Harmony Blend

Submitted for

Statistical Machine Learning CSET211

Submitted by:

(E23CSEU0268) AARUSHI ARORA

(E23CSEU0245) GYAN DOGRA

(E23CSEU0259) TVISHI VASUDEVA

Submitted to

Mr. Tapas Badal

July-Dec 2024

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

INDEX

Abstract

This project presents a playlist-based song recommendation system that utilises Spotify's API to extract and analyse audio features from user-defined playlists. The system leverages content-based filtering, incorporating features such as danceability, energy, loudness, tempo, and instrumentalness, along with metadata like artist names and genres. By calculating the Euclidean distance between songs based on their feature vectors, the system identifies tracks that are most similar to those in the user's playlist. This approach provides tailored song recommendations, enhancing the user's music discovery experience. The project is implemented using Python, Spotipy, and Pandas, and demonstrates effective use of machine learning concepts in creating personalised recommendations.

Introduction

Music recommendation systems play a crucial role in enhancing the listening experience by suggesting songs tailored to user preferences. This project focuses on building a playlist-based song recommendation system that utilises Spotify's API to analyse audio features and metadata from user playlists. By understanding the characteristics of the songs, such as danceability, energy, tempo, and instrumentalness, along with metadata like artist names and genres, the system identifies patterns and similarities within the user's preferences.

The recommendation engine employs Euclidean distance to measure the similarity between songs based on their feature vectors, enabling precise content-based filtering. This approach ensures recommendations align with the mood, style, and energy of the user's playlist.

The project leverages Python libraries such as Pandas, Spotipy, and NumPy for data handling and analysis, and Streamlit for building an interactive user interface. The outcome is a personalised music recommendation system that combines data-driven techniques with real-world applications, demonstrating the practical integration of machine learning concepts into everyday experiences.

Methodology

The project follows a structured approach to build a playlist-based song recommendation system. The key steps in the methodology are as follows:

1. Data Collection

- Spotify's Web API, accessed using the Spotipy library, is used to retrieve information about songs from a user-provided playlist.
- For each track, audio features such as **danceability**, **energy**, **tempo**, **loudness**, and **valence**, along with metadata like **artist name** and **genres**, are extracted.

2. Data Preprocessing

- The raw data retrieved from the API is cleaned and structured into a Pandas DataFrame.
- Missing or inconsistent data points are handled appropriately to ensure smooth analysis.
- Numerical features are **normalized** to scale them between 0 and 1, ensuring fair comparison between features with different ranges.

3. Feature Engineering

- The audio features serve as a numerical representation of each song, creating feature vectors for comparison.
- Metadata such as artist and genre is incorporated to enhance the recommendation system's relevance.

4. Similarity Calculation

- **Euclidean distance** is used as the similarity metric. The distance between feature vectors of songs in the user playlist and other available songs is computed.
- Songs with the smallest Euclidean distances to the user playlist tracks are considered the most similar and are selected as recommendations.

5. Recommendation Generation

- A list of recommended songs is generated based on the calculated similarities.
- The results are presented to the user, including song names, artists, and associated genres.

6. Implementation and User Interface

- The recommendation system is implemented using Python.
- A user-friendly interface is built with Streamlit, allowing users to input their playlist IDs and view personalized recommendations.

7. Evaluation and Testing

- The system's recommendations are manually evaluated for quality and relevance.
- Further iterations of the system are conducted to refine and optimize its performance.

This methodology ensures a robust, scalable, and efficient song recommendation system that leverages Spotify's comprehensive dataset and content-based filtering techniques.

Hardware Requirements

1. **Computer System**: A laptop or desktop with the following minimum specifications:
   - Processor: Intel Core i5 or equivalent
   - RAM: 8 GB or higher (for smooth execution of the program)
   - Storage: At least 500 MB of free space for data storage and library installations
   - Internet Connection: Required for accessing Spotify's API

Software Requirements

1. **Operating System**:
   - Compatible with Windows, macOS, or Linux
2. **Python Environment**:
   - **Python Version**: 3.8 or higher
   - IDE/Code Editor:
     - Anaconda with Jupyter Notebook (preferred for development and visualization)
     - Alternatively, IDLE or any other Python-supported IDE

1. **Python Libraries and Packages**:
   - **Spotipy**: For accessing Spotify's Web API to fetch song details and audio features.
   - **Pandas**: For handling and processing tabular data.
   - **NumPy**: For numerical computations and normalization of data.
   - **Streamlit**: For building an interactive user interface.
2. **Spotify Developer Account**:
   - Necessary for obtaining API credentials (client ID and client secret).
3. **Browser**:
   - A web browser (e.g., Chrome, Firefox) for accessing the Spotify Developer Dashboard and Streamlit app interface.
4. **Other Tools**:
   - **GitHub**: For hosting and sharing the project code and related files.
   - **Text Editor**: (Optional) Sublime Text, VS Code, or similar for editing files.

Experimental Results

Model 1: Cosine Similarity

- **Approach**:
  This model used **cosine similarity** to measure the similarity between user-provided playlist songs and potential recommendations. The similarity was calculated based on normalized audio features such as danceability, energy, loudness, etc.
- **Findings**:
  - Recommendations showed high relevance in terms of matching audio features.
  - Songs with similar moods and styles were frequently recommended.
  - However, genre and artist preferences were not directly incorporated, which led to occasional irrelevant artist or genre matches.
  - This approach excelled in finding feature-alike songs but lacked contextual alignment like artist or genre.

Model 2: Euclidean Distance

- **Approach**:
  This model used **Euclidean distance** to calculate the similarity between songs based on normalized audio features. Songs with the smallest distances were recommended as they were considered most similar to the user's playlist.
- **Findings**:
  - Recommendations were diverse yet somewhat relevant to the playlist's audio profile.
  - Compared to cosine similarity, this approach struggled slightly with high-dimensional feature comparisons.
  - Although it considered proximity in feature space, the absence of genre and artist information made some recommendations contextually less appealing.

Model 3: Euclidean Distance with Genre and Artist Filtering

- **Approach**:
  This model combined **Euclidean distance** for audio feature similarity with **genre and artist-based filtering** to prioritize songs by artists and genres in the user's playlist.
- **Findings**:
  - This model outperformed the previous two in terms of both relevance and diversity.
  - By incorporating genres and artists as filters, the recommendations aligned better with the user's preferences.
  - Songs from similar genres or by related artists were often prioritized, creating a cohesive listening experience.
  - The inclusion of contextual metadata (genre and artist) improved the overall user satisfaction with the recommendations.

Conclusion

- **Cosine Similarity** performed well for audio feature-based recommendations but lacked context.
- **Euclidean Distance** was simple but less effective in handling complex, high-dimensional data.
- **Euclidean Distance with Genre and Artist Filtering** delivered the most relevant and user-aligned recommendations by leveraging both numerical features and metadata.
- Therefore, **Model 3** is the most balanced and effective solution for the playlist-based song recommendation system.

Conclusions

The project aimed to develop a playlist-based music recommendation system utilizing Spotify's audio features, genres, and artist metadata. Multiple models were implemented and compared to evaluate their effectiveness in delivering personalized song recommendations.

1. **Key Findings**:
   - **Cosine Similarity**: Efficient in identifying songs with similar audio profiles but lacked contextual filtering through artist and genre preferences.
   - **Euclidean Distance**: Provided an intuitive measure of similarity but performed better when combined with genre and artist filtering.
   - **Euclidean Distance with Metadata**: The inclusion of artist and genre improved the relevance and accuracy of recommendations, aligning with both numerical and contextual preferences.
2. **Significance of Audio Features and Metadata**:
   By integrating features like danceability, tempo, energy, and valence with artist and genre metadata, the models ensured that recommendations captured both the "feel" and the context of user preferences.

3. **Impact and Applications**:
   The models have potential applications in music streaming services, personalized playlists, and audio-based AI systems, where user-centric recommendations are key.

Future Scope

Incorporation of User Feedback:

Allow users to rate recommendations or provide explicit feedback (e.g., thumbs up/down).
Use this feedback to iteratively refine and improve the recommendation algorithm through reinforcement learning or adaptive models.

Dynamic Feature Weighting:

Implement methods to dynamically adjust the importance of audio features (e.g., danceability, tempo) based on user preferences or playlist themes.

Expanding Metadata:

Include additional metadata such as song lyrics, release year, popularity metrics, or album details to enrich the recommendation context.
Explore sentiment analysis of lyrics to align song recommendations with specific moods or themes.

Hybrid Recommendation Models:

Combine collaborative filtering (based on user behavior) with content-based filtering for a more holistic recommendation approach.
Leverage deep learning techniques, such as autoencoders, to capture complex patterns in song features and user preferences.

Scalability for Large Datasets:

Optimize the models to handle large-scale datasets for a broader range of recommendations.
Utilize distributed computing or cloud-based solutions for efficient data processing.

Real-Time Recommendations:

Develop real-time recommendation capabilities for applications like live playlist generation during events or mood-based curation.

Enhanced User Interface:

Create an intuitive and interactive user interface with advanced filtering options (e.g., by genre, artist, or mood).
Include visualizations for song recommendations, showing how they match user preferences.

Integration with Other Platforms:

Expand the project to include integration with other music services like YouTube Music or Apple Music.
Enable cross-platform playlist recommendations for a seamless user experience.

Advanced Audio Analysis:

Use deep learning to analyze raw audio data for uncovering hidden features that traditional analysis may miss.
Include audio segmentation for identifying specific sections (e.g., chorus, verses) that influence user preference.

Diversity and Novelty:

Incorporate mechanisms to ensure diversity in recommendations to prevent repetitive suggestions.
Implement a novelty factor to introduce users to new and unfamiliar music while maintaining relevance.

GitHub Link of Complete Project

https://github.com/aarushiarora18/Song-Recommender