PROJECT REPORT

on

CONTEXT-AWARE MUSIC RECOMMENDER SYSTEM

Course Code: ITIR-32

By

Raghav (12013114)

Under the supervision of

Dr. VIJAY VERMA

DEPARTMENT OF COMPUTER ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, KURUKSHETRA

HARYANA (INDIA)

DECLARATION

I hereby certify that the work which is being presented in this project report entitled “Context-Aware Music Recommender system”, in partial fulfillment of the requirements for the award of the Bachelor of Technology in Information Technology is an authentic record of my work carried out during a period from January 2023 to June 2023 under the supervision of Dr.Vijay Verma, Assistant Professor, Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

Signature of Candidate
Raghav(12013114)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Signature of Faculty Mentor

Dr. VIJAY VERMA

ACKNOWLEDGEMENTS

I would like to express gratitude, and respect towards my professor Dr. Vijay Verma, for guiding me and supporting me at every stage of my project work. I thank him duly for believing in my capabilities and showing me the path to progress.

I am also thankful to my peers and colleagues for explaining to me certain topics during my study which would otherwise be difficult to grasp.

I must also mention my family members whose love, and support kept me motivated to go on even when I was about to quit.

Without all the above-mentioned people I wouldn't be able to perform this experiment. I once again extend my love and gratitude towards them.

ABSTRACT

In today's digital era, personalized recommendation systems play a vital role in enhancing user experiences by suggesting content tailored to individual preferences. This report presents an in-depth analysis of a personalized music recommendation system that employs a collaborative filtering approach enriched with contextual information. The system aims to enhance music discovery and user satisfaction by suggesting tracks aligned with individual tastes.

The core principle of the recommender system lies in analyzing users' historical listening behaviors and identifying patterns to predict their potential preferences. By considering user-item interactions and leveraging collaborative filtering methods, the system generates recommendations that align with users' musical preferences, fostering a more engaging and satisfying music consumption experience. The system's architecture encompasses three key components: data collection and preprocessing, recommendation algorithm implementation, and user interaction interface. The data collection process aggregates user listening data and song attributes, while preprocessing involves data cleaning, normalization, and handling missing values to ensure data quality and usability. The recommendation algorithm employs both user-based and item-based collaborative filtering approaches to identify users with similar preferences and songs that share characteristics. By calculating similarity scores and utilizing matrix factorization techniques, the system predicts users' preferences for unrated songs, ultimately generating personalized music recommendations.

In conclusion, the system's modular architecture, encompassing data preprocessing, recommendation algorithms, and user interaction, offers a comprehensive solution for enhancing music discovery and providing users with an enriched music-listening experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML: Machine Learning

RS: Recommender System

CARS: Context-Aware Recommender System

U: Urban

M: Mountains

CL: Coastline

CS: Countryside

MAE: Mean Absolute Error

CV: Cosine Vector

Tp: True Positive

Fp: False Positive

Fn: False Negative

CHAPTER 1: RECOMMENDER SYSTEM

1.1 Introduction

A Recommender System is a type of artificial intelligence software and technique that provides suggestions for items or content that are most likely of interest to a particular user. These suggestions are often recommended in the analysis of data on a user’s preferences, behaviors, and patterns to recommend relevant items or content. These systems are used to provide personalized recommendations for products, services, movies, books, and other types of content to users based on their interests and past interactions. Recommender systems are widely used by businesses and online retailers to increase customer engagement and improve customer satisfaction. They have also been used in the fields of healthcare, education, and social networking to make personal recommendations to users. Recommender systems, in general, play a significant role in assisting users in navigating the large quantity of information available and enhancing user happiness by recommending pertinent and personalized things of interest.

Many different domains employ recommender systems to make suggestions for products or content that users might find interesting. A few illustrations of recommender systems are as follows:

- **Music recommendations:** Recommender systems are used by music streaming services like Spotify and Youtube Music to make playlists and suggest songs based on a user's listening history, favorite songs, and genre preferences. To deliver personalized music suggestions, these algorithms take into account elements including the user's favorite musicians, comparable artists, etc.
- **Movie recommendations:** Recommender systems are used by services like Netflix and Amazon Prime Video to make recommendations for films or TV series based on a user's viewing habits, ratings, and preferences. For the purpose of providing tailored recommendations, these systems examine the user's behavior in addition to the qualities of the films, such as the genre, actors, and directors.
- **Social Media Recommendations:** Recommender systems are used by social media sites like Facebook and Instagram to propose users to friends, groups, or pages based on those users' social connections, hobbies, and behaviors. To suggest suitable social connections and material for users to interact with, these systems use different recommendation techniques.

- **Travel Recommendations:** Travel companies like Booking.com use recommender systems to make hotel, location, or activity suggestions to users based on their travel choices, past travel experiences, and reviews. These systems integrate information from user evaluations, location, price, and amenities to create tailored suggestions.

1.2 Recommender Systems' Function

A recommender system's primary job is to give users personalized, pertinent recommendations. These are a recommender system's main duties:

- **Increase the number of goods sold:** A commercial RS's ability to sell a different range of goods than those that are typically sold without any form of recommendation is likely its most crucial role. The recommended things are likely to meet the user's needs and wants, therefore this purpose is satisfied. After trying multiple recommendations, it is assumed that the consumer will realize this.
- **Sell a wider variety of goods:** Another key role of an RS is to give the customer the option to choose goods that can be challenging to locate without a specific recommendation. For instance, in a tourist RS, the service provider wants to advertise all the tourist attractions in the region, not just the most well-known ones. Without an RS, this could be challenging because the service provider cannot afford to take the chance of advertising locations that are unlikely to appeal to a certain user's tastes. As a result, an RS recommends or promotes unpopular locations to the appropriate people.
- **Enhance user satisfaction:** A well-designed RS can also enhance the user's interaction with the website or application. With a well-designed human-computer interaction, the user will find the recommendations to be engaging and pertinent, and they will also like using the system. The user's subjective assessment of the system will rise when it combines useful, accurate recommendations with an intuitive user interface. The likelihood that the advice will be followed will rise as a result, increasing system usage.
- **A better comprehension of the user's goals:** The description of the user's preferences, which are either explicitly collected by the system or projected by it, is another crucial feature of an RS that

may be used in many other applications. After that, the service provider may elect to use this information again for a variety of objectives, such as enhancing the management of the product's supply or production. For instance, in the tourist industry, destination management companies may decide to market a particular area to emerging client segments or to market a specific kind of promotional message generated from an analysis of the information gathered by the RS.

1.3 Recommendation Techniques

We would want to use a taxonomy Burke [1] provided to give a brief overview of the many RS kinds. This taxonomy has evolved into a standard means of identifying and referring to recommender systems. Burke [1] makes a distinction between six sorts of recommendation methods:

1. **Content-Based:** The system learns to suggest products that are comparable to those that the consumer has previously enjoyed. The 12 features identified by F. Ricci et al. as being connected with the compared items are used to determine how similar the things are. For instance, if a user gives a song from a cultural genre a high rating, the system can learn to suggest other songs from this genre. Instead of utilizing keywords to describe the item and user profiles, semantic indexing strategies employ ideas. The authors present two major categories of semantic indexing approaches: bottom-up and top-down. While the latter group relies on a lightweight semantic representation based on the idea that a word's meaning depends on how it is used in a large corpus of textual documents, the former group relies on the integration of external knowledge sources, such as ontologies and data from the Linked Data cloud.
2. **Collaborative Filtering:** The first and most straightforward application of this strategy delivers suggestions to the active user based on goods that other users with comparable tastes have previously enjoyed. Two users' similarities in rating history are used to determine how similar their tastes are. It is believed that collaborative filtering is the most well-liked and frequently used strategy in RS. It is possible to categorize collaborative filtering into two subtypes further:
    a. **User-based Collaborative Filtering:** for example, finds people who share similar tastes and suggests products that these users have enjoyed or rated highly.

b. **Item-based Collaborative Filtering:** This technique suggests products that are comparable to those that a user has already rated or liked by using user ratings to identify similar things.

3. **Demographic:** This kind of technology makes product recommendations depending on the user's demographic profile. For various demographic niches, it is assumed that different recommendations should be produced. Numerous websites use quick and efficient demographic-based customization techniques. Users might be directed to specific websites according to their language or location, for instance. Alternatively, suggestions might be tailored to the user's age. Despite the fact that these methods are widely used in marketing literature, there hasn't been any thorough RS research on demographic systems.
4. **Knowledge-Based:** Knowledge-based systems make product recommendations based on specialized domain knowledge about how individual item attributes fit consumers' wants and preferences and, ultimately, how the item benefits the user. Case-based recommender systems are notable examples on knowledge-based systems. A similarity function in these systems calculates how closely the user's needs match the recommendations. The utility of the advice for the user can be easily inferred from the similarity score in this case. When first deployed, knowledge-based systems typically outperform other approaches. However, if they lack learning components, they risk being surpassed by less sophisticated techniques that can make use of the logs of human-computer interaction.
5. **Context-Aware:** Many current approaches to recommender systems concentrate on recommending the most pertinent products to specific consumers without taking into account any contextual information, like time, place, and the presence of other people (for example, for watching films or going out to eat). To put it another way, conventional recommender systems only work with applications that have two categories of entities: users and objects, and they do not contextualize these entities while making recommendations. So, to recommend based on the context of the situation that the user is present, a new type of technique I developed known as context-aware.
6. **Hybrid Recommender Systems:** These RSs are built using a combination of the procedures discussed above. A hybrid system that combines approaches A and B attempts to leverage the benefits of A to address the drawbacks of B. For instance, new-item issues or the inability to propose things with no ratings are two issues with CF techniques. Since the prediction of new

things is dependent on their description (features), which are often readily available, this does not restrict content-based techniques. Several methods have been suggested for combining two (or more) fundamental RSs techniques to develop a new hybrid system.

1.4 Challenges

Recommender systems play a critical role in providing personalized suggestions to users, but they also face several challenges. Some of the key challenges for recommender systems include:

- **Data Sparsity:** In many cases, the available data is sparse, meaning that not all users have interacted with all items. This makes it difficult to accurately predict user preferences for items that have limited or no interaction history.
- **Cold Start Problem:** Recommender systems struggle when dealing with new users or items that have little to no interaction data. It's challenging to provide accurate recommendations in these cases because there's not enough information to base predictions on.
- **Scalability:** As the number of users and items grows, the computational demands of generating recommendations increase significantly. Scalability is crucial to ensure that the system can handle large volumes of data and deliver recommendations in a timely manner.
- **Dynamic Preferences:** User preferences can change over time due to various factors. A good recommender system should be able to adapt to these changes and provide up-to-date recommendations.
- **Data Quality and Noise:** Noisy or incorrect data can negatively impact the performance of a recommender system. Outliers, fake reviews, and other forms of data noise can lead to inaccurate recommendations.
- **Contextual Recommendations:** Many recommendation scenarios involve contextual information, such as time, location, or social context. Incorporating these contextual factors to improve recommendations adds complexity to the system.

- **Multi-Criteria Recommendations:** Users often have multiple preferences and constraints when making decisions. Designing recommender systems that can handle multiple criteria and constraints is a challenge.
- **Cold Start for Items:** Similar to the cold start problem for users, new items also face challenges in getting initial exposure and recommendations.
- **Long-Tail Recommendations:** Recommending items from the long tail (less popular items) is important for providing diverse and personalized recommendations, but it's difficult to balance with popular items that have more interaction data.

1.5 Structure

This project report is divided into 5 sections.

Chapter 1 provides an introduction to the topic of the project and describes it.

Chapter 2 contains a brief review of the literature. It provides an overview of the recommendation technique that is used in this project.

Chapter 3 provides insight into the approach used in this project.

Chapter 4 provides an insight into the results obtained by the student and its evaluation.

Chapter 5 contains the list of all the references used for the purpose of building this project.

CHAPTER 2: CONTEXT-AWARE RECOMMENDATION SYSTEMS

2.1 Introduction

Many current approaches to recommender systems concentrate on recommending the most pertinent products to specific users without taking any context into account, such as time, place, and the presence of other people.
(for example, for dining out or attending films). To put it another way, conventional recommender systems only work with applications that have two categories of entities: users and objects, and they do not contextualize these entities while making recommendations.
Due to the increasing demand for more personalized content according to the environment of the user, the demand for the technique which recommends content according to the context is growing i.e. context-aware recommender technique. By taking into account the contextual information surrounding a user's choices, context-aware recommender systems have become an effective tool for developing personalized recommendations. To make recommendations that fit users' current circumstances, these systems take advantage of numerous variables like time, location, social context, device kind, and user behavior. The idea, guiding principles, and advantages of context-aware recommender systems are examined in this research.
Over the past 10–15 years, context-aware recommendation capabilities have been developed by academic researchers and applied in a variety of different application settings, including movie recommenders [2], restaurant recommenders [3], travel recommenders, and tourist guides [4], general music recommenders [5], specialized music recommenders (e.g., for places of interest [6], in-car music [7], or music while reading [8]), mobile information search [9], news recommenders [10], shopping assistants [11], mobile advertising [12], mobile portals [13], mobile app recommenders [14], and many others. In particular,

mobile recommender systems constitute an important special case of context-aware recommenders, where context is often defined by location and time.

One of the key advantages of context-aware recommender systems is their ability to adapt recommendations in real-time. As the user's context changes, the system can dynamically update the recommendations to reflect the new situation. This adaptability ensures that the recommendations remain relevant and aligned with the user's evolving needs.

2.2 Understanding Context in Recommender Systems

The context in recommender systems is the extra data about a user, thing, or interaction that is taken into consideration to increase the precision and relevance of recommendations. Contextual data can take many different forms and include things like the current time, location, type of device being used, weather, social connections, user preferences, and more. Recommender systems can offer recommendations that are more meaningful and personalized and that are in line with the user's requirements and present circumstances by taking these contextual aspects into account. Here are a few significant context-related elements in recommender systems:

1. **Temporal Context:** A user's tastes and demands can be influenced by time-related information, such as the day of the week, the hour of the day, the season, or certain events. For instance, a user might prefer different films on weekends than on weekdays or might be looking for suggestions for a particular holiday or occasion.
2. **Spatial Context:** The relevance of recommendations can be considerably impacted by location-based information. Recommender systems can use a user's present or previous locations to deliver recommendations that are particular to that area. The user's present location may be taken into account by a restaurant recommendation system, which may then suggest nearby dining options.
3. **Social Context:** Recommendations can be greatly influenced by social relationships and interactions. Utilizing the preferences of users in a social network or users who are similar to them, collaborative filtering algorithms produce recommendations. A recommender system might, for instance, suggest films or books based on what other users with comparable interests have enjoyed or rated highly.

4. **Device Context:** The suggestions may differ depending on the user's device. Different devices may have various interfaces, features, or screen sizes, which may affect how recommended goods are presented and if they are appropriate. Systems that are contextually aware can change their recommendations accordingly.
5. **User Behaviour:** Context can be provided through previous user interactions and behavior. Users' preferences, interests, and trends can be determined by looking at their past purchases, reviews, searches, or browsing history. Utilizing this data, recommendations can be made that are customized to the user's individual interests and previous interactions.
6. **Environmental Context:** It is possible to take into account other environmental aspects, such as the current weather or special contextual data pertaining to the suggestions domain. For instance, a travel recommendation system might use weather forecasts or travel warnings to propose appropriate locations or activities.

The accuracy, relevance, and personalization of recommendations made by recommender systems can be improved by introducing contextual information into the process. situation-aware techniques give the systems the ability to dynamically modify recommendations based on the user's current situation, improving the user experience.

2.3 Working principles of CARS (Context-Aware Recommender Systems)

- **Context Acquisition:** Acquiring pertinent contextual data is the first step in creating a recommender system that is context-aware. There are many ways to gather context, including:
  - **Sensors and APIs:** To collect information about location, time, weather, and other environmental elements, systems can make use of sensors that are built into devices or that are available from other sources. Contextual data can also be obtained using APIs offered by external services like social networks or weather services.
  - **User Interactions:** User behaviors like clicks, purchases, ratings, and searches can provide contextual indications for the system. Better personalization is made possible by the identification of user preferences and interests through an analysis of these interactions.
  - **User profiles:** user profiles provide important context for creating suggestions and may contain demographic information, preferences, historical behaviors, or explicit feedback.

- **Context Modeling:** In context-aware recommender systems, context modeling is essential since it entails processing and organizing contextual data to produce a representation that can be incorporated into the recommendation process[15]. A thorough explanation of the context modeling procedure is provided here:

  - **Data preprocessing:** Contextual data is preprocessed as the initial stage in context modeling to ensure its reliability, consistency, and compliance with the recommendation system. The following tasks may be part of this preprocessing step:
    - **Data Cleaning:** Contextual data frequently includes noise, missing numbers, or inconsistencies. To address these problems, data cleaning techniques like data imputation, outlier identification, and data validation are used.
    - **Data Transformation:** In order to be used for additional analysis, contextual data may need to be changed into a suitable format. This may entail scaling to guarantee equal weight across several contextual dimensions or normalization to place data within a standardized range.
    - **Data Fusion:** Contextual data can be gathered from numerous sources or sensors using data fusion. Data from several sources are combined and integrated into one representation using data fusion techniques. Data averaging, weighted aggregation, or more sophisticated methods like sensor fusion or data integration algorithms are examples of fusion methods.

  - **Feature Selection:** Relevant features must be chosen from the contextual data after preprocessing. The goal of feature selection is to determine the most enlightening and discriminating features that support the creation of reliable recommendations. For feature selection, a variety of methods can be used, including:
    - **Statistical Analysis:** Using statistical techniques like correlation analysis, chi-square tests, or mutual information measures, it is possible to pinpoint the characteristics that are most closely associated with a user's preferences or that have the greatest predictive accuracy.
    - **Machine Learning methods:** To assess the relevance and significance of each feature, machine learning methods can be used, such as feature importance rankings from decision trees, random forests, or support vector machines.

        - **Subject Knowledge:** Choosing contextual characteristics that are known to have a significant impact on the recommendation process can be aided by expert subject knowledge. For instance, variables like cuisine type, pricing range, or customer dietary preferences may be important in a restaurant recommendation system.

    - **Context Representation:** The chosen contextual features must be supplied in a structured style that the recommendation algorithm can easily understand[16]. The relationships between users, items, and contextual elements can be represented in various ways:
        - **Feature Vectors:** A feature vector can be used to represent contextual features, with each dimension denoting a different contextual component. To create a composite representation, these vectors can be concatenated with user and item feature vectors.
        - **Ontologies:** Ontologies provide concepts and connections in a particular domain with a formal representation. Ontologies can be used to model contextual elements, enabling semantic comprehension and context-based reasoning.
        - **Knowledge Graphs:** In a graph structure, knowledge graphs represent the connections between items and their properties. Contextual data can be visualized as nodes in a graph that are connected to people and things through contextual edges, allowing for effective context integration and reasoning.

    - **Updating Contextual Models:** Contextual models must be changed over time to take into account shifting user preferences, changing situations, or additional contextual dimensions. Iterative context modeling should be used so that the contextual representation can be improved continuously[17]. This may entail strategies like online learning, incremental updates, or retraining the model on occasion using fresh contextual data.

By employing data preprocessing, feature selection, and appropriate context representation techniques, context modeling enables context-aware recommender systems to capture and utilize contextual information effectively. This allows for more accurate and personalized recommendations that align with the user's current context and preferences.

- **Context Integration:** Context integration involves incorporating the contextual model into the recommendation algorithm. This step enables the system to adapt recommendations based on the user's current context. Several techniques can be employed for context integration:
    - **Rule-Based Approaches:** Contextual rules or heuristics can be defined to modify the recommendation process based on specific context conditions. These rules can be simple if-then statements or more complex decision trees.

    - **Bayesian Networks:** Bayesian networks are probabilistic graphical models that represent the relationships between variables. They can be used to model the dependencies between user preferences, contextual factors, and item recommendations, allowing for personalized and context-aware recommendations.
    - **Matrix Factorization:** Matrix factorization techniques, such as collaborative filtering or factorization machines, can be extended to incorporate contextual information. By factoring in the user-item-context interaction matrix, these methods can capture the complex relationships and dependencies between users, items, and contextual factors.
    - **Deep Learning Approaches:** Deep learning models, such as neural networks, can be trained to jointly learn the representations of users, items, and context. Multi-layer architectures, such as recurrent neural networks (RNNs) or transformers, can capture temporal dependencies and sequential patterns in contextual data.

- **Contextual Filtering and Ranking:** Once the contextual model is integrated, the recommender system filters and ranks the items based on both the user's preferences and the current context[18]. This step ensures that the recommendations are personalized and aligned with the user's immediate needs. Various filtering and ranking techniques can be applied, including:
    - **Content-Based Filtering**
    - **Collaborative Filtering**
    - **Hybrid Approaches**

2.4 Benefits of CARS (Context-Aware Recommender Systems)

1. **Improved Personalization:** By incorporating contextual information, these systems offer recommendations that align with the user's immediate needs and preferences, enhancing the overall personalized experience.
2. **Increased Relevance:** Context-aware recommendations consider situational factors, leading to more relevant suggestions that match the user's current requirements and circumstances.
3. **Dynamic Adaptability:** Context-aware recommender systems can adapt recommendations in real-time, accommodating changes in the user's context, preferences, and evolving circumstances.
4. **Enhanced User Satisfaction:** Providing recommendations that are timely, tailored, and aligned with the user's context leads to higher user satisfaction and engagement.
5. **Increased Serendipity:** By leveraging context, these systems can introduce unexpected and novel recommendations, promoting serendipitous discovery and user exploration.

2.5 Challenges and Future Directions

1. **Contextual Data Collection:** Efficiently acquiring and processing contextual information while respecting user privacy and data protection remains a challenge.
2. **Context Integration:** Developing robust models that effectively integrate context with traditional recommendation algorithms is an ongoing research area.
3. **Scalability:** Scaling context-aware recommender systems to handle large-scale datasets and user populations is a significant challenge.
4. **Evaluation Metrics:** Developing evaluation metrics that accurately assess the performance of context-aware recommender systems is crucial for further advancement.

2.6 Conclusion

By taking into account the contextual aspects surrounding users, contextually aware recommender systems present a promising way to improve personalized recommendations. These systems can offer more pertinent, timely, and personalized suggestions by incorporating context, which boosts user satisfaction and engagement. Although there are obstacles, it is anticipated that ongoing research and technology developments will propel the development of context-aware recommender systems and determine the direction of personalized recommendation services.

CHAPTER 3: APPROACH

The recommender module utilizes collaborative filtering techniques to generate recommendations based on user preferences and contextual information. Overall, this project is divided into four code modules Interface, preprocessor, recommender, and evaluation. Together, these four code modules provide an interface for the user, preprocess input, provide recommendations, and assess the effectiveness of a recommendation system. The usefulness and accuracy of the recommendation system might be thoroughly assessed with proper implementation and integration of these modules[19]. The interface module presents an interactive system so that the user can interact with the system and get recommendations for their IDs.The data preprocessing module handles data extraction, duplication handling, and missing value imputation. The evaluation module measures the performance of the recommendation system using metrics such as mean absolute error (MAE), precision, and recall. It compares predicted ratings with true ratings and evaluates the accuracy of the recommendations and the system's ability to predict user behavior.

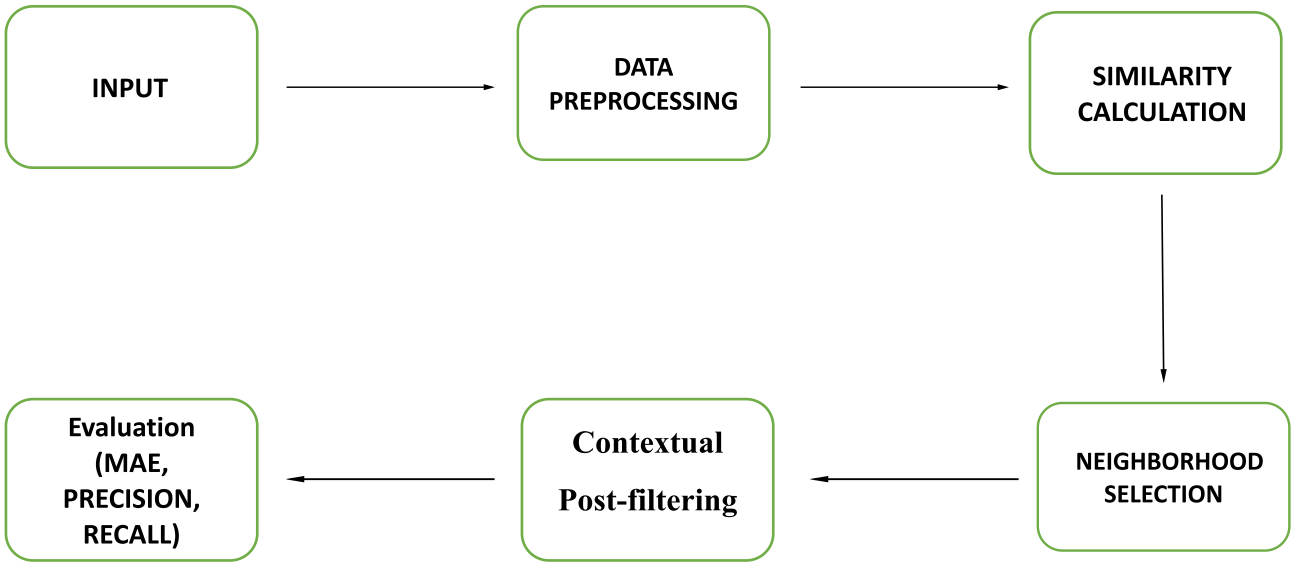

Fig 1: Proposed Approach

3.1 Data Preprocessing

Data preprocessing, which is necessary for handling situations in which a user has numerous ratings for the same item, is the first thing the code does. The primary data and song data are read from dataset files and prepared for further analysis. Additionally, it averages numerous ratings for the same object, deals with duplicate ratings, and creates random landscape contexts for missing values. The preprocess function creates a new data frame called processed_main_dataframe by averaging the ratings for duplicate items. To enable collaborative filtering, this phase ensures that each user-item combination has a single rating. The code begins by importing necessary libraries and modules, including pandas for data manipulation and sklearn for model evaluation.

The "Preprocessor.py" module offers tools for the recommendation system's data preprocessing. The input data handling, information extraction, and preparation for future analysis are the main focuses of the code. The following is the module's main duties:

- fetch_data(): This function retrieves the main data and returns the processed main data frame, user ID list, and item ID list.
- get_user_id_list(data frame): It extracts the unique user IDs from the data frame and returns them as a list.
- get_item_id_list(data frame): This function retrieves the item IDs from the song data frame and returns them as a list.
- get_user_ratings_preprocessing(user_id, filtered_main_dataframe): It retrieves the ratings of a specific user from the filtered main data frame.
- preprocess(): This function performs data preprocessing tasks such as handling duplicate ratings, averaging multiple ratings for an item, and generating random landscape contexts for missing values.
- generate_random_contexts(): It generates random landscape contexts for data points that do not have a specified context.

3.2 Recommendation Generation

As the techniques used in this project are collaborative and contextual information of the user so both techniques are used to generate the recommendations for the user.

3.2.1 Similarity Calculation

The calculation of user similarity is the foundation of collaborative filtering. The cosine similarity between the target user and every other user in the dataset is calculated using the compute_similarities function. It computes the cosine similarity based on user ratings for related items and compares those values. The similarity_dict contains the resultant similarity scores.

The Cosine Vector (CV) (or Vector Space) similarity [20, 21, 22] between two objects a and b, which is a common measure of similarity used in information retrieval, is calculated between these two vectors.
such vectors:

$$cos\left(x_a, x_b\right) = \frac{x_a x_b}{\left\|x_a\right\|\left\|x_b\right\|}$$

In the context of item recommendation, this measure can compute user similarities by considering a user u as a vector $x_u \in R^{|I|}$, where $x_{ui} == r_{ui}$ if user u has rated item i, and 0 otherwise. The similarity between two users u and v would then be computed as

$$CV(u, v) = cos\left(x_u, x_v\right) = \frac{\sum_{1 \in J_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in J_u} r_{U_i}^2 \sum_{j \in J_v} r_{v_j}^2}}$$

where I_{uv} once more denotes the items rated by both u and v. A problem with this measure is that it does not consider the differences in the mean and variance of the ratings made by users u and v.

- The compute_similarities function calculates the cosine similarity between the target user and all other users in the dataset.
- It iterates over each user and compares their ratings for common items.
- The cosine similarity is computed based on the ratings given by users to the same items.

The code determines the degree of similarity and selects the neighbors who are most similar to the target user or item by computing the similarity between users or items. As a result, the recommendation system can use the preferences of users or products that are comparable to its own to produce precise and individualized recommendations.

3.2.2 Neighborhood Selection

The number of nearest neighbors to choose from and the standards by which they are chosen can significantly affect how well the recommender system performs. The get_user_neighbourhood function chooses the N users with the highest similarity scores as the most similar users. Only a list of the N nearest neighbors and each one's individual similarity weight is stored for each user or item. N should be carefully chosen to prevent issues with effectiveness or accuracy. As a result, if N is too big, it will take a lot of memory to keep the neighborhood lists, and forecasting ratings will be laborious. On the other hand, choosing a number for N that is too low may result in the recommendation method's coverage being reduced, which prevents some items from ever being recommended. User IDs serve as keys in a neighborhood dictionary, while similarity scores serve as values. The recommendations are created based on this neighborhood. The prediction of new ratings is typically made with the k-nearest-neighbors, that is, the k neighbors whose similarity weight has the greatest magnitude, once a list of possible neighbors has been computed for each user or item. The system's accuracy and performance can both be significantly impacted by the choice of k[23].

To implement user-based collaborative filtering, the code locates a group of comparable individuals near each target user. Using a similarity metric like cosine similarity or Pearson correlation coefficient determines how similar users are to one another. The neighborhood for the target user is then determined by the code by choosing a subset of individuals with the highest similarity scores.

For item-based collaborative filtering, a neighborhood of comparable things is identified by the code for each target item. Based on the user ratings, it determines how similar the two goods are. Once more, a

similarity metric is used to assess how similar two items are, and the algorithm chooses the neighborhood for the target item from among the subset of items with the highest similarity scores.
The code creates suggestions based on the ratings of the users and products in the neighborhood after the neighborhoods have been built. It forecasts ratings for products that the target user still needs to give them and suggests products with the highest forecasted ratings.

The algorithm uses neighborhood selection to provide precise recommendations by drawing on the preferences of comparable users or objects. To make personalized and pertinent music recommendations to the target user, this method considers the local community's collective wisdom.

3.2.3 Contextual Post-filtering

Contextual post-filtering is incorporated into the code as a component of the music recommendation system. The practice of modifying the recommendations based on extra contextual data about the user's surroundings or preferences is referred to as contextual post-filtering. In this instance, the code considers the user's landscape environment (such as urban, mountainous, rural, or coastal) while making recommendations[24].

The data is first preprocessed by the code, which deals with duplicate ratings and creates random landscape contexts for entries without any. The data is then divided into training and testing sets.
By determining how similar users and objects are, the code uses collaborative filtering techniques to generate recommendations. It uses an algorithm to forecast the user's ratings for things that haven't yet been reviewed, taking into account the user's previous ratings. These anticipated ratings are used to construct the recommendations.
Contextual post-filtering then enters the picture. According to the user's existing landscape context, the algorithm filters the recommendations. It determines whether the suggested things are appropriate for the particular landscape environment and then adjusts the recommendations. This makes sure that the suggested music fits the user's surroundings and improves their general ability to enjoy music while driving.

3.2.4 Evaluation

In conclusion, the algorithm blends collaborative filtering with contextual post-filtering to suggest music to users based on their past tastes and the environment they are now in. The code seeks to offer unique and pertinent music suggestions that are appropriate for the user's driving situation by taking context into account. The code also offers evaluation measures for measuring the effectiveness of the recommendation system, such as Mean Absolute Error (MAE), precision, and recall. Precision and recall quantify how accurately the recommendations were made in terms of whether they were followed or not, whereas MAE gauges the average absolute difference between the expected and actual scores.

CHAPTER 4: RESULTS AND EXPERIMENTAL SETUP

4.1 Experimental Setup

This experiment was performed on the corrected datasets present in the following link. The datasets can be obtained from this link: dataset

The code imports libraries in Python 3.9.16 such as pandas, numpy, math, scipy, and collections . Visual Studio Code IDE was used for the purpose of building this project. A computer with 8GB RAM, a Ryzen 5 processor, and Windows 11 was used for the project. Hyperparameters were manually adjusted to give the best results**.**

4.2 Results

This code module recommends music based on the context of the location that the user is present so first, when the
program is loaded the interface of the code starts and asks for the id of the user. For the sake of simplicity and fast calculation, only 42 valid IDs are present in the database and the recommendations are generated for those IDs (1001 to 1042). If a user tries to use an invalid Id other than these the program will ask you to enter the ID again.
Now after this, the program goes ahead and asks for the context of the user that is in this case the location of the user (u for urban, m for mountains terrain, cs for countryside, cl for coastline). After the user enters its context or location the program recommends music according to their previous ratings for the songs and their contexts. A few examples of the same are presented here:

In Table (1) the input is 1002 and recommendations are generated for a context which is a location that is cl (coastline).

ItemID	Song Title	Artist
733	Niemeyer	Herbert Gr&ouml
254	Brandenburg Concerto 3	Bach
258	The Devil Went Down to Georgia	Charlie Daniels Band
265	I Will Survive	Gloria Gaynor
266	Funkytown	Lipps Inc

Table 1: Results for user 1002 (Coastline)

In Table (2) the input is 1002 but this time recommendations are generated for a different context which is a location that is m (mountains).

ItemID	Song Title	Artist
248	The Thrill is Gone	B.B.King
252	Stormy Monday	T-Bone Walker
253	Four Seasons	Antonio Vivaldi
254	Brandenburg Concerto 3	Bach
255	Symphony 5	Beethoven

Table 2: Results for user 1002 (Mountains)

In Table (3) the input is 1042 and recommendations are generated for a context which is a location that is u (Urban).

ItemID	Song Title	Artist
251	Hellhound On My Trail	Robert Johnson
248	The Thrill is Gone	B.B.King
760	Superman Tonight - Album Version	Bon Jovi
250	Crazy Blues	Mamie Smith
679	Replay (Album Version)	Iyaz

Table 3: Results for user 1042 (Urban)

In Table (4) the input is 1042 but this time recommendations are generated for a different context which is a location that is cs (Countryside).

ItemID	Song Title	Artist
251	Hellhound On My Trail	Robert Johnson
253	Four Seasons	Antonio Vivaldi
746	Foundations - Full Explicit Version	Kate Nash
269	Gangsta Paradise	Coolio
263	Stayin Alive	Bee Gees

Table 4: Results for user 1042 (Countryside)

4.3 Evaluation and Discussion

The code calculates various metrics to assess the performance of a recommendation system, including mean absolute error, precision, and recall. These metrics are commonly used to evaluate the quality of recommendations and how well the system predicts user preferences. The code is designed to work with a user-item rating dataset, where users provide ratings to different items in different contexts (urban, mountains, countryside, coastline).
The precision and recall calculations are based on binary ratings, where ratings above a certain threshold are considered positive and others are considered negative.

UserID	Context	Mean Absolute Error (MAE)	Precision	Recall
1002	Coastline	2.155716354989188	0.5	0
1002	Mountains	2.270059293135340	1.0	0.25
1042	Urban	1.991795746469462	0.2	0.5
1042	Countryside	2.00380200934858	0.1	0.8

Table 5: Evaluation of recommendation generated for different users

Here are the formulas used to evaluate Mean Absolute Error (MAE), Precision, and Recall:

4.3.1 Mean Absolute Error (MAE): It is a commonly used metric to measure the accuracy of a predictive model or system, including recommendation systems. It quantifies how far off the predictions are from the actual values on average. In the context of recommendation systems, MAE indicates how well the system's predicted ratings match the actual ratings given by users.

This function takes several arguments, including the main data frame, parameters for generating recommendations (R, N, threshold), and user and item lists. It calculates the MAE between the predicted ratings generated by your recommendation system and the true ratings from the test dataset.

The MAE in code is calculated between the predicted ratings and the true ratings for a set of recommendations. The formula remains the same as the standard MAE formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|$$

In the context of code:

- N is the total number of predictions
- y_i is the true rating of *i-th* prediction
- $\hat{y}_i$ is the predicted rating of *i-th* prediction

4.3.2 Precision: It is used to calculate the Precision of your recommendation system's predictions. Specifically, it evaluates whether the recommendations made by the system are accurate in terms of positive predictions (items that the system predicts the user will like).

For each recommended item that matches an item actually rated by the test user, the function tracks whether the prediction was accurate. It counts True Positives (TPs) when the recommendation was correct, False Positives (FPs) when the recommendation was incorrect, but the system predicted it as relevant, and False Negatives (FNs) when the recommendation was relevant, but the system didn't predict it as such.

the function calculates Precision using the formula:

$$Precison = \frac{True\ Positives}{True\ Posiives + False\ Positives}$$

This calculates the proportion of correct positive predictions (True Positives) out of all predicted positive recommendations (True Positives + False Positives).

In essence, the Precision calculation in your code assesses how accurate your recommendation system is in predicting relevant items. It's particularly useful when you want to understand how reliable the positive recommendations made by your system are.

4.3.3 Recall: It measures the system's ability to identify all relevant items from the dataset, indicating whether the system is good at capturing items that the user would actually like.

For each recommended item that matches an item actually rated by the test user, the function tracks whether the prediction was accurate. It counts True Positives (TPs) when the recommendation was correct, False Positives (FPs) when the recommendation was incorrect but the system predicted it as relevant, and False Negatives (FNs) when the recommendation was relevant, but the system didn't predict it as such.

the function calculates Recall using the formula:

$$Recall = \frac{True\ Positives}{True\ Posiives + False\ Negatives}$$

This calculates the proportion of correct positive predictions (True Positives) out of all predicted positive recommendations (True Positives + False Positives).
In summary, the Recall calculation in your code assesses the system's ability to capture all relevant items in the dataset. It's especially important when you want to ensure that your recommendation system is not missing out on items that users would find interesting. A higher Recall value indicates that the system is better at suggesting relevant items to users.

REFERENCES

1. Burke, R.: Hybrid web recommender systems. In: The Adaptive Web, pp. 377–408. Springer Berlin / Heidelberg (2007)
2. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS) 23(1), 103–145 (2005)
3. Mahmood, T., Ricci, F., Venturini, A.: Improving recommendation effectiveness: Adapting a dialogue strategy in online travel planning. J. of IT & Tourism 11(4), 285–302 (2009).
4. Park, H.S., Yoo, J.O., Cho, S.B.: A context-aware music recommendation system using fuzzy Bayesian networks with utility theory. In: Proceedings of the Third International Conference on Fuzzy Systems and Knowledge Discovery, FSKD'06, pp. 970–979. Springer-Verlag, Berlin, Heidelberg (2006).
5. Braunhofer, M., Kaminskas, M., Ricci, F.: Recommending music for places of interest in a mobile travel guide. In: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, pp. 253–256. ACM, New York, NY, USA (2011).
6. Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lke, K.H., Schwaiger, R.: Incarmusic: Context-aware music recommendations in a car. In: C. Huemer, T. Setzer (eds.) E-Commerce and Web Technologies, Lecture Notes in Business Information Processing, vol. 85, pp. 89–100. Springer Berlin Heidelberg (2011).
7. ai, R., Zhang, C., Wang, C., Zhang, L., Ma, W.Y.: Musicsense: Contextual music recommendation using emotional allocation modeling. In: Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA '07, pp. 553–556. ACM, New York, NY, USA (2007).
8. Church, K., Smyth, B., Cotter, P., Bradley, K.: Mobile information access: A study of emerging search behavior on the mobile internet. ACM Trans. Web 1(1) (2007).
9. Lee, H., Park, S.J.: Moners: A news recommender for the mobile web. Expert Systems with Applications 32(1), 143–150 (2007).
10. Sae-Ueng, S., Pinyapong, S., Ogino, A., Kato, T.: Personalized shopping assistance service at ubiquitous shop space. In: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops, AINAW '08, pp. 838–843. IEEE Computer Society, Washington, DC, USA (2008).

11. Bulander, R., Decker, M., Schiefer, G., Kolmel, B.: Comparison of different approaches for mobile advertising. In: Proceedings of the Second IEEE International Workshop on Mobile Commerce and Services, WMCS ’05, pp. 174–182. IEEE Computer Society, Washington, DC, USA (2005).
12. Smyth, B., Cotter, P.: Mp3 mobile portals, profiles and personalization. In: Web Dynamics, pp. 411–433. Springer Berlin Heidelberg (2004).
13. Karatzoglou, A., Baltrunas, L., Church, K., Böhmer, M.: Climbing the app wall: Enabling mobile app discovery through context-aware recommendations. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM ’12, pp. 2527–2530. ACM, New York, NY, USA (2012).
14. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM 35(12), 61–70 (1992)
15. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2010)
16. Park, D.H., Kim, H.K., Choi, I.Y., Kim, J.K.: A literature review and classification of recommender systems research. Expert Systems with Applications 39(11), 10,059–10,072 (2012)
17. Sinha, R.R., Swearingen, K.: Comparing recommendations made by online systems and friends. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries (2001)
18. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: C. Cattuto, G. Ruffo, F. Menczer (eds.) Hypertext, pp. 73–82. ACM (2009)
19. Schwartz, B.: The Paradox of Choice. ECCO, New York (2004)
20. Balabanovic, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Communi- ´ cations of the ACM 40(3), 66–72 (1997)
21. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Modeling and UserAdapted Interaction 10(2–3), 147–180 (2000)
22. Lang, K.: News Weeder: Learning to filter netnews. In: Proc. of the 12th Int. Conf. on Machine Learning, pp. 331–339. Morgan Kaufmann Publishers Inc.: San Mateo, CA, USA (1995)
23. Taghipour, N., Kardan, A., Ghidary, S.S.: Usage-based web recommendations: a reinforcement learning approach. In: Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys 2007, Minneapolis, MN, USA, October 19–20, 2007, pp. 113–120 (2007)
24. Verbert, K., Drachsler, H., Manouselis, N., Wolpers, M., Vuorikari, R., Duval, E.: Datasetdriven research for improving recommender systems for learning. In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge, LAK ’11, pp. 44–53. ACM, New York, NY, USA (2011).