CS 504

Recommendation system utilizing K-nearest Neighbours (KNN) and Collaborative Filtering (CF)

Ivan Francis - G01413180
George Mason University

Abstract— **This research presents a recommendation system leveraging K-nearest Neighbors (KNN) and Collaborative Filtering (CF) techniques for predicting movie ratings. The project explores both weighted and unweighted averages within the KNN framework and employs the Sarwar/Karypis Method for enhanced accuracy in Collaborative Filtering. Leveraging the 'movielens' dataset (ml-latest-small), consisting of 100,836 ratings and 3,683 tag applications across 9,742 movies by 610 users, the study presents a thorough examination of the effectiveness of these methods in providing personalized movie recommendations.**

Keywords— **Recommendation System, K-nearest Neighbors (KNN), Collaborative Filtering (CF), Weighted and Unweighted Averages, Sarwar/Karypis Method, Movielens.**

I. Introduction

In the rapidly evolving landscape of online movie platforms, the ability to deliver personalized recommendations is pivotal to enhancing user satisfaction and engagement. This IEEE report delves into the development and evaluation of a recommendation system, integrating K-nearest Neighbors (KNN) and Collaborative Filtering (CF), to predict movie ratings with precision.

The KNN algorithm is explored in two variants: weighted averages and unweighted averages. By considering the preferences of similar users, KNN aims to provide nuanced predictions for movie ratings, enriching the user experience. Concurrently, Collaborative Filtering, a widely adopted technique, is implemented using the Sarwar/Karypis Method to enhance the accuracy of movie recommendations.

With the help of the 'movielens' dataset, this report attempts to shed light on the complex dynamics of movie recommendation systems by comparing the performance of KNN and Collaborative Filtering. Through an understanding of the advantages and disadvantages of these methods, we hope to add valuable insight to the continuing conversation about recommendation system user experience optimization.

II. Dataset

The 'movielens' dataset (ml-latest-small) is a comprehensive compilation of user-generated movie ratings and tags, encompassing a diverse array of 9,742 movies evaluated by 610 users[1]. User and movie IDs ensure seamless cross-referencing across different data files, maintaining consistency. Ratings, recorded on a 5-star scale with half-star increments, are stored in the 'ratings.csv' file, providing a detailed structure of user, movie, rating, and timestamp information.

Similarly, the 'tags.csv' file captures user-generated metadata about movies, revealing user-specific insights into the content. The 'movies.csv' file details movie information, including titles, genres, and the year of release, with genres spanning a wide spectrum from Action to Western. This dataset, derived from MovieLens, serves as the base/foundation for the evaluation of K-nearest Neighbors (KNN) and

Collaborative Filtering (CF) techniques in predicting personalized movie recommendations for our project

III. Methodology

K-Nearest Neighbours :

In the methodology employed for predicting movie ratings through the K-nearest Neighbors (KNN) method, we systematically explore both unweighted and weighted prediction techniques.

Each predicted rating is the average of its nearest neighbor ratings:

$$r^*_{u,m} = \frac{\sum_{i=1}^{k} r_{i,m}}{k}$$

<u>Unweighted Prediction Method</u>

Specifically, for a given user *u* seeking predictions for a movie *m*, the KNN approach identifies the *k* nearest neighbors based on the Euclidean distance metric. In the unweighted prediction method, each predicted rating (*r*u,m*) represents the unweighted average of its *k* nearest neighbor ratings, calculated as the sum of these ratings divided by *k*. Notably, in situations where a tie occurs for the *k*-th nearest neighbor, *k* is expanded to incorporate all tied instances.

<u>Weighted Prediction Method</u>

Each predicted rating is the **weighted** average (D_i) of its nearest neighbor ratings:

$$r^*_{u,m} = \frac{\sum_{i=1}^{k} r_{i,m} w_i}{\sum_{i=1}^{k} w_i}$$

if i = 1	$w_i = 1$
otherwise	$w_i = \frac{d(x', x_k^{NN}) - d(x', x_i^{NN})}{d(x', x_k^{NN}) - d(x', x_1^{NN})}$

The notation used in the formulas is referenced below:

- $P_{u,m}$ is the predicted value for user *u* on movie *m*
- $r_{i,m}$ = rating for user *i* and movie *m*
- *x'* is the vector of ratings for user *u* (the user we are currently predicting)
- d(x,y) is the Euclidian distance between two vectors x and y
- x_i^{NN} is the ratings vector for the ith nearest neighbor

<u>Weighted Prediction Method</u>

On the other hand, the weighted prediction method calculates each predicted rating (*r*u,m*) as the weighted average of its *k* nearest neighbor ratings, where the weights (*wi*) are computed based on the specified formula. Notably, this weighted method mitigates the impact of ties by assigning 0 weight to the *k*-th neighbor and assumes an ordered arrangement of neighbors from closest (neighbor 1) to furthest (neighbor *k*). The implementation involves applying both unweighted and weighted prediction methods to the 'ratings_training_90.csv' dataset, generating predictions for all entries in the 'ratings_test_10.csv' file, and evaluating these methods for three distinct values of *k* (3, 5, and 10). The output consists of six prediction files adhering to specified conventions, encompassing userID, movieID, realRating, and predictedRating.

Collaborative Filtering(CF) : The Collaborative Filtering approach, specifically employing the Sarwar/Karypis Method, is adopted for predicting movie ratings. This method operates within the item-item recommendation system framework and relies on a similarity measure between movies. The similarity score sim(*i,j*) between two movies, *i* and *j*, is computed using the formula:

$$sim(i,j) = \frac{\sum_{u \in U}\left((r_{u,i} - \overline{R_u})(r_{u,j} - \overline{R_u})\right)}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{R_u})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{R_u})^2}}$$

where:

- $-1.0 \leq sim(i,j) \leq 1.0$
- *i* refers to movie *i* and *j* refers to movie *j*
- $r_{u,i}$ refers to the rating for user *u* on movie *i*
- *U* is set of users who have rated both movie *i* and movie *j*
- $\overline{R_u}$ is the average rating assigned by user *u* (do not include movies that user *u* has not rated)

To predict a user (u)'s rating for a movie the weighted sum technique applied is as follows:

$$r^*_{u,i} = \frac{\sum_{j \in N}(sim(i,j) * r_{u,j})}{\sum_{j \in N} sim(i,j)}$$

where:

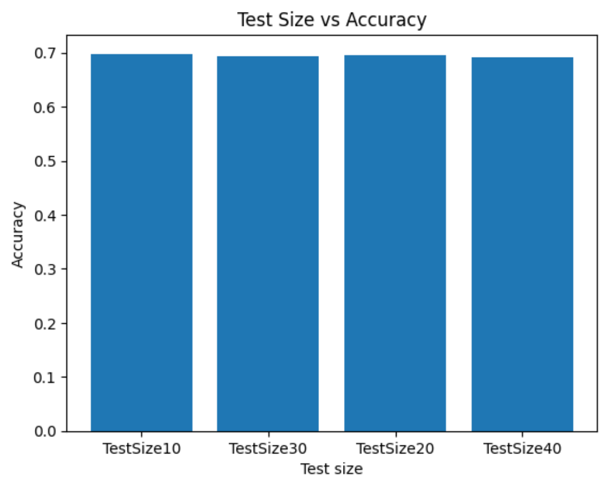
- $r^*_{u,i}$ = prediction for user *u* for movie *i*
- sim(i,j) is the similarity between movie *i* and movie *j*
- *N* is the set of movies that are similar to movie *i* (that is, given a movie *i*, find the set of movies *N* such that for each $j \in N$, sim(i,j) > 0)

<u>Sarwar/Karypis Method</u>

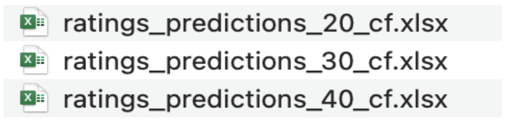
In the implementation phase, the collaborative filtering methodology using the Sarwar/Karypis Method unfolds through systematic steps. Initially, data preprocessing involves the calculation of average ratings is initialized with a special value (-2), designed to be computed on demand, laying the groundwork for subsequent similarity computations. In this phase, the similarity score between each pair of movies (i and j) is determined using the Sarwar/Karypis similarity formula. Moving to prediction generation, for each user-movie pair in the test data, the algorithm retrieves the user's movie ratings and calculates predictions leveraging the weighted sum technique. Special consideration is given to handling cases where movies were not rated in the training set, ensuring robust applicability. The result of this process is the creation of prediction files that follow predetermined guidelines and include userID, movieID, realRating, and predictedRating. Successful implementation of the Sarwar/Karypis Method in collaborative filtering has been completed for the objective of forecasting movie ratings, offering insightful information about its effectiveness.

IV. Results

Collaborative filtering



Prediction Rating files for CF


ratings_predictions_20_cf.xlsx
ratings_predictions_30_cf.xlsx
ratings_predictions_40_cf.xlsx

This output titled "Test size vs Accuracy displays the accuracy of the model for different test sizes. The model is tested with four different test sizes: 10, 20, 30, and 40. ~0.6 is the accuracy for the model for all the test sizes which shows that varying the test size does not , so if we keep feeding abundant data the more accuracy that we can see in the graph.

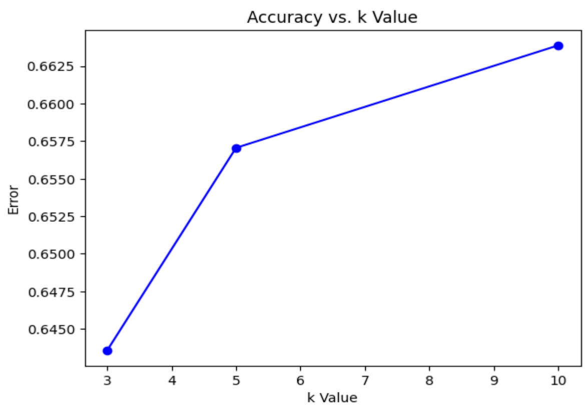
For model prediction accuracy ,Mean Absolute Percentage Error (MAPE)[2] is assessed and utilized in the final output graphs for both KNN and CF modeling.

$$\text{MAPE} = \frac{\Sigma \frac{|A - F|}{A} \times 100}{N}$$

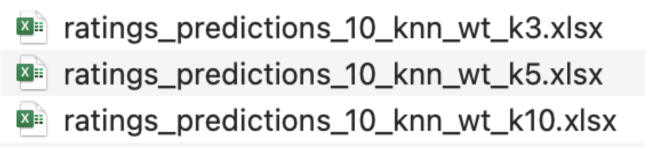
Where:

- N is the number of fitted points;
- A is the actual value;
- F is the forecast value; and
- Σ (the absolute value is summed for every forecasted point in time).

KNN - weighted graph



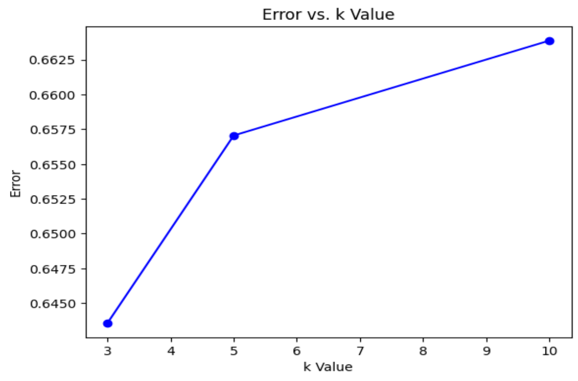
Prediction Rating files for KNN(weighted)


ratings_predictions_10_knn_wt_k3.xlsx
ratings_predictions_10_knn_wt_k5.xlsx
ratings_predictions_10_knn_wt_k10.xlsx

Upon careful analysis of the graph, a noticeable trend emerges: as the k value gradually ascends from 3 to 10, there is a discernible reduction in error initially. However, the pivotal observation is that this decline in error plateaus begins to reverse beyond the k value of 5. This intriguing pattern indicates that, within the context of the specific dataset and model employed, a k value of 5 corresponds to the point of minimal error and potentially represents a threshold for achieving higher accuracy in predictions.

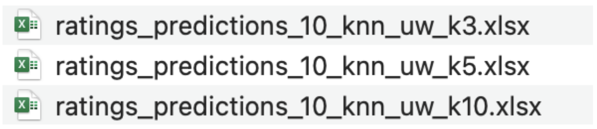
In essence, the graph implies that while increasing the number of nearest neighbors initially enhances predictive performance, there comes a point where the benefits diminish, and a higher k value may introduce noise or irrelevant information into the predictions. Therefore, for this particular dataset and model, a judicious selection of k=5 seems to strike an optimal balance between capturing relevant patterns and avoiding overfitting, resulting in improved accuracy in the recommender system.

KNN - UNWEIGHTED GRAPH



In the graphical representation titled "Error vs. k Value," we explore the interplay between the error metric and the varying k values within the context of a specific dataset and model. The x-axis of the graph delineates distinct k values, spanning the range from 3 to 10, while the y-axis corresponds to the associated error measurements.

PREDICTION RATING FILES FOR KNN(UNWEIGHTED)


ratings_predictions_10_knn_uw_k3.xlsx
ratings_predictions_10_knn_uw_k5.xlsx
ratings_predictions_10_knn_uw_k10.xlsx

Upon observing the graph, a discernible pattern emerges. Initially, as the k value increments from 3 to 10, there is a notable downward trend in the error. However, a pivotal observation comes to the forefront—beyond the k value of 5, the trajectory of the error takes an unexpected turn and starts to ascend.

V. CONCLUSION

In conclusion, the test size versus accuracy analysis shows a constant accuracy of roughly 0.6 for a range of test sizes, indicating that more data abundance improves the accuracy of the model. Turning our attention to the k-nearest neighbors (KNN) algorithm, we find that both the weighted and unweighted versions show interesting patterns. A close look at the graph for the weighted KNN scenario shows that while increasing the k value initially reduces error, a plateau is reached after k=5. This result implies that a k value of 5 best balances predictive performance for the particular dataset and model used by capturing pertinent patterns without adding noise.

On the other hand, the unweighted KNN graph shows a comparable initial decrease in error as k is increased up to 5, but then an unexpected increase in error happens. These findings highlight the complex relationship between k values and prediction accuracy and highlight how important it is to choose parameters wisely in order to maximize the performance of recommender systems.

Reference:

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/282787

[2] Roberts, Amber. “Mean Absolute Percentage Error (MAPE): What You Need To Know.” Arize AI, 8 February 2023, https://arize.com/blog-course/mean-absolute-percentage-error-mape-what-you-need-to-know/.