Project Report: Spotify Playlist Extender

Team Members

• Yashvardhan – Registration No: 229310377
• Tanishq Patil – Registration No: 229311050
• Sairaj Sirsat – Registration No: 229311122

1. Objective

The main objective of the Spotify Playlist Extender project is to combine the analytical capabilities of data science with the personalization features of modern music platforms. The project aims to enhance the user’s Spotify playlist experience by intelligently analyzing the lyrical and emotional essence of existing songs and extending playlists with new, thematically consistent tracks. This system merges Spotify’s extensive music database with Genius’s lyrical information to provide a more personalized, emotionally driven recommendation experience. By doing so, the system ensures that users can discover new music that truly matches the tone, themes, and sentiment of the playlists they love.

2. Problem Statement

Although Spotify’s inbuilt recommendation engine is powerful, it mainly relies on metadata such as user history, tempo, genre, and collaborative filtering. It lacks a deep understanding of the lyrical and emotional context of songs, which many users use as a basis for curation. Music is an emotional experience, and playlists are often centered around specific moods such as heartbreak, nostalgia, love, or hope. However, current recommendation systems fail to analyze lyrical data to match these sentiments. Hence, the challenge is to build a system that can analyze the emotional and lyrical content of an existing playlist and intelligently recommend similar tracks, thereby providing a more holistic and emotionally resonant listening experience.

3. Methodology

3.1 Tools and APIs Used

• **Spotify API (via Spotipy library):** Provides access to user playlists, tracks, and metadata while allowing modifications like adding new tracks.
• **Genius API:** Enables lyrical data search and retrieval to analyze songs on a textual and emotional basis.
• **BeautifulSoup:** Used for parsing and scraping lyrics from Genius web pages when direct API access is insufficient.
• **Python Libraries:** requests, re, json, collections.Counter, and time were used for data handling, parsing, and rate-limiting. These libraries collectively manage API communication, data extraction, and error-handling mechanisms.

3.2 Workflow

1. **Authentication:** The system begins by authenticating the user via Spotify OAuth, granting permissions to access and modify playlists securely.
2. **Playlist Extraction:** The user's selected playlist is fetched, and its metadata—including song names, artists, and popularity scores—is stored.
3. **Lyrics Collection:** For each track, the system queries the Genius API using the artist's name and song title, then scrapes the corresponding webpage for lyrics.
4. **Sentiment Analysis:** Using a predefined dictionary-based approach, lyrics are analyzed to determine their dominant sentiment (positive, negative, or neutral).
5. **Theme Extraction:** Keywords are matched against predefined theme sets (romance, heartbreak, hope, nostalgia, longing) to identify recurring motifs.
6. **Insight Generation:** The application aggregates metrics such as top recurring artists, lyrical sentiment distribution, and dominant themes within the playlist.
7. **Recommendation Phase:** The Genius API is then used again to find songs with similar themes or from related artists. These recommendations are cross-referenced with Spotify's search API to retrieve track URIs.
8. **Playlist Update:** The application finally adds the most relevant tracks directly into the user's Spotify playlist, effectively 'extending' it.

4. Features

- Emotion-aware music recommendation using lyric sentiment analysis.
- Smart playlist expansion based on lyrical similarity and artist correlation.
- Automated Spotify playlist modification with direct track insertion.
- Built-in error-handling for failed API calls and connection issues.
- Console-based feedback with detailed progress and insights.
- Modular, object-oriented Python design that allows easy scalability and maintenance.

5. Results

Upon execution, the system successfully authenticates users and analyzes playlists using Spotify and Genius data. It identifies emotional and thematic trends within playlists and generates curated song suggestions. The sentiment analysis effectively distinguishes between upbeat, melancholic, and neutral songs, while theme extraction provides additional context. The Genius-powered search and Spotify integration ensure that all recommended tracks are available for immediate playback and addition. Users receive an expanded playlist that feels consistent and emotionally unified, improving overall satisfaction with automated curation.

Through testing, the application was able to analyze playlists with varying genres, accurately detect their mood tendencies, and propose relevant new tracks. The automated addition of these tracks to the user's Spotify account validates the system's complete functional loop—from analysis to action.

6. Conclusion

The Spotify Playlist Extender project successfully demonstrates how natural language analysis and API integration can be merged to produce emotionally intelligent recommendations. Unlike standard algorithmic approaches, this system goes beyond acoustic features and incorporates lyrical depth to enhance personalization. It bridges the gap between technical precision and human emotion in digital music recommendation systems. Future work could involve integrating NLP models such as BERT or spaCy for advanced lyrical sentiment detection, implementing GUI or web-based user interfaces, and incorporating audio features like tempo, key, and danceability for even more holistic recommendations.

7. References

- Spotify Web API Documentation – https://developer.spotify.com/documentation/web-api
- Genius API Documentation – https://docs.genius.com
- Spotipy Python Library – https://spotipy.readthedocs.io
- BeautifulSoup Documentation – https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Python Official Documentation – https://docs.python.org/3/