Collaborative Filtering for Book Recommendation System

Gautam Ramakrishnan, V. Saicharan, K. Chandrasekaran, M. V. Rathnamma and V. Venkata Ramana

Abstract Collaborative filtering is one of the most important techniques in the market nowadays. It is prevalent in almost every aspect of the internet, in e-commerce, music, books, social media, advertising, etc., as it greatly grasps the needs of the user and provides a comfortable platform for the user to find what they like without searching. This method has a few drawbacks; one of them being, it is based only on the explicit feedback given by the user in the form of a rating. The real needs of a user are also demonstrated by various implicit indicators such as views, read later lists, etc. This paper proposes and compares various techniques to include implicit feedback into the recommendation system. The paper attempts to assign explicit ratings to users depending on the implicit feedback given by users to specific books using various algorithms and thus, increasing the number of entries available in the table.

Keywords Recommender systems · Book recommendation · Collaborative filtering · Implicit feedback · Explicit ratings

G. Ramakrishnan (✉) · V. Saicharan · K. Chandrasekaran
National Institute of Technology, Surathkal, Karnataka, India
e-mail: 16co219.gautam@nitk.edu.in

V. Saicharan
e-mail: vsaicharan1998@gmail.com

K. Chandrasekaran
e-mail: kchnitk@ieee.org

M. V. Rathnamma
Kandula Srinivasa Reddy Memorial College of Engineering, Kadapa, Andhra Pradesh, India
e-mail: rathnamma@ksrmce.ac.in

V. V. Ramana
Chaitanya Bharathi Institute of Technology, Proddatur, Andhra Pradesh, India
e-mail: ramanajntusvu@gmail.com

© Springer Nature Singapore Pte Ltd. 2020

K. N. Das et al. (eds.), *Soft Computing for Problem Solving*,
Advances in Intelligent Systems and Computing 1057,
https://doi.org/10.1007/978-981-15-0184-5_29

1 Introduction

Recommender systems are type of intelligent systems which can analyze past user behavior on items or service to make personalized recommendations on items to users. The two most widely used methods in recommendation systems are content-based algorithms and collaborative filtering memory-based methods. These are a set of methods which use the entire database (user-item database) to make a recommendation to a user. These methods can also be classified into two types: Item-based collaborative filtering: In this particular method, two items are compared and given a similarity score between the two using some sort of metrics like the cosine distance, Pearsons correlation, or Jaccard distance. User-based collaborative filtering: In this method, two users are compared for similarity and assigned a similarity score between them. The same metrics as mentioned above are used here. Using the similarity scores between the users, active recommendations are made for a user. Content-based filtering methods work by constructing a model to map a user to a set of items using either explicit features like ratings or implicit features like search history or number of page views. They compare user preferences to item description to make their predictions. Recommendation systems are very crucial for a company to be successful. Netflix claims their recommender system saves them $1bn every year [1]. In this paper, we discuss one particular problem of recommender systems in general. The problem discussed is the lack of inclusion of implicit data in recommendation systems. Elaborating, there exist two types of data on which similarity can be calculated and recommendations can be provided. The first type is explicit data, which consists of data which user explicitly provides, in the form of ratings, or small surveys. This data is generally well understood as it provides a concrete understanding of user interests. The other type of data is implicit ratings. This is the type of data which the user implicitly provides, through their actions on the platform, including but not restricted to page views, adding to wish lists, commenting on an item, etc. This type of data is generally vague and less understood, as the user does not provide a very clear indication of their preferences for that particular item. One of the significant issues faced by recommender systems is the inclusion of implicit ratings into the recommendation. Implicit feedback is a very dominant metric in determining recommendations as the innate requirements of the user are expressed very freely in their online behavior. However, due to the very vague and misunderstood nature of implicit data, it becomes a very tedious task to implement recommendation using implicit data. Also, the likelihood of the user rating an object is minimal compared to the data received from studying browsing patterns of the users.

Also, the recent trend is that many major organizations have integrated implicit ratings into their recommendation systems. For example, YouTube recommends videos by checking which videos are generally viewed by users in the same viewing session [2]. This is classified as implicit feedback as watching a video does not involve a user giving a rating. Also, most music streaming applications recommend music based on what music the user frequently listens to, rather than expecting a user to assign a rating for a song. In this paper, we discuss content-based collaborative filtering for a book recommendation system and propose solutions to improve collaborative, item–item based filtering by using implicit data. The implicit data in this case is the list of books each user has marked for future reading, but has not rated explicitly. The same method can be extended to other implicit data.

This paper is divided into five main sections. Section 2 deals with the past work that has been carried out in the domain of recommender systems and implicit feedback. Section 3 discusses the implementation of the book recommender system and various approaches for including implicit feedback. Section 4 elaborates on all the algorithms implemented. Section 5 discusses the outcome of each method and compares the efficiency of various methods stated. Section 6 concludes the paper, followed by future work.

2 Related Works

The related works described here are the two collaborative filtering methods, memory, and model-based. Similarity metrics, evaluation of the system and implicit feedback are also discussed. The two filtering methods form the core part of recommender systems and are described below, along with their submethods.

2.1 Collaborative Filtering

Collaborative-based filtering methods predict items for a particular user using data of other similar users or items. These methods are very popular on the web today and are used by E-commerce giants like Amazon, eBay, Flipkart, and others. They are also used in the domain of social media to recommend other users who are similar and also used to recommend news articles. Spotify, the music application uses this particular technique to recommend personalized music tracks and albums to its customers [3]. It has enjoyed great popularity and success mainly due to its simplicity in implementation and effectiveness. Collaborative filtering can be performed in two ways mainly, model-based and memory-based.

2.2 Memory Based

In memory-based collaborative filtering, the whole dataset is used to make a recommendation.

2.2.1 User Based

In user-based filtering, similarity between users is found based on how similar their ratings are for the books they have rated. It suffers from serious drawbacks of scalability and sparsity [4]. With a large number of users ranging in the order of millions, it is difficult to account for new users to join. Also, even the most active user would have rated less than 1% of the items in general, which makes it difficult for the nearest neighbors algorithm to make a decision.

2.2.2 Item Based

In item-based filtering, the similarity is found between every pair of items, by matching how many common users have rated the same pair of books. It was initially introduced by Amazon in 1998, due to its distinct advantages over user-based collaborative filtering as the matrix size for item similarity is generally smaller and the items are more stable compared to users [5].

2.3 Model Based

This method does not use the complete dataset for generating recommendations. Instead, this technique works by using a user-item database as training data to learn a model, which might be simple or complex depending on the learning algorithm. The model then makes recommendations on the test data. The model can be evaluated by testing its accuracy on real-world data. There exist several algorithms; for such methods, some of them include singular value decomposition, clustering, and Bayesian networks [6].

2.4 Similarity Metrics

Similarity metrics are various methods used to assign similarity to a pair of users or items. The most commonly used traditional metrics are cosine, Pearson's coefficient, Spearman's correlation, Jaccard index, Euclidean, mean squared difference, and adjusted cosine. Appropriate choice of similarity metrics is crucial as the quality of recommendation is directly related to the similarity values generated [4].

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

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

Equations 1 and 2 illustrate the formula to calculate cosine similarity and Pearson's similarity, respectively. Here, I_{uv} refers to all books rated by both users u and v, r_{ui} refers to rating assigned to book i by user u, r_{vi} refers to rating assigned to book i by user v, μ_u and μ_v are the average ratings assigned by users' u and v, respectively on all books.

2.5 Evaluation of Recommender Systems

It is highly essential that the quality of recommendation is constantly tested and made sure that the recommendations are as accurate as possible. There are various metrics that measure the quality/accuracy of prediction. Some of them are root mean square error, mean absolute error, Z-score, coverage, and normalized mean average error [7].

$$MeanAverageError = \frac{1}{\left|\hat{R}\right|} \sum_{r_{ui} \in \hat{R}} \left|r_{ui} - \hat{r_{ui}}\right| \tag{3}$$

$$RootMeanSquareError = \sqrt{\frac{1}{\left|\hat{R}\right|} \sum_{r_{ui} \in \hat{R}} (r_{ui} - \hat{r_{ui}})^2} \tag{4}$$

Equations 3 and 4 illustrate the calculation of mean average error (MAE) and root mean square error (RMSE). Here, $\hat{R}$ refers to all the predictions in ratings and $\hat{r_{ui}}$ refers to prediction in rating by the recommender system on the rating given by user u to book i.

2.6 Implicit Feedback

Implicit feedback is the process of collecting data passively, instead of letting users give quantitative feedback in the form of a rating, score, etc. Implicit feedback may involve page views, user clicks, bookmarks, etc. A substantial amount of work has been carried in this domain. Hu et al. [8] explore the idea of combining implicit data along with explicit ratings to make better recommender systems. This paper proposes to use implicit feedback to associate it with positive or negative value with varying confidence levels. The paper proposes a model similar to the matrix factorization method to include implicit feedback.

Liu et al. [9] combine explicit feedback model and the implicit feedback model using a factorized neighborhood model. This is an extension of the traditional nearest item-based model in which the item–item similarity matrix is approximated via low-rank factorization.

Our idea is to use implicit feedback in an item–item based collaborative filtering model. We have implemented, analyzed, compared various methods and evaluated the performance of our method.

3 Implicit Rating in Book Recommendation System

According to our literature survey, the availability of recommender systems for books was minimal. In this paper, we have attempted to build a recommendation system for books using item-based collaborative filtering methods, using K-nearest neighbors algorithm to pick the recommendation. We also aim to improve the accuracy of these generic methods through our novel approaches. For this purpose, we have used the GoodBooks-10k dataset. It has data for 10,000 books and 53,424 users. It has ratings numbered 1-5, 1 implying not very interested, with 5 implying great interest in the book. It also has a separate set of data for users who have added books to their wish list. As stated before, just like the lack of book recommenders, there is a shortage of good dataset for books, but the GoodBooks-10k dataset seemed to be extremely apt.

3.1 Cleaning the Dataset

The GoodBooks-10k dataset was obtained from Kaggle. The dataset was for most of the part without any noise. There were a total of 980,000 ratings. The first step was to remove duplicate ratings. A pair of book ID and user ID with the same value was considered to be a duplicate. 4487 duplicate ratings were found. The second step involved removing all users who have rated lesser than four books. This was done because users with very few explicit ratings contribute very less to evaluation and add to the unnecessary size of data. 8364 such users were found and were removed from the dataset. The dataset was further analyzed to observe certain trends.

The distribution of ratings was plotted in Fig. 1. Most of the ratings ranged from 3 to 5, and very few ratings were found to be in the range 1-2. The number of ratings per user was also plotted in Fig. 2. The observation obtained from the dataset was that very few users rated more than ten books. This chart gives a good idea of users who are many raters and who are not, which might be useful while considering feedback.

The number of ratings per book was plotted in Fig. 3. From this data, we can conclude that most of the books in the dataset have around 18-20 ratings(rated by users) on average. Very few books have less than 1 or more than 30 ratings.

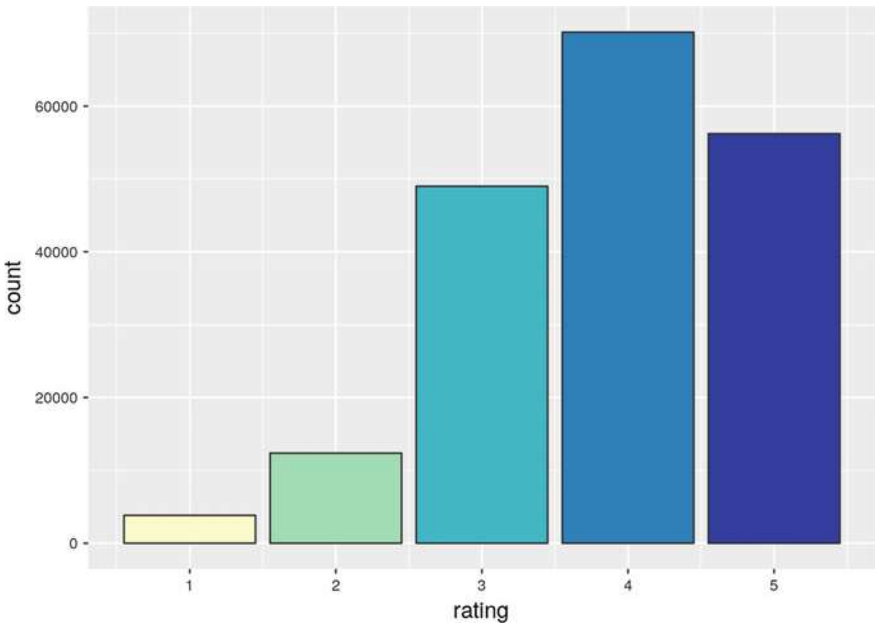

Fig. 1 Distribution of ratings

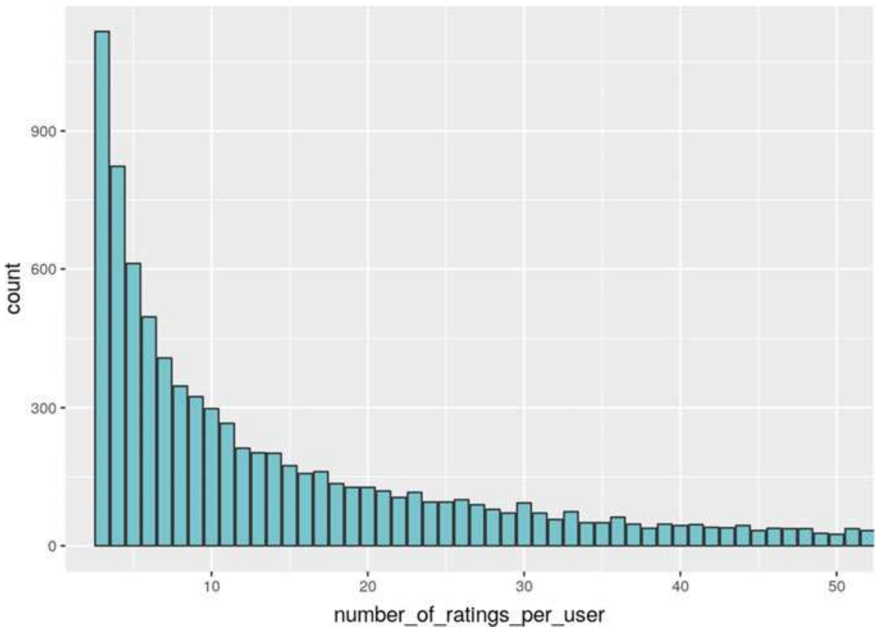

Fig. 2 Number of ratings per user

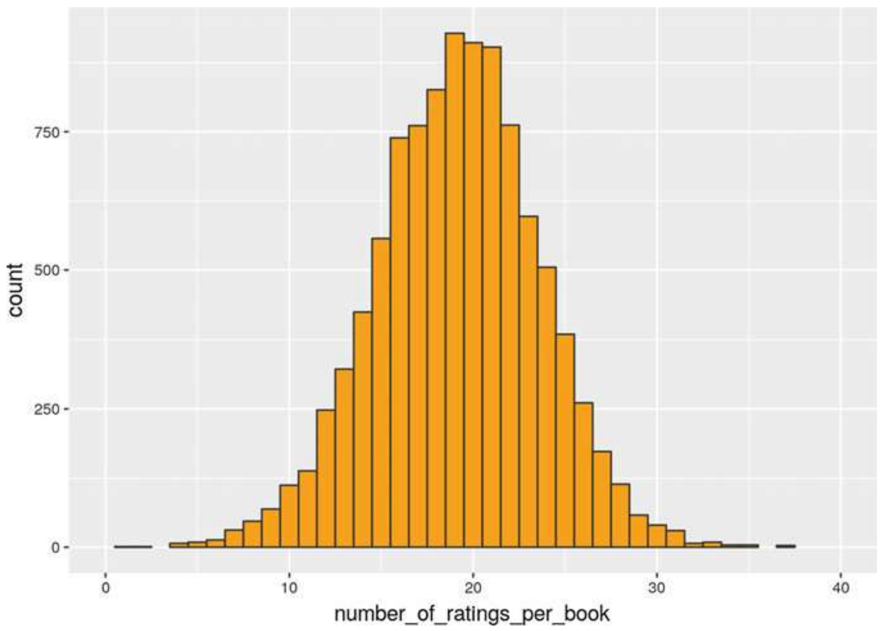

Fig. 3 Number of ratings per book

3.2 Implicit Rating in Book Recommendation System

Our main objective in this paper is to cause a meaningful improvement in the accuracy of predictions. In this paper, we present a few techniques to reduce the sparsity of the rating table to improve performance. Our approach was to assign a rating in the user rating table for every book added by the user into their to-read list. We have considered multiple approaches for this.

- Assign the book's average rating by all users' wherever a user has added a book to wish list.
- Assign average of that particular users book ratings, wherever a user has added a book to wish list.
- Assign a random rating between 1 and 5.
- Compare cosine similarity matrix. For a given user and book whose rating is to be guessed, take a weighted average of all the users book ratings, with weight as the cosine similarity of the book to be guessed and the other book the user has read. Only, consider books on average whose similarity value is more than 0.7 (as it amounts to a 45° of deviation in terms of angle).
- Consider the users deviation in ratings from the average trend. If the standard deviation is less than a threshold, assign the average rating of that book as the rating.

We constructed a book recommender using collaborative item–item filtering. For this purpose, we used an open-source recommendation system package for Python known as surprise [10]. It is currently in development by Nicolas Hug. Surprise has the standard collaborative filtering algorithms inbuilt. We took up the process of adding our novel approach to reduce the RMSE value of the pure ITEM-based filtering. The other frameworks we used were Pandas; for handling large datasets, SciKit learns for running the K-nearest neighbors algorithm and NumPy for its data structure capabilities. Our choice of programming language was Python. We chose these as it provided a convenient way to run and test our work effortlessly.

Our approach was inspired by the fact that the extrapolation of results would highly depend on the average ratings a user has given to a book and average book rating in general. We wished to combine these two factors along with the item–item similarities. The description of the method is as follows:

For a given user U and a book B, the user U has added to their wish list the following.

Find the weighted average of the ratings of the other books U has rated, considering only books in the average whose similarity with book B is greater than or equal to $1/\sqrt{2}$. We considered that value as $1/\sqrt{2}$ is equal to arcos(45) A 45 deviation of similarity seemed ideal to consider, anything about that is closer to being dissimilar than similar.

$$\hat{r_{ui}} = \frac{\sum_{v \in N_i^k(u)} sim_{u,v} \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u, v)} \tag{5}$$

where $\hat{r_{ui}}$ is the predicted rating that user U would give item I and sim(u, v) is the similarity of user u with user v and $r_{u,i}$ is the rating assigned to item i by user U.

But then, this method alone, even though takes into account a lot of factors for making the prediction, does not fill in a lot of values because if the user added a book out of his interest zone to the wish list, it will not assign a rating. Hence, we added a second part to this method. If there was no book with the similarity of more than, average book rating could have been used. But using average book rating directly may not be very accurate. So, to increase the strictness while picking, we calculated the standard deviation of the user book ratings with the average book rating. If the standard deviation was less than 1, we came to the conclusion that the users ratings are generally on par with overall average ratings. In that case, we found it appropriate to assign the average book rating directly.

This constitutes our method to increase the density of rating matrix, which at the same time, takes into account various factors and trends in user ratings and book similarities while making a meaningful and personalized choice to improve recommendation accuracy.

4 Experiment Conducted

Normal Item-based collaborative filtering: We ran the standard item-based collaborative filtering algorithm along with the k-nearest neighbors algorithm to give recommendations. We tabulated the RMSE value obtained. The similarity metric used in this case was cosine distance, and for prediction, we used the K-nearest neighbors approach.

4.1 Assigning Random Ratings

We assigned a random rating to the books which users have added to their to-read list. We then conducted the standard item-based collaborative filtering algorithm on the new dataset which includes the books implicitly rated by the user and the random rating assigned to them.

This method could significantly increase the density of the rating matrix. Hence, we tried this method. The disadvantage of this method is that it assigns values with absolutely no correlation to any of either user or book similarities. However, it is a good way to test how the density of the rating matrix affects recommendation and also checks the extent to which alteration in the RMSE value occurs, which will provide a baseline for comparing RMSE values.

4.2 Assigning Average Book Rating

For every book users have added to their read list, we added the books average rating in the rating table. This is a better indicator than a random rating, but then it really does not take into account the real interest of the user. This could be thought of like a popularity-based metric where a user is assigned a rating which is globally calculated considering all other users. It still picks a value based on a well-known parameter, which represents the general interest of the users. The RMSE value was tabulated.

4.3 Assigning Average of User's Book Rating

For every book users have added to their to-read list, we added the average of the users book ratings to the rating table. For example, if a user rated three books and gave them a rating of 3, 4, and 5, we assigned a value (3 + 4 + 5)/3 = 4. This method considers the users general ratings, but does not take into account the affinity of the given book and the user. It represents the users interests better, but fails if the user selects a book, not in their interest zone. The RMSE value was tabulated.

4.4 Assigning Values Based on Our Approach

Finally, we used the approach we formulated to assign ratings. This method takes into account all intricacies of user interest, trends in user ratings, and general opinions. The drawback this method faces is that it does not assign a rating for every book in the wish list, as it cares for the multiple other parameters and does not take a wild guess at a rating. The RMSE value was tabulated.

Algorithm 1 Assign implicit rating to book

```
procedure ASSIGNRATING($U$ , $B$)                                    ▷ User U and book B
    for book $A$ in user $U$'s rated books do
        if similarity($A$ , $B$) > 0.7 then
                $score$ += similarity($A$ , $B$)*Rating($A$)
                $count1$ += similarity($A$ , $B$)
        end if
        $distance = (AverageRating(A) - UserRating(A))^2$
        $count2+ = 1$
    end for
    if $count1 > 0$ then
                $Rating(B) = score/count1$
    end if
          else
    if $distance/count2 <= 1.0$ then
                $Rating(B) = AverageRating(B)$
    end if
end procedure
```

5 Observations and Analysis

We plot the RMSE values and we got with each of the results.

Each of the methods discussed above was tested three times to make sure a good value of RMSE was obtained. This was also done to eliminate any noise or randomness present in data (Fig. 4).

Even though the number of predictions made in our approach was just 247,038 entries, it still managed to fare very well. There was a great decrease in RMSE values while using book average and the users' book average. However, the number of predictions made in these methods was 912,705. Our understanding from this experiment was that the density of ratings in the matrix might have been a more dominant factor in reducing the RMSE values, over the reliability of the method. The random assignment fared very poorly, making it an undesirable action, by drastically increasing the RMSE values. It does show that increasing the density of the matrix may lead to such a large change in RMSE values, but in this case, it is for the worse.

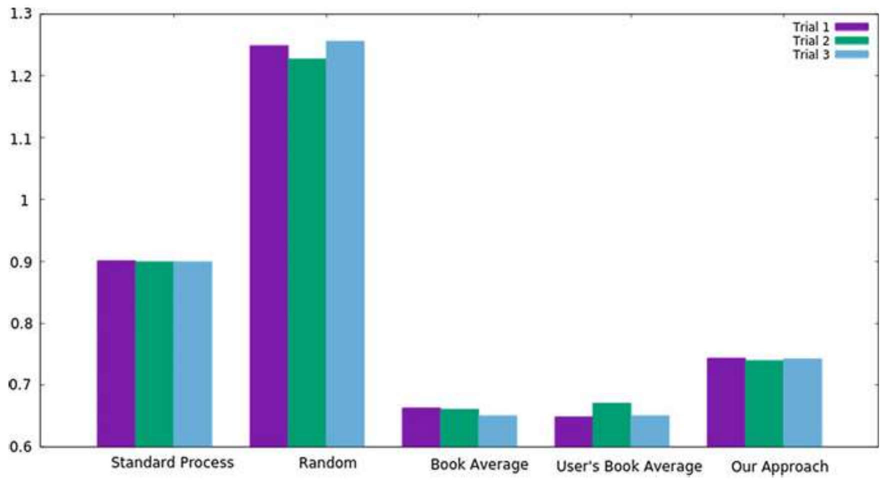

Fig. 4 Various algorithms and their RMSE values

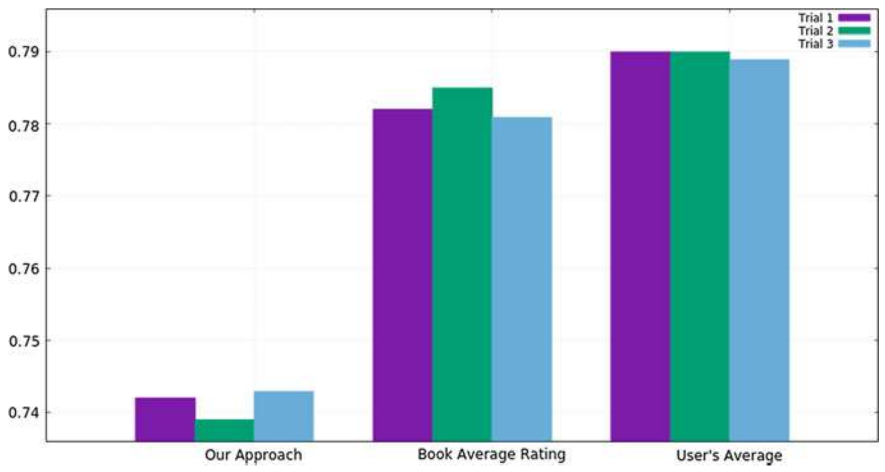

Fig. 5 Algorithms and their RMSE values

Our algorithm fares well compared given the fact that it assigned ratings only for about 30% of the total dataset.

For a better understanding and a more fair comparison of our algorithm with the book average and users rating average, we conducted the same method for a subset of 300,000 new entries in the case of book average and users rating average.

We chose to compare with 300,000 new entries as it is close to but still higher than 247,038 entries. The results are plotted in Fig. 5.

To compare the efficiency of our algorithm, we made the average user rating and average book rating method make predictions for only 300,000 entries and compared it with the results of our prediction algorithm. Our method fared much better in this case.

Thus, it implies that our assumption that the density of the matrix was more dominant compared to the efficiency of the assignment of values was correct.

6 Conclusion and Future Works

In this paper, we demonstrated a novel and innovative approach to make use of implicit feedback for a book recommendation system. In conclusion, the inclusion of implicit ratings proved to be a good improvement over the normal method of considering only explicit ratings. We also plan to improve upon the same method in future works by including other kinds of implicit data which will give a complete profile of the user. As the usage of recommendation systems becomes increasingly prevalent, it is critical to building a recommendation engine to improve the user experience. At the same time, there is also a thin line between collecting implicit feedback and respecting the privacy of the user. Implicit feedback collected must not exploit the information provided by the user in any manner.

In the future, we aim to improve upon the work discussed in this paper. Some possible extensions could be to use a user–user similarity metric. In our method, we only used item–item similarity. We could also try the same method on a different dataset, which gives a wider variety of implicit feedback. The adjustment will have to be made as multiple implicit feedback is present.

References

1. Gomez-Uribe, C.A., Hunt, N.: The Netflix recommender system: algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. **6**(4), Article 13, 19 p. (2015). https://doi.org/10.1145/2843948
2. Davidson, J., Livingston, B., Sampath, D., Liebald, B., Liu, J., Nandy, P., Vleet, T.V., Gargi, U., Gupta, S., He, Y., Lambert, M.: The YouTube video recommendation system, Proceedings of the fourth ACM conference on Recommender systems - RecSys 10 (2010)
3. Jacobson, K., Murali, V., Newett, E., Whitman, B., Yon, R.: Music personalization at Spotify. In: Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16). ACM, New York, NY, USA, pp. 373–373 (2016). https://doi.org/10.1145/2959100.2959120
4. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms, Proceedings of the tenth international conference on World Wide Web - WWW 01 (2001)
5. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 7680 (2003)
6. Bobadilla, J., Ortega, F., Hernando, A., Gutirrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109132 (2013)
7. Karypis, G.: Evaluation of Item-Based Top-N Recommendation Algorithms. In Proceedings of the tenth international conference on Information and knowledge management (CIKM '01). In: Paques, H., Liu, L., Grossman, D.: (eds.) ACM, New York, NY, USA, pp. 247–254 (2001)

8. Hu, Y., Koren, Y., Volinsky, C.: Collaborative Filtering for Implicit Feedback Datasets. In: 2008 Eighth IEEE International Conference on Data Mining, Pisa, pp. 263–272 (2008). https://doi.org/10.1109/ICDM.2008.22
9. Liu, N.N., Xiang, E.W., Zhao, M., Yang, Q.: Unifying explicit and implicit feedback for collaborative filtering. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10). ACM, New York, NY, USA, pp. 1445–1448 (2010). https://doi.org/10.1145/1871437.1871643
10. https://github.com/NicolasHug/Surprise
11. Choi, K., Yoo, D., Kim, G., Suh, Y.: A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis. Electron. Commer. Res. Appl. **11**(4), 309317 (2012)
12. Jain, A., Vishwakarma, S.K.: Collaborative filtering for movie recommendation using RapidMiner. Int. J. Comput. Appl., **169**(6), 2933 (2017)
13. Lee, S.K., Cho, Y.H., Kim, S.H.: Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. Inf. Sci. **180**(11), 21422155 (2010)
14. Gomez-Uribe, C.A., Hunt, N.: The Netflix recommender system: algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. **6**(4), Article 13, 19 p. (2015). https://doi.org/10.1145/2843948
15. Casino, F., Patsakis, C., Puig, D., Solanas, A.: On privacy preserving collaborative filtering: current trends, open problems, and new issues. In: 2013 IEEE 10th International Conference on e-Business Engineering, Coventry, pp. 244–249 (2013). https://doi.org/10.1109/ICEBE.2013.37
16. Kim, Heung-Nam, Ji, Ae-Ttie, Ha, Inay, Jo, Geun-Sik: Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electron. Commer. Res. Appl. **9**(1), 73–83 (2010)
17. Herlocker, J.L., Konstan, J.A.: Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information (1999)
18. Casino, F., Patsakis, C., Puig, D., Solanas, A.: On privacy preserving collaborative filtering: current trends, open problems, and new issues. In: 2013 IEEE 10th International Conference on e-Business Engineering, Coventry, pp. 244–249 (2013)
19. Balabanovi, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72. (1997). https://doi.org/10.1145/245108.245124