iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VI **Month of publication:** June 2022

DOI: https://doi.org/10.22214/ijraset.2022.44381

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

Song Recommendation System using TF-IDF Vectorization and Sentimental Analysis

B Venkata Sai Chirasmayee[1], G Sharmila[2], D Sahithi[3], Vadakattu Prabhakar[4]
[1, 2, 3]*Students of Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad*
[4]*Assistant Professor, Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad*

Abstract: ***In this project, we will build a machine learning model that recommends songs based on the songs already present in the playlist. We will implement Content based recommendation system. We will also create a web page in which the user can give the public playlist URI as input, and we will provide 5 to 20 song recommendations. Each song recommendation will be a link to the song and the song can be played on Spotify. Developers are now able to access millions of songs and obtain information like the song's genre, the artist's name, song's release date, album name and so on through the Spotify API A developer can obtain can build Machine Learning models using the Spotify API using secret key and client id specific to the developer. We will be using spotipy package for Authentication of Spotify Developer credentials and for accessing the songs data. We will be using python scikit learn module for building Machine learning model. Flask module is used for web development. The final output will be a website deployed on local host.***
Keywords: Recommender system, Machine Learning, Content-based filtering

I. INTRODUCTION

Media content consumption has become part of every person's life irrespective of their financial status. Music streaming services and OTT platforms are growing in number, and they are marketing very effectively to more subscribers. To get more active users, these companies must offer something unique to every user. They must provide a personalized user experience for the subscribers. This is accomplished through recommender systems.

Recommender systems are useful to both service providers and users. They have proved to improve decision-making process and quality. In our life, we take other people's recommendations or suggestions when we do not have knowledge or experience. Recommender systems use this idea of assistance and augment it into computational realm.

The Spotify API is very powerful and provides information regarding any artist or song on Spotify. It provides data on what is the "feel" of the song, with variables like "danceability", "instrumentalness" and "speechiness" and features describing the artist's popularity and song's popularity.

We can attain information regarding key, modal, valence, and tempo in which the song was played, beats in the songs and analysis of the song at an advanced level. Spotify provides a console to test functionality of your implementation which helps you to debug your implementation.

II. BACKGROUND AND RELATED WORK

A. Spotify Million Playlist Dataset

According to a study published in May 2022, the number of monthly active users of Spotify has increased from 68 million in first quarter of 2015 to 422 million by 2022 first quarter[1].

So, Spotify is the largest music streaming service across the world. Spotify Research group collects user data and uses it for analysis to market their product better, to improve user experience. Spotify Research group provided Spotify million playlists dataset as a resource in ACM Recsys Challenge 2018[2]. This challenge was an attempt to find creative song recommendation algorithms from the challenge participants.

The most popular thing about the Spotify API is the playlist feature because people are loving the playlists more than the albums because we can save the songs we like to listen again and again. We can create them according to our mood. The Spotify playlists contain information about the user's emotion. We can suggest more songs based on user's playlist, which makes the user want to listen to more songs.

B. Recommender Systems and the Types

There are 6 types in recommender systems: Collaborative Recommender system, Content-based recommender system, Demographic based recommender system, Utility based recommender system, Knowledge based recommender system and Hybrid recommender system.

1) *Collaborative Recommender System:* As the name suggests, collaborative recommender system collaborates with other users' data who have similar taste to the user in question or active user and then recommends the items that the other users have like to the active user. Aggregation of ratings, accumulating commonalities among subscribers, make comparisons among users (inter-user) and then come up with recommendations. These techniques are not dependent on object's representation, and they are suitable for complex objects that determine fluctuations in preferences based on fluctuations in taste. CF makes an assumption that users' taste will remain same in the future and that they will prefer what they preferred in the past.
2) *Content based Recommender System:* Content-based recommender systems use subscriber's data only. They do not take other similar subscribers' data into consideration like collaborative recommender system. This recommender learns from the profile of the subscriber's preference depending on the features, in items the subscriber has rated. This filtering is dependent on keywords that describe the objects. This recommender system recommends items similar to the ones that the subscriber has preferred in the past.
3) *Demographic based Recommender System:* This Recommender System recommends items to the users based on the demographic they belong to. These recommender systems use demographics like gender, age, geographic location etc. to make recommendations. This technique does not need user's history or user's rating data to make recommendations.
4) *Utility based Recommender System:* Unlike Collaborative filtering and content-based filtering, utility-based recommender system focuses on items instead of users. It considers each item's utility and uses attributes like availability of vendor and availability of item in computation. This recommender system only suggests items that are available in real time to the user.
5) *Knowledge based Recommender System:* This recommender system has resourceful knowledge of how a particular item that the user has preferred meets the user's need. It understands the relationship between the rated item and the user. Based on this reasoning, it provides recommendations to the user.
6) *Hybrid Recommender System:* Any recommender system alone has a weakness when concerned with one field. Hybrid recommender systems are two or more recommender systems put together for a particular domain. These recommender systems are created for a company and the developers team up with the company. Strengths of more than one recommender systems remove the weaknesses only one recommender system has.

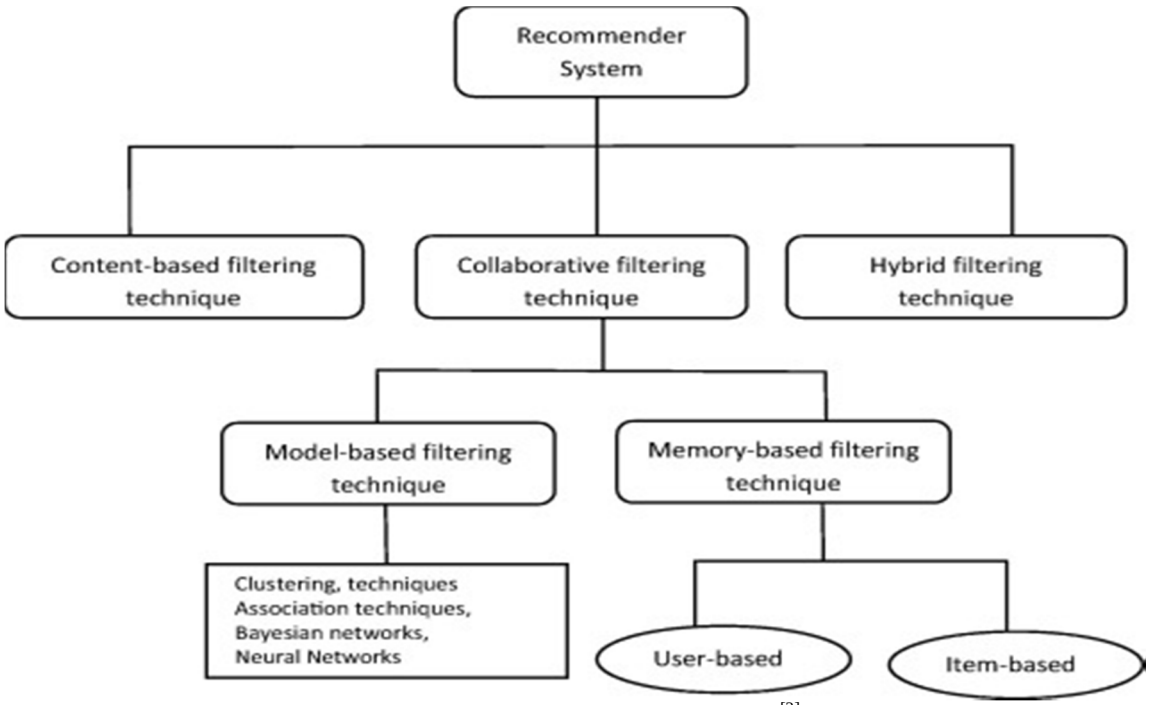


Fig 2.2.1 Recommendation systems classification [3]

C. Content Based Filtering and its Methods

Content-based filtering is a domain-based algorithm which concentrates on analysis of attributes of items to generate predictions. In CBF, recommendation will be based on the user profiles using features extracted from the content of items the user has evaluated in the past. CBF uses different types of models to find similarity between items to generate correct recommendations. CBF uses Vector Space Model such as Term Frequency Inverse Document Frequency (TF/IDF) or Neural Networks or Probabilistic models like Naïve Bayes Classifier to model the relationship between documents.

1) Pros of CBF

- Provides recommendations irrespective of ratings and user preferences
- If user preference changes, this filtering has the capability to adjust its recommendations in a short span of time.

2) Cons of CBF

- Highly dependent on items' metadata. The dataset provided for training the model must have rich description of items and should be well organized. This is called limited content analysis
- Content overspecialization: The recommendations are highly like the items present in the user profile.

III. EXISTING SYSTEM VS PROPOSED SYSTEM

A. Existing System

Music streaming platforms like Spotify, Amazon Music, Gaana, Wynk etc. need to pay royalties to the artists, recording labels and publishers. Before going public in February 2018, Spotify has stated that it has paid more than $9.7 billion since 2006. The only way Spotify can make profits is by increasing the number of monthly active users and paid subscribers. This means that they must attract more subscribers and increase the number of monthly active users.

Music streaming platforms can recommend songs that make Billboard top 100, or a new album released by a popular artist. They can recommend songs based on the user's demographic. But these types of recommendations alone are not sufficient. Due to boom of content creation and freedom to create and post music online via YouTube or any other internet service, more independent music artists are gaining popularity. The number of albums or single tracks released every day is also increasing. So, music streaming platforms should focus more on how to utilise the user data to understand each user's preference and recommend songs based on that data to improve user experience.

B. Proposed System

According to a study carried out in 2016 by the Music Business Association[4] as part of their Music Biz Consumer Insights program[5], playlists accounted for 31% of music listening time among listeners in the United States, which is more than albums (22%), but less than single tracks (46%). In a 2017 study conducted by Nielsen[6], it was found that 58% of users in the United States create their own playlists, 32% share them with others.

Other studies, conducted by MIDiA[7] ,show that 55% of music streaming service subscribers create music playlists, using streaming services. Studies like these suggest a growing importance of playlists as a mode of music consumption, which is also reflected in the fact that the music streaming service Spotify currently hosts over 2 billion playlists. For this reason, we propose recommendation system based on the playlists created by the user.

IV. DESIGN AND IMPLEMENTATION

In this project, we will create a song recommendation system that provides recommendations for the public playlist URI provided by a Spotify user. We will implement content-based filtering with TF-IDF vectorization and perform sentimental analysis on every song/track name.

A. Data Import and Data pre-processing

Spotify million playlist dataset is given as a series of thousand json files each file containing thousand playlists thus marking a total of 1 million playlists. During the development phase we have only considered the first json file thus creating raw data from only the first thousand playlists. First_1000.json file contains the first thousand playlists. In load_data.py file we are extracting all the tracks that are present in each playlist. Then we are saving the data into raw_data.csv file

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

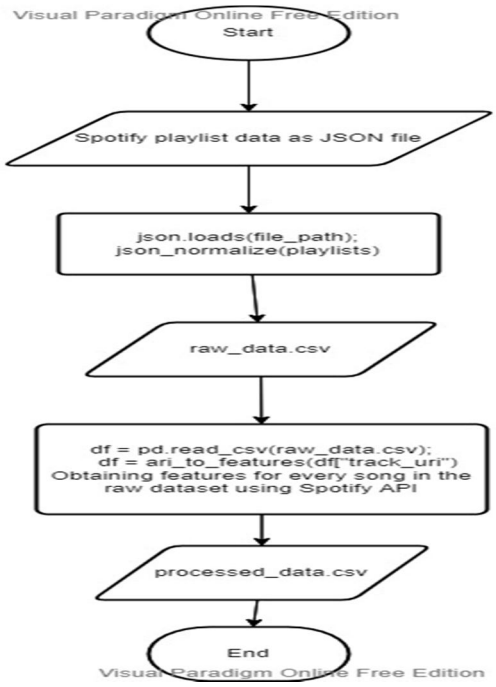


Fig 4.1.1 Parsing json file, saving processed data

Data import step is important because we need to obtain metadata of every song track and create a rich dataset. We need rich dataset because CBF performs limited content analysis. After obtaining raw data, we need to extract features on every track. These features are provided by Spotify API: pos, artist_name, track_uri, artist_uri, track_name, album_uri, duration_ms, album_name, name, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, type, id, uri, track_href, analysis_url, duration_ms_y, time_signature

To obtain this information regarding every track, we should sign up on Spotify for Developers and create a project and obtain client_id and client_secret of the project. These credentials are used to authenticate the developer and their project by Spotify. We created a project named “MySongCharm”. We saved the client_id and client_secret in “secret.txt” file. After obtaining metadata on every song, we save the dataset as processed_data.csv. Before feature extraction, we must conduct data pre-processing that will cater specifically for content-based filtering:

1) Useful Data Selection

a) Drop duplicates: Multiple users save same song across different playlists. Songs will repeat across playlists. Such songs are removed from the dataset.

b) Selecting specific columns from the dataset.

2) List Concatenation

a) Genre pre-processing:

b) Playlist pre-processing

B. Feature Extraction

Now that the data is usable, we can now feature-engineer the data for the purpose of the recommendation system. In this project, the following process is conducted into a pipeline for feature generation.

1) Sentiment Analysis

2) One-hot Encoding

3) TF-IDF

4) Normalization

a) *Sentimental Analysis:* Sentiment analysis is a type of subjectivity analysis (Wiebe 1994) that focuses on identifying positive and negative opinions, emotions, and evaluations expressed in natural languages. It has been a central component in applications ranging from recognizing inflammatory messages (Spertus 1997), to tracking sentiments over time in online discussions (Tong 2001), to classifying positive and negative reviews (Pang, Lee, and Vaithyanathan 2002; Turney 2002). In our data, we will use TextBlob package and import subjectivity and polarity from the package to perform sentimental analysis on the track name.

Subjectivity (0,1): The amount of opinion and information contained in the text.

Polarity (-1,1): The degree of strong or clearly defined sentiment accounting for negation.

Implementing prior polarity subjectivity lexicon [8] means that we will classify each word as highly subjective or weakly subjective. Words that are subjective in most contexts are considered strong subjective clues, indicated by the strongsubj tag. Words that may only have certain subjective usages are considered weak subjective clues, indicated by the weaksubj tag.

A word in the lexicon is tagged as positive if out of context it seems to evoke something positive, and negative if it seems to evoke something negative. A word may have positive and negative meanings, then it is assigned with what seems to be more common. A word is tagged as neutral if it is at the same time both positive and negative. We will then use one-hot encoding to list the sentiment of the song titles as one of the inputs.

b) *TF IDF Vectorization*

TF-IDF Vectorization is a vector space model that is dependent on the aspect of similarity [9]. The model assumes that the relevance of a document to query is roughly equal to the document-query similarity. TF-IDF stands for term frequency and inverse document frequency. These two are the most used concepts in retrieval of textual information. These are the two matrices that are closely interrelated and search and figure out the relevancy of a given word to a document given a larger body of document. Term frequency refers to the number of times that a term t occurs in document d. The inverse document frequency is a measure of whether a term is common or rare in each document corpus. It is obtained by dividing the total number of documents by the number of documents containing the term in the corpus. All TF means is how often a given word occurs in each document so within one web page one Wikipedia article, how common is a given word within that document, what is the ratio of that word occurrence rate throughout all the words in that document that's it. TF measures how often a word occurs in a document. A word that occurs frequently is probably important to that document's meaning. DF is how often a word occurs in an entire set of documents, i.e., all of Wikipedia or every web page. this tells us about common words that just appears everywhere no matter what the topic, like 'a', 'the', 'and', etc. Word with high TF and DF both might not be important measure relevancy of a word to a document. So, a measure of relevancy of a word to a document is:

TF/DF (or) Term frequency * Inverse Document Frequency

That is, how many times the word appears in a document, over how often it just appears everywhere. That gives you a measure of how important and unique this word is for this document. We use the log of the Inverse document frequency since word frequencies are distributed exponentially. That gives us a better weighting of words overall popularity. TF-IDF assumes a document is just a "bag of words". Words can be represented as a hash value (number) for efficiency. The general formula for calculating TF-IDF is:

Term Frequency × Inverse Document Frequency

Term Frequency (TF): The number of times a term appears in each document divided by the total word count in the document.

Inverse Document Frequency (IDF): The log value of the document frequency i.e., number of documents containing the word

TF-IDF: Term x in document y

$$W_{x,y} = TF_{x,y} * \log(N/DF_x)$$

$$TF_{x,y} = frequency\ of\ x\ in\ y$$

$$DF_x = number\ of\ documents\ containing\ x$$

$$N = total\ number\ of\ documents$$

The motivation is to find words that are not only important in each document but also important in entirety of all documents. The log value was taken to decrease the impact of a large N, which would lead a very large IDF compared to TF. TF focuses importance of a word in a document, while IDF focuses on the importance of a word across documents. In this project, the documents are analogous to songs. Therefore, we are calculating the most prominent genre in each song and their prevalent across songs to determine the weight of the genre. There are no weights to determine how important and widespread each genre is, leading to overweighting on uncommon genres which makes this a better way than the one hot encoding.

Lastly, we need to normalize some variables. As shown below, the popularity variables are not normalized to 0 to 1, which would be problematic in the cosine similarity function later. In addition, the audio features are also not normalized. To solve this problem, we used the MinMaxScaler() function from scikit learn which automatically scales all values from the min and max into a range of 0 to 1. The feature set created from the above method will be used to generate recommendations. complete_feature.csv contains entire feature set values of all the tracks given in the dataset. allsong_data.csv only contains the float type columns in the processed_data.csv

C. Content Based Filtering Recommendation

The next step is to perform content-based filtering based on the song features we have. To do so, we concatenate all songs in a playlist into one summarization vector. Then, we find the similarity between the summarized playlist vector with all songs (not including the songs in the playlist) in the database. Then, we use the cosine similarity measure to retrieve the most relevant song that is not in the playlist to recommend it.

There are three steps in this section:

1) *Choose Playlist:* In this part, we retrieve a playlist
2) *Extract Features:* In this part, we retrieve playlist-of-interest features and non-playlist-of-interest features.
3) *Find Similarity:* In this part, we compare the summarized playlist features with all other songs.

a) *Choose Playlist:* In this step, the user can select a public playlist and copy it's URI and give it as input on the webpage. Public playlist URIs start with "open.spotify". We have written code for extracting the playlist id from the URI and saving all the playlist tracks into a DataFrame. In this way, we retrieve the playlist.

b) *Extract Playlist Features:* The next step is to generate all the features. We need to first use the id to differentiate songs that are in the playlist and those that are not. Then sum up all the features for all the tracks in the playlist to give a new vector as summary.

c) *Cosine Similarity:* The last puzzle is to find the similarities between the summarized playlist vector and all other songs. There are many similarity measures but one of the most common measures is cosine similarity. Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. [10]

Mathematical formula for cosine similarity:

$$Cosine\ Sim(X,Y) = \frac{X.Y}{||X||*||Y||} = \frac{\sum_{i=1}^{n} X_i * Y_i}{\sqrt{\sum_{i=1}^{n} Xi^2} * \sqrt{\sum_{i=1}^{n} Y_i^2}}$$

Imagining our songs vectors as only two dimensional, the visual representation would look like the figure. In scikit_learn we have cosine_similarity() function. We used this measure to find similarity between each song and the summarized playlist vector. An advantage of cosine similarity is that it has the same time complexity as matrix multiplication. We are performing the cosine similarity measure between each row vector (song) and the column vector of summarized playlist feature.

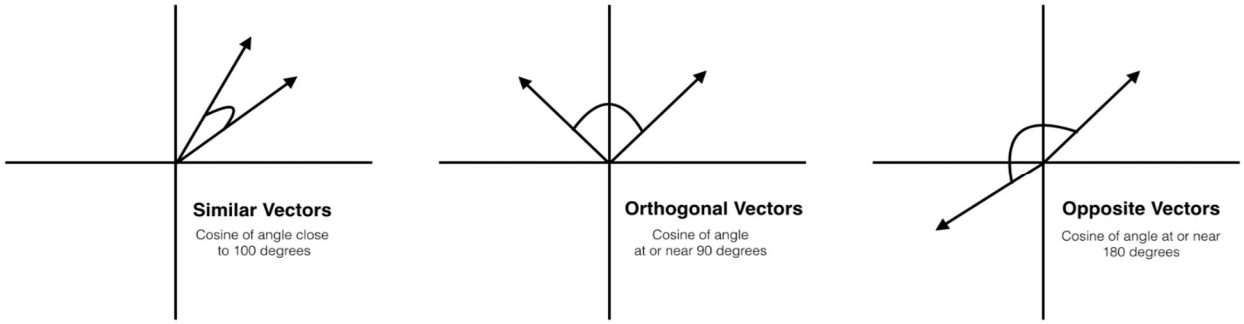


Figure 4.3.3.1: Cosine similarity based on cosine angle degree

V. RESULTS

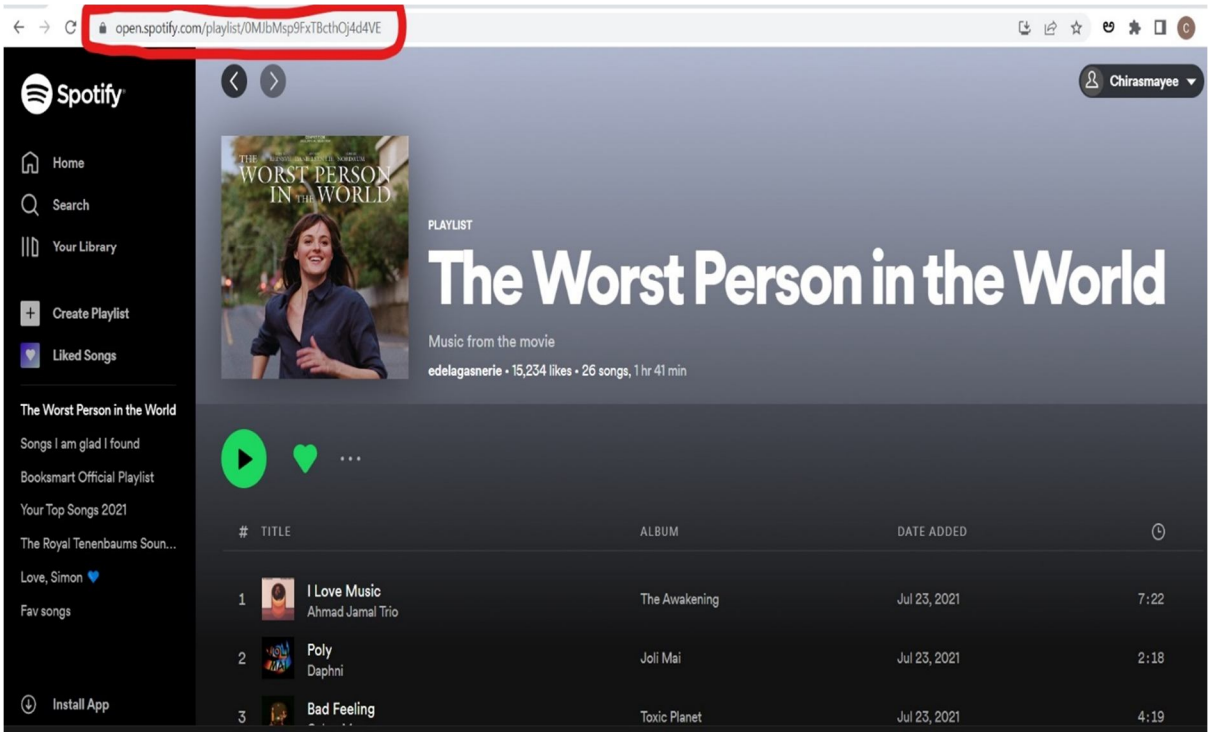


Fig 5.1 Selecting a public playlist URI

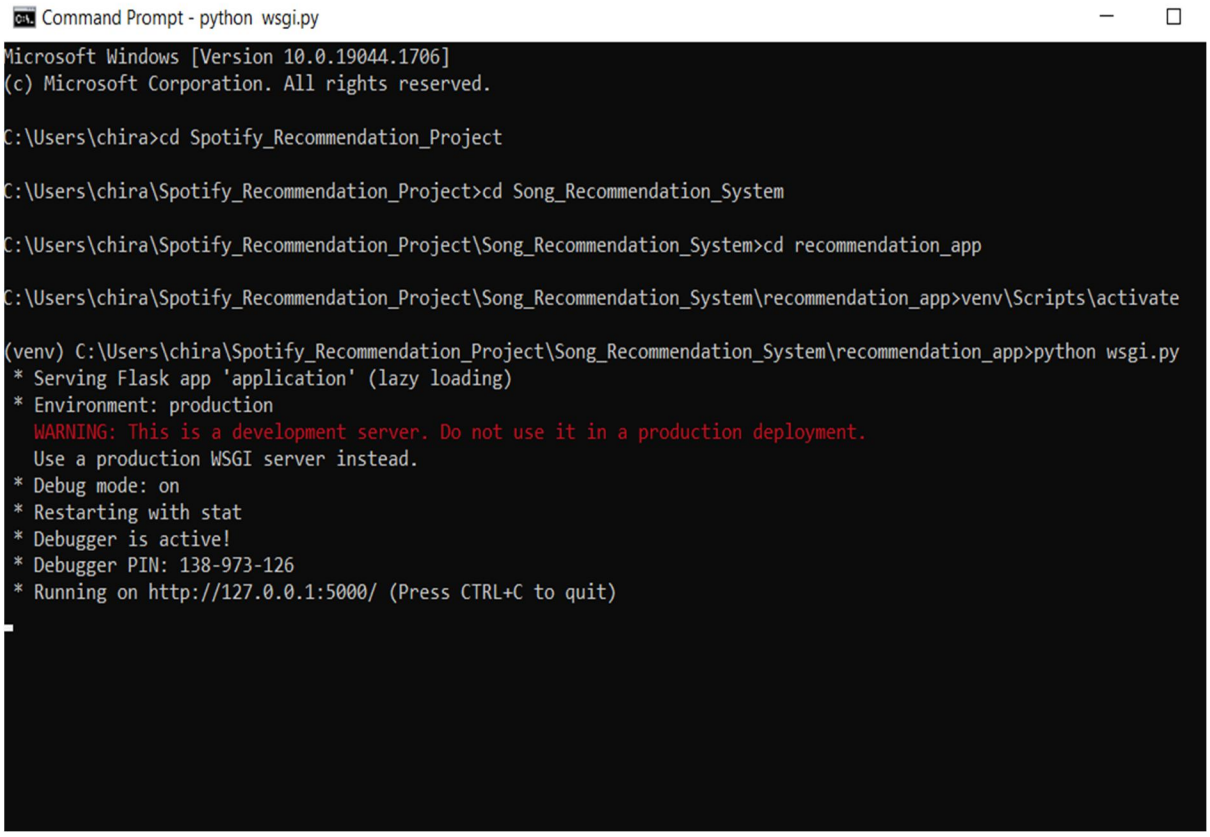


Fig 5.2 Running wsgi.py file

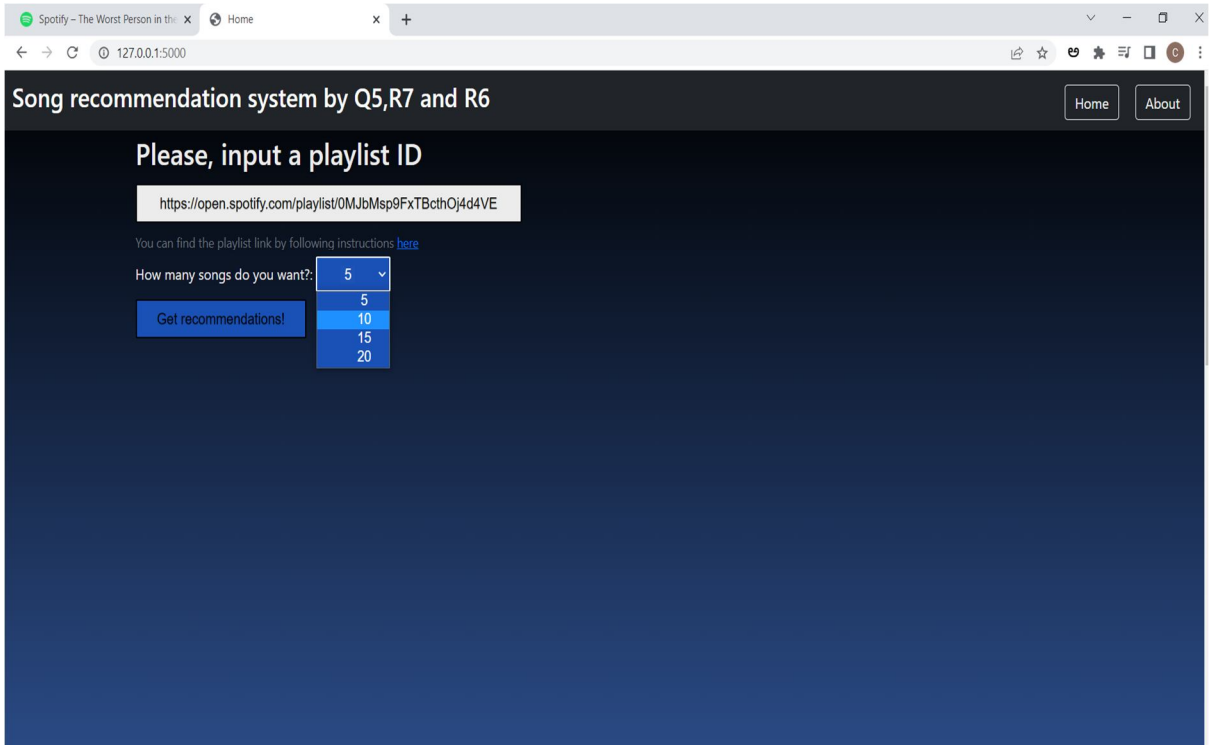


Fig 5.3 Providing public playlist URI as input

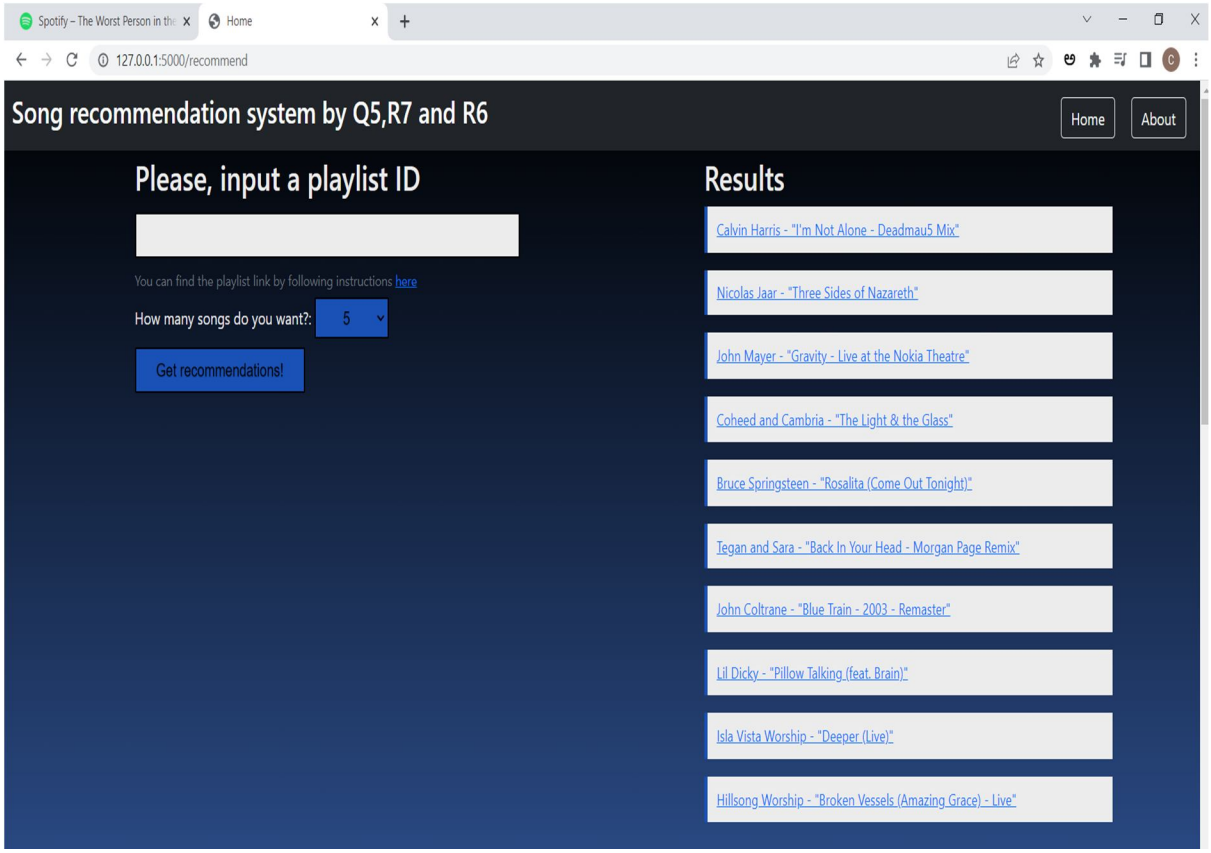


Fig 5.4 Getting recommendations

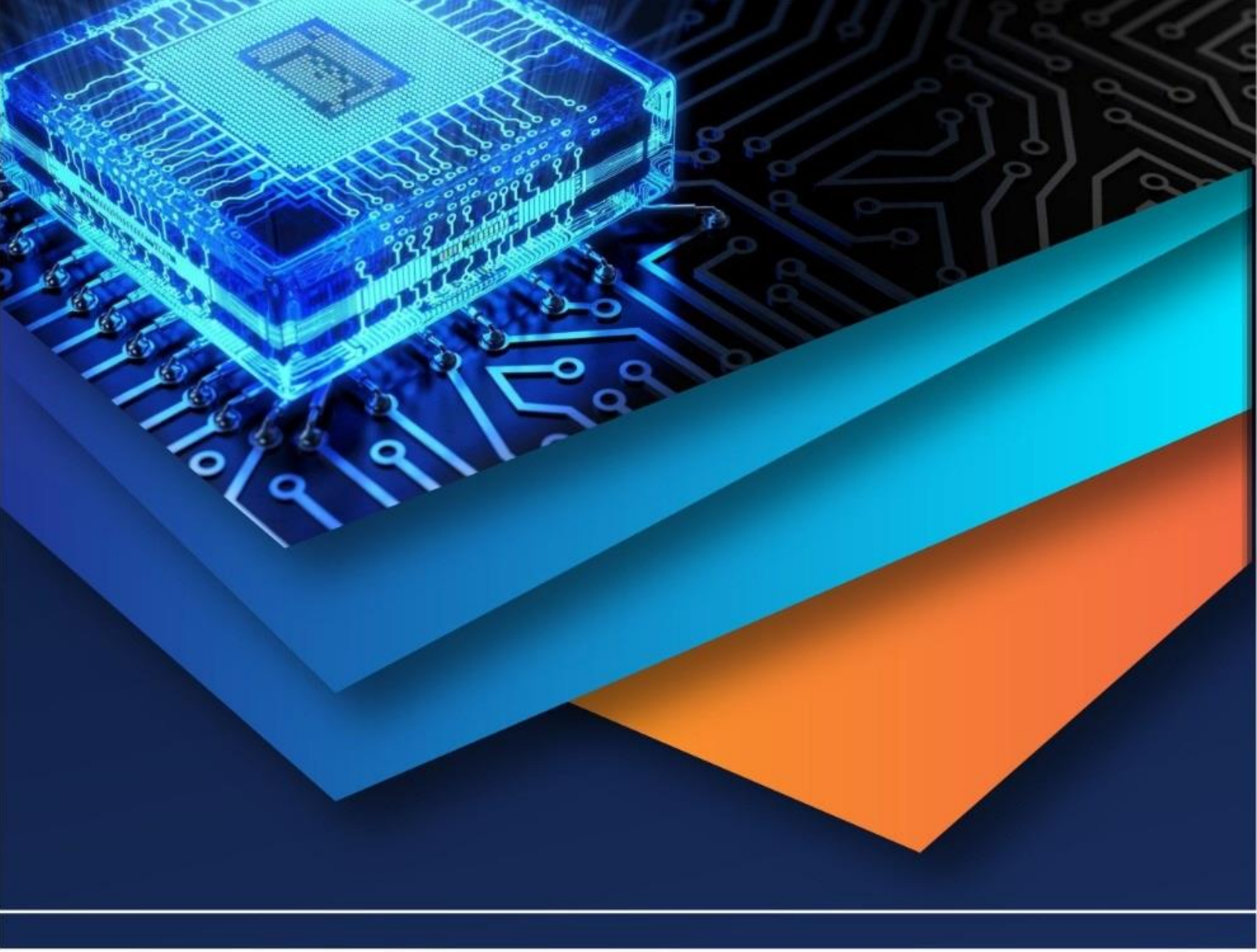
1) *Input:* https://open.spotify.com/playlist/0MJbMsp9FxTBcthOj4d4VE (The Worst Person in the World Soundtrack)
2) *Produced Output:* ["Calvin Harris I'm Not Alone - Deadmau5 Mix", 'Nicolas Jaar Three Sides of Nazareth', 'John Mayer Gravity - Live at the Nokia Theatre', 'Coheed and Cambria The Light & the Glass', 'Bruce Springsteen Rosalita (Come Out Tonight)', 'Tegan and SaraBack In Your Head - Morgan Page Remix', 'John Coltrane Blue Train - 2003 - Remaster', 'Lil Dicky Pillow Talking (feat. Brain)', 'Isla Vista Worship Deeper (Live)', 'Hillsong Worship Broken Vessels (Amazing Grace) - Live']
3) *Cosine Similarity:* [0.9999999999997357, 0.999999999999725, 0.9999999999996839, 0.9999999999996804, 0.9999999999996744, 0.9999999999996565, 0.9999999999996556, 0.9999999999996543, 0.9999999999996514, 0.9999999999996447]

VI. CONCLUSION

We have successfully built a song recommendation system that performs content-based filtering on a given playlist and implements Term Frequency- Inverse Document Frequency Vectorization on genre data and Sentimental analysis on track name of each song.

REFERENCES

[1] Spotify's monthly active users 2015-2022. Published by Marie Charlotte Götting, May 3, 2022 (https://www.statista.com/statistics/367739/spotify-global-mau/)
[2] An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation (https://arxiv.org/pdf/1810.01520.pdf)
[3] F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh, Recommendation systems: Principles, methods, and evaluation, Egyptian Informatics Journal, Volume 16, Issue 3, 2015, Pages 261-273, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2015.06.005. (https://www.sciencedirect.com/science/article/pii/S1110866515000341)
[4] Music Business News Story 1: https://musicbiz.org/news/playlists-overtake-albums-listenership-says-loop-study
[5] Music Business News Story 2: https://musicbiz.org/resources/tools/music-biz-consumer-insights/consumer-insights-portal
[6] Nielsen Report: http://nielsen.com/us/en/insights/reports/2017/music-360-2017-highlights.html
[7] MIDiA report: https://midiaresearch.com/blog/announcing-midias-state-of-the-streaming-nation-2-report
[8] Theresa Wilson, Janyce Wiebe, Paul Hoffmann; Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. Computational Linguistics 2009; 35 (3): 399–433. doi: https://doi.org/10.1162/coli.08-012-R1-06-90
[9] Computational Analysis and Understanding of Natural Languages: Principles, Methods, and Applications Venkat N. Gudivada, ... Amogh R. Gudivada, in Handbook of Statistics, 2018 (https://www.sciencedirect.com/handbook/handbook-of-statistics)
[10] Data Mining: Concepts and Techniques, A volume in The Morgan Kaufmann Series in Data Management Systems, 2.4.7 Cosine similarity.
[11] https://research.atspotify.com/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY