TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

----- 🙡 🕮 🙣 -----



**GROUP REPORT**

***Subject:* Movie Recommendation Using Collaborative Filtering**

Lecturer: **Ph.D. Nguyen Nhat Quang**

Class: 139399

|  |  |  |
| --- | --- | --- |
| Student Name | Student ID | Email |
| Tong Tran Minh Duc | 20205147 | [duc.ttm205147@sis.hust.edu.vn](mailto:duc.ttm205147@sis.hust.edu.vn) |
| Dang Trong Luat | 20205187 | luat.dt205187@sis.hust.edu.vn |

**Hà Nội, năm 2022**

MỤC LỤC

[MỤC LỤC 2](#_Toc137674119)

[I. INTRODUCTION 3](#_Toc137674120)

[**1. Problem description** 3](#_Toc137674121)

[**2. Dataset** 3](#_Toc137674122)

[**3. Methodology** 4](#_Toc137674123)

[II. APPROACHING THE PROBLEM 4](#_Toc137674124)

[**1.** **Cosine Similarity metric :** 4](#_Toc137674125)

[**2.** **Item-Based Collaborative filtering** 5](#_Toc137674132)

[**3.** **User-Based Collaborative Filtering** 15](#_Toc137674133)

[CONCLUSION 21](#_Toc137674135)

[REFERENCES 24](#_Toc137674142)

**I. INTRODUCTION**

**1. Problem description**

* The current movie recommendation system lacks personalized suggestions and fails to capture the unique preferences of individual users. This leads to a suboptimal user experience and reduces the chances of users discovering movies that align with their interests. The objective is to improve the recommendation system by implementing two approach, item-based collaborative filtering, which leverages the similarities between movies to make more accurate and personalized movie recommendations and user-based collaborative filtering, which different from the user-based approach, find similarities between users and recommend appropriate content based on those similarities.
* To calculate the distance between two vetors in the user-item matrix, our group used the cosine similarity metric to measure the correlation between 2 users or 2 movies.

**2. Dataset**

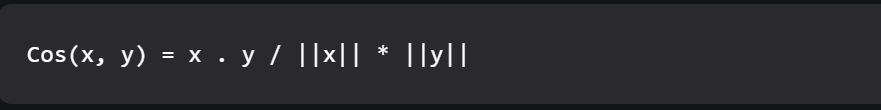
* The project utilizes a dataset consisting of movie ratings provided by users. The dataset contains information about the movies, users, and their corresponding ratings.
* This dataset (ml-latest-small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018.
* Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

**3. Methodology**

* For this problem, we have collectively agreed on the collaborative filtering method, using a collection of ratings to dictate the prediction for each user-movie pair rating.
* Our approach is separated into two methods: Item-based collaborative filtering and User-based filtering. The item-based approach rely on the similarities in each movie’s rating while user-based filtering focuses on the similarity score between each user’s set of ratings.
* Both approaches utilizes the cosine similarity metric to evaluate how similar each vector(movie or user) is to the other movies or users to make up the “collaborative” part of the approach. Then the recommender selects the most similar vectors (using KNN or sort the scores in descending order) to factor the prediction for ratings of a particular user-movie pair in the dataset which make up the “filtering” part.

**II. APPROACHING THE PROBLEM**

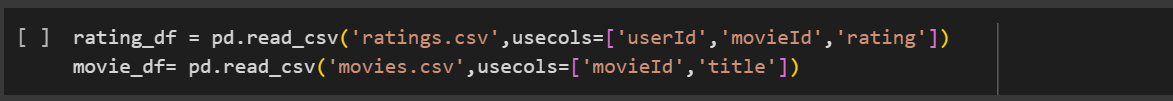
1. **Cosine Similarity metric :**

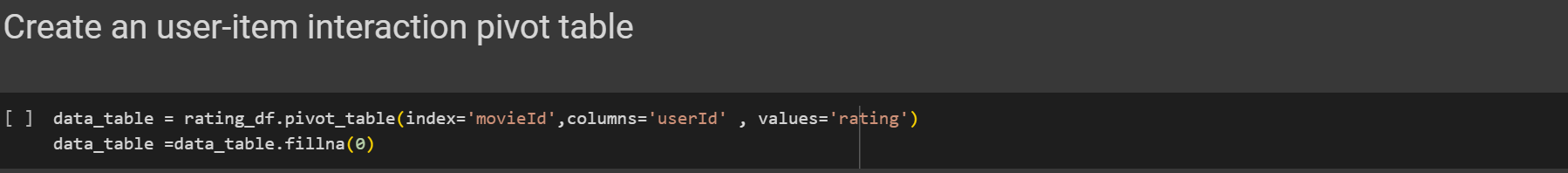
* Cosine similarity is a metric, helpful in determining, how similar the data objects are irrespective of their size. We can measure the [similarity between two sentences in Python](https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/) using Cosine Similarity. In cosine similarity, data objects in a dataset are treated as a vector. The formula to find the cosine similarity between two vectors is :

Where :

* + x . y : the product of vector x and y.
  + ||x|| and ||y|| : the length of vector x and y.
  + ||x|| \* ||y||: the cross product of 2 vectors x and y.
* Cosine similarity can calculate the similarity score between each vector, in this case, movies and users that in the dataset have similar features and patterns. Then sorting them in descending order will return the top most similar vectors or calling a KNN model and obtaining the nearest neighbors of the vector being considered.

1. **Item-Based Collaborative filtering**
2. **Data preprocessing :**

* For this approach, the timestamp data from the ratings.csv file and the genre column in the movies.csv file is not needed, so after reading the csv files, the recommender drops those two columns of the dataframes.
* After importing the datasets, to evaluate the model’s performance and accuracy at a later time, the recommender splits the dataset into a training set and test set and export the the training set and test set in the .csv format.
* Then to prepare the data for training the Machine learning model, the recommender coverts the training and test dataframes into pivot tables with userId as the columns and movieId as the rows matching each row and column to a rating value, all of the user-movie pair that does not have a rating value, or the an user did not rate a movie, the recommender fill that cell in the table with the zero value.



A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with low confidence

* Note that a pivot table is also create for the training set to prepare for the data training phase

1. **Global average values**

* In the process of creating the movie recommendations system, there were a lot a vectors that were not similar to any other vectors in the dataset, vectors that when calculating the cosine similarity score with each of the other vectors returned a score of zero, and since the rating predictions has to be normalized to fit in the rating scale (0 to 5). The recommender create global average values, average rating of a movie across all users and average rating of a user across every movie in the dataset, so that whenever the recommender encounters the vectors that return cosine similarity of 0, the average value of both that user and movie is filled in as the prediction.
* For creating these average values, the recommender create a csr sparse matrix:

A picture containing text, screenshot, font

Description automatically generated

A screenshot of a computer code

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with low confidence

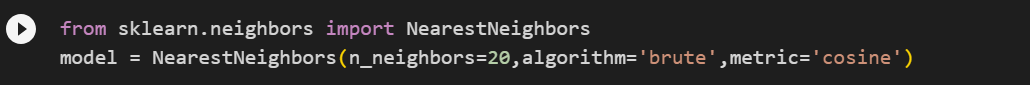
A screenshot of a computer program

Description automatically generated with low confidence

* Since when evaluating the recommender on the test set, we don’t want any patterns of the training set to appear in the test set, so the recommender calculate different global average values of the train set and the test set.

1. **Model Selection**

* Our choice for machine learning model is K nearest neighbors, using the cosine similarity metric.
* After importing the model, tuning the parameters, the model is trained on the training set converted into a pivot table.



A screenshot of a computer

Description automatically generated with medium confidence

* In the process of testing and implementing KNN into the system, the number of neighbors to consider is set to 20, since that value produces the most stable and static result in the evaluation phase of the recommendation system.

1. **Obtaining predictions**

* The predictions on the training and test set is determined by the get\_predictions function, whichs produces the predicted rating for each user movie pair in the training and test set.

A picture containing text, screenshot, software, font

Description automatically generated

A picture containing text, screenshot, software

Description automatically generated

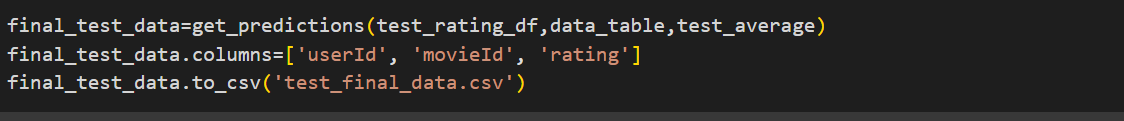
* The formula to calulate the predicted rating is as follows:

R(m,u)= { sum( S(m,j) \* R(j,u) ) }/ sum( S(m,j)

* + R(m,u) : the rating for the movie m by user u
  + S(m,j): the similarity between movie m and movie j
  + j in J is the set of movies similar to movie m
* At first the function creates a dataframe and every time a pair completes calculations, the function creates a row and append it to the end of the dataframe.

A picture containing text, screenshot, font

Description automatically generated



* For the evaluation phase, the output of the recommender is exported in the .csv format.

1. **Making recommendations**

A screenshot of a computer

Description automatically generated with medium confidence

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screen shot of a computer code

Description automatically generated with low confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

* The system can make recommendations to any user in the dataset, to choose, type in the userId of the user to make recommendations.
* Then using KNN model created earlier, the model selects the most similar movies to the movies that the user rated and factor those movies ratings using the formula above and normalize the results.
* The list of movies are then sort in descending order to make out the top 10 most fit movies to recommend to the user.
* **The Threshold value:**
  + The threshold value in this model is set to 50, as in a movie’s ratings is only considered if it is rated at least by 50 users.
  + The reason for this is because there exists obsure movies in the dataset that only rated by a handful of people(1 or 2), but get a high correlation score with a popular movie, this “Black sheep” object of the dataset can dictate the recommender to make inappropriate recommendations to the user.
  + The threshold value can also be changed to a smaller value(in the case that the user wants tp see more unpopular movies) and tuned up to 100 (in that case the results can be blockbusters and popular movies).

1. **Data evaluation :**

* The metric being used to evaluate the output of the system is Root-Mean-Squared-Error (RMSE) and Mean Absolute Percentage Error(MAPE) to calculate the average error a prediction makes on a single prediction.
* RMSE measures the squared loss, while MAPE measures the absolute loss. Lower values mean lower error rates and thus better performance.

RMSE = sqrt((r-r^)^2/N)

* r is the actual rating
* r^ is the predicted rating
* N is the total prediction

MAPE = (1/N)\*(sum( abs(r-r^)/r )) \*100

* r is the actual rating
* r^ is the predicted rating
* N is the total prediction

A picture containing text, multimedia software, font, software

Description automatically generated

- Testing results on the training set:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence- Testing results on the test set

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

- Overall on average the predictions are off by about 1.02 – 1.03 on the rating scale which is about 32% error on both training and test set.

1. **User-Based Collaborative Filtering**
2. **Data preprocessing :**

* For this approach, the timestamp data from the ratings.csv file and the genre column in the movies.csv file is not needed, so after reading the csv files, the recommender drops those two columns of the dataframes.

A picture containing text, font, screenshot, line

Description automatically generated

* After importing the datasets, to evaluate the model’s performance and accuracy at a later time, the recommender splits the dataset into a training set and test set and export the the training set and test set in the .csv format.

A picture containing text, font, line, screenshot

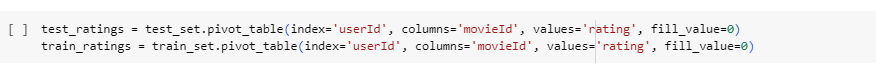
Description automatically generated

* Then to prepare the data for training the Machine learning model, the recommender coverts the training and test dataframes into pivot tables with userId as the columns and movieId as the rows matching each row and column to a rating value, all of the user-movie pair that does not have a rating value, or the an user did not rate a movie, the recommender fill that cell in the table with the zero value.

A screenshot of a computer

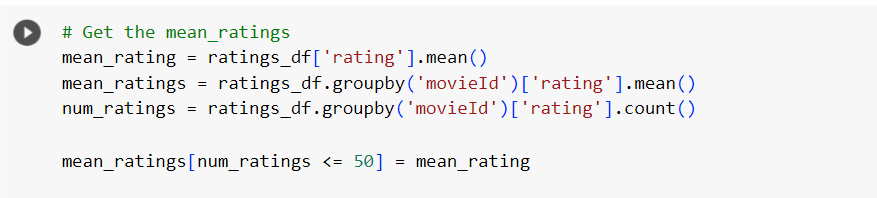
Description automatically generated with low confidence

* Note that a pivot table is also create for the training set to prepare for the data training phase



1. **Global average values**

Since we use k-similar users approach, there might be cases when a movie has not been rated by any neighbors. Its neccessary to fill this value with a mean rating of that movie accross the train\_set



1. **Model Selection**

* Our choice for machine learning model is user-based nearest neighbor model, using the cosine similarity metric.
* After importing the model, tuning the parameters, the model is trained on the training set converted into a pivot table.

A picture containing text, font, screenshot, line

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

* In the process of testing and implementing KNN into the system, the number of neighbors to consider is set to 5, since that value produces the most stable and static result in the evaluation phase of the recommendation system.

1. **Obtaining predictions**

* The predictions on the training and test set is determined by the get\_predictions function, whichs produces the predicted rating for each user movie pair in the training and test set.

A screen shot of a computer code

Description automatically generated with low confidence

A screenshot of a computer program

Description automatically generated with medium confidence

* The formula to calulate the predicted rating is as follows:

R(m,u)= { sum( S(u,j) \* R(m,j) ) }/ sum( S(u,j)

* + R(m,u) : the rating for the movie m by user u
  + S(u,j): the similarity between user u anduser j
  + j in J is the set of users similar to user u

1. **Making recommendations**

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

* The system can make recommendations to any user in the dataset, to choose, type in the userId of the user to make recommendations.
* Then using k-most similar user model created earlier, the model selects the highest rated movie to the user based on other similar users’rating and factor those movies ratings using the formula above and normalize the results.
* The list of movies are then sort in descending order to make out the top 10 most fit movies to recommend to the user.
* **The Threshold value:**
  + The threshold value in this model is set to 50, as in a movie’s ratings is only considered if it is rated at least by 50 users, else the mean value is taken into account.
  + The reason for this is because there exists obsure movies in the dataset that only rated by a handful of people(1 or 2), but get a high correlation score with a popular movie, this “Black sheep” object of the dataset can dictate the recommender to make inappropriate recommendations to the user.
  + The threshold value can also be changed to a smaller value(in the case that the user wants tp see more unpopular movies) and tuned up to 100 (in that case the results can be blockbusters and popular movies).

1. **Data evaluation :**

* The metric being used to evaluate the output of the system is Root-Mean-Squared-Error (RMSE) to calculate the average error a prediction makes on a single prediction.
* RMSE measures the squared loss, while MAPE measures the absolute loss. Lower values mean lower error rates and thus better performance.

RMSE = sqrt((r-r^)^2/N)

* r is the actual rating
* r^ is the predicted rating
* N is the total prediction

A picture containing text, screenshot, font

Description automatically generated

- Testing results on the test set

A picture containing text, screenshot, font, line

Description automatically generated



- Overall on average the predictions are off by about 2.08 on the rating scale which is about 40% error on test set.

**CONCLUSION**

In conclusion, there are still improvements to be made to this system.For a modern recommendation system to be efficient, the recommended RMSE score is under 2 (excellent if it is under 0.8). The «Black sheep« and «cold start« problem is still not handle efficiently, there is also the problem of having the cosine similarity between objects equal to 0, hence making ratings predictions harder to produce.Overall, the recommendation system still produced a range of appropriate movies for each users in the dataset and at an acceptable rate and a high metric score.

In this study, we implemented both Item-Based Collaborative Filtering (IBCF) and User-Based Collaborative Filtering (UBCF) techniques for movie recommendation. We evaluated the performance of these two approaches using the Root Mean Squared Error (RMSE) metric to assess the accuracy of the predicted ratings.

Our findings indicate that the RMSE value obtained from the Item-Based Collaborative Filtering approach was lower than that of the User-Based Collaborative Filtering approach. This suggests that the Item-Based approach outperformed the User-Based approach in terms of rating prediction accuracy.

The lower RMSE value achieved by the Item-Based Collaborative Filtering approach can be attributed to its ability to capture item-item similarity and leverage the overall rating patterns of movies. By considering the similarities between movies and their corresponding ratings, the Item-Based approach can provide accurate predictions for unrated movies based on the ratings of similar items.

This result implies that the Item-Based Collaborative Filtering technique may be more suitable for our movie recommendation system. The accurate predictions generated by the Item-Based approach can enhance the user experience by providing relevant and personalized movie recommendations based on item similarities..

It is important to note that the performance of these collaborative filtering approaches can vary depending on the dataset, its characteristics, and the specific implementation. Therefore, further experimentation and evaluation may be necessary to validate these findings and explore other factors that can impact the recommendation accuracy.

**REFERENCES**

|  |  |
| --- | --- |
| [1] | [RMSE vs MAPE, which is the best regression metric?](https://stephenallwright.com/rmse-vs-mape/) |
| [2] | [Cosine Similarity - GeeksforGeeks](https://www.geeksforgeeks.org/cosine-similarity/) |
| [3] | [Item-item collaborative filtering - Wikipedia](https://en.wikipedia.org/wiki/Item-item_collaborative_filtering" \l ":~:text=Item%2Ditem%20collaborative%20filtering%2C%20or,by%20Amazon.com%20in%201998.) |
| [4] | [Collaborative filtering](https://en.wikipedia.org/wiki/Collaborative_filtering) |
|  |  |