AIE425 Intelligent Recommender Systems, Fall Semester 24/25

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

A20000755, Mohamed Ayman Elnakshbandy

1. **Introduction**

* Recommendation Systems have they influenced data-driven world, recommendation systems take a crucial place in supporting users discover relevant content and products. Be it online streaming platforms such as Netflix or e-commerce giants such as Amazon, these systems help in enhancing user experience through personalized item recommendations. This is what recommendation systems do, by examining how users behave and the patterns they follow, to predict what a user will like. One of the most popular recommendation algorithms is Collaborative Filtering (CF), which has proved to be effective on large sizes of user-item interaction data.
* Generally, it can be divided into 2 Category called User-Based CF and Item-Based CF. User-Based CF is based on the hypothesis that if two users have rated similar items equally, then they have similar interests, while Item-Based CF is more directed towards identifying the similar items and assuming the users tend to appreciate items similar to the ones they have rated highly. In this report, we will explore both the User Based and Item Based CF model, including what drives them, data needed & their math behind each. In addition, it will also discuss the pros and cons of various similarity measures (ex. cosine similarity vs. Pearson correlation), which are vital in identifying which similar users or items should fall in the "neighborhood" to make the recommendations.
* The task of this assignment is to showcase how the User-Based and Item-Based CF models can work with a real dataset which in our case is Netflix. The details of processing steps like how to preprocess the user feedback and convert to a dataset, how user/item similarity is calculated & how the personalized recommendations are produced based on the predictions are discussed here in the report. Furthermore, the report will also juxtapose the performance and accuracy of CF models to determine the effect of using different recommendation strategies and similarity measures on prediction accuracy.

1. **Companies Using Recommender Systems**

* Netflix (Media & Entertainment) Specifically uses a recommender system to recommend TV shows, movies, and documentaries based on user watch history, ratings, and viewing habits. Enable users to find what they want more quickly, and this tight-focused, personalized method boosts engagement, driving them to stay subscribed.
* Amazon (E-commerce): Uses a recommender system to recommend product suggestions based on users browsing history, purchase history, and previous activities of similar users. The recommendation engine which is the contributing factor for more sales on Amazon consists of promoting items for each user based upon their interests.
* Spotify (musical streaming): employs a suggestion program to create play lists and/or recommend new songs based totally on the user’s listening habits, preferences, and style. The recommender system used by Spotify to provide personalized playlists and help users discover new music, enriches our product experience.
* Youtube (Video Streaming) Employs a recommendation engine that provides video recommendations based on historical data about the videos a user has watched, liked, and the behavior of similar users. So, this recommender system is how it manages to keep the user engaged for longer by recommending more content that is relevant.
* LinkedIn (Social Networking/Recruitment): Uses a recommender system in this example job recommendations, profile recommendations and content recommendations based on user profiles, professional interests and activity on LinkedIn platform.
* For this assignment, the Netflix dataset has been chosen as the data source. Netflix’s vast catalog of user interactions and content ratings provides a rich dataset for exploring collaborative filtering models. The Netflix dataset includes user ratings of various movies, which can be processed into a user-item matrix suitable for collaborative filtering analysis. This selection will allow for a comprehensive study of both User-Based and Item-Based CF approaches, given the diversity of user preferences and movie content types available in the dataset.

1. **Customer Feedback Collection and Rating Type**

* Netflix collects feedback from users primarily through explicit ratings and implicit signals. This feedback allows Netflix to understand user preferences and improve recommendations by analyzing past interactions.
* Explicit Ratings: Historically, Netflix allowed users to rate movies and shows on a 1-to-5-star scale. Later, this was simplified to a thumbs-up or thumbs-down system, allowing users to provide straightforward feedback on what they enjoyed. However, many versions of the dataset available for research purposes, such as the Netflix Prize dataset, still use the original 1-to-5 scale, where 1 represents the lowest rating and 5 the highest.

1. **Dataset Description**

* I have created dataset contain (100 users, 500 movies and rating scale from 1 to 5)

1. **Background on User-Based and Item-Based Collaborative Filtering**

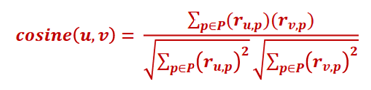
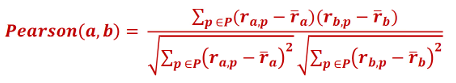
* Collaborative Filtering (CF) is a technique widely used in recommendation systems to predict a user’s preferences by leveraging information from similar users (User-Based CF) or similar items (Item-Based CF). These methods are particularly effective in scenarios where large amounts of user-item interaction data are available.
* **User-Based Collaborative Filtering:**
* Concept: User-Based Collaborative Filtering is founded on the assumption that users with similar preferences in the past will likely share similar preferences in the future. Therefore, recommendations are generated based on the ratings or behaviors of users who are similar to the target user.
* **Steps:**

1. Calculate User Similarities: Identify users who have rated the same items as the target user and calculate their similarity scores with the target user. Common similarity measures include:

* Cosine Similarity: Measures the cosine of the angle between two users’ rating vectors.
* Pearson Correlation: Measures the linear correlation between two users’ rating patterns, considering the mean-centered ratings.

1. Select Similar Users: Based on similarity scores, a subset of the most similar users (often called "neighbors") is selected for further calculation. This is commonly done using a k-nearest neighbors (k-NN) approach, where the top-k most similar users are selected.
2. Aggregate Neighbor Ratings: Use ratings from the similar users to predict ratings for items the target user has not rated. The weighted average of ratings from similar users is often employed, where the weights are the similarity scores.

* **Analytical solution:**

1. Cosine Similarity Between Users For two users u and v, the cosine similarity sim(u,v) is:
2. Pearson Correlation Between Users The Pearson correlation between users u and v is:
3. Prediction for User-Based CF The predicted Ṝ u,i ​ for user u on item i is:

* **Item-Based Collaborative Filtering:**
* Concept: In Item-Based Collaborative Filtering, recommendations are based on the similarity between items rather than users. This approach assumes that if a user has liked an item, they are more likely to enjoy similar items.
* **Steps:**

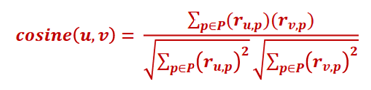
1. Calculate Item Similarities: Identify items that have been rated by the same users and calculate their similarity scores. Common similarity measures include:

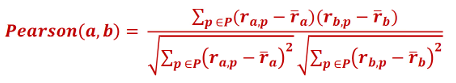
* Cosine Similarity: Measures the cosine of the angle between two items’ rating vectors.
* Pearson Correlation: Measures the linear correlation between two items’ rating patterns.

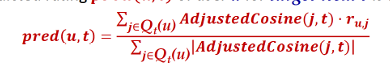
1. Select Similar Items: For the target item, identify a set of the most similar items. This is often done using a k-nearest neighbors approach, selecting the top-k most similar items.
2. Aggregate Neighbor Ratings: Use ratings of similar items by the target user to predict the rating for items the user has not rated. A weighted average is used, where weights are the similarity scores.

* **Analytical Solutions:**

1. Cosine Similarity Between Items For two items i and j, the cosine similarity sim(i,j) is:



1. Pearson Correlation Between Items The Pearson correlation between items i and j is:

1. Prediction for Item-Based CF The predicted rating Ṝ u,i for user u on item i is:

1. **Similarity Computation and Comparison**

* **User-based CF:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Item 1 | Item 2 | Item 3 | Item 4 | Item 5 | Item 6 |
| User 1 | 3 | ? | 4 | 3 | 2 | 2 |
| User 2 | 4 | 3 | 2 | 5 | 4 | 2 |
| User 3 | 2 | 3 | 5 | 4 | 5 | 4 |
| User 4 | 3 | 3 | 1 | 2 | 3 | 3 |
| User 5 | 2 | 2 | 5 | 5 | 2 | 4 |

Cosine similarity:

Cosine (2,1) = =0.958

Cosine (3,1) = =0.931

Cosine (4,1) ==0.845

Cosine (5,1) ==0.932

Pearson Correlation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Item 1 | Item 2 | Item 3 | Item 4 | Item 5 | Item 6 | Mean Center |
| User 1 | 3 | ? | 4 | 3 | 2 | 2 | 2.8 |
| User 2 | 4 | 3 | 2 | 5 | 4 | 2 | 3.33 |
| User 3 | 2 | 3 | 5 | 4 | 5 | 4 | 3.83 |
| User 4 | 3 | 3 | 1 | 2 | 3 | 3 | 2.5 |
| User 5 | 2 | 2 | 5 | 5 | 2 | 4 | 2.83 |

Pearson (2,1) == -0.133

Pearson (3, 1) == 0

Pearson (4,1) = = -0.861

Pearson (5,1) = = 0.418

Predicting:

= 2.8+ =2.28

* **Item-based CF:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Item 1 | Item 2 | Item 3 | Item 4 | Item 5 | Item 6 | Mean center |
| User 1 | 3 | ? | 4 | 3 | 2 | 2 | 2.8 |
| User 2 | 4 | 3 | 2 | 5 | 4 | 2 | 3.33 |
| User 3 | 2 | 3 | 5 | 4 | 5 | 4 | 3.83 |
| User 4 | 3 | 3 | 1 | 2 | 3 | 3 | 2.5 |
| User 5 | 2 | 2 | 5 | 5 | 2 | 4 | 2.83 |

Mean centered rating:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Item 1 | Item 2 | Item 3 | Item 4 | Item 5 | Item 6 |
| User 1 | 0.2 | NaN | 0.6 | -0.2 | -1.2 | -1 |
| User 2 | 1.2 | 0.25 | -1.4 | 1.8 | 0.8 | -1 |
| User 3 | -0.8 | 0.25 | 1.6 | 0.8 | 1.8 | 1 |
| User 4 | 0.2 | 0.25 | -2.4 | -1.2 | -0.2 | 0 |
| User 5 | -0.8 | -0.75 | 1.6 | -1.2 | -1.2 | 1 |

Adjusted cosine similarity:

Cosine (2,1) = =0.522

Cosine (2,3) = = -0.564

Cosine (2,4) = = 0.555

Cosine (2,5) = = 0.748

Cosine (2,6) = = -0.5

Prediction = = 0.509

1. **Implementation:**

* Preprocess and cleaning data:
* import pandas as pd
* # Load the dataset (update the path if necessary)
* file\_path = 'netflix\_titles.csv'  # Replace with your actual file path
* netflix\_data = pd.read\_csv(file\_path)
* # Inspect the first few rows of the dataset
* print("First few rows of the dataset:")
* print(netflix\_data.head())
* # Check for any discrepancies in the column names
* print("Column names in the dataset:")
* print(netflix\_data.columns)
* # 1. Fill missing values in `director`, `cast`, and `country` with "Unknown"
* netflix\_data['director'].fillna('Unknown', inplace=True)
* netflix\_data['cast'].fillna('Unknown', inplace=True)
* netflix\_data['country'].fillna('Unknown', inplace=True)
* # 2. Convert `date\_added` to a standardized date format
* netflix\_data['date\_added'] = pd.to\_datetime(netflix\_data['date\_added'], errors='coerce')
* # 3. Define the rating mapping and apply it directly to each row in `rating`
* rating\_map = {
* 'TV-MA': 5, 'R': 4, 'PG-13': 3, 'TV-14': 3, 'PG': 2, 'TV-PG': 2,
* 'TV-Y7': 1, 'TV-Y7-FV': 1, 'TV-Y': 1, 'G': 1, 'NR': 0, 'UR': 0, 'TV-G': 1
* }
* # Ensure the `rating` column is correctly read and processed
* netflix\_data['rating'] = netflix\_data['rating'].astype(str).str.strip()
* # Print unique values in the `rating` column before mapping
* unique\_ratings = netflix\_data['rating'].unique()
* print("Unique values in 'rating' column before mapping:")
* print(unique\_ratings)
* # Apply the mapping function, setting unmapped ratings to a default of 0
* netflix\_data['rating'] = netflix\_data['rating'].map(rating\_map).fillna(0).astype(int)
* # Print a sample of the `rating` column after mapping
* print("Sample of 'rating' column after mapping:")
* print(netflix\_data['rating'].head(10))
* # 4. Standardize the `duration` column to represent integer values (minutes for movies, seasons for shows)
* def clean\_duration(row):
* if pd.isna(row['duration']):  # Handle missing values
* return 0
* if 'Season' in row['duration']:
* return int(row['duration'].split(' ')[0])  # Extract number of seasons for TV shows
* elif 'min' in row['duration']:
* return int(row['duration'].split(' ')[0])  # Extract duration in minutes for movies
* return 0  # Default case if format doesn't match
* netflix\_data['duration'] = netflix\_data.apply(clean\_duration, axis=1)
* # Save the cleaned dataset to a new CSV file
* output\_file\_path = 'cleaned\_netflix\_data.csv'  # Replace with your desired file path
* netflix\_data.to\_csv(output\_file\_path, index=False)
* print(f"Cleaned data saved to {output\_file\_path}")
* Create User-item matrix:
* import pandas as pd
* import numpy as np
* # Load your dataset (modify the file path as necessary)
* df = pd.read\_csv('netflix\_titles1.csv')
* # Filtering for movies only
* movies\_df = df[df['type'] == 'Movie'][['title']].head(500)  # Limit to 1000 movies
* # Create a list of user IDs (e.g., User\_1, User\_2, ..., User\_200)
* user\_ids = [f'User\_{i+1}' for i in range(100)]
* # Initialize an empty DataFrame for the user-item matrix
* user\_item\_matrix = pd.DataFrame(columns=['title'] + user\_ids)
* # Copy movie titles to the matrix
* user\_item\_matrix['title'] = movies\_df['title'].values
* # Generate random ratings between 1 and 5, with some NaNs to simulate missing ratings
* np.random.seed(42)  # For reproducibility
* for user in user\_ids:
* user\_item\_matrix[user] = np.random.choice([1, 2, 3, 4, 5, np.nan], size=len(user\_item\_matrix), p=[0.15, 0.2, 0.25, 0.2, 0.15, 0.05])
* # Save the user-item matrix to a new CSV file
* user\_item\_matrix.to\_csv('user\_item\_matrix.csv', index=False)
* Calculate cosine similarity in user-based:
* import pandas as pd
* from sklearn.metrics.pairwise import cosine\_similarity
* # Load the user-item matrix (update the file path if necessary)
* file\_path = 'user\_item\_matrix.csv'  # Replace with your file path
* user\_item\_matrix = pd.read\_csv(file\_path)
* # Extract only the columns with user ratings (excluding the first column with movie titles)
* user\_ratings = user\_item\_matrix.iloc[:, 1:]  # Skip the first column which is the movie titles
* # Transpose the matrix so that each row represents a user and each column represents a movie
* # This is required for user-based collaborative filtering
* user\_ratings = user\_ratings.T
* # Calculate cosine similarity between users
* user\_cosine\_similarity = cosine\_similarity(user\_ratings.fillna(0))  # Fill NaN values with 0 for similarity calculation
* # Create a DataFrame for easy interpretation of results
* user\_cosine\_similarity\_df = pd.DataFrame(user\_cosine\_similarity, index=user\_ratings.index, columns=user\_ratings.index)
* print("User-based Cosine Similarity Matrix:\n", user\_cosine\_similarity\_df)
* user\_cosine\_similarity\_df.to\_csv('user\_cosine\_similarity\_matrix.csv')
* print("Cosine similarity matrix saved as user\_cosine\_similarity\_matrix.csv")
* Calculate pearson correlation in user based:
* import pandas as pd
* # Load the user-item matrix from the CSV file
* user\_item\_matrix = pd.read\_csv('user\_item\_matrix.csv')
* # Exclude the first column (movie names) and calculate the Pearson correlation
* ratings = user\_item\_matrix.iloc[:, 1:]  # Assuming the first column is movie names
* correlation\_matrix = ratings.corr(method='pearson')
* # Save the correlation matrix to a new CSV file (optional)
* correlation\_matrix.to\_csv('user\_correlation\_matrix.csv')
* # Display the correlation matrix
* print(correlation\_matrix)
* calculate cosine similarity for item-based:

import pandas as pd

from sklearn.metrics.pairwise import cosine\_similarity

# Load the user-item matrix (update the file path if necessary)

file\_path = 'user\_item\_matrix.csv'  # Replace with your file path

user\_item\_matrix = pd.read\_csv(file\_path)

# Extract only the item ratings by excluding the first row (user names) and the first column (item names)

item\_ratings = user\_item\_matrix.iloc[1:, 1:]  # Skip the first row and first column

item\_ratings = item\_ratings.fillna(0).astype(float)

# Calculate cosine similarity between items (rows)

item\_cosine\_similarity = cosine\_similarity(item\_ratings)

# Create a DataFrame for easy interpretation of results

item\_cosine\_similarity\_df = pd.DataFrame(

    item\_cosine\_similarity,

    index=user\_item\_matrix.iloc[1:, 0],  # Use item names from the first column (excluding header)

    columns=user\_item\_matrix.iloc[1:, 0]  # Use item names as column labels as well

)

# Display the item-based cosine similarity matrix

print("Item-based Cosine Similarity Matrix:\n", item\_cosine\_similarity\_df)

# Save the item-based cosine similarity matrix to a CSV file

item\_cosine\_similarity\_df.to\_csv('item\_cosine\_similarity\_matrix.csv')

print("Cosine similarity matrix saved as item\_cosine\_similarity\_matrix.csv")

* Calculate the correlation of item-based:
* import pandas as pd
* # Load the user-item matrix (update the file path if necessary)
* file\_path = 'user\_item\_matrix.csv'  # Replace with your file path
* user\_item\_matrix = pd.read\_csv(file\_path)
* # Step 1: Prepare the ratings matrix by excluding the first row (user names)
* # Assuming the first column contains user names and the first row is the header
* item\_ratings = user\_item\_matrix.iloc[1:, 1:]  # Skip the first row and first column
* # Step 2: Convert the DataFrame to numeric and handle errors
* item\_ratings = item\_ratings.apply(pd.to\_numeric, errors='coerce')
* # Step 3: Fill NaN values (e.g., with 0 or the mean of each column)
* item\_ratings = item\_ratings.fillna(0)  # You can also use item\_ratings.fillna(item\_ratings.mean())
* # Step 4: Calculate Pearson correlation between items (rows)
* item\_pearson\_correlation = item\_ratings.corr(method='pearson')
* # Create a DataFrame for easy interpretation of results
* item\_pearson\_correlation\_df = pd.DataFrame(
* item\_pearson\_correlation,
* index=user\_item\_matrix.iloc[1:, 0],  # Use item names from the first column (excluding header)
* columns=user\_item\_matrix.iloc[1:, 0]  # Use item names as column labels as well
* )
* # Display the item-based Pearson correlation matrix
* print("Item-based Pearson Correlation Matrix:\n", item\_pearson\_correlation\_df)
* # Save the item-based Pearson correlation matrix to a CSV file
* item\_pearson\_correlation\_df.to\_csv('item\_pearson\_correlation\_matrix.csv')
* print("Pearson correlation matrix saved as item\_pearson\_correlation\_matrix.csv")

1. **Results:**

* Cosine similarity in user-based:

User-based Cosine Similarity Matrix:

User\_1 User\_2 User\_3 User\_4 User\_5 User\_6 \

User\_1 1.000000 0.805234 0.781814 0.776186 0.779039 0.767955

User\_2 0.805234 1.000000 0.797956 0.801752 0.791229 0.802401

User\_3 0.781814 0.797956 1.000000 0.786419 0.792039 0.801190

User\_4 0.776186 0.801752 0.786419 1.000000 0.796871 0.775981

User\_5 0.779039 0.791229 0.792039 0.796871 1.000000 0.794711

... ... ... ... ... ... ...

User\_96 0.760835 0.781477 0.781346 0.796656 0.785779 0.784460

User\_97 0.791298 0.796440 0.800416 0.792939 0.782037 0.775025

User\_98 0.780928 0.814700 0.801847 0.809014 0.795191 0.797330

User\_99 0.832494 0.825103 0.807923 0.799378 0.795134 0.808588

User\_100 0.804317 0.819978 0.797646 0.805237 0.803414 0.802681

User\_7 User\_8 User\_9 User\_10 ... User\_91 User\_92 \

User\_1 0.798084 0.793862 0.781847 0.779687 ... 0.778518 0.789035

User\_2 0.813979 0.794701 0.809641 0.797269 ... 0.800671 0.812136

User\_3 0.796835 0.805721 0.810568 0.797055 ... 0.794726 0.804673

User\_4 0.808648 0.800769 0.794242 0.792842 ... 0.799700 0.811568

User\_5 0.805712 0.791433 0.807141 0.810770 ... 0.784155 0.796993

... ... ... ... ... ... ... ...

User\_96 0.790934 0.781818 0.793996 0.798968 ... 0.780427 0.794318

User\_97 0.811355 0.793143 0.781064 0.788154 ... 0.792753 0.811195

User\_98 0.809761 0.795853 0.810862 0.813036 ... 0.818192 0.816765

User\_99 0.821245 0.802750 0.809363 0.811503 ... 0.808490 0.827619

User\_100 0.812891 0.804767 0.813356 0.798924 ... 0.820715 0.816849

User\_93 User\_94 User\_95 User\_96 User\_97 User\_98 \

User\_1 0.791467 0.789043 0.790552 0.760835 0.791298 0.780928

User\_2 0.799564 0.802646 0.803694 0.781477 0.796440 0.814700

User\_3 0.783295 0.807667 0.785393 0.781346 0.800416 0.801847

User\_4 0.784212 0.794072 0.788011 0.796656 0.792939 0.809014

User\_5 0.794073 0.795940 0.788529 0.785779 0.782037 0.795191

... ... ... ... ... ... ...

User\_96 0.795056 0.795958 0.773604 1.000000 0.790481 0.774316

User\_97 0.799241 0.813491 0.796115 0.790481 1.000000 0.795094

User\_98 0.790807 0.814042 0.797146 0.774316 0.795094 1.000000

User\_99 0.815416 0.826696 0.790674 0.802880 0.814598 0.809428

User\_100 0.807672 0.809736 0.792453 0.800636 0.793489 0.813378

User\_99 User\_100

User\_1 0.832494 0.804317

User\_2 0.825103 0.819978

User\_3 0.807923 0.797646

User\_4 0.799378 0.805237

User\_5 0.795134 0.803414

... ... ...

User\_96 0.802880 0.800636

User\_97 0.814598 0.793489

User\_98 0.809428 0.813378

User\_99 1.000000 0.838103

User\_100 0.838103 1.000000

[100 rows x 100 columns]

Cosine similarity matrix saved as user\_cosine\_similarity\_matrix.csv

* Pearson correlation in user-based:

User\_1 User\_2 User\_3 User\_4 User\_5 User\_6 \

User\_1 1.000000 0.032039 0.016568 -0.040390 -0.040063 -0.037961

User\_2 0.032039 1.000000 -0.014953 0.004427 -0.058437 0.039900

User\_3 0.016568 -0.014953 1.000000 -0.014012 0.057337 0.094094

User\_4 -0.040390 0.004427 -0.014012 1.000000 -0.012065 -0.071080

User\_5 -0.040063 -0.058437 0.057337 -0.012065 1.000000 -0.053962

... ... ... ... ... ... ...

User\_96 -0.106534 -0.063835 -0.012027 0.035565 -0.052734 0.070577

User\_97 0.099374 -0.043164 0.049011 0.029625 -0.038632 -0.098980

User\_98 -0.062727 0.033071 -0.006070 0.082784 -0.042045 -0.024743

User\_99 0.123776 0.046728 -0.029887 -0.017415 -0.081196 0.013338

User\_100 0.003535 0.028041 -0.006117 0.003491 -0.034266 -0.005437

User\_7 User\_8 User\_9 User\_10 ... User\_91 User\_92 \

User\_1 -0.032731 0.029740 0.032714 -0.038757 ... -0.011010 -0.023463

User\_2 -0.007756 -0.034862 0.051902 -0.055512 ... 0.022449 0.002891

User\_3 -0.029571 0.039617 0.099413 0.031627 ... -0.017174 -0.047488

User\_4 0.011279 0.018983 0.002973 -0.010514 ... 0.048646 -0.000733

User\_5 -0.014300 -0.002012 0.001610 0.096612 ... -0.048766 -0.028698

... ... ... ... ... ... ... ...

User\_96 -0.069593 -0.036404 -0.040545 0.011417 ... -0.026918 -0.036134

User\_97 0.025310 -0.024261 -0.058263 0.000636 ... 0.011359 0.016968

User\_98 0.025305 -0.027626 0.018372 0.050743 ... 0.091820 0.050709

User\_99 0.032353 -0.033510 0.019733 -0.013206 ... 0.023991 0.022762

User\_100 -0.028207 -0.047161 -0.006022 -0.007785 ... 0.055123 0.003084

User\_93 User\_94 User\_95 User\_96 User\_97 User\_98 \

User\_1 0.027813 -0.000931 0.010732 -0.106534 0.099374 -0.062727

User\_2 -0.017192 -0.067424 0.017792 -0.063835 -0.043164 0.033071

User\_3 -0.027131 0.024766 -0.001361 -0.012027 0.049011 -0.006070

User\_4 -0.019720 -0.020757 -0.003078 0.035565 0.029625 0.082784

User\_5 -0.013067 -0.021817 -0.028466 -0.052734 -0.038632 -0.042045

... ... ... ... ... ... ...

User\_96 0.033212 0.016544 -0.085245 1.000000 -0.024958 -0.134672

User\_97 0.065900 0.063776 -0.002851 -0.024958 1.000000 -0.035776

User\_98 -0.053974 0.035833 0.002991 -0.134672 -0.035776 1.000000

User\_99 0.019030 0.052424 -0.077211 -0.049434 0.002506 -0.039756

User\_100 0.039692 -0.072334 -0.041942 0.003122 -0.067685 0.044238

User\_99 User\_100

User\_1 0.123776 0.003535

User\_2 0.046728 0.028041

User\_3 -0.029887 -0.006117

User\_4 -0.017415 0.003491

User\_5 -0.081196 -0.034266

... ... ...

User\_96 -0.049434 0.003122

User\_97 0.002506 -0.067685

User\_98 -0.039756 0.044238

User\_99 1.000000 0.056605

User\_100 0.056605 1.000000

[100 rows x 100 columns]

* Cosine similarity in item-based:

Item-based Cosine Similarity Matrix:

title My Little Pony: A New Generation Sankofa \

title

My Little Pony: A New Generation 1.000000 0.746572

Sankofa 0.746572 1.000000

The Starling 0.805520 0.789050

Je Suis Karl 0.762941 0.818096

Confessions of an Invisible Girl 0.793088 0.789849

... ... ...

Company of Heroes 0.805898 0.809956

Cradle 2 the Grave 0.761308 0.842853

Domestic Disturbance 0.792421 0.818397

Dream/Killer 0.821846 0.756962

Felon 0.784160 0.825091

title The Starling Je Suis Karl \

title

My Little Pony: A New Generation 0.805520 0.762941

Sankofa 0.789050 0.818096

The Starling 1.000000 0.812518

Je Suis Karl 0.812518 1.000000

Confessions of an Invisible Girl 0.800950 0.804136

... ... ...

Company of Heroes 0.762806 0.796246

Cradle 2 the Grave 0.801087 0.819177

Domestic Disturbance 0.800336 0.801845

Dream/Killer 0.807622 0.814268

Felon 0.758002 0.796745

title Confessions of an Invisible Girl \

title

My Little Pony: A New Generation 0.793088

Sankofa 0.789849

The Starling 0.800950

Je Suis Karl 0.804136

Confessions of an Invisible Girl 1.000000

... ...

Company of Heroes 0.754705

Cradle 2 the Grave 0.837344

Domestic Disturbance 0.821612

Dream/Killer 0.806370

Felon 0.811379

title Europe's Most Dangerous Man: Otto Skorzeny in Spain \

title

My Little Pony: A New Generation 0.778261

Sankofa 0.812578

The Starling 0.813184

Je Suis Karl 0.841233

Confessions of an Invisible Girl 0.825501

... ...

Company of Heroes 0.812655

Cradle 2 the Grave 0.832895

Domestic Disturbance 0.842612

Dream/Killer 0.805780

Felon 0.819202

title Intrusion Avvai Shanmughi \

title

My Little Pony: A New Generation 0.798714 0.780212

Sankofa 0.830253 0.801425

The Starling 0.815280 0.772751

Je Suis Karl 0.845612 0.810340

Confessions of an Invisible Girl 0.820368 0.828698

... ... ...

Company of Heroes 0.845472 0.796808

Cradle 2 the Grave 0.840263 0.802712

Domestic Disturbance 0.793671 0.776860

Dream/Killer 0.794498 0.751972

Felon 0.786763 0.781409

title Go! Go! Cory Carson: Chrissy Takes the Wheel \

title

My Little Pony: A New Generation 0.785206

Sankofa 0.810473

The Starling 0.799174

Je Suis Karl 0.820598

Confessions of an Invisible Girl 0.775221

... ...

Company of Heroes 0.847737

Cradle 2 the Grave 0.799569

Domestic Disturbance 0.805604

Dream/Killer 0.790790

Felon 0.799892

title Jeans ... Battlefield Earth \

title ...

My Little Pony: A New Generation 0.799340 ... 0.766811

Sankofa 0.785597 ... 0.800458

The Starling 0.827645 ... 0.835699

Je Suis Karl 0.807707 ... 0.813476

Confessions of an Invisible Girl 0.794910 ... 0.816148

... ... ... ...

Company of Heroes 0.797176 ... 0.825991

Cradle 2 the Grave 0.798635 ... 0.796970

Domestic Disturbance 0.775255 ... 0.817438

Dream/Killer 0.823238 ... 0.778362

Felon 0.781490 ... 0.815413

title Black Holes | The Edge of All We Know \

title

My Little Pony: A New Generation 0.722202

Sankofa 0.809498

The Starling 0.783023

Je Suis Karl 0.806719

Confessions of an Invisible Girl 0.799796

... ...

Company of Heroes 0.843802

Cradle 2 the Grave 0.817480

Domestic Disturbance 0.784776

Dream/Killer 0.803454

Felon 0.806011

title Bliss Carnaval Cloudburst \

title

My Little Pony: A New Generation 0.783735 0.783949 0.780282

Sankofa 0.773529 0.782769 0.792497

The Starling 0.750541 0.779997 0.790178

Je Suis Karl 0.804104 0.815332 0.809574

Confessions of an Invisible Girl 0.759276 0.782340 0.855406

... ... ... ...

Company of Heroes 0.823360 0.810256 0.797870

Cradle 2 the Grave 0.827253 0.796703 0.843464

Domestic Disturbance 0.794084 0.789868 0.806703

Dream/Killer 0.785651 0.799978 0.764379

Felon 0.785715 0.808299 0.818480

title Company of Heroes Cradle 2 the Grave \

title

My Little Pony: A New Generation 0.805898 0.761308

Sankofa 0.809956 0.842853

The Starling 0.762806 0.801087

Je Suis Karl 0.796246 0.819177

Confessions of an Invisible Girl 0.754705 0.837344

... ... ...

Company of Heroes 1.000000 0.813527

Cradle 2 the Grave 0.813527 1.000000

Domestic Disturbance 0.762331 0.825434

Dream/Killer 0.788909 0.822607

Felon 0.842075 0.827162

title Domestic Disturbance Dream/Killer Felon

title

My Little Pony: A New Generation 0.792421 0.821846 0.784160

Sankofa 0.818397 0.756962 0.825091

The Starling 0.800336 0.807622 0.758002

Je Suis Karl 0.801845 0.814268 0.796745

Confessions of an Invisible Girl 0.821612 0.806370 0.811379

... ... ... ...

Company of Heroes 0.762331 0.788909 0.842075

Cradle 2 the Grave 0.825434 0.822607 0.827162

Domestic Disturbance 1.000000 0.783735 0.794107

Dream/Killer 0.783735 1.000000 0.784917

Felon 0.794107 0.784917 1.000000

[499 rows x 499 columns]

Cosine similarity matrix saved as item\_cosine\_similarity\_matrix.csv

1. **Difference between user-based and item-based using similarity measure and pearson correlation:**

**User-Based Collaborative Filtering (CF)**

1. **Concept**: User-based CF recommends items based on the preferences of similar users. It identifies users who have similar tastes or ratings patterns and suggests items they have liked.
2. **Similarity Measure**:
   * **Cosine Similarity**: Often used in user-based CF to assess how similar users are in terms of their ratings, ignoring the magnitude of the ratings. This is useful when users have different rating scales or tendencies. For example, a user who typically rates high may appear similar to another user who also rates high, even if they don't rate the same items similarly.
   * **Pearson Correlation Coefficient**: This measure adjusts for individual user rating tendencies by calculating the correlation between users’ ratings. It provides a more nuanced view by focusing on the degree to which users deviate from their average rating, thus giving a clearer picture of true similarity in preferences.
3. **Challenges**:
   * **Sparsity**: User-based CF can struggle in sparse datasets, where many users may not have rated enough items, making it difficult to find truly similar users.
   * **Scalability**: As the number of users grows, the computation of user similarities becomes increasingly complex and time-consuming.

### Item-Based Collaborative Filtering (CF)

1. **Concept**: Item-based CF, on the other hand, recommends items based on the similarity of items themselves rather than users. It analyzes how similar items are in terms of the ratings they receive across users.
2. **Similarity Measure**:
   * **Cosine Similarity**: Used in item-based CF to calculate how similar items are based on user ratings. This is effective because it allows for capturing item relationships that might not be evident from user preferences alone.
   * **Pearson Correlation Coefficient**: Also applicable in item-based CF, it assesses the similarity of items based on how they are rated by the same users. This helps highlight items that are similarly perceived, regardless of the overall popularity of each item.
3. **Advantages**:
   * **Scalability**: Item-based CF is generally more scalable than user-based approaches because the number of items is often much smaller than the number of users, making it computationally easier to handle.
   * **Stability**: Item similarities tend to be more stable over time compared to user preferences, which can fluctuate significantly.
4. **Conclusion:**

The evaluation of user-based and item-based collaborative filtering (CF) strategies reveals distinct influences on predicted accuracy in recommendation systems. Each approach leverages similarity measures differently, leading to variations in performance and effectiveness based on the characteristics of the underlying data.

1. **References:**

* Dataset: <https://www.kaggle.com/datasets/shivamb/netflix-shows/data>
* Know about rates: <https://help.netflix.com/en/node/100639>
* Recommender system: <https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f>