

CPS 842 Movie Recommender System Project Report

How to Run the Program?

```
pip install -r requirements.txt
```

```
python main.py
```

On your browser open: <http://127.0.0.1:5000/>

Or

Live Website: _____

Detailed Design of System

Dataset

There are 100,000 ratings ranked from (1-5) from 943 users on 1682 movies.

Each user has rated at least 20 movies.

In u.data, I use the following fields (user_id | movie_id | rating).

In u.item, I use the following fields (movie_id | movie_title).

<https://grouplens.org/datasets/movielens/100k/>

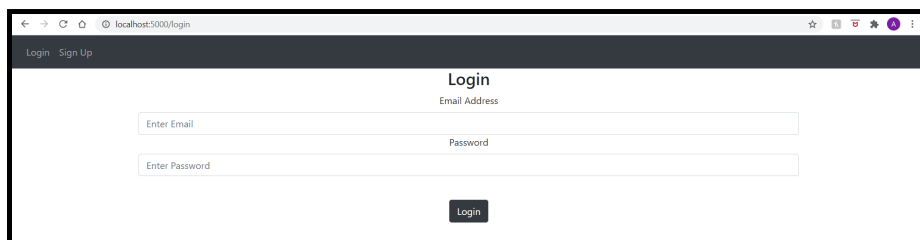
I generated the user similarity matrix using generate_user_similarity_matrix.py which reads data from u.data and u.item and outputs the results to user_similarity_matrix.txt. Please open user_similarity_matrix.txt using Notepad++ for proper formatting.

Frontend

Includes basic user management functions that allow users to sign up for an account, login to their account, and logout. I have implemented error handling to these functions.

When a user signs up for a new account, their first name and email characters must be greater than a certain number. Also, their passwords must match, and a user cannot use an email that already exists in the database. Passwords are hashed using the SHA-256 algorithm.

Users cannot access movies.html until they login to their account, which means they have to sign up for an account to login.

A screenshot of a web browser window showing a login form. The browser's address bar displays 'localhost:5000/login'. The page has a dark header with 'Login' and 'Sign Up' links. The main content area is titled 'Login' and contains two input fields: 'Email Address' with a placeholder 'Enter Email' and 'Password' with a placeholder 'Enter Password'. A 'Login' button is positioned below the password field.

login.html

[Login](#) [Sign Up](#)

Sign Up

First Name

Email Address

Enter Email

Password

Enter Password

Password (Confirm)

Confirm Password

sign_up.html

On movies.html, a user can rate up to 20 movies.

Division by zero errors can happen when a user gives the same rating for all movies handled in the backend by assigning a zero value so the program does not crash.

If a user rates the same movie more than once, the last rating will be used in the program's backend.

If a user enters no input, no recommendations will be given.

If a user enters a movie with no corresponding rating or vice versa, an error message will be displayed asking them to enter the movie/rating correctly.

[Movies](#) [Logout](#)

Logged in successfully!

Movies

Movie Options	Your Ratings
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen

movies.html

The screenshot shows a web application for movie recommendations. At the top, there are two identical input sections, each with a dropdown menu labeled 'Please Pick a Movie' and a button labeled 'Haven't Seen'. Below these is a dark button labeled 'Click Here for Recommended Movies'. Underneath is a red button labeled 'Clear Results'. The main content area is titled 'Your Inputs:' and lists ten movies with their corresponding scores: Toy Story (1995) - 5, GoldenEye (1995) - 3, Four Rooms (1995) - 4, Get Shorty (1995) - 3, Copycat (1995) - 3, Shanghai Triad (1995) - 5, Twelve Monkeys (1995) - 4, Babe (1995) - 1, Dead Man Walking (1995) - 5, and Richard III (1995) - 3. Below this list is the section 'Your Results:', which displays three recommended movies: Star Wars (1977) with a score of 4.0, Fargo (1996) with a score of 3.08, and Raiders of the Lost Ark (1981) with a score of 2.84.

movies.html

Backend

The backend ([website/collaborative_filter.py](#)) uses the user-based collaborative filtering approach to predict the rating of movies the user has not rated based on items previously rated by other users.

For the user similarity calculation, I used the Pearson correlation coefficient which gives a result from -1 to +1 when comparing the movie ratings of each user in the dataset to the user's input. Where -1 is a perfect negative correlation between two users' ratings, 0 indicates no relationship between the two users' ratings, and +1 is a perfect positive correlation between two users' ratings.

I considered only similarity calculation greater than or equal to 0.5 in the weighted-sum aggregate function for this project.

For each movie the user has not watched, I calculated the weighted-sum which predicts the movie's rating between 0 to 5 using the similarity scores and movie ratings of the users considered. I returned the top 3 movies with the highest rating.

Functions in the System

In ([website/collaborative_filter.py](#))

```
def main(selected_movies, ratings, movie):
```

The driver code which calls each of the following functions in sequential order. Then the top three scores and their corresponding movies are returned and displayed on movies.html if recommendations have been found; in other words, similarity scores > 0.5 were calculated.

```
def create_matrix(selected_movies, ratings, movies):
```

Returns a matrix of movie scores from u.data where each row is a user id, and their rating for each movie represents the columns.

```
def calculate_average(matrix, user_ratings):
```

Calculates and returns the average rating between the users' input and each user in the dataset for movies both users have rated.

```
def calculate_similarity_scores(matrix, user_ratings, averages):
```

Calculates and returns the Pearson correlation coefficient (a score between -1 and +1) for the user and each user in the dataset using their average ratings and the movie scores both users have rated.

```
def calculate_weighted_sum_rating(matrix, similarity_scores):
```

Calculates and returns the score for each movie the user has not rated using similarity scores > 0.5.

References:

For the user interface: <https://github.com/techwithtim/Flask-Web-App-Tutorial>

For the dataset: <https://grouplens.org/datasets/movielens/100k/>