WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Music Recommendation System

Akash S. Verulkar[1], Adesh N. Mulik[2], Suraj A. Dukare[3], Prof. Swati S. Bharad[4]

[1,2,3,4] Dept. of Computer Science and Engineering, Anuradha Engi. College, Chikhli, SGBAU University Amravati
Email:- akashverulkar059@gmail.com[1], mulikadesh17@gmail.com[2], surajdukare20k1@gmail.com[3], sbharad215@gmail.com[4]

ABSTRACT

In our project, we will be using a sample data set of songs to find correlations between users and songs so that a new song will be recommended to them based on their previous history. We will implement this project using libraries like NumPy, Pandas.We will also be using Cosine similarity along with CountVectorizer. Along with this,a front end with flask that will show us the recommended songs when a specific song is processed.

Keywords: Pandas, Numpy, Music Recommendation System, User Interaction

I.INTRODUCTION

The history of music recommendation systems using Python and machine learning began with the emergence of platforms like Pandora and Last.fm in the late 20th century. These platforms initially relied on user preferences and collaborative filtering but did not incorporate machine learning algorithms. However, with the rise of machine learning techniques in the 2010s, music recommendation systems evolved rapidly. Python, with its extensive libraries such as scikit-learn, became a popular choice for implementing these systems. Collaborative filtering, a technique that predicts user preferences by gathering data from many users, became a fundamental approach. Python libraries like Surprise and LightFM provided tools for building collaborative filtering models. Additionally, content-based filtering, which recommends items based on their characteristics, gained prominence. Overall, the integration of Python and machine learning has propelled the development of sophisticated music recommendation systems, enhancing user experiences and personalization.

In the early 2000s, platforms like Pandora and Last.fm pioneered personalized music recommendations by utilizing user feedback and collaborative filtering techniques. However, these systems initially relied on rule-based algorithms and simple statistical methods.

The advent of machine learning, particularly in the mid-2000s, revolutionized music recommendation systems. Python, with its versatility and robust machine learning libraries, became a preferred language for building these systems. Around this time, collaborative filtering algorithms gained traction, allowing systems to analyze user interactions and preferences to generate personalized recommendations.

As the field progressed, hybrid recommendation approaches emerged, combining collaborative filtering with content-based filtering methods. Content-based filtering leverages features of items (in this case, music tracks) to recommend similar items to users based on their preferences. Python's extensive libraries facilitated the implementation of these hybrid models, enabling more accurate and diverse recommendations.

With the proliferation of streaming services like and Apple Music in the 2010s, music recommendation systems became integral to enhancing user engagement and satisfaction. These platforms invested heavily in data science and machine learning to develop sophisticated recommendation algorithms capable of understanding complex user preferences and behaviors.

Today, the field of music recommendation systems continues to advance rapidly, driven by innovations in machine learning techniques such as deep learning and reinforcement learning. Python remains a dominant language in this space, empowering developers and data scientists to create increasingly personalized and effective music recommendation experiences for users worldwide.

Music recommendation systems have revolutionized the way people discover and enjoy music in the digital age. These systems leverage advanced algorithms and machine learning techniques to analyze user preferences and music characteristics, delivering personalized playlists and song suggestions. Python, with its versatile libraries and frameworks for data science and machine learning, has become instrumental in the development of these recommendation systems.

This paper provides an overview of music recommendation systems using Python and machine learning. It explores the evolution of these systems, from early collaborative filtering methods to modern hybrid approaches that combine collaborative and content-based filtering techniques. The paper also discusses the impact of Python's rich ecosystem on the development of sophisticated recommendation algorithms, enhancing user experiences and engagement in the ever-growing digital music landscape.

In the digital era, music recommendation systems have emerged as indispensable tools for guiding listeners through the vast ocean of musical content available online. These systems employ intricate algorithms and machine learning methodologies to sift through extensive datasets of user preferences and music attributes, tailoring bespoke playlists and song recommendations to each individual user. Python, renowned for its extensive array of libraries and frameworks for data science and machine learning, has become the cornerstone of developing such recommendation systems.

This paper embarks on a journey through the realm of music recommendation systems, focusing on their development utilizing Python and machine learning techniques. It delves into the intricacies of how these systems have evolved over time, from rudimentary collaborative filtering mechanisms to the more sophisticated hybrid models that seamlessly blend collaborative and content-based filtering approaches. Additionally, it highlights the pivotal role of Python in this evolution, providing developers and data scientists with the necessary tools and resources to craft highly personalized and effective music recommendation experiences for users worldwide.

Through an exploration of the methodologies, challenges, and innovations in music recommendation systems, this paper aims to shed light on the dynamic intersection of music, technology, and data science, shaping the way we discover and appreciate music in the digital age.

II. LITERATURE REVIEW

This paper makes use of a recommender system that receives the user's profile as an input and processes it to provide the first set of recommendations based on user's mood. This is done after the first recommendation phase, which sorts the artists in the first output and performs content-based filtering of the songs, and after this phase, it provides songs that fit the user's mood. Through an interactive drop-down list, users can access the three components of the web system—input, visualisation view, and recommendation panels—using a web browser. Users enter the names of artists to create their profiles. Based on given mood data associated with profile singers, the technology places a user icon in a previously computed visual mood environment and suggests new artists [1].

In this work,they conducted research on a song recommendation system that makes music suggestions to users based on their usage patterns and song history. This model is distinctive in that it incorporates the user-produced output into its subsequent predictions. Three factors are used to determine a user's behaviour: their posts and tweets on social media apps, their likes of those posts, and their uploaded photos to social media sites. The Microsoft Azure Cloud platform is used to develop the recommendation engine. The design also makes advantage of PySpark for processing large amounts of data, and it trains the data using several machine learning techniques like naive bayes, svm, and random forest.It achieves an accuracy of around 74% [2].

It takes an extended period to sort through all of this digital music, and it makes you information-tired. Therefore, developing a system for automatically searching through music libraries and suggesting appropriate songs to users is very helpful.Using a music recommender system, a music provider can anticipate and then present the appropriate songs to their users based on the traits of the music that has already been heard. Research is being done in this paper to create a music recommendation system that can make suggestions based on the similarity of features on audio signals. In this study, similarity distance is used to determine whether two features are similar and convolutional recurrent neural networks (CRNN) are used to extract features. According to the study's findings, users favour recommendations that take music genres into account over those that are solely based on similarity [3].

Pasquale Lops, Marco American state Gemmis, andGiovanni Semeraro, 2019 [1] in their paper Content-based Recommender Systems: State of the Art andTrends discusses the most problems associated withthe illustration of things, ranging from easytechniques for representing structured information toa lot of complicated techniques returning from {theinformation|the knowledge|the information}Retrieval analysis space for unstructured data.This work is split into three components. The primaryhalf presents the essential ideas of content-basedrecommender systems, a high-level design, and theirmain blessings and disadvantages. The second half areview of the state of the art of systems adopted inmany application domains by describing each classicaland advanced technique for representing things anduser profiles. The foremost wide adopted techniquesfor learning user profiles also are conferred [4].

Sappadla et.al., in 2017, presented a system based on C and CBF using K-Nearest-Neighbour (KNN) method and experiments were performed on the Movielens dataset. The performance of user-based CF and CBF techniques is 0.885 and 0.9554 respectively, according to mean squared error (MSE), in this experiment the data type 100k was used. The user-based CF technique gives the best performance [5].

III. PROBLEM DEFINITION

The problem addressed by the music recommendation system lies in the difficulty users face in discovering new music that aligns with their tastes and preferences. With an overwhelming abundance of songs available across various genres and artists, users often struggle to navigate through the vast catalog of music to find tracks that resonate with them.

Key challenges include:

Information Overload: Users are inundated with a vast array of songs, making it challenging to identify and explore new music that suits their individual preferences.

Limited Exposure: Users may be unaware of lesser-known artists or genres that could potentially appeal to them, resulting in a narrow scope of musical exploration.

Time Consumption: Manual searching and browsing through extensive music libraries require significant time and effort, hindering efficient discovery of new music.Implies that the recommendation system uses a dataset, which may include information about songs, user preferences, and other relevant data.

The goal of the music recommendation system is to address these challenges by providing users with personalized recommendations tailored to their unique preferences and listening habits. By analyzing user behavior, preferences, and interactions with the music platform, the system aims to deliver accurate and relevant song suggestions, thereby enhancing the user's music discovery experience.

IV. PROPOSED SOLUTION

Problem solution for a music recommendation system using machine learning and Python

Data Collection: Gather user listening data, music attributes, and user feedback from sources like streaming platforms, APIs (e.g., Spotify, Last.fm), or public datasets.

Data Preprocessing: Clean the data to handle missing values, outliers, and inconsistencies. Normalize numerical features and encode categorical variables. Merge datasets if necessary (e.g., user profiles, music metadata).

Feature Engineering: Extract relevant features from the data, such as user demographics, listening history, music genres, artist information, and temporal patterns. Generate additional features if needed (e.g., popularity scores, sentiment analysis of user reviews).

Model Selection and Training: Choose appropriate recommendation algorithms based on the nature of the problem and available data (e.g., collaborative filtering, content-based filtering, hybrid methods). Split the data into training and testing sets. Train the selected models using the training data and optimize hyperparameters if necessary.

Evaluation: Evaluate the performance of the trained models using evaluation metrics such as accuracy, precision, recall, and F1-score. Use techniques like cross-validation to ensure robustness and generalization of the models.

Deployment: Deploy the trained models into a production environment, such as a web application or mobile app. Integrate the recommendation system with the user interface, ensuring seamless interaction and real-time recommendations. Monitor the system's performance and gather user feedback for further improvements.Continuous Improvement: Collect user feedback and usage data to iteratively improve the recommendation algorithms.

V.ALGORITHM

TF-IDF:

TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a technique used in natural language processing and information retrieval to understand the importance of a word in a document or a corpus (collection of documents).

Term Frequency (TF):

Term frequency measures how often a word appears in a document.

It's calculated by dividing the number of times a word appears in a document by the total number of words in that document.

For example, if the word "music" appears 5 times in a document containing 100 words, the term frequency of "music" in that document would be 5/100 = 0.05.

Inverse Document Frequency (IDF):

Inverse document frequency measures how important a word is across multiple documents in a corpus.

It's calculated by taking the logarithm of the total number of documents in the corpus divided by the number of documents containing the word, and then adding 1 to prevent division by zero.

Words that appear in many documents have a low IDF score, while words that appear in fewer documents have a higher IDF score.

TF-IDF Score:

The TF-IDF score combines the term frequency and inverse document frequency to determine the importance of a word in a specific document relative to the entire corpus.

It's calculated by multiplying the term frequency (TF) of a word in a document by its inverse document frequency (IDF).

Words with a high TF-IDF score are those that are frequent in a document but rare across other documents, indicating their importance in that document.

Let's break down TF-IDF step by step:

Step 1: Term Frequency (TF)

Count Words: First, we count how often each word appears in a document.

Calculate Frequency: Then, we calculate the frequency of each word by dividing the number of times it appears by the total number of words in the document.

Example: If the word "music" appears 5 times in a document with 100 words, the term frequency of "music" would be 5/100 = 0.05.

Step 2: Inverse Document Frequency (IDF)

Count Documents: Next, we count how many documents contain each word.

Calculate IDF: We then calculate the inverse document frequency by taking the logarithm of the total number of documents divided by the number of documents containing the word, and adding 1 to prevent division by zero.

Example: If there are 1,000 documents in the corpus and the word "music" appears in 100 of them, the IDF score of "music" would be log(1000 / 100 + 1) = log(10) ≈ 1.

Step 3: TF-IDF Score

Multiply TF and IDF: Finally, we calculate the TF-IDF score for each word by multiplying its term frequency (TF) in the document by its inverse document frequency (IDF).

Example: Using our previous examples, if the term frequency of "music" in a document is 0.05

VI.APPLICATION

User Profile Creation: Users create profiles where they can input their musical preferences, favorite genres, artists, and songs. They may also connect their accounts with existing music streaming services to import their listening history.

Content Analysis: The system analyzes the attributes of music tracks, such as genre, tempo, mood, instrumentation, and lyrics. It may also consider metadata like release date, popularity, and artist information.

Recommendation Algorithms: The application employs recommendation algorithms to generate personalized recommendations for each user. These algorithms may include:

Collaborative filtering: Recommending music based on the preferences of users with similar tastes.

Content-based filtering: Recommending music based on the characteristics of songs the user has liked in the past.

Hybrid approaches: Combining collaborative and content-based filtering techniques for more accurate recommendations.

Real-time Updates: The system continuously updates recommendations based on user interactions, such as likes, skips, and playlist creations. It adapts to changes in user preferences over time.

Integration with Music Services: The application may integrate with popular music streaming services like Spotify, Apple Music, or YouTube Music, allowing users to listen to recommended tracks directly within the app or redirecting them to their preferred streaming platform.

Social Features: The application may include social features such as sharing playlists, following other users, and seeing what friends are listening to. Social interactions can enhance the discovery experience and foster a sense of community among users.

VII.CONCLUSION

In conclusion, the implementation of a music recommendation system represents a significant advancement in the field of music technology. Through the amalgamation of machine learning algorithms, user preferences, and extensive music databases, these systems have the potential to revolutionize the way individuals discover and engage with music.

One key takeaway is the importance of data quality and diversity in training the recommendation algorithms. Robust datasets comprising various genres, artists, and user listening patterns are crucial for ensuring accurate and personalized recommendations. Additionally, the incorporation of user feedback mechanisms can further refine the recommendations over time, enhancing the overall user experience.

VIII. REFERENCES

1.LEE, jongseol, et al. ”music recommendation system based on genre distance and user preference classification.” journal of theoretical and applied information technology 96.5 (2018).

2.millecamp, martijn, et al. ”controlling spotify recommendations: effects of personal charac-teristics on music recommender user Interfaces.” proceedings of the 26th conference on user modeling, adaptation and personalization. acm, 2018.

3. o bryant, jacob. ”a survey of music recommendation and possible improvements.” (2017).

4. knees, peter, and markus schedl. ”a survey of music similarity and recommendation from music context data.” acm transactions on multimedia computing, communications, and applications (tomm) 10.1 (2013): 2.

5. ferretti, stefano. ”clustering of musical pieces through complex networks: an assessment over guitar solos.” ieee multimedia (2018).

6. song, yading, simon dixon, and marcus pearce. ”a survey of music recommendation sys-tems and future perspectives.” 9th international symposium on computer music modeling and retrieval. vol. 4. 2012.

7. deshmukh, puja, and geetanjali kale. ”a survey of music recommendation system.” (2018).

8. skowronek, janto, martin f. mckinney, and steven van de par. ”ground truth for automatic music mood classification.” ismir. 2006