Music Collection Analyzer

Stefano Scola
UPF-MTG

Here is a link to my repository for this assignment.

1 Introduction

This report described the architecture and functionality of a system for extracting features from collection of tracks. In short it outlines the development of two user interfaces designed to facilitate navigation through the audio collection based on the extracted sonic features. Lastly, the quality of the latter will be briefly discussed.

2 Feature Generation

To generate the features the implementation follows 2 steps: an audio loader and a feature extractor.

2.1 Audio Loader

The implemented class named AudioFilesLoader is responsible for loading audio files from a specified directory. Upon initialization, it scans the provided directory for audio files with supported extensions by leveraging the functionality of the python *pathlib* module. The class provides a private method _load_audio to load individual audio files, converting them into stereo and mono formats, and resampling them to a sample rate of 16000 Hz. Additionally, it offers a generator function yield_all to iterate over all valid audio files in the directory, yielding tuples containing the loaded audio data. By yielding the data, the usage of ram is optimized. It should be noted that the extractors that will be introduced in the next sub-section are supposed to receive audio sampled at different rates. To this end, the AudioFilesLoader class has been implemented by allowing users to easily adjust that for future cases.

2.2 Feature Extractor

In the FeatureExtractor class, different algorithms are employed. Initially, musicnn and discogs-effnet models are utilized to generate embeddings for individual tracks. These embeddings are then used as input by various classifiers to predict voice/instrumental, arousal/valence, danceability, and genre-style activations. Additionally, digital signal processing (DSP) techniques are applied to extract features like beats per minute (bpm), loudness, and key information. The decision to not use a deep learning model to extract bpm was driven by the need of keeping the implementation lightweight.

3 Statistics

For simplicity and ease of use, a Python notebook has been employed for reporting statistics. Histograms and bar plots are utilized to illustrate the distribution of the data. When reporting genre activations from the Discogs Effnet model, plots are generated with information solely about the genre, while the complete data distribution is saved in a tab-separated values file. In general, it is evident that most of the exctracted descriptors are not normally distributed. For instance, both voice/instrumental and danceability display heavy-tailed distributions whereas genres are heavily skewed towards categories such as rock and electronic. On the other hand, the distribution of bpm and loudness seems to be normal. In fact, the distribution for loudness makes sense as most of the tracks have between -14 and -7 LUFS. Finally, about key detection, all the different three profiles detect that the majority of the tracks is in the key of C major.

4 Interface Building

4.1 Playlist App

The application interface has been structured using the sidebar of Streamlit, allowing users to select their preferred feature for playlist creation via a dropdown menu. Additionally, a plot displaying the distribution of the data before querying has been included to provide users with helpful insights. To save a playlist, a toggle button has been implemented along with a text box for entering the playlist name. Currently, users need to specify the saving location before playlist generation, but future enhancements could include a saving functionality after a playlist is generated.

4.2 Embeddings App

This simple Streamlit app was implemented to enables users to compare the similarity between songs using embeddings generated by the Discogs Effnet and MusicCNN models. It computes cosine similarity scores between the selected track and others, presenting in ascending order the most similar tracks from both embeddings. The embedding matrix of each track has been averaged, over all the frames, before the computation of cosine similarity.

5 Feature Quality Observations

Regarding the quality of the extracted features, certain descriptors such as voice/instrumental lack informativeness, particularly when averaged across frames. Arousal and valence present challenges in easy emotional mapping, while danceability appears culturally and subjectively contingent, with the model's tendency to associate fast music with danceability. On the other hand, BPM and genre-style activations emerge as reliable and informative features, whereas detected keys exhibit less reliability.

Finally, it's worth noting that Discogs embeddings excel in capturing music similarity between tracks compared to Musicnn. This disparity arises because Discogs-effnet is specifically crafted to capture style and genre activations, thereby directing towards tracks with high genre similarity. On the other hand, Musicnn, designed for music tagging purposes, identifies a distinct type of music similarity.