

**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

КУРСОВОЙ ПРОЕКТ

Тема: «Разработка телеграмм бота для анализа Youtube видео при помощи ИИ»

Обучающийся: Грачев Дмитрий Александрович, группа К3160

Санкт-Петербург 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Описание проекта.....	5
2 Работа над проектом.....	6
2.1 Процессы работы над всем проектом.....	6
2.2 Задачи, поставленные передо мной.....	6
2.3 Ход работы.....	7
3 Коммуникация в команде.....	13
ЗАКЛЮЧЕНИЕ.....	15
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	16
ПРИЛОЖЕНИЕ.....	17

ВВЕДЕНИЕ

В современном мире видеоконтент стал неотъемлемой частью нашей повседневной жизни. Каждый день миллионы людей просматривают обучающие видеоролики, следят за новостями и смотрят развлекательные видео на YouTube. Однако, несмотря на изобилие контента и его доступность, у нас всех есть одна общая проблема – ограниченность времени. Просмотр длинных видео требует значительных временных затрат, что не всегда возможно в условиях современного насыщенного ритма жизни. Более того, для извлечения конкретной информации из большого объема материала приходится тратить дополнительные усилия, что увеличивает когнитивную нагрузку.

Вдохновленные этой проблемой, наша команда поставила перед собой цель: разработать инструмент, который позволит эффективно взаимодействовать с видеоконтентом, экономить время пользователей и снизить нагрузку на восприятие и анализ информации. Мы стремились создать решение, которое не только обеспечит удобство использования видеоматериалов, но и поможет повысить продуктивность, предлагая краткое и точное содержание длинных видеороликов.

Главной целью нашего проекта было создание Telegram-бота, способного принимать ссылки на видео с YouTube, извлекать из них текстовую информацию, анализировать ее с помощью нейросети YandexGPT и возвращать пользователю лаконичное и информативное резюме.

Для реализации этой цели нам пришлось решить несколько ключевых задач:

- разработка и настройка сервера на основе Flask, обеспечивающего обработку запросов,
- создание функции, позволяющей получать субтитры из YouTube-видео,

- реализация механизма взаимодействия между сервером и другими компонентами системы,
- интеграция с нейросетью YandexGPT для анализа и обработки текстовой информации,
- создание удобного и функционального Telegram-бота с помощью библиотеки Aiogram.

Этот проект представляет собой синтез современных технологий и ориентирован на улучшение пользовательского опыта, позволяя легко и быстро получать суть из больших объемов видеоконтента. Мы уверены, что наше решение будет востребовано среди людей, ценящих своё время и стремящихся к максимальной эффективности в работе с информацией, а также среди людей, которые часто взаимодействуют с видеоконтентом.

1 Описание проекта

Итоговый продукт нашего проекта – Telegram-бот, который по запросу пользователя, возвращает ему лаконичное и информативное краткое содержание видеоролика с YouTube, полученное с помощью нейросети YandexGPT.

Работа бота строится на простой, но эффективной архитектуре, представленной на рисунке 1.

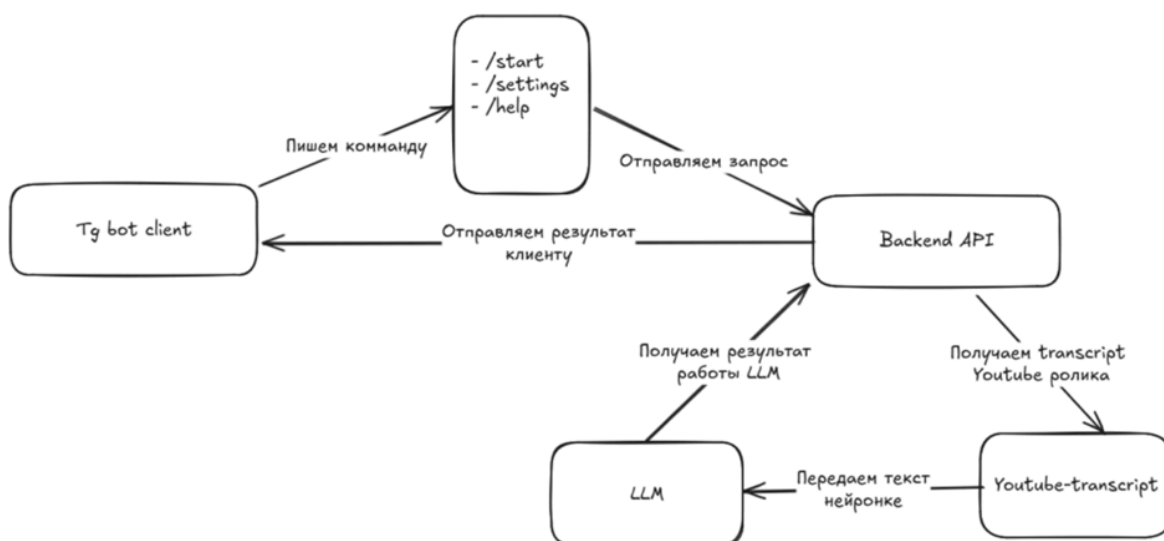


Рисунок 1 – схема работы Telegram-бота

Пользователь отправляет боту ссылку на YouTube-видео. Бот отправляет запрос на наш сервер. Сервер, в свою очередь, используя написанный нами модуль, получает субтитры с YouTube. Затем, текст субтитров передается в нейросеть YandexGPT, которая генерирует краткое содержание видео. Полученный результат отправляется обратно пользователю.

Важно отметить, что мы использовали асинхронную модель работы бота, что обеспечивает высокую производительность.

2 Работа над проектом

2.1 Процессы работы над всем проектом

Для обеспечения успешного выполнения проекта был составлен подробный и четкий план действий, который охватывает все ключевые этапы работы. Этот план служил ориентиром для всех участников команды и включал следующие шаги:

- 1) создание технического задания,
- 2) распределение задач и установка сроков их выполнения,
- 3) выполнение задач участниками команды,
- 4) тестирование и доработка получившейся программы,
- 5) подготовка к защите проекта, создание презентации,
- 6) командная защита проекта,
- 7) написание итогового отчета.

2.2 Задачи, поставленные передо мной

Конкретно передо мной стояла задача разработать модуль, который должен был принимать на вход ссылку на YouTube-видео, автоматически извлекать его субтитры и возвращать результат работы обратно в программу.

Ключевым требованием было то, чтобы функция была асинхронной. Это было необходимо для обеспечения высокой скорости выполнения операций, так как обработка больших объемов данных могла занимать значительное время. Кроме того, асинхронная реализация позволяла сохранить стабильную и эффективную работу всей программы даже при увеличении нагрузки или работе с несколькими запросами одновременно.

На этапе разработки я также столкнулся с рядом технических вызовов, включая интеграцию API для обработки субтитров, обеспечение корректного форматирования текста, а также оптимизацию производительности, чтобы пользователи могли получать результаты максимально быстро.

2.3 Ход работы

Для начала я поставил перед собой задачу создать базовую функцию, которая могла бы получать субтитры к видео по ссылке на YouTube. Это было первым шагом в разработке моего модуля. Для реализации функции я выбрал язык программирования Python. Этот выбор был обоснован тем, что Python является современным, популярным и мощным языком программирования, который предоставляет удобные инструменты для работы с искусственным интеллектом и обработкой данных. Более того, все участники нашей команды на момент начала работы уже имели опыт программирования на Python, что значительно облегчило процесс разработки нашего Telegram-бота.

Совместно с руководителем проекта было решено использовать для выполнения моей задачи библиотеку `youtube-transcript-api`. Моя работа началась с детального изучения документации библиотеки `youtube-transcript-api` [1], чтобы понять, как эффективно интегрировать ее возможности в наш проект. После изучения документации я разработал первый вариант функции. На данном этапе главной целью было реализовать функцию, которая могла бы извлекать субтитры из YouTube-видео. Этот вариант модуля представлен на рисунке 2.

```

def fetch_transcript(url: str) -> Optional[str]:
    """
    Асинхронно получает транскрипцию YouTube видео.
    Пытается найти транскрипцию на любом из поддерживаемых языков.

    Args:
        url (str): URL YouTube видео

    Returns:
        Optional[str]: Отформатированный текст транскрипции или None в случае ошибки
    """
    try:
        # Извлекаем ID видео из URL
        video_id_match = re.search(r"(?:v=|\/)([0-9A-Za-z_-]{11}).*", url)
        if not video_id_match:
            return None
        video_id = video_id_match.group(1)

        # Получаем транскрипцию видео
        transcript = YouTubeTranscriptApi.get_transcript(
            video_id,
            languages=LANGUAGES,
        )

        # Форматируем текст
        formatter = TextFormatter()
        formatted_text = formatter.format_transcript(transcript).replace("\n", " ")

        return formatted_text

    except Exception as e:
        print(f"Ошибка при получении транскрипции: {str(e)}")
        return None

```

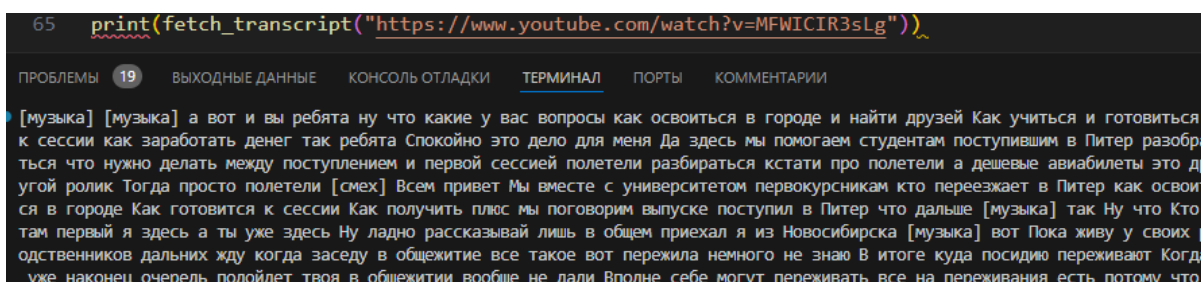
Рисунок 2 – первый вариант модуля получения субтитров

На этом этапе реализации модуль еще не был асинхронным, однако он успешно справлялся со своей основной задачей – извлечением субтитров из YouTube-видео. Используемая библиотека `youtube-transcript-api` предоставляла удобный интерфейс для работы с субтитрами, но она принимала на вход не саму ссылку на видео, а его уникальный идентификатор (ID). Поэтому для корректной работы модуля мне нужно было написать алгоритм извлечения ID видео из YouTube ссылки с помощью регулярных выражений.

Работа с регулярными выражениями была для меня новым опытом, поэтому я начал с изучения основ. Для этого я обратился к документации библиотеки регулярных выражений на Python [2] и к статье на Хабре под

названием “Регулярные выражения (regex) – основы” [3]. В ней подробно объяснялись базовые принципы и синтаксис регулярных выражений, что помогло мне быстро освоить этот инструмент. На основании полученных знаний я смог написать регулярное выражение, которое извлекало ID из ссылки на видео. Это решение позволило добиться совместимости моего модуля с полным спектром стандартных YouTube-ссылок, включая короткие ссылки и ссылки с параметрами.

Несмотря на отсутствие асинхронности, разработанный модуль показал себя надежным и эффективным. Он позволял получать текст субтитров и обрабатывать его для последующего использования. Пример вызова функции представлен на рисунке 3.



```
65 print(fetch_transcript("https://www.youtube.com/watch?v=MFWICIR3sLg"))
```

ПРОБЛЕМЫ 19 ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ КОММЕНТАРИИ

[музыка] [музыка] а вот и вы ребята ну что какие у вас вопросы как освоиться в городе и найти друзей Как учиться и готовиться к сессии как заработать денег так ребята Спокойно это дело для меня Да здесь мы помогаем студентам поступившим в Питер разобраться что нужно делать между поступлением и первой сессией полетели разбираться кстати про полетели а дешевые авиабилеты это другой ролик Тогда просто полетели [смех] Всем привет Мы вместе с университетом первокурсникам кто переезжает в Питер как освоится в городе Как готовится к сессии Как получить плюс мы поговорим выпуске поступил в Питер что дальше [музыка] так Ну что Кто там первый я здесь а ты уже здесь Ну ладно рассказывай лишь в общем приехал я из Новосибирска [музыка] вот Пока живу у своих родственников дальних жду когда заседу в общежитие все такое вот пережила немного не знаю В итоге куда посидию переживают Когда уже наконец очередь подойдет твоя в общежитии вообще не дали Вполне себе могут переживать все на переживания есть потому что

Рисунок 3 – пример работы первого варианта модуля получения субтитров

Такой подход стал хорошей основой для дальнейших улучшений, включая переход к асинхронной реализации и оптимизацию производительности.

Тема асинхронного программирования на языке Python была для меня новой и представляла собой интересный вызов, требующий дополнительного изучения. Асинхронность позволяет значительно повысить производительность приложения, особенно при выполнении операций ввода-вывода. Чтобы понять, как внедрить этот подход в свою работу, я решил обратиться к документации библиотеки `asyncio` [4] и к циклу статей на Хабре под названием “Асинхронный python без головной боли” [5]. Библиотека `asyncio` является стандартным, но достаточно мощным инструментом для создания асинхронных программ на Python.

Прочитав документация я смог понять базовые концепции асинхронного программирования на Python, такие как использование корутин, событийного цикла и задач (tasks). По мере чтения я узнал, как `asyncio` позволяет запускать несколько операций параллельно, не блокируя основное выполнение программы.

После изучения основ я приступил к модернизации своей функции. Основной целью было оптимизировать её работу с учётом особенностей асинхронного программирования. Это потребовало не только изменить структуру функции, но и пересмотреть её взаимодействие с библиотекой `youtube-transcript-api`. Теперь функция могла выполнять свою задачу, не блокируя другие процессы, что особенно важно при работе с большим количеством запросов или при интеграции в более сложные системы.

В результате моя функция была успешно преобразована в асинхронную. Это позволило значительно повысить её производительность и сделать модуль более гибким для использования в программе. Код финальной асинхронной версии моего модуля представлен на рисунке 4.

```

async def fetch_transcript(url: str) -> Optional[str]:
    """
    Асинхронно получает транскрипцию YouTube видео.
    Пытается найти транскрипцию на любом из поддерживаемых языков.
    Ограничивает выходной текст до 2000 токенов.

    Args:
        url (str): URL YouTube видео

    Returns:
        Optional[str]: Отформатированный текст транскрипции или None в случае ошибки
    """
    try:
        # Извлекаем ID видео из URL
        video_id_match = re.search(r"(?:v=|\/)([0-9A-Za-z_-]{11}).*", url)
        if not video_id_match:
            return None
        video_id = video_id_match.group(1)

        # Запускаем получение транскрипции в отдельном потоке через asyncio
        transcript = await asyncio.get_event_loop().run_in_executor(
            None,
            lambda: YouTubeTranscriptApi.get_transcript(
                video_id,
                languages=LANGUAGES,
            ),
        )

        # Форматируем текст
        formatter = TextFormatter()
        formatted_text = formatter.format_transcript(transcript).replace("\n", " ")

        return formatted_text

    except Exception as e:
        print(f"Ошибка при получении транскрипции: {str(e)}")
        return None

```

Рисунок 4 – финальная версия модуля получения субтитров

Могу с уверенностью сказать, что задача, поставленная передо мной руководителем проекта, была успешно выполнена. Благодаря тщательному подходу, постоянному изучению новых инструментов и вниманию к деталям, мне удалось разработать функциональный, эффективный и оптимизированный модуль. Созданный модуль не только отвечает всем требованиям, но и демонстрирует высокую производительность, что было особенно важно для успешной интеграции в наш проект.

Во время выполнения курсового проекта я приобрел множество ценных знаний и навыков. Я укрепил свои навыки программирования на Python, освоил работу с библиотекой youtube-transcript-api и научился эффективно использовать ее возможности для решения прикладных задач. Важным

этапом стало изучение регулярных выражений, что позволило мне разрабатывать алгоритмы для извлечения идентификатора видео из различных форматов ссылок. К тому же, я впервые столкнулся с асинхронным программированием, и благодаря изучению библиотеки `asyncio` я освоил концепции корутин, задач и событийных циклов, что позволило оптимизировать работу функции и повысить ее производительность.

3 Коммуникация в команде

Для удобства и эффективного взаимодействия между членами команды руководителем проекта была создана группа в Telegram. В этой группе мы регулярно общались, делились своими успехами и обсуждали текущий прогресс. Несколько раз в неделю проводились созвоны, где мы совместно анализировали выполненные задачи, планировали дальнейшие шаги и решали возникающие трудности. Такой формат общения позволял оставаться на одной волне, даже несмотря на то, что работа над проектом велась полностью удаленно.

К тому же, для организации работы над проектом и упрощения процесса разработки были созданы два репозитория на платформе GitHub. Первый репозиторий, `tg-youtube-shortener-backend`, отвечал за серверную часть нашего бота. Второй, `tg-youtube-shortener`, был посвящён клиентской стороне. Эта структура обеспечила чёткое разделение обязанностей и позволила команде эффективно сосредоточиться на выполнении своих задач. В процессе работы я неоднократно созванивался с моим коллегой, который также отвечал за `backend`. Совместные обсуждения помогали нам находить оптимальные решения, делиться идеями и успешно преодолевать возникающие сложности.

Отдельно хочу отметить работу нашего руководителя проекта – Александра Коровина. Александр не только организовал отличное взаимодействие внутри команды, но и всегда был готов поддержать, ответить на возникающие вопросы и предложить помощь в сложных ситуациях. Его открытость, внимательность и профессионализм сделали работу над проектом комфортной и продуктивной для всех членов команды. А благодаря его лидерским качествам, проект продвигался уверенно и был выполнен в задуманные заранее сроки.

Поэтому, с уверенностью могу сказать, что Александр заслуживает самой высокой оценки за свою работу. Его подход к руководству стал залогом

успешной реализации нашего проекта и положительного опыта разработки для каждого члена команды.

ЗАКЛЮЧЕНИЕ

Проект по разработке Telegram-бота для анализа YouTube-видео с использованием искусственного интеллекта можно считать успешно выполненным. Основная цель проекта – создание инструмента, который позволяет извлекать текстовую информацию из видеороликов и генерировать их краткое содержание с помощью нейросети YandexGPT – была достигнута. Все ключевые задачи, включая разработку серверной части, интеграцию с библиотеками для работы с субтитрами и создание удобного пользовательского интерфейса, выполнены в полном объеме.

Единственным препятствием, которое время от времени замедляло работу, стало освоение членами команды новых технологий, необходимых для выполнения поставленных перед нами задач. Однако благодаря изучению соответствующих документаций и правильно спланированной командной работе эти вызовы были успешно преодолены.

Мой личный вклад в достижение цели проекта заключался в разработке модуля для извлечения субтитров из YouTube-видео. Для реализации этого функционала мною была создана асинхронная функция на языке Python с использованием библиотеки `youtube-transcript-api`, что позволило достичь высокой производительности. Также я реализовал алгоритм для обработки различных форматов YouTube-ссылок с помощью регулярных выражений, что сделало модуль универсальным и удобным в использовании.

Таким образом, проект объединил усилия всех участников команды и стал важным опытом для каждого из нас, а полученный продукт готов к практическому использованию и может быть доработан и расширен в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Python Software Foundation. youtube-transcript-api – URL: <https://pypi.org/project/youtube-transcript-api/> (дата обращения 19.11.2024)
- 2) Python Software Foundation. re — Regular expression operations – URL: <https://docs.python.org/3/library/re.html> (дата обращения 20.11.2024)
- 3) Хабр. Регулярные выражения (regex) — основы – URL: <https://habr.com/ru/articles/545150/> (дата обращения 20.11.2024)
- 4) Python Software Foundation. asyncio — Asynchronous I/O – URL: <https://docs.python.org/3/library/asyncio.html> (дата обращения 26.11.2024)
- 5) Хабр. Асинхронный python без головной боли – URL: <https://habr.com/ru/articles/667630/> (дата обращения 26.11.2024)

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 может 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):

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