Evaluation of Collaborative Filtering for Recommender Systems

Maryam Al-Ghamdi[1], Hanan Elazhary[2], Aalaa Mojahed[3]
College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia[1,2,3]
Computers and Systems Department Electronics Research Institute Cairo, Egypt[2]

***Abstract*—Recently, due to the increasing amount of data on the Internet along with the increase in products' purchasing via e-commerce websites, Recommender Systems (RS) play an important role in guiding customers to buy products they may prefer. Furthermore, these systems help the companies to advertise their products to the most potential customers, and therefore raise their revenues. Collaborative Filtering (CF) is the most popular RS approach. It is classified into memory-based and model-based filtering. Memory-based filtering is in turn classified into user-based and item-based. Several algorithms have been proposed for CF. In this paper, a comparison has been performed between different CF algorithms to assess their performance. Specifically, we evaluated K-Nearest Neighbor (KNN), Slope One, co-clustering and Non-negative Matrix Factorization (NMF) algorithms. KNN algorithm is representative of the memory-based CF approach (both user-based and item-based). The other three algorithms, on the other hand, are under the model-based CF approach. In our experiments, we used a popular MovieLens dataset based on six evaluation metrics. Our results reveal that the KNN algorithm for item-based CF outperformed all other algorithms examined in this paper.**

***Keywords*—Co-clustering; collaborative filtering; KNN; NMF; recommender systems; slope one**

I. Introduction

Nowadays, most people tend to buy products from online websites and due to the huge amount of data available on the Internet, making the right decision to choose the most appropriate products has become more difficult. Thus, tools like Recommender Systems (RS) are very necessary to help them to make the right decisions.

RS can be defined as software tools and techniques that help the user in decision-making processes, such as what products to buy, what books to read, and what movies to watch [1]. Furthermore, these systems help the companies to raise their revenues. Amazon and eBay are examples of companies that strongly depend on RS to increase their sales and financial profits. RS can be generally classified into two main categories, Content-based Filtering (CBF), and Collaborative Filtering (CF) [1]. CBF is one of the simplest approaches in RS. It recommends to the users a list of items that are similar to the items they liked in the past. The system analyzes the item's textual information, such as item's descriptions and user's preferences, then finds the similar items to the ones they liked in the past. After that, CBF makes recommendations using some classification algorithms [2]. For example, the system recommends to the users books from the same genre of the books they already liked or recommends a product with a shape and color similar to what they liked before.

CF is the most popular recommender systems approach. It recommends items based on the user's past behavior as well as similar decisions made by other users. The first CF system that was proposed is Tapestry. It was developed by Goldberg et al. [3] in 1992. Tapestry is mainly developed to handle the problem of a huge stream of incoming documents via e-mail. They proposed a way to use CF, in addition to CBF, to filter documents coming from e-mails. Their simple idea of CF is that people help each other to filter these documents by recording their reactions to them. In this research, we focus on the CF approach only, since it is the most popular and generally more efficient in comparison to other approaches. The CF approach is categorized into memory-based and model-based [4]. In this paper, we evaluate and compare several algorithms under those two classes using a popular MovieLens dataset and based on six evaluation metrics.

The rest of this paper is structured as follows: Section II describes the different CF algorithms. Related work is summarized in Section IV. Evaluation metrics are presented in Section III. Section V discusses the research methodology that we adopted. Results are discussed in Section VI. Finally, a conclusion is provided in Section VII.

II. Collaborative Filtering

In this section we discuss the two classes of the collaborative filtering approach and the corresponding algorithms that we evaluate in this paper.

A. Memory-based Approach

Memory-based or *neighborhood-based* approach uses user-item ratings matrix to generate the recommendations [5]. From this matrix, the system computes the similarities between users, or between items. Then, it saves computed similarity scores to a similarity matrix. There are several methods that have been used to calculate the similarities such as Euclidean, cosine, and mean squared distances.

The memory-based methods suffer from two main issues, which are sparsity and scalability [6]. In the sparsity case, it will be hard for the system to provide good recommendations due to the small number of items that each user rated. Scalability problem occurs when the numbers of users and items increase exceedingly. In such case, there will be a lot of information and it will be hard for the system to deal with it [6]. As previously noted, the memory-based approach is classified into user-based and item-based filtering [7].

a) User-based:
In the user-based approach, recommendations are made based on similar user preferences. K-Nearest Neighbor (KNN) is one of the algorithms that could be used in this approach. Equation (1) shows the prediction formula:

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} sim(u,v)} \tag{1}$$

Where $\hat{r}_{ui}$ is the predicted rating of user u for item i, μ_u is the mean of all ratings given by user u and $sim(u, v)$ is the similarity value between users u and v. The value of $sim(u, v)$ can be computed using cosine similarity measure as shown in the following equation:

$$sim(u,v) = \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}} \tag{2}$$

b) Item-based:
In the item-based approach, the system makes recommendations based on the similarities among items. KNN prediction formula for item-based CF is as follows:

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} sim(i,j)} \tag{3}$$

Where $\hat{r}_{ui}$ is the predicted rating of user u for item i, μ_i is the mean of all ratings given to item i and $sim(i, j)$ is the similarity value between items i and j. The value of $sim(i, j)$ can be computed using cosine similarity measure as follows:

$$sim(i,j) = \frac{\sum\limits_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum\limits_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} r_{uj}^2}} \tag{4}$$

B. Model-based Approach

To overcome the aforementioned issues of the memory-based approach, the model-based approach has been proposed. Model-based CF works by grouping different users into a small number of classes based on their ratings patterns. Many machine learning algorithms can be used to build such a model. In this research, we focused on three algorithms which are: Slope One, co-clustering, and NMF.

a) Slope one:
Lemire et al. [8] proposed a model-based CF algorithm called Slope One. One of strengths of this algorithm is that it takes into account two types of information, information about other users who rated the same item (similar to user-based), and information about other items that the same user rated before (similar to item-based).

We will illustrate the basis of the Slope One approach by an example. Suppose we have two users, A and B, and two items, i and j as shown in Table I. Item j is rated by both users (A and B). User A gave it a rating of 2, while user B gave it a rating of 4. User A also gave item i a rating of 2.5. We notice that item i is rated more than j by 2.5 - 2 = 0.5. Now we can use this information to predict that user B will give item i a rating of 4 + 0.5 = 4.5.

TABLE I. SLOPE ONE EXAMPLE.

	Item i	**Item j**
User A	2.5	2
User B	N/A	4

The process of the prediction is done by computing the average differences between the ratings of one item and another for users who rated both. The prediction formula for Slope One algorithm is as follows [8]:

$$\hat{r}_{ui} = \mu_u + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} dev(i,j) \tag{5}$$

Where $\hat{r}_{ui}$ is the predicted rating of user u for item i, μ_u is the mean of all ratings given by user u and $R_i(u)$ is the set of items j rated by user u which also have at least one common user with item i. $dev(i, j)$ is considered as the average difference between item i's ratings (r_{ui}) and item j's ratings (r_{uj}) as shown in the following equation [8]:

$$dev(i,j) = \frac{1}{|U_{ij}|} + \sum_{u \in U_{ij}} r_{ui} - r_{uj} \tag{6}$$

b) Co-clustering:
Clustering is a powerful technique in the data mining field that refers to the process of grouping objects in a way that similar objects will belong to the same group or cluster. There are various clustering methods that could be used based on the type of the data. In case of CF, the data is the user-item ratings matrix, so we need a way to cluster rows and columns. This process is called co-clustering. In this paper, we used co-clustering algorithm that has been proposed by George et al. in [9]. This algorithm is based on weighted co-clustering algorithm proposed in [10]. The idea is to compute the neighborhoods for the users and items via co-clustering and then make predictions according to the average ratings of the co-clusters while taking into consideration the users and items individual biases. The prediction formula is as the following:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}) \tag{7}$$

Where $\overline{C_{ui}}$ is the average rating of co-cluster C_{ui}, and $\overline{C_u}$ is the average rating of u's cluster, $\overline{C_i}$ is the average rating of i's cluster, μ_u is user u's average rating and μ_i is item i's average rating.

It is worth noting that if the user is new (not existing before) but the item is known, the prediction value $\hat{r}_{ui}$ will be the average rating given to item i. If the item is unknown (new) but the user is known, the prediction value $\hat{r}_{ui}$ will be the average rating given by user u. In case both the user and the item are unknown, the prediction value of $\hat{r}_{ui}$ will be the global average of all the existing ratings.

c) Non-negative matrix factorization (NMF):
Matrix Factorization-based (MF-based) modeling is one of the CF approaches that are widely used in recent years. It is highly accurate and scalable in several cases [11]. MF-based models work by decomposing the user-item matrix into two low-rank matrices. The first one is user-features matrix and the other is item-features matrix. We can make any predictions by calculating the dot product of two lower dimensionality rectangular matrices.

Various matrix factorization algorithms have been proposed, such as Singular Value Decomposition (SVD), Probabilistic Matrix Factorization (PMF) and Non-negative Matrix Factorization (NMF). In this research, we focus on NMF as an example of the MF approach. In this algorithm, the prediction $\hat{r}_{ui}$ is computed as follows:

$$\hat{r}_{ui} = q_i^T p_u \tag{8}$$

Where q_i is an item factors matrix and p_u is user factors matrix.

Different optimization algorithms could be used in MF-based models. The NMF algorithm uses Stochastic Gradient Descent (SGD) optimization algorithm. At each step of the SGD procedure, the factors (features) f of user u and item i are updated as follows:

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r_{ui}} + \lambda_u |I_u| p_{uf}} \tag{9}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r_{ui}} + \lambda_i |U_i| q_{if}} \tag{10}$$

where λ_u and λ_i are regularization parameters.

III. Evaluation Metrics

Several metrics have been proposed to evaluate the performance of recommender system's algorithms. In addition to training time and testing time, examples of those metrics include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Fraction of Concordant Pairs (FCP), and coverage. In the following subsections, we will present each of the latter four metrics in details.

A. Mean Absolute Error (MAE)

MAE computes the average absolute difference between the observed and predicted ratings. The MAE is given by:

$$MAE = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}| \tag{11}$$

Where $|\hat{R}|$ is the total number of predicted ratings, r_{ui} is the true rating value that user u gave to item i and $\hat{r}_{ui}$ is the predicted rating value that user u gave to item i. A lower value of MAE means the predictions are more accurate and so the performance of the algorithm is better.

B. Root Mean Squared Error (RMSE)

RMSE is very similar to MAE, except that instead of summing the absolute values of the rating prediction errors, we sum their squares using the following formula:

$$RMSE = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2} \tag{12}$$

Where $|\hat{R}|$ is the total number of predicted ratings, r_{ui} is the true rating value that user u gave to item i and $\hat{r}_{ui}$ is the predicted rating value that user u gave to item i. A lower value of RMSE means the predictions are more accurate and so the algorithm's performance is better.

C. Fraction of Concordant Pairs (FCP)

One of the issues for MAE and RMSE is that they don't take into consideration rating scales that vary from one user to another [12]. Thus, in addition to MAE and RMSE, we have used FCP to evaluate the algorithms. It is calculated using equation 13 [12] such that a higher value of FCP means the algorithm is more accurate.

$$FCP = \frac{n_c}{n_c + n_d} \tag{13}$$

Where,

$$n_c = \sum | \; \{(i,j) \,|\, \hat{r_{ui}} \; > \; \hat{r_{uj}} \; and \, r_{ui} \; > \; r_{uj} \; \} \, | \tag{14}$$

$$n_d = \sum | \; \{(i,j) \,|\, \hat{r_{ui}} \; < \; \hat{r_{uj}} \; and \, r_{ui} \; < \; r_{uj} \; \} \, | \tag{15}$$

D. Coverage

Coverage refers to the percentage of items that the system was able to successfully recommend. It is computed using the following formula [13]:

$$Coverage = \frac{n_{pi}}{n_i} \tag{16}$$

Where n_i is the total number of items that the system predict and n_{pi} is the total number of items that were successfully predicted by the system.

IV. Related Work

This section discusses the research papers that compared different RS algorithms. Benin in [14] made a comparison of RS for crowdfunding projects. This study aims to compare different types of RS which are CBF, CF and hybrid RS, which combines both CBF and CF. The popular MovieLens-1M [15] dataset was used in the experiments. To evaluate the algorithms, they did both quantitative and qualitative analysis. The quantitative analysis relied on RMSE and MAE. The qualitative analysis, which is the analysis of the quality of the produced recommendations, was achieved via eyeballing-produced recommendations. However, evaluating the recommendations using this method is considered primitive and imprecise since it may vary from one point of view to another. This study concluded that hybrid RS outperform CF and CBF.

Arsan et al. [16] made a comparison between user-based and item-based algorithms to observe their performance and accuracy. Since similarity measures play an important role in the user-based and item-based predictions accuracy, they applied various similarity measures, which are Euclidean distance, log likelihood ratio, Pearson correlation coefficient, Tanimoto coefficient, uncentered cosine, and Spearman correlation coefficient. The authors applied the algorithms to the MovieLens-100K dataset. MAE and RMSE were used to evaluate the algorithms' accuracy. Time spent to make the recommendations was also calculated. Based on their experiments, they concluded that item-based algorithms perform better than user-based algorithms.

Najafi et al. [17] assessed the item-based CF and the MF-based FunkSVD algorithms. The idea of their study is to compare the performance of these algorithms when the data is scaled. MovieLens 100k and MovieLens-1M were used. They used MAE and RMSE to evaluate the algorithms. Their results shows that the FunkSVD algorithm is more accurate than the item-based CF when the data is scaled.

Lemire et al. [8] proposed three algorithms which are Slope One, weighted Slope One and bi-polar Slope One. They compared their proposed algorithms with four other algorithms which are: bias from mean, adjusted cosine item-based (model-based), per user average and Pearson (memory-based) algorithms. Both EachMovie [18] and MovieLens datasets were used in their experiments. They tested the algorithms using the evaluation metric MAE. Their results showed that the proposed Slope One algorithms achieved comparable accuracy to the other selected algorithms.

George and Merugu [9] proposed a novel CF algorithm which is based on weighted co-clustering algorithm [10]. They compared their proposed algorithm with SVD, NMF, and classic correlation-based CF algorithms. The experiments were applied on MovieLens-100K dataset. The MAE was used to compute the prediction accuracy. Their results indicate that their proposed algorithm has a high accuracy with much lower computational cost in comparison to the other algorithms.

Our comparison is different from the above-discussed papers in many aspects. First, we made a comparison between memory-based (both user-based and item-based) KNN algorithms and model-based (NMF, Slope One, and co-clustering) algorithms. Besides, we evaluated these algorithms using six different metrics which are MAE, RMSE, FCP, coverage, training time and testing time.

V. Methodology

This section discusses the methodology used to complete this research. Section V-A introduces the dataset used in the experiments and some of its statistical analysis results. Section V-B describes our experimental setup including hyperparameters tuning and used libraries.

A. Dataset Exploration

We have used MovieLens-25M dataset [15], which is one of the most popular datasets used by researchers in the field of CF. It describes a 5-star rating and tagging activity. It contains 62,423 movies, 25,000,095 ratings and 1,093,360 tag applications. However, in this research, we have focused on the CF algorithms, where only the ratings data are considered. The data were created by 162,541 users throughout 24 years and 10 months, from 9 January 1995 to 21 November 2019 and it was released on December 2019. In this dataset, the users are randomly selected and each user is represented merely by an Id [15]. Fig. 1 shows the ratings histogram for the MovieLens dataset. From the figure, we notice that 26.56% of the ratings are 4.0 and 19.59% are 3.0. This indicates that users tend to rate the movies they preferred. However, in our experiment, we haven't used the whole dataset because the memory-based filtering algorithms don't scale very well to such a big data size. So, we have randomly selected 100,000 ratings of 54,778 users on 10,271 movies with a sparsity of 99.9822%.

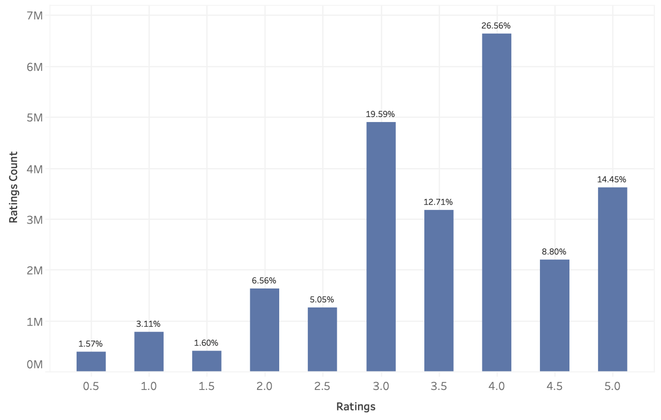


Fig. 1. Ratings Distribution for MovieLens Dataset.

B. Experimental Setup

In this section, we discuss our experimental setup. First, in our experiment with the KNN algorithm, we selected cosine similarity measure to compute the similarities between users or items. Second, in order to achieve the highest machine learning predictive model performance, hyperparameter values need to be selected carefully. This is one of the important steps in building any machine learning model [19]. Table II shows the hyperparameters, the corresponding test values that we selected to optimize the algorithms' performance and the best values that we obtained. The test values were chosen with the help of the default values in the library [20] and values found in other studies in the related work.

Third, for training, we adopted K-fold cross validation, with k equals 5. In this method, the dataset is divided into K folds and the model is trained K times, each time on different K-1 folds, and then tested on the remaining fold. The average performance of the K results is then calculated. K-fold cross validation approach avoids overfitting to the particular division of the training and testing sets that may appear in the other approaches, which split the data into training and testing sets and run the algorithm once [1], [21].

The code used in this research was implemented mainly using Python. Surprise library was used, which is a Python

TABLE II. List of Hyperparameter Values

Method	Hyperparameter	Test values	Best value
KNN	k, number of neighbors	10, 20, 30	10
Co-clustering	n_cltr_u, number of user clusters	3, 5	3
	n_cltr_i, number of item clusters	3, 5	5
NMF	n_factors, the number of factors	5, 10, 15	15
	n_epochs, the number of iterations of SGD	15, 20, 25	20

Scikit library for CF [20]. In addition, Pandas [22] and Numpy [23] libraries were used. All the work is done using MacBook Pro with CPU 2.5 GHz Intel Core i5 and 16 GB RAM. It is worth noting that in order to visualize the dataset and our results in graphs, Tableau desktop software - professional edition, version 2020.3 was used.

VI. Results and Discussion

In this section, we will discuss the results of our experiments. As observed in Fig. 2, the item-based KNN CF algorithm outperformed all the other algorithms in terms of MAE, RMSE and FCP. It also took less training time compared to the others. This is at the expense of taking longer testing time in comparison to Slope One, NMF, and co-clustering algorithms. Regarding the user-based KNN algorithm, it's noticeable that it is less effective in terms of accuracy and speed; it was too slow in both training and testing. These results were expected since our dataset contains 10,271 movies only while the number of users is as large as 54,778. This surely has a significant effect in helping the item-based CF algorithm to work very well compared to others.

However, when we look at the coverage results, all the algorithms except co-clustering were not able to make predictions for all the testing dataset. Specifically, they were able to predict only about 57% of the dataset while co-clustering was able to predict 100% of it. The full results for all the algorithms are reported in Table III.

VII. Conclusion

In this paper, we have performed a comparison between five different CF algorithms to assess their performance. The selected algorithms are KNN for user-based, KNN for item-based, Slope One, co-clustering, and NMF. The algorithms have been evaluated using six metrics which are MAE, RMSE, FCP, coverage, training time and testing time. Our results show that the KNN algorithm for item-based CF outperformed all other algorithms examined in this paper. It achieved the lowest error values and thus the highest accuracy. As future work, we plan to run the algorithms on a larger sample of the dataset to assess their scalability. In addition, we plan to consider more algorithms and more evaluation metrics.

References

[1] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Springer US, 2011. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-85820-3

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/421425

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992. [Online]. Available: http://doi.acm.org/10.1145/138859.138867

[4] L. E. Molina and S. Bhulai, "Recommendation system for netflix," 2018.

[5] M.-P. T. Do, D. Nguyen, and L. Nguyen, "Model-based approach for collaborative filtering," in *6th International Conference on Information Technology for Education*, Ho Chi Minh City, Vietnam, August 2010, pp. 217–228.

[6] S. Gong, H. Ye, and H. Tan, "Combining memory-based and model-based collaborative filtering in recommender system," in *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, Chengdu, China, May 2009, pp. 690–693.

[7] X. Liang, Z. Xia, L. Pang, L. Zhang, and H. Zhang, "Measure prediction capability of data for collaborative filtering," *Knowledge and Information Systems*, vol. 49, no. 3, pp. 975–1004, 2016.

[8] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, vol. 5, 2007.

[9] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, November 2005, pp. 625–628.

[10] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: Association for Computing Machinery, December 2004, pp. 509–514. [Online]. Available: https://doi-org.sdl.idm.oclc.org/10.1145/1014052.1014111

[11] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[12] Y. Koren and J. Sill, "Collaborative filtering on ordinal user feedback," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13. Beijing, China: AAAI Press, Jun 2013, pp. 3022–3026. [Online]. Available: http://dl.acm.org/citation.cfm?id=2540128.2540570

[13] F. Hdioud, B. Frikh, and B. Ouhbi, "A comparison study of some algorithms in recommender systems," in *2012 colloquium in information science and technology*. IEEE, Oct 2012, pp. 130–135.

[14] A. C. Benin, "A comparison of recommender systems for crowdfunding projects," *Adriano Carniel Benin*, July 2018.

[15] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19:1–19:19, 2015. [Online]. Available: http://doi.acm.org/10.1145/2827872

[16] T. Arsan, E. Köksal, and Z. Bozkus, "Comparison of collaborative filtering algorithms with various similarity measures for movie recommendation," *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, vol. 6, no. 3, pp. 1–20, 2016.

[17] Z. Salam Patrous and S. Najafi, "Evaluating prediction accuracy for collaborative filtering algorithms in recommender systems," *KTH Royal Institute of Technology*, 2016.

[18] P. McJones, "Eachmovie Collaborative Filtering Dataset, DEC Systems Research Center,http://www.research.compaq.com/src/eachmovie/," 1997.

[19] J. Gaudillo, J. J. R. Rodriguez, A. Nazareno, L. Baltazar, J. Vilela, R. Bulalacao, M. Domingo, and J. Albia, "Machine learning approach to single nucleotide polymorphism-based asthma prediction," *PLOS ONE*, vol. 14, p. e0225574, 2019.

TABLE III. Results.

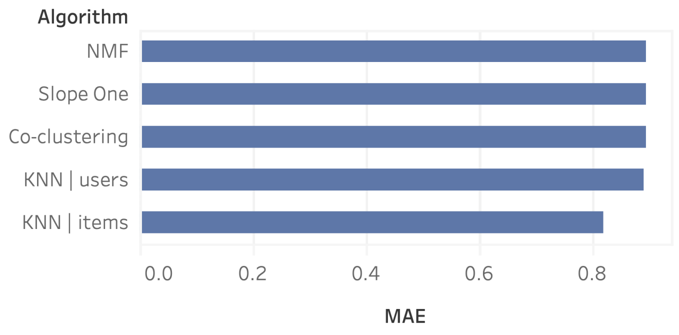
Metric	**KNN (User-based)**	**KNN (Item-based)**	**Slope One**	**Co-clustering**	**NMF**
MAE	0.8891	0.8191	0.8934	0.8934	0.8940
RMSE	1.1513	1.0446	1.1575	1.1437	1.1365
FCP	0.4883	0.5202	0.4928	0.5097	0.5138
Coverage	57.07%	57.07%	57.07%	100%	57.07%
Training Time (in sec)	196.2037	3.8485	4.3010	13.7191	4.8723
Testing Time (in sec)	1.5243	0.3690	0.3147	0.2202	0.3043

[20] N. Hug, “Surprise: A python library for recommender systems,” *Journal of Open Source Software*, vol. 5, no. 52, p. 2174, 2020. [Online]. Available: https://doi.org/10.21105/joss.02174

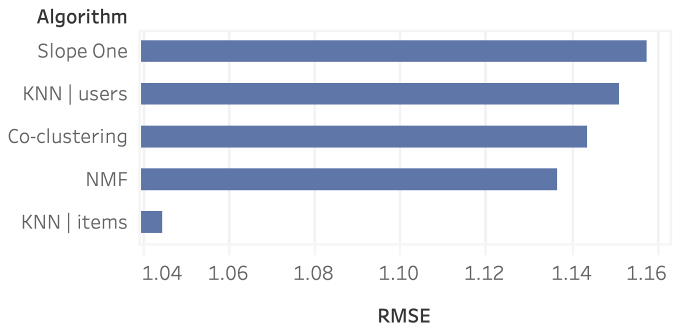
[21] K. Falk, *Practical recommender systems*. Shelter Island, NY: Manning Publications, 2019.

[22] T. pandas development team, “pandas-dev/pandas: pandas,” 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

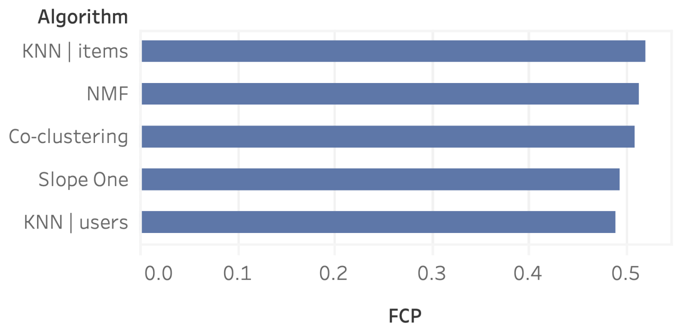
[23] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2



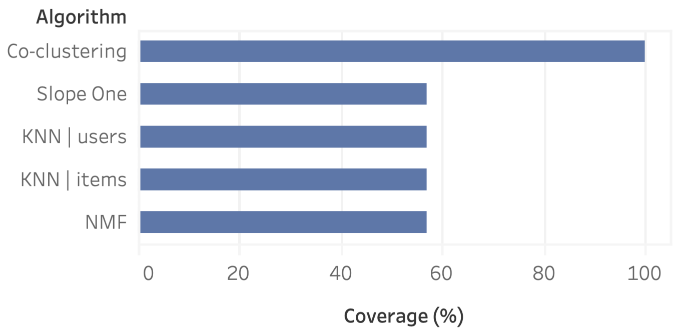
(a) MAE results



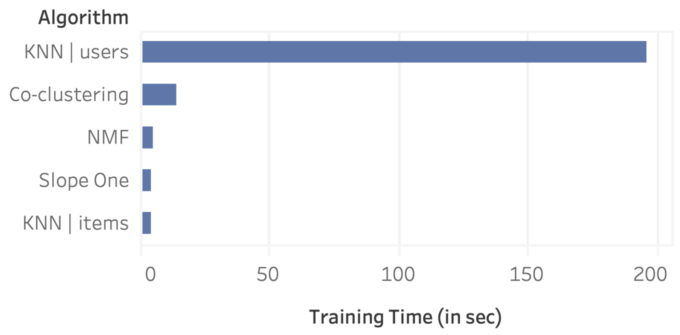
(b) RMSE results



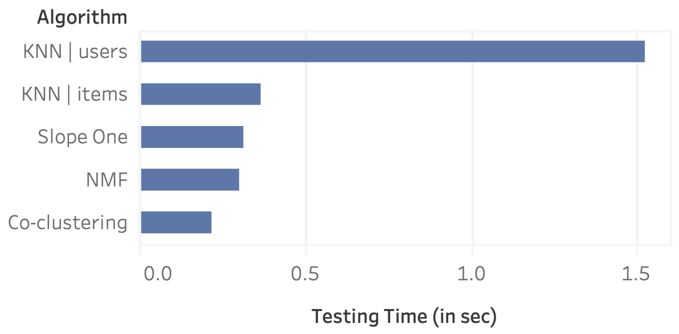
(c) FCP results



(d) Coverage percentage results



(e) Training time results (in sec)



(f) Testing time results (in sec)

Fig. 2. Results for all Algorithms.