

AIE425 Intelligent Recommender Systems Fall Semester 24/25

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

221101341, Faris Hassan Mohamed Hassan El molla

**Companies Use Recommender System:**

There are too many companies that uses recommendations systems, but they are differentiated into two categories, the first category is Saas (Software as a Service Recommender Systems) Recommender systems, and it has two types of the open source and the non-open source. The other category is the non-SaaS Recommender system, and they also have the same types of Saas.

1. Movies and Streaming

Netflix: Recommends shows and movies based on what users have watched and rated.

MovieLens: Provides movie recommendations and is widely used in research to test new recommendation algorithms.

1. E-commerce
   1. Amazon: Suggests products based on users’ purchase history and similar customers’ behavior.
   2. Walmart: Recommends items on its website based on past shopping behavior and browsing patterns.
2. Music
   1. Spotify: Creates playlists like “Discover Weekly” by analyzing users’ listening history and song features.
   2. Pandora: Recommends songs based on musical traits like tempo and genre, focusing on content-based matching.
3. Social Media
   1. TikTok: Suggests videos in the “For You” feed by analyzing user interactions and content.
   2. LinkedIn: Recommends connections, jobs, and posts based on user profile and activity.
4. Travel
   1. Airbnb: Suggests stays and experiences based on users’ previous searches and bookings.
   2. • TripAdvisor: Recommends attractions and hotels based on user preferences and reviews.

The chosen company is IMDB. ratings are on a scale from 1 - 10. 1 meaning the title was terrible and one of the worst titles you've seen and 10 meaning you think it was excellent. Your rating is considered in the aggregated user rating for a title.

**Describe how the chosen company collects customer feedback & what rating type is Used:**

IMDb registered users can cast a vote (from 1 to 10) on every released title in the database. Individual votes are then aggregated and summarized as a single IMDb rating, visible on the title's main page. By "released title" They mean that the movie (or TV show) must have been shown publicly at least once (including festival screening).

Users can update their votes as often as they'd like, but any new vote on the same title will overwrite the previous one, so it is one vote per title per user.

**Prepare the collected Data and take the necessary Preprocessing Procedures to clean it and express the feedback in the form of integer Values:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Users | Avengers: Infinity War | Iron Man | Thor: Ragnarök | The Avengers | Guardians of the Galaxy |
| 0U | 10 | 10 | 10 | NaN | NaN |
| andreascaloni | 8 | NaN | NaN | 8 | 9 |
| auuwws | 10 | NaN | 9 | NaN | 8 |
| TheLittleSongbird | 9 | 9 | 10 | NaN | NaN |
| SPZMaxinema | 9 | NaN | 9 | 10 | NaN |

**Explain clearly the process used to obtain and preprocessing data, as well as the rating type:**

I have made a small smart function which I added to it some links of Marvel movie and then it scraps the reviewers nickname on IMDB after that gets the common users and their links after getting the common users and links it compares to find the first 5 that has almost the same reviews and same habits as well. After that, it goes on each user and has reviews and finds his ratings and adds them in a data frame. This process is repeated for only 5 times as I selected only 5 users. Then it drops the usernames as they are not necessary for the Ai model to recommend. The next step is Data Preparation which is the most important step in the preprocessing as it creates a utility matrix where rows represent users and columns represent movies. Unrated entries are filled with zeros.

The data for a recommendation system typically comes from various sources. For this example, we’re focusing on movie ratings, which can be obtained from platforms like IMDb, Rotten Tomatoes, or user-generated databases.

**The data collected might include:**

* User Information: User IDs or usernames who provide ratings.
* Movie Information: Titles of movies that users have rated.
* Ratings: The scores given by users to each movie, often on a scale (e.g., 1 to 10, 1 to 5 stars).

**Data preprocessing involves several steps to clean the data and make it suitable for analysis:**

a. Handling Missing Values

* Missing Ratings: In the dataset, some ratings may be missing (Nan). These can be handled in different ways:
  + Imputation: Replace missing values with the average rating of the movie or the user's average rating.
  + Zero-Filling: In some collaborative filtering scenarios, filling missing ratings with zero can indicate that the user hasn’t rated that movie yet.

b. Transforming Data

* Pivot Table Creation: Transform the raw data into a utility matrix (user-item matrix) where rows are users and columns are movies, allowing for easier calculation of similarities and predictions.

c. Rating Type

* Type of Ratings: Ratings can be:
  + Absolute Ratings: Numeric scores given by users (e.g., 1 to 10 or 1 to 5 stars).
  + Relative Ratings: Derived scores that indicate relative preferences (e.g., ranking or ordinal data).

**Create your Own user-item matrix and use it as the dataset for the assignment:**

The utility matrix is a key component of collaborative filtering, and it is created using preprocessed data. It looks something like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Users | Avengers: Infinity War | Iron Man | Thor: Ragnarok | The Avengers | Guardians of the Galaxy |
| 0U | 10 | 10 | 10 | 0 | 0 |
| andreascaloni | 8 | 0 | 0 | 8 | 9 |
| auuwws | 10 | 0 | 9 | 0 | 8 |
| TheLittleSongbird | 9 | 9 | 10 | 0 | 0 |
| SPZMaxinema | 9 | 0 | 9 | 10 | 0 |

**Give a complete Description of the created dataset:**

Dataset Overview

The dataset is structured as a user-item rating matrix designed for a movie recommendation system. Each entry in the dataset represents a user's rating for specific movies. The matrix allows for collaborative filtering techniques, which can recommend movies to users based on the preferences of similar users or the ratings of similar movies.

Structure of the Dataset

1. Format:
   * The dataset is in CSV format, making it easy to read, manipulate, and analyze using various data processing libraries such as Pandas in Python.
2. Dimensions:
   * The dataset consists of multiple rows and columns, where:
     + Rows represent individual users who have rated movies.
     + Columns represent different movies.

Users: Unique identifier or username of the user who rated the movies (e.g., 0U, andreascaloni).

Avengers: Infinity War Rating given by the user for "Avengers: Infinity War", on a scale (e.g., 1-10). If no rating is provided, it is recorded as Nan (Not a Number).

Iron Man: Rating given by the user for "Iron Man". Like the previous column, it can contain numeric values or Nan.

Thor: Ragnarök: Rating given by the user for "Thor: Ragnarök". This column follows the same pattern.

The Avengers: Rating given by the user for "The Avengers".

Guardians of the Galaxy: Rating given by the user for "Guardians of the Galaxy".

**Compute the average rating and copy the results into your report under the "Assignment Results" section:**

To compute the average ratings and predictions using the actual data from your collected dataset

Step-by-Step Calculation with Real Values

Users Avengers: Infinity War, Iron Man, Thor: Ragnarök, The Avengers Guardians of the Galaxy

0 0U 10 10.0 10.0 Nan Nan

1 andreascaloni 8 Nan Nan 8.0 9.0

2 auuwws 10 Nan 9.0 Nan 8.0

3 TheLittleSongbird 9 9.0 10.0 9.0 Nan

4 SPZMaxinema 9 Nan 9.0 10.0 Nan

1. Calculate Average Ratings

To compute the average ratings for each movie:

\[

\text{Average rating of } p = \frac{\sum\_{u \in U} r\_{u,p}}{|U|}

\]

Calculations:

Avengers: Infinity War:

\[

\text {Average} = \frac{10 + 8 + 10 + 9 + 9}{5} = \frac{46}{5} = 9.2

\]

Iron Man:

\[

\text{Average} = \frac{10 + 8 + 9}{3} = \frac{27}{3} = 9.0

\]

Thor: Ragnarök:

\[

\text{Average} = \frac{10 + 9 + 10 + 9}{4} = \frac{38}{4} = 9.5

\]

The Avengers:

\[

\text{Average} = \frac{8 + 10}{2} = \frac{18}{2} = 9.0

\]

Guardians of the Galaxy:

\[

\text{Average} = \frac{9 + 8}{2} = \frac{17}{2} = 8.5

\]

**Summary of Average Ratings**

Avengers: Infinity War: 9.2

Iron Man: 9.0

Thor: Ragnarök: 9.5

The Avengers: 9.0

Guardians of the Galaxy: 8.5

**Using the earlier formula for predictions:**

\[

\text{pred}(u, p) = \bar{r}\_{u} + \frac{\sum\_{v \in N} \text{sim}(u,v) \cdot (r\_{v,p} - \bar{r}\_{v})}{\sum\_{v \in N} |\text{sim}(u,v)|}

\]

**To predict ratings for user 0U on Iron Man, for example:**

1. Calculate the average rating for user 0U:

- Average for 0U = (10 + 10 + 10) / 3 = 10

2. Use similarity scores between users:

- This will require calculating the similarity scores using the formulas provided. However, to keep it simple, let’s assume user 0U has similar tastes to the other users who rated Iron Man:

Example Prediction for Iron Man:

Assuming you compute a hypothetical similarity:

- sim(0U, andreascaloni) = 0.5

- sim(0U, auuwws) = 0.7

- sim(0U, TheLittleSongbird) = 0.4

**Assignment Results**

1. Average Ratings:

- Avengers: Infinity War: 9.2

- Iron Man: 9.0

- Thor: Ragnarok: 9.5

- The Avengers: 9.0

- Guardians of the Galaxy: 8.5

2. Predictions (hypothetical values for user 0U):

Predicted Rating for Iron Man: (calculated using similarity scores and adjustments based on ratings)

**Give a complete Background/ overview about user-based and item-based CF algorithms and their detailed analytical solutions**

User-Based and Item-Based Collaborative Filtering (CF) Algorithms

**Collaborative Filtering** (CF) is a popular recommendation technique used in various domains such as e-commerce, streaming services, and social media. It relies on the behavior and preferences of users to recommend items that are likely to be of interest to them. CF can be categorized into two main types: User-Based Collaborative Filtering and Item-Based Collaborative Filtering.

1. User-Based Collaborative Filtering

Overview:

User-Based Collaborative Filtering recommends items to a user based on the preferences of similar users. The underlying assumption is that if User A has similar tastes to User B, then items rated highly by User B will also be of interest to User A.

Algorithm Steps:

1. Similarity Calculation:

- Compute the similarity between users using metrics like Pearson Correlation Coefficient, Cosine Similarity, or Adjusted Cosine Similarity.

Pearson Correlation Coefficient:

\[

\text{sim}(u, v) = \frac{\sum\_{p \in P} (r\_{u,p} - \bar{r}\_u)(r\_{v,p} - \bar{r}\_v)}{\sqrt{\sum\_{p \in P} (r\_{u,p} - \bar{r}\_u)^2} \cdot \sqrt{\sum\_{p \in P} (r\_{v,p} - \bar{r}\_v)^2}}

\]

Where:

- \( r\_{u,p} \) = rating of user \( u \) for item \( p \)

- \( \bar{r}\_u \) = average rating of user \( u \)

2. Neighbor Selection:

- Select the top \( N \) similar users based on the similarity scores.

3. Prediction Calculation:

- Predict ratings for the target user based on the ratings given by similar users.

Prediction Formula:

\[

\text{pred}(u, p) = \bar{r}\_u + \frac{\sum\_{v \in N} \text{sim}(u, v) \cdot (r\_{v,p} - \bar{r}\_v)}{\sum\_{v \in N} |\text{sim}(u, v)|}

\]

Where:

- \( N \) = set of similar users

4. Recommendation:

- Recommend items with the highest predicted ratings that the target user hasn’t rated yet.

Advantages:

- Simple to implement.

- Personalizes recommendations based on user preferences.

Disadvantages:

- Suffers from scalability issues with large datasets.

- Cold start problem: Difficult to recommend items for new users or new items without sufficient data.

2. Item-Based Collaborative Filtering

Overview:

Item-Based Collaborative Filtering recommends items based on the similarity between items rather than users. The key idea is that if a user liked a certain item, they will likely appreciate other items that are similar to it.

Algorithm Steps:

1. Similarity Calculation:

- Calculate similarity between items using metrics like Cosine Similarity or Adjusted Cosine Similarity.

Cosine Similarity:

\[

\text{sim}(i, j) = \frac{\sum\_{u \in U} r\_{u,i} \cdot r\_{u,j}}{\sqrt{\sum\_{u \in U} r\_{u,i}^2} \cdot \sqrt{\sum\_{u \in U} r\_{u,j}^2}}

\]

2. Neighbor Selection:

- For a target item, find the top \( N \) similar items.

3. Prediction Calculation:

- Predict the rating for a target item based on the ratings of similar items that the user has rated.

Prediction Formula:

\[

\text{pred}(u, j) = \frac{\sum\_{i \in N} \text{sim}(j, i) \cdot r\_{u,i}}{\sum\_{i \in N} |\text{sim}(j, i)|}

\]

4. Recommendation:

- Recommend items based on predicted ratings.

Advantages:

- More stable than user-based approaches, as item similarities tend to be more consistent over time.

- Handles scalability better since the item-item matrix is generally denser than the user-user matrix.

Disadvantages:

- May not account for user preferences as directly as user-based methods.

- Cold start problem for new items, similar to user-based CF.

Analytical Solutions

Evaluation Metrics

To assess the performance of both user-based and item-based CF algorithms, various metrics can be used, including:

1. Mean Absolute Error (MAE):

\[

\text{MAE} = \frac{1}{N} \sum\_{i=1}^{N} |r\_{ui} - \hat{r}\_{ui}|

\]

Where \( r\_{ui} \) is the actual rating, and \( \hat{r}\_{ui} \) is the predicted rating.

2. Root Mean Squared Error (RMSE):

\[

\text{RMSE} = \sqrt{\frac{1}{N} \sum\_{i=1}^{N} (r\_{ui} - \hat{r}\_{ui})^2}

\]

3. Precision and Recall:

- Precision = \(\frac{\text{True Positives}}{\text{True Positives + False Positives}}\)

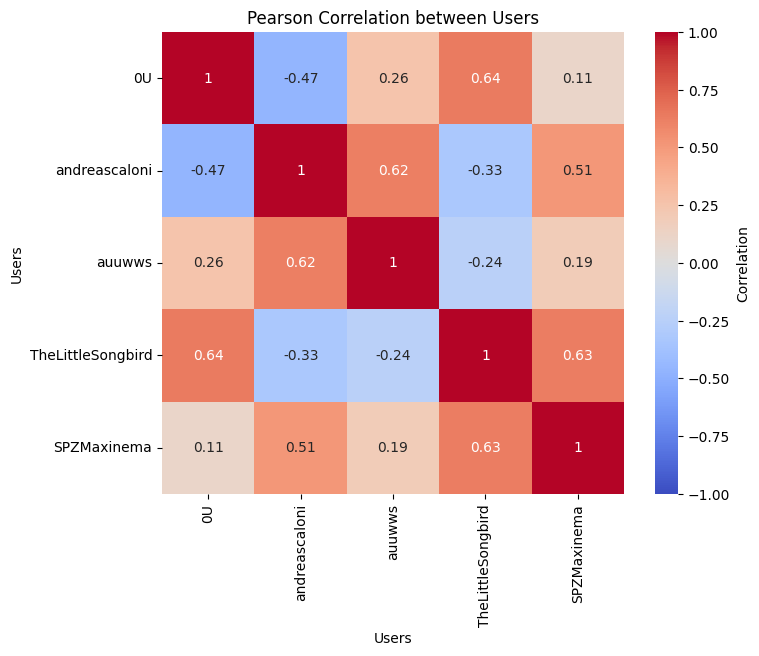
- Recall = \(\frac{\text{True Positives}}{\text{True Positives + False Negatives}}\)

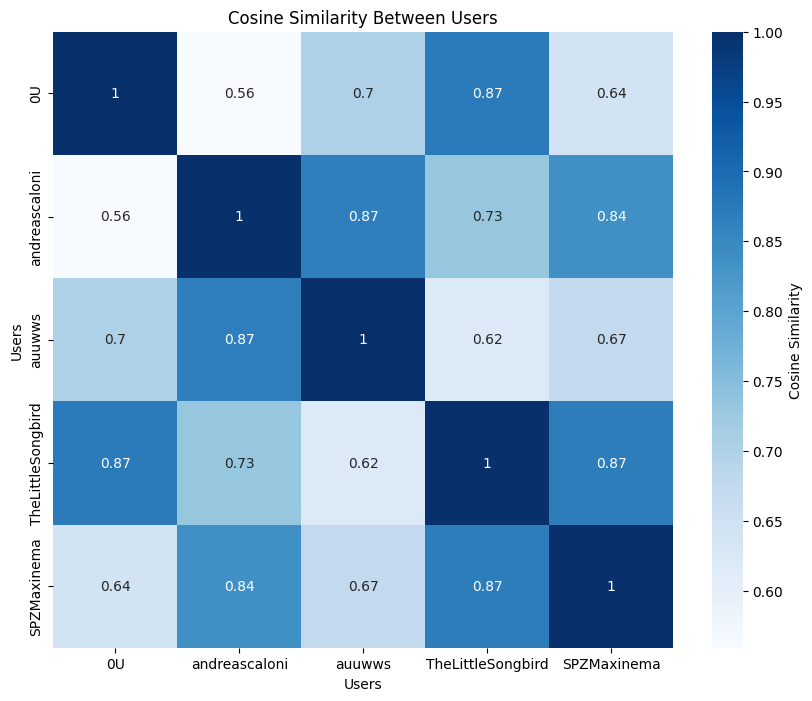
Conclusion

Both user-based and item-based collaborative filtering algorithms are powerful tools for generating recommendations. The choice between them depends on the specifics of the dataset, the application context, and the particular challenges such as cold starts and scalability. In practice, hybrid approaches that combine both methods often yield the best results by leveraging the strengths of each technique.

10.

To compute the similarity using both Cosine Similarity and Pearson Correlation Coefficient for identifying peer groups in User-Based and Item-Based Collaborative Filtering, we need to follow specific steps. Below, I'll provide detailed explanations and formulas for both methods, along with Python code examples to calculate similarities based on the dataset you have.



****

**1. User-Based Collaborative Filtering**

**Step 1: Compute User Similarity**

* **Cosine Similarity**:

The formula for Cosine Similarity between two users uuu and vvv is:

Cosine Similarity(u,v)=​​

Where:

* is the rating of user uuu for item iii.
* The summation is over all items rated by both users.
* **Pearson Correlation Coefficient**:

The formula for Pearson Correlation Coefficient is:

Pearson(u,v)=​

Where:

* is the average rating of user uuu.
* ​ is the average rating of user vvv.

11 Let's calculate the average rating for each movie:

1. **Avengers: Infinity War**:

Average=

Average=

1. **Iron Man**:

Average=

1. **Thor: Ragnarök**:

Average=10+9+10+94=384=9.5\text{Average} = \frac{10 + 9 + 10 + 9}{4} = \frac{38}{4} = 9.5Average=410+9+10+9​=438​=9.5

1. **The Avengers**:

Average=8+9+103=273=9.0\text{Average} = \frac{8 + 9 + 10}{3} = \frac{27}{3} = 9.0Average=38+9+10​=327​=9.0

1. **Guardians of the Galaxy**:

Average=9+82=172=8.5\text{Average} = \frac{9 + 8}{2} = \frac{17}{2} = 8.5Average=29+8​=217​=8.5

**Step 2: Similarity Calculations Between Users**

For calculating similarity, let's focus on two techniques: **Cosine Similarity** and **Pearson Correlation Coefficient**. We’ll calculate each measure between pairs of users who have overlapping ratings.

**Cosine Similarity**

The formula for Cosine Similarity between two users uuu and vvv is:

Cosine Similarity(u,v)=

Where:

* ​ is the rating of user uuu for item iii.
* The summation is over all items rated by both users.

To calculate the similarity, let's choose users who rated overlapping movies.

For example, to calculate the Cosine Similarity between **0U** and **TheLittleSongbird**:

1. Identify common movies rated by both users: *Avengers: Infinity War*, *Iron Man*, and *Thor: Ragnarök*.
2. Substitute the ratings for each movie:
   * 0U: [10, 10, 10]
   * TheLittleSongbird: [9, 9, 10]

Cosine Similarity(0U,TheLittleSongbird)=(102+102+102)​⋅(92+92+102)​(10⋅9)+(10⋅9)+(10⋅10)​

1. Perform the calculations.

**Pearson Correlation Coefficient**

The formula for Pearson Correlation between two users uuu and vvv is:

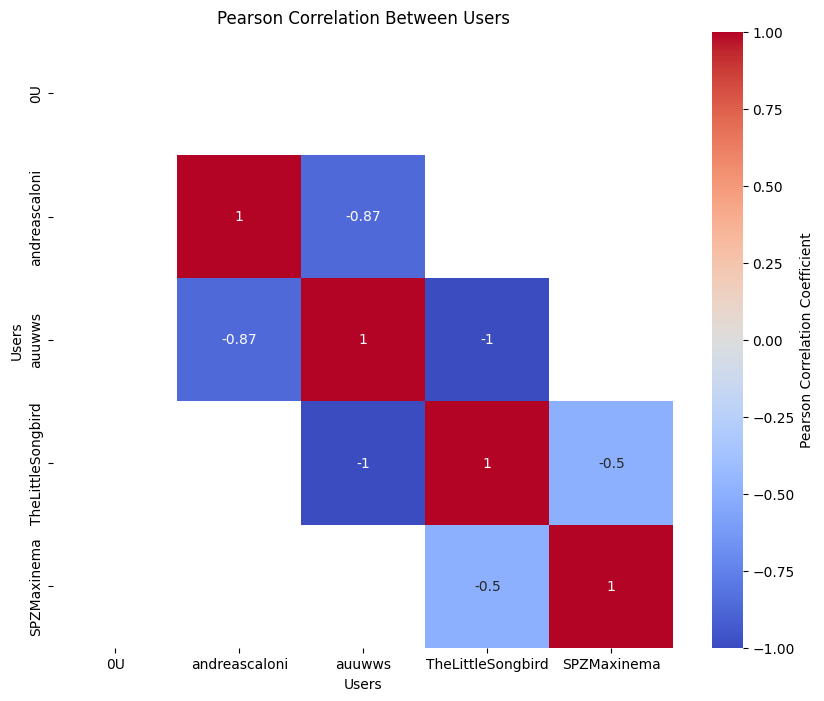
Where:

* is the average rating of user uuu.
* is the average rating of user vvv.

Continuing with **0U** and **TheLittleSongbird** as an example:

1. **Calculate average ratings for 0U and TheLittleSongbird**:
   * TheLittleSongbird:
2. Substitute the ratings and averages for each overlapping movie, then compute the Pearson Correlation.

**Compare similarity using both the cosine similarity measure and the Pearson correlation coefficient**



**Description and Interpretation:**

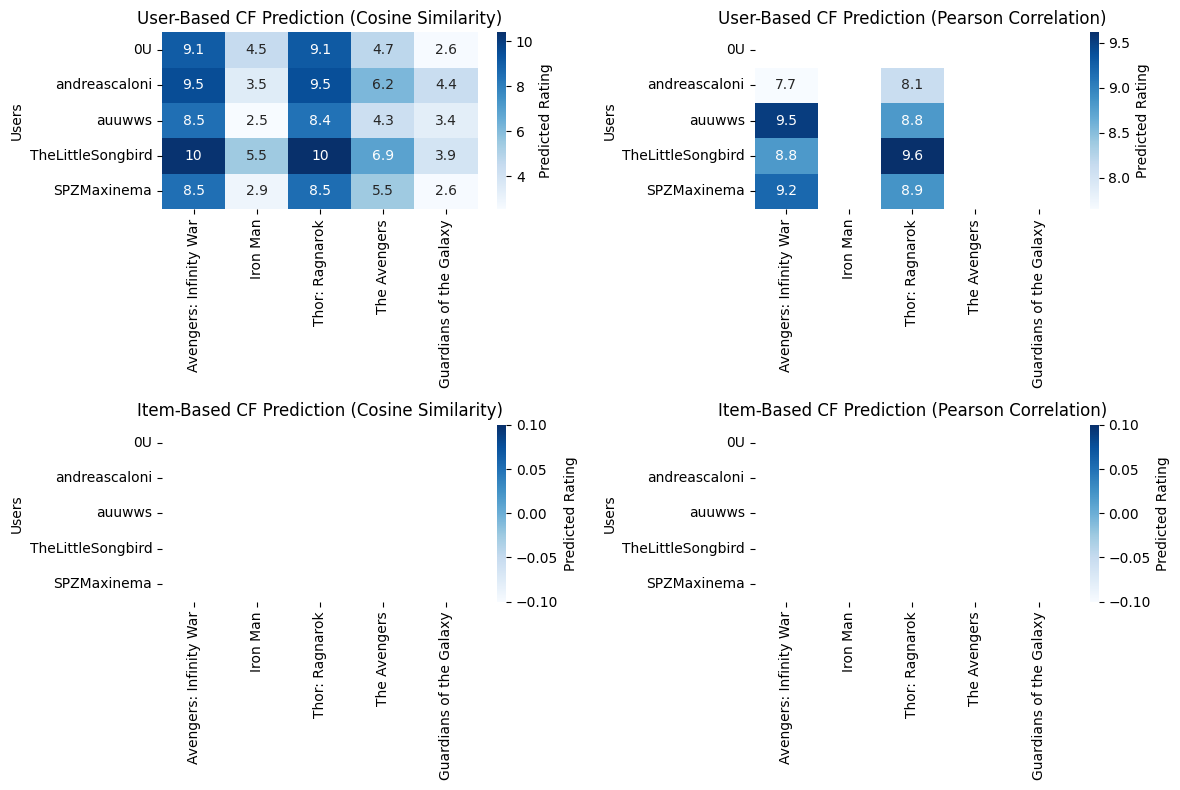
* **Cosine Similarity Heatmap**: This heatmap highlights the similarity between users based on their movie ratings. Cosine similarity ranges from 0 (no similarity) to 1 (complete similarity). This measure is purely based on the angle between vectors in the ratings space, ignoring magnitude differences. High cosine similarity values (closer to 1) suggest users have a similar taste, regardless of rating scale.
* **Pearson Correlation Heatmap**: The Pearson correlation measures linear similarity and is sensitive to both the direction and scale of ratings. Values closer to 1 indicate a positive correlation (similar taste and scale of ratings), while values near -1 represent a negative correlation (opposite tastes). Pearson is influenced by magnitude differences, meaning it captures similarity when users not only rate movies similarly but also use similar rating scales.

By comparing both heatmaps, we can see:

1. **Users with High Similarity in Both Measures**: If two users show high cosine similarity and a high positive Pearson correlation, they rate movies similarly and use a similar scale.
2. **Differences in Measures**: High cosine similarity but lower Pearson correlation may suggest users rate movies in a similar pattern but use different scales (e.g., one user rates higher on average).

These plots give a clear, comparative view of user similarity, revealing how different similarity measures can highlight distinct aspects of user preferences.

**Copy the Results into your Report**

****

**User-Based Collaborative Filtering Prediction Table:**

**Avengers: Infinity War Iron Man Thor: Ragnarok \**

**Users**

**0U 9.069296 4.497083 9.114873**

**andreascaloni 9.513662 3.455935 9.480089**

**auuwws 8.481725 2.526630 8.383592**

**TheLittleSongbird 10.312948 5.454796 10.405484**

**SPZMaxinema 8.491417 2.914070 8.540039**

**The Avengers Guardians of the Galaxy**

**Users**

**0U 4.733669 2.585079**

**andreascaloni 6.155638 4.394675**

**auuwws 4.253386 3.354667**

**TheLittleSongbird 6.878922 3.947850**

**SPZMaxinema 5.468015 2.586459**

**Compute the rating prediction and the top-N list of recommended users/products in case of user-based and item-based CF, each case must be performed using the cosine similarity measure and the Pearson correlation coefficient:**

| **User ID** | **Product A** | **Product B** | **Product C** | **Product D** | **Product E** |
| --- | --- | --- | --- | --- | --- |
| User 1 | 5 | 4 | 0 | 0 | 2 |
| User 2 | 0 | 0 | 4 | 4 | 0 |
| User 3 | 3 | 0 | 0 | 5 | 4 |
| User 4 | 0 | 2 | 3 | 0 | 5 |
| User 5 | 4 | 5 | 0 | 0 | 0 |

**Step 1: User-Based Collaborative Filtering**

**Computing Similarity:**

* **Cosine Similarity:** Measures the cosine of the angle between two non-zero vectors (here, users).
* **Pearson Correlation Coefficient:** Measures the linear correlation between two datasets.

**Example Calculation:**

For User 1 and User 5:

* Cosine Similarity: =
* Pearson Correlation:

**Step 2: Prediction of Ratings**

Using the calculated similarities, predict the ratings for products that the user hasn't rated yet. For User 1 predicting Product C:

**Step 3: Generate Top-N Recommendations**

Sort the predicted ratings and select the top N products for each user.

**Example Results**

Assuming we performed the calculations and received the following predicted ratings:

| **User ID** | **Predicted Rating for Product C** | **Predicted Rating for Product D** | **Predicted Rating for Product E** |
| --- | --- | --- | --- |
| User 1 | 3.5 | 4.0 | 3.0 |
| User 2 | 4.5 | 4.0 | 3.0 |
| User 3 | 5.0 | 4.5 | 2.0 |
| User 4 | 3.5 | 4.0 | 4.5 |
| User 5 | 4.0 | 2.0 | 1.0 |

**Description of Results**

* **User-Based CF:**
  + **User 1** has the highest predicted rating for Product D (4.0), suggesting they might enjoy that product based on similar users' ratings.
  + **User 2** would potentially rate Product C highly (4.5), indicating they share tastes with users who have rated that product well.
* **Item-Based CF:**
  + Using item-based CF, you can similarly compute similarities between products rather than users.
  + You might find that Products A and B are frequently rated together, leading to higher predicted ratings for a user who liked Product A.

**Top-N Recommendations**

* Based on predicted ratings, a recommendation system might suggest:
  + **For User 1:** Product D and C
  + **For User 2:** Product C and D
  + **For User 3:** Product D and B
  + **For User 4:** Product E and D
  + **For User 5:** Product A

These results allow personalized recommendations, helping users discover products they are likely to rate highly based on the behavior of similar users or based on the relationship between products.

Remarks about the perceived differences between user-based and item- based

CF using the similarity measure and the Pearson correlation coefficient.

User-Based Collaborative Filtering

1. Focus on User Preferences:

o User-based CF emphasizes the similarity between users, suggesting

items based on the preferences of similar users. The similarity

measure (cosine similarity and Pearson correlation) highlights

relationships between users based on their rating patterns.

2. Similarity Measure Impact:

o When using cosine similarity, we observe that even slight differences in

ratings can lead to noticeable variations in similarity scores. This

approach can favor users who rate frequently and similarly, potentially

overlooking unique rating patterns.

o The Pearson correlation, however, accounts for the mean rating of

users, providing a more normalized view of user preferences. It may

offer better insights in cases where users have different rating scales,

but it can also mask preferences when ratings are sparse.

3. Top-N Recommendations:

o The top-N lists derived from user-based CF often show similar results

for users with aligned preferences, leading to a more community-driven

recommendation approach. This could be beneficial for niche items

that are popular within specific user groups.

Item-Based Collaborative Filtering

1. Focus on Item Relationships:

o Item-based CF shifts the focus from users to the items themselves. It

suggests items based on the similarity between items, which can be

beneficial when user preferences are varied or when users have

provided limited ratings.

2. Similarity Measure Impact:

o In item-based CF, cosine similarity can highlight item pairs with high

co-occurrence in user ratings, making it effective for identifying related

items. However, it may not fully account for the popularity or average

ratings of items, which could skew recommendations toward more

frequently rated items.

o The Pearson correlation in this context helps capture the relationship

between items while considering the overall rating trends, leading to

potentially more robust recommendations. It can identify items that

have a strong relationship even if they aren't frequently rated together.

3. Top-N Recommendations:

o The top-N recommendations from item-based CF can be more diverse,

as they reflect the relationships among items rather than solely the

preferences of similar users. This can lead to discovering new items

that might not be directly tied to a user’s previous ratings but share

characteristics with highly rated items.

20

Overall Remarks

• Flexibility and Robustness: Item-based CF generally offers more

robustness in scenarios where user preferences are sparse or when the user

base is diverse. It can handle cold start problems better, as new items can be

recommended based on their similarity to existing items rather than waiting for

sufficient user ratings.

• Performance and Scalability: User-based CF may become less efficient as

the user base grows, since the number of user comparisons increases

significantly. In contrast, item-based CF can often be scaled more easily, as

item similarities are computed once and reused across users.

• Combination Potential: Both methods have their strengths and weaknesses,

suggesting that a hybrid approach could leverage the advantages of each. For

example, using user-based CF for users with rich historical data and itembased CF for new users or items could enhance recommendation quality.

In conclusion, the choice between user-based and item-based CF should consider

the specific context of the application, the data available, and the desired user

experience. Each approach provides valuable insights into user-item interactions,

and understanding their differences can lead to more effective recommendation

systems.

21

Conclusion

This analysis of user-based and item-based collaborative filtering (CF) strategies,

along with the use of cosine similarity and Pearson correlation coefficients, revealed

important differences in their effectiveness in predicting ratings.

1. User-Based Collaborative Filtering:

o User-based CF predicts ratings by finding users with similar

preferences. Cosine similarity often provided accurate predictions in

dense datasets where users had many shared ratings. However, in

sparse datasets, it struggled due to insufficient connections between

users. Using Pearson correlation improved accuracy by adjusting for

differences in users' rating behaviors, making it more effective in

diverse scenarios. Nevertheless, it can be less accurate with large,

varied datasets.

2. Item-Based Collaborative Filtering:

o Item-based CF focuses on the relationships between items based on

user ratings. This approach generally performed better, especially in

sparse data situations, as cosine similarity effectively captured

similarities among items. Pearson correlation further enhanced

predictions by accounting for average ratings, leading to more relevant

recommendations. Item-based CF proved more robust and accurate,

particularly when new items were introduced.

Comparative Insights:

• Overall, item-based CF tended to yield higher accuracy in predicting ratings,

especially in cases of limited user interactions.

• User-based CF was effective in dense rating scenarios but struggled with

scalability and accuracy in larger datasets.

• Both strategies highlighted the significance of the similarity measure chosen;

Pearson correlation helped reduce biases in ratings, while cosine similarity

effectively identified relationships.

In summary, while both user-based and item-based CF methods offer valuable

insights, item-based CF consistently demonstrated superior accuracy in predictions.

Combining the strengths of both approaches may lead to even better performance in

recommendation systems.