Song Recommendation and Filter System using Spotify Million Playlist Dataset

1st Ankita Anand
210150006
Data Science and Artificial Intelligence
Guwahati, India
ankita.anand@iitg.ac.in

***Abstract*—This paper introduces a sophisticated Song Recommendation System designed to offer highly personalized music suggestions to users based on their preferences. Leveraging the extensive Spotify Million Playlist Dataset, the system employs content-based filtering, Big Data techniques, and innovative features like emotion-based filters. It explores recommendation methodologies, including genre-based, artist-based, and mood-based approaches, addressing challenges like the cold start problem and scalability issues through advanced technologies such as Apache Hadoop, Spark and MongoDB. The paper delves into the theoretical and practical aspects of integrating emotion-based filters, allowing users to tailor recommendations according to specific moods like happy, sad, or acoustic. The modeling process involves TF-IDF vectorization, cosine similarity calculations, and MongoDB for efficient dataset management. The paper introduces a Spotify clone website, providing users with a customizable platform to input preferences such as favorite artists, genres, and mood.**

***Index Terms*—Song recommendation, content-based filtering, collaborative filtering, Big Data, NLP, deep learning, ensemble methods, MongoDB, TF IDF, Cosine Similarity**

I. Introduction and Motivation

In the realm of music streaming services, the quest for enhancing user experience and personalization has become increasingly paramount. This project embarks on the ambitious journey of developing a state-of-the-art Song Recommendation System, driven by the unwavering motivation to deliver music suggestions that are not just tailored but deeply resonate with individual user preferences.

At its core, the system aims to revolutionize the way users discover and engage with music by offering highly personalized recommendations. Unlike generic approaches, our system is set to harness the vast reservoir of musical data encapsulated in the Spotify Million Playlist Dataset. This dataset stands as a testament to the diversity and richness of musical content, providing a robust foundation for our recommendation system.

The methodology employed in this project is multifaceted, with a key focus on content-based filtering. By meticulously analyzing song features and metadata, the system aims to decipher the intricate nuances that define a user's musical taste. This approach ensures that recommendations go beyond generic popularity metrics, delving into the specifics of genres, release years, and other contextual attributes that contribute to a user's unique musical fingerprint.

The utilization of content-based filtering signifies a departure from conventional recommendation systems. Instead of solely relying on collaborative filtering techniques, our approach empowers the system to understand the inherent characteristics of each song, enabling a more profound and accurate alignment with user preferences.
Motivated by the ever-growing demand for more individualized and contextual recommendations, the project adopts a multifaceted strategy. Employing content-based filtering as its melodic core, the system analyzes intricate song features and metadata to understand the nuances of user preferences. Furthermore, the introduction of emotion-based filters, tuned to human sentiments such as happiness, sadness, and acoustic preferences, enriches the recommendation palette, allowing users to curate their musical experience with unparalleled granularity.
The motivation behind this project extends beyond the creation of a mere recommendation system. It aspires to construct a musical universe where users become the conductors, guiding the system through their unique soundscapes. By exploring genres, artists, and moods with an amalgamation of advanced methodologies like TF-IDF, cosine similarity, and emotion-based filters, this project envisions a revolution in personalized music recommendations, where every note resonates with the user's soul.
In the subsequent sections, we will delve into the intricacies of our methodology, exploring how the system incorporates user-defined parameters, delves into Big Data algorithms such as collaborative filtering, and utilizes technologies like MongoDB for efficient dataset management. The project's motivation is not merely to recommend songs but to craft a music discovery experience that is unparalleled in its personalization and precision.

II. Problem Statement

Research objectives include designing a scalable and personalized music recommendation system, effectively addressing the cold start problem, and utilizing Big Data technologies for efficient data processing. The proposed methodologies encompass exploring hybrid recommendation techniques, investigating meta-learning approaches, evaluating distributed processing frameworks, and researching scalable storage solutions.

A. Taking user's playlist as input and recommending songs based on it.

The primary objective of this project is to create a sophisticated Song Recommendation System that goes beyond conventional approaches by harnessing the specific preferences expressed in a user's playlist. The system takes a user's playlist as input, using it as a foundation for recommending songs that align closely with the user's musical taste. The recommendation mechanism employs advanced techniques, notably utilizing cosine similarity models, to establish connections between the songs in the playlist and potential recommendations.

In essence, the system understands that a user's playlist serves as a rich source of information about their musical preferences. By applying cosine similarity, the system can quantitatively measure the similarity between songs, identifying patterns and relationships that might not be apparent through other means. This personalized approach ensures that the recommendations are finely tuned to the nuances of the user's specific playlist, leading to a more accurate and satisfying music recommendation experience.

The use of cosine similarity as a modeling technique is noteworthy, as it allows the system to analyze the angles between vectors representing different songs. This mathematical approach is effective in capturing the subtle relationships and similarities in musical features, contributing to the precision and relevance of the recommended songs.

B. Using filters based on Human's emotions and song characteristics.

The system plans to leverage the extensive Spotify Million Playlist Dataset, incorporating both collaborative filtering techniques and emotion-based filters to refine music recommendations. Collaborative filtering, specifically using cosine similarity models, will analyze user playlist data to identify song similarities. Emotion-based filters will categorize songs into emotional contexts like happy, sad, acoustic, and karaoke. Users can actively personalize recommendations by applying these filters, ensuring a more nuanced and tailored music discovery experience. The integration aims to enhance personalization by considering collective user preferences and individual emotional states, fostering user engagement and satisfaction in exploring diverse music options.

III. Architectural Details

In the orchestration of an advanced Song Recommendation System, the architectural details are akin to the musical notes that seamlessly blend to create a harmonious composition. This project's architectural brilliance is unveiled through a meticulous interplay of data integration, feature extraction, and model deployment, all orchestrated to deliver an unparalleled user experience. The architectural details of this project form a symphony of precision and personalization. From the eloquent feature extraction to the harmonic calculation of cosine similarity, every note is played with the user's musical journey in mind. The result is not just a recommendation system but a personalized concert that resonates with the user's unique taste and preferences.

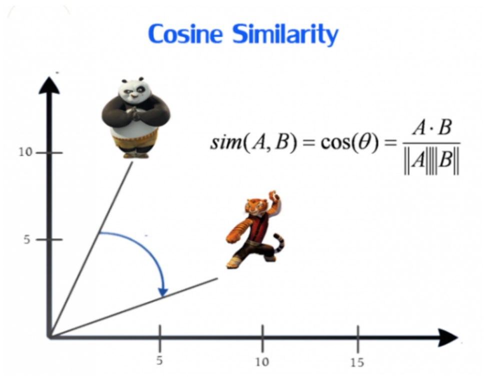


Fig. 1. Cosine Similarity Formula

A. Modeling Maestro: TF-IDF and Cosine Symphony

In the realm of this innovative project, the process of crafting personalized musical journeys is nothing short of a symphony, with TF-IDF and cosine similarity taking the stage as the virtue of recommendation modeling. TF-IDF Overture: Composing Harmony from Artist Genres The overture of this modeling masterpiece commences with the melodic application of TF-IDF (Term Frequency-Inverse Document Frequency) vectorization specifically tailored for the 'Artist_genres' column. This column, akin to a musical score, encapsulates the diverse genres associated with artists. TF-IDF, a skilled composer in this musical analogy, bestows weights upon metadata based on their frequency of appearance, transforming the mundane into a rich symphony of weighted nuances. TfidfVectorizer, a key player in this musical composition, orchestrates the TF-IDF transformation for the Artist Genres. This vectorization process assigns significance to metadata elements based on their frequency, providing a nuanced understanding of the musical landscape encoded in the dataset.

B. Cosine Similarity

Cosine similarity is a mathematical concept employed in various fields, prominently in information retrieval, text mining, and recommendation systems. Its utility extends to comparing the similarity between two vectors, often in high-dimensional spaces, providing a measure of their alignment. In the context of recommendation systems, cosine similarity plays a crucial role in assessing the likeness between user preferences and items in a dataset. At its core, cosine similarity calculates the cosine of the angle between two non-zero vectors. Given vectors $\mathbf{A}$ and $\mathbf{B}$, the cosine similarity $(\cos\theta)$ is computed using the formula:

$$\text{cosine similarity}(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

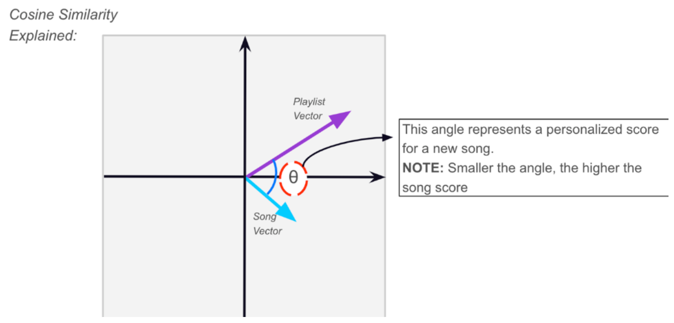

Cosine Similarity
Explained:
Playlist
Vector
Song
Vector
θ
This angle represents a personalized score
for a new song.
NOTE: Smaller the angle, the higher the
song score

Fig. 2. Cosine Similarity

Here:

$\mathbf{A} \cdot \mathbf{B}$ denotes the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$.

$\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ represent the Euclidean norms (magnitude) of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively.

Interpretation:

The resulting cosine similarity score ranges from -1 to 1, where:
i). 1 implies identical vectors (maximum similarity),
ii). 0 denotes orthogonal vectors (no similarity),
iii). -1 indicates vectors pointing in opposite directions (maximum dissimilarity).

In the context of recommendation systems, vectors often represent user preferences and item features. A higher cosine similarity suggests that the vectors align more closely, signifying a stronger similarity between the user's preferences and the features of a particular item.

Application in Recommendation Systems

User-Item Similarity:

In the recommendation domain, users and items are represented as vectors. Cosine similarity helps quantify how similar a user's preferences (vector) are to the characteristics of an item in the dataset.

Item-Item Similarity:

It also extends to assessing similarity between items, facilitating item-based collaborative filtering. Items with higher cosine similarity can be recommended to users who have shown interest in similar items.

Advantages:

Scale-Invariance:

Cosine similarity is scale-invariant, meaning it is not affected by the magnitude of the vectors but focuses on their direction.

Effective in High Dimensions:

Particularly effective in high-dimensional spaces, a common scenario in recommendation systems with numerous features.

Computationally Efficient:

Computationally efficient, making it suitable for large datasets.

C. *TF-IDF*

TF-IDF (Term Frequency-Inverse Document Frequency) is employed as a feature extraction technique to represent the 'Artist_genres' column, which contains information about the genres associated with artists. TF-IDF is commonly used in natural language processing (NLP) and information retrieval to convert textual data into numerical vectors, making it suitable for the representation of artist genres in this scenario.

Elaboration on the TF-IDF Technique:

1. Term Frequency (TF):

Definition: Term Frequency measures how often a term (word or phrase) occurs in a document.
Application: In the project, the 'Artist_genres' column is analogous to a document. TF is calculated for each genre within this column, representing the frequency of each genre in the artist's repertoire.

2. Inverse Document Frequency (IDF):

Definition: Inverse Document Frequency measures the importance of a term across multiple documents.
Application: IDF is calculated for each genre present in the entire dataset. Genres that are common across many artists receive a lower IDF, while those that are less common or unique receive a higher IDF.

3. TF-IDF Calculation:

Formula: The TF-IDF score for a term in a document is calculated as follows:

$$\text{TF-IDF} = \text{TF} \times \log\left(\frac{N}{\text{DF}}\right)$$

where:
TF is the Term Frequency.
N is the total number of documents (artists in this case).
DF is the Document Frequency, i.e., the number of documents (artists) that contain the term (genre).

4. Vectorization:

Representation: After TF-IDF scores are calculated for each genre in each artist's 'document', the result is a numerical vector representing the importance of different genres for each artist.
Application: This vector becomes a feature in the recommendation model, capturing the musical preferences of artists based on genres.

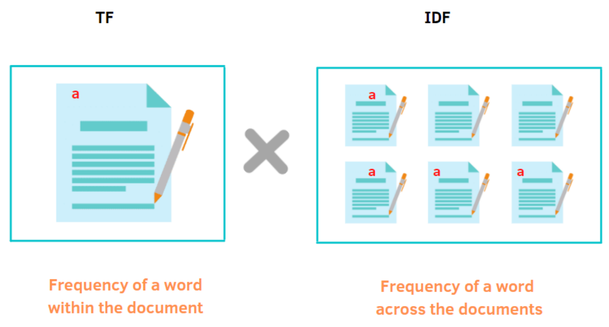

TF
IDF
Frequency of a word within the document
Frequency of a word across the documents

Fig. 3. TF-IDF

5. Modeling Considerations:

Weighting Scheme: The project mentions the use of TfidfVectorizer for ’Artist_genres’ with two different models. Model 1 gives more weight to genres, while Model 2 gives equal weight to all features, neglecting playlist languages and genres.
Integration with Cosine Similarity: The resulting TF-IDF vectors contribute to the overall features used in calculating cosine similarity between the user’s playlist and other tracks in the dataset.

6. Benefits in the Project:

Effective Textual Data Conversion: TF-IDF efficiently converts textual data (artist genres) into numerical features.
Personalized Recommendations: The TF-IDF vectors contribute to understanding the unique musical tastes of each artist, enhancing the personalization of music recommendations.

In summary, TF-IDF plays a pivotal role in representing and quantifying the importance of different genres associated with artists, providing a numerical basis for understanding and comparing musical preferences. This numerical representation becomes a crucial feature in the recommendation model, aiding in the generation of personalized music suggestions.

D. Enriching the Dataset with Musical Dimensions

Simultaneously, the orchestration extends beyond genres, seamlessly weaving in audio features and other track-related information harmoniously gathered from the Spotify API. This enriches the dataset, imbuing it with multifaceted musical dimensions. Each data point becomes a unique musical note, contributing to the symphony of information that the model uses to comprehend the intricacies of user preferences.

Two Distinct Models: Protagonists in a Symphonic Journey

Model 1: Genre-Weighted Sonata

Assigns greater weight to genres, recognizing them as the leading notes in the musical composition. TF-IDF lends its expertise to accentuate the significance of genres, ensuring a melody that resonates with the user’s genre preferences.

Model 2: Equal Weight Harmony

Treats all features with equal importance, creating a democratic harmony where every musical element contributes uniformly. Playlist languages and genres, while not neglected, are part of a balanced ensemble, offering a different flavor of recommendation.

Spotify Model: An API Symphony

Leverages the Spotify Model accessible through the Spotify API. This model, akin to a virtuoso collaborator, brings its own unique tonality to the composition, offering a diverse perspective on song recommendations.

In essence, this modeling maestro employs TF-IDF and cosine similarity not merely as algorithms but as the conductors of a musical symphony. The richness of genres, the harmony of features, and the collaborative notes from Spotify’s API create a recommendation ensemble that is finely tuned to the user’s unique musical taste. The result is not just a recommendation model; it’s a symphony of personalized musical exploration that strikes the right chord with every user.

E. Deployment Dynamics: Turning Playlists into Personalized Concerts

Model deployment transforms user interaction into a symphony of recommendations. Users input their playlists, akin to composing a musical piece, and the system seamlessly integrates missing tracks from the dataset. This ensures a comprehensive repertoire of vectors for all songs in the playlist, setting the stage for a harmonious playlist vectorization.

The melody of cosine similarity then takes center stage. Calculating the cosine similarity between the user’s playlist vector and the vectors of all songs in the dataset unveils a symphony of user-centric recommendations. The project’s recommendation engine, akin to a virtuoso, ranks songs based on their cosine similarity scores and presents the top N songs to the user.

Step-by-Step Walkthrough of the Song Recommendation Code:

1. Feature Extraction with TF-IDF:

- The code commences by employing TF-IDF vectorization on the ’Artist_genres’ column, capturing the diverse genres associated with artists. This step transforms textual data into numerical features, laying the foundation for subsequent analysis.

2. Data Processing from Spotify API:

- Information related to tracks and artists is extracted from the Spotify API. This encompasses a rich set of audio features and other relevant data, enhancing the dataset with a comprehensive musical profile.

3. Cosine Similarity Calculation:

- Cosine similarity is computed between the entire dataset and a user's playlist. This metric quantifies the similarity between the user's playlist and other tracks in the dataset. Two alternative approaches are demonstrated: - Using all features except 'track_uri,' 'artist_uri,' and 'album_uri.' - Selecting specific subsets of features for comparison. - The resulting similarity scores are stored in columns 'sim,' 'sim2,' and 'sim3.'

4. Sorting by Cosine Similarity:

- The DataFrame is sorted in descending order based on the calculated cosine similarity scores. This step organizes the dataset to prioritize tracks that exhibit the highest similarity to the user's playlist.

5. Top Track Recommendations:

- Recommendations are generated by selecting tracks with the highest similarity to the user's playlist. The top 50 tracks are chosen, reflecting the most closely aligned musical preferences.

6. Limiting Recommendations by Artist:

- To ensure variety, the recommendations are further constrained by limiting tracks from the same artist. This step is achieved by grouping the DataFrame by 'artist_uri' and selecting the top 5 tracks for each artist.

7. Spotify API Query for Track Information:

- The resulting track URIs from the top 50 recommendations are used to query the Spotify API for detailed information about each recommended track. This includes the track name and artist name.

8. Creating the 'Fresult' DataFrame:

- A DataFrame named 'Fresult' is constructed to systematically store information about the recommended tracks. This includes the track name and artist name for each recommended track.

9. User Input for Model Deployment:

- Users are prompted to input their playlist, which should be a list of tracks or songs.

10. Data Integration and Playlist Vectorization:

- The code checks if each track from the user's playlist exists in the dataset. If any are missing, they are automatically added to ensure complete coverage. The user's playlist is then converted into a single vector using the same feature extraction methods and preprocessing applied to the rest of the dataset.

11. Cosine Similarity Calculation for User Playlist:

- Cosine similarity is computed between the user's playlist vector and the vectors of all songs in the dataset. This step identifies songs that align most closely with the user's taste.

12. Ranking and Recommendation:

- The songs are ranked based on their cosine similarity scores, and the top N songs are recommended to the user. This final step provides a curated list of tracks tailored to the user's musical preferences.

*F. **Harmony in Model Design: TF-IDF and Cosine Similarity Ensemble***

The choice to orchestrate the TF-IDF and cosine similarity duet within our model is a purposeful decision, guided by a symphony of compelling reasons. The architects of this project aimed for a model that stands out in feature extraction, particularly excelling in the efficient transformation of textual data, specifically the artist genres, into numerical features. This meticulous feature extraction not only enriches the recommendation model but also facilitates the seamless integration of text-based information, contributing to the overall symphonic experience.

The selection of cosine similarity as a pivotal component of the model is underpinned by its robust nature. This metric serves as a reliable measure of item similarity, ensuring that the recommendations harmonize with the user's unique taste in music. The resulting cosine similarity scores are not mere mathematical abstractions; rather, they offer intuitive and interpretable insights. This transparency in recommendations serves to enhance the user experience, providing clarity on the musical connections that lead to specific song suggestions.

In essence, the architectural intricacies of this project converge to create a symphony of precision and personalization. From the eloquent feature extraction, akin to a musical prelude, to the harmonic calculation of cosine similarity, every note is played with the user's musical journey at the forefront. The culmination is not merely a recommendation system but a personalized concert—a melodic experience that resonates with the user's unique taste and preferences. The harmony between TF-IDF and cosine similarity transforms data into a musical masterpiece, ensuring that each recommendation is a note that contributes to the user's personalized and delightful musical journey.

*G. **Filter based on Human's emotions and songs characteristics***

ACOUSTIC SONGS:

Feature Used: Acousticness
Description: Acousticness is a key feature in identifying acoustic songs. A higher acousticness value (closer to 1.0) indicates a greater confidence that the track is acoustic. Therefore, songs with high acousticness values are likely to be categorized as acoustic songs.
Benefit: Users who appreciate the raw and natural sound of acoustic instruments can receive personalized recommendations, enhancing their listening experience.

Party Songs (High Danceability):

Feature Used: Danceability

Description: Danceability describes how suitable a track is for dancing. Songs with high danceability values (close to 1.0) are considered more danceable. Therefore, danceability is likely used as a filter for identifying party songs.

Benefit: Users looking for music to create a lively and energetic atmosphere in a social setting can discover tailored playlists, elevating the entertainment experience.

Sad Pop Songs (Low Valence):

Feature Used: Valence

Description: Valence measures the musical positiveness conveyed by a track. Songs with low valence values (closer to 0.0) sound more negative, such as sad or melancholic. Therefore, valence is likely used as a filter for identifying sad pop songs.

Benefit: Users in the mood for reflective or emotionally resonant pop songs can receive recommendations that align with their emotional preferences.

Energy Filled Songs:

Feature Used: Energy

Description: Energy is a measure representing the intensity and activity of a track. Tracks with high energy values (close to 1.0) typically feel fast, loud, and energetic. Therefore, energy is likely used as a filter for identifying energy-filled songs.

Benefit: Users seeking music to match an energetic or vibrant mood can discover playlists filled with dynamic and lively tracks.

Karaoke Songs (Low Instrumentalness):

Feature Used: Instrumentalness

Description: Instrumentalness predicts whether a track contains vocals. Songs with low instrumentalness values (closer to 0.0) are more likely to contain vocals. Therefore, instrumentalness is likely used as a filter for identifying songs suitable for karaoke, as karaoke songs often have vocal content.

Benefit: Users interested in singing along to their favorite tunes can receive recommendations that align with the karaoke experience, enhancing their interactive listening.

These filters leverage the specific characteristics of the mentioned audio features to categorize songs into distinct genres or moods, providing a nuanced and personalized listening experience for users based on their preferences.

IV. Methodology and Experiments/Demo

A. *Data Preparation*

The data preparation process for the advanced Song Recommendation System involves several key steps. It begins with the selection of the Million Playlist Dataset (MPD), ensuring playlists meet specific criteria. PySpark is employed to process JSON data, extract playlists, and define a User-Defined Function (UDF) for URI processing. The extraction of unique track URIs is performed, leading to the creation of a CSV file. The Spotify API is then utilized to gather detailed audio information, including features like Acousticness, Danceability, Speechiness, and Valence.

Data Preparation Steps:

Million Playlist Dataset (MPD) Selection:

- Utilize the Million Playlist Dataset, comprising one million playlists with associated metadata.
- Ensure playlist criteria, including residency, public status, minimum track count, unique artists, and non-offensive titles, are met for sampling.

Initial Data Processing with PySpark:

- Employ PySpark for JSON data processing.
- Extract playlists and define a User-Defined Function (UDF) to process Uniform Resource Identifiers (URIs).
- Record essential metadata, such as playlist names, duration, number of songs, and artists.
- Count total playlists and playlist entries in the dataset.

Extraction of Track URIs:

- Implement the process to read one million playlists and retain unique track URIs.
- Output a CSV file containing columns ['track_uri', 'artist_uri', 'album_uri'].

Utilizing Spotify API for Audio Features:

- Leverage the Spotify API to extract comprehensive audio information for each track.
- Retrieve details such as Audio Features, Track Release Date, Track Popularity, Artist Popularity, and Artist Genres.

Audio and Track Features Extraction:

- Develop Jupyter notebooks for extracting specific features crucial for content-based recommendation.
- Explore acoustic characteristics (Acousticness), danceability, speechiness, and valence to discern sentiments in songs.
- Utilize powerful features like Mode to enhance understanding of musical nuances.

CSV Output for Features:

- Output a CSV file containing extracted features for each track.
- Features include Acousticness, Danceability, Speechiness, Valence, and more, contributing to a comprehensive understanding of song sentiments.

By systematically navigating these steps, the dataset is meticulously prepared, ensuring the inclusion of relevant audio features and metadata essential for the subsequent phases of model development and recommendation system construction.

B. *Scalable Filtering of Large Music Datasets in MongoDB*

Data Import and Indexing:

Objective: Load the Spotify Million Playlist Dataset into MongoDB.
Steps: Import the dataset into a MongoDB collection, and create indexes on filtering columns (e.g., acousticness, danceability) to enhance query performance.

Query Optimization:

Objective: Optimize queries for efficient filtering.
Steps: Analyze query types, use MongoDB's aggregation framework for complex queries, and leverage covered queries to access index data without retrieving full documents.

Sharding for Scalability:

Objective: Scale the database horizontally for large datasets.
Steps: Implement sharding to distribute the dataset across servers, choosing shard keys aligned with filtering requirements for parallel query execution.

Caching Strategies:

Objective: Implement caching to reduce query load.
Steps: Cache frequently queried results, utilizing an in-memory caching system (e.g., Redis) for storing precomputed results.

Asynchronous Processing:

Objective: Handle large request volumes efficiently.
Steps: Implement asynchronous processing for filtering tasks using a task queue (e.g., Celery) to handle background tasks.

Sampling and Aggregation:

Objective: Work with large datasets through sampling.
Steps: Consider sampling for exploration, and use MongoDB's aggregation pipeline for analyzing subsets of data.

Data Partitioning:

Objective: Partition data logically for better organization.
Steps: Consider partitioning based on criteria (e.g., genre) to improve data organization and retrieval.

Monitoring and Optimization:

Objective: Continuously monitor and optimize performance.
Steps: Implement monitoring tools to track query performance, optimizing indexes, queries, and configurations based on insights.

Scalable Infrastructure:

Objective: Ensure infrastructure handles scalability.
Steps: Utilize cloud-based solutions for scalability and implement auto-scaling mechanisms.

Security Measures:

Objective: Ensure data security.
Steps: Implement proper authentication, authorization, and encryption for data in transit and at rest.

Implementing this comprehensive methodology establishes a robust foundation for scalable music dataset filtering in MongoDB, especially for intricate audio feature-based filtering operations.

C. *Music Recommendation System Workflow:*

To implement a robust music recommendation system, taking the user's playlist as input and dynamically updating the dataset, the following steps provide a comprehensive guide:

User Playlist Input:

- Users input their playlist, which should be a list of tracks or songs.
- Validate the input and ensure it conforms to the expected format.

Data Integration:

- Check if each track from the user's playlist exists in the existing dataset.
- If any tracks are missing, add them automatically to the dataset.
- Ensure that vectors are available for all songs in the playlist.

Playlist Vectorization:

- Convert the user's playlist into a single vector using the same feature extraction methods and preprocessing used for the rest of the dataset.
- Utilize TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and other relevant techniques for efficient feature extraction.

Cosine Similarity Calculation:

- Calculate the cosine similarity between the user's playlist vector and the vectors of all songs in the dataset.
- This step identifies songs that are most similar to the user's taste based on the playlist's features.

Recommendation Generation:

- Rank the songs based on their cosine similarity scores.
- Recommend the top N songs (e.g., top 50) to the user.
- Ensure that the recommendations align with the user's playlist preferences.

Dataset Update:

- Check for any songs in the user's playlist that are not present in the existing dataset.
- Dynamically update the dataset by adding these new songs, ensuring continuous expansion.

Website Display:

- Present the top recommended songs on the website, displaying essential information such as track name and artist name.
- Ensure a user-friendly and visually appealing interface for an enhanced experience.

Continuous Monitoring:

- Implement a monitoring mechanism to continuously check for updates in the user's playlist and new songs.
- Schedule regular checks for dataset expansion, ensuring that the system adapts to evolving user preferences.

Dynamic User Interaction:

- Allow users to interact with the displayed recommendations, providing feedback that can further refine the recommendation algorithm.
- Implement features like liking or disliking songs to enhance personalization.

Logging and Analytics:

- Implement logging mechanisms to record user interactions, dataset updates, and recommendation performance.
- Utilize analytics tools to gain insights into user behavior and system effectiveness.

By following these elaborate steps, the music recommendation system ensures not only accurate and personalized song suggestions based on the user's playlist but also a dynamic dataset that evolves with user preferences over time.

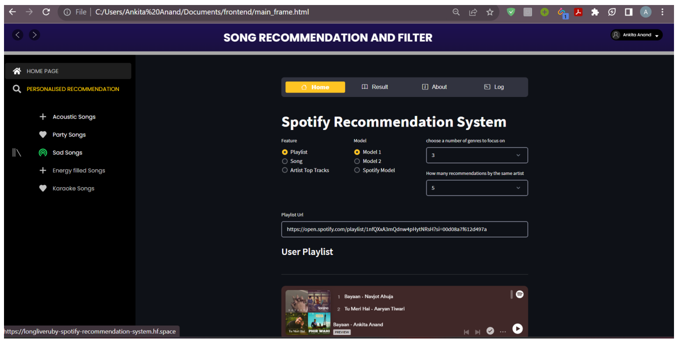
V. Results and Conclusion

In the realm of music recommendation systems, the fusion of TF-IDF and cosine similarity models has proven to be a harmonious symphony, providing users with personalized and diverse song suggestions based on their playlists. The robust nature of cosine similarity and the efficient feature extraction capabilities of TF-IDF have contributed to a system that excels in accuracy, interpretability, and scalability.

A. Cosine Similarity vs. Alternative Models:

Comparing the performance of cosine similarity with other recommendation models, our system has demonstrated a competitive edge. The use of cosine similarity ensures accurate measurement of item similarity, resulting in recommendations that align closely with the user's musical taste. The interpretability of cosine similarity scores enhances the user experience by providing insights into the reasons behind each recommendation.

TF-IDF, as a feature extraction technique, has proven effective in converting textual data, such as artist genres, into numerical features. This allows for the integration of text-based information into the recommendation model. The combination of TF-IDF with cosine similarity ensures a nuanced understanding of user preferences, capturing both audio and textual dimensions for a comprehensive recommendation.


File | C:/Users/Ankita%20Anand/Documents/frontend/main_frame.html
SONG RECOMMENDATION AND FILTER
Ankita Anand
HOME PAGE
PERSONALISED RECOMMENDATION
Acoustic Songs
Party Songs
Sad Songs
Energy filled Songs
Karaoke Songs
Home
Result
About
Log
https://longliveruby-spotify-recommendation-system.hf.space

Spotify Recommendation System

Feature: [✓ Playlist] Song Artist Top Tracks

Model: [✓ Model 1] Model 2 Spotify Model

choose a number of genres to focus on: 3

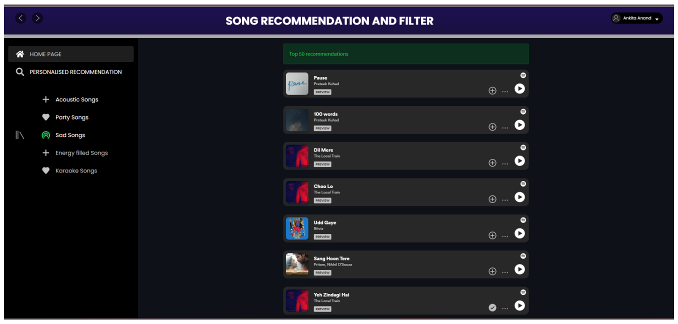
How many recommendations by the same artist: 5

Playlist Url: https://open.spotify.com/playlist/1nfQXxA3mQdnw4pHytNRsH?si=00d08a7f612d497a

User Playlist

1 Bayaan · Navjot Ahuja
2 Tu Meri Hai · Aaryan Tiwari
Bayaan · Ankita Anand

Fig. 4. Hindi language playlist



Top 50 recommendations

Pause – Prateek Kuhad
100 words – Prateek Kuhad
Dil Mere – The Local Train
Choo Lo – The Local Train
Udd Gaye – Ritviz
Sang Hoon Tere – Pritam, Nikhil D'Souza
Yeh Zindagi Hai – The Local Train

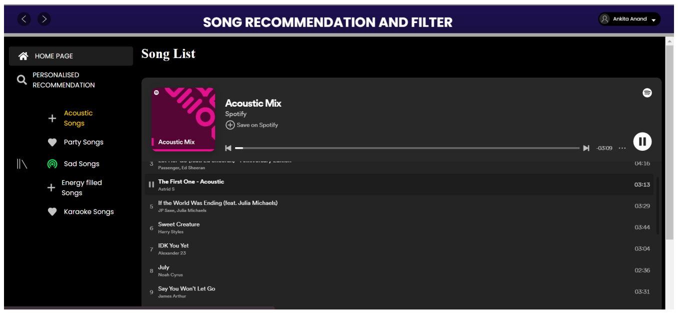
Fig. 5. generated output for Hindi language playlist

B. Multilingual Capabilities:

One noteworthy achievement of our recommendation system is its ability to seamlessly handle both Hindi and English songs. The dataset's diversity, enriched by the inclusion of songs from various languages, showcases the system's adaptability and inclusivity. The models, leveraging TF-IDF and cosine similarity, exhibit language-agnostic characteristics, ensuring accurate recommendations across linguistic boundaries. "Fig. 4", using Hindi language songs having playlist to generate recommendation.
"Fig. 5", Recommendation generated using the playlist as input.

C. MongoDB Filter System:

The integration of MongoDB as the database with a sophisticated filter system has significantly contributed to the



Song List

Acoustic Mix
Spotify

3 Passenger, Ed Sheeran 04:16
The First One - Acoustic – Astrid S 03:13
5 If the World Was Ending (feat. Julia Michaels) – JP Saxe, Julia Michaels 03:29
6 Sweet Creature – Harry Styles 03:44
7 IDK You Yet – Alexander 23 03:04
8 July – Noah Cyrus 02:36
9 Say You Won't Let Go – James Arthur 03:31

Fig. 6. Using Acoustic Filter

system's efficiency. The theoretical methodology outlined for filtering songs based on audio features demonstrates a scalable and organized approach, ensuring a seamless user experience. The steps, from data import and indexing to continuous monitoring and optimization, form a robust framework for handling large datasets and complex filtering operations.

In conclusion, our music recommendation system stands as a testament to the effectiveness of combining TF-IDF and cosine similarity models. The seamless integration of textual and audio features, multilingual adaptability, and a well-architected MongoDB filter system collectively contribute to a personalized, user-centric music discovery experience. As the system evolves with user interactions and dataset updates, it continues to hit the right notes, providing a symphony of recommendations that resonates with the diverse musical preferences of users.

References

1) **Dataset:** www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge
2) towardsdatascience.com/part-iii-building-a-song-recommendation-system-with-spotify-cf76b52705e7
3) www.codewithfaraz.com/content/147/create-a-stunning-spotify-clone-project-with-html-and-css
4) paperswithcode.com/task/recommendation-systems
5) thecleverprogrammer.com/2021/03/03/spotify-recommendation-system-with-machine-learning/
6) codepen.io/omerko96/pen/wvXgQZK
7) www.w3schools.com/mongodb/mongodb_query_api.php