Etykietowanie muzyki (music tagging) za pomocą metod sztucznej inteligencji

Raport z Pracowni Inżynierskiej II

Jakub Robaczewski

2022Z

Wstęp

Celem mojej pracy inżynierskiej będzie napisanie aplikacji, która umożliwi etykietowanie muzyki tagami związanymi z gatunkiem muzycznym (np. rock, pop, rap), nastrojem (smutny, wesoły), tempem oraz innymi grupami. Planuje umożliwić wsparcie dla etykietowania własnych utworów oraz tagowanie za pomocą dwóch modeli: MusicNN oraz zagnieżdżeniach OpenL3. Dodatkowo modele te zostaną porównane pod względem skuteczności predykcji.

Zakres prac

W ciągu tego semestru stworzyłem postawy do trenowania i testowania modeli oraz pre-procesing danych z zbioru danych MagnaTagATune.

Struktura projektu

data

Zawiera pliki muzyczne w formatach mp3 oraz pliki przetworzone na format liczbowy .npy, który jest podstawnym formatem wejściowym moich modeli.

models

Zawiera gotowe i wytrenowane modele.

notebooks

Zawiera skryty z eksperymentami, oraz operacje na zbiorze danych.

split

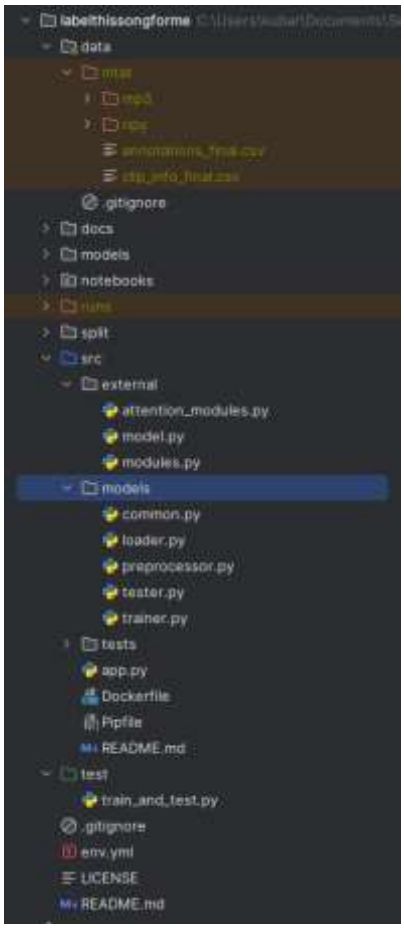
Zawiera ścieżki do plików podzielone na zbiór testowy, walidacyjny i testowy (sam skrypt podziału znajduje się w notebooks/02_MTAT_SPLIT.ipynb.

runs

Zawiera logi TensorBoard, które pozwalają na analizę procesu trenowania.

src

Główny folder kodu źródłowego, zawiera zewnętrznie importowane moduły z gotowych sieci w folderze external oraz klasy trenujące i testujące w folderze models.



Zewnętrzne moduły

Moduły zewnętrzne pobrałem z repozytorium State-of-the-art Music Tagging Models (minzwon/sota-music-tagging-models (github.com)) i pozwalające one na zaimplementowanie modelu Musicnn, który będę wykorzystywał w mojej pracy w porównaniu modelem opartym na zagnieżdżeniach.

Klasy do trenowania i testowania

Podstawowymi klasami w tej części projektu są 3 klasy: PreProcessor, Trainer i Tester.

- Preprocesor wczytuje pliki w skompresowanym formacie .mp3 i zapisuje je w numerycznym formacie .npy (próbkowanie jest zmienną, domyślnie ustawioną na 16000/s).
- Trainer wczytuje pliki, dzieli je na kilka fragmentów o określonej trudności i nadzoruje proces trenowania, zapisując określone zmienne w każdej iteracji, umożliwiając późniejszą analizę tych danych przez narzędzie TensorBoard. Po każdej skończonej iteracji następuje określenie jakości na zbiorze walidacyjnym i ewentualne zapisanie modelu do pliku, jeśli będzie lepszy niż poprzedni.
- Tester testuje działanie nauczonego modelu za pomocą zbioru testowego.

Wszystkie klasy przyjmują ustawienia konfiguracyjne za pomocą ujednoliconej konfiguracji:

- `num_workers – ilość równoległych rdzeni loaderze danych`
- `model – wykorzystywany model`
- `n_epochs – liczba epok`
- `batch_size – rozmiar jednego batcha danych (określa liczbę podziałów jednego utworu)`
- `lr – prędkość uczenia (learning rate)`
- `model_save_path – ścieżka zapisu modelu`
- `data_path – ścieżka do folderu z danymi`
- `log_step – częstotliwość logowania na konsoli`
- `input_length – długość wczytywanego segmentu`
- `train_path, valid_path, test_path – ścieżki do poszczególnych zbiorów danych`
- `binary_path – ścieżka do zbioru połączeń utwór–etykiety`

test

W folderze test znajduje się skrypt testowy do trenowania modeli, który korzysta z klas w folderze src.

Co już zostało zrobione?

- Wybranie zbioru danych i wczytanie ich
- Pre-procesing danych i wykorzystanie ich do nauki sieci
- Implementacja uniwersalnego szkieletu aplikacji pod trenowanie i testowanie
- Implementacja jednego modelu i jego wstępne wytrenowanie

Co będę robił dalej?

- Prosta aplikacja testowa stworzona za pomocą frameworku streamlit
- Drugi model oparty na zagnieżdżeniach OpenL3
- Trenowanie modelu (obecnie trenowanie odbywa się części danych i przez krótki czas)

Podsumowanie

Największymi trudnościami w tym etapie projektu było wybranie odpowiedniego datasetu i wczytanie go. Rozważałem wykorzystanie dużo większego zbioru MillionSongsDataset, który zawiera dane już po preprocesingu, jednak wczytanie danych w formacie numerycznym okazało się niezbyt wygodne. Również rozmiar danych, który był potencjalnym atutem okazał się dużym problemem. Dlatego ostatecznie wybrałem mniejszy, ale bardziej przyjazny zbiór MagnaTagATune w formacie mp3. Całość projektu jest zbudowana modularnie, dlatego późniejsze rozszerzenie je na inne datasety i modele nie powinno być większym problemem.