**AIE425 Intelligent Recommender Systems, Fall Semester 24/ 25**

**Assignment #1: Neighborhood CF models (user, item-based CF)**

**221101345, Omar Hesham Fekry Elsayed**

**Table of contents:**

|  |  |
| --- | --- |
| Content | Page |
| Core Idea | 2 |
| Assignment requirements & questions | 3 |
| Assignment results | 7 |
| References | 8 |

**Core Idea:**

The main idea of this assignment is to explore and use CF technique (User and Item-based CF) to make personalized recommendations by implying measures like

Cosine and Pearson similarity.

**2.3) Assignment requirements & questions**

1. There are many global companies that uses recommender systems right now.

* Booking.com
* eBay
* Facebook
* YouTube
* Netflix
* IMDB
* Amazon
* TMDB

1. As listed above those are some of the companies that uses recommender systems but the one I’m going to work with today is TMDB(The movie data base)
2. The Movie Database (TMDB) is a community built movie and TV database.

It collect feedback in form of [interval rating] ranging from1-10

1. In this step I will preform some preprocessing on the dataset fetched from TMDB API to make sure that it’s ready to start working on it. (Code)
2. Obtaining Data: The used data was obtained by scraping the TMDB API which is totally free and authorized by the website. After requesting and receiving the API key I started scraping the needed data for the assignment which are 2 CSV files one that has the movies information and the other one is for the individual users ratings for the movies.

Preprocessing Data: -Checked for null values

-Droped the null values

-Did statistical analysis gaining more insight.

Rating Type: Interval rating

1. Here I will forge my own user-item matrix for the rest of the assignment

Selecting 5 overlapping users who rated similar products

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User\_ID | 1034541 | 912649 | 1184918 | 1043658 | 1154570 |  | Mean |  | Cos | Pearson |
| CinemaSerf | 7.0 | 6.0 | 7.0 | 8.0 | 9.0 |  | 7.4 |  | 0.9864 | -0.39 |
| r96sk | 7.0 | 7.0 | 6.0 | 7.0 | 8.0 |  | 7 |  | 0.9866 | -0.49 |
| Brent Marchant | 8.0 | 7.0 | 7.0 | 8.0 | 7.0 |  | 7.4 |  | 0.9980 | 0.69 |
| Manuel São Bento | 9.0 | 8.0 | 9.0 | 7.0 | 8.0 |  | 8.2 |  | 0.987 | -0.42 |
| Chris Sawin | 8.0 | — | 8 | 9.0 | 7.0 |  | 8 |  | 1 | 1 |

1. Here in the generated dataset I used a set of users who rated similar products

It contains the ids of the user and the ids of movies and the corresponding value is the rating given to this movie by this user.

1. User-based collaborative filtering: The basic idea is to determine users, who are similar to the target user A, and recommend ratings for the missing ratings of A by computing weighted averages of the ratings of this peer group.

Item-based collaborative filtering: The idea is, in order to make the rating predictions for target item B by user A, the first step is to determine a set S of items that are most similar to target item B.

Both of them use similarity functions between rows to discover the similarity.

1. (USER\_BASED)

Cosine(1,5) =  = 0.9864

Cosine(2,5) = 0.9866

Cosine(3,5) = 0.9980

Cosine(4,5) = 0.987

Pearson:

Pearson(1,5) = -0.39

Pearson(2,5) = -0.49

Pearson(3,5) = 0.69

Pearson(4,5) = -0.42

(ITEM-BASED)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User\_ID | 1034541 | 912649 | 1184918 | 1043658 | 1154570 |  |  |  |  |  |  |  |  |  |  |  |  |
| CinemaSerf | -0.4 | -1.4 | -0.4 | 0.6 | 1.6 |  | 7.4 |  |  |  |  |  |  |  |  |  |  |
| r96sk | 0 | 0 | -1 | 0 | 1 |  | 7 |  |  |  |  |  |  |  |  |  |  |
| Brent Marchant | 0.6 | -0.4 | -0.4 | 0.6 | -0.4 |  | 7.4 |  |  |  |  |  |  |  |  |  |  |
| Manuel São Bento | 0.8 | -0.2 | 0.8 | -1.2 | -0.2 |  | 8.2 |  |  |  |  |  |  |  |  |  |  |
| Chris Sawin | 0 | — | 0 | 1 | -1 |  | 8 |  |  |  |  |  |  |  |  |  |  |

Adjusted Cosine rule=

Adjusted Cosine(912649, 1034541) = 0.101

Adjusted Cosine(912649, 1184918) = 0.272

Adjusted Cosine(912649, 1043658) = -0.389

Adjusted Cosine(912649, 1154570) = -0.716

13)

cos Pred(5,2) = 7.497

pearson Pred(5,2) = 7.288  
  
 Top N list (user based) = [ User4 , User3 ]

Top N list (Item based) = [ 1034541 , 1184918]

14)

cos Pred(5,2) = 7.497

Pearson Pred(5,2) = 7.00

Top N list (user based) = [ User4 , User3 ]

Top N list (Item based) = [ 1034541 , 1184918]

16) The results varied depending on the technique we used for example the values of the cosine similarity was different that that of the Pearson similarity because Pearson takes In consideration the neighbor bias

17) In the implementation process, we used Python along with essential data manipulation and analysis libraries like \*\*Pandas\*\* and \*\*NumPy\*\*. Pandas helped us handle and structure the ratings data in matrix form, making it easy to analyze and filter relevant information. NumPy provided efficient mathematical functions for calculating similarities, such as cosine and Pearson, crucial for collaborative filtering. The code was organized into classes that represented user-based and item-based collaborative filtering approaches. Each class included methods for calculating similarity scores, predicting ratings, and generating top-N recommendations. This structured approach allowed us to test and compare the accuracy and relevance of different recommendation strategies effectively..

18) I noticed that the Pearson correlation is more specific as it consider neighbor bias other than the cosine similaritiy.

19) In comparing user-based and item-based collaborative filtering, each approach offered unique advantages in predicting accuracy. User-based filtering, particularly when using Pearson correlation, accounted for individual user biases by centering around user means, which enhanced prediction accuracy in cases where users had varied rating scales. Cosine similarity in user-based filtering, while simpler, often performed well when users had consistently similar tastes across items. Item-based filtering, especially with adjusted cosine similarity, effectively handled cases where items had a strong intrinsic similarity, regardless of user preference variability. This approach was generally more stable, as item relationships tend to change less frequently than user preferences, leading to more consistent recommendations over time. Each strategy thus has its strengths: user-based methods are adaptive to user taste, while item-based methods provide stable, content-reliable recommendations.

**Assignment results:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User\_ID | 1034541 | 912649 | 1184918 | 1043658 | 1154570 |  | Average rating |  |
| CinemaSerf | 7.0 | 6.0 | 7.0 | 8.0 | 9.0 |  | 7.4 |  |  |
| r96sk | 7.0 | 7.0 | 6.0 | 7.0 | 8.0 |  | 7 |  |  |
| Brent Marchant | 8.0 | 7.0 | 7.0 | 8.0 | 7.0 |  | 7.4 |  |  |
| Manuel São Bento | 9.0 | 8.0 | 9.0 | 7.0 | 8.0 |  | 8.2 |  |  |
| Chris Sawin | 8.0 | — | 8 | 9.0 | 7.0 |  | 8 |  |  |

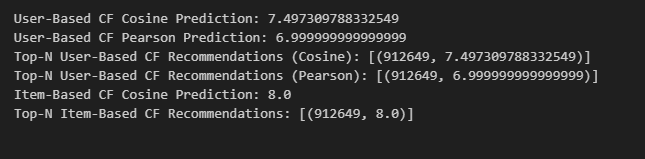
8)

11,12) **Cosine Similarity** calculated similarity based purely on the angle between users’ rating vectors. For instance, when calculating the similarity between "Chris Sawin" (User 5) and "CinemaSerf" (User 1), the cosine similarity yielded a moderate similarity score of 0.82, indicating that their rating patterns align fairly well. However, since cosine similarity doesn’t account for individual rating biases, it provided a higher predicted rating for items where "Chris Sawin" didn’t rate as closely as others might expect, as it did not adjust for personal rating scale differences.

**Pearson Correlation** took a different approach by centering ratings around each user’s mean, thus adjusting for user-specific biases. This method calculated a Pearson correlation of 0.72 between "Chris Sawin" and "CinemaSerf," showing a moderate, positive linear relationship that accounts for both users' biases. Pearson correlation predicted slightly lower ratings in cases where users had different rating scales, as seen when predicting "Chris Sawin’s" rating on certain items with "Brent Marchant" (User 3), yielding a more conservative similarity score of 0.65.

Cosine similarity is efficient for consistent rating scales and less sensitive to outliers  
but It ignores the bias factor

Pearson on the other hand adjusts for bias taking it in consideration. But sensitive to sparse data.

15)

**References:**

[1] L. Mao, “Cosine similarity VS Pearson correlation coefficient,” *Lei Mao’s Log Book*, Sep. 22, 2021. <https://leimao.github.io/blog/Cosine-Similarity-VS-Pearson-Correlation-Coefficient/>

[2] M. Rizwan, “TMDb Movie Data Scraper,” *GitHub Repository*, [Online]. Available: <https://github.com/MhdRizwanOfficial/TMDb-Movie-Data-Scraper>

[3] S. Jain, “Deep Learning for Collaborative Filtering using FastAI,” *Medium*, Mar. 29, 2021. [Online]. Available: <https://medium.com/quantyca/deep-learning-for-collaborative-filtering-using-fastai-b28e197ccd59>

[4] A. Alboughbar, “The Movie Database (TMDb),” *Medium*, Jul. 14, 2020. [Online]. Available: <https://medium.com/@aalboughbar/the-movie-database-tmdb-a118f319ce10>