Song Recommender System

Introduction

This project is focused on developing a song recommender system using content-based filtering. The main objectives were to collect and process song metadata, audio features, and lyrics, and to create a recommendation algorithm based on these processed features.

Approach

Data Collection

1. **Scraping Spotify:**
    - To create a robust recommender system, it was essential to collect detailed audio features and metadata from Spotify. This process involved leveraging Spotify's API to retrieve relevant data points.
    - Initial scraping for a small subset of songs was straightforward.
    - When the dataset expanded to 25k songs, scraping features and metadata became challenging.
    - **Solution:** The dataset was broken into smaller parts and distributed among team members for parallel processing.
    - **Features Scraped from Spotify:**
        - **Track Popularity:** A measure of how popular a track is.
        - **Explicit:** Whether the track contains explicit content.
        - **Album Type:** The type of album (e.g., single, album, compilation).
        - **Danceability:** How suitable a track is for dancing based on tempo, rhythm stability, beat strength, and overall regularity.
        - **Energy:** A measure of intensity and activity.
        - **Key:** The key in which the track is composed.
        - **Loudness:** The overall loudness of a track in decibels (dB).
        - **Mode:** Indicates the modality (major or minor) of a track.
        - **Speechiness:** The presence of spoken words in a track.
        - **Acousticness:** A measure of how acoustic a track is.
        - **Instrumentalness:** The likelihood of a track containing no vocals.
        - **Liveness:** The presence of a live audience in the recording.
        - **Valence:** The musical positiveness conveyed by a track.
        - **Tempo:** The speed or pace of a track.
2. **Scraping Lyrics:**
    - Initially, we attempted to scrape lyrics using AZLyrics, Gaana, and JioSaavn. This process required extensive trial and error due to various site structures and anti-scraping measures. Despite the challenges, we managed to develop a working solution.

    - However, upon receiving the diverse dataset, it became clear that our initial approach was insufficient. The diversity and volume of songs necessitated a more scalable and reliable method for scraping lyrics.
    - **New Approach:** We pivoted to using the Genius API for its comprehensive lyrics database. For songs not found on Genius, we implemented a fallback mechanism to scrape from Musixmatch. This two-tiered approach significantly improved our coverage and reliability.
3. **Attempt to Scrape Audio Features:**
    - Recognizing the importance of audio features for the recommender system, we made a concerted effort to scrape these features.
    - Spent three whole days attempting to scrape audio features, but faced multiple issues such as persistent errors and excessive processing time per audio file.
    - Notebook has been attached for the same as well
    - **Future Plans:** Although these attempts were initially unsuccessful, we plan to revisit and resolve these issues to incorporate audio features into the system.

Data Processing

1. **Lyrics Processing:**
    - **Initial Cleaning:**
        - **Remove Punctuation:** All punctuation marks were removed to ensure the text was clean and standardized.
        - **Convert to Lowercase:** All text was converted to lowercase to maintain uniformity and avoid duplication of words with different cases.
        - **Remove Stop Words:** Commonly used words that do not contribute to the overall meaning (stop words) were removed to focus on the significant words.
        - **Lemmatization:** Words were lemmatized, meaning they were reduced to their base or root form (e.g., "running" becomes "run"), to treat different forms of a word as a single item.
    - **Language Detection:**
        - Realising the importance of recommending songs in the same or similar languages, we extracted the language of each song from its lyrics using the `langdetect` Python library. This allowed us to ensure language-specific recommendations.
    - **Generating Embeddings:**
        - To generate embeddings, we considered various models and ultimately settled on the Huggingface BERT multilingual uncased model.
        - **BERT Multilingual Uncased Model:** This model captures the contextual meaning of words in multiple languages without case sensitivity. It provides a dense vector representation of the lyrics that captures semantic relationships but does not focus specifically on sentiments.
        - These embeddings are crucial for understanding the nuanced meanings in lyrics and for comparing songs based on their lyrical content.

- **Sentiment Analysis:**
    - To capture the emotional tone of the lyrics, we used the `emotion-english-distilroberta-base` model for sentiment analysis.
    - **Model:** `emotion-english-distilroberta-base` is a transformer-based model that identifies and categorizes sentiments into different emotions such as joy, sadness, anger, fear, surprise, and disgust.
    - This additional layer of sentiment analysis provided another dimension for understanding and recommending songs based on their emotional content.
- **Topic Modeling:**
    - We initially explored topic modeling to identify common themes in the lyrics using models such as Latent Dirichlet Allocation (LDA).
    - However, we decided to drop topic modeling because it did not significantly improve the recommendation quality and added unnecessary complexity to the processing pipeline.

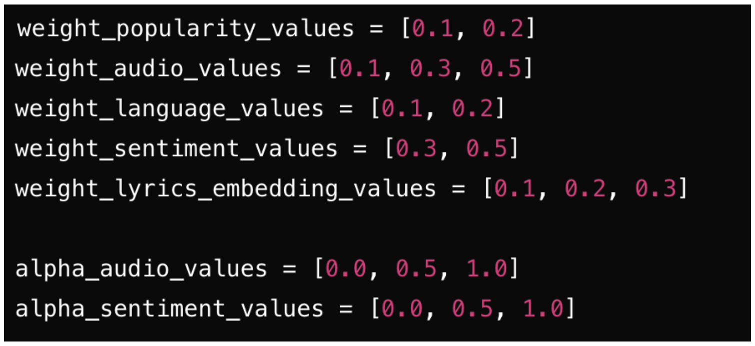
2. **Metadata Processing:**
    - After an in-depth analysis of the metadata features, we decided to drop certain features based on their low correlation with other features and the overall recommendation objective:
        - **Tempo and Mode:** These features were found to have low correlation with the other features in our dataset, as shown in the attached correlation matrix.
        - **Explicit and Album Type:** These features were deemed less relevant for the recommendation purposes. The explicit nature of a song and its album type (e.g., single, album, compilation) did not significantly influence the recommendations in a meaningful way.

Similarity Functions

1. **Standard Scaling:**
    - We applied standard scaling to the entire dataset. This step was essential to normalize the features so that they have a mean of 0 and a standard deviation of 1. Standard scaling ensures that all features contribute equally to the similarity calculations, preventing features with larger ranges from dominating the results.
2. **Feature Grouping:**
    - We grouped the features into five categories: popularity, audio, language, sentiment, and lyrics embedding. This grouping was necessary because these features represent different types of vectors and have varying value ranges. By categorizing them, we could apply appropriate similarity functions tailored to each group's characteristics.

3. **Similarity Functions:**
    - Different similarity functions were applied to each category, and a weighted average was taken.
    -

Feature Category	Similarity Function
Popularity	Custom similarity using absolute distance, aimed at recommending songs with similar popularity levels. Other similarity measures like cosine similarity were rejected because they are not suitable for ordinal data like popularity.
Audio	Custom function combining cosine similarity and distance, capturing both aspects for better accuracy. This approach balances the importance of both magnitude and direction in the audio features. Other measures like Euclidean distance alone were rejected because they did not account for direction.
Language	Binary similarity (1 if the same language, 0 if different), to ensure language-specific recommendations. Other similarity measures were rejected because language similarity is inherently binary.
Lyrics Embedding	Cosine similarity for high-dimensional vector, ideal for capturing the contextual meaning in lyrics embeddings. Other measures like Euclidean distance were rejected because they are not as effective in high-dimensional spaces
Sentiment	Custom function combining cosine similarity and distance, capturing both aspects for better accuracy. This approach allows us to consider both the direction and magnitude of sentiment. Other measures like simple difference were rejected because they did not capture the nuance of sentiment variations.

4. **Hyperparameter Tuning:**
    - To optimise the recommendation system, we defined the following hyperparameter ranges for the weights of different feature categories and the alpha values for custom similarity functions:
    - 

```
weight_popularity_values = [0.1, 0.2]
weight_audio_values = [0.1, 0.3, 0.5]
weight_language_values = [0.1, 0.2]
weight_sentiment_values = [0.3, 0.5]
weight_lyrics_embedding_values = [0.1, 0.2, 0.3]

alpha_audio_values = [0.0, 0.5, 1.0]
alpha_sentiment_values = [0.0, 0.5, 1.0]
```

    - Using these ranges, we performed hyperparameter tuning by iterating over all possible combinations.

After extensive trial and error, the best combination of hyperparameters we found was:

- `weight_popularity = 0.1`
- `weight_audio = 0.1`
- `weight_language = 0.2`
- `weight_sentiment = 0.5`
- `weight_lyrics_embedding = 0.1`
- `alpha_audio = 1.0`
- `alpha_sentiment = 0.0`

From this, we inferred that the sentiment value should be significantly higher to achieve good results, and the best alpha values for the custom similarity functions were 1 for audio and 0 for sentiment. This combination effectively balanced the different aspects of the audio and sentiment similarity, leading to better overall recommendations.

Subsequently, we continued our experimentation and discovered an even more effective combination:

- `weight_popularity = 0.025`
- `weight_audio = 0.005`
- `weight_language = 0.015`
- `weight_sentiment = 0.95`
- `weight_lyrics_embedding = 0.005`
- `alpha_audio = 1`
- `alpha_sentiment = 0`

This final combination provided the best results, confirming that giving a higher weight to sentiment significantly improved the recommendation quality. The values of alpha further solidified the balance between the different aspects of audio and sentiment similarity, leading to superior overall performance.

.

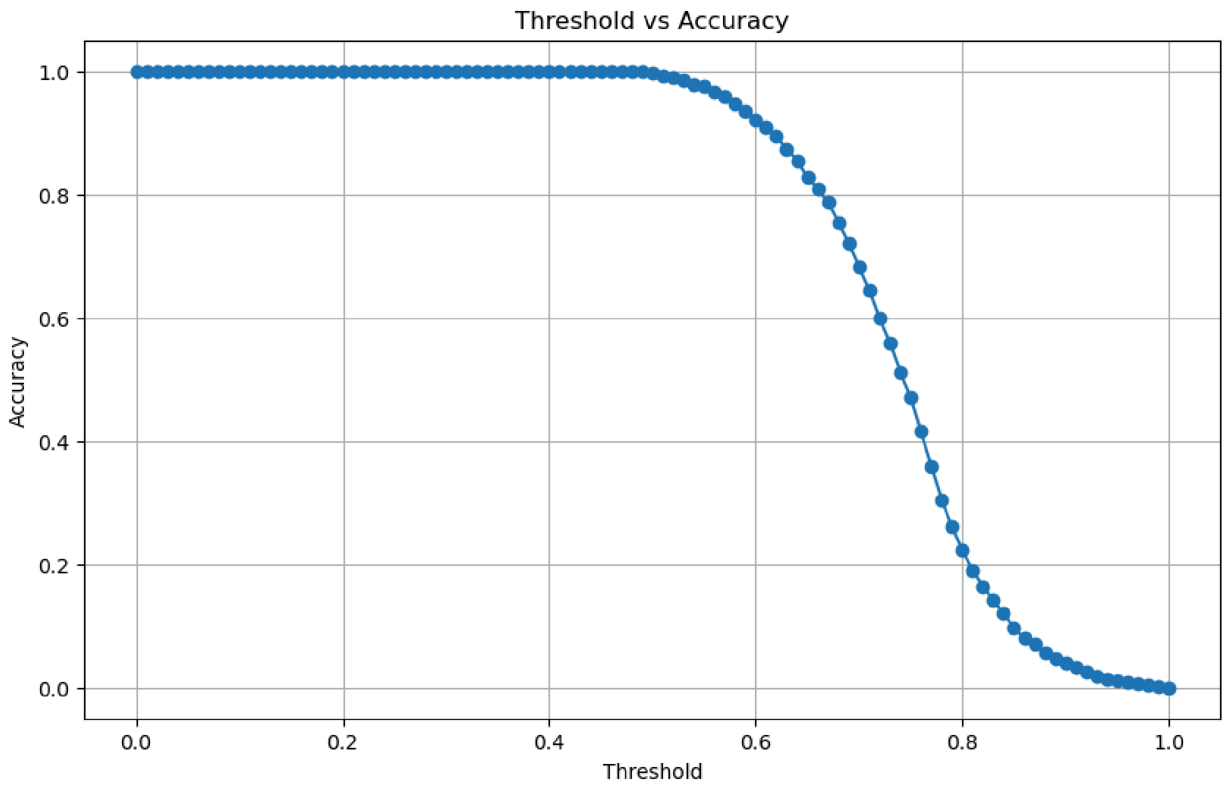
Blockers

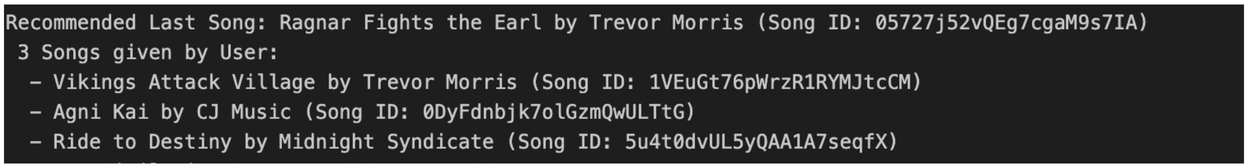
- **Scraping Spotify Data:** Managing a large dataset (25k songs) was challenging, but breaking it into smaller parts and dividing it among team members helped.
- **Scraping Lyrics:** Initial methods were insufficient due to the diversity of the songs. Switching to a combination of Genius API and Musixmatch improved coverage.

- **Scraping Audio Features:** Spent three whole days attempting to scrape audio features, but faced issues such as errors and excessive time per audio file. This issue might be resolved in the future.
- **Feature Correlation Confusion:** During the analysis phase, we faced confusion and spent a significant amount of time calculating the correlation of each feature with one another. This was an attempt to identify which features were most relevant for our recommendation model, but it led to some delays and required further clarification of our approach.

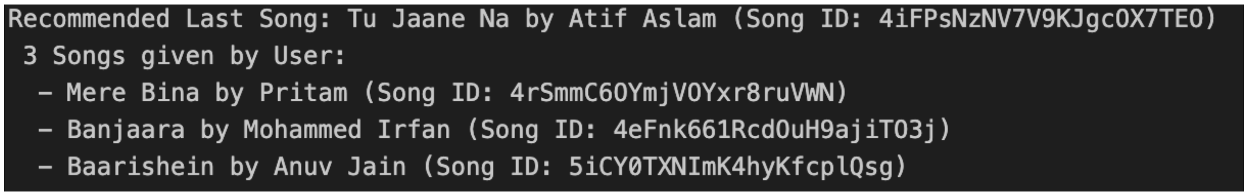
Results

Combination	Weight Popularity	Weight Audio	Weight Language	Weight Sentiment	Weight Lyrics Embedding	Alpha Audio	Alpha Sentiment	Final Avg Similarity Score	Final Overall Accuracy (in %)
1	0.1	0.1	0.2	0.5	0.1	1	0	0.577	24.2
2	0.1	0.1	0.1	0.5	0.2	0	0.5	0.442	11.6
3	0.2	0.1	0.1	0.5	0.1	0.5	0	0.575	18.7
4	0.1	0.1	0.2	0.5	0.1	0.5	0	0.559	21.9
5	0.1	0.3	0.2	0.3	0.1	1	1	0.377	8.1
6	0.025	0.005	0.015	0.95	0.005	0.8	0.2	0.694	50.3
7	0.025	0.005	0.015	0.95	0.005	1	0	0.737	68.2

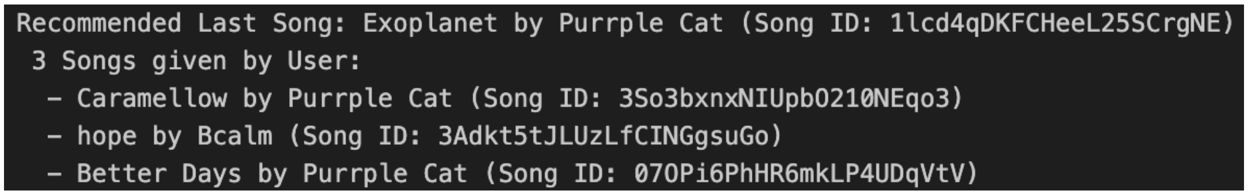




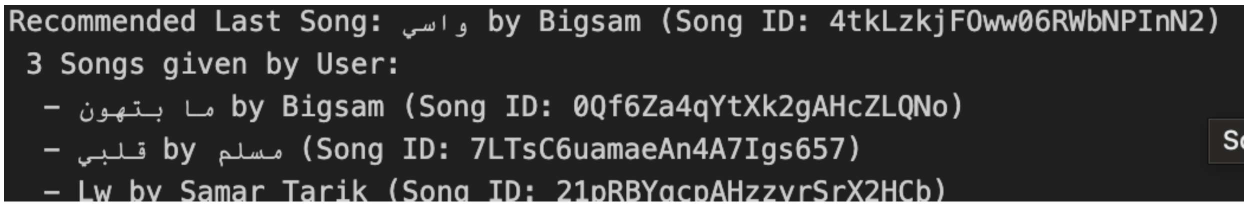
```
Recommended Last Song: Ragnar Fights the Earl by Trevor Morris (Song ID: 05727j52vQEg7cgaM9s7IA)
 3 Songs given by User:
  - Vikings Attack Village by Trevor Morris (Song ID: 1VEuGt76pWrzR1RYMJtcCM)
  - Agni Kai by CJ Music (Song ID: 0DyFdnbjk7olGzmQwULTtG)
  - Ride to Destiny by Midnight Syndicate (Song ID: 5u4t0dvUL5yQAA1A7seqfX)
```



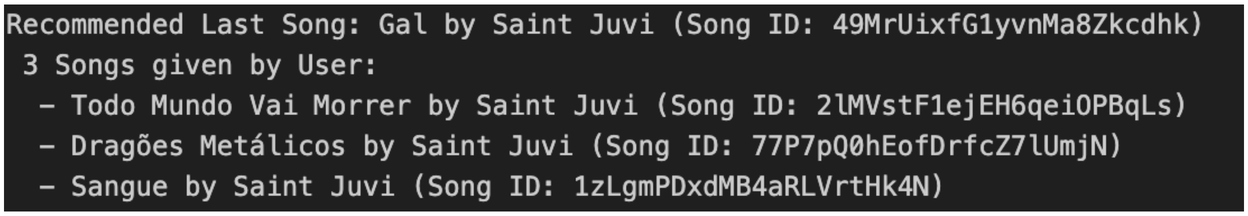
```
Recommended Last Song: Tu Jaane Na by Atif Aslam (Song ID: 4iFPsNzNV7V9KJgcOX7TEO)
 3 Songs given by User:
  - Mere Bina by Pritam (Song ID: 4rSmmC6OYmjVOYxr8ruVWN)
  - Banjaara by Mohammed Irfan (Song ID: 4eFnk661RcdOuH9ajiTO3j)
  - Baarishein by Anuv Jain (Song ID: 5iCY0TXNImK4hyKfcplQsg)
```



```
Recommended Last Song: Exoplanet by Purrple Cat (Song ID: 1lcd4qDKFCHeeL25SCrgNE)
 3 Songs given by User:
  - Caramellow by Purrple Cat (Song ID: 3So3bxnxNIUpbO21ONEqo3)
  - hope by Bcalm (Song ID: 3Adkt5tJLUzLfCINGgsuGo)
  - Better Days by Purrple Cat (Song ID: 07OPi6PhHR6mkLP4UDqVtV)
```



```
Recommended Last Song: واسي by Bigsam (Song ID: 4tkLzkjFOww06RWbNPInN2)
 3 Songs given by User:
  - ما بتهون by Bigsam (Song ID: 0Qf6Za4qYtXk2gAHcZLQNo)
  - قلبي by مسلم (Song ID: 7LTsC6uamaeAn4A7Igs657)
  - Lw by Samar Tarik (Song ID: 21pRBYgcpAHzzvrSrX2HCb)
```



```
Recommended Last Song: Gal by Saint Juvi (Song ID: 49MrUixfG1yvnMa8Zkcdhk)
 3 Songs given by User:
  - Todo Mundo Vai Morrer by Saint Juvi (Song ID: 2lMVstF1ejEH6qeiOPBqLs)
  - Dragões Metálicos by Saint Juvi (Song ID: 77P7pQ0hEofDrfcZ7lUmjN)
  - Sangue by Saint Juvi (Song ID: 1zLgmPDxdMB4aRLVrtHk4N)
```

Future Prospects

In future iterations of the project, there are several areas we plan to explore to further enhance the recommender system:

- **Incorporating Audio Features:** We aim to resolve the issues encountered during the scraping of audio features and successfully integrate these features into our recommendation algorithm. This would enable the system to consider the audio characteristics of songs, potentially improving the accuracy of recommendations.
- **Including Genre Information:** We also plan to include genre information for each song. By incorporating genre data, the recommender system can better understand the musical preferences of users and provide more targeted recommendations based on genre similarities.