Recommender Systems

Brendan George Lauterborn
Department of Computer Science
Towson University
Towson, United States
blaute3@students.towson.edu

***Abstract*—Recommender Systems are essential in driving customer sales by enhancing user experiences and providing personalized recommendations. This paper aims to build multiple recommender systems using the MovieLens latest-small dataset. A user-based Collaborative Filtering system will first be built as the baseline. Next, we use a model-based Collaborative Filtering known as Alternating Least Squares Matrix Factorization to attempt to improve the system. To evaluate the systems, we will use precision@k, recall@k, MAP, and NDCG.**

I. Introduction

Digital content continues to grow in the cloud-based era. This leads to challenges in providing users with a personalized experience online. Users are exposed to so much information at one time, making it difficult to identify what information is interesting to them. By providing the user with information that they care about, we can improve user satisfaction, increase engagement, and boost sales.

Recommender Systems identifies this issue directly by providing the user with content that interests them. These systems are used and developed by many large platforms such as Netflix, Spotify, and Amazon. The three most used systems include Collaborative Filtering, Content-Based Filtering, and Knowledge-Based Filtering. This study focuses on the first approach. The goal of this study is to build and compare the results of each system and understand the benefits and limitations of each one.

II. Literature Review

User-user Collaborative Filtering is one of the most commonly used recommender systems. In fact, user-user CF was the first of the automated CF methods [1]. The algorithm is straightforward. Users that have similar past ratings to that of the target user are used to predict the rating of the target user. There are several functions to find the similarity among users for CF filtering. These consist of Pearson correlation, Constrained Pearson correlation, Spearman rank correlation, and Cosine similarity. One issue to consider in user-based CF is deciding how many neighbors to select when determining predictions. Herlocker et al. found that selecting 20-50 neighbors seems to be reasonable, but it may vary depending on the user and dataset [1].

The Alternating Least Squares CF method is commonly used in many large-scale data processing environments [2]. Latent factor models are very important due to their ability to determine hidden relationships between the user and item. These models have also demonstrated more accuracy than other methods that rely on neighbors to make predictions. [2]. In the context of movies, the latent feature may represent the genre of the movie, while the user's latent vector may represent their preference for that specific genre.

However, CF systems suffer from the cold-start problem. When there is a new user with no previous data, the system struggles to recommend movies to this target user. Overall, user-based CF and model-based CF scale well and are extremely important in recommending items to the target user.

III. Problem Definition

Online streaming platforms only grow larger with the overwhelming amount of content that is being released, making it difficult for users to navigate and discover movies that are of interest to them. There are simply too many movies to randomly navigate.The user desires personalization from the platform that they pay for monthly. Implementing a powerful recommendation system can improve the user experience.

This study focuses on solving the issue of recommending movies of interest to a specific user. We aim to build multiple systems that recommend movies that the user would enjoy to improve their experience on the platform.

IV. Approach Description

To solve this problem, we will be using the MovieLens latest-small dataset. We start out by building a user-based Collaborative Filtering system as the baseline. Next, we use Alternating Least Squares Matrix Factorization as the second system to attempt to improve the movie recommendations.

A. User-based Collaborative Filtering

The user-based Collaborative Filtering system serves as the baseline model we will use for this study. Given an active user, there are many movies that this user has not rated. The goal is to estimate the active users rating for this movie. To do this, first find other users who liked the same items as this active user has in the past. This is achieved by measuring user similarity using cosine similarity in our case. The average ratings of similar users for that movie are used to predict the active user's rating for that unseen movie. This is done for all of the movies the user has not seen and the best rated movie will be recommended to that user. The cosine similarity function used in this study is defined as:

$$\text{sim}(a,b) = \frac{\sum_{p \in P} r_{a,p} \cdot r_{b,p}}{\sqrt{\sum_{p \in P} r_{a,p}^2} \cdot \sqrt{\sum_{p \in P} r_{b,p}^2}}$$

where a, b are users, r_{ap} is the rating of user a for item p, and p is the set of all items rated by user a, b. The prediction function used in this study is defined as:

$$\text{pred}(a,p) = \bar{r}_a + \frac{\sum_{b \in N} \text{sim}(a,b) \cdot (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} \text{sim}(a,b)}$$

where $\bar{r}_a$ and $\bar{r}_b$ are the mean ratings of users a and b, $r_{b,p}$ is the rating of user b for item p, and N denotes the set of top-k most similar neighbors to user a.

This method is known as memory-based since it relies directly on neighbors with similar interests to generate recommendations. In this approach, we find similarities between users.

B. Alternating Least Squares Matrix Factorization

The second approach, Alternating Least Squares Matrix Factorization, is a model-based approach rather than memory-based. A mathematical model is built in this approach based on the user-item ratings.

Matrix Factorization breaks down the user–item rating matrix $R \in \mathbb{R}^{m \times n}$ into two lower-dimensional matrices:

$$R \approx UV^T$$

where $U \in \mathbb{R}^{m \times k}$ represents the hidden feature matrix for users and $V \in \mathbb{R}^{n \times k}$ represents the hidden feature matrix for items. The dot product:

$$\hat{R}_{ij} = U_i \cdot V_j^T$$

Where U_i represents the hidden feature matrix for each user i and V_j represents the hidden feature matrix for each item j. By taking the dot product, we are essentially comparing the user's tastes to a movie's features. The goal is to minimize the error between the actual and predicted ratings:

$$\min_{U,V} \sum_{(i,j) \in \kappa} (R_{ij} - U_i V_j^T)^2 + \lambda(||U_i||^2 + ||V_j||^2)$$

where κ represents the set of known ratings and λ is the regularization term that prevents overfitting the system. The Alternating Least Squares algorithm solves this optimization by first initializing the the matrices with random numbers. It then fixes one matrix and solves for the other. Then alternating between the two for each iteration. This is done many times until the error term is minimized.

V. Experimentation

A. Dataset Description

The MovieLens small dataset contains 100836 ratings across about 600 users and 9000 movies. For the user-based CF system we built, we used two files from the dataset. We used ratings.csv and movies.csv. The first dataset contains four attributes, userId, movieId, rating, and timestamp. UserId and movieId is the unique number given to a user and movie respectively. Rating corresponds to the user's rating of each movie, on a scale from 1-5. The second dataset, movies.csv, contains three attributes, movieId, title, and genre. Here, movieId is the same as before. Title represents the name of the movie and genre represents the type of movie that it is.

B. User-based CF system

We began building this model by first splitting the ratings.csv into a training and testing set 80/20. From there, we created the user-item matrix from the training set. We applied mean-centering to all user ratings to remove bias from each user's ratings. After that, we filled in all missing ratings with a value of 0 only for the cosine similarity to determine similarities between the users. We chose to use $k = 50$ neighbors in our approach. After we applied cosine similarity, we implemented the rating predictor that estimates a user's movie rating for an unseen movie. This is done using the weighted average of the neighbors' deviations from their means. Then adding the target user's mean back to the value. This is the prediction equation. Once we completed that, we built the function that ranks unseen movies by the user and returns the top k movies using the k most similar users to the target user. The system is then tested for performance using precision@k, recall@k, MAP, and NDCG.

C. ALS Matrix Factorization

Building this system began by first splitting the ratings.csv into a training and testing set 80/20. We build the user-item matrix for a training set and a testing set. The implicit library is designed for implicit feedback data, so we needed to convert the user ratings to confidence weights by multiplying the matrix by α = 15. After preparing the confidence matrix, we trained the ALS model using 20 latent factors, 20 iterations, and a regularization factor of .1 to help prevent overfitting. The ALS algorithm factorizes the user–item rating matrix $R \in \mathbb{R}^{m \times n}$ into two lower-dimensional matrices U and V. Where U represents the user matrix and V represents the item matrix. These matrices represent the latent features of users and items learning by training. During this training, ALS alternates between optimizing U and V, while keeping the other matrix fixed, to minimize the error term, $\min_{U,V} \sum_{(i,j) \in \kappa} (R_{ij} - U_i V_j^T)^2 + \lambda(||U_i||^2 + ||V_j||^2)$. Once the model is trained, we obtained the prediction scores by taking the dot product $\hat{R}_{ij} = U_i \cdot V_j^T$. This estimates the unseen rating for user i for movie j. The resulting matrix, $\hat{R} \in \mathbb{R}^{m \times n}$, contains all predicted user-item ratings. Lastly, for each user we rank the ratings and recommend the top-k rated movies.

In the final section, we highlight the benefits and limitations of each system. We also discuss the performance across each method.

VI. Analysis

The User-Based Collaborative Filtering performance metrics are shown in Fig. 1. Precision@K represents the recommended

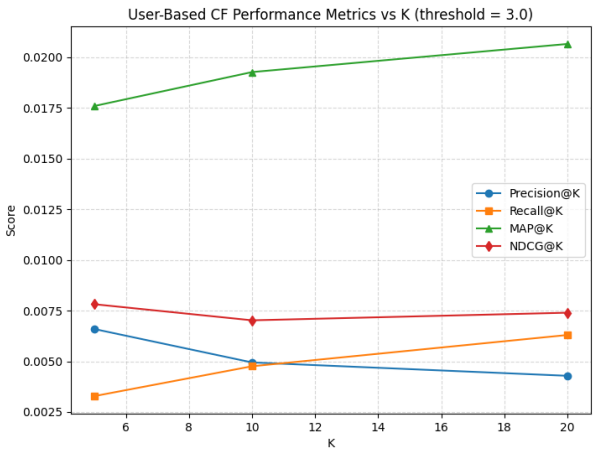

Fig. 1: User-based CF performance metrics vs K

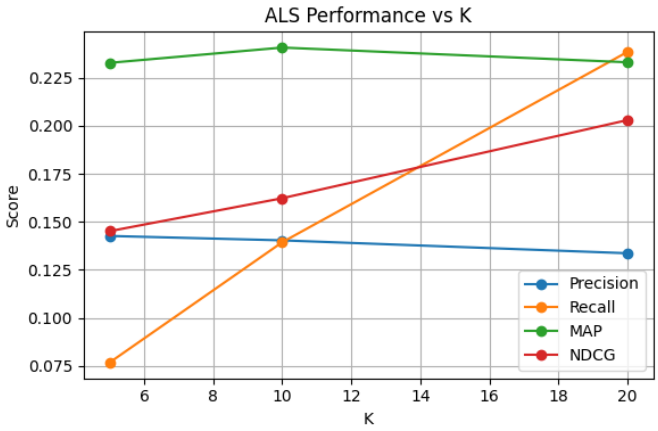

Fig. 2: ALS Matrix Factorization performance metrics vs K

movies that were actually relevant. Precision@K slightly decreased from 0.0066 to 0.0043 as K increased from 5 to 20. Meaning as we recommend more movies to user i, the precision decreases. Recommending more movies results in including some that start to be slightly more irrelevant.

Recall@K represents the fraction of all relevant movies in the test set that were actually recommended. As we increase K from 5 to 20, Recall@K increases from 0.0033 to 0.0063. Showing that the model covers more of user i's relevant movies as the number of recommended movies increases.

MAP represents how well we ranked the movies for user i. It measures the number of relevant movies and their placement. Our MAP value stays relatively consistent as K increases, only increasing from 0.0176 to 0.0207. Meaning that the model is consistent in ranking quality.

NDCG determines the ranking quality of our model. Our NDCG score stays consistent, ranging from 0.0078 to 0.0074 as K increases from 5 to 20. Meaning that our model consistently places relevant movies near the top as K increases.

Additionally, our RMSE is .876 over 19347 pairs. Meaning our predicted ratings are ± 0.876 from the actual rating.

The ALS Matrix Factorization performance metrics are shown in Fig. 2. Precision@k represents the recommended movies that were actually relevant. Precision@k slightly decreased from .142 to .133 as k increased from 5 to 20. Meaning as we recommend more movies to user i, the precision decreases. Recommending more movies results in including some that start to be slightly more irrelevant.

Recall@K represents the fraction of all relevant movies in the test set that were actually recommended. As we increase k from 5 to 20, Recall@k increases from .077 to .239. Showing that the model covers more of user i's relevant movies as the number of recommended movies increases.

MAP represents how well we ranked the movies for user i. It measures the number of relevant movies and their placement. Our MAP value stays relatively consistent as k increases, only increasing from .2328 to .2331. Meaning that the model is consistent in ranking quality.

NDCG determines the ranking quality of our model. Our NDCG score increases from .145 to .203 as k increases from 5 to 20. Meaning that our model places relevant movies near the top as k increases. User i would be able to find relevant movies from this list with no problem.

After comparing both models, it is clear that ALS Matrix Factorization outperforms our user-based CF model in every metric. Our ALS model produces higher precision@k, recall@k, MAP. and NDCG scores. This indicated that the model-based approach was generally more effective at recommending movies to the target user than the user-based approach. This could be due to matrix factorization capturing latent features between users and items. The MovieLens dataset is also sparse. Users only rate a small percentage of available movies, resulting in less overlapping rating between the users. Thus, making it harder for the model to capture user-user similarities.

VII. Conclusion

Our results show that the ALS Matrix Factorization approach outperforms our user-based CF approach in recommending movies to a target user. Due to the sparsity of the dataset, the ALS model was a better fit for our ALS system. Future models would include content-based filtering and deep learning approaches to better solve this problem.

References

[1] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative Filtering Recommender Systems," *Foundations and Trends in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011. doi: 10.1561/1100000009.

[2] M. Winlaw, M. B. Hynes, A. Caterini, and H. De Sterck, "Algorithmic Acceleration of Parallel ALS for Collaborative Filtering: Speeding up Distributed Big Data Recommendation in Spark," *arXiv preprint* arXiv:1508.03110, 2015. Available: https://arxiv.org/abs/1508.03110

[3] S. Learns Things, "A gentle introduction to Alternating Least Squares (ALS)," Apr. 5 2018. Available: https://sophwats.github.io/2018-04-05-gentle-als.html

[4] GeeksforGeeks, "What is Collaborative Filtering?," GeeksforGeeks, 2023. Available: https://www.geeksforgeeks.org/machine-learning/what-are-recommender-systems/#method-1-collaborative-filtering

[5] CodeSignal, “Implementing the Alternating Least Squares Algorithm,” CodeSignal Learn. Available: https://codesignal.com/learn/courses/diving-deep-into-collaborative-filtering-techniques-with-als/lessons/implementing-the-alternating-least-squares-algorithm
[6] “Alternating Least Squares (ALS) Recommender,” YouTube Video: G4MBc40rQ2k, posted by YouTube, [date unknown]. Available: https://www.youtube.com/watch?v=G4MBc40rQ2k
[7] “ALS Tutorial – Collaborative Filtering,” YouTube Video: 3ecNC-So0r4, posted by YouTube, [date unknown]. Available: https://www.youtube.com/watch?v=3ecNC-So0r4
[8] Evidently AI, “Precision & Recall at K,” EvidentlyAI, [date unknown]. Available: https://www.evidentlyai.com/ranking-metrics/precision-recall-at-k
[9] Evidently AI, “Mean Average Precision (MAP),” EvidentlyAI, [date unknown]. Available: https://www.evidentlyai.com/ranking-metrics/mean-average-precision-map