Audio Clustering for Automatic Playlist Generation

Technical Report

1. Introduction

This report describes the implementation of an automatic music clustering system that groups songs into playlists based on audio similarity. The system was built in two versions: 1. A classical approach using hand-crafted audio features 2. A deep learning approach using pre-trained audio models

Both implementations aim to create playlists that contain similar-sounding songs without relying on metadata, tags, or user history.

2. Approach and Methodology

2.1 Version 1: Classical Audio Feature Extraction

In the first version, we extracted traditional audio features using the librosa library:

- **Mel-frequency cepstral coefficients (MFCCs)**: The most widely-used features for audio analysis, capturing the short-term power spectrum of sound
- **Chroma features**: Represents the 12 different pitch classes, helping to identify harmonic and melodic characteristics
- **Spectral features**:
  - Spectral centroid: Indicates where the "center of mass" of the spectrum is, correlating with the brightness of a sound
  - Spectral rolloff: The frequency below which a specified percentage of the total spectral energy lies
- **Temporal features**:
  - Zero-crossing rate: Rate at which the signal changes from positive to negative or vice versa
  - Energy: Overall acoustic energy of the signal
- **Rhythm features**:
  - Tempo: Estimated beats per minute

After extraction, the features were: 1. Standardized using `StandardScaler` to normalize their ranges 2. Clustered with K-means algorithm to create the specified number of playlists

2.2 Version 2: Deep Learning-Based Approach

The second version leverages a pre-trained audio model:

- **Microsoft WavLM**: A self-supervised learning model trained on diverse audio datasets, which captures higher-level audio representations

WavLM was chosen because it: 1. Provides high-quality speech and audio representations 2. Was trained on a mixture of speech and non-speech audio 3. Has shown strong performance on audio classification tasks

The implementation: 1. Loads each audio file and resamples it to the model's required sampling rate 2. Converts stereo to mono if needed 3. Extracts embeddings from the model's hidden states 4. Uses mean pooling over the time dimension to get a fixed-size representation 5. Applies K-means clustering to group songs with similar embeddings

3. Research Process and Design Decisions

3.1 Feature Selection for Version 1

Several audio feature extraction techniques were considered:

Feature Type	Considered	Selected	Reasoning
MFCCs	Yes	Yes	Industry standard for timbral audio analysis
Chroma	Yes	Yes	Captures harmonic content regardless of pitch shifts
Spectral Contrast	Yes	No	Redundant with MFCCs and centroid features
Spectral Centroid	Yes	Yes	Good indicator of brightness/darkness of sound
Spectral Rolloff	Yes	Yes	Helps differentiate instrumental textures
Zero-crossing Rate	Yes	Yes	Useful for distinguishing voiced/unvoiced sounds
Tempo	Yes	Yes	Critical for style and genre differentiation

Feature Type	Considered	Selected	Reasoning
Onset Strength	Yes	No	Partially captured by tempo and energy features

The final feature set was chosen to balance: - Computational efficiency - Information relevance - Minimizing redundancy

3.2 Model Selection for Version 2

Several pre-trained audio models were evaluated:

Model	Considered	Selected	Reasoning
WavLM	Yes	Yes	Strong general audio representation capabilities
Wav2Vec2	Yes	No	More focused on speech than music
PANN	Yes	No	Excellent for audio tagging but more complex to implement
CLAP	Yes	No	Specific to contrastive learning between audio and text
HTSAT	Yes	No	Strong for audio event detection but overkill for this application

WavLM was selected because it: - Has been trained on a diverse range of audio data - Provides rich and general-purpose audio embeddings - Has a straightforward implementation via the Hugging Face transformers library

3.3 Clustering Algorithm Selection

Several clustering algorithms were evaluated:

Algorithm	Considered	Selected	Reasoning
K-means	Yes	Yes	Simple, efficient, and effective for this application
DBSCAN	Yes	No	Doesn't guarantee a specific number of clusters
Hierarchical	Yes	No	Computationally expensive for large datasets
Gaussian Mixture	Yes	No	More complex than needed for this application
Spectral	Yes	No	Performance issues with large datasets

K-means was chosen because: - It aligns with the requirement to generate exactly N playlists - It works well with both high and low-dimensional feature spaces - It's computationally efficient and scalable

4. Observations and Conclusions

4.1 Comparison of Both Approaches

Aspect	Version 1 (Classical)	Version 2 (Deep Learning)
Feature extraction time	Faster	Slower
Computational resources	Lower	Higher
Feature dimensionality	Lower	Higher
Capture of subtle patterns	Limited	Superior
Handling of diverse audio types	Limited	More robust
Explainability	Higher	Lower

4.2 Quality of Generated Playlists

When testing with a diverse set of songs, we observed:

1. **Version 1 (Classical Features)**:
   - Effectively grouped songs by broad characteristics like tempo and energy
   - Sometimes mixed different genres with similar structural characteristics
   - Struggled with complex audio textures and production styles

2. **Version 2 (Deep Learning)**:
    - Created more coherent playlists with songs that "sound similar"
    - Better captured subtle production qualities and timbral characteristics
    - Successfully grouped songs with similar moods despite different tempos

4.3 Limitations and Future Work

Current limitations: - No consideration of song lyrics or semantic content - Fixed-length representations may not capture temporal dynamics effectively - K-means assumes spherical clusters which may not match perceptual audio similarity

5. Summary

This project demonstrates two approaches to automatic playlist generation based on audio similarity. The classical approach offers efficiency and interpretability, while the deep learning approach captures more nuanced audio characteristics at the cost of higher computational requirements.

The deep learning approach (Version 2) generally produced more perceptually coherent playlists, suggesting that learned representations capture more of what makes songs "sound similar" to human listeners than hand-crafted features alone.

These methods provide a solid foundation for automated music organization tools that could eventually replace manual playlist creation, offering music listeners a more efficient way to organize and discover music based on how it actually sounds.