Automatic Music Clustering Based on Audio Similarity

Arseny Khrutsky

Introduction

In this project, the goal was to develop a system that automatically groups songs into playlists based on their audio similarity — without using metadata or human labels. Two different approaches were implemented:

- Version 1: Clustering using classical audio features (MFCCs, tempo)
- Version 2: Clustering using deep learning-based features (CNN14 embeddings)

Each version accepts a folder of audio files and the desired number of playlists (via --n), and outputs a .json file with clustered song IDs.

Version 1: Classical Features

Features Used:

- MFCCs (Mel-Frequency Cepstral Coefficients): capture the timbral characteristics of a song
- Tempo: estimated via beat tracking (librosa)

Each song is converted into a fixed-length feature vector by taking the mean of each MFCC dimension and appending the tempo.

Reasoning:

- MFCCs are widely used in music analysis and speech recognition for representing frequency content in a perceptually meaningful way.
- Tempo captures rhythmic differences that are important in distinguishing genres or moods.

Clustering:

- Feature vectors are grouped using KMeans clustering
- Songs are clustered purely based on sound patterns

Version 2: Deep Learning Embeddings

Features Used:

- CNN14 model from PANNs (Pretrained Audio Neural Networks)
- Extracts high-dimensional embeddings trained on the AudioSet dataset
- Embeddings represent complex semantic features like instruments, genre, texture

Reasoning:

- CNN14 is pretrained on over 2 million human-labeled audio clips
- Its embeddings are more expressive than handcrafted features, allowing for more nuanced similarity comparisons
- Eliminates the need for manual feature engineering

Model Integration:

- Model loaded from Cnn14_mAP=0.431.pth
- Audio files are resampled to 32 kHz and converted to mono if needed
- Embeddings are extracted from the final embedding layer
- These embeddings are then clustered using Kmeans

Research Process and Implementation Notes

- Version 1 was implemented using librosa, which allows for easy access to standard music features
- Version 2 required deeper exploration:
    - Identifying the correct layer for embeddings (output['embedding'])
    - Installing and using torchlibrosa

    - Making sure all audio inputs were standardized for the CNN (mono, 32kHz)

A flexible CLI was built to handle arguments like --path, --n, --output, --logfile, and logging levels (--debug, --silent).

Observations

- Version 1 tends to cluster based on simple sound features like timbre or tempo. It sometimes struggles with songs that share MFCC characteristics but differ stylistically.
- Version 2 performs significantly better in separating different "types" of tracks. Songs with vocals vs instrumentals, or electronic vs acoustic, tend to cluster more naturally.
- Clustering stability improved when preprocessing (resampling, mono conversion) was applied consistently across both versions.
- The CNN14 embeddings showed clear grouping behavior even with only 3 clusters, despite no genre or metadata input.

Conclusions

- Deep learning-based embeddings (Version 2) clearly outperform the classical approach in quality of clustering.
- However, Version 1 is lightweight, interpretable, and easier to deploy in resource-constrained environments.
- Both approaches are completely unsupervised and require no metadata — making them suitable for automating large music libraries.
- Future improvements could include:
    - Using t-SNE or UMAP for visualization
    - Adding silence trimming or loudness normalization
    - Exploring other models (e.g. CLAP, Wav2Vec, openL3)

Deliverables

- main_v1.py and main_v2.py scripts
- requirements.txt
- playlists_v1.json and playlists_v2.json
- This report