос в полнотекстовое поисковое выражение. Такой вариант удобен, потому что пользователь может вводить обычный текстовый вопрос без знания специального синтаксиса. Найденные фрагменты ранжируются с помощью функции `ts_rank_cd`. Чем выше оценка релевантности, тем выше фрагмент располагается в выдаче.

Полнотекстовый поиск хорошо работает в случаях, когда запрос пользователя содержит те же термины, что и транскрипция. Например, если пользователь ищет конкретное понятие, название конструкции языка или устойчивую формулировку, лексическое совпадение позволяет быстро найти нужные фрагменты. Однако такой поиск зависит от совпадения слов. Если пользователь формулирует вопрос иначе, чем лектор, релевантный фрагмент может быть не найден. Дополнительные трудности создают ошибки автоматической транскрипции, из-за которых термины могут быть распознаны неточно.

Для решения этой проблемы был реализован семантический поиск. Он основан не на совпадении слов, а на векторных представлениях текста. Для построения эмбеддингов использовалась модель `intfloat/multilingual-e5-large`. Она поддерживает мультиязычные данные, включая русский язык, и подходит для задач поиска по смысловой близости. Эмбеддинги строятся как для фрагментов транскрипций, так и для пользовательского запроса. Для запросов используется префикс `query:`, соответствующий особенностям моделей семейства E5.

После кодирования текстовые фрагменты и запрос представляются векторами в одном пространстве. Если вектор запроса находится близко к вектору фрагмента, такой фрагмент считается близким по смыслу. В PostgreSQL сравнение выполняется с использованием оператора `<=>`, который вычисляет косинусное расстояние. Результаты сортируются по возрастанию расстояния: чем меньше расстояние, тем выше фрагмент располагается в выдаче. Для ускорения поиска по полю `embedding` создаётся HNSW-индекс.

Семантический поиск оказался особенно важен для работы с естественными пользовательскими вопросами. Пользователь не всегда знает точную формулировку лектора и может задавать вопрос своими словами. Например, вместо термина из лекции он может описать ситуацию или попросить объяснить назна-

чение некоторой конструкции. Семантический поиск позволяет находить фрагменты, близкие по смыслу, даже при отсутствии точных совпадений. Однако он также имеет ограничения: иногда в выдачу попадают фрагменты, относящиеся к теме запроса, но не содержащие прямого ответа.

Чтобы объединить преимущества обоих подходов, был реализован гибридный поиск. В нём сначала независимо выполняются семантический и полнотекстовый поиск, а затем их результаты объединяются. Семантический компонент обеспечивает устойчивость к переформулированным вопросам, а полнотекстовый компонент усиливает фрагменты, содержащие точные термины из запроса. Для объединения результатов используется метод Reciprocal Rank Fusion. Его преимущество состоит в том, что он учитывает позиции фрагментов в отдельных выдачах, а не абсолютные значения оценок, которые у разных поисковых методов имеют различную природу.

В реализации системы семантический поиск возвращает больше кандидатов, так как он используется как основной источник смысловой близости. Полнотекстовый поиск возвращает меньшее число кандидатов и добавляет лексический сигнал. Использовались параметры `semantic_k = 30`, `keyword_k = 5`, сглаживающий коэффициент `k = 60`, вес семантического поиска `1.0` и вес полнотекстового поиска `0.5`. Если один и тот же фрагмент найден обоими методами, его итоговая оценка увеличивается за счёт вклада обоих списков. После вычисления итоговых оценок фрагменты сортируются по убыванию и передаются в RAG-компонент.

RAG-компонент отвечает за формирование ответа на вопрос пользователя. Он получает запрос, вызывает гибридный поиск и выбирает наиболее релевантные фрагменты. По умолчанию в контекст передаются 10 фрагментов. Это значение выбрано как компромисс между полнотой контекста и количеством лишней информации. Если фрагментов слишком мало, модель может не получить нужные сведения. Если фрагментов слишком много, в контекст попадает избыточный текст, который может затруднить формирование точного ответа.

Найденные фрагменты преобразуются в текстовый контекст. Для каждого фрагмента указывается номер, название лекции, временные метки и текстовое содержание. Затем формируется промпт для языковой модели. В инструкции указывается, что модель должна отвечать только по предоставленным фрагментам, не добавлять факты от себя, объединять информацию из нескольких

фрагментов при необходимости и отказываться от ответа, если контекста недостаточно. Такая формулировка промпта важна для RAG-системы, поскольку её задача состоит в генерации ответа именно по материалам лекций, а не по общим знаниям модели.

Генерация ответа в системе выделена в отдельный этап. В демонстрационном приложении предусмотрены два варианта работы: локальный запуск модели через Ollama и обращение к модели через внешний API. При локальном запуске промпт отправляется HTTP-запросом на эндпоинт `/api/generate`. В запросе указываются название модели, сам промпт и параметры генерации, включая температуру и максимальную длину ответа. При использовании внешнего API остальные этапы работы системы остаются такими же: меняется только способ обращения к языковой модели.

Итоговая функция RAG-компонента возвращает не только сгенерированный ответ, но и список фрагментов, использованных при его составлении. Это решение повышает прозрачность работы системы. Пользователь может увидеть, на каких частях лекций основан ответ, а разработчик может проанализировать, на каком этапе возникла ошибка. Если найденные фрагменты нерелевантны, проблема связана с поисковым компонентом. Если фрагменты релевантны, но ответ неверный или неполный, ошибка относится к этапу генерации и интерпретации контекста языковой моделью.

Серверная часть реализована с использованием FastAPI. Она объединяет поисковый компонент, RAG-компонент и пользовательский интерфейс. Бэкенд принимает запрос от клиента, передаёт его в функцию генерации ответа и возвращает результат в формате JSON. В системе реализованы эндпоинты для проверки состояния сервиса, получения списка доступных моделей и отправки пользовательского вопроса. Основным эндпоинтом является `/ask`. Он принимает POST-запрос, содержащий текст вопроса, количество фрагментов `top_k` и название выбранной модели. Для взаимодействия с клиентским приложением настроен CORS.

Пользовательский интерфейс разработан на React с использованием Vite. Он представляет собой одностраничное web-приложение, через которое пользователь может задать вопрос по видеолекциям, выбрать модель, указать количество фрагментов и получить ответ. Основной компонент приложения хранит состояние интерфейса: текст вопроса, значение `topK`, выбранную модель,

результат запроса, состояние загрузки и сообщение об ошибке. На странице размещены поле для ввода вопроса, элементы выбора параметров и кнопка отправки запроса.

После получения ответа данные преобразуются из JSON и сохраняются в состоянии приложения. Результат отображается в двух частях. Сначала пользователь видит ответ языковой модели и название модели, которая его сформировала. Затем отображаются фрагменты транскрипций, использованные как контекст. Для каждого фрагмента выводятся название лекции, название курса, временные метки, оценка релевантности и текст. Временные метки переводятся из секунд в формат минуты:секунды, что делает их удобными для поиска соответствующего места в видеозаписи.

Такой интерфейс выполняет роль демонстрационного клиента и показывает полный цикл работы системы. Пользователь задаёт вопрос на естественном языке, сервер выполняет гибридный поиск, RAG-компонент формирует промпт, языковая модель генерирует ответ, а интерфейс выводит как итоговый ответ, так и исходные фрагменты.

После реализации основных компонентов была проведена экспериментальная оценка системы. Для тестирования использовался набор видеолекций о языке C++. Лекции были подготовлены студенческими клубами разработки СГУ и посвящены различным темам: устройству языка, компиляции, типам данных, переменным, указателям, работе с памятью и другим аспектам C++. Всего в базе данных использовалось 10 видеолекций. После транскрипции и разбиения текста было получено 2116 фрагментов.

Для оценки были подготовлены два набора запросов. Первый набор включал 15 вопросов и использовался для проверки поискового компонента. Второй набор включал 30 вопросов и применялся для оценки RAG-компонента. Запросы относились к нескольким типам. Часть вопросов была направлена на определение понятий, например, стандарта языка, переменной или ссылки в C++. Другая часть требовала сравнения понятий, например различия между `float` и `double`, массивом и указателем, `if` и `switch`. Также использовались вопросы о причинах и последствиях ошибок, например о выходе за границы массива или разыменовании неверного указателя.

В набор были включены и такие вопросы, для ответа на которые может потребоваться информация из нескольких фрагментов лекции. Это важно для

проверки RAG-подхода, поскольку ответ в видеолекции не всегда содержится в одном месте. Кроме того, использовались вопросы, на которые в лекциях нет ответа, например вопрос о различиях между C++ и Java. Такие запросы позволяют проверить, будет ли модель корректно отказываться от ответа при недостатке контекста или начнёт использовать сведения, отсутствующие в материалах лекций.

Разметка результатов выполнялась вручную. Для поискового компонента оценивалась релевантность найденных фрагментов по шкале от 0 до 2. Оценка 2 означала, что фрагмент напрямую отвечает на вопрос или содержит существенную часть ответа. Оценка 1 означала частичную релевантность. Оценка 0 соответствовала нерелевантному фрагменту. Для сравнения методов использовались метрики NDCG, MRR и MAP. Они позволяют оценивать не только наличие релевантных результатов, но и их положение в выдаче.

В оценке поискового компонента сравнивались случайная выдача, полнотекстовый поиск, семантический поиск и гибридный поиск. Случайная выдача использовалась как нижняя граница качества. Она показала значения NDCG 0.528, MRR 0.280 и MAP 0.373. Эти значения не равны нулю, поскольку все запросы относятся к одному набору лекций по C++, и даже случайно выбранные фрагменты иногда оказываются частично связанными с темой вопроса.

Полнотекстовый поиск показал NDCG 0.609, MRR 0.567 и MAP 0.561. Его результаты оказались нестабильными. В тех случаях, когда слова запроса совпадали с терминами в транскрипции, полнотекстовый поиск находил релевантные фрагменты и располагал их высоко. Однако для ряда запросов он не возвращал результатов, так как формулировка пользователя отличалась от формулировки лектора. На тестовом наборе полнотекстовый поиск вернул хотя бы один непустой результат для 9 из 15 запросов. При этом на подмножестве запросов, где было найдено достаточное число фрагментов, его показатели были высокими: NDCG 0.969, MRR 1.000 и MAP 0.971. Это показывает, что основная проблема полнотекстового поиска связана не с ранжированием найденных результатов, а с полнотой поиска.

Семантический поиск оказался более устойчивым. Он возвращал содержательные результаты для всех тестовых запросов и показал NDCG 0.866, MRR 0.571 и MAP 0.843. Эти значения подтверждают, что для поиска по транскрипциям видеолекций важно учитывать смысловую близость вопроса и фрагмента.

Пользователь может задавать вопрос естественным языком, не повторяя точные слова лектора, и семантический поиск лучше справляется с такими ситуациями.

Наилучшие результаты показал гибридный поиск. Его значения составили NDCG 0.882, MRR 0.629 и MAP 0.862. Улучшение по сравнению с семантическим поиском оказалось не очень большим, так как семантический поиск уже хорошо справлялся с большинством запросов. Однако добавление полнотекстового сигнала позволило повысить позиции фрагментов, содержащих точные термины. Поэтому гибридный поиск был выбран в качестве основного метода для RAG-компонента. Он объединяет устойчивость семантического поиска и точность лексического совпадения.

Оценка RAG-компонента проводилась отдельно от оценки поиска. Для каждого вопроса сохранялись текст запроса, найденные фрагменты, ответ языковой модели и служебная информация. Все модели оценивались при одинаковом поисковом компоненте и одинаковом промпте. Это позволило сравнивать качество генерации без влияния разных способов поиска. Для разметки использовались четыре критерия: релевантность найденного контекста, корректность ответа, соответствие ответа контексту и полнота ответа. Каждый критерий оценивался по шкале от 0 до 2.

Критерий релевантности контекста показывал, достаточно ли найденных фрагментов для ответа на вопрос. Критерий корректности отражал фактическую правильность ответа относительно содержания лекций. Критерий соответствия контексту показывал, опирается ли модель на найденные фрагменты или добавляет сведения, отсутствующие в них. Критерий полноты оценивал, раскрывает ли ответ основные аспекты вопроса. Например, для вопроса на сравнение было недостаточно просто назвать два понятия: требовалось объяснить различие между ними.

В экспериментах использовались модели:

— `gemma3:4b-it-q4_K_M`;
— `gemma3:12b`;
— `qwen-3-235b-a22b-instruct-2507`.

Первые три модели запускались локально через Ollama, а последняя использовалась через внешний API. Модель `gemma3:4b-it-q4_K_M` показала значения correctness 1.27, faithfulness 1.50 и completeness 1.10. Она редко формировала сильные галлюцинации, но часто отвечала слишком кратко и не раскрыва-

ла вопрос полностью. Модель `gemma3:12b` получила correctness 1.43, faithfulness 1.43 и completeness 1.30. Её ответы были более развёрнутыми, однако иногда возникали неточности при интерпретации найденного контекста. Модель `qwen3:8b` показала correctness 1.50, faithfulness 1.67 и completeness 1.47. Она чаще использовала найденные фрагменты более полно и формировала связные ответы. Наилучшие результаты получила модель `qwen-3-235b-a22b-instruct-2507`: correctness 1.63, faithfulness 1.83 и completeness 1.60.

Дополнительно были рассмотрены только те запросы, для которых поисковый компонент нашёл хотя бы частично релевантный контекст. На этом подмножестве результаты всех моделей стали выше. Для `gemma3:4b` значения correctness, faithfulness и completeness составили 1.52, 1.56 и 1.32. Для `gemma3:12b` они составили 1.72, 1.72 и 1.56. Для `qwen3:8b` значения достигли 1.80, 1.76 и 1.76. Для `qwen3:235b` были получены значения 1.96, 1.88 и 1.92. Эти результаты показывают, что качество ответа RAG-системы существенно зависит от качества найденных фрагментов.

В ходе анализа ошибок были выделены несколько основных типов проблем. Первый тип связан с ситуацией, когда поисковый компонент не находит релевантный фрагмент. В этом случае языковая модель не получает достаточной информации и либо отказывается отвечать, либо пытается сформировать ответ на основе собственных знаний. Для RAG-системы второй вариант является критической ошибкой, так как ответ должен быть основан на материалах лекций. Даже если такой ответ выглядит правдоподобно, он не подтверждается найденным контекстом.

Второй тип ошибок возникает при слабом контексте. В таких случаях система находит фрагменты, относящиеся к теме вопроса, но недостаточные для полного ответа. Например, в контексте может быть определение, но отсутствовать пример или объяснение отличий от близкого понятия. Тогда модель формирует частично правильный ответ, но он оказывается неполным. Такие ошибки показывают, что качество разбиения транскрипций и качество ранжирования напрямую влияют на итоговую работу RAG-компонента.

Третий тип ошибок связан с неполнотой ответа. Он особенно заметен у компактных локальных моделей. Модель может получить релевантный контекст, но использовать только часть информации и сформировать слишком короткий ответ. Для образовательного сценария это важно, потому что пользователю

обычно требуется не только краткий факт, но и понятное объяснение. Например, если пользователь спрашивает о различии двух конструкций, ответ должен не просто упомянуть обе конструкции, а показать, в чём состоит их отличие и когда они применяются.

Четвёртый тип ошибок связан с галлюцинациями. При недостатке информации модель может добавлять сведения, отсутствующие в найденных фрагментах. Более крупные модели чаще дают развёрнутые ответы, но при слабом контексте иногда дополняют их общими знаниями. Для обычного вопросно-ответного сценария такой ответ может быть приемлемым, если он фактически верен, однако для системы поиска по конкретному набору лекций это считается ошибкой. Пользователь ожидает получить ответ именно по загруженным материалам, а не по внешним знаниям модели.

Пятый тип ошибок связан с неверной интерпретацией контекста. В этом случае релевантные фрагменты найдены, но модель неправильно понимает их смысл, смешивает близкие понятия или делает слишком широкий вывод. Такие ошибки показывают, что наличие подходящего контекста не гарантирует правильный результат. Языковая модель должна не только получить нужные фрагменты, но и корректно связать их между собой, отделить основную информацию от второстепенной и сформировать ответ в соответствии с вопросом пользователя.

Разработанная система показывает, что сочетание гибридного поиска и RAG-подхода применимо для поиска информации в образовательных видеолекциях. В работе был реализован полный цикл обработки материалов: от транскрипции и разбиения лекций на фрагменты до поиска релевантного контекста и генерации ответа на вопрос пользователя. Экспериментальная оценка подтвердила, что гибридный поиск работает устойчивее полнотекстового и семантического поиска по отдельности, поскольку объединяет точные лексические совпадения и смысловую близость запроса к фрагментам.

Оценка RAG-компонента показала, что качество ответа сильно зависит от найденного контекста. При релевантной выдаче модели чаще дают корректные и полные ответы, а при слабом контексте возникают неполнота, неверная интерпретация или добавление сведений вне материалов лекций. Таким образом, полученная система решает поставленную задачу поиска по видеолекциям и формирования ответов на основе найденных фрагментов транскрипций.