Project Report: Movie Recommendations based on User Ratings and Aggregation Methods

Alessio Marinucci, 546778 → repo GitHub: https://github.com/alemari7/AdvancedTopics

Introduction: In this report, we discuss the development and implementation of a movie recommendation system based on the MovieLens 100K dataset. The goal is to provide personalized movie recommendations for a group of three randomly selected users using collaborative filtering techniques. We explore three aggregation methods, namely the average method, the least misery method and the disagreement-aware method, to aggregate individual user recommendations into group recommendations.

Code Description: The code is written in Python and utilizes the following libraries:

- **pandas**: for manipulation and analysis of tabular data.
- **numpy**: for mathematical operations and numerical computations.

The code is divided into several parts:

1. **Data Loading:** Movie rating data is loaded from a CSV file using the pandas library. Timestamps are removed as as they are not required for this project.
2. **Recommendations Calculation:** The `calculate_recommendations` function computes and returns personalized movie recommendations for a specific user within a group of users, based on their movie ratings and the similarity between their preferences. This implementation is chosen because it utilizes Pearson correlation, which is a common measure of user similarity based on the correlation of their ratings. Calculating similarity with all other users in the group enables obtaining personalized recommendations for the target user based on the preferences of other similar users. In this way we can see if the aggregation methods works well. It is correlated with the `predicted_rating` function described below.
3. **Rating Prediction:** The `predicted_rating` function predicts the rating of a movie for a given user using ratings from similar users and Pearson correlation. This implementation is chosen because it leverages the similarity between users to provide personalized rating predictions for movies, taking into account the deviations of similar users' ratings from their mean ratings. The weighted sum allows for a more refined prediction, giving higher weight to similar users with more similar preferences.
4. **Aggregation Methods:** Two aggregation methods are implemented to obtain recommendations for a group of users:
    - ***Average Method*:** This method calculates the average of predicted ratings for all movies for each user in the group and returns the top 10 movies with the highest average. I chose to implement this method because calculating the average of predicted scores provides a balanced view of group preferences. In other words, this approach takes into account the general consensus within the group by assigning equal weight to each user's opinion. Since it considers everyone's opinion, the aggregated recommendations reflect a more comprehensive view of the group's preferences. If a user gives an extremely high or low rating to a movie, it can significantly influence the average, leading to recommendations that may not necessarily reflect the preferences of the majority.
    - ***Least Misery Method*:** This method selects, for each movie in the group, the minimum rating among all predicted ratings by users and returns the top 10 movies with the highest minimum rating.

      This method was chosen to provide recommendations that minimize discomfort for group users. If a movie receives a low rating from at least one user, it's likely not appreciated by the group overall. Therefore, selecting the lowest predicted score for each movie reduces the risk of recommending movies that might not be liked by anyone. Reduces the risk of recommending movies that might receive low scores from at least one user, thereby ensuring a higher overall satisfaction level in the group. This method might not give enough weight to the preferences of some users, favoring movies that are mediocre for everyone rather than those that are loved by some members of the group. This could reduce the variety of recommendations.

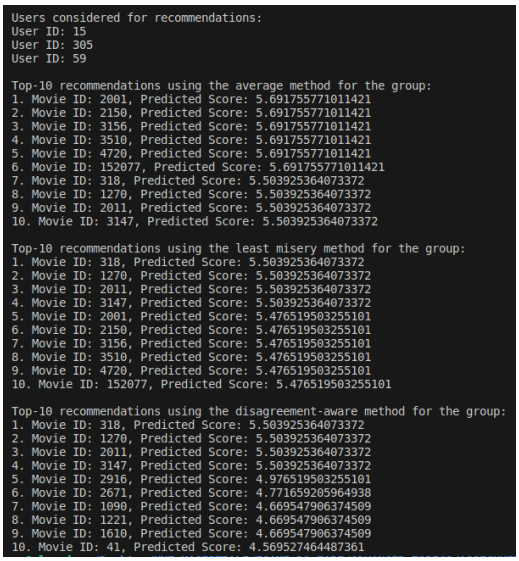
5. **Generating Recommendations for the Group:** Recommendations are calculated for each user in the group and then aggregated using the two aforementioned aggregation methods.
6. **Calculating disagreements among users:** To enhance the recommendation system's personalization, we introduced a method to measure the discrepancies in movie preferences between users within a group. These discrepancies, or disagreements, are quantified based on the differences in their ratings for common movies. I chose this implementation because it efficiently calculates the standard deviation of predicted scores for each movie, focusing only on movies with multiple predictions. This allows us to identify movies where there is a significant discrepancy in predicted ratings among users, providing valuable recommendations. Additionally, using the standard deviation as a measure of disagreement is a common approach in statistical analysis to quantify the spread or variability of a dataset, making it a suitable choice for this task. Here's a detailed overview of the process:
7. **Method that takes disagreements into account when computing suggestions for the group:** The function `aggregate_recommendations_disagreement` is designed to aggregate individual recommendations from multiple users while considering the level of disagreement among the predicted scores for each movie. I chose this implementation because it incorporates the concept of disagreement among predicted scores while aggregating recommendations. By weighting the average predicted scores based on the level of disagreement, it aims to provide more reliable and robust recommendations, giving higher priority to movies with more consistent predictions among users. The approach also ensures that movies with high disagreement receive lower weights, effectively reducing their impact on the final recommendations. This helps mitigate the influence of outliers or unreliable predictions, leading to more accurate and balanced recommendations for the group.

- The function `calculate_disagreement` takes as input a set of movie recommendations, where each recommendation is a list of (movie_id, predicted_score) pairs for a particular user. The function computes the standard deviation of the predicted rating scores for each movie across users who received recommendations for that movie. This helps evaluate how much the predicted scores for a particular movie vary among users.
- For each recommendation, the function extracts the movie_id and predicted_score, and adds them to the `movie_scores` dictionary. If a movie appears multiple times in the recommendations, its predicted scores are stored in a list associated with its movie_id.
- For each movie that appears in at least two recommendations, the function computes the standard deviation of users' predicted scores. This is done using the `np.std(scores)` function from the NumPy library.
- The function returns a `disagreement` dictionary, where the keys are the movie_ids of movies with at least two different predicted scores, and the values are the associated standard deviations.

Results and Discussion: After executing the code, the following results were obtained:

- The top 10 movie recommendations for each user in the group were identified using the Pearson Correlation method.
- Aggregate recommendations for the group were calculated using the two aggregation methods: Average and Least Misery.
- We use a third method that calculates the results using the disagreements between users.
- The results obtained for the three aggregation methods were printed to the screen.

An interesting result is that some of the movie IDs printed are the same, but with different score values, as it depends on the weight that the two methods give to the recommendations. The average method gives equal importance to all users, while the least misery method prioritizes the least preferred recommendation for the group.



Conclusions:

In conclusion, we have examined various techniques for generating recommendations for a group of users using the MovieLens 100K dataset. We compared two main approaches for aggregating recommendations: the average-based approach and the least misery-based approach. Both approaches have shown advantages and disadvantages in different situations.

The average-based approach proved to be effective in producing recommendations that reflect the group's average preferences, providing a good diversity of movies. However, this method can be influenced by extreme ratings and may not take into account the degree of agreement or disagreement among users.

On the other hand, the least misery-based approach has proven useful in mitigating the issue of extreme ratings, producing recommendations that maximize the group's satisfaction level. This method identifies movies that are at least acceptable for all users, minimizing the risk of dissatisfaction. However, it may lose sight of individual preferences, favoring movies considered acceptable but not necessarily preferred.

Additionally, we introduced an additional approach based on the concept of disagreement among users. This method considers the dispersion of ratings among users and weights recommendations based on the degree of agreement. Users with more cohesive ratings contribute more to the generation of recommendations. This approach aims to provide recommendations that maximize consensus within the group, taking into account differences of opinions among users.

In the future, we may explore further aggregation methods that consider not only the degree of agreement/disagreement among users but also their individual preferences and the diversity of recommendations. The goal will be to provide personalized and satisfying recommendations for each user within the context of a group.