Project Report: Rating Prediction Using User-Based Collaborative Filtering (UBCF) with Decayed Similarity

1. Problem Statement: Unveiling User Preferences in the Digital Realm

Recommender systems have become an indispensable part of our digital lives, subtly guiding our choices for movies, music, products, and more. At their core, these systems aim to predict user preferences for items. This project focuses on the classic problem of movie recommendation. Our goal is to predict the rating a user might assign to a specific movie, even if they haven't rated it yet. Leveraging a dataset of historical user-movie ratings, we tackle the task of estimating these missing ratings using User-Based Collaborative Filtering (UBCF). The project's success is evaluated using the Root Mean Squared Error (RMSE) metric, ensuring a balance between prediction accuracy and computational efficiency. Ultimately, this work contributes to building more personalized and engaging user experiences within recommendation platforms.

2. Algorithms and Methodology: Mining for Similar Tastes

2.1 User-Based Collaborative Filtering (UBCF): Finding Your Taste Twins

UBCF operates on the intuitive principle that users who have exhibited similar rating patterns in the past are likely to have similar preferences in the future. To predict a user's rating for a movie, UBCF analyzes the ratings provided by users deemed "similar." The process unfolds in the following steps:

Calculate Average User Affinity: We begin by computing the average rating for each user. This provides a baseline understanding of individual rating tendencies, which is crucial for the subsequent similarity calculations.

Identify Taste Neighbors: Mean-Centered Cosine Similarity: The heart of UBCF lies in identifying users with similar tastes. We employ the mean-centered cosine similarity metric to quantify the similarity between users. By subtracting each user's average rating from their individual ratings, we effectively normalize for differing rating scales (some users might consistently rate higher or lower than others). This allows us to focus on the patterns of preferences rather than just the absolute rating values.

Predicting with Wisdom of the Crowd: Weighted Average with Decayed Similarity: Once we've identified similar users, we predict the target user's rating as a weighted average of the ratings given by these similar users. The weight assigned to each neighbor's rating is determined by their similarity score. Crucially, we introduce a decay factor to the similarity scores. This decay, implemented using the exponential function, emphasizes the

contribution of highly similar users while diminishing the influence of those with weaker similarity. This helps to refine the prediction and reduce the impact of less reliable neighbors.

2.2 Mean-Centered Cosine Similarity: Measuring Shared Preferences

This metric quantifies the similarity between two users by comparing their mean-centered ratings. It measures the cosine of the angle between the two user's rating vectors in a shared item space.

$$\text{Similarity}(A, B) = \frac{\sum_{i \in I} (A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i \in I} (A_i - \bar{A})^2} \cdot \sqrt{\sum_{i \in I} (B_i - \bar{B})^2}}$$

Where:

A_i and B_i are the ratings for item i by users A and B, respectively.

$\bar{A}$ and $\bar{B}$ are the average ratings of users A and B, respectively.

I is the set of commonly rated items by both users A and B.

2.3 Prediction Formula: Aggregating Neighborly Insights

The predicted rating for a user u and movie m is calculated using the following formula:

$$\hat{r}_{u,m} = \bar{r}_u + \frac{\sum_{v \in S} \text{sim\_decayed}(u, v) \cdot (r_{v,m} - \bar{r}_v)}{\sum_{v \in S} |\text{sim\_decayed}(u, v)|}$$

Where:

$\hat{r}_{u,m}$: Predicted rating for user u on movie m.

$\bar{r}_u$: User u's average rating.

S: The set of the k most similar users to user u.

sim_decayed(u,v): The decayed similarity between users u and v, calculated as exp(meanCenteredCosineSimilarity(u, v)). This exponential decay emphasizes stronger similarities.

$r_{v,m}$: The actual rating given by user v to movie m.

$\bar{r}_v$: User v's average rating.

3. Implementation: Bringing the Algorithm to Life

3.1 Data Structures: Organizing the Information Flow

unordered_map<int, unordered_map<int, float>> rating_map: This nested map efficiently stores user ratings for movies. The outer key represents the user ID, and the inner map's key-value pairs represent the movie ID and the corresponding rating. This structure allows for quick access to a specific user's ratings for a particular movie.

unordered_map<int, float> user_average_ratings: This map stores the pre-computed average rating for each user, with the user ID as the key. Calculating these averages upfront avoids redundant computations during the prediction phase.

Vectors: Vectors are used to temporarily store similarity scores between users and to hold the test data pairs. Their dynamic nature makes them suitable for collecting and sorting similarity values.

3.2 Key Functions: The Building Blocks of Prediction

calculateUserAverageRatings(): This function iterates through the rating_map to compute the average rating for each user and stores these values in the user_average_ratings map.

meanCenteredCosineSimilarity(ratings1, ratings2, avgRating1, avgRating2): This function implements the mean-centered cosine similarity calculation as described in Section 2.2. It takes the rating maps and average ratings of two users as input and returns their similarity score.

predictRating(userId, movieId, k): This is the core prediction function. It identifies the k most similar users to the target user using the meanCenteredCosineSimilarity function. It then calculates the predicted rating for the given movie using the weighted average formula, incorporating the decayed similarity of the neighbors. The parameter k controls the number of neighbors considered for the prediction.

3.3 Workflow: From Data to Predictions

Data Ingestion and Preparation: The process begins by parsing the input dataset, separating it into training and test sets. The training data is used to build the rating_map and user_average_ratings, while the test data contains user-movie pairs for which we need to predict ratings.

Calculating User Baselines: The calculateUserAverageRatings function is called to compute the average rating for every user in the training dataset.

Predicting Ratings for Test Cases: For each user-movie pair in the test dataset:

The predictRating function is invoked.

This function calculates the similarity between the target user and all other users who have rated the target movie.

The similarities are sorted in descending order.

The top k most similar users are selected.

The predicted rating is calculated as a weighted average of their ratings, using the decayed similarity as weights.

Generating Output: The predicted ratings for all test pairs are then outputted.

5. Conclusion: A Step Towards Personalized Recommendations

This project successfully implemented a User-Based Collaborative Filtering (UBCF) algorithm with the incorporation of a decayed similarity measure to predict movie ratings based on historical user data. By leveraging mean-centered cosine similarity to identify like-minded users and applying a weighted average of their ratings, with an exponential decay factor emphasizing stronger similarities, the system achieved accurate predictions with an RMSE below 1.0. This demonstrates the effectiveness of the approach and its potential for powering personalized recommendation systems. The inclusion of decayed similarity refined the prediction process, giving more weight to truly similar users.

6. Team Members and Responsible Parts

Halil Özer – Calculating Average Rating for All Users Function Part

Ömer Faruk Tokgöz – Mean Centered Cosine Similarity Function Part

Mehmet Efe Çakır – Predict Rating Function Part (including the decayed similarity implementation)

Adil Bora Binici – Main (Reading the data and Output) Part