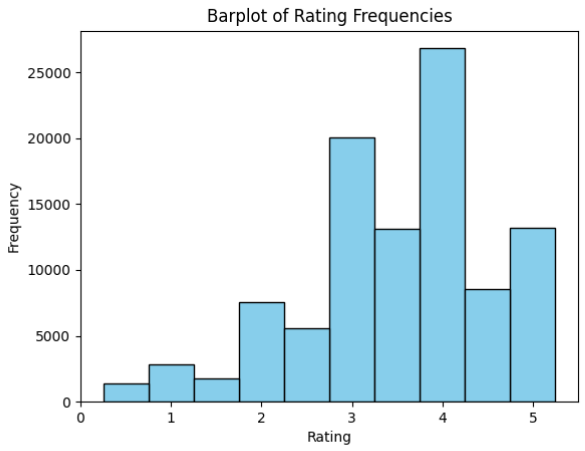
IRRS Lab 5

Andreja Andrejic, Julian Luneburg

December 2024

1 Statistics of the data

In the first part of the exercise, we calculate some statistics describing movie ratings in the file *statistics_1.py*. The provided code analyzes the movie ratings to calculate key statistics and insights. First, it computes the average movie rating and the variance of the ratings, providing an overall sense of how movies are rated and how much the ratings vary. The average movie rating is 3.50 and the variance is 1.09. Next, the code visualizes the distribution of ratings using a bar chart, showing how frequently each rating score appears.



It also determines the number of unique users (610) in the dataset and calculates the average number of movies rated per user (165.3), giving an idea of user activity levels. Finally, the code examines the genres of the movies, counts how often each genre appears, and ranks them by frequency, revealing the most popular genres. It can be seen that the top four genres are in descending order: Drama, Comedy, Thriller and Action. Together, these statistics and visualizations help summarize the behavior of users, the ratings they provide, and the popularity of different movie genres.

2 Explanation of the recommender systems

2.1 Naive Recommender

Naive recommender is created as a baseline model. It recommends movies by its overall rating, thus all users get the same recommendations. The complexity of this model is $O(n) = n * log_2 n$.

2.2 User-based Recommender

The user-based recommender is a user-based collaborative filter recommendation system. The main difficulties for this part were creating a robust similarity metric and efficient execution of the algorithm.
Since the rating data has a lot of missing values, we could not simply use the Pearson Correlation, we adjusted the algorithm. To be able to run the algorithm at all, we only considered the similarity considering the movies that both users have rated. However, we also wanted to include in the metric the fact that the similarity between two users based on 10 movies is more accurate than the one based on a single movie. Since the scalar similarity value lies between -1 and 1, we added a multiplier to the final score:

$$similarity = similarity * min(moviesInCommon/100, 1)$$

This way, similarity between users is scaled linearly by the number of movies in common. Intuitively, this puts similarity based on less movies closer to zero, therefore showing less certainty in the decision.
For the user recommendations, we first calculate the similarity of all users with respect to the target user and take 10 closest neighbors. Number 10 was chosen empirically based on validation. Then we compute the average ratings for target and all neighbor users. Finally, we calculate the predicted ratings for all movies by the target user using the formula:

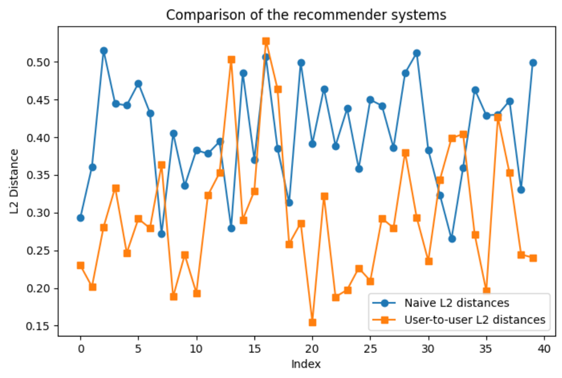
$$\text{interest}(a, s) = \overline{r_a} + \sum_b w(a, b)(r_{b,s} - \overline{r_b}),$$

where w is the similarity.

3 Validation based on genres

To validate the two recommender systems, we evaluate them as described in the question: Each system is assessed based on how closely the genre frequencies in its top-k recommended movies (with k=10) resemble the genre frequencies of the target user's movies in the validation set. The corresponding code can be found in the file *validation.py*.

The methodology compares the performance of the naive and user-based recommender systems. First, a genre matrix is constructed to map movies to their associated genres, and the user ratings are split into training and validation sets, reserving five movies per user for validation. The user-based recommender generates recommendations by constructing a user-movie utility matrix from the training data, while the naive recommender provides a popularity-based recommendation. The genre distributions of the top-k recommendations from both systems are calculated using the genre matrix and are compared to the validation set's genre distribution using the L2-norm (Euclidean distance) as a measure of similarity. This process is repeated for multiple randomly selected users, with results visualized in a plot that shows the L2 distances for both systems. Additionally, the average L2 distances are computed to summarize overall performance. An example plot illustrating the L2 distances for 50 users can be seen below:



As observed, the user-to-user recommender system consistently achieves smaller L2 distances compared to the naive system. On average, the naive recommender system has an L2 distance of 0.499, while the user-to-user recommender system achieves a lower average of 0.240. This difference demonstrates that the user-to-user system in most cases aligns more closely with the genre preferences in the validation set, making it the better-suited system overall.

4 Improved recommender engine

The information from the *tags.csv* file can be utilized to make the user-to-user recommender system more robust by providing access to a broader set of movie ratings. Before incorporating this data, it is important to note that the ratings should be rescaled to match the same boundaries. We propose two methods to enhance the user-to-user recommender system. First, the average movie ratings from external sources can be used to apply a baseline correction to user ratings. Specifically, subtracting the external average rating from the user's ratings allows us to focus more on user preferences relative to others. Given the large amount of data from external sources, these averages are likely to be statistically reliable. Second, the average movie ratings from external sources can serve as a baseline to address cold-start problems. Since external sources offer more comprehensive data compared to our *movies.csv* dataset, this baseline would be more precise and improve recommendations for new users.