|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729. 07.04-01 81 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **РАЗРАБОТКА MVP СЕРВИСА СИНТЕЗА РЕЧИ НА АНГЛИЙСКОМ ЯЗЫКЕ**  **Пояснительная записка**  **RU.17701729.** **07.04-01 81 01-1**  **Листов 35** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

Москва 2022

**СОДЕРЖАНИЕ**

[ГОЛОССАРИЙ 4](#_Toc103627848)

[1. ВВЕДЕНИЕ 6](#_Toc103627849)

[1.1 Наименование программы 6](#_Toc103627850)

[1.2 Основания для разработки 6](#_Toc103627851)

[2. НАЗНАЧЕНИЕ РАЗРАБОТКИ И ОБЛАСТЬ ПРИМЕНЕНИЯ 7](#_Toc103627852)

[2.1 Назначение разработки 7](#_Toc103627853)

[2.2 Область применения 7](#_Toc103627854)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 8](#_Toc103627855)

[3.1 Постановка задачи на разработку программы 8](#_Toc103627856)

[3.2 Описание и обоснование выбора применяемых математических методов. 9](#_Toc103627857)

[3.3 Описание алгоритма и функционирования программы 9](#_Toc103627858)

[3.3.1 Acoustic Service 9](#_Toc103627859)

[3.3.2 Telegram Bot Service 10](#_Toc103627860)

[3.3.3 Организация взаимодействия микросервисов. 11](#_Toc103627861)

[3.4 Организация входных и выходных данных 11](#_Toc103627862)

[3.5 Описание и обоснование выбора и состава технических и программных средств 11](#_Toc103627863)

[3.5.1 Серверная часть 11](#_Toc103627864)

[3.5.2 Клиентская часть 12](#_Toc103627865)

[3.5.1 Обоснование выбора и состава технических и программных средств 12](#_Toc103627866)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 13](#_Toc103627867)

[5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc103627868)

[ПРИЛОЖЕНИЕ 1. ACOUSTIC SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ СКРИПТОВ 15](#_Toc103627869)

[ПРИЛОЖЕНИЕ 2. ACOUSTIC SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ 16](#_Toc103627870)

[ПРИЛОЖЕНИЕ 3. ACOUSTIC SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ И ПОЛЕЙ И ГЛОБАЛЬНЫX ПЕРЕМЕННЫX 17](#_Toc103627871)

[ПРИЛОЖЕНИЕ 4. TELEGRAM BOT SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ СКРИПТОВ 21](#_Toc103627872)

[ПРИЛОЖЕНИЕ 5. TELEGRAM BOT SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ 22](#_Toc103627873)

[ПРИЛОЖЕНИЕ 6. TELEGRAM BOT SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ И ГЛОБАЛЬНЫX ПЕРЕМЕННЫX 23](#_Toc103627874)

[ПРИЛОЖЕНИЕ 7. ПРИМЕР МЕЛ-СПЕКТРОГРАММЫ 27](#_Toc103627875)

[ПРИЛОЖЕНИЕ 8. СХЕМА АРХИТЕКТУРЫ ПРОГРАММЫ 28](#_Toc103627876)

[ПРИЛОЖЕНИЕ 10. ПРИМЕР ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С БОТОМ 29](#_Toc103627877)

[ПРИЛОЖЕНИЕ 11. ПРИМЕР ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С БОТОМ 30](#_Toc103627878)

[ПРИЛОЖЕНИЕ 12. ПРИМЕР ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С БОТОМ 31](#_Toc103627879)

[ПРИЛОЖЕНИЕ 13. СТРУКТУРА МОДЕЛИ FASTPITCH 32](#_Toc103627880)

[ПРИЛОЖЕНИЕ 14. ГРАФИК РЕЗУЛЬТАТОВ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ 33](#_Toc103627881)

[ПРИЛОЖЕНИЕ 14. ИНСТРУКЦИЯ ПРОВЕРЯЮЩЕГО АУДИО. 34](#_Toc103627882)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 35](#_Toc103627883)

ГОЛОССАРИЙ

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| TTS (Text-To-Speech), синтез речи | Формирование речевого сигнала по текстовому представлению |
| MVP (Minimum Viable Product) | Начальная версия продукта, обладающая минимальным набором функций, достаточным для ее презентации и тестирования на первых потребителях. |
| Микросервисная архитектура | Подход к разработке ПО, при котором монолита системы разбивается на отдельные независимые компоненты - сервисы. |
| Семиотический класс | Множество лексических единиц, которые обозначают объекты одного вида и имеют схожие правила написания и произношения. |
| Нормализация текста | Процесс, при котором текст разбивается на лексические единицы, каждая из которых соотносится с определенным семиотическим классом. |
| NLP (Natural Language Processing) | Обработка естественного языка. |
| Фонема | Единица звукового строя языка. |
| Mel шкала | Шкала, основанная на восприятии тонов (частот) человеком. Слушатели оценивают любые частоты на расстоянии 1 мел, как равные по расстоянию друг от друга. |
| Мел-спектрограмма | Спектрограмма, в которой частоты преобразуются в шкалу mel (Приложение 7) График, в котором по оси x показано время, по оси y – частота. Цветом выделяется амплитуда (Дб). |
| Вокодер (vocoder) | Устройство (или алгоритм), которое синтезирует речь на основе некоторой информации о ней. |
| Акустическая модель | Модель, которая принимает в качестве входных данных признаки на небольшом участке акустического сигнала (фрейме) и подает на выход распределение вероятностей различных фонем на этом фрейме. |
| GPU (graphics processing unit) | Специализированный вид микропроцессора |
| CUDA | Программно-аппаратная архитектура вычислительных процессов, которая использует процессоры Nvidia |

1. ВВЕДЕНИЕ
   1. Наименование программы

Наименование программы на русском языке: «Разработка MVP сервиса синтеза речи на английском языке».

Наименование программы на английском языке: «Development of an MVP service for speech synthesis in English».

* 1. Основания для разработки

Разработка ведется на основании учебного плана подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденной академическим руководителем программы «Разработка MVP сервиса синтеза речи на английском языке».

1. НАЗНАЧЕНИЕ РАЗРАБОТКИ И ОБЛАСТЬ ПРИМЕНЕНИЯ
   1. Назначение разработки

Программа предоставляет возможность пользователям мессенджера Telegram переводить текстовые сообщения на английском языке в аудиофайлы c записью речи на английском языке, которая является озвучкой введенного текста.

* 1. Область применения

Синтезом речи называется процесс, в ходе которого из текста на некотором языке создается аудиo, содержащего речь, информационное сообщение в которой носитeль данного языка воспринимает как эквивалентное, передаваемому текстом. Синтез речи часто используется в системах с голосовым управлением, например, в голосовых помощниках.

Вo некоторых ситуациях возможно предварительно записать куски речи диктора и “склеивать” их при необходимости вывода информации в виде аудио. Такой подход является жизнеспособным в системах с oограниченном числом реплик, например, в навигаторах. Однако, в более сложных программах требуются синтезирование произвольных фраз, что делает описанный ранее подход невозможным. Данная программа использует NLP-подходы, что помогает решать описанные ограничения и позволяет превращать произвольные фразы на английском языкe в речь.

Данная программа является MVP проектом, представляющим Telegram бота, позволяющего синтезировать аудио файл с речью на английском языке по введенному пользователем тексту.

Данная пояснительная записка раскрывает описывает детали двух микросервисов, являющихся составными частями командного проекта: Acoustic Service и Telegram бота.

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
   1. Постановка задачи на разработку программы

В рамках командной работы необходимо реализовать МVP сервис для синтеза речи на английском языке. Программа имеет микросервисную архитектуру. В ее состав входят следующие микросервисы: Normalization Service (Нормализация текста), Acoustic Service (Генерация мел-спектрограмм по нормализованному тексту), Vocoder Service (Генерация аудио по мел-спектрограммам), Dispatcher Service (Обеспечивает взаимодействие описанных ранее сервисов), Telegram Bot Service (Telegram bot, обеспечивающий взаимодействие пользователя и сервисов синтеза речи, предоставляет информацию о работе программы разработчикам).

Таким образом, схему работы конечного программного продуктa (Приложение 8) можно описать следующим образом. Текст сообщения на английском языке, отправленного пользователем боту на платформе мессенджера Telegram, передается в качестве параметра http запроса Telegram Bot сервисом Dispatcher сервису. Dispatcher Service в свою очередь передает текст в параметрах http запроса к Acoustic Service и получает от последнего ответ в формате json, cсодержащий нормализованный текст. Затем Dispatcher Service передает нормализованный текст в параметрах http-запроса к Acoustic Service, от которого получает json ответ, содержащий мел-спектрограмму. Полученную мел-спектрограмму при помощи http-запроса передается Dispatcher сервисом Vocoder сервису, который возвращает json ответ, содержащий аудио сообщение в формате wav. Данный ответ Dispatcher Service возвращает Telegram Bot сервису, который в свою очередь отправляет через бота сообщение пользователю, содержащее аудио файл в формате wav, озвучивающий исходное сообщение.

В процессе работы Dispatcher сервис кэширует полученные аудио (отправляет пользователю без повторной генерации ответ в случае, если такой текст уже был обработан ранее), и статистическую информации об обработке запроса.

Для разработчиков (заранее определенного круга пользователей бота) доступны также дополнительные команды бота, позволяющие получить сообщение о статистике, собранной Dispatcher service, а также возможность провести нагрузочное тестирование сервисов синтеза речи с заданными параметрами и получить сообщение об его результатах.

В рамках индивидуальных задач при реализации командного продукта необходимо:

1. Изучить информацию об open source реализации модели FastPitch [11], основанную на архитектуре Transformer [9];
2. Обучить модель FastPitch [8] на датасете LJSpeech-1.1[13];
3. Реализовать Acoustic Service на основе обученной модели;
4. Реализовать Telegram Bot Service;
5. Обеспечить возможность запуска Acoustic Service и Telegram Bot Service в Docker container [10].
6. Обеспечить взаимодействие всех микросервисов MVP с помощью Docker Compose [12].
7. Обработать информации, полученную при нагрузочном тестировании с различными параметрами.

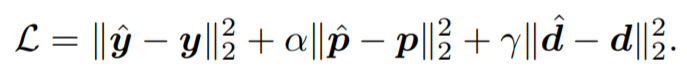
Более подробное описание задач разработки по каждому микросервису и всему продукту в целом представлено в командном Техническом задании, личном Техническом задании, а также в отчетах, подготовленных остальными участниками команды.

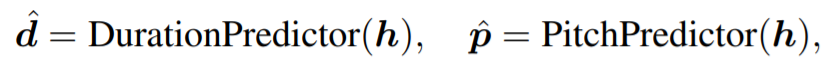
* 1. Описание и обоснование выбора применяемых математических методов.

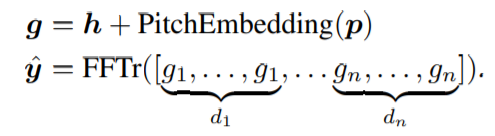
При реализации Acoustic Service использована модель Fastpitch, основанная на архитектуре Transformer. Детальное описание архитектуры представлено в статье [9]. Детальное описание модели Fastpitch представлено в статье [8].

Модель Fastpitch построена на основе 2-х feed-forward трансформеров (FFT). После первичного преобразования входных токенов первый блок трансформера генерирует представление входной последовательности (h), которое используется для предсказания длительности и средней высоты звука, соответствующего каждому символу. Результаты объединяются и отправляются во второй блокFFT, который преобразует данные в мел спектрограммы.

При обучении вместо предсказаний о длительности и высоте звука используются реальные данные. Модель оптимизирует среднеквадратичную ошибку (MSE):

, где





Модель является параллельной, позволяет быстро синтезировать высокоточные мелкомасштабные спектрограммы с высокой степенью контроля над параметрами речи. Модель предоставляет возможность для регуляции темпа, выразительности, высоты тона и интонации, что позволяет сделать синтезируемую речь более естественной.

* 1. Описание алгоритма и функционирования программы
     1. Acoustic Service

Сервис Acoustic Service – является оболочкой для взаимодействия с обученной моделью преобразования речи в мел-спектрограммы. Сервис отвечает на Get запросы формата;

http://{адрес сервиса в локальной сети}:5000/?{текст+сообщения+пользователя}

Дальше сервис отправляет текст на обработку в экземпляр класса MelGenerator, обеспечивающего взаимодействия с Open Source кодом модели Fastpitch.

После обработки запроса сервис возвращает ответ, содержащий матрицу мел-спектрограммы, сериализованную в формате json.

Для обеспечения работоспособности микросервиса, данные об обученной модели Fastpitch хранятся в файле FastPitch\_checkpoint\_1000.pt и загружаются при первом обращении к сервису.

Описание диаграммы классов представлено в приложении.

Для запуска программы в docker container написан файл с инструкциями docker Dockerfile

* + 1. Telegram Bot Service

Сервис обеспечивает взаимодействие пользователя через бот на платформе мессенджера Telegram c Dispatcher Service.

Телеграм бот отвечает начинает работу при вводе пользователем текстового сообщения с командой ‘/start’. После начала работы пользователю выводится приветственное сообщение, в котором предлагается ввести текст на английский. Дальнейшие текстовые сообщения пользователя Telegram bot service обрабатывает и отправляет пользователю через бота в ответ на текстовое сообщение аудио файл в формате wav в случае корректного завершения работы всех микросервисов или сообщения об ошибке в случае ошибки хотя бы на одном этапе обработки текста.

После получения текстового сообщения от пользователя Telegram bot Service отправляет запрос в формате:

http://{адрес Dispatcher service в локальной сети}:7000/?{текст+сообщения+пользователя}

к Dispatcher Service. При корректном завершении работы описываемый сервис получает ответ, содержащий wav файл, сериализованный в формат json. Полученный файл отправляется пользователю. В случае ошибки на любом этапе пользователю выводится сообщение об ошибке. Пример диалога пользователя и бота Приложение 10.

Разработчикам (заранее определенному кругу пользователей бота) доступны дополнительные команды.

При отправке пользователем боту команды ‘/analytics’ Telegram bot Service отправляет запрос в формате:

http://{адрес Dispatcher service в локальной сети}:7000/analytics

Полученные от Dispatcher service в формате json текстовые данные и изображения сервис отправляет через бот сообщением пользователю. Пример диалога пользователя и бота Приложение 12.

При отправке пользователем боту команды ‘/latency\_test {Количество параллельных пользователей} {время теста в секундах} ’ Telegram bot Service проводит нагрузочное тестирования. При нагрузочном тестировании сервер эмитирует нагрузку, создаваемую заданным количеством параллельными пользователями на протяжении заданного периода времени, и замеряет показатели. Каждый пользователь отправляет примерно раз в секунду запрос на синтез аудио к Dispatcher service, содержащий недлинный стандартный текст (текст незначительно модифицируется при каждом запросе, с целью избежать выдачи кэшированного результата). Текстовое описание результатов и диаграмма отправляются пользователю через Telegram бота. Пример диалога пользователя и бота Приложение 11.

Описание диаграммы классов представлено в приложении.

Для запуска программы в docker container написан файл с инструкциями docker Dockerfile.

* + 1. Организация взаимодействия микросервисов.

Каждый микросервис запускается в отдельном Docker контейнере Инструкции для программы docker [10] представлены в отдельных для каждого микросервиса файлах Dockerfile.

Инструкции для отдельного запуска и взаимодействия с микросервисами описаны в файлах ReadMe.md.

Также для запуска всех микросервисов при помощи команды ‘docker-compose up –build’ из корня MVP прописаны инструкции программе docker-compose [12] в файле docker-compose.yml. При запуске данной команды каждый микросервис запускается в отдельном контейнере, при этом контейнеры объединяются в локальную сеть, что обеспечивает взаимодействие сервисов.

* 1. Организация входных и выходных данных

Ввод данных производит пользователь в диалоге с ботом на платформе мессенджера Telegram, путем отправления текстового сообщения с одной из доступных команд или текстом на английском языке (подробное описание в пункте 3.3).

Вывод программы также осуществляется в диалоге пользователя с ботом. Бот присылает в зависимости от типа обрабатываемого запроса изображение, текст или аудио файл в сообщении пользователю.

* 1. Описание и обоснование выбора и состава технических и программных средств
     1. Серверная часть

Программа реализована на языке программирования Python 3.6

Программа запускается на операционной системе Ubuntu 20.04.4.

Программа должна использовать NVIDIA® Tesla® V100 в Yandex Cloud для синтеза аудио.

Для запуска необходимо наличие программ Docker и Docker Compose

Необходимо иметь зарегистрированного в телеграмме бота.

Полный список библиотек необходимых для запуска каждого сервиса описаны в фалах в соответствующих файлах requirements.txt.

* + 1. Клиентская часть

Для подключения программы к платформе Telegram необходимо иметь зарегистрированного в этом мессенджере бота.

Пользователю программы необходимо иметь быть зарегистрированным пользователем Telegram и иметь к мессенджеру.

* + 1. Обоснование выбора и состава технических и программных средств

Для реализации программы был выбран язык Python, так как на момент написания язык является одним из самых распространённых инструментов для работы с машинным обучением. Также выбор такого состава средств обеспечивает совместимость с используемыми open source реализациями моделей синтеза речи.

Каждый микросервис запускается в отдельном docker container, также используется docker compose, что позволяет разделить разработку независимых сервисов между участниками команды, а также облегчить работу по запуску готовой программы, сделав автоматизированным загрузку и запуск необходимых сред для микросервисов.

1. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Было проведено нагрузочное тестирование микросервисов синтеза речи (Dispatcher service, Normalization service, Acoustic service, Vocoder service) на параллельных пользователях на протяжении трех минут. Каждый пользователь делал примерно 1 запрос в секунду, содержащий уникальный текст длиной около 50 символов. Время ответа на запросы оказалась приемлемым для MVP проекта (Приложение 14). В случае если нагрузку на сервисы осуществлял только один пользователь время ответа в среднем составляло меньше секунды. При росте нагрузки до двух-трех пользователей среднее время ответа сервера возрастает до 2 секунд. При дальнейшем увеличении количества пользователей среднее время ответа также возрастало. Так при максимальной для MVP нагрузки в 10 параллельных пользователей среднее время синтеза речи увеличивается до 6 секунд, при этом максимальное время ответа за все время тестирования (достигалось при 9 параллельных пользователях) было меньше 14 секунд.

Замеры скорости синтеза речи (без учета времени необходимого для отправления результата пользователю) показали, что за одну секунду сервисы способны синтезировать 4.5 секунды аудио. Что означает, что сервер генерирует аудио в 4.5 раза быстрее чем в realtime, что является достаточным результатом для MVP.

Также было проведен замер качества синтеза речи на тестовой выборке датасета LJSpeech-1.1[13] состоящей из 500 текстов и аудио. В ходе тестирования проверяющие (83 человека), следуя инструкции (Приложение 14), оценивали качество речи как реальных аудио, так и сгенерированных сервисами синтеза речи (текст аудио был идентичен). В результате тестирования речь, синтезируемая сервисами, получила 3.8 балла из 5 возможных, в то время как реальные аудио получили 4 балла из 5.

Данная работа является MVP, цель которого продемонстрировать возможность синтеза речи из текста. Экономическая оценка данной программы и ее сравнение с полноценными системами синтеза речи не является целесообразной и не предусматривается в рамках данного проекта.

1. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ
2. ГОСТ 19.102-77 Стадии разработки. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.101-77 Виды программ и программных документов. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.104-78 Основные надписи. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.105-78 Общие требования к программным документам. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. Adrian Łancucki // FastPitch: Parallel Text-to-speech with Pitch Prediction// NVIDIA Corporation //[Электронный ресурс]: Technical Report, 2021 – Режим доступа: <https://arxiv.org/abs/2006.06873>, свободный. (дата обращения: 1.05.22).
10. Ashish Vaswani// Attention Is All You Need // Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin // [Электронный ресурс]: Technical Report, 2017 – Режим доступа: <https://arxiv.org/abs/1706.03762>, свободный. (дата обращения: 1.05.22).
11. Docker overview [Электронный ресурс] /Documentation. Режим доступа: <https://docs.docker.com/get-started/overview/> свободный. (дата обращения: 1.05.22)
12. NVIDIA, FastPitch 1.1 for PyTorch [Электронный ресурс] / GitHub. Режим доступа: <https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/FastPitch>, свободный. (дата обращения: 1.05.22)
13. Overview of Docker Compose [Электронный ресурс] /Documentation. Режим доступа: <https://docs.docker.com/compose/> свободный. (дата обращения: 1.05.22)
14. The LJ Speech Dataset [Электронный ресурс] / Dataset. Режим доступа: [https://keithito.com/LJ- Speech-Dataset/](https://keithito.com/LJ-%20Speech-Dataset/), свободный. (дата обращения: 1.05.22)

ПРИЛОЖЕНИЕ 1. ACOUSTIC SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ СКРИПТОВ

Директории cmudict, faspitch, common содержат файлы оpen source реализации модели faspitch.

|  |  |
| --- | --- |
| **Скрип** | **Описание** |
| app.py | Описание работы сервиса (Flask приложение); точка входа в сервис. |
| inference.py | Описание MelGenerator |
| check.py | Пример запроса к сервису |

ПРИЛОЖЕНИЕ 2. ACOUSTIC SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| MelGenerator | Представление обученной модели, взаимодействие с open source кодом модели Faspitch для сгенерации мел-спектрограммы по введенному тексту. |
| MeasureTime (Вложенный класс в MelGenerator) (supper – list) | Измерение времени при работе модели. |

ПРИЛОЖЕНИЕ 3. ACOUSTIC SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ И ПОЛЕЙ И ГЛОБАЛЬНЫX ПЕРЕМЕННЫX

**Скрипт app.py**

|  |  |
| --- | --- |
| **Глобальная переменная** | **Описание** |
| mel\_generator | Экземпляр класса MelGenerator, который является представлением обученной модели, загруженной из файла FastPitch\_checkpoint\_1000.pt |

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| get\_mel | Метод вызывается при обращении к сервису по запросу вида: http://{адрес сервиса в локальной сети}:5000/?{текст+сообщения+пользователя}  Метод возвращает значение сериализованную в формат json матрицу мел-спектограммы.  Метод вызывает обработку текста экземпляром класса MelGenerator |

**Скрипт inference.py**

**Класс MelGenerator** (все поля не статические)

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| generator | Представление обученной модели Fastpitch |
| p\_arpabet | Флаг использования ARPABET набора фонетических транскрипций |
| cmudict\_path | Путь к словарю с фонемами. |
| device | Представления устройства, на котором запускается скрипт. |
| checkpoint | Путь к файлу, содержащему обученную модель |
| cudnn\_benchmark | Флаг включения cudnn benchmark mode |
| output | Директория для сохранения выходных данных программы |
| cuda | Истина, если сервис запускается на gpu с использованием cuda |
| pace | Регулирует скорость речи |
| speaker | id спикера |
| batch\_size | Размер пакетов, на которые разбиваются входные данные |
| symbol\_set | Тип символов |
| text\_cleaners | Тип очисток символов |
| sampling\_rate | Частота дискретизации |
| log\_file | Путь к файлу с логами сервиса |
| pitch\_transform\_amplify | Изменчивость высоты тона (регулирует эмоциональность речи) |
| pitch\_transform\_custom | Флаг использования нестандартного преобразования речи |
| pitch\_transform\_flatten | Флаг выравнивания речи |
| pitch\_transform\_invert | Флаг изменения инверсии интонации относительно среднего значения. |
| pitch\_transform\_shift | Регулирует высоту голоса |
| args | Остальные аргументы модели |

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Параметры** | **Статический** | **Описание** |
| \_\_init\_\_ | - | Нет | Конструктор класса, в котором загружается обученная модель и задаются параметры генерации мел-спектрограммы. |
| parse\_args | parser – Парсер аргументов командной строки | Нет | Добавление аргументов для модели |
| load\_model\_from\_ckpt | model – модель | Нет | Загрузка обученной модели из файла |
| load\_and\_setup\_model | parser—Парсер аргументов командной строки  unk\_args – Некоторые аргументы модели | Нет | Загрузка модели и установка параметров |
| prepare\_input\_sequence | fields – Словарь, содержащий входной текст | Нет | Подготовка исходного текста, разбиение текста на куски. |
| build\_pitch\_transformation | - | Нет | Предсказание длины и задание параметров речи. |
| get\_mel | text - входной текст на английском языке | Нет | Преобразует текст на английском языке в мел-спектрограмму. |

**Класс MeasureTime**

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Параметры** | **Статический** | **Описание** |
| \_\_init\_\_ | args (необязательный) – количество параметров  cudа – тип используемого устройства  kwargs (необязательный) – словарь параметров | Нет | Конструктор |
| \_\_enter\_\_ | - | Нет | Замер стартового времени. |
| \_\_exit\_\_ | - | Нет | Замер времени окончания и добавление продолжительности |
| \_\_add\_\_ | other – Другой экземпляр данного класса. | Нет | Сложение времени вычисления двух экземпляров класса. |

**Скрипт model.py**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Параметры** | **Описание** |
| get\_model\_config | args – параметры модели | Создание конфигурации модели |
| get\_model | model\_config - конфигурация модели  device – отображение используемого устройства | Инициализация модели |
| init\_bn | model - модель | Инициализация Batch Normalization |
| parse\_model\_args | Parser – Парсер аргументов командной строки  add\_help – Флаг вывода подсказок | Парсинг аргументов модели |

ПРИЛОЖЕНИЕ 4. TELEGRAM BOT SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ СКРИПТОВ

|  |  |
| --- | --- |
| **Скрип** | **Описание** |
| main.py | Описание обработки сообщений ботом |
| states.py | Описание состояний бота |
| text.py | Описание текстов, используемых ботом |
| config.py | Файл содержит 2 переменные Token – токен бота в телеграмме и ADMIN\_USER\_IDS – список id пользователей- разработчиков, которым доступны дополнительные команды. Из-за конфиденциальности информации не добавляется в репозиторий проекта. |
| admin\_analytics.py | Описание алгоритма выполнения дополнительных команд бота, доступных разработчикам |

ПРИЛОЖЕНИЕ 5. TELEGRAM BOT SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ

**Скрипт states.py**

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| User (supper – StatesGroup) | Класс определяющий состояния пользователя при общении с ботом |

**Скрипт admin\_analytics.py**

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| User (supper – HttpUser) | Класс определяющий поведение пользователя (создаваемого программой) при нагрузочном тестировании |
| SafeMassage | Сохранение информации во время нагрузочного тестирования |

ПРИЛОЖЕНИЕ 6. TELEGRAM BOT SERVICE: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ И ГЛОБАЛЬНЫX ПЕРЕМЕННЫX

**Скрипт main.py**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Параметры** | **Описание** |
| start\_message | message – описание сообщения  state – состояние пользователя | Вывод приветственного сообщения после ввода команды ‘/start’ |
| get\_analytics | message – описание сообщения  state – состояние пользователя | Вывод статистики работы программы посде ввода команды '/analytics’ |
| latency\_test | message – описание сообщения  state – состояние пользователя | Вывод результатов нагрузочного тестирования посде ввода команды  '/latency\_test' |
| enter\_text\_message | message – описание сообщения  state – состояние пользователя | Преобразование введенного текстового сообщения в аудио и отправка результата пользователю |
| wrong\_command | message – описание сообщения  state – состояние пользователя | Уведомление пользователя о некорректной работе бота с инструкциями по возобновлению диалога. |

|  |  |
| --- | --- |
| **Глобальная переменная** | **Описание** |
| bot | Телеграм бот |
| MAX\_TEXT\_LENGTH | Максимальная возможная длина текстового сообщения |

**Скрипт states.py**

**Класс SafeMassage** (все поля не статические)

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Entering\_text | Состояние, в котором пользователь после начала диалога вводит текст для синтеза речи по нему. |

**Скрипт admin\_analitics.py**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Параметры** | **Описание** |
| parse\_analytics | data – словарь информации (текста и изображений) переданный Dispatcher service в качестве ответа на запрос аналитика работы сервиса  prefix – отступ от начала строки | Рекурсивная обработка словаря информации (текста и изображений) переданного Dispatcher service в качестве ответа на запрос аналитика работы сервиса |
| latency\_test | user\_count– количество параллельных пользователей  test\_time – время тестирования в секундах | Проведение нагрузочного тестирования с заданным числом пользователей в течение заданного промежутка времени |

**Класс User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Параметры** | **Статический** | **Описание** |
| My\_task | - | Нет | Выполнение запроса |

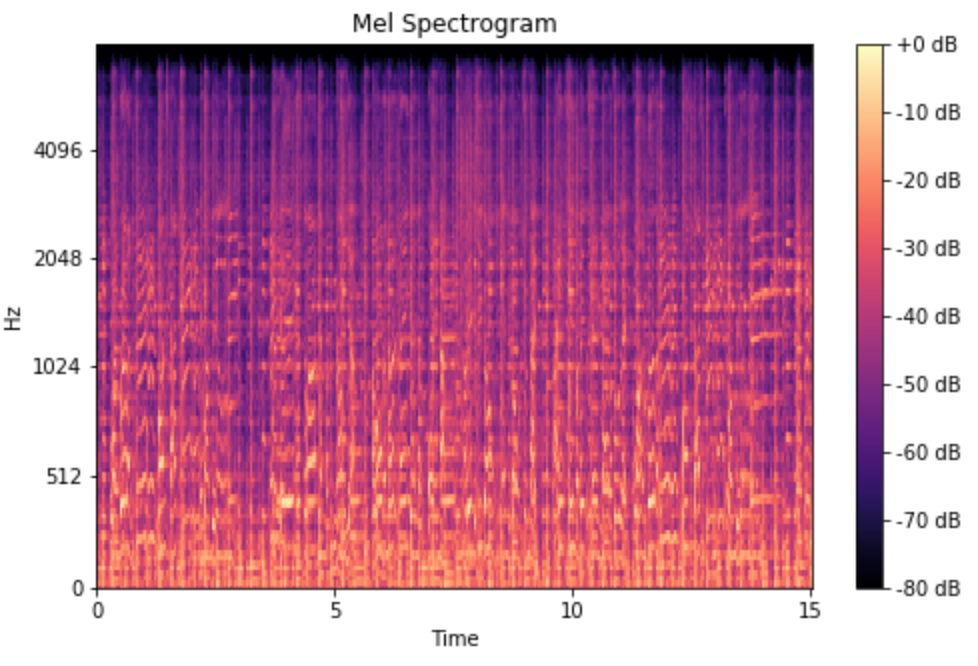
|  |  |
| --- | --- |
| **Переменная** | **Описание** |
| wait\_time | Функция, определяющая ожидание пользователя между запросами |
| host | Адрес dispatcher\_service |

**Класс SafeMassage** (все поля не статические)

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| data | Словарь, в котором хранится информация о запросах |
| log | Сообщение, отправляемое пользователю |

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Параметры** | **Статический** | **Описание** |
| \_\_init\_\_ | env – среда выполнения нагрузочного тестирования | Нет | Конструктор |
| get\_message | users\_count – количество параллельных пользователей  test\_time – время тестирования | Нет | Генерация текстового сообщения, содержащего статистику тестирования |
| get\_data | - | Нет | Получение таблицы с данными тестирования |
| get\_image | users\_count – количество параллельных пользователей  time – время тестирования | Нет | Получение графика, характеризующего тестирование |
| on\_request | request\_type – тип запроса  name – тело запроса  response\_time – время обработки запроса  response\_length – длина текста запроса  exception – исключение, если оно возникало при запросе  context – ответ на запрос  start\_time – время начала запроса  kwargs (необязательные) – дополнительные аргументы | Нет | Сбор информации каждый раз, когда пользователь отправляет запрос. |

ПРИЛОЖЕНИЕ 7. ПРИМЕР МЕЛ-СПЕКТРОГРАММЫ



ПРИЛОЖЕНИЕ 8. СХЕМА АРХИТЕКТУРЫ ПРОГРАММЫ

Пользователь

ПРИЛОЖЕНИЕ 10. ПРИМЕР ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С БОТОМ

Реплики бота могут незначительно отличаться от приведенного в приложении

Изображение выглядит как текст

Автоматически созданное описание

ПРИЛОЖЕНИЕ 11. ПРИМЕР ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С БОТОМ

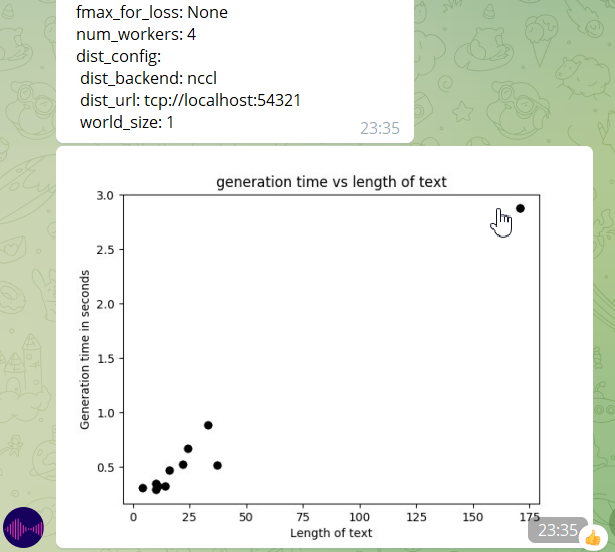
Реплики бота могут незначительно отличаться от приведенного в приложении

Изображение выглядит как текст

Автоматически созданное описание

ПРИЛОЖЕНИЕ 12. ПРИМЕР ДИАЛОГА ПОЛЬЗОВАТЕЛЯ С БОТОМ

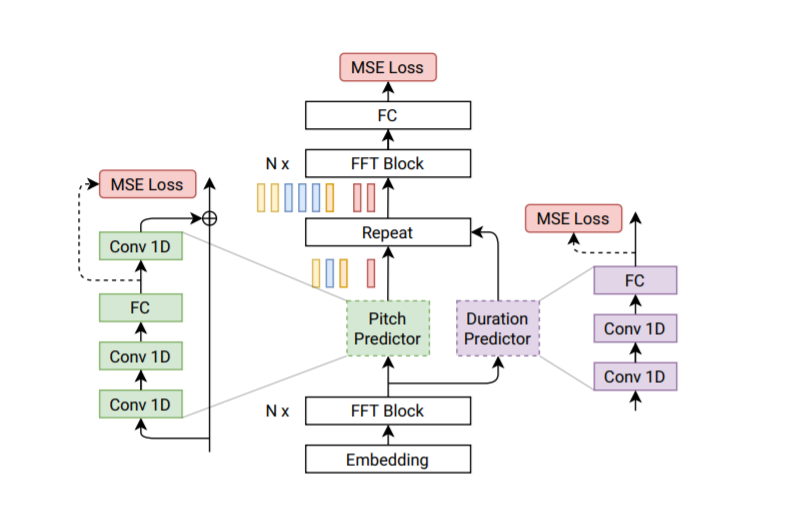
Реплики бота могут незначительно отличаться от приведенного в приложении

 Изображение выглядит как текст

Автоматически созданное описание

ПРИЛОЖЕНИЕ 13. СТРУКТУРА МОДЕЛИ FASTPITCH

Источник изображения – [11]



ПРИЛОЖЕНИЕ 14. ГРАФИК РЕЗУЛЬТАТОВ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ

ПРИЛОЖЕНИЕ 14. ИНСТРУКЦИЯ ПРОВЕРЯЮЩЕГО АУДИО.

IMPORTANT: in the task you have to listen to audio. Please do not execute it if at least one of the following conditions is true:

You don't have headphones on right now

You have hearing problems

There is noise in the background that interferes with listening (metro, street noise, people talking, etc.)

In this task, you need to listen to an audio recording and rate the naturalness of speech on a scale of 1 to 5.

1 - bad, speech is not natural at all

2 - unsatisfactory, mostly unnatural speech

3 - satisfactory, speech is equally natural and unnatural

4 - good, mostly natural speech

5 - excellent, speech is indistinguishable from human

Also, in controversial cases, it is possible to mark values between different ratings (for example, a value between satisfactory and good as 3.5).

Please listen to each audio at least twice, with a pause of at least one second between different examples.

When grading you need to focus on intonation, fluency of speech, its speed, loudness and some extraneous sounds.

You can also add some of your comments on the audio recording in the comments.

Good luck!

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |