МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
Школа компьютерных наук

ОТЧЕТ ПО ДИСЦИПЛИНЕ  
«ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ»  
ПО ТЕМЕ «РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ПОМОЩНИКА ОБУЧАЮЩИХСЯ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

|  |  |  |
| --- | --- | --- |
| Выполнил обучающийся 4 курса, 2021 года поступления | *(подпись)* | Голованов Сергей Евгеньевич |
|  |  |  |
| Руководитель, Доцент | *(подпись)* | Перевалова Мария Николаевна |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_heading=h.gjdgxs)

[ГЛАВА 1. РАБОТА С ДАННЫМИ 6](#_heading=h.2et92p0)

[1.1 ОПИСАНИЕ ДАННЫХ 6](#_heading=h.tyjcwt)

[1.2 ОПИСАНИЕ ОБРАБОТКИ ДАННЫХ 7](#_heading=h.3dy6vkm)

[1.3 ПОДГОТОВКА ДАННЫХ ДЛЯ ОБУЧЕНИЯ МОДЕЛИ 9](#_heading=h.hcibho8udujs)

[ГЛАВА 2. РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ПОМОЩНИКА 10](#_heading=h.1t3h5sf)

[2.1 ФУНКЦИОНАЛ 10](#_heading=h.6s4yz9er1pcq)

[2.2 АРХИТЕКТУРА 12](#_heading=h.n7idffcn7eot)

[2.3 РЕАЛИЗАЦИЯ МОДЕЛИ 14](#_heading=h.45wpv0d71qpn)

[2.4 РАЗРАБОТКА ЧАТ-БОТА 15](#_heading=h.oegpykdzjuai)

[2.5 РАЗРАБОТКА БАЗЫ ДАННЫХ 16](#_heading=h.o5j2c0b0j8tw)

[2.6 ПОДБОР СТЕКА ТЕХНОЛОГИЙ 18](#_heading=h.5cx6rpxpo5mo)

[ЗАКЛЮЧЕНИЕ 20](#_heading=h.17dp8vu)

[СПИСОК ЛИТЕРАТУРЫ 21](#_heading=h.lnxbz9)

[ПРИЛОЖЕНИЕ 1 23](#_heading=h.glmo3rgi388l)

# ВВЕДЕНИЕ

За последнее время в мире наблюдается увеличение количества сервисов, использующих технологии искусственного интеллекта (далее – ИИ) при работе с клиентской базой. Согласно опросу аналитического центра НАФИ (Москва), проведенного в марте 2024 года, около 33% представителей малого и среднего бизнеса (из числа имеющих опыт использования) используют ИИ-инструменты во взаимодействии с клиентами[[1]](#footnote-1).

Одним из направлений использования технологий ИИ являются современные образовательные системы, которые сталкиваются с рядом вызовов, таких как увеличение объема учебного материала, разнообразие стилей обучения и необходимость индивидуализации подхода к каждому ученику. Использование искусственного интеллекта открывает новые возможности для создания адаптивных образовательных платформ, которые смогут учитывать особенности каждого обучающегося [4].

Одним из форматов для реализации интеллектуального помощника (ассистента) с применением технологий искусственного интеллекта может быть чат-бот. Чат-боты являются многофункциональными инструментами коммуникации, которые направлены на облегчение и упрощении жизни частного пользователя, а также оптимизацию коммуникаций в компаниях [1]. Применение чат-ботов в образовательном процессе является эффективным инструментом выстраивания коммуникации между преподавателями и обучающимися, распространения структурированной и достоверной информации [2].

Использование в чат-ботах языковых моделей позволяет перевести взаимодействие пользователя с сервисом на естественный язык. Языковые модели обучаются на огромных объемах текстов, что позволяет им понимать контекст и генерировать тексты, максимально приближенные к человеческому стилю общения. Преимуществами больших языковых моделей (далее – LLM, Large language model) является возможность персонализации обучения, автоматизации рутинных задач, доступ к знаниям. Однако, недостатками LLM являются сомнительная достоверность информации, необходимость постоянного подключению к интернету, а также ряд этических аспектов [5].

Одним из способов повышения достоверности информации является использование метода поисковой расширенной генерации (RAG). RAG-системы динамически извлекают информацию из внешних источников, таких как базы данных, интернет-ресурсы или конкретные наборы документов различных форматов, в режиме реального времени. После получения информации LLM использует этот контекст для создания ответов. Ответы зависят от точности найденных данных и могут ссылаться на источники информации [6].

Метод RAG уже был успешно применен для реализации ИИ-ассистента в формате чат-бота в Тюменском государственном университете в 2024 году. Разработанный чат-бот отвечал на вопросы студентов по организации образовательного процесса, в частности по правилам проведения зачетов, экзаменов и выставления баллов [3].

В рамках федерального проекта **«**Развитие кадрового потенциала IT-отрасли**»** в Тюменском государственном университете реализуется проект **«**Цифровая кафедра**»**. В 2024 году на образовательные программы проекта поступило около 4000 студентов. При этом курс «Прикладной анализ данных» курирует только один преподаватель. Из-за большого количества студентов и ограничения во времени, преподаватель не успевает отвечать на все вопросы обучающихся. Таким образом, существует проблема необходимости наличия инструмента для ускорения процесса “вопрос-ответ”. Решение данной проблемы позволит отсечь простые вопросы и увеличить время преподавателю на разъяснение студентам более сложных вопросов при изучении учебного материала.

Целью данной работы является разработка чат-бота, который будет использовать искусственный интеллект, чтобы формулировать ответы на вопросы обучающихся **«**Цифровой кафедры**»**, что позволит ускорить процесс “вопрос-ответ”, а также обеспечить индивидуальный подход к изучению учебного материала.

Для достижения данной цели были выделены следующие задачи:

* получение и подготовка данных для обучения модели;
* выбор и обучение большой языковой модели (LLM) на полученных данных;
* реализация чат-бота на основе платформы Telegram, с использованием полученной модели;
* проверка работоспособности бота на аудитории.

# ГЛАВА 1. РАБОТА С ДАННЫМИ

## 1.1 ОПИСАНИЕ ДАННЫХ

Исходными данными являлись скриншоты текстов вопросов и ответов ассессмента "Цифровой кафедры". Данные скриншоты были собраны другими студентами и присланы на электронную почту «Школы компьютерных наук» ТюмГУ.

Присланные письма были скачаны и сохранены в формате .eml с помощью программы Mozilla Thunderbird. Всего было сохранено 1313 файлов писем, из которых 181 письмо содержало скриншоты.

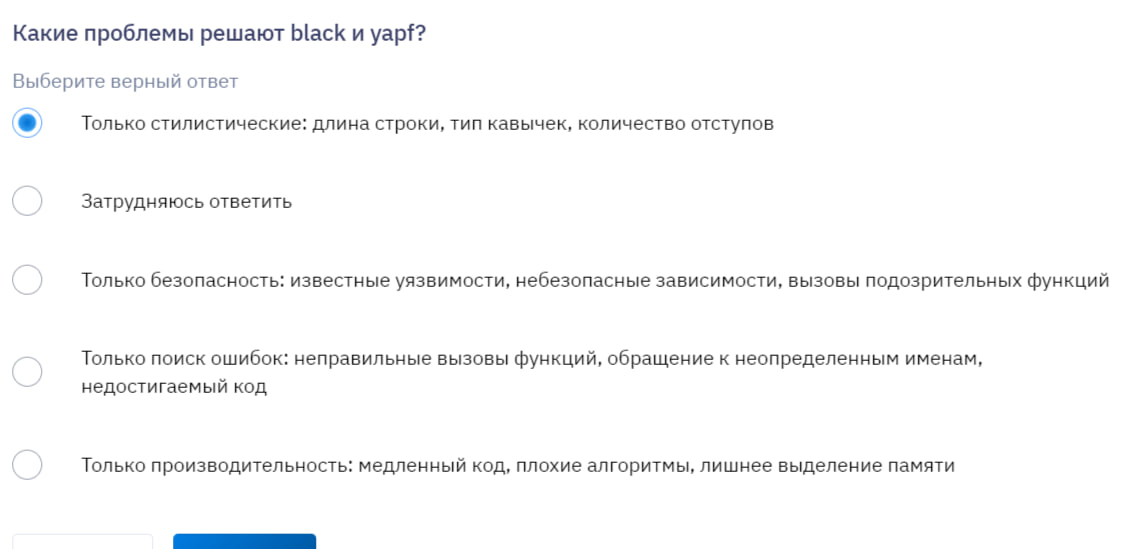
Для извлечения скриншотов из файлов писем был написан скрипт на языке Python (Приложение 1). Данный скрипт проходится по каждому письму, определяет содержит ли письмо вложения, извлекает и сохраняет их на диск. Форматы и количество вложений представлены в таблице 1.

Таблица 1

Форматы и количество вложений

|  |  |
| --- | --- |
| **Формат** | **Количество** |
| Изображение PNG | 1696 |
| Изображение JPEG | 1474 |
| Документ Microsoft Word (.docx) | 29 |
| Документ Adobe PDF (.pdf) | 22 |
| Архив ZIP | 20 |
| Архив RAR | 4 |
| Архив 7z | 4 |
| Презентация Microsoft PowerPoint (.pptx) | 2 |

Файлы вложений, не являющимися изображениями, также содержали скриншоты и были извлечены в виде изображений. Всего было извлечено 5023 скриншота (Рисунок 1).

  
Рис. 1. Пример исходных данных

## 1.2 ОПИСАНИЕ ОБРАБОТКИ ДАННЫХ

Для извлечения текстов вопросов из скриншотов были применены методы оптического распознавания символов, далее OCR. В частности, был использован движок Tesseract от компании Google. Полученный текст содержал лишнюю информацию, «мусор», зачастую символы распознавались неправильно, части текста на английском языке распознаются как русский и наоборот (Рисунок 2).

Рис. 2. Пример распознанного текста

Какие проблемы решают black и yapf?\nВыберите верный ответ\n®) = Tonbko стилистические: длина строки, тип кавычек, количество отступов\nЗатрудняюсь ответить\nТолько безопасность: известные уязвимости, небезопасные зависимости, вызовы подозрительных функций\nТолько поиск ошибок: неправильные вызовы функций, обращение к неопределенным именам, недостигаемый код\nТолько производительность: медленный код, плохие алгоритмы, лишнее выделение памяти

В связи с этим, а также большим количеством исходных данных, было принято решение обработать данные с помощью большой языковой модели Llama 3.1 8B Instruct. Для использования данной модели требуется подать ей на вход prompt (промпт, запрос), содержащий инструкцию для выполнения на естественном языке (Рисунок 3). Промпт для модели состоял из нескольких частей: system и user. В часть system передается инструкция для выполнения, в часть user передается распознанный текст.

Рис. 3. Инструкция для очистки и форматирования

Ниже приведён текст на русском языке в котором содержатся вопрос или несколько вопросов, а также от 1 до нескольких вариантов ответа. Выдели вопрос(ы) и ответы, и определи правильный ответ(ы). Игнорируй посторонний шум. Верни их в формате json: {\"question\": \"\",\"answers\": [],\"correct\_answer\": \"\"}.

Однако, качество данной обработки нельзя назвать хорошим, так как зачастую модель неправильно определяла вопросы и ответы (Рисунок 4).

Рис. 4. Пример текста после очистки и форматирования

{

"question": "Какие проблемы решают black и yapf?",

"answers": [

"®) = Tonbko стилистические: длина строки, тип кавычек, количество отступов",

"Только безопасность: известные уязвимости, небезопасные зависимости, вызовы подозрительных функций",

"Только поиск ошибок: неправильные вызовы функций, обращение к неопределенным именам, недостигаемый код",

"Только производительность: медленный код, плохие алгоритмы, лишнее выделение памяти ",

"Затрудняюсь ответить"

],

"correct\_answer": "®) = Tonbko стилистические: длина строки, тип кавычек, количество отступов"

}

В связи с чем было принято решение вручную обработать полученные данные и использовать их для тестовой тренировки модели.

Для отбора лучших вариантов полученные тексты были сгруппированы по схожести текстов вопросов, что позволило дедуплицировать данные. Варианты ответов были убраны и остались только пары "вопрос-правильный ответ” в количестве 850 шт. После ручной обработки осталось всего 621 пара.

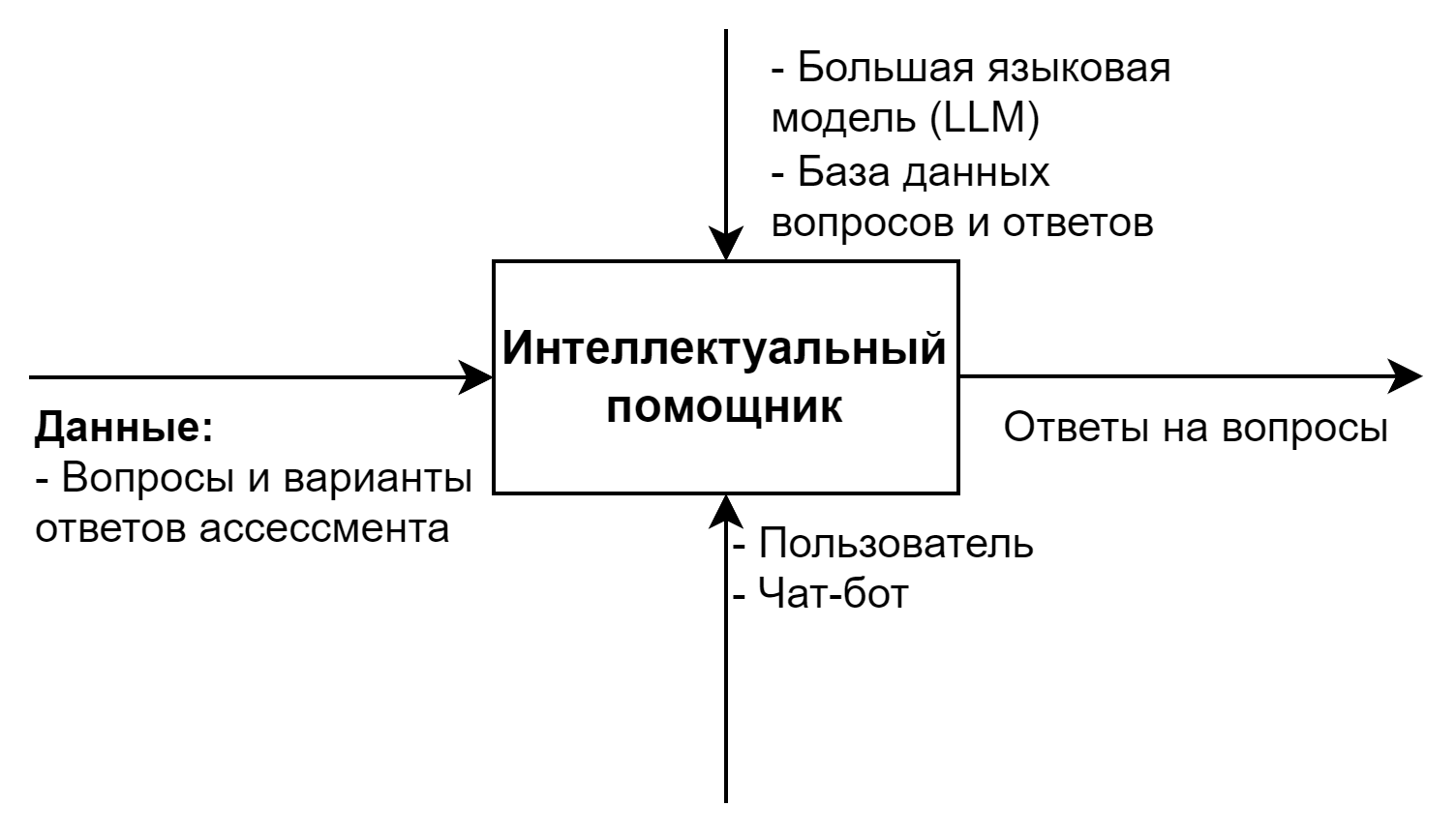
# ГЛАВА 2. РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ПОМОЩНИКА

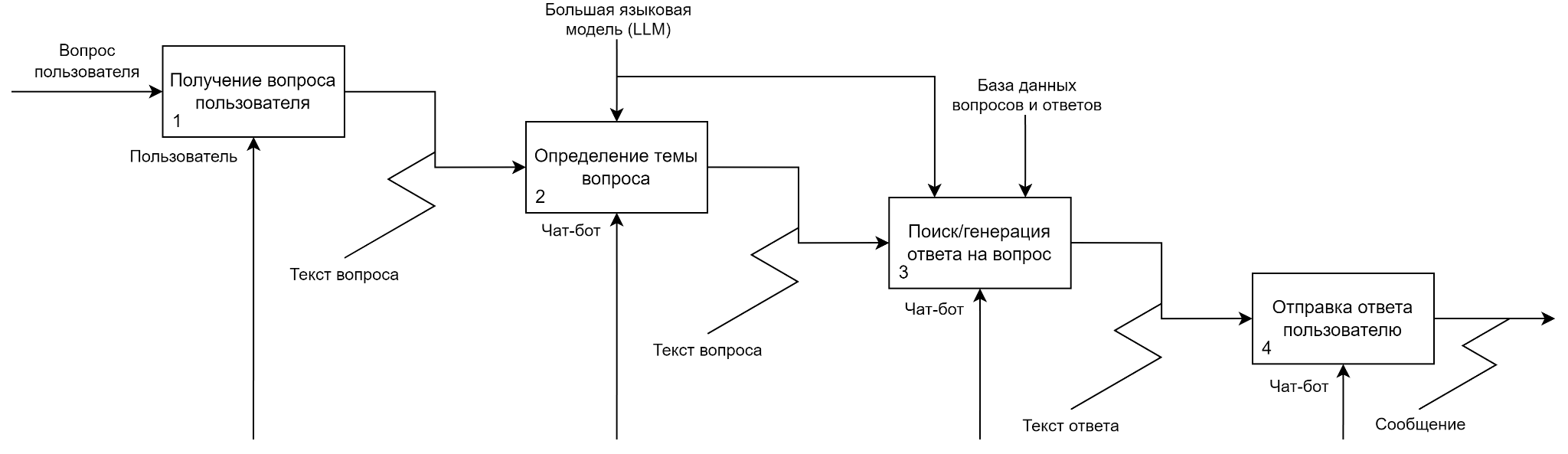
## 2.1 ФУНКЦИОНАЛ

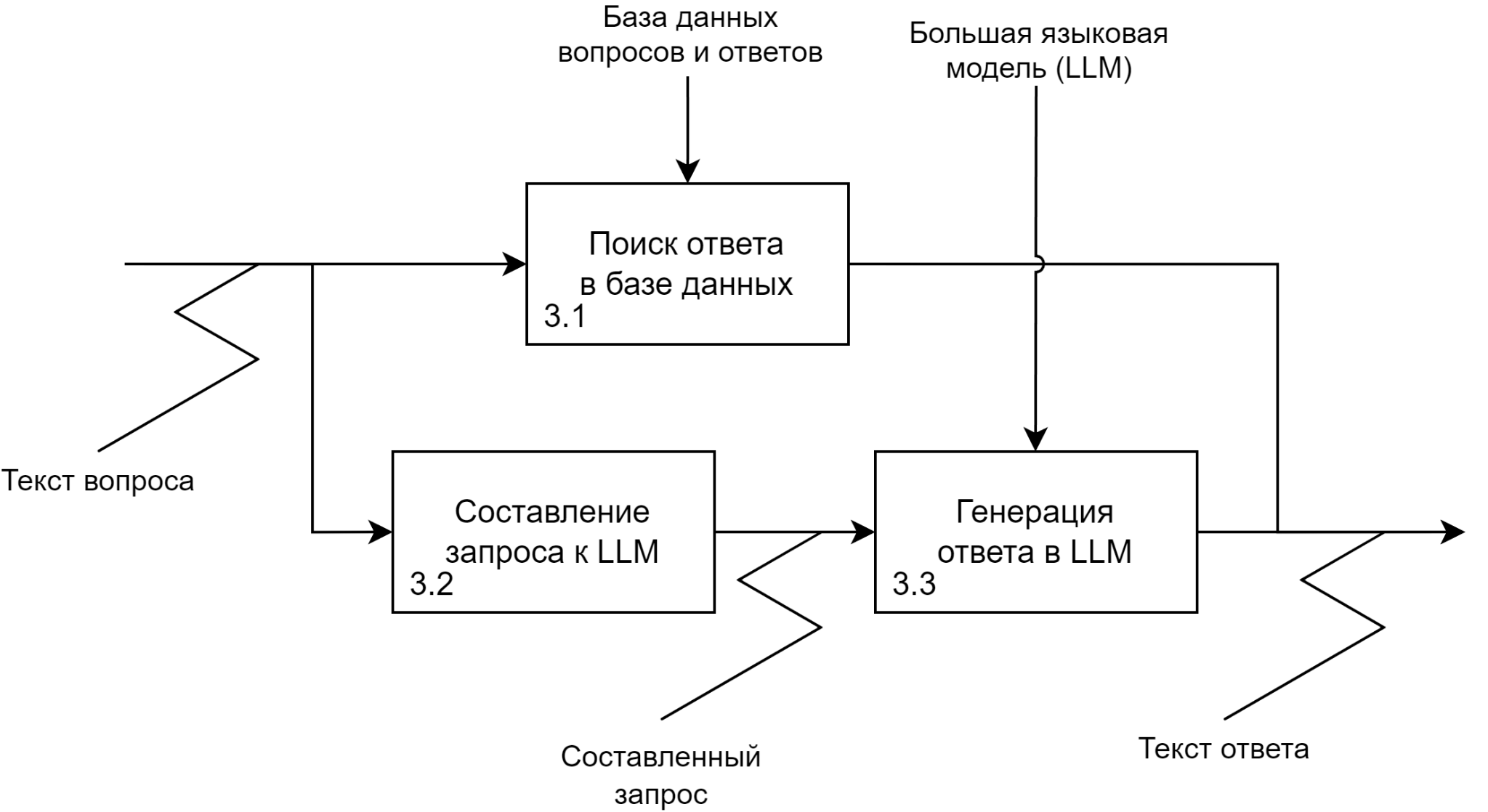
Разрабатываемое приложение должно обладать следующим функционалом:

* Классификация вопросов пользователей на соответствие темам «Цифровой кафедры» и последующая фильтрация вопросов, не прошедших проверку.
* Чат-бот должен работать в нескольких режимах:
  + в личных сообщениях (один-на-один);
  + в групповом чате.

На рисунках 5-7 представлена функциональная диаграмма приложения.

  
Рис. 5. Функциональная диаграмма начального уровня

  
Рис. 6. Функциональная диаграмма нулевого уровня

  
Рис. 7. Функциональная диаграмма первого уровня

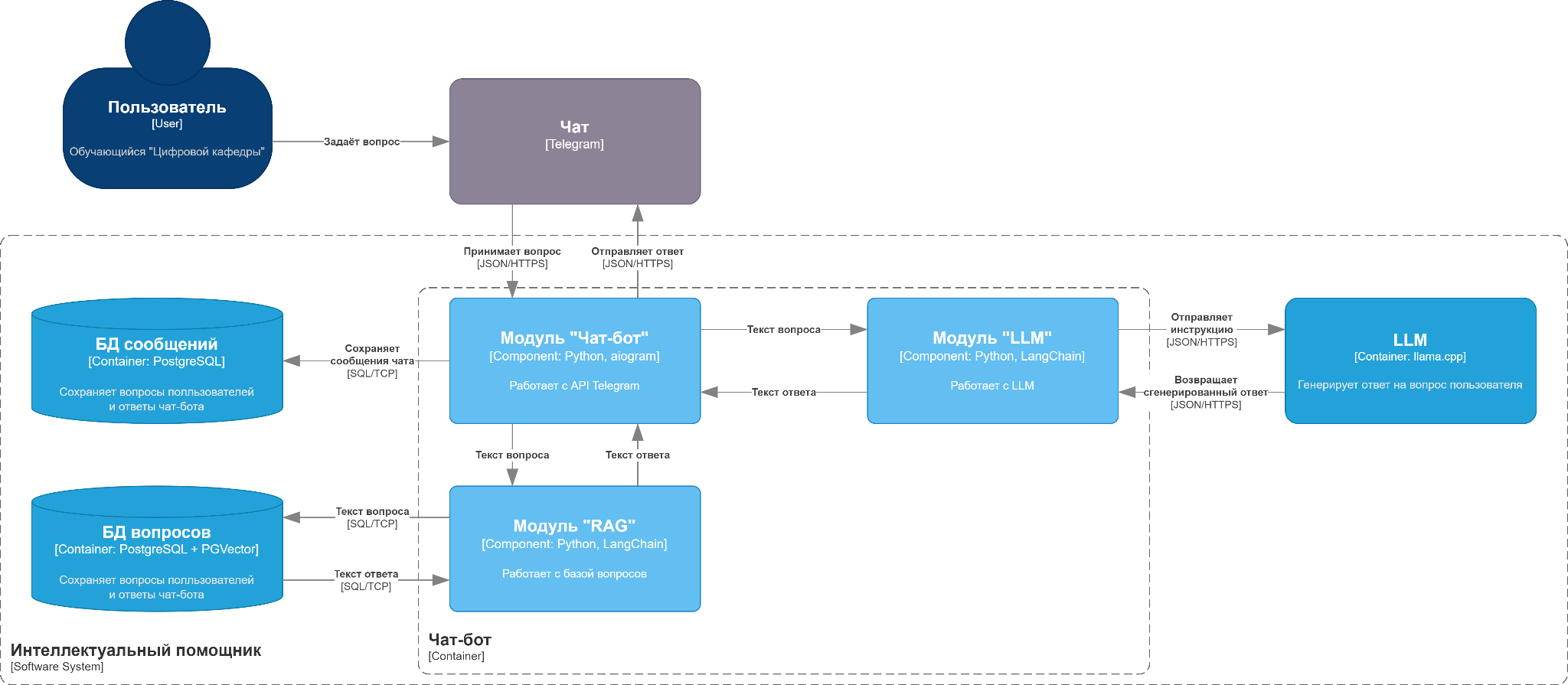
## 2.2 АРХИТЕКТУРА

На рисунке 8 представлена архитектура приложения. Она состоит из трёх компонентов: чат-бот, база данных PostgreSQL (для удобства разделена на две части) и LLM движок.

Модуль «Чат-бот» взаимодействует с API Telegram, сохраняет сообщения пользователя и найденные/сгенерированные ответы в базе данных сообщений.

Модуль «RAG» взаимодействует с базой данных вопросов, получает текст вопроса пользователя для поиска и возвращает текст найденного ответа (если вопрос есть в БД).

Модуль «LLM» взаимодействует с API llama.cpp, форматирует инструкции для большой языковой модели и обрабатывает полученные ответы.

Рис. 8. Архитектура приложения в нотации «C4»

## 2.3 РЕАЛИЗАЦИЯ МОДЕЛИ

За основу была взята большая языковая модель Llama 3.1 8B Instruct, так как она поддерживает русский язык.

В качестве обучающего датасета использовалась подготовленная ранее пары “вопрос-ответ”. Для обучения их следовало переформатировать в формат шаблона выбранной модели.

Рис. 9. Шаблон запроса для модели

### Instruction:

Ты ИИ-ассистент, который отвечает на вопросы пользователей. Отвечай на русском языке и только на русском языке. Постарайся давать однозначные и исчерпывающие ответы.

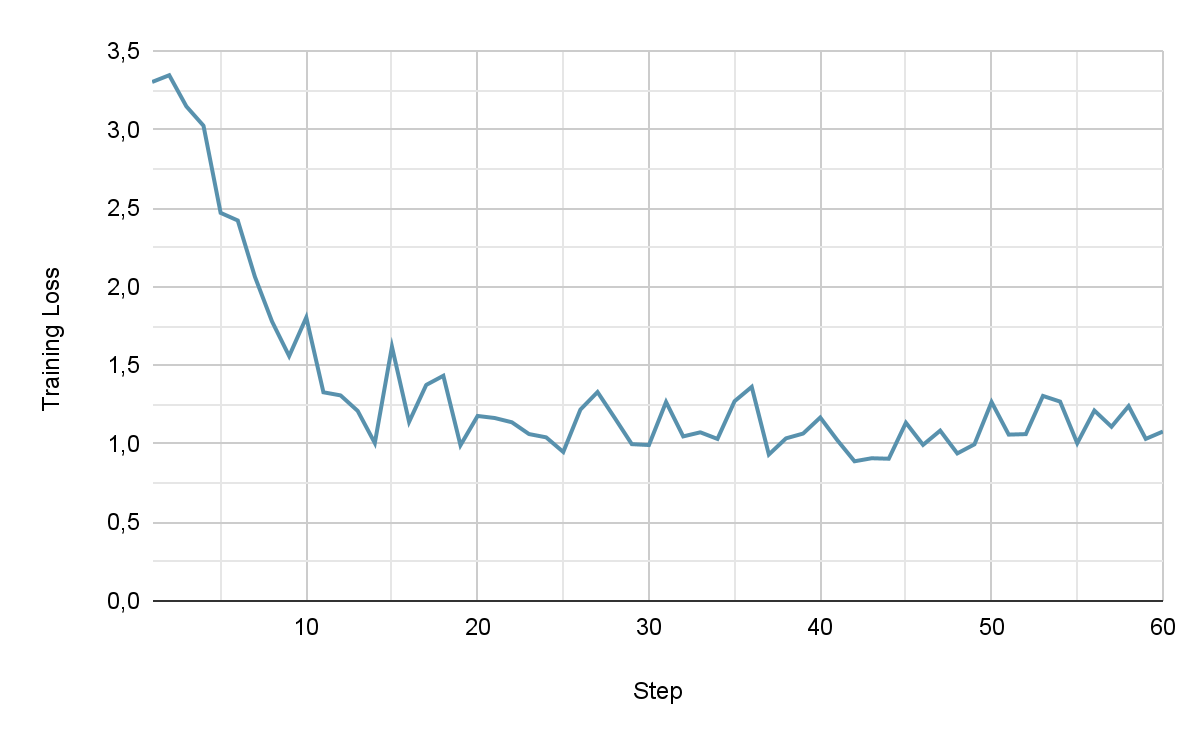
### Input:

Какие проблемы решают black и yapf?

### Response:

Только стилистические: длина строки, тип кавычек, количество отступов.

Обучение проводилось на платформе Google Collab. Как видно из рисунка 10, обучение перестало приносить значимый результат после 15 итерации. Полученная модель была сохранена в формате .gguf, для дальнейшего запуска с помощью llama.cpp.

  
Рис. 10. График обучения модели

## 2.4 РАЗРАБОТКА ЧАТ-БОТА

В качестве интерфейса взаимодействия пользователей с моделью был создан телеграм-бот с помощью библиотеки aiogram. Бот принимает сообщения из чата с помощью API Telegram. Каждое полученное сообщение проверяется на то, является ли оно вопросом и/или просьбой о помощи, а также на соответствие сообщения теме «Цифровой кафедры» с помощью LLM.

Таблица 2

Шаблон запроса для проверки сообщений пользователя

|  |  |
| --- | --- |
| Шаблон запроса | <|begin\_of\_text|>  <|start\_header\_id|>system<|end\_header\_id|>  Определи является ли приведенный текст вопросом или просьбой о помощи. Верни значение в формате bool: True или False. Если тема вопроса не относится к IT, Computer Science или управлению бизнес процессами, возвращай False.  <|eot\_id|>  <|start\_header\_id|>user<|end\_header\_id|>  {user\_prompt}  <|eot\_id|>  <|start\_header\_id|>assistant<|end\_header\_id|> |
| Описание переменных | user\_prompt – текст сообщения пользователя |

Если сообщение проходит проверку, то модель начинает генерировать ответ на заданный вопрос и/или просьбу. После генерации ответа, бот отправляет его обратно в чат, сопроводив кнопками для оценки качества ответа.

## 2.5 РАЗРАБОТКА БАЗЫ ДАННЫХ

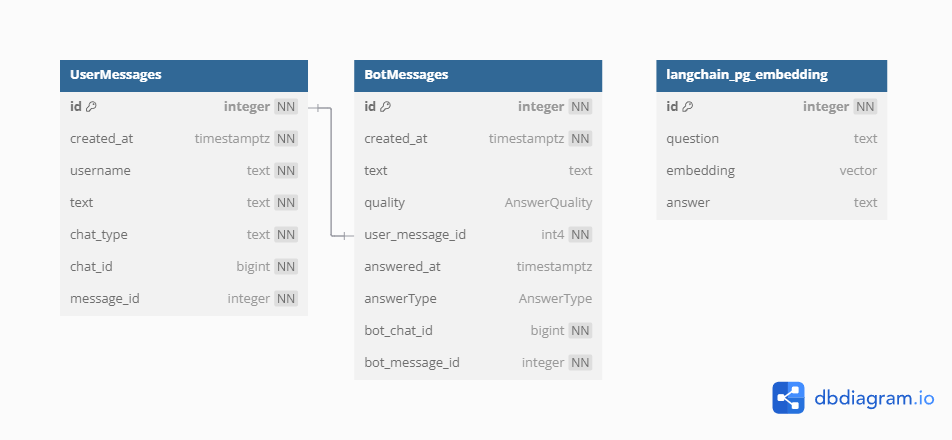
Каждое отправленное сообщение пользователя, как и ответ бота с пользовательской оценкой, сохраняются в базу данных для дальнейшего анализа и использования при повторном обучении модели.

Таблица 3

Описание сущностей базы данных

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Комментарий |
| Таблица Questions (вопросы) | | |
| id | INTEGER | Первичный ключ |
| text | TEXT | Текст вопроса |
| embedding | VECTOR | Векторное представление вопроса |
| answer | TEXT | Текст ответа |
| Таблица UserMessages (сообщения пользователей) | | |
| id | SERIAL | Первичный ключ |
| created\_at | TIMESTAMPTZ | Дата-время отправки сообщения пользователем |
| username | TEXT | Имя пользователя (может быть NULL) |
| text | TEXT | Текст сообщения пользователя |
| chat\_type | TEXT | Тип чата, в котором было отправлено сообщение пользователя |
| chat\_id | BIGINT | Номер чата из API Telegram |
| message\_id | INTEGER | Номер сообщения из API Telegram |
| Таблица BotMessages (сообщения бота) | | |
| id | SERIAL | Первичный ключ. |
| created\_at | TIMESTAMPTZ | Дата-время принятия сообщения ботом. |
| text | TEXT | Текст сообщения бота; может быть NULL. |
| quality | ENUM | Оценка качества ответа бота, поставленная пользователем; может быть GOOD, BAD или NULL. |
| user\_message\_id | INTEGER | Идентификатор сообщения пользователя, на которое отвечает чат-бот, внешний ключ. |
| answered\_at | TIMESTAMPTZ | Дата-время отправки сообщения ботом. |
| answerType | ENUM | Тип ответа; может быть RAG или LLM. |
| bot\_chat\_id | BIGINT | Номер чата из API Telegram. |
| bot\_message\_id | INTEGER | Номер сообщения из API Telegram. |

На рисунке 11 приведена схема базы данных чат-бота.

  
Рисунок 11. Схема базы данных чат-бота

## 2.6 ПОДБОР СТЕКА ТЕХНОЛОГИЙ

Для разработки чат-бота на платформе Telegram был выбран язык программирования Python, библиотека aiogram, СУБД PostgreSQL и программа llama.cpp.

**aiogram** – это асинхронный фреймворк на языке Python для разработки ботов в Telegram, основанный на библиотеке asyncio. Он обеспечивает высокую производительность за счет использования асинхронного программирования и поддерживает все актуальные возможности Telegram Bot API. aiogram обладает модульной архитектурой, позволяя создавать масштабируемые и легко расширяемые боты. Благодаря простоте в использовании и эффективному управлению запросами, фреймворк широко применяется в автоматизации сервисов, создании чат-ботов и интеграции Telegram с внешними системами.

**llama.cpp** – это высокоэффективная реализация языковых моделей LLaMA для процессоров (CPU), разработанная на C++ с использованием библиотеки ggml. Проект оптимизирован для работы на устройствах с ограниченными вычислительными ресурсами за счет поддержки инструкций AVX, AVX2 и AVX-512, а также методов квантования. Поддерживается кроссплатформенность и работа с моделями в формате GGUF. llama.cpp позволяет запускать локальные языковые модели без использования облачных сервисов, обеспечивая приватность данных, и применяется в автономных чат-ботах, встраиваемых системах и исследовательских задачах.

# ЗАКЛЮЧЕНИЕ

В ходе работы был разработан интеллектуальный помощник в виде чат-бота Telegram, для ответов на вопросы студентов по темам “Цифровой кафедры”. Также была собрана база вопросов ассессмента “Цифровой кафедры”.

# СПИСОК ЛИТЕРАТУРЫ

1. Кузьменко, Ф. В. Чат-бот как инновационная система интернет-коммуникаций / Ф. В. Кузьменко – Текст: электронный // ДОСТИЖЕНИЯ ВУЗОВСКОЙ науки 2019 : сборник статей VII Международного научно-исследовательского конкурса, Пенза, 15 марта 2019 года. – Пенза: "Наука и Просвещение" (ИП Гуляев Г.Ю.), 2019. – С. 41-43.
2. Пожалуева, В. И. Применение чат-ботов мессенджера Telegram в системе взаимодействия педагога и обучающегося / В. И. Пожалуева, О. А. Милькевич – Текст: электронный // Студенческая наука Подмосковью : Сборник материалов Международной научной конференции молодых ученых, Орехово-Зуево, 13 апреля 2022 года. – Орехово-Зуево: Государственный гуманитарно-технологический университет, 2022. – С. 525-529.
3. Чикирева Е. Д., Белоглазова Е. В., Глухих И. Н. Нейроассистент в области консультации по образовательному процессу / Е. Д. Чикирева, Е. В. Белоглазова, И. Н. Глухих – Текст: электронный // Математическое и информационное моделирование: материалы Всероссийской конференции молодых ученых. Вып. 22. – Тюмень, 2024. – ТюмГУ-Press, 2024.
4. Видова Т. А., Романова И. Н. Возможности применения технологий искусственного интеллекта в образовательном процессе / Т. А. Видова, И. Н. Романова – Текст: электронный // Образовательные ресурсы и технологии. – 2023. – №. 1 (42). – С. 27-35.
5. Чунгулова Г. К., Оразалиева Э. Н. ВОЗМОЖНОСТИ И ПРОБЛЕМЫ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ В ОБРАЗОВАНИИ НА ПРИМЕРЕ CHATGPT / Г. К. Чунгулова, Э. Н. Оразалиева – Текст: электронный // Наука и реальность/Science & Reality. – 2024. – №. 4 (20). – С. 85-91.
6. Федоров В.О. Большие языковые модели с поисковой расширенной генерацией: обзор и перспективы / В.О. Федоров, Р.А. Поляков. – Текст: электронный // Оригинальные исследования. – 2023. – Т. 13, № 12. – С. 43-47.

# ПРИЛОЖЕНИЕ 1

Исходный код скрипта для извлечения скриншотов из файлов писем

|  |
| --- |
| eml3.py |
| import email  import os  from email import policy  def get\_file\_extension(content\_type):  content\_type = content\_type.lower()  match content\_type:  case "text/plain":  return "txt"  case "text/html":  return "html"  case "image/png":  return "png"  case "image/jpeg" | "image/jpg":  return "jpg"  case "application/vnd.openxmlformats-officedocument.wordprocessingml.document":  return "docx"  case "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet":  return "xlsx"  case "application/vnd.openxmlformats-officedocument.presentationml.presentation":  return "pptx"  case "application/pdf":  return "pdf"  case "application/zip":  return "zip"  case "application/x-rar":  return "rar"  case "application/x-7z-compressed":  return "7z"  case \_:  return None  def process\_part(part, eml\_filename, index, output\_dir):  if part.is\_multipart():  for j, subpart in enumerate(part.get\_payload()):  new\_index = f"{index}\_{j}"  process\_part(subpart, eml\_filename, new\_index, output\_dir)  else:  try:  content\_type = part.get\_content\_type()  file\_ext = get\_file\_extension(content\_type)  if not file\_ext:  print(f"Skipping part {index} ({content\_type})")  return  filename = os.path.join(  output\_dir, f"{eml\_filename}\_{index}.{file\_ext}")  os.makedirs(os.path.dirname(filename), exist\_ok=True)  payload = part.get\_payload(decode=True)  if payload is None:  print(f"Empty payload in part {index}")  return  with open(filename, "wb") as f:  f.write(payload)  print(f"Saved {os.path.basename(filename)}")  except Exception as e:  print(f"Error processing part {index}: {str(e)}")  def main(eml\_filename, input\_dir, output\_dir):  eml\_path = os.path.join(input\_dir, eml\_filename)  try:  with open(eml\_path, "rb") as f:  msg = email.message\_from\_binary\_file(f, policy=policy.default)  except Exception as e:  print(f"Failed to process {eml\_filename}: {str(e)}")  return  if not msg.is\_multipart():  content\_type = msg.get\_content\_type()  file\_ext = get\_file\_extension(content\_type) or "bin"  filename = os.path.join(output\_dir, f"{eml\_filename}.{file\_ext}")  os.makedirs(output\_dir, exist\_ok=True)  try:  payload = msg.get\_payload(decode=True)  with open(filename, "wb") as f:  f.write(payload)  print(f"Saved {os.path.basename(filename)}")  except Exception as e:  print(f"Error saving main content: {str(e)}")  else:  for i, part in enumerate(msg.get\_payload()):  process\_part(part, eml\_filename, str(i), output\_dir)  if \_\_name\_\_ == "\_\_main\_\_":  input\_dir = "emls"  output\_dir = "eml1"  if not os.path.exists(input\_dir):  print(f"Missing input directory: {input\_dir}")  exit(1)  if not os.path.exists(output\_dir):  print(f"Missing output directory: {output\_dir}")  exit(1)  for eml in os.listdir(input\_dir):  if os.path.isfile(os.path.join(input\_dir, eml)):  main(eml, input\_dir, output\_dir) |

1. Аналитический центр НАФИ, Каждый третий представитель МСП использует искусственный интеллект в работе, URL: https://nafi.ru/analytics/kazhdyy-tretiy-predstavitel-msp-ispolzuet-iskusstvennyy-intellekt-v-rabote/ [↑](#footnote-ref-1)