Performance Evaluation of Collaborative Filtering Algorithms on Video Streaming Platform

Supanat Jintawatsakoon and Seksan Sangsawad*

College of Digital Innovation Technology, Rangsit University, Pathum Thani, Thailand
*Corresponding author, E-mail: seksan.s@rsu.ac.th

Abstract

The objective of this research is to evaluate the performance of a recommendation system that employs the Collaborative Filtering method through a comparative analysis of two algorithms - Neighborhood Based Algorithm and Matrix Factorization Algorithm. The study seeks to compare the performance of rating prediction on datasets with varying sizes and levels of sparsity. The findings reveal that the accuracy of prediction was better on datasets that were smaller and less sparse. Moreover, the research also indicates that the SVD++ algorithm demonstrated the highest accuracy, despite having the longest processing time.

Keywords*: *Recommendation System, Collaborative Filtering

1. Introduction

A Recommendation System is a system designed to suggest products to customers, and it has become a critical component in the competitive landscape of e-commerce. For many companies, these systems have become a key strategy for improving their services and gaining an edge over their competitors. Notably, Netflix has reported that 80% of users choose to watch streaming movies on the platform based on recommendations generated by the system (McAlone, n. d.), and Amazon has disclosed that 35% of its purchases are influenced by its recommendation system (Mangalindan, n. d.).

With the vast number of products available and the intense competition in the e-commerce industry, product referrals can benefit both consumers and companies. From a consumer perspective, the recommendations can help them find products that meet their needs and preferences.

For businesses, recommending products to potential customers can increase the chances of generating revenue. Additionally, a good recommendation system that suggests relevant products can enhance user engagement with the store or service, leading to an improved shopping experience overall.

Currently, Collaborative Filtering-Based recommendation systems are widely accepted and utilized (Wei, He, Chen, Zhou, & Tang, 2017). Multiple algorithms have been designed with this concept in consideration. Nonetheless, creating product recommendations that are precise and customized to satisfy customer requirements continues to pose a challenge due to the fact that not every algorithm is appropriate for every situation or circumstance.

The subsequent sections of this paper are structured in the following manner: section 2 provides an overview of the background and related works, while section 3 details the materials and methods that were utilized. In section 4, the results are analyzed and discussed, and lastly, section 5 outlines the conclusion of the study.

2. Background and Related Work

Collaborative Filtering is a highly effective and practical technique in the field of recommendation system development, specifically within the domain of recommendation systems. The technique involves recommending items to users by identifying other users with similar characteristics, such as purchase or viewing history. For instance, if User 1 buys items A, B, C, D, E, and F, and User 2 buys items A, B, D, E, and F, Collaborative Filtering infers that user 2 is similar to User 1 and recommends Product C to them.

The effectiveness of Collaborative Filtering depends on the user's preference score for each item, which can be either Explicit Rating, derived from the user's own rating of each item as shown in Table 1, or

Implicit Rating (Wei et al., 2017), which is derived from the user's behavior, such as clicks, visits, or purchases of an item.

Collaborative Filtering predicts scores for items that have not yet been rated by the user so that high-scoring items can be recommended. Table 1 shows that users have rated some items, referred to as Known Rating, while others are yet to be rated, called Unknown Rating.

Moreover, Collaborative Filtering can be classified into two main categories, Neighborhood-Based and Matrix Factorization-Based. The former predicts ratings for an item based on ratings from similar users, while the latter utilizes a matrix of user-item interactions to predict ratings.

Table 1 The matrix shows the user ratings for each item.

	I1	**I2**	**I3**	**I4**
U1	5	1		
U2		2	3	
U3	4			4
U4	2		3	

2.1 Neighborhood-Based

This method utilizes the evaluations provided by users for the items as shown in Table 1 to calculate the similarity between each user or item. This enables the identification of similarities between users (referred to as user-to-user similarity) or between items (referred to as item-to-item similarity). Cosine similarity is commonly employed in Collaborative Filtering for this purpose. To determine the similarity between User u and User v, Equation (1) can be applied. The equation uses r_{ui} to indicate the score provided by User u for Item i, r_{vi} to represent the score given by User v for Item i, and I_{uv} to represent the set of all items rated by both User u and User v.

$$\text{cosine\_sim}\,(u,\ v) = \frac{\sum_{i \in I_{uv}} r_{u\,i} \cdot r_{v\,i}}{\sqrt{\sum_{i \in I_{uv}} r_{u\,i}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{v\,i}^2}} \tag{1}$$

Similarly, Equation (2) (Ricci, Rokach, & Shapira, 2015) can be used to calculate the similarity between Item i and Item j. The equation uses U_{ij} to represent the set of users who have evaluated both Item i and Item j. These methods are commonly used in Collaborative Filtering.

$$\text{cosine\_sim}\,(i,\ j) = \frac{\sum_{u \in U_{ij}} r_{u\,i} \cdot r_{u\,j}}{\sqrt{\sum_{u \in U_{ij}} r_{u\,i}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{u\,j}^2}} \tag{2}$$

In order to find similarity, the Pearson Correlation Coefficient is another method that can be used. To calculate the similarity between users, Equation (3) can be utilized with μ_u representing the average score given by User u to every Item (Ricci, Rokach, & Shapira, 2015). The similarity between Item i and Item j can be obtained from Equation (4) (Ricci, Rokach, & Shapira, 2015).

$$\text{pearson\_sim}\,(u,\ v) = \frac{\sum_{i \in I_{uv}} (r_{u\,i} - \mu_u) \cdot (r_{v\,i} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u\,i} - \mu_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{v\,i} - \mu_v)^2}} \tag{3}$$

$$\text{pearson\_sim}(i,\ j) = \frac{\sum_{u \in U_{ij}} (r_{u\,i} - \mu_i) \cdot (r_{u\,j} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{u\,i} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{u\,j} - \mu_j)^2}} \quad (4)$$

The next step involves utilizing the similarity between users or the similarity between items to predict the Unknown Rating score of User u and Item i. This prediction process is shown in Equation (5) for User-Based Similarity and Equation (6) for Item-Based Similarity. The Estimated Rating that user u will give to Item i is represented by $\hat{r}_{ui}$. The similarity value between Users is represented by $sim(u,v)$, while the similarity value between Items is represented by $sim(i,j)$. Both similarity values are derived from the similarity metrics that are being used. The set of users who have rated Item i and are most similar to User u is represented by $N_i^k(u)$, while the set of Items that User u has rated and are most similar to Item i is represented by $N_u^k(i)$ (Ricci et al., 2015).

$$\hat{r}_{u\,i} = \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,\ v) \cdot r_{v\,i}}{\sum_{v \in N_i^k(u)} \text{sim}(u,\ v)} \quad (5)$$

$$\hat{r}_{u\,i} = \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,\ j) \cdot r_{u\,j}}{\sum_{j \in N_u^k(i)} \text{sim}(i,\ j)} \quad (6)$$

Once the score has been predicted, the algorithm can select the N Items with the highest prediction score to create a recommendation list for the user.

2.2 Matrix Factorization Based

Matrix factorization-based algorithms, such as SVD and SVD++ (Koren, Bell, & Volinsky 2009; Mnih & Salakhutdinov, 2007), are utilized to approximate the rating a user u has not yet assigned to an Item i. The underlying principle involves operating on a Rating Matrix (R), as illustrated in Figure 1, where white and black elements represent unknown and known ratings, respectively. The matrix R has dimensions m x n, where m represents the number of users and n denotes the number of items. The matrix is partitioned into two matrices: P of size n x k and Q^T of size k x m, where k represents the latent factor (Koren, 2010). The matrices P and Q^T are used to predict the Unknown Rating (Funk, n. d.).

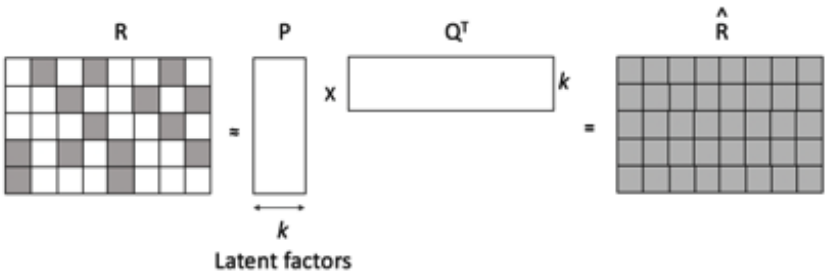

Figure 1 Approximating the ratings matrix with row and column factors (Ardimansyah, Huda, & Baizal, 2017)

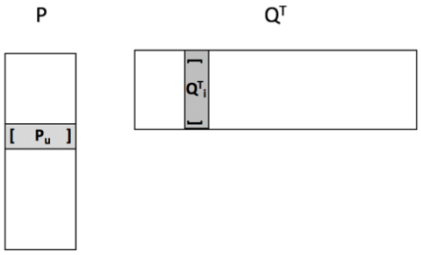

Figure 2 Calculating a predicted rating from the row and column factors (Qwiklabs, n. d.)

The loss function is specified by Equation (8) (Qwiklabs, n. d.), and to forecast the rating $\hat{r}_{ui}$ assigned by User u to Item i, the dot product of P_u (the vector at row u of P) and Q_i^T (the vector at column i of Q^T) is computed, as shown in Figure 2 (Qwiklabs, n. d.) and described in Equation (7).

$$\hat{r}_{u\,i} = q_i^T\, p_u \tag{7}$$

$$L = \sum_{u,i} (r_{u\,i} - \mathbf{q}_i^T \cdot \mathbf{p}_u)^2 \tag{8}$$

Equation (9) (Qwiklabs, n. d.) introduces the concept of incorporating regularized terms to avoid overfitting and predict the Unknown Rating.

$$L = \sum_{u,i} (r_{u\,i} - \mathbf{q}_i^T \cdot \mathbf{p}_u)^2 + \lambda \sum_i \| \mathbf{q}_i \|^2 + \lambda \sum_u \| \mathbf{p}_u \|^2 \tag{9}$$

2.3 Data Sparsity

Data sparsity is a significant challenge that impacts the effectiveness of Collaborative Filtering in rating systems (Bobadilla & Serradilla, 2009). To illustrate, consider a system with 1 million items, of which only 10,000 have been bought. Although the dataset seems extensive, the number of purchased items is only 1%, resulting in a sparse dataset with a density of 1% and sparsity of 99%. This sparsity issue leads to Unknown Ratings in the rating matrix.

2.4 Related Work

In a study conducted by Yehuda Koren in 2008, the accuracy of SVD, Asymmetric-SVD, and SVD++ algorithms were compared by analyzing their RMSE at varying numbers of latent factors, revealing that an increase in the number of latent factors led to a decrease in RMSE. Specifically, RMSEs of 0.9046, 0.9025, and 0.9009 were produced by the SVD algorithm, RMSEs of 0.9037, 0.9013, and 0.9000 were produced by the Asymmetric-SVD algorithm, and RMSEs of 0.8952, 0.8924, and 0.8911 were produced by the SVD++ algorithm at 50, 100, and 200 latent factors, respectively, although this increase also resulted in longer computation times. Therefore, researchers such as Zheng, Yang, He, and Huang (2016) focus on optimizing either accuracy or computational time. Additionally, for clustering users based on similar behavior, other techniques such as Singular Value Decomposition (SVD), Matrix Decomposition, and Cosine Similarity can be utilized.

The impact of data sparsity on performance was investigated by Bobadilla and Serradilla (2009) which demonstrated that predictive efficiency decreased with increasing sparsity levels, highlighting the significant issue of data sparsity in Collaborative Filtering. Furthermore, Mochamad Iqbal Ardimansyah, Huda, and Baizal (2017) conducted a study on the effect of preprocessing matrix factorization before estimating ratings and compared it with Collaborative Filtering without preprocessing.

3. Materials and Methods

In this study, two datasets of different sparsity levels were utilized, namely the MovieLens-100K Dataset and the Yahoo Movies dataset. The former was obtained by the GroupLens Research Lab at the University of Minnesota from 943 viewers who rated 1,682 movies, resulting in 100,000 records. On the other hand, the latter was based on the ratings of 7,642 viewers who rated 11,916 movies, resulting in 221,354 records. Both datasets were utilized to create a rating matrix comprising user_id, movie_id, and rating values from 1 to 5, as presented in Table 2.

Table 2 Rating Transaction

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

The MovieLens-100K rating matrix, which is illustrated in Table 3, is composed of 1,586,126 elements (1682 x 943). However, only 100,000 of these elements represent known ratings, while the remaining 1,486,126 are classified as Not a Number (NaN) because they are unrated. As such, the rating matrix has a sparsity of 93.69%, which was calculated using Equation (10) (Zhou & Luo, 2010).

$$sparsity = \frac{unknown\ elements}{matrix\ size} \times 100 \tag{10}$$

Table 3 Rating Matrix (Sparse Matrix)

user_id	1	2	3	4	5	6	7	8	9	10	…
0	5.0	4.0	NaN	NaN	4.0	4.0	NaN	NaN	NaN	4.0	…
1	3.0	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	…
2	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	…
3	3.0	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	4.0	…
4	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	…

A heatmap is presented in Figure 3, which depicts the distribution of sparsity/density based on the ratings. It was revealed from the analysis of the heatmap that movies with higher movie_id values were rated more frequently, potentially due to their longer availability and consequently larger audience, while relatively fewer viewers rated movies with lower movie_id values.

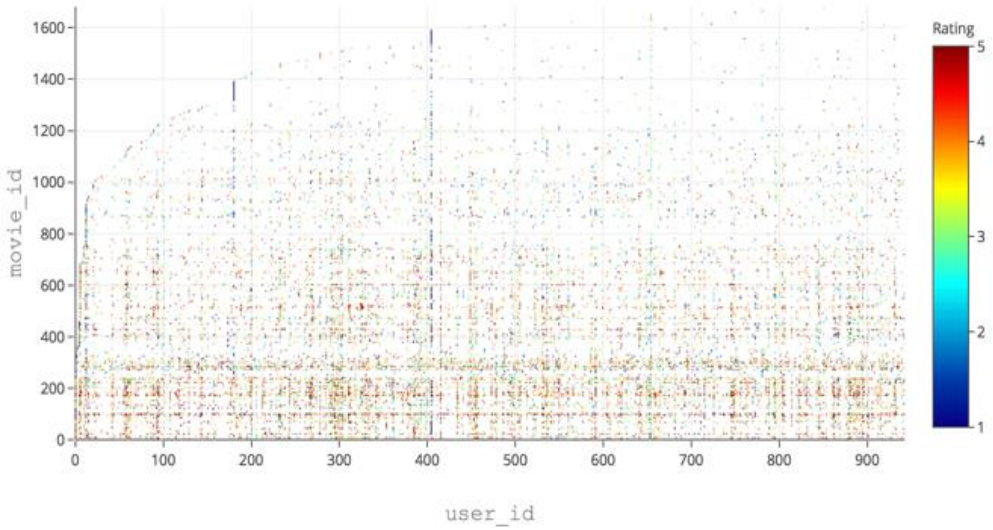

Figure 3 Heatmap of Ratings

The Yahoo Movies Dataset, on the other hand, has a rating matrix of 91,062,072 elements, with only 221,364 representing known ratings and the remaining 90,840,708 classified as Unknown Ratings. This translates to a sparsity of 99.75%, as presented in Table 4.

Table 4 Size and Sparsity Comparison

Dataset	Matrix Size	Unknown Rating	Known Rating	Sparsity (%)
MovieLens (100k)	1,586,126	1,486,126	100,000	93.69
Yahoo Movies	91,062,072	90,840,708	221,364	99.75

4. Results and Discussion

The study evaluated the performance of the group algorithm using two methods: Neighborhood-Based Analysis with Similarity Metric Cosine and Pearson, and Matrix Factorization Based Algorithm. Table 5 and Table 6 show the average RMSE from 5-fold cross-validation and processing time.

According to Table 5, the MovieLens-100K dataset was evaluated using the KNN (Cosine), KNN (Pearson), SVD, and SVD++ algorithms, with their performance measured in terms of RMSE. The obtained results were 0.96197, 0.95715, 0.94526, and 0.92847, respectively. Regarding the computational time, the recorded values were 0:00:07, 0:00:07, 0:00:06, and 0:02:19, respectively.

Table 5 Compare the performance of each algorithm on MovieLens-100K Dataset

Algorithms	RMSE	Time
KNN (Cosine)	0.96197	0:00:07
KNN (Pearson)	0.95715	0:00:07
SVD	0.94526	0:00:06
SVD++	0.92847	0:02:19

In Table 6, the results obtained on the YahooMovies Dataset are presented. Specifically, the KNN (Cosine) algorithm achieved an RMSE of 1.04596 and took 0:00:49 to run, while KNN (Pearson) achieved an RMSE of 1.07740 and took 0:00:35. The SVD algorithm had an RMSE of 1.01193 and required 0:00:10 to complete. Lastly, the SVD++ algorithm took 0:04:08 to run and resulted in an RMSE of 0.99915.

Table 6 Compare the performance of each algorithm on YahooMovies Dataset

Algorithms	RMSE	Time
KNN (Cosine)	1.04596	0:00:49
KNN (Pearson)	1.07740	0:00:35
SVD	1.01193	0:00:10
SVD++	0.99915	0:04:08

The experimental findings reveal that SVD++ outperformed the other algorithms on both datasets, as evidenced by its lower RMSE. SVD and KNN came second and third in performance, respectively. However, it is important to consider that SVD++ demands considerably more computational resources than the other algorithms.

5. Conclusion

In summary, the present study aimed to predict the Unknown Ratings of two datasets, namely MovieLens-100k and YahooMovies, with varying matrix sizes and sparsity percentages. Based on the results, SVD++ and SVD, both belonging to the Matrix Factorization Based category, emerged as the best-performing algorithms. However, it is important to note that SVD++ requires considerably more computational time than SVD. Moreover, the Neighborhood-Based Pearson Similarity algorithm exhibited higher accuracy than Cosine Similarity in the group algorithm section. Finally, when comparing the RMSE values, MovieLens-100k, despite having lower sparsity, outperformed YahooMovies, which had a higher sparsity level.

6. References

Ardimansyah, M. I., Huda, A. F., & Baizal, Z. K. A. (2017). Preprocessing matrix factorization for solving data sparsity on memory-based Collaborative Filtering. *2017 3rd International Conference on Science in Information Technology (ICSITech)*, p. 521–525. https://doi.org/10.1109/ICSITech.2017.8257168

Bobadilla, J., & Serradilla, F. (2009). The Effect of Sparsity on Collaborative Filtering Metrics. In *Conferences in Research and Practice in Information Technology Series*, *92*, p. 17).

Funk, S. (n. d.). *Netflix Update: Try This at Home*. Retrieved 3 March 2023, from https://sifter.org/simon/journal/20061211.html

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434. https://doi.org/10.1145/1401890.1401944

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data*, *4*(1), 1:1-1:24. https://doi.org/10.1145/1644873.1644874

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37. https://doi.org/10.1109/MC.2009.263

Mangalindan, J. (n. d.). *Amazon's recommendation secret | Fortune*. Retrieved 3 March 2023, from https://fortune.com/2012/07/30/amazons-recommendation-secret/

McAlone, N. (n. d.). *Why Netflix thinks its personalized recommendation engine is worth $1 billion per year*. Business Insider. Retrieved 3 March 2023, from https://www.businessinsider.com/netflix-recommendation-engine-worth-1-billion-per-year-2016-6

Mnih, A., & Salakhutdinov, R. R. (2007). Probabilistic Matrix Factorization. NeurIPS Proceedings on *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, p. 1-8. https://proceedings.neurips.cc/paper/2007/hash/d7322ed717dedf1eb4e6e52a37ea7bcd-Abstract.html

Qwiklabs. (n. d.). *Recommendation Systems on Google Cloud | Google Cloud Skills Boost*. Qwiklabs. Retrieved 3 March 2023, from https://www.cloudskillsboost.google/course_templates/39

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems: Introduction and Challenges. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 1–34). Springer US. https://doi.org/10.1007/978-1-4899-7637-6_1

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative Filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, *69*, 29–39. https://doi.org/10.1016/j.eswa.2016.09.040

Zheng, L., Yang, S., He, J., & Huang, Z. (2016). An optimized Collaborative Filtering recommendation algorithm. *2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT)*, p. 89–92. https://doi.org/10.1109/CCIOT.2016.7868309

Zhou, J., & Luo, T. (2010). A novel approach to solve the sparsity problem in Collaborative Filtering. *2010 International Conference on Networking, Sensing and Control (ICNSC)*, p. 165–170. https://doi.org/10.1109/ICNSC.2010.5461512