**НИТУ «МИСИС»**

**Кафедра инженерной кибернетики**

**ОТЧЕТ**

по

**ЛАБОРАТОРНОЙ РАБОТЕ**

*«Исследование возможностей и оценка качества генерации русскоязычных текстов NLP-моделями «семейства» ruGPT-3.х»*

учебная дисциплина «Методы искусственного интеллекта»

**Студент:** Ишмухамедов А.А.

**Группа**: БПМ-21-1

**Преподаватель: Хонер П. Д.**

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**2024 г.**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc187517250)

[1. ОПИСАНИЕ ЗАДАЧИ 4](#_Toc187517251)

[1.1. Основное содержание задачи 4](#_Toc187517252)

[1.2. Решение включает 4](#_Toc187517253)

[1.3. Особенности решения 5](#_Toc187517254)

[3. ВЫБРАННЫЕ СРЕДСТВА ДЛЯ РАЗРАБОТКИ 6](#_Toc187517255)

[4. ИСТОЧНИКИ ИСХОДНЫХ ДАННЫХ 8](#_Toc187517256)

[5. РЕЗУЛЬТАТЫ 10](#_Toc187517257)

[5.1. Сравнение API 14](#_Toc187517258)

[5.1.1. Качество распознавания 14](#_Toc187517259)

[5.1.2. Доступность и удобство работы с API и библиотеками 14](#_Toc187517260)

[5.1.3. Интеграция и документация 14](#_Toc187517261)

[6. ВЫВОДЫ 16](#_Toc187517262)

[ИСТОЧНИКИ 17](#_Toc187517263)

[ПРИЛОЖЕНИЕ А. Скриншоты с результатами Whisper 18](#_Toc187517264)

[ПРИЛОЖЕНИЕ Б. Скриншоты с результатами AssemblyAI 28](#_Toc187517265)

[ПРИЛОЖЕНИЕ В. Исходный код программы 38](#_Toc187517266)

ВВЕДЕНИЕ

Целью данной лабораторной работы является изучение возможностей современных русскоязычных NLP-моделей «семейства» ruGPT-3.х для генерации текстов. Работа направлена на выработку навыков взаимодействия с NLP-моделями на уровне программного кода, а также на анализ качества их работы в задаче генерации связного текста на заданную тему.

1. ОПИСАНИЕ ЗАДАЧИ
   1. Основное содержание задачи

Задача состоит в разработке приложения, способного взаимодействовать с NLP-моделями ruGPT-3.х для генерации осмысленного текста на русском языке. В качестве исходных данных используются стартовые фразы, задающие тематику и направленность генерируемого текста. Основное внимание уделяется качеству и связанности полученного текста.

* 1. **Постановка задачи**

1. Разработать программное обеспечение для взаимодействия с моделями ruGPT-3.х
2. Провести генерацию текстов на заданные темы:

* Краткая аннотация КНИР, длиной до 150 слов.
* Расширенный текст, связанный с КНИР, длиной не менее 300 слов. Получение текстовых результатов.
* Свободное общение

1. Оценить качество сгенерированных текстов по критериям: связность, осмысленность, релевантность тематике.
2. ВЫБРАННЫЕ СРЕДСТВА ДЛЯ РАЗРАБОТКИ

Для разработки программного обеспечения были использованы следующие инструменты и технологии:

1. NLP-модели: ruGPT-3 Small
2. Язык программирования Python — основной инструмент разработки, обеспечивающий гибкость интеграции и обработки данных.
3. Среда разработки: Visual Studio Code — кроссплатформенная интегрированная среда разработки, использованная для написания, отладки и тестирования программного кода.
4. Библиотеки:

* transformers — библиотека для работы с языковыми моделями, предоставляемая Hugging Face. Она включает предобученные модели, такие как ruGPT-3 Small, и средства для их использования, настройки и дообучения. transformers упрощает взаимодействие с современными NLP-моделями, предоставляя удобный интерфейс для выполнения задач, таких как генерация текста, перевод, анализ тональности и классификация.
* dotenv — для безопасного хранения конфиденциальной информации, включая API-ключи, в файле .env.
* PyQt5 — библиотека для создания графического пользовательского интерфейса приложения, предоставляющего пользователю возможность выбора API, загрузки файлов и отображения результатов. [4]

Использование данных инструментов позволило реализовать программное обеспечение, соответствующее требованиям поставленной задачи, а также обеспечить удобство работы пользователя в процессе тестирования.

Модель ruGPT-3 Small базируется на архитектуре GPT-2 и была предварительно обучена на последовательностях длиной 1024 токена. После этого модель была дообучена с учетом увеличенного контекста в 2048 токенов, что позволяет ей генерировать более связные и осмысленные тексты. Тренировка модели проводилась на корпусе данных объемом 80 миллиардов токенов, что включало три эпохи обучения. Полный процесс обучения занял около одной недели с использованием 32 GPU.

Данные для обучения модели включали разнообразные корпуса, распределение которых представлено на диаграмме (рис. 1):

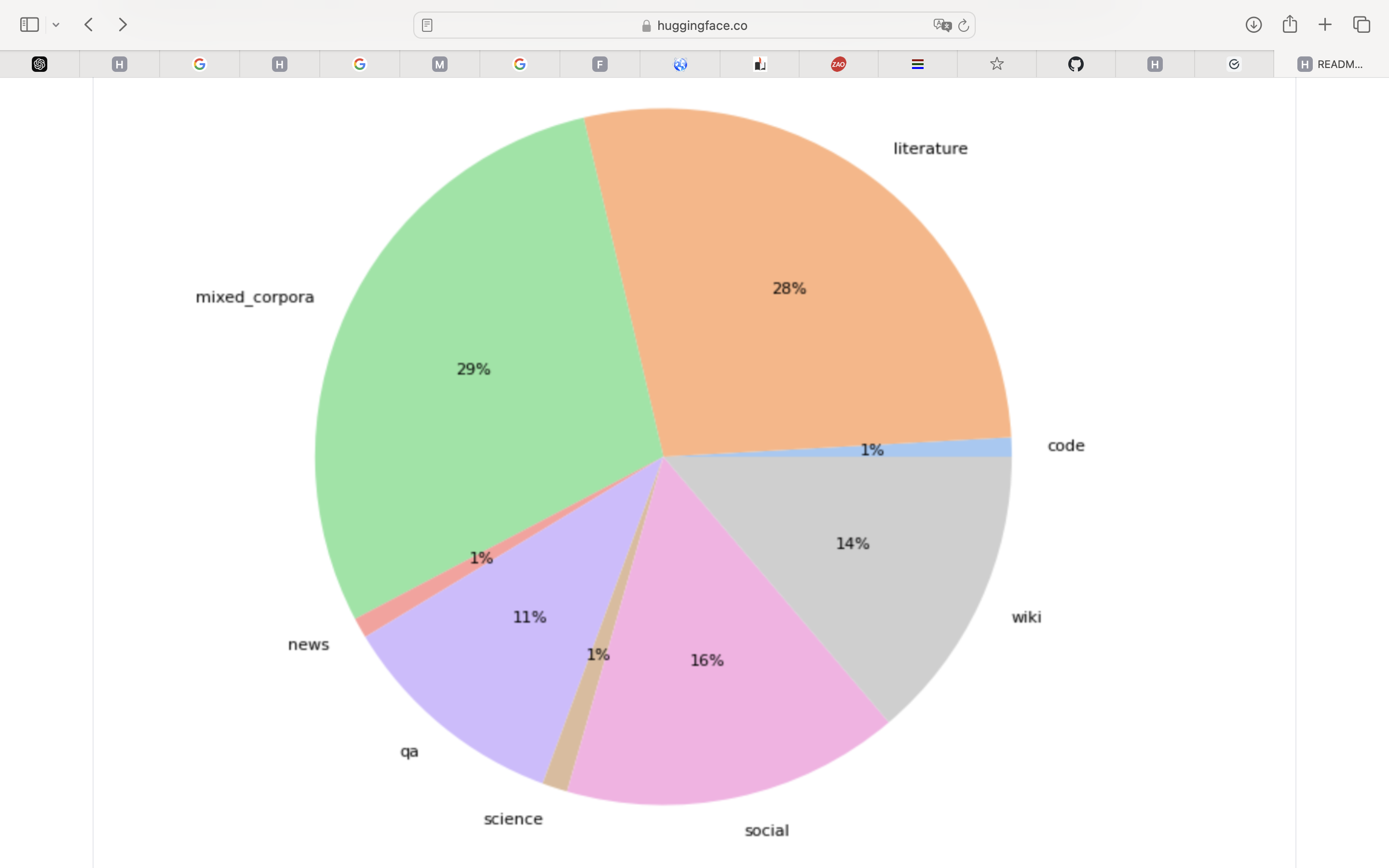


Рис.1. Распределение данных среди обучающей выборки

Диаграмма визуализирует распределение типов данных в обучающем корпусе модели ruGPT-3 Small.

Особенности ruGPT-3 Small:

* Компактность: оптимизирована для работы с ограниченными ресурсами.
* Подходит для задач с русскоязычными текстами.

Отличия от ruGPT-3 Large и 13B:

* Меньшее количество параметров (что делает модель быстрее и менее ресурсоемкой).
* Более узкий контекстный анализ по сравнению с большими версиями, но достаточный для базовых задач.

1. ИСТОЧНИКИ ИСХОДНЫХ ДАННЫХ

Для тестирования и определения качества генерации был взят КНИР студента Ишмухамедова Артёма Альбертовича за 7 семестр обучения на тему: «Исследование возможности применения методов статического анализа и машинного обучения для обнаружения аномалий в поведении смарт-контрактов». С помощью данной курсовой-научной работы проверялось качество генерации следующим образом:

* На основании содержания КНИР предложить модели сгенерировать аннотацию работы и текст, связанный с тематикой.
* Также для тестирования использовался простой вопрос: «Как метеорологи предсказывают погоду?».

Полученные результаты будут сравниваться с текстами, сгенерированными моделью ChatGPT 4.0. Анализ проводился по критериям по субъективному мнению автора:

* Связность текста.
* Осмысленность и логичность.
* Соответствие заданной тематике.

1. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ РАБОТЫ NLP-МОДЕЛИ ruGPT-3

Была разработана программа с простым графическим интерфейсом, с возможностью выбора api и выбора аудиофайла, а также с полем для вывода распознанного текста (рис. 2, рис. 3)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 2. Графический интерфейс программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 3. Выбор api в графическом интерфейсе

Программа была протестирована на указанных файлах. Скриншоты с результатами распознавания приведены ниже. Whisper и AssemblyAI успешно обработали все файлы, предоставив текст с высокой точностью, особенно записанный собственноручно файлы были расшифрованы успешно, точь-в-точь. Не помешали распознаванию и посторонние шумы на записях. Обе модели успешно справились с расшифровкой аудиофайлов в текст, содержащие отрывки из песен, но и Whisper и AssemblyAI плохо справляются с распознаванием сленга и англицизмов. Примером плохого распознавания является результат распознавание текста аудифайла содержащего отрывок из песни Fike & Jambazi “Minimun”, в котором AssemblyAI показала лучше результат. Отрывок речи 5-летнего ребенка с заиканием был распознан лучше чем отрывок содержащий сленг.

AssemblyAI API вызвал ошибку 500, что исключило возможность использовать AssemblyAI API, поэтому использовалась библиотека assemblyai. Исходный код программы можно посмотреть в репозитории github [5] в папке lab01 и в Приложении В.

Результаты работы Whisper соответствующие порядку файлов, описанных в предыдущем пункте, скриншоты с соответствующими результатами представлены в Приложении А:

1. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
2. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работу и выбрать лучший вариант.
3. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
4. Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
5. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
6. Тестовая запись под шумом. Попытаемся распознать мой голос. Это тестовая запись и голосовое сообщение. Попытаемся распознать мой голос с помощью приложения под шумом.
7. Мы с тобой шёпотом, шёпотом Спрашивали, что потом, что потом Будет шёпотом, шёпотом Не хочу кричать о том, что друг друга забудем
8. Можешь говорить о том, что я закрывал за собой двери, когда шел сюда и танцевал, для немного колорита, но шоу не кардинал телек, и кипятит это белая вдова в телек Утром в голове, под вечер спать ножом и вилкой, эту встречу не случайность, я общу линии за биткоин Два года назад я бы назвал это разминкой, сообщение в WhatsApp, но я потерял свой пин-код
9. We menu, ma'am, we're not a minimum, man, Man, fan of men, fan of nam, Senabhaughan men, we've, we've, even for one, even the same one, no, no, said, come, no, no, no, fight, no minimum, f'am'n't, a minimum, Crish, r'rv' not a minimum, vass, not minimum, you've seen, you've, you've, you've, we've, been, we've, , you know, we've, not, minimum,
10. Что тебе нравится в детском садике? Например, зашевелённость, нарушаясь. Он говорит, поёт, лёгко. Третья вопроса – кто? Не зашевелённость, как будто он живёт, молчит, не лает.

Результаты работы AssemblyAI соответствующие порядку файлов, описанных в предыдущем пункте, скриншоты с соответствующими результатами представлены в Приложении Б:

1. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл, выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
2. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работу и выбрать лучший вариант.
3. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
4. Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любое mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
5. Привет! Ты тестируешь приложение для преобразования голоса в текст. Просто загрузи любой mp3 файл и выбери один из двух API, чтобы увидеть результат. Это поможет тебе сравнить их работы и выбрать лучший вариант.
6. Тестовая запись под шумом. Попытаемся распознать мой голос. Это тестовая запись и голосовое сообщение. Попытаемся распознать мой голос помощью приложения.
7. Мы с тобой шёпотом, шёпотом Спрашивали, что потом, что потом будет Шёпотом, шёпотом Не хочу кричать о том, что друг друга забудем
8. Говорить о том, что я закрывал за собой двери Когда шел сюда, я танцевал, да и немного колорита Но шоу не гавновал телек, дикий аппетит Это белая вдова в телек Утром в голове под вечер спать ножом и вилкой Эту встречу не случайность, я общу линии за биткоин Два года назад я бы назвал это разминкой Сообщение в WhatsApp, но я потерял свой пин-код
9. Попала мне на минимум, фейк на минимум, нам не на минимум. Крышу рвет не на минимум, бас не на минимум.
10. Что тебе нравится в детском садике? Дрянь, например, зашевелённый, нарушаясь. Он разговаривает, поёт, вздрёхает. Дрянь на вопрос «Кто?» Он не зашевелённый, просто лежит, молчит, не лает.
    1. Сравнение API
       1. Качество распознавания

Качество распознавания речи в обоих решениях — **AssemblyAI** и **Whisper** — практически идентично. Обе системы обеспечивают высокую точность распознавания речи даже при наличии посторонних шумов или дефектов записи. Однако **AssemblyAI** показал себя лучше в распознавании сленга и нестандартных выражений, таких как англицизмы и жаргон, что делает его более подходящим для приложений, которые обрабатывают разговорную речь или речи с неформальными элементами.

* + 1. Доступность и удобство работы с API и библиотеками

Whisper API используется через платформу RapidAPI, что удобно с точки зрения интеграции и простоты работы с API. Однако количество бесплатных запросов мало.

AssemblyAI API на платформе RapidAPI вызывает ошибку 500. Поэтому использована библиотека AssemblyAI [2], библиотека предоставляет приятную и подробную документацию. Количество запросов в бесплатном плане достаточно большое, а платная подписка открывает дополнительные возможности и больше запросов. Также интерфейс личного кабинета отображает количество оставшихся запросов, что упрощает отслеживание и планирование работы.

* + 1. Интеграция и документация

С точки зрения разработчика, работа с библиотекой AssemblyAI более удобна и приятна, поскольку она предоставляет простой и читаемый код, легко интегрируется в проекты и позволяет тонко настроить параметры распознавания. Документация на AssemblyAI также является приятной для изучения, что облегчает её использование.

В случае с Whisper API через RapidAPI также не возникает серьёзных проблем, но для использования через API необходимо следить за ограничениями по количеству запросов. И корректность передачи параметров и заголовков

**Таблица 1. Сравнительная таблица: AssemblyAI библиотека и Whisper API**

|  |  |  |
| --- | --- | --- |
| Параметр | **AssemblyAI Библиотека** | **Whisper API** |
| **Качество распознавания речи** | Отличное | Отличное, но слабее в распознавании сленга и жаргона |
| **Удобство работы** | Легко интегрируется | Удобный интерфейс и доступ через RapidAPI |
| **API доступность** | Ошибка 500 | Не было проблем с доступом |
| **Количество запросов** | Много бесплатных запросов, есть платные планы | Малое количество бесплатных запросов через RapidAPI |
| **Интерфейс отображения запросов** | Приятный интерфейс с отображением оставшихся запросов | Нет отображения оставшихся запросов в API интерфейсе |
| **Гибкость настройки** | Позволяет тонко настраивать параметры | Менее гибкая настройка, но всё же достаточна для большинства задач |
| **Документация** | Подробная, легко читаемая документация | Хорошая документация через RapidAPI |

1. ВЫВОДЫ

В ходе выполнения лабораторной работы был подробно изучен процесс взаимодействия с различными ИИ-сервисами для обработки аудиофайлов, в частности, с использованием Whisper и AssemblyAI. Оба сервиса продемонстрировали высокую точность в распознавании речи, несмотря на наличие посторонних шумов, что подчеркивает их способность эффективно работать в условиях неидеальных аудио-сигналов.

Однако, при использовании AssemblyAI API возникла ошибка 500, что привело к переходу на использование локальной библиотеки assemblyai. Несмотря на технические проблемы с доступом к API, использование библиотеки AssemblyAI в значительной степени компенсировало этот момент, так как она продемонстрировала превосходные результаты в распознавании не только стандартной речи, но и сложных, менее формальных выражений, таких как сленг и англицизмы.

При сравнении двух систем можно сделать вывод, что Whisper и AssemblyAI обе превосходно справляются с расшифровкой речи даже при наличии постороннего шума или дефектов записи. Однако обе модели показывают определенные трудности при обработке специфичных выражений, таких как англицизмы и специализированные термины, что может быть ограничением в приложениях, требующих точности в этих областях.

Вывод: Обе модели, несмотря на их высокую общую эффективность, имеют слабые места при распознавании специфичных выражений и терминов, в частности англицизмов и жаргонизмов. Тем не менее, AssemblyAI показал себя более устойчивым в распознавании таких сложных элементов речи, что может быть полезным для приложений, требующих точности в этих аспектах.

ИСТОЧНИКИ

1. OpenAI Whisper Speech-to-Text API [Электронный ресурс]. URL: https://rapidapi.com/data-elysium-software-incorporation-data-elysium-software-incorporation-default/api/openai-whisper-speech-to-text-api (дата обращения: 15.10.2025).
2. AssemblyAI Documentation [Электронный ресурс]. URL: https://www.assemblyai.com/docs (дата обращения: 15.10.2025).
3. Requests: HTTP for Humans [Электронный ресурс]. URL: https://requests.readthedocs.io/en/latest/ (дата обращения: 15.10.2025
4. PyQt5 Documentation [Электронный ресурс]. URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 15.10.2025).
5. AI Methods GitHub Repository [Электронный ресурс]. URL: https://github.com/fesevu/AI-methods (дата обращения: 23.10.2025).

ПРИЛОЖЕНИЕ А. Скриншоты с результатами Whisper

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 4. Результат Whisper для файла с мужской быстрой речью.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 5. Результат Whisper для файла с мужской медленной речью.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 6. Результат Whisper для файла с женской стандартной речью.

**Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание**

Рис. 7. Результат Whisper для файла с женской быстрой речью.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рис. 8. Результат Whisper для файла с синхронной мужской и женской речью

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 9. Результат Whisper для файла с мужской речью с посторонними шумами.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рис. 10. Результат Whisper для файла с отрывком из песни Сергея Лазарева “Шепотом”.

**Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание**

Рис. 11. Результат Whisper для файла с отрывком из песни Скриптонита “Привычка”.

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рис. 12. Результат Whisper для файла с отрывком из песни Fike & Jambazi “Minimun”.

**Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание**

Рис. 13. Результат Whisper для файла с речью 5-летнего ребёнка с заиканием.

ПРИЛОЖЕНИЕ Б. Скриншоты с результатами AssemblyAI

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 14. Результат AssemblyAI для файла с мужской быстрой речью.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 15. Результат AssemblyAI для файла с мужской медленной речью.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рис. 16. Результат AssemblyAI для файла с женской стандартной речью.

**Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание**

Рис. 17. Результат AssemblyAI для файла с женской быстрой речью.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рис. 18. Результат AssemblyAI для файла с синхронной мужской и женской речью

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис. 19. Результат AssemblyAI для файла с мужской речью с посторонними шумами.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рис. 20. Результат AssemblyAI для файла с отрывком из песни Сергея Лазарева “Шепотом”.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рис. 21. Результат AssemblyAI для файла с отрывком из песни Скриптонита “Привычка”.

**Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание**

Рис. 22. Результат AssemblyAI для файла с отрывком из песни Fike & Jambazi “Minimun”.

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание**

Рис. 23. Результат AssemblyAI для файла с речью 5-летнего ребёнка с заиканием.

ПРИЛОЖЕНИЕ В. Исходный код программы

Содержимое файла main.py

**import** **sys**

**from** **PyQt5.QtWidgets** **import** QApplication

**from** **interface** **import** SpeechToTextApp

# Запуск приложения

**def** **main**():

**print**("Запуск приложения...")

app = QApplication(sys.argv)

window = SpeechToTextApp()

window.show()

sys.exit(app.exec\_())

**if** \_\_name\_\_ == "\_\_main\_\_":

main()

Содержимое файла interface.py

**from** **PyQt5.QtWidgets** **import** QMainWindow, QVBoxLayout, QLabel, QComboBox, QPushButton, QTextEdit, QFileDialog

**from** **PyQt5.QtCore** **import** QThread, pyqtSignal

**from** **integration** **import** transcribe\_with\_whisper, transcribe\_with\_assembly

**class** **TranscribeThread**(QThread):

result\_ready = pyqtSignal(str)

**def** **\_\_init\_\_**(self, file\_name, api):

super().\_\_init\_\_()

self.file\_name = file\_name

self.api = api

**def** **run**(self):

**if** self.api == "Whisper":

result = transcribe\_with\_whisper(self.file\_name)

**elif** self.api == "AssemblyAI":

result = transcribe\_with\_assembly(self.file\_name)

**else**:

result = "Неверный выбор API."

self.result\_ready.emit(result)

**class** **SpeechToTextApp**(QMainWindow):

**def** **\_\_init\_\_**(self) -> None:

super().\_\_init\_\_()

**print**("Инициализация SpeechToTextApp...")

self.setWindowTitle("Speech to Text App")

self.setGeometry(**200**, **200**, **500**, **400**)

# Компоненты интерфейса

layout = QVBoxLayout()

self.api\_selector = QComboBox()

self.api\_selector.addItems(["Whisper", "AssemblyAI"])

layout.addWidget(self.api\_selector)

self.file\_label = QLabel("Файл не выбран")

layout.addWidget(self.file\_label)

self.select\_file\_btn = QPushButton("Выбрать аудиофайл")

self.select\_file\_btn.clicked.connect(self.select\_file)

layout.addWidget(self.select\_file\_btn)

self.transcribe\_btn = QPushButton("Начать расшифровку")

self.transcribe\_btn.clicked.connect(self.transcribe\_audio)

layout.addWidget(self.transcribe\_btn)

self.result\_area = QTextEdit()

layout.addWidget(self.result\_area)

self.central\_widget = QLabel()

self.central\_widget.setLayout(layout)

self.setCentralWidget(self.central\_widget)

self.selected\_file: str = ""

**def** **select\_file**(self) -> None:

"""

Функция для открытия диалогового окна и выбора аудиофайла.

Устанавливает выбранный путь к файлу в self.selected\_file.

"""

**print**("Открытие диалогового окна для выбора аудиофайла...")

options = QFileDialog.Options()

file\_name, \_ = QFileDialog.getOpenFileName(

self, "Выбрать файл", "", "Аудиофайлы (\*.mp3 \*.wav)", options=options)

**if** file\_name:

self.selected\_file = file\_name

self.file\_label.setText(f"Выбран файл: {file\_name}")

**print**(f"Выбран файл: {file\_name}")

**else**:

**print**("Файл не выбран.")

**def** **transcribe\_audio**(self) -> None:

"""

Функция для расшифровки выбранного аудиофайла с использованием выбранного API (Whisper или AssemblyAI).

Отображает расшифрованный текст в текстовом поле результата.

"""

selected\_api = self.api\_selector.currentText()

**print**(f"Выбранный API для расшифровки: {selected\_api}")

**if** self.selected\_file:

**print**(f"Начало расшифровки для файла: {self.selected\_file}")

self.thread = TranscribeThread(self.selected\_file, selected\_api)

self.thread.result\_ready.connect(self.display\_result)

self.thread.start()

**else**:

self.result\_area.setText("Пожалуйста, выберите файл.")

**print**("Файл для расшифровки не выбран.")

**def** **display\_result**(self, result: str) -> None:

"""

Функция для отображения результата расшифровки.

"""

self.result\_area.setText(result.strip())

**print**(f"Результат расшифровки: {result}")

Содержимое файла integration.py

**import** **os**

**import** **requests**

**import** **assemblyai** **as** **aai**

**from** **dotenv** **import** load\_dotenv

# Загрузка переменных окружения

load\_dotenv()

# Установка API ключей из переменных окружения

aai.settings.api\_key = os.getenv("ASSEMBLYAI\_API\_KEY")

RAPIDAPI\_KEY = os.getenv("RAPIDAPI\_KEY")

# Расшифровка аудио с использованием Whisper API

**def** **transcribe\_with\_whisper**(file\_name: str) -> str:

"""

Функция для расшифровки аудио с использованием Whisper API.

Аргументы:

file\_name (str): Путь к аудиофайлу для расшифровки.

Возвращает:

str: Расшифрованный текст из аудиофайла или сообщение об ошибке.

"""

**print**(f"Начало расшифровки Whisper для файла: {file\_name}")

url = "https://openai-whisper-speech-to-text-api.p.rapidapi.com/transcribe"

**with** open(file\_name, 'rb') **as** audio\_file:

files = {

'file': ('audio.mp3', audio\_file),

'type': (None, 'RAPID'),

'response\_format': (None, 'JSON')

}

headers = {

"x-rapidapi-key": RAPIDAPI\_KEY,

"x-rapidapi-host": "openai-whisper-speech-to-text-api.p.rapidapi.com"

}

**try**:

response = requests.post(

url, files=files, headers=headers, timeout=**60**)

**print**(

f"Получен ответ от Whisper API с кодом статуса: "

f"{response.status\_code}"

)

**if** response.status\_code == **200**:

result = response.json()

**print**(f"Результат расшифровки: {result}")

**return** result['response'].get('text', 'Текст не найден в ответе.')

**else**:

**print**(f"Ошибка при расшифровке Whisper: {response.text}")

**return** f"Ошибка: {response.status\_code}, {response.text}"

**except** requests.exceptions.RequestException **as** e:

**print**(f"Ошибка при выполнении запроса Whisper: {e}")

**return** f"Ошибка при выполнении запроса: {e}"

# Расшифровка аудио с использованием AssemblyAI API

**def** **transcribe\_with\_assembly**(file\_name: str) -> str:

"""

Функция для расшифровки аудио с использованием AssemblyAI API.

Аргументы:

file\_name (str): Путь к аудиофайлу для расшифровки.

Возвращает:

str: Расшифрованный текст из аудиофайла или сообщение об ошибке.

"""

**print**(f"Начало расшифровки AssemblyAI для файла: {file\_name}")

transcriber = aai.Transcriber()

config = aai.TranscriptionConfig(language\_code="ru")

transcript = transcriber.transcribe(file\_name, config)

**print**(f"Статус расшифровки AssemblyAI: {transcript.status}")

**if** transcript.status == aai.TranscriptStatus.error:

**print**(f"Ошибка при расшифровке AssemblyAI: {transcript.error}")

**return** transcript.error

**else**:

**print**(f"Результат расшифровки: {transcript.text}")

**return** transcript.text