Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Прикладной информатики**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

Образовательная программа **Языковые модели и искусственный интеллект**

К У Р С О В О Й П Р О Е К Т

Тема: «Разработка телеграмм бота для анализа YouTube видео при помощи Искусственного Интеллекта»

Обучающийся: Короткова Алиса Александровна, K3160

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный мир переполнен информацией, в том числе и видеоконтентом. Как известно, видеоконтент становится всё более и более популярным. Миллионы людей ежедневно смотрят обучающие, новостные и развлекательные видео. YouTube является крупнейшей платформой для размещения видео, и ежедневно на нее загружаются миллионы часов видеоматериалов. Однако у всех нас есть одна общая проблема – время. Время – самый ценный ресурс, из всех, которыми мы располагаем. То, на что мы его тратим, определяет наше будущее. Это подтверждает старинное суждение: “Самое ценное, что вы можете подарить кому – либо – это свое время. Ты отдаешь то, что никогда не сможешь вернуть”. Поиск нужной информации в этом массиве данных может быть трудоемким процессом. Не всегда есть возможность просматривать длинные ролики, чтобы получить нужную информацию. Разработка инструмента, способного извлекать суть из видео и предоставлять краткое содержание, является актуальной задачей, позволяющей экономить время пользователей и снизить когнитивную нагрузку.

Такой инструмент пригодится каждому, особенно студентам, которым часто приходится просматривать видеоролики или лекции онлайн. Также такой бот поможет людям с ограниченными возможностями, например, для тех, кому трудно воспринимать длинные видео, краткое содержание может облегчить понимание информации.

Цель нашего проекта – разработать Telegram бота для анализа и получения краткого содержания YouTube видео при помощи Искусственного Интеллекта (нейросети YandexGPT).

Перед началом работы над проектом были поставлены следующие задачи:

1) Создать Telegram бота с помощью библиотеки aiogram,
2) Создать и запустить веб сервер с использованием Flask,
3) Написать функцию получения субтитров – обработчик ссылок,

4) Реализовать работу с сервером – запрос со стороны Telegram бота,
5) Реализовать работу с Искусственного Интеллекта – запрос к LLM модели,
6) Протестировать и наладить работу бота.

Также хочется обосновать выбор Telegram в качестве платформы для реализации проекта. Во-первых, Telegram – это популярный и доступный мессенджер с широкой аудиторией, что обеспечивает возможность охвата большого числа пользователей. Во-вторых, Telegram Bot API предоставляет удобный и хорошо документированный инструмент для разработки ботов, а также множество готовых библиотек, что упрощает процесс создания. В-третьих, функциональность Telegram ботов, в том числе возможность отправки текстовых сообщений, соответствует задаче предоставления краткого содержания видео. Telegram является мобильной платформой, что позволяет пользователям получить доступ к боту где угодно, что делает его использование удобным и доступным. Таким образом, Telegram бот – это оптимальное решение для реализации данного проекта.

1 Суть проекта

Сутью данного проекта является разработка Telegram бота, который, получив от пользователя ссылку на YouTube видео, осуществляет ряд последовательных операций: во-первых, проверяет корректность ссылки и извлекает id видео. Во-вторых, отправляет запрос на сервер для обработки, который, в свою очередь, получает субтитры из видео, преобразует их в текст, и на основе полученного текста с помощью методов искусственного интеллекта, используя нейросеть Yandex GPT, создает краткое содержание видео. Затем бот отправляет сгенерированный текст обратно пользователю в Telegram. Основная цель проекта – предоставить пользователю удобный и эффективный инструмент для быстрого ознакомления с содержанием видеороликов. Программное решение разделилось на 2 части: телеграмм бот и backend (веб сервер). Наглядную схему работы бота можно посмотреть на Рисунке 1.

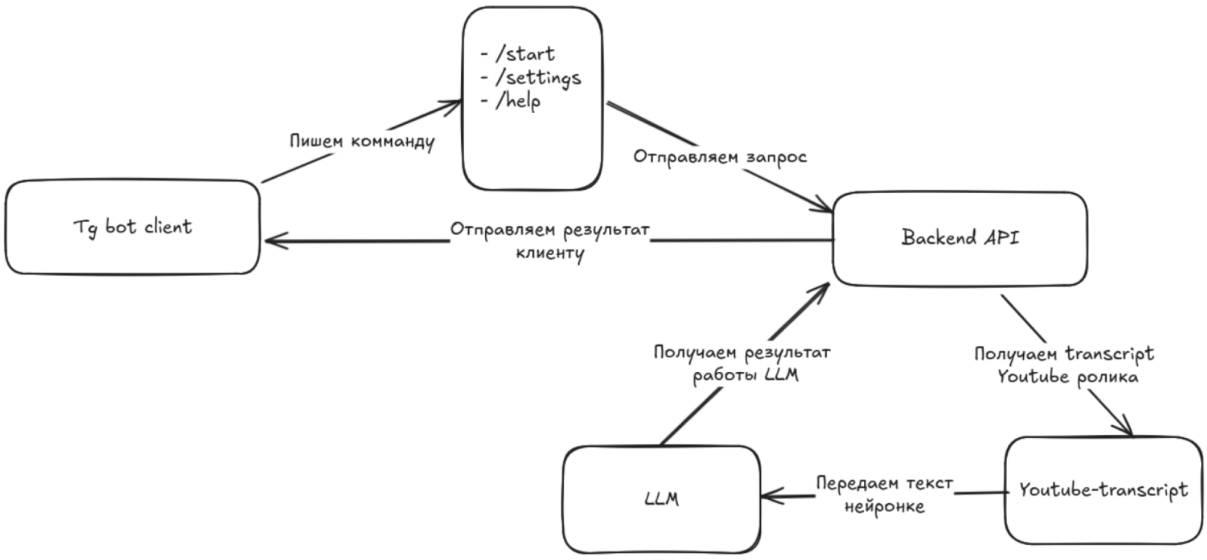

Рисунок 1 – Схема работы Телеграм бота

Система построена на основе клиент-серверной архитектуры, где Telegram бот выступает в качестве клиента, а Backend API – в качестве сервера, который взаимодействует с другими сервисами для обработки запросов. Данные в системе передаются в форме текстовых сообщений или запросов API.

Tg bot client (Telegram бот клиент) – это интерфейс, через который пользователь взаимодействует с системой. Пользователь отправляет команды

боту через Telegram. Бот обрабатывает текстовые сообщения, введенные пользователем, и распознает специальные команды, например, /start, /settings, /help. После обработки запроса бот отправляет результат пользователю в виде текстового сообщения.

Backend API – это центральный компонент, который отвечает за обработку запросов от бота, координацию работы других сервисов и предоставление данных обратно боту. Он получает запросы от бота и определяет дальнейшие шаги, обращается к LLM (Large Language Model) и сервису YouTube-transcript (получение транскрипции видео) для обработки данных, формирует ответ на основе данных, полученных от LLM и сервиса YouTube-transcript, и отправляет его Telegram боту.

LLM (Large Language Model) – представляет собой большую языковую модель, которая используется для генерации текстовых ответов, обработки естественного языка или других задач, связанных с текстом. API передает LLM задачу на обработку. После обработки LLM генерирует текстовый ответ и возвращает его в Backend API. В нашем проекте мы решили использовать нейросеть Yandex GPT.

Youtube-transcript – сервис, который предназначен для получения текстовой расшифровки (транскрипции) видео с YouTube.

Таким образом, бот решает проблему избытка видеоинформации, предоставляя возможность эффективного доступа к содержанию. Использование современных технологий, таких как асинхронные запросы и обработка данных в формате JSON, повышает производительность и надежность работы бота.

2 Процесс работы над проектом

На начальном этапе была четко определена цель проекта – создание Telegram бота для анализа YouTube видео и предоставления краткого содержания. Были также сформулированы задачи, включающие изучение технологий, разработку отдельных модулей (функций), интеграцию и тестирование.

На этапе планирования было проведено исследование предметной области, включающее анализ существующих решений для получения краткого содержания текста, изучение API YouTube и Telegram Bot API, а также анализ возможностей моделей искусственного интеллекта. Был разработан подробный план работ, включающий в себя разделение проекта на отдельные этапы, также был осуществлен процесс распределения ролей и задач между членами команды, с учетом их навыков и предпочтений. Еще были выбраны следующие технологии: Python как основной язык программирования, Telegram Bot API для интеграции с мессенджером, requests для отправки HTTP запросов, и модели искусственного интеллекта для суммирования текста. Изначально наш руководитель планировал писать бота на JavaScript, но оказалось, что мы все знакомы с Python намного лучше, поэтому мы приняли решение писать на нем, чтобы наша работа была более эффективной.

Затем следовал этап разработки. Для начала мы должны были создать сервер и бот отдельно. Сервером занимался Дмитрий Афанасьев, он быстро справился со своей задачей, и мы начали выполнять свои. Ксения создала бот и основную логику, я настроила связь между сервером и ботом, Дмитрий Грачев написал функции для обработки запросов на сервере, а Анастасия работала с нейросетью.

После слияния всех наших веток и репозиториев был проведен этап тестирования, включающий в себя проверку функциональности каждого модуля (валидация, отправка запроса, обработка запроса, получения id, получение транскрипции видео, получение краткого содержания, отправка ответа пользователю) по отдельности, а также тестирование всей системы в

целом. В процессе тестирования были выявлены ошибки, связанные с некорректной обработкой ответов сервера и передачей данных, но они были устранены.

Финальным этапом работы над проектом была защита, мы всей командой готовились к ней, еще раз проверили работоспособность бота и убедились в том, что все работает исправно.

3 Проблема (задача) и ее решение

Передо мной была поставлена задача – написать запрос к ручке (route) веб сервера и отправлять его по мере поступления ссылки на youtube, выполняя проверку на принадлежность ссылки к YouTube, обрабатывать ответ сервера и отправлять текст видео пользователю. Простыми словами – настроить связь между ботом и сервером.

Для выполнения задачи предварительно я изучила основы aiogram и асинхронных функций, которые используются для создания телеграмм ботов, потому что моя часть работы тесно связана не с backend частью, а с ботом. Я работала с репозиторием tg-youtube-shortener, где создала свою ветку, чтобы, пока работаю с кодом, не нарушить работу общей программы, и вносить правки только в свою отдельную ветку.

Основной алгоритм моей задачи – пользователь отправил ссылку, ссылка проверяется на принадлежность к youtube, отправляется POST request на сервер Flask с body, получаем и обрабатываем ответ веб сервера и отправляем текст видео пользователю.

POST (или POST request) — это один из нескольких методов HTTP (Hypertext Transfer Protocol), которые используются для взаимодействия между клиентом (например, Telegram ботом) и сервером. POST запросы используются для отправки данных на сервер с целью их обработки и хранения [1]. В нашем проекте они служат для отправки ссылки на YouTube видео на сервер, где происходит анализ и формирование краткого содержания.

Таким образом, сначала я реализовала функцию валидации ссылки, то есть функцию проверки принадлежности ссылки к YouTube, для этого я использовала регулярные выражения, которые мне позволили зафиксировать первую часть ссылки и проверять оставшуюся (id video).

Регулярные выражения (regex) – это мощный инструмент для поиска и обработки текстовой информации на основе заданных шаблонов. Они представляют собой последовательность символов, которая определяет шаблон поиска в тексте. Регулярные выражения позволяют эффективно

решать задачи поиска, замены и проверки строк, существенно упрощая обработку текстовых данных [2].

В контексте нашего проекта регулярные выражения применяются для проверки формата ссылки на YouTube: перед отправкой ссылки на сервер для обработки, необходимо убедиться, что ссылка действительно ведет на YouTube видео. Для этого используется регулярное выражение, которое проверяет, соответствует ли введенная пользователем строка определенному формату URL адреса YouTube. Шаблон включает в себя проверку наличия ключевых элементов URL, таких как домен youtube.com, а также наличия необходимых параметров, определяющих видео, например кода, состоящего из цифр и букв, который далее используется как id видео.

Это регулярное выражение я поместила в переменную regex, которая используется при проверке. Программное решение можно посмотреть на Рисунке 2.

```
YOUTUBE_REGEX = re.compile(
    r'(https?://)?(www\.)?(youtube\.com|youtu\.be)/(watch\?v=[\w-]+|[\w-]+|playlist\?list=[\w-]+)'
)

def is_valid_youtube_url(url):
    youtube_regex = r"^https://www\.youtube\.com/watch\?v=[a-zA-Z0-9_-]{11}$"
    return bool(re.match(youtube_regex, url))


# Сюда ли это писать, или в бот или в отдельный файл?
# Дописала часть Ксюши, возможно стоит вынести отельно часть summarize
@router.message()
async def process_any_message(message: Message):
    text = message.text
    if match := YOUTUBE_REGEX.search(text):
        youtube_url = match.group(0)
        if youtube_url and is_valid_youtube_url(youtube_url):
            try:
                video_text = await summarize_command(youtube_url)
                await message.reply(video_text)
            except Exception as e:
                await message.reply(f"Error processing YouTube link: {e}")
        elif youtube_url:
            await message.reply("Invalid YouTube URL")
        else:
            await message.reply("I don't understand this message.")
    else:
        await message.answer("В вашем сообщении нет YouTube ссылки.")
```

Рисунок 2 – Реализация валидации YouTube ссылки

Следующим этапом в реализации функциональности Telegram бота была отправка запроса на веб сервер для обработки полученной от пользователя ссылки на YouTube-видео. Для этой цели использовался HTTP-метод POST.

Как уже отмечалось, метод POST используется для отправки данных на сервер, в отличие от метода GET, который применяется для запроса данных с сервера. При использовании POST запроса, данные передаются в теле запроса,

а не в URL, что позволяет передавать больший объем информации и делает передачу более безопасной, особенно при отправке конфиденциальных данных. В контексте нашего проекта использование POST запроса было обусловлено необходимостью передачи URL адреса YouTube видео на сервер для его последующей обработки.

Для отправки HTTP запроса на сервер в нашем проекте была использована библиотека requests [3] для языка программирования Python. Эта библиотека является широко распространенным инструментом для выполнения HTTP запросов и предоставляет удобный интерфейс для работы с различными аспектами HTTP протокола.

Для отправки POST запроса с использованием requests, используется метод requests.post(). Этот метод принимает несколько аргументов, одним из которых является data. Аргумент data предназначен для передачи данных, которые будут отправлены в теле POST запроса [4].

В целях повышения производительности и улучшения пользовательского опыта, запросы к серверу были реализованы асинхронно. Асинхронная обработка позволяет избежать блокирования главного потока выполнения программы, позволяя боту обрабатывать другие задачи параллельно. Это было достигнуто с помощью использования асинхронной сессии (session) и метода session.post(). Асинхронный вызов session.post() не блокирует программу, пока запрос не будет выполнен [5]. Это достигается с использованием механизмов асинхронного ввода и вывода, которые позволяют программе продолжать выполнение других задач, пока она ожидает ответа от сервера.

После отправки POST запроса, сервер обрабатывает полученные данные и отправляет ответ. Он включает в себя коды состояния (например, 200 OK), заголовки и тело ответа (данные). Полученный от сервера ответ был преобразован в объект JSON с помощью метода response.json(). Это позволило нам легко получить доступ к данным, содержащимся в JSON сообщении, то есть к краткому содержанию видео.

При отправке запроса и получении ответа могут возникнуть различные ошибки, например, сетевые ошибки (проблемы с подключением к интернету или с доступностью сервера), ошибки сервера (сервер может вернуть ошибку, например, в случае проблем с его работой или при неправильной обработке запроса), ошибки валидации (некорректный формат данных, отправленных в запросе), ошибки обработки ответа (неожиданный формат ответа или проблемы при его обработке) [6].

Для того чтобы обеспечить надежную работу Telegram бота, в коде была реализована обработка всех возможных ошибок. В случае возникновения ошибки, бот отправляет пользователю соответствующее сообщение. Это не только улучшило надежность работы бота, но и сделало пользовательский опыт более приятным и понятным. Программное решения для этой задачи можно увидеть на Рисунке 3.

```python
#Функция отправления запроса к серверу для получения текста видео
async def video_summarize(session, url):
    try:
        async with session.post("http://127.0.0.1:5000/summarize", json={"video_url": url}, timeout=10) as response:
            if response.status == 200:
                try:
                    data = await response.json()
                    return data.get("summary", "Server didn't return video text")
                except aiohttp.ContentTypeError:
                    logger.error(f"Server returned non-JSON response for URL: {url}")
                    return f"Error: Server returned non-JSON response"
            else:
                logger.error(f"Server returned status code {response.status} for URL: {url}")
                return f"Error: Server returned status code {response.status}"
    except asyncio.TimeoutError:
        logger.error(f"Timeout while fetching video text for URL: {url}")
        return "Error: Request timed out"
    except aiohttp.ClientError as e:
        logger.error(f"Error fetching video text for URL: {url}, Error: {e}")
        return f"Error: {e}"

async def summarize_command(youtube_url):
    youtube_regex = r"^https://www\.youtube\.com/watch\?v=[a-zA-Z0-9_-]{11}$"
    if not re.match(youtube_regex, youtube_url):
        return "Invalid YouTube URL"

    async with aiohttp.ClientSession() as session:
        result = await video_summarize(session, youtube_url)
        return result
```

Рисунок 3 – Реализация отправления запроса

4 Анализ результатов и процесса работы

Мне удалось успешно реализовать функциональность отправки запросов к веб серверу. Мой код корректно обрабатывает ссылки на YouTube, проверяет их принадлежность к платформе, отправляет POST запрос на сервер и обрабатывает ответ, преобразовывая его в формат JSON. Также была реализована передача данных пользователю.

Реализованная проверка ссылок надежно отсеивает не-YouTube ссылки, обеспечивая работу остального кода. Пользователь получает сообщение, если ссылка не удовлетворяет условия.

Я смогла реализовать обработку ответов сервера, включая успешные ответы с текстовым содержанием видео. Также я предусмотрела обработку возможных ошибок при отправке запроса и при получении ответа. В таких случаях бот выдает пользователю соответствующее сообщение об ошибке.

На начальном этапе у меня возникли трудности с пониманием API веб сервера и формата данных, которые ожидались в запросе и в ответе. Потребовалось время, чтобы разобраться со структурой JSON запросов и ответов. Также возникли проблемы с асинхронным кодом. Я впервые работала с асинхронными запросами, поэтому потребовалось время и дополнительные материалы, чтобы разобраться со всеми нюансами реализации. В частности, с использованием async и await. Я не смогла реализовать функцию автоматического повтора запроса при получении ошибки от сервера, поэтому пользователю нужно заново отправлять запрос. На это не хватило времени и знаний.

Мне удалось работать планомерно, но иногда было сложно выделить время на работу над проектом из-за нагрузки другими предметами (особенно математикой). Созвоны помогли с этим справиться, мы обсуждали наши планы и задачи.

В процессе выполнения проекта я приобрела новые навыки в работе с HTTP запросами, API, обработке JSON, отладке асинхронного кода и обработке ошибок, изучила библиотеку aiogram, научилась работать с

Telegram ботами. Также я улучшила свое понимание архитектуры ботов и взаимодействия между клиентом и сервером.

Этот проект дал мне ценный опыт в практическом применении полученных знаний и помог улучшить мои навыки программирования на Python. Я стала лучше понимать, как работает код, и как находить решения возникающих проблем.

Наша команда успешно справилась со своими задачами, мы смогли создать Telegram бота для анализа YouTube видео. Пример работы бота можно увидеть на Рисунке 4.

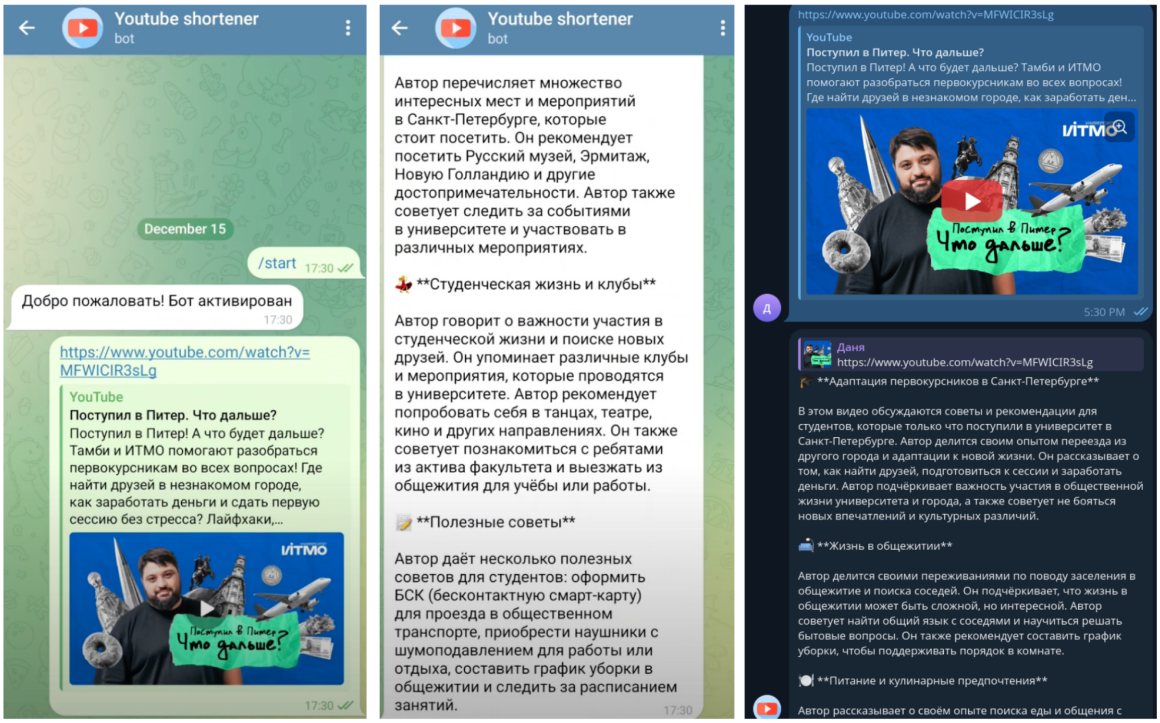

Рисунок 4 – Telegram bot YouTube shortener

5 Взаимодействие с командой и руководителем

Со многими в команде я уже была знакома до работы над проектом, потому что у нас пересекались пары, поэтому мы сразу начали слаженно работать. На первом созвоне команды мы рассказали о себе и о своих навыках, о своих представлениях и ожиданиях от проекта. В начале проекта задачи распределялись на основе навыков и предпочтений каждого участника команды. В нашем проекте каждый участник прикоснулся к коду, а то есть смог поучаствовать в разработке самого Telegram бота. Дмитрий Грачев и Дмитрий Афанасьев занимались серверной частью (backend), они создали наш сервер с помощью Flask и написали основную логику обработки данных (обращение и получение транскрипции видео и отправка обратно на сервер). Анастасия Лебедева реализовывала команду получения краткого содержания, она провела исследование нескольких нейросетей и выбрала Yandex GPT, затем прописала как сервер должен обращаться к искусственному интеллекту и с каким запросом. Ксения Сигаева занималась самим Telegram ботом, она его создала и вбила нужные команды, занималась внедрением других языков и обработкой сообщений пользователя. А я реализовывала логику соединения нашего сервера и бота, отправления запроса к серверу и получения ответа.

Наше взаимодействие происходило в чате в Телеграм и на платформе для проектов, каждый мог задать любой вопрос и получить на него ответ, также все проблемы, возникшие в ходе работы, обсуждались на созвонах. Так, когда я работала над своей частью проекта, я обратилась за помощью к Дмитрию Грачеву, я попросила сделать review моего кода и дать комментарии. С кодом все было отлично, но работа в паре тоже была интересной. Также я всегда координировала свою работу с Ксенией, потому что мы работали в одном репозитории Git, где хранился код проекта, что позволяло отслеживать изменения и работать над кодом параллельно. Но стоить отметить, что мне важно было знать, что происходит с сервером, потому что я писала к нему запросы. Каждый участник нес ответственность за выполнение поставленных

перед ним задач, а также за своевременное предоставление результатов – все успешно справились с этим, все соблюдали дедлайны.

В целом, я оцениваю работу команды как эффективную. Мы успешно достигли поставленных целей и своевременно завершили все запланированные задачи. Тем не менее, в будущем можно было бы лучше наладить процесс тестирования и проведения code review.

Руководитель провел первый созвон, он был ознакомительным, мы рассказали о себе, послушали еще раз планы Александра Коровина на проект и поделились своими ожиданиями от работы. После взаимодействие с руководителем проекта было регулярным и плодотворным. Мы общались еженедельно (1 или 2 раза в неделю) на общих созвонах в Telegram, используя их для обсуждения технических вопросов, планирования и отчета по уже выполненным задачам. Руководитель оказывал существенную поддержку, консультировал по сложным вопросам и давал ценные советы, он делился ссылками на хорошие ресурсы, много рассказывал о своем опыте работы в проектах и учебе в университете. Он также регулярно следил за прогрессом проекта, напоминал нам о дедлайнах и всегда был готов помочь. Также Александр познакомил нас со своим другом, разработчиком, который тоже поделился своим опытом в сфере IT и дал много интересных советов.

Я рада, что попала к нему в команду, наш руководитель смог хорошо организовать работу. Он правильно распределил задачи между нами, что в итоге привело нас к запланированному результату.

ЗАКЛЮЧЕНИЕ

Цель проекта, которая заключалась в разработке Telegram бота, способного анализировать YouTube видео и предоставлять пользователю краткое содержание, была успешно достигнута. Разработанный бот корректно выполняет все основные функции. Задачи, поставленные в начале проекта, были выполнены. Это включает в себя изучение необходимых технологий, разработку отдельных функций бота, тестирование и отладку. Наша команда работала слаженно, каждый с умом подошел к решению задач, поэтому я уверена, что мы успешно справились с проектом.

Мой личный вклад в проект заключается в реализации модуля, отвечающего за отправку HTTP запросов к веб серверу, проверку ссылок и обработку ответов. Я также занималась поиском и исправлением ошибок в коде, помогала другим членам команды, если у них возникали проблемы. Мой вклад был важен для работы бота, так как я обеспечила его связь с веб сервером, реализуя ключевую функциональность запросов. От правильной работы моего кода зависела возможность получения данных с сервера и, соответственно, корректная работа всего бота. Поэтому я с большой ответственностью отнеслась к работе и старалась выполнить ее на максимум.

Работа над проектом стала для меня отличной возможностью значительно расширить свои знания программирования. Я получила практический опыт в создании и отправке HTTP запросов, глубоко изучила принципы работы API, а также освоила методы обработки данных в формате JSON. Помимо этого, я усовершенствовала свои навыки отладки кода и научилась эффективно обрабатывать различные ошибки, которые могут возникать в процессе взаимодействия с сервером. Знакомство с библиотекой aiogram позволило мне освоить разработку Telegram ботов. Этот проект предоставил мне бесценный опыт применения теоретических знаний на практике, благодаря чему я существенно улучшила свои навыки программирования на Python.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Методы HTTP: GET, POST, PUT и другие. // Хекслет – URL: https://ru.hexlet.io/blog/posts/metody-http-get-post-put-i-drugie (дата обращения: 10.11.2024).
2) Регулярные выражения в Python от простого к сложному. Подробности, примеры, картинки, упражнения. // Хабр – URL: https://habr.com/ru/articles/349860/ (дата обращения: 10.11.2024).
3) Модуль requests. // Яндекс образование – URL: https://education.yandex.ru/handbook/python/article/modul-requests (дата обращения: 13.11.2024).
4) Краткое руководство по библиотеке Python Requests. // PythonRu – URL: https://pythonru.com/biblioteki/kratkoe-rukovodstvo-po-biblioteke-python-requests (дата обращения: 13.11.2024)..
5) Асинхронные методы. // JavaRush – URL: https://javarush.com/quests/lectures/ru.javarush.python.core.lecture.level14.lecture02 (дата обращения: 15.11.2024).
6) Полное руководство по модулю asyncio в Python. // Хабр – URL: https://habr.com/ru/companies/wunderfund/articles/700474/ (дата обращения: 15.11.2024).

ПРИЛОЖЕНИЕ

Техническое задание

Наименование проекта

Разработка телеграмм бота для анализа YouTube видео при помощи ИИ

Исполнитель

Коровин Александр Игоревич

Сроки выполнения

11.2024 (начало: месяц, год) – 12.2024 (окончание: месяц, год)

Цель проекта

Разработать телеграмм-бота для анализа YouTube видео с использованием искусственного интеллекта.

Ключевые слова и словосочетания

Анализ видео YouTube, API взаимодействия с нейросетью, Искусственный интеллект, Сводка содержания видео, Таймкоды ключевых моментов видео, Мультиязыковая поддержка (русский и английский языки), Телеграмм-бот (конечный продукт).

Требования к проекту

Бот должен принимать ссылки на ютуб видео и обрабатывать их. после обработки, юзер при помощи команд должен взаимодействовать с ботом и получать краткую информацию из видео. Также пользователь моет прочитать инструкцию по использованию бота.

Бот должен работать во всех официальных приложениях Telegram (мобильных, веб- и десктопных версиях).

Бот должен проверять валидность ссылок и выдавать сообщение об ошибке, если ссылка недействительна. Пользовательский интерфейс бота должен быть интуитивно понятным.

Содержание работы (этапы по срокам) в Таблице 1

Таблица 1 – Этапы работы

Этап работы	Срок выполнения	Документ о выполнении этапа

Получить необходимое оборудование и ПО, API	05.11.2024	Выписка о получении API
Реализовать доступ к YouTube видео	10.11.2024	График подключения к youtube.com
Разработать оболочку бота (Python, YandexGPT)	15.11.2024	Ссылка на бот
Написать алгоритм анализа видео	30.11.2024	Отчет о работе алгоритма
Внедрить алгоритм анализа в боте	02.12.2024	Ссылка на бот
Создать удобный пользовательский интерфейс	10.12.2024	Ссылка на бот
Разместить бота на сервере	12.12.2024	Ссылка на бот

Основной результат работы и форма его предоставления

Результат: Телеграмм-бот для анализа YouTube видео, предоставляющий краткую сводку материала на языке запроса

Форма представления: Готовый к использованию бот, доступный в Telegram.

Исполнитель: Коровин Александр Игоревич

Заказчик: Горлушкина Наталия Николаевна