МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка интеллектуального сервиса для генерации аннотаций к аудиофайлам

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-538.ВКР

Научные руководители:
ст. преподаватель кафедры СП
_____ Н.С. Силкина,

м.н.с. кафедры СП
_____ А.Е. Старков

Автор работы,
студент группы КЭ-402
_____ М.С. Беляков

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Белякову Максиму Сергеевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка интеллектуального сервиса для генерации аннотаций к аудиофайлам.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Ott M., Edunov S., Baevski A., Fan A., Gross S., Ng N., Grangier D., Auli M. Fairseq: A fast, extensible toolkit for sequence modeling. [Электронный ресурс] // arXiv.org. 2019. Дата обновления: 01.04.2019 г. URL: https://arxiv.org/pdf/1904.01038 (дата обращения: 09.03.2024 г.).
3.2. PyTorch [Электронный ресурс] URL: https://pytorch.org/ (дата обращения: 10.02.2024 г.).
3.3. Radford A., Kim J., Xu T., Brockman G., McLeavey C., Sutskever I. Robust Speech Recognition via Large-Scale Weak Supervision. [Электронный ресурс] // arXiv.org. 2023. Дата обновления: 06.12.2023 г. URL: https://arxiv.org/abs/2212.04356 (дата обращения: 21.04.2024 г.).
3.4. Radfar M., Barnwal R., Swaminathan R., Chang F., Strimel G., Susanj N., Mouchtaris A. ConvRNN-T: Convolutional Augmented Recurrent Neural Network Transducers for Streaming Speech Recognition. [Электронный ресурс] // arXiv.org. 2022. Дата обновления: 29.09.2022 г. URL: https://arxiv.org/abs/2209.14868 (дата обращения: 16.04.2024 г.).

4. Перечень подлежащих разработке вопросов

4.1. Осуществить поиск и проанализировать аналоги.

4.2. Выбрать архитектуру модели.

4.3. Реализовать модель.

4.4. Провести оптимизацию модели.

4.5. Разместить модель на веб-сервисе.

5. Дата выдачи задания: 29.01.2024 г.

Научные руководители:

ст. преподаватель кафедры СП Н.С. Силкина

м.н.с. кафедры СП А.Е. Старков

Задание принял к исполнению М.С. Беляков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В последние годы наблюдается устойчивая тенденция к широкомасштабному внедрению и использованию систем распознавания речи, что обусловлено повышением их доступности и эффективности. Востребованность данных технологий напрямую связана с экспоненциальным ростом объема аудиоданных. Учитывая, что речевая информация представляет собой ценный источник данных, необходимых для решения различных задач, таких как информационный поиск, анализ настроения, тематическое моделирование и многие другие, существует высокая потребность в автоматизированных методах обработки аудиоданных.

Цифровая обработка аудиофайлов и генерация аннотаций к ним становятся неотъемлемой частью современного мира. Технологии распознавания речи находят применение в различных сферах жизни. Голосовые команды управляют смартфонами, умными домами и автомобилями. Виртуальные ассистенты, основанные на распознавании речи, такие как Siri, Alexa или Google Assistant, выполняют разнообразные задачи, понимая естественную речь. Автоматическая транскрипция аудиозаписей позволяет быстро получать текстовые расшифровки лекций, интервью или деловых встреч, упрощая обработку и анализ аудиоданных. Системы голосового управления обеспечивают более естественное и интуитивное взаимодействие с устройствами и сервисами, повышая удобство и эффективность использования. Благодаря внедрению этих технологий, становится возможным более естественное и эффективное взаимодействие с цифровой средой посредством голосовых команд и запросов.

С каждым годом наблюдается значительный рост производительности домашних персональных компьютеров. Увеличение вычислительной мощности и доступности высокопроизводительных процессоров и графических карт открывает новые возможности для использования систем автоматического распознавания речи (ASR) в домашних условиях. Благодаря

этому прогрессу, все больше людей получают доступ к технологиям ASR, которые ранее были доступны только на специализированных серверах или суперкомпьютерах.

Постановка задачи

Целью выпускной квалификационной работы является разработка интеллектуального сервиса для генерации аннотаций к аудиофайлам.

Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области;
2) выбрать технологии реализации;
3) обучить акустическую модель;
4) создать языковую модель;
5) создать веб-сервис для демонстрации ASR системы.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 32 страницы, объем списка литературы – 23 источника.

В первой главе производится анализ предметной области.

Вторая глава посвящена проектированию моделей и веб-сервиса.

Третья глава посвящена реализации моделей и веб-сервиса.

Четвертая глава посвящена тестированию моделей и веб-сервиса.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ существующих решений для реализации проекта

В ходе анализа выявилась особенность большинства решений для ASR систем. У них не предусмотрен интерфейс для автоматического перевода данных из аудио в текстовый формат. При работе с аудио данными принято выделять, а затем приводить акустические признаки в графический вид. Примером такого формата зачастую становится спектрограмма.

Сам процесс ASR для большей эффективности использования ресурсов системы и улучшению метрик оценки состоит из нескольких частей.

1. Акустическая модель. Это компонент, отвечающий за распознавание фонем или звуков, составляющих произнесенные слова. Обычно акустическая модель основана на скрытых марковских моделях (HMM) или нейронных сетях, таких как глубокие нейронные сети (DNN) или сверточные нейронные сети (CNN).

2. Языковая модель. Этот компонент помогает определить последовательность слов на основе контекста и вероятностей словосочетаний в языке. Языковая модель может быть статистической, основанной на n-граммах, или использовать методы машинного обучения, такие как рекуррентные нейронные сети (RNN) или трансформеры.

3. Декодер. Этот компонент объединяет акустическую и языковую модели для генерации наиболее вероятной последовательности слов на основе входного аудио. Декодер может использовать алгоритмы поиска, такие как beam search или алгоритмы декодирования на основе вероятностной модели.

Сам процесс виден на рисунке 1. Проходит он по следующей последовательности:

1) пользователь передает аудиофайл акустической модели;

2) акустическая модель передает свою последовательность вероятностей в декодер;

3) декодер удаляет повторяющиеся символы и символы заполнители и передает эту последовательность языковой модели;

4) языковая модель свою последовательность вероятностей обратно декодеру;

5) декодер находит последовательности слов с наибольшей совместной вероятностью;

6) пользователю выдается финальный результат.

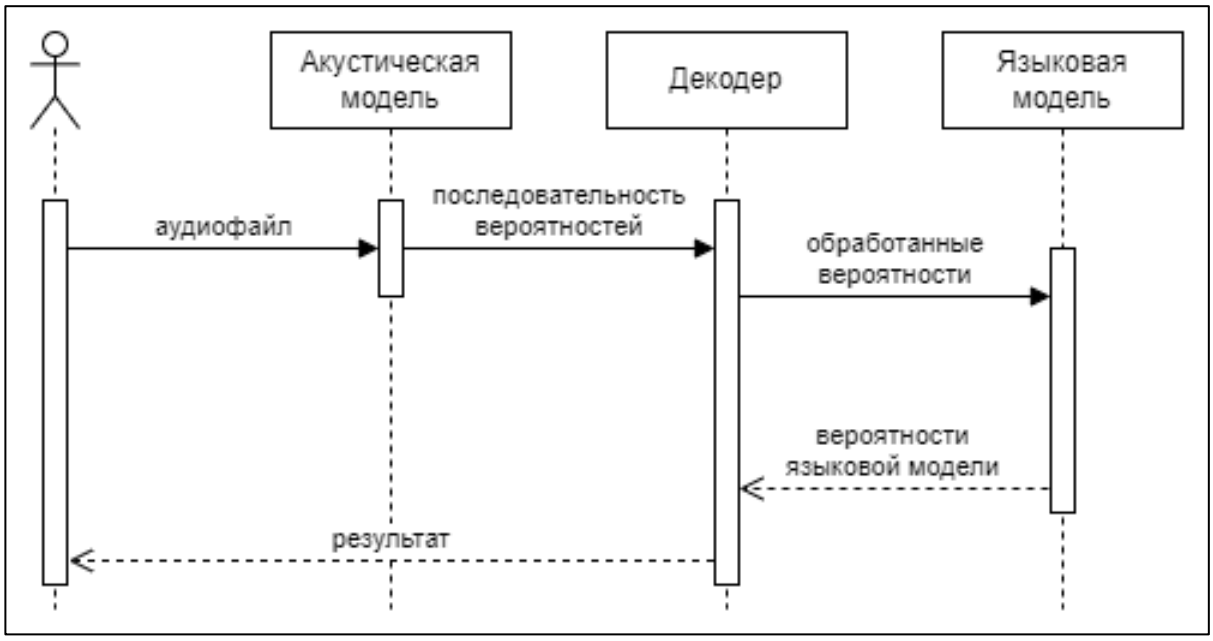

Рисунок 1 – Диаграмма последовательности

1.2. Сравнение аналогов

Deepgram [1]

Компания Deepgram, известная своими передовыми разработками в сфере обработки аудиоинформации, представила революционное приложение, способное произвести настоящий прорыв в этой области. Данная интеллектуальная платформа, основанная на сложных алгоритмах глубокого обучения, демонстрирует впечатляющую точность и эффективность при распознавании человеческой речи.

С помощью Deepgram пользователи могут транскрибировать разнообразные аудиозаписи с высокой скоростью и детализацией. Приложение

справляется с обработкой огромных массивов аудиоданных, выделяя ключевые моменты и предоставляя пользователям ценные инсайты.

Функционал Deepgram не ограничивается лишь транскрибацией. Платформа предлагает широкий спектр возможностей, включая анализ контент-центров, классификацию аудиофайлов по различным параметрам, определение контекста и эмоциональной окраски речи. Такой набор инструментов делает Deepgram полезным помощником в самых разных сферах: от научных исследований и бизнес-аналитики до медицины и образования

С помощью Deepgram пользователи получают возможность глубоко и масштабно исследовать и анализировать аудиоданные, предоставляя специалистам мощный и удобный инструментарий для решения амбициозных задач.

Google Cloud Speech-to-Text [2]

Google Cloud представляет решение Speech-to-Text, позволяющее конвертировать аудиозаписи в текстовый формат с высокой точностью и скоростью. В основе этого сервиса лежат самые современные технологии машинного обучения и нейронных сетей, разработанные ведущими специалистами Google. Они обеспечивают высочайшую эффективность и надежность распознавания речи.

Возможности Google Cloud Speech-to-Text дают широкий спектр возможностей, включая распознавание речи в реальном времени, поддержку различных языков и диалектов, адаптацию к специфическим акцентам и условиям записи. Благодаря своей масштабируемости и надежности, этот сервис является идеальным инструментом для различных приложений, включая автоматическую транскрипцию аудио- и видеофайлов, создание субтитров для видеоконтента, анализ аудиоданных в реальном времени и многое другое.

Большое преимущество Google Cloud Speech-to-Text заключается в его интеграции с другими сервисами и инструментами Google Cloud, что

позволяет пользователям создавать сложные и мощные приложения для обработки и анализа аудиоданных. В целом, Google Cloud Speech-to-Text открывает новые возможности для различных отраслей, предоставляя пользователям инструменты для эффективного управления и анализа речевых данных.

Whisper AI [3]

Whisper AI – это нейронная сеть, разработанная и открытая для использования, которая достигает уровня устойчивости и точности распознавания речи на уровне человеческого. Это автоматическая система распознавания речи, обученная на 680 000 часах мультитасковых данных различных языков, собранных из интернета. Использование такого обширного и разнообразного набора данных позволяет улучшить устойчивость к акцентам, фоновому шуму и технической лексике, а также осуществлять транскрипцию на нескольких языках и перевод на английский.

Архитектура Whisper представляет собой простой энд-то-энд подход, реализованный в виде кодера-декодера Transformer. Входной аудиофайл разбивается на участки по 30 секунд, конвертируется в логарифмическую мел-спектрограмму и подается на вход кодеру. Декодер обучен предсказывать соответствующий текст, а также выполнять дополнительные задачи, такие как идентификация языка, временные метки на уровне фраз, транскрипция мульти тональной речи и перевод речи на английский.

Одна из ключевых особенностей Whisper заключается в использовании обширного и разнообразного набора данных, включающего не только английский, но и другие языки. Этот подход позволяет системе эффективно учиться переводу речи на текст и превосходит текущие модели на нулевом этапе перевода CoVoST2 на английский язык.

Вывод по первой главе

В рамках данной главы была рассмотрена последовательность выполнения процесса ASR по современным стандартам, а также характеристики аналогичных ASR систем.

2. ПРОЕКТИРОВАНИЕ МОДЕЛИ И СЕРВИСА

2.1. Требования к системе

Функциональные требования

Функциональные требования описывают только поведение системы в отрыве от ее реализации. К данной работе были выявлены следующие функциональные требования:

1) ASR модель должна уметь преобразовывать аудиофайл в текстовый формат;

2) пользователь должен иметь возможность загружать свой аудиофайл для модели;

3) у пользователя должна быть возможность взаимодействия с моделью через веб UI;

4) система должна уметь конвертировать аудио в нужный формат.

Нефункциональные требования

Нефункциональные требования, в отличие от функциональных требований, описывают свойства и ограничения системы. Были выявлены следующие нефункциональные требования:

1) модель должна быть создана, используя такие библиотеки и технологии, как: Python [4], PyTorch [5], NeMo [6], KenLM [7];

2) серверная сторона сервиса должна быть написана на Flask [8];

3) клиентская сторона сервиса должна использовать React [9];

4) скорость работы клиентской стороны должна соответствовать современным стандартам разработки.

На данный момент язык программирования Python является стандартом в сфере машинного обучения. Библиотека PyTorch написана на Python и является одной из самых больших библиотек для машинного обучения. Например, fairseq [10] – инструментарий, разработанный Facebook над PyTorch. Помимо большого количества цитирований он также пользуется большим спросом среди разработчиков.

2.2. Архитектура нейронной сети.

Акустическая модель

Акустическая модель является ключевым компонентом системы распознавания речи. Она отвечает за преобразование входных аудиоданных в последовательность фонем или других лингвистических единиц. Для реализации акустической модели в данном проекте рассматривались различные архитектуры нейронных сетей, учитывая доступные системные ресурсы: видеокарту NVIDIA RTX 3060, 16 ГБ оперативной памяти и процессор Intel Core i5-12400F.

Основные архитектуры для этой модели это CNN, RNN и LTSM. Модели на базе CNN могут учитывать пространственные данные, следственно работают с представлением аудио в спектрограмме. Модели RNN учатся на последовательностях данных, а LTSM помимо этого могут учитывать признаки, прослеживающийся на протяжении всего датасета и успешно используются в ASR моделях [11].

CNN архитектура учитывает пространственные признаки и поэтому для ее обучения нужно преобразовывать текст в спектрограммы.

RNN архитектура позволяет обрабатывать последовательности переменной длины и учитывать контекст предыдущих слов. Однако, RNN страдают от проблемы исчезающего градиента, что затрудняет обучение на длинных последовательностях [12].

LSTM архитектура является усовершенствованной версией RNN, которая решает проблему исчезающего градиента. LSTM ячейки имеют специальные гейты (вентили), которые контролируют поток информации и позволяют сохранять долгосрочные зависимости.

Были проанализированы следующие архитектуры.

1. DeepSpeech. Это архитектура, предложенная компанией Mozilla, использует глубокие рекуррентные нейронные сети с долгой краткосрочной памятью (LSTM). DeepSpeech показывает высокую точность распознавания, но требует значительных вычислительных ресурсов и времени обучения.

2. Wave2Vec. Архитектура, основанная на самообучении, использует сверточные нейронные сети (CNN) для извлечения признаков из аудиоданных. Может обучаться на большом количестве неразмеченных данных, что позволяет улучшить качество распознавания. Имеет недостаток в виде необходимости к большому объему памяти и вычислительных ресурсов для обучения.

3. ContextNet. Легковесная архитектура, сочетающая сверточные и рекуррентные слои. ContextNet обеспечивает баланс между точностью и скоростью распознавания, что делает ее подходящей для систем реального времени. Тем не менее, для достижения высокой точности требуется тщательная настройка гиперпараметров.

4. QuartzNet15x5. Архитектура, основанная на глубоких сверточных нейронных сетях с остаточными связями. QuartzNet15x5 демонстрирует высокую точность распознавания при относительно небольшом количестве параметров. Данная архитектура эффективно использует вычислительные ресурсы и может обучаться за приемлемое время на доступном оборудовании.

После анализа преимуществ и недостатков каждой архитектуры, с учетом имеющихся системных ресурсов и временных ограничений, была выбрана архитектура QuartzNet15x5 для реализации акустической модели. Данный выбор обусловлен несколькими факторами:

1) высокая базовая точность распознавания без тщательного подбора гиперпараметров;

2) эффективное использование вычислительных ресурсов;

3) возможность дообучения модели на специфичных данных для улучшения качества распознавания в конкретной предметной области.

Из данных факторов следует, что на данной архитектуре модель может иметь приемлемую эффективность распознавания при условии сжатых сроков обучения модели, а также возможности впоследствии улучшить показатели модели для определенных условий.

Языковая модель

Языковая модель является вторым ключевым компонентом в системе генерации аннотаций к аудиофайлам. Она необходима для понимания и генерации естественного языка на основе входных данных, полученных от акустической модели. Языковая модель помогает определить наиболее вероятные последовательности слов и фраз, учитывая контекст и грамматические правила языка.

Одним из распространенных подходов к построению языковых моделей являются n-gram модели. Они основаны на статистическом анализе последовательностей слов длины n в обучающих данных. Они используют вероятностное распределение для предсказания следующего слова на основе предыдущих (n-1) слов. Однако, n-gram модели имеют ограничения, такие как невозможность учета долгосрочных зависимостей и проблема разреженности данных.

Альтернативой n-gram моделям являются нейронные языковые модели, основанные на архитектурах рекуррентных нейронных сетей (RNN), долгой краткосрочной памяти (LSTM) и трансформеров (Transformers).

Трансформеры – это современная архитектура, основанная на механизме самовнимания (self-attention). Трансформеры позволяют моделировать зависимости между словами без использования рекуррентных связей, что делает их эффективными для параллельных вычислений и обработки длинных последовательностей [13].

Для декодирования последовательности слов из языковой модели часто используется алгоритм beam search. Beam search рассматривает несколько наиболее вероятных гипотез на каждом шаге декодирования и выбирает наилучшую последовательность на основе совокупной вероятности. Языковая модель комбинируется с акустической моделью для определения наиболее вероятной последовательности слов, соответствующей входному аудиосигналу.

Таким образом, на основе приведенных характеристик целевой системы и ограничений по времени обучения, использование n-gram модели может быть оптимальным выбором для построения языковой модели в рамках системы генерации аннотаций к аудиофайлам. N-gram модели основаны на статистическом анализе последовательностей слов фиксированной длины n в обучающих данных и используют вероятностное распределение для предсказания следующего слова на основе предыдущих (n-1) слов. Несмотря на наличие ограничений, таких как невозможность учета долгосрочных зависимостей и проблема разреженности данных, n-gram модели относительно просты в обучении и не требуют большого количества вычислительных ресурсов. В перспективе, при необходимости использования более сложных и точных языковых моделей, можно рассмотреть применение нейронных архитектур, таких как RNN, LSTM или трансформеры.

Помимо этого, будет использоваться алгоритм beam search для выбора лучших слов из предположений акустической и языковой модели. За основу была взята библиотека CTC-decoders [14].

2.3. Сервисная часть

Для разработки сервиса будет использоваться архитектура клиент-сервер. Серверная часть (backend) будет реализована на фреймворке Flask, который является легковесным и гибким решением для создания веб-приложений на языке Python.

Backend будет принимать POST HTTP запросы с аудиофайлами от клиентской части (frontend). При получении запроса сервер будет выполнять следующие шаги:

1) полученный аудиофайл будет конвертирован в формат MP3 с помощью инструмента ffmpeg [15] для возможности модели работать с аудиофайлом;

2) после конвертации будет запущена ASR модель, которая будет извлекать текст из аудио;

3) извлеченный текст будет возвращен клиентской части в формате JSON в качестве ответа на исходный POST запрос.

Клиентская часть будет разработана с использованием современного инструмента Vite [16], который обеспечивает быструю сборку и оптимизацию веб-приложений. В качестве основной библиотеки для построения пользовательского интерфейса будет использоваться React, который предоставляет компонентный подход к разработке и позволяет создавать интерактивные и динамические веб-страницы.

Для повышения надежности и облегчения разработки, frontend будет написан на языке TypeScript [17].

Пользовательский интерфейс будет предоставлять возможность выбора аудиофайла с помощью элемента ввода файла. После выбора файла пользователь сможет отправить его на сервер, нажав соответствующую кнопку.

После получения ответа от сервера, frontend будет отображать полученный текст на странице. Текст будет представлен в удобочитаемом формате, возможно с использованием элементов форматирования, таких как абзацы или списки.

2.4. Оптимизация обучения модели

Обучение языковых моделей с нуля на больших объемах данных может занимать значительное время и требовать существенных вычислительных ресурсов. В зависимости от размера модели и объема обучающих данных, процесс обучения может занимать от нескольких дней до нескольких недель, даже на мощных GPU.

Если рассматривать обучение модели на доступном оборудовании и зависимость от выбранной архитектуры и размера обучающего набора данных. Для трансформерной модели среднего размера (например, с 12 слоями и 768 скрытыми юнитами), обучение на наборе данных из нескольких миллионов предложений может занять около 1–2 недели. Для LSTM модели

аналогичного размера, время обучения может быть немного меньше, около 1 недели.

Однако, для ускорения процесса обучения и снижения требований к вычислительным ресурсам, можно применить подход transfer learning (перенос обучения). Transfer learning – это техника, при которой модель, предварительно обученная на большом наборе данных для решения похожей задачи, используется в качестве отправной точки для обучения новой модели на целевом наборе данных.

Преимущество transfer learning заключается в том, что предварительно обученная модель уже содержит богатые языковые представления и знания, полученные из большого объема данных. Вместо обучения модели с нуля, могут использоваться веса предварительно обученной модели с последующей настройкой их на конкретную задачу с помощью дообучения (fine-tuning) на целевом наборе данных.

Использование transfer learning может значительно сократить время обучения и требования к вычислительным ресурсам. Вместо недель обучения, процесс дообучения модели на целевом наборе данных может занять от нескольких часов до нескольких дней, в зависимости от размера данных и выбранных гиперпараметров. Это позволяет сэкономить значительное количество времени и ресурсов по сравнению с обучением модели с нуля.

2.5. Как оценивать модель

Для разработки интеллектуального сервиса генерации аннотаций к аудиофайлам важнейшим компонентом является система автоматического распознавания речи (ASR). Качество работы ASR модели напрямую влияет на точность и полезность создаваемых аннотаций. Поэтому выбор подходящей метрики оценки является ключевым аспектом при разработке такого сервиса.

Существует несколько распространенных метрик для оценки эффективности ASR систем, такие как:

1) Word Error Rate (WER) – показатель ошибки на уровне слов;

2) Character Error Rate (CER) – показатель ошибки на уровне символов;

3) Sentence Error Rate (SER) – показатель ошибки на уровне предложений.

Наиболее часто используемой и информативной метрикой является Word Error Rate (WER). Она вычисляется как отношение суммы ошибочно распознанных, пропущенных и вставленных слов к общему количеству слов в эталонной транскрипции, представленной в формуле (1):

$$WER = (S + D + I)/N, \quad (1)$$

где S – количество неправильно распознанных слов;

D– количество пропущенных слов;

I – количество лишних вставленных слов;

N – общее число слов в эталонной транскрипции.

Чем ниже значение WER, тем лучше качество распознавания речи. Идеальная ASR модель будет иметь WER, равный нулю.

WER является наиболее подходящей метрикой для оценки ASR в контексте генерации аннотаций по причинам того, что она:

1) учитывает все три типа ошибок (замены, пропуски, вставки), что дает полную картину качества распознавания;

2) работает на уровне слов, что важно для создания осмысленных аннотаций и извлечения ключевой информации из аудио;

3) интуитивно понятна и широко используется в индустрии, что облегчает сравнение с другими ASR системами.

Таким образом, для разрабатываемого сервиса целесообразно использовать метрику Word Error Rate (WER) для оценки и оптимизации качества

работы ASR модели. Это позволит обеспечить высокую точность распознавания речи и, как следствие, генерацию информативных и полезных аннотаций к аудиофайлам.

2.6. Датасеты

Этап поиска и подготовки данных является одним из наиболее важных этапов в процессе разработки модели машинного обучения ведь невероятно сложно или невозможно натренировать приемлемую модель на плохих или недостаточных данных.

Качество данных напрямую влияет на производительность модели. Если данные содержат шум, ошибки или несоответствия, модель может выдавать неправильные прогнозы или иметь низкую точность.

Common Voice [18]

Common Voice – это открытый краудсорсинговый проект, инициированный Mozilla для сбора и публикации большого количества голосовых данных на различных языках. Цель проекта – сделать технологии распознавания речи более доступными и инклюзивными.

Особенности датасета Common Voice:

1) содержит голосовые записи и соответствующие транскрипции на множестве языков;

2) данные собираются от добровольцев по всему миру, обеспечивая разнообразие голосов, акцентов и демографических характеристик;

3) лицензируется под лицензией Creative Commons Zero (CC0), что позволяет свободно использовать данные для любых целей;

4) постоянно растет и обновляется благодаря активному участию сообщества.

Датасет Common Voice является ценным ресурсом для обучения ASR моделей, особенно для языков с ограниченными голосовыми данными. Разнообразие и качество данных помогают создавать более робастные и универсальные модели распознавания речи.

Golos [19]

Golos – это открытый датасет русской речи, созданный компанией Base Analytics. Он предназначен для обучения и тестирования систем автоматического распознавания речи на русском языке.

Особенности датасета Golos:

1) содержит более 1000 часов транскрибированной русской речи;

2) записи сделаны в различных акустических условиях (студия, улица, дом и т.д.);

3) включает в себя речь носителей языка разного пола, возраста и диалектов;

4) лицензируется под лицензией Creative Commons Attribution 4.0 (CC BY 4.0), позволяющей свободное использование при указании авторства;

5) организован в виде структурированных файлов аудио и соответствующих текстовых транскрипций.

Датасет Golos является одним из крупнейших открытых ресурсов русской речи и может использоваться для обучения высококачественных ASR моделей, специализированных для русского языка.

Использование таких датасетов, как Common Voice и Golos, позволяет разработчикам создавать более эффективные и адаптированные ASR модели для широкого круга приложений, в том числе для генерации аннотаций к аудиофайлам. Качество и разнообразие данных в этих датасетах способствуют улучшению точности распознавания речи и робастности моделей в различных условиях.

Вывод по второй главе

В рамках данной главы были определены требования к системе, выбрана архитектура для акустической модели, выбраны технологии для реализации веб-сервиса, определен метод для ускорения обучения модели, выбрана метрика оценки качества модели и были выбраны датасеты.

3. РЕАЛИЗАЦИЯ

На протяжении обучении модели и создании веб-сервиса будет использоваться система Git [20] для контроля версий проекта. Помимо него будет использоваться WSL [21] для возможности работы со всеми библиотеками и технологиями, приведенными в работе.

3.1. Работа с датасетами

Golos

Датасет содержит более 1000 часов данных, размеченных вручную. Суммарный размер датасета составляет 144 Гб, и он разделен на разные архивы (таблица 1).

Таблица 1 – Архивы датасета Golos

Наименование	Архив	Размер
Датасет дальнего звука	train_farfield.tar	15,4 ГБ
Датасет толпы ч.1	train_crowd0.tar	11 ГБ
Датасет толпы ч.2	train_crowd1.tar	14 ГБ
Датасет толпы ч.3	train_crowd2.tar	13,2 ГБ
Датасет толпы ч.4	train_crowd3.tar	11,6 ГБ
Датасет толпы ч.5	train_crowd4.tar	15,8 ГБ
Датасет толпы ч.6	train_crowd5.tar	13,1 ГБ
Датасет толпы ч.7	train_crowd6.tar	15,7 ГБ
Датасет толпы ч.8	train_crowd7.tar	12,7 ГБ
Датасет толпы ч.9	train_crowd8.tar	12,2 ГБ
Датасет толпы ч.10	train_crowd9.tar	8,08 ГБ
Тестовые данные	test.tar	1,3 ГБ

В датасете для каждого аудиофайла есть строка в «manifest.jsonl» файле, она содержит такие поля как:

1) `audio_filepath` – путь к файлу;
2) `id` – идентификатор записи;
3) `duration` – длительность аудио;
4) `text` – расшифровка аудио.

Common Voice

На данный момент датасет содержит 274 часа записанных аудио и 235 проверенных часов. Количество голосов составляет 3206. Эти аудиозаписи

были собраны с помощью волонтеров со всего мира, которые зачитывали предложения на своих родных языках. К примеру, последняя версия датасета на английском языке содержит 3508 записанных и 2615 проверенных часов данных, а также 92325 голоса.

Датасет постоянно пополняется новыми данными, и они проходят проверку другими волонтерами.

Разметка датасета состоит из множества tsv файлов.

Основным файлом является validated.tsv, он содержит 163389 строк данных.

Использование датасетов

Для работы с файлами типа json и jsonl для ASR в NeMo есть функция AudioToCharDataset. Разметка датасета Common Voice состоит из tsv файлов поэтому они были преобразованы в понимаемый NeMo формат в файл «manifest.jsonl».

3.2. Дообучение акустической модели

Не смотря на использование transfer learning, модель все равно может тренироваться относительно долгое время.

В начале тренировки были выбраны 10 часов данных, и на них были перебраны случайным образом следующие гиперпараметры:

1) `learning_rate` – скорость обучения;
2) `batch_size` – размер батча;
3) `dropout` – исключение определенного процента данных.

После этого модель была обучена на 150 часах данных с лучшими выбранными гиперпараметрами. Данные были частично из Golos и Common Voice. Обучение было в 50 эпох и длилось примерно 90 часов. Код тренировки представлен в листинге 1.

Листинг 1 – Дообучение акустической модели

```
import torch
from torch import nn, optim
import pytorch_lightning as pl
```

```
from nemo.collections.asr.data import audio_to_text
from nemo.collections.asr.models import EncDecCTCModel

golos_dataset = audio_to_text.AudioToCharDataset(
    manifest_filepath='./data/train/100hours.jsonl',
    labels=None
)

common_voice_dataset = audio_to_text.AudioToCharDataset(
    manifest_filepath='./data/ru/manifest.jsonl',
    labels=None
)

model = EncDecCTCModel.from_pretrained(model_name="stt_ru_transformer",
strict=False)

trainer = pl.Trainer(gpus=1, max_epochs=100, precision=16, amp_level='O1')
trainer.fit(model, train_dataloader=common_voice_dataset,
val_dataloaders=golos_dataset)

model.save_to("trained_acoustic_model.nemo")
```

В этом листинге нельзя увидеть инициализацию параметров и метрик, так как модель `EncDecCTCModel` устанавливает нужные значения по умолчанию.

3.3. Построение языковой модели

Для разработки языковой модели была задействована библиотека KenLM, включающая в себя инструментарий для построения и использования статистических языковых моделей. Модели, созданные с помощью этой библиотеки легковесные и быстрые [22]. В предварительной стадии необходимо было конвертировать исходные датасеты в формат, совместимый с этой библиотекой, что достигалось путем извлечения текстовых данных из датасетов и их объединения в единый файл.

Для построения языковой модели n-gram использовалась утилита lmplz, которая является частью библиотеки KenLM. Этот процесс требовал указания параметра n (длины n-граммы) и пути к текстовому файлу с данными. Следующим этапом было преобразование построенной модели в бинарный формат, что способствовало сокращению размера файла и повышению скорости его загрузки.

Для использования языковой модели вместе с акустической моделью необходимо было установить CTC-Decoders. Метод CTC предполагает обучение нейронной сети таким образом, чтобы она могла предсказывать вероятности символов в выходной последовательности без явного выравнивания с входными данными.

Для интеграции языковой модели с акустической моделью необходимо было установить CTC-Decoders, которые представляют собой инструментарий для декодирования выходных последовательностей, полученных от нейронной сети с использованием метода Connectionist Temporal Classification (CTC). Этот метод обучения нейронной сети позволяет ей предсказывать вероятности символов в выходной последовательности без явного выравнивания с входными данными.

3.4. Веб-сервис

Для обеспечения удобного доступа пользователей к функционалу сервиса было разработано клиентское веб-приложение. В качестве основы для разработки использовался современный инструментарий, включающий в себя следующие технологии.

1. Vite. Это инструмент сборки, позволяющий собирать приложение без необходимости настраивать конфигурацию.
2. React. Это Javascript библиотека позволяющая писать понятный функциональный код для разработки компонентов.
3. Typescript. Это расширение языка Javascript добавляющее типы.
4. Tailwind CSS и DaisyUI. Это CSS фреймворк и библиотека компонентов. Они нужны для стилизации интерфейса.

Вместе они представляют хороший набор инструментов, который можно использовать для коммерческих продуктов. Результат представлен на рисунке 2.

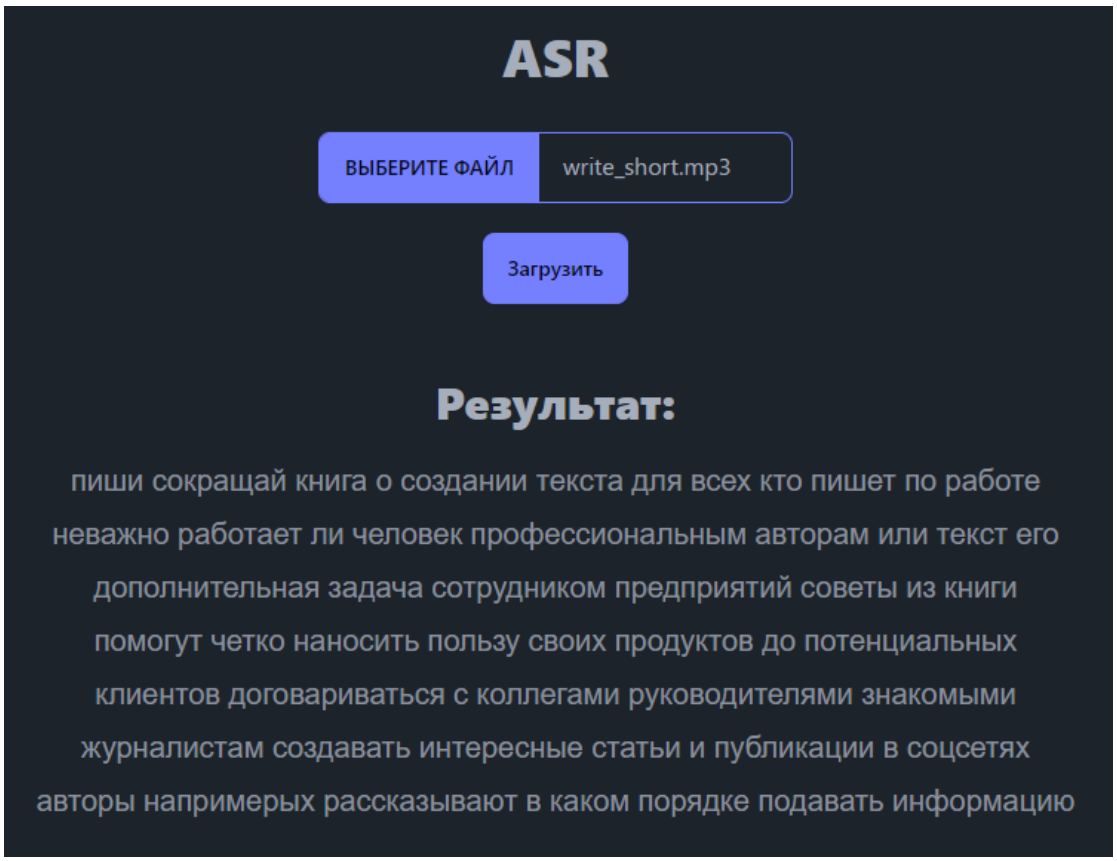

Рисунок 2 – Скриншот UI с результатом работы модели

Серверная часть была быстро реализована на фреймворке Flask. Для этого был создан endpoint для POST запроса, который получает аудиофайл, после обрабатывает его ASR моделью и возвращает результат работы клиентской части. Помимо POST запроса было необходимо настроить CORS чтобы запросы клиента на сервер не блокировались. Код endpoint запроса представлен в листинге 2.

Листинг 2 – Код endpoint запроса

```
@app.route('/asr/upload_audio', methods=['GET', "POST", "OPTIONS"])
def upload_file():
    if request.method == 'OPTIONS':
        return '', 200
    elif request.method == 'POST':
        if 'file' not in request.files:
            return 'No file part'

        file = request.files['file']

        if file.filename == '':
            return 'Файл не выбран', 200

        audio_path = './files/' + file.filename
        file.save(audio_path)
```

```
        transcript = decrypt(audio_path)

        if len(transcript) == 0:
            return jsonify({"text": "Ответа нет"})

        return jsonify({'text': transcript})

    return 'NO RESPONSE', 200
```

Вывод по третьей главе

В рамках данной главы было сделано следующее.

1. Были приведены размеры и количество часов, а также разобраны форматы датасетов Golos и Common Voice. Отдельно разобрана конвертация датасета Common Voice в нужный формат.

2. Проведен предварительный подбор гиперпараметров на 10 часах данных, а также обучена акустическая модель на 150 часах данных с использованием transfer learning.

3. Построена языковая модель с помощью утилиты lmplz, входящей в библиотеку KenLM вместе с ее оптимизацией в бинарный формат, а также разобрано использование CTC-decoders в контексте выбора результата между акустической и языковой моделями.

4. Разработан веб-сервис с клиентской частью с помощью Vite, React, Tailwind CSS и с серверной частью написанной на Flask.

Также было отмечено использование Git как системы контроля версий для проекта, и WSL для удобства работы с библиотеками.

4. ТЕСТИРОВАНИЕ И ЭКСПЕРИМЕНТЫ

Оценка качества работы модели

Во время процесса обучения модель прошла проверку на нескольких аудиофайлах, которые были записаны с использованием настольного микрофона. Метрика WER (Word Error Rate), отражающая качество распознавания речи, показала результаты, варьирующиеся в диапазоне от 16,8% до 8,5%, что довольно хороший показатель. что является довольно хорошим показателем для данной модели. Для сравнения, базовая модель Whisper, протестированная на широко известном датасете LibriSpeech, демонстрирует результат WER на уровне 6,7% [23], что служит ориентиром для оценки эффективности разработанной модели. Пример лучшего результата представлен на рисунке 3.

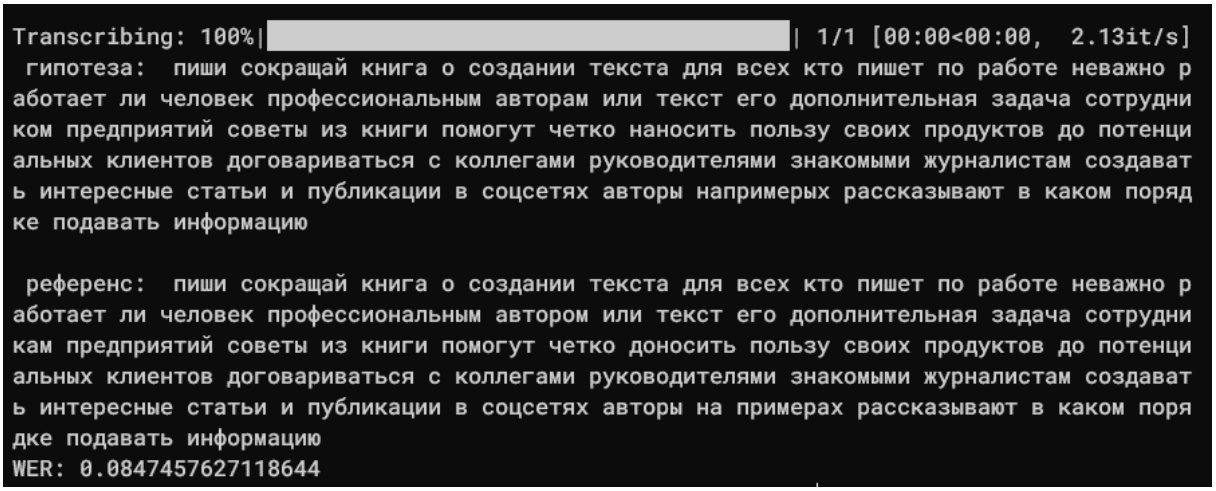

Рисунок 3 – Пример работы модели

В соответствии с информацией, представленной в той же статье, посвященной оценке точности базовой модели Whisper, можно сделать вывод, что значение метрики WER (Word Error Rate) около 7% является достаточно хорошим результатом для offline моделей распознавания речи. Однако, когда речь идет о практическом применении моделей в режиме реального времени (real-time), точность распознавания может быть несколько ниже, что является своеобразным компромиссом в пользу облегчения вычислительной нагрузки и повышения скорости обработки аудиоданных.

Сравнение работы без языковой модели и с ней

Для сравнения был взял аудиофайл с результатом 8,5% и обработан без языковой модели. На примере, представленном на рисунке 4 видно, что без языковой модели акустическая модель может путать окончания. К примеру, вместо «задача» выдавать результат «задачи». Дополнительно стоит отметить, что метрика WER стала равна 17%.

Из этого можно сделать вывод что, если слова сильно похожи по звучанию есть большая вероятность выбора не верного слова в контексте предложения. Для предотвращения таких случаев как раз и была добавлена языковая модель.

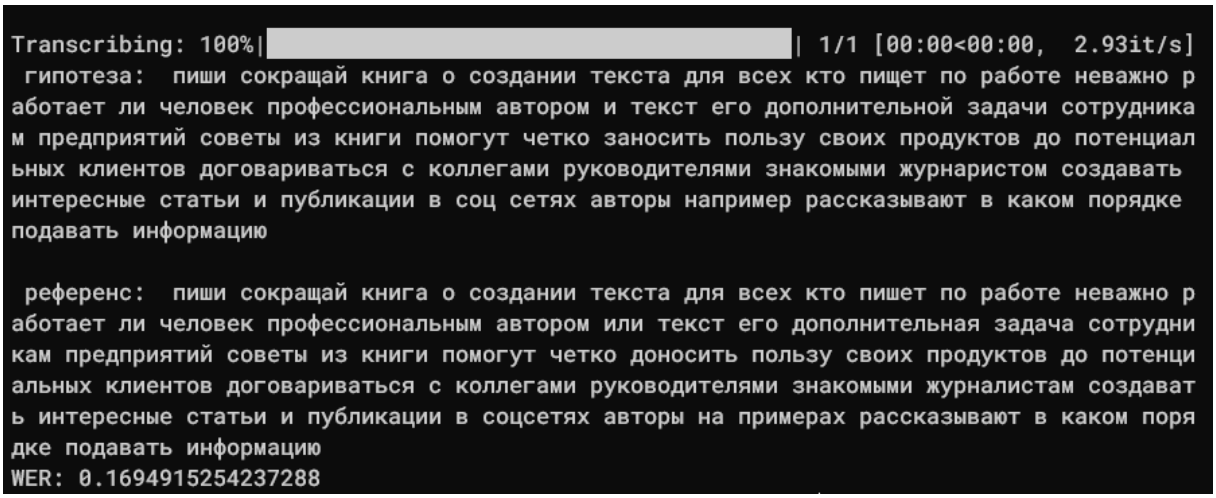

Рисунок 4 – Пример результата без использования языковой модели

Вывод по четвертой главе

В этой главе были оценены и сравнены варианты обработки аудиофайла вместе с языковой моделью и без нее. По результатам сравнения было отмечено, что для достижения хорошего результата работы ASR модель должна использоваться языковая модель. А также отмечено, что для real-time моделей точность работы может быть ниже.

ЗАКЛЮЧЕНИЕ

Из-за важности высокого качества модели, которое определяется его точностью и скоростью, в рамках данной работы было уделено внимание подготовке данных, подбору параметров и использованию transfer learning. Без этой работы качество модели могло получиться низким, что могло бы быть не удовлетворительным при выполнении реальных задач. В ходе работы были выполнены следующие задачи:

1) выполнен анализ предметной области;
2) выбраны технологии реализации;
3) была обучена акустическая модель;
4) была создана языковая модель;
5) был создан веб-сервис для демонстрации модели.

Несмотря на относительно малое время тренировки с ограниченными ресурсами для обучения модели, она показала неплохие результаты и имеет потенциал в использовании в реальных приложениях при соответствующем уровне дообучения.

Несмотря на то, что модель обучена на архитектуре QuartzNet15x5 она может быть адаптирована к более современным архитектурам, например к ContextNet, что позволит ей стать на порядок эффективней.

Эта работа может послужить методическим материалом для тренировки своей модели и использованию ее для своих нужд. Например, создать своего голосового ассистента соответственно дообучив модель на данных записанного голоса. Работать ассистент может не только в привычном виде на мобильном телефоне, но и на пк, также выполняя любые функции, которые только может придумать разработчик.

Модель можно будет использовать и по ее прямому назначению – созданию аннотаций к каким-либо аудиофайлам для экономии времени их транскрибации.

ЛИТЕРАТУРА

1. Deepgram. [Электронный ресурс] URL: https://deepgram.com/ (дата обращения: 08.02.2024 г.).

2. Google cloud speech-to-text. [Электронный ресурс] URL: https://cloud.google.com/speech-to-text/ (дата обращения: 08.02.2024 г.).

3. Whisper AI. [Электронный ресурс] URL: https://openai.com/research/whisper/ (дата обращения: 08.02.2024 г.).

4. Python. [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 10.02.2024 г.).

5. PyTorch. [Электронный ресурс] URL: https://pytorch.org/ (дата обращения: 10.02.2024 г.).

6. NeMo. [Электронный ресурс] URL: https://github.com/NVIDIA/NeMo/ (дата обращения: 12.02.2024 г.).

7. KenLM. [Электронный ресурс] URL: https://github.com/kpu/kenlm/ (дата обращения: 12.02.2024 г.).

8. Flask. [Электронный ресурс] URL: https://flask-docs.readthedocs.io/ (дата обращения: 08.04.2024 г.).

9. React. [Электронный ресурс] URL: https://react.dev/ (дата обращения: 08.04.2024 г.).

10. Ott M., Edunov S., Baevski A., Fan A., Gross S., Ng N., Grangier D., Auli M. Fairseq: A fast, extensible toolkit for sequence modeling. [Электронный ресурс] // arXiv.org. 2019. Дата обновления: 01.04.2019 г. URL: https://arxiv.org/pdf/1904.01038 (дата обращения: 09.03.2024 г.).

11. Radfar M., Barnwal R., Swaminathan R., Chang F., Strimel G., Susanj N., Mouchtaris A. ConvRNN-T: Convolutional Augmented Recurrent Neural Network Transducers for Streaming Speech Recognition. [Электронный ресурс] // arXiv.org. 2022. Дата обновления: 29.09.2022 г. URL: https://arxiv.org/abs/2209.14868 (дата обращения: 16.04.2024 г.).

12. Martinez B., Ma P., Petridis S., Pantic M. Lipreading using temporal convolutional networks. [Электронный ресурс] // arXiv.org. 2020. Дата обновления: 23.01.2020 г. URL: https://arxiv.org/pdf/2001.08702 (дата обращения: 17.04.2024 г.).

13. Moritz N., Hori T., Roux J. Streaming automatic speech recognition with the transformer model. [Электронный ресурс] // arXiv.org. 2020. Дата обновления: 08.01.2020 г. URL: https://arxiv.org/pdf/2001.02674v1 (дата обращения: 09.02.2024 г.).

14. CTC-decoders. [Электронный ресурс] URL: https://github.com/githubharald/CTCDecoder/ (дата обращения: 10.02.2024 г.).

15. FFmpeg. [Электронный ресурс] URL: https://ffmpeg.org/ (дата обращения: 10.02.2024 г.).

16. Vite. [Электронный ресурс] URL: https://vitejs.dev/ (дата обращения: 10.02.2024 г.).

17. Typescript. [Электронный ресурс] URL: https://www.typescriptlang.org/ (дата обращения: 10.02.2024 г.).

18. Ardila R., Branson M., Davis K., Henretty M., Kohler M., Meyer J., Morais R., Saunders L., M. F., Weber G. Common Voice: A Massively-Multilingual Speech Corpus. [Электронный ресурс] // arXiv.org. 2019. Дата обновления: 13.12.2019 г. URL: https://arxiv.org/abs/1912.06670 (дата обращения: 18.04.2024 г.).

19. Karpov N., Denisenko A., Minkin F. Golos: Russian Dataset for Speech Research. [Электронный ресурс] // arXiv.org. 2020. Дата обновления: 23.01.2020. URL: https://arxiv.org/abs/2106.10161 (дата обращения: 17.04.2024 г.).

20. Git. [Электронный ресурс] URL: https://git-scm.com/ (дата обращения: 10.02.2024 г.).

21. WSL. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/windows/wsl/ (дата обращения: 10.02.2024 г.).

22. Heafield K. KenLM: Faster and Smaller Language Model Queries. [Электронный ресурс] // anthology.org. 2011. Дата обновления: 16.09.2011 г. URL: https://arxiv.org/abs/1912.06670 (дата обращения: 20.04.2024 г.).

23. Radford A., Kim J., Xu T., Brockman G., McLeavey C., Sutskever I. Robust Speech Recognition via Large-Scale Weak Supervision. [Электронный ресурс] // arXiv.org. 2023. Дата обновления: 06.12.2023 г. URL: https://arxiv.org/abs/2212.04356 (дата обращения: 21.04.2024 г.).