Техническая документация к решению команды [RASCAR] O2

Сервис по диагонстике, мониторингу и профилактике речевых деффектов

Основные ссылки

Сама платформа https://tbankspeech.ru (https://tbankspeech.ru) (логин: test, пароль: 123456)

Свагер к АПИ https://tbankspeech.ru/api/docs/ (https://tbankspeech.ru/api/docs/)

Оркестратор для запуска цикла обучения: http://188.225.34.42:3000/ (http://188.225.34.42:3000/)

Оглавление

Метрики

Macro F1 на приватной выборке: 0.5426 (1 место лидерборда всех этапов)

Общая концепция

Обучение происходит в несколько этапов В качестве модели была взята модель wav2vec

1. **Предобучение** на предсказания кол-ва "маркерных" букв - букв, которые помогут модели лучше находить потенциальное место, где наблюдается деффект речи. На этом этапе вся разметка выполняется с помощью модели whisper. Напрмер, мы использовали

- для картавости - букву "Р"
- для фрикативного Г - букву "Г"
- для заикания не понадобилось добавлять дополнительных букв в этап предобучения, что говорит о робастности получившейся модели;

2. **Обучение** - решение целевой задачи предсказания деффектов речи. Обучается несколько голов для предсказания по словам, в которых встречаются и не встречаются маркерные буквы соответственно.

Общая архитектура сервиса

1. ML-сервис - запускается инференс полученной модели

- инференс может быть запущен на любом сервере, в том числе без графического процессора (RAM 4gb +);
- инференс модели построен на основе фрейморка **BentoML**, что делает систему более производительной, устойчивой к отказам и масштабируемой;
- интерфейс взаимодействия - **RestAPI**, подробную информацию по методам можно найти в свагере;

2. Веб-сервис - реализует интерфейс для взаимодействия с системой;

- **функциональные** возможности:
  - авторизация;
  - прохождение тренировки (пользователь читает сгенерированную по его критериям скороговорку, система оценивает наличие дефектов речи);
  - анализ статистики;
  - обработка сразу большого набора звонков (например, в контексте применения для колл-центров);
- реализован на основе фреймворка **Streamlit**;
- приложение **адаптировано** под экраны любого формата;

3. Оркестратор

- расположен в одном виртуальном окружении с основным ML сервисом - чтобы не тратить время на передачу тяжелых файлов обученных моделей;
- функциональные возможности:
  - конфигурировать, запускать весь пайплайн обучения (включай претрейн и трейн) на любых данных;
  - внутри пайплайн обучения представляется в виде ориентированного графа DAG, что позволяет эффективно управлять данными для запуска дополнительных пайплайнов;
  - в случае успешного завершения пайплайна полученную модель сразу же можно использовать в API - все нобходимые файлы конфигурацию генерируются автоматически;
- одно из возможных применений - регулярное дообучение модели на данных приложения (без разметки можем использовать данные для претрейна, если разметить - для всего пайплайна)

4. Ноутбуки с обучением моделей - также мы предоставляем ноутбуки с кодом для воспроизведения всех этапов обучения.

Код по-сути аналогичен коду оркестратора, но в исходном формате - jupyter notebook-ов

- необходимо установить зависимости, такие же, которые устанавливаются в докерфайле (см. requirements.txt);
- код разделен на несколько частей (транскрибация виспером, предобработка, претрейн, трейн + инференс)
- реализация универсальна и может быть гибко настроена под новые деффекты речи;
- код использует **cometML** для удобного

Инструкция по использованию оркестратора:

1. Необходимо сформировать архив с данными и загрузить на гугл диск. Формат данных:

```
data_train.csv - здесь лежит разметка для аудио
data_mp3
        |
        train
             | (здесь должны лежать все аудио)
             | (здесь должны лежать все аудио)
             | (здесь должны лежать все аудио)
             | (здесь должны лежать все аудио)
```

Все это необходимо заахивировать в .zip, загрузить на гугл диск и прикрепить ссылку в оркестратор. Пример ссылки на архив тренировочных данных: https://drive.google.com/file/d/16NmrEiqS5Up_jcwbGNAC62yAfR3ZkJrs/view?usp=drive_link (https://drive.google.com/file/d/16NmrEiqS5Up_jcwbGNAC62yAfR3ZkJrs/view?usp=drive_link) (мини-пример для тестирования механизма оркестирования)

Инфраструктура

Сервис с фронтендом отправляет запросы в сервис с инференсом. Пайплайн с дообучением запущен в том же контейнере, что и инференс. Все сервисы контейниризованы (запущены в Docker-е), что делает нашу систему быстрой для разворачивания и позволяет ей бесшовно интегрироваться во внутрейнний контур компании.

Пример обращения к API

```
# file_path - путь до аудио
# file_name - имя соответствующего аудио

with open(file_path, "rb") as audio_file:
    files = {"file": (file_name, audio_file, "audio/mpeg")}
    response = requests.post(url, files=files)

data = json.loads(response.content.decode('utf-8'))
```

Запуск ML-сервиса

Сервис с API и оркестрацией дообучения можно запустить следующей командой

```
>>> docker build . -t app:latest
>>> docker run -p 80:80 -p 443:443 -p 8000:8000 -p 8501:8501 -p 3000:3000 --gpus all app:latest
```

Структура приложения

```
> в корне лежат файлы конфигурации и самого веб приложения
    > api - реализация api и самого инференса
    > pipeline - функции обучения для оркестратора Dagster
```