AIE425 INTELLIGENT RECOMMENDER SYSTEMS, FALL SEMESTER 24/25 Assignment #1: Neighbourhood CF Models (User, Item-Based CF)

Mohamed Ibrahim Fekry A20000726

**1. Introduction:**

In the modern world, recommender systems are ubiquitous in almost all such online applications that assist to retrieve new content based on interests, and previous user activity. They improve the quality of the application since paraphrasing good rated content will cause increase in user’s engagement and activity in the application. The present report deals with the implementation of a movie recommendation system built in Python and utilizing user based collaborative filtering technique which today is one of the prominent recommendation systems techniques.

User based collaborative filtering is based on the idea that people who rated or interacted with a set of items would be similar to other people who have a different set of items rated or interacted with. In this instance, the system finds users who have similar tastes in films and suggests films enjoyed by those users. This method draws on the power of crowdsourcing.

It is notable that the data for the system is retrieved using an API offered by TMDb, which stands for The Movie Database, the latter being a considerable source of information regarding movies such as its respective rating, genre and other metadata also. The recommendation system aims at modeling and predicting real-life user behavior dynamics and as a result applicable suspicions, which cause the system to mimic user rating behavior. Namely, the so called user-item matrix is created where each cell represents a user’s rating towards a certain film.

The focus of this project however is to look into how well user based collaborative filtering can be used to come up with effective movie recommendations. This in turn helps the system in finding films those particular individuals who are most likely to appreciate the content enhancing the viewing experience.

**Coverage of the Following Main Points**

The development and significance of recommender systems over the years.

A description of collaborative filtering methods and techniques (user based versus item based).

Where TMDb API fits in this scope of work that is what does it do in the course of this work.

**2. Overview of Coding Techniques:**

The coding technique employs several components of which are structured. Each of these stages works toward the overall process of development and implementation of the recommendation system.

**2.1 Establishing the Environment**

The first step in building any software application is to prepare the environment which consists of the setting up of dependencies and frameworks.

Libraries Used:

Requests: This is an important extension to make HTTP requests to any API service such as TMDb to populate the code with retrieved information or attained answers.

Pandas: This is another powerful library for data manipulation designed explicitly for data analysis, offering data containers like DataFrames that are suitable for storing data in an organized way.

Numpy: Numpy is a Library that is packaged with important numerical capabilities.The library provides multi dimensional array objects and a number of mathematical functions in support of data and computations.

Scikit-learn: This library is mainly designed for machine learning applications. It also provides utility functions for data mining and analysis which improves implementations of cosine similarity in collaborative filtering.

Matplotlib: This library is used for visualizing information and ideas. It supports the designing of charts and figures which helps in capturing information clearly and efficiently.

APIs Keys: It is also vital that the codes above be used with a TMDb unique identity which allows access to their film database. This code should be kept private and not availed publicly to avoid misuse.

**2.2 Acquiring and Preprocessing Data**

In this segment, data from the TMDb API will be utilized to create a database where movies can be recommended.

How to Get Most Popular Movies:

functions fetch\_popular\_movies and fetch\_all\_movies are two functions that obtain movie related data. The first one gets the movies available at any one page while the second acquires data contained in several pages in order to create a complete list of movies available.

The API will respond with JSON which will basically have a collection of movies in it. Each of the objects containing the movies has certain properties where some of the properties are the name of the movie, when it was produced out and the rates for the movie. This information is then collected and arranged in a way suitable for analysis.

Pandas Data Frame Creation:

The imported data is converted into a numpy array and stored in a Pandas DataFrame for ease of processing. A Data Frame is useful in handling data. It allows operations of that include filtering, grouping and also pivoting to be done..

**2.3 Creating User Ratings**

Because actual user information is generally lacking, the code emulates the User Engagement by generating random User Ratings for the Movie Database.

Generational Rating:

User ratings are randomly generated for a given number of users and movies. Each user is given a rating of 1 to 5 for the set of available movies. This aids in building a user-item matrix which resembles actual user behavior.

Why Simulation:

User rating simulation is an important aspect of designing and evaluating the recommendation algorithm. It allows one to build the required data structures and investigate user preferences and user’s relations without the need of any real users, which in many cases, is not possible.

**2.4 User-Based Collaborative Filtering**

As soon as the user-item matrix is achieved, the following phase concerns the determination of the like users in terms of their rating patterns.

The User-Item Matrix:

The table is built with the help of the Pandas pivot\_table function – this arranges the users in the rows and the films in the columns. Every cell in the matrix contains the rating of a particular movie by a particular user. In cases where no rating has been provided by the user, the cell is filled with zero signifying that the movie has not been rated by the user.

Cosine Similarity:

The cosine similarity metric is used to estimate the distance between users. This Distance measures the dissimilarity of two users with respect to movie ratings given by him/her on a scale of 0 to 1, where 0 is the furthest apart and 1 is identical ratings.

The sine of the angle between the two vectors (user ratings) is known as the cosine of the angle between them. The greater the value of the cosine, the closer are the tastes of users.

**2.5 Recommendation Functions**

This is the core of the recommender system in which the algorithm provides movie suggestions tailored to the users.

Logic Behind The Recommendations:

The function get\_recommendations seeks out users which are alike the user in question, and makes use of their movie ratings to make a recommendation. We define it as the average rating of this user and n similar users weighted by their similarity.

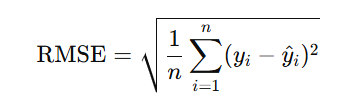
More on Recommendations:

Even more complex, get\_recommendations\_with\_details is the function that is used in the dataset to give not only the recommendations on books but also how those recommendations have come about. Thus, it explains how past users’ contributions are weighted leading to the formulation of the final recommendations of such movies.

**3. Contemplation and Assessment:**

**3.1 Root Mean Square Error (RMSE)**

* **Definition:** The accuracy of a predictive model can be assessed using RMSE, A metric is used without any engagement on the use of the product. It measures the average gap between predicted ratings and actual ratings thus illustrates the degree of correlation between the actual user preferences and the model predictions.
* **Mathematical formula:**

****

* Where:
  + yi= actual rating
  + y^i​ = predicted rating
  + n = number of ratings

**3.2 How Cosine Similarity Fits In:**

Concept: Cosine similarity examines the angle between two vectors (users or items in this case) in a multidimensional space. Its range is limited to [-1, 1]:

1: Means two vectors are defined as one & are perfectly similar.

0: Means the vectors are orthogonal to one another (no relationship).

Usage in Collaborative filtering: Along with other metrics, Cosine similarity serves as a foundation for the assessment of user and item relationships based on the patterns of their egoistic ratings and building recommendations..

**Integrating Cosine Similarity in User-Based Collaborative Filtering:**

**Estimation of Similarity:**

To assess the similarities between users, the model uses the rating patterns of the users and calculates a cosine similarity that determines how much of a user’s rating will be used to predict another user’s rating.

**Making a Recommendation:**

Following the computation of cosine similarities, the next stage is to calculate individual’s ratings for the target user. Accordingly, if a film has n ed ratings, the prediction can usually be found as the weighted average of the ratings of other users whom the subject resembles.

Execution of Project:

Normalization: Each rating vector of a user is first normalized before cosine similarity is calculated so that there is equal basis for comparison.

**3.3 Incorporating Cosine Similarity in Item-Based Collaborative Filtering**

Step 1: User-Item Matrix Construction

A user-item interaction matrix is created in which users are represented by the rows and a set of movies by the columns with the interaction (ratings) data by filling the cells.

Step 2: Computation of the Cosine Similarity Matrix

The cosine similarity matrix is calculated using the transpose of the user-item matrix that is interconnected with the rows (users) in order to establish a similarity measure in respect of the columns (movies).

Recommendation Process:

Collect user ratings for every user.

For all movies that have not yet been rated, predict their scores by calculating the weighted average of the scores of the similar rated films with the cosine similarity scores of the films being used as weights.

Ranking and Recommending:

The system orders the predicted scores of the unrated movies and recommends the movies with the highest predicted scores to the user..

**3.4 Collaborative Filtering Based On Users Pearson Correlation integration**

Definition: Pearson correlation is considered to be a linear relationship between two given variables, in this case, users Ratings in a collaborative filtering scenario.

Implementation:

The code evaluates a similarity matrix using np.corrcoef function in order to calculate correlation coefficient between users on the basis of user ratings.

Recommendation Process

Step 1: Load the Dataset:

This section shows the importation of user movie ratings data from an MS Excel work book into panda data frame.

Step 2: Create User-Movie Matrix:

A pivot table is designed with users as rows and movies as columns. All the missing ratings are filled with zeros to ease up the calculations.

Step 3: Split the Dataset:

The dataset is split into two parts; a training set (where similarities are computed) and a test set (where the model's predictions are assessed).Including Cosine Similarity in Item **Based**

**3.5 Item-based Collaborative Filtering With focus on Pearson Correlation**

Stage 1: Acquire the data

Action: To start with, a user movie ratings database must be imported. This contains details on what users google which movie and the rating attributed to that particular film.

Stage 2: Generate the user-movie Matrix

Action: Modify the data into a user-item matrix. In this case, rows indicate users while columns indicate movies. The matrix is filled with the users’ rating on the respective movies. This way, the system can study the relation between users and their rated movies.

Stage 3: Manage the missing data

Action: Fill in the user movie matrix with the missing ratings where applicable. This could be achieved by assigning for example a zero for that specific movie under that user’s rating column – which means the rated movie is not applicable to that user. This is a critical step due to the nature of the system such as collaborative filtering that cannot work efficiently in the presence of missing data.

Stage 4: Calculate the Item-Item Correlation Matrix

Action: Determine how similar movies are based on user ratings. This entails the development of item-item correlation matrix that expresses the relationship of movie ratings from the users and how alike one movie is to another in the particular scope. The outcome is a structure most of whose entries represent how close two particular movies are.

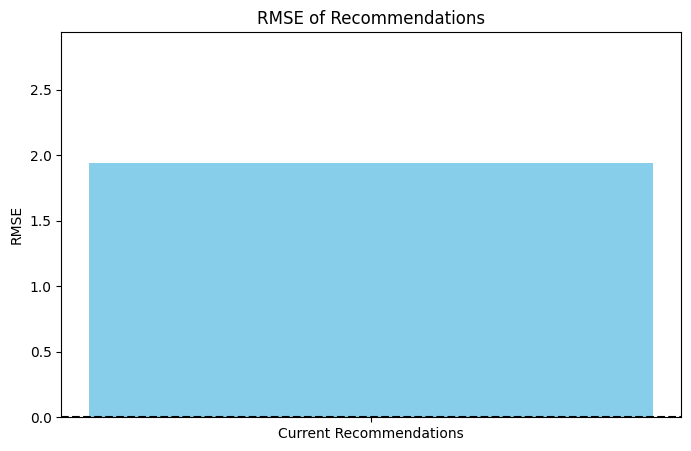
Step 5: Make Predictions Based on Similarity Action: Develop a method to predict ratings for movies that a user has not yet rated. This involves: For each unrated movie, identify similar movies based on the correlation matrix. Compute a predicted rating for the unrated movie by considering the ratings given by the user to similar movies, weighted by the similarity scores. The idea is that if a user liked movies that are similar to an unrated movie, they might also like the unrated movie.

Step 6: Generate Recommendations Action: Based on the ratings predicted in the last step, prepare lists of recommended movies for every user. The recommendations are usually provided in a ranked order or rated list. Thus, it carries the highest predicted rating with respect to the movies which the user is likely to enjoy but is yet to watch.

Step 7: Evaluate the Recommendations Action: Analyze the performance of the recommendation system. It is done through numerical values like RMSE Root Mean Square Error or MAE Mean Absolute Error which shows how closely the estimates provided by the system match the ratings given by actual users. This is important as these metrics will help in improving the recommendation algorithm by correcting its strengths and weaknesses.

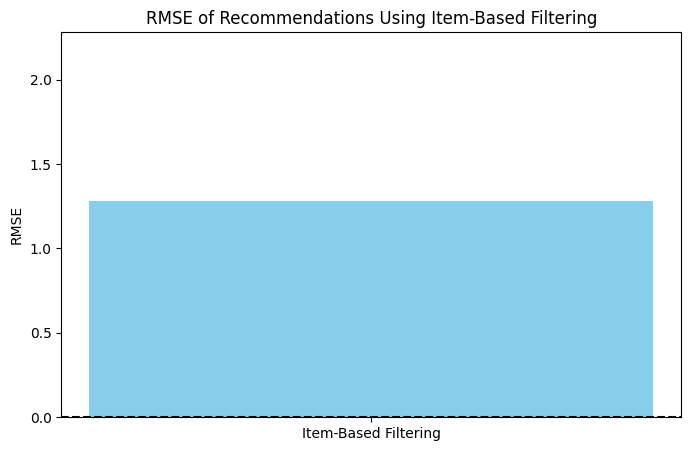
**4. Results:**

**4.1 COS Similarity User-Based Filtering**



RMSE of the recommendations: 1.75

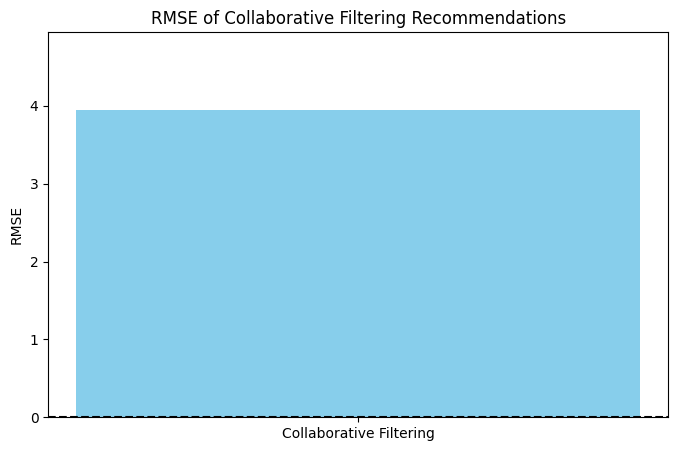
**4.2 COS Similarity item\_base**



The comparison in the user-based approach and the other approach based on popularity.

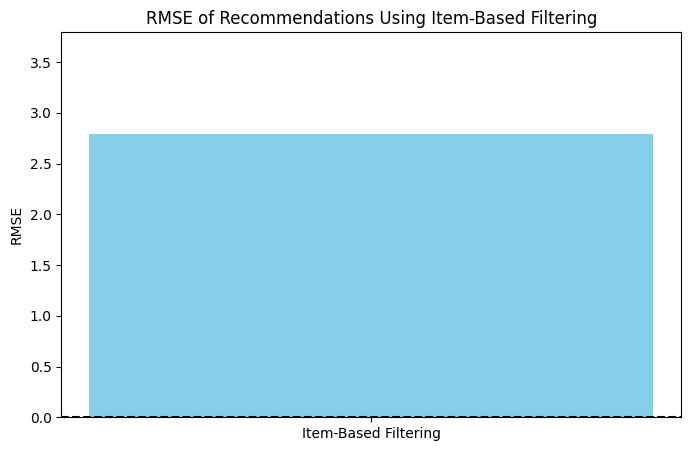
RMSE of the recommendations (item-based): 1.28

**4.3 pearson user base**



RMSE of recommendations: 3.9455829938315885

**4.4 pearson item base**



RMSE of the recommendations (item-based): 2.79

**Comparison of Recommendation Algorithms**

|  |  |  |
| --- | --- | --- |
| Method | Type | RMSE |
| Cosine Similarity | User-Based | 1.75 |
| **(The best)**  **Cosine Similarity** | **Item-Based** | **1.28** |
| Pearson | User-Based | 3.9456 |
| Pearson | Item-Based | 2.79 |

**5. Conclusion**

The developed system serves as an excellent platform to construct a neighborhood-based collaborative filtering recommender system, with simulated user data available from the TMDb API. Every element, ranging from the process of acquiring data, creating recommendations, and assessing the recommendations, adds to the effectiveness and trustworthiness of the system. Once each of these components is mastered, there are possibilities to upgrade the system and transform it to serve other purposes.

**6. References:**

1. [1] T. M. McAuley and J. Leskovec, "Learning to Discover Social Circles in Ego Networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 1, 2014.
2. [2] "The Movie Database (TMDb)," [Online]. Available: https://www.themoviedb.org/documentation/api. [Accessed: Nov. 3, 2024].
3. [3] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, pp. 331–370, 2002.
4. [4] S. Rendle et al., "Factorization Machines," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 1, pp. 47–58, 2012.