K Nearest Neighbour Collaborative Filtering for Expertise Recommendation Systems

Kazi Omar Faruk, Anika Rahman, Sanjida Ali Shusmita, Md Sifat Ibn Awlad, Prasenjit Das, Md Humaion Kabir Mehedi, Shadab Iqbal, and Annajiat Alim Rasel

BRAC University, 66 Mohakhali, Dhaka - 1212
`{kazi.omar.faruk, anika.rahman, sanjida.ali.shusmita, md.sifat.ibn.awlad, prasenjit.das, humaion.kabir.mehedi, shadab.iqbal}@g.bracu.ac.bd`
`annajiat@gmail.com`
`https://www.bracu.ac.bd/`

Abstract. With digitization the number of internet users are increasing and a wide variety of businesses based on the internet have emerged and modern ecommerce is one of them. In modern ecommerce there are a large number of products and a huge number of users accessing those to fulfill their needs. Searching for products is a time consuming task and a user may not have knowledge about every category of products listed. The recommendation system is important in the digital space and e-commerce because it suggests content to users based on their preferences. Modern recommendation systems based on machine learning models can more precisely recommend items to a user which they actually like. Collaborative filtering is one of them. It works by recommending items to a user by finding similar users preferences based on the predicting ratings of unknown items. In this work we have applied the k nearest neighbor(KNN) collaborative filtering algorithm on the Amazon Kindle Store dataset which is the largest online e-book store on the internet. We have also proposed a modified version of this algorithm by considering expert users on the system to generate more precious recommendations for a user. We evaluated our models using RMSE, MAE, hit rate and coverage and achieved outstanding results compared to the baseline algorithm.

Keywords: K Nearest Neighbour · Collaborative Filtering · Cosine Similarity · Pearson Correlation · Hit Rate · Coverage.

1 Introduction

A recommendation system aims to suggest items to the users of a system according to their personalized interests. In recent years, with the rise of the user interactive web services such as YouTube, Netflix, Amazon, Spotify, etc. which suggests users their preferred playlist of songs, movies, clothes, and other various contents, the use of recommendation systems has increased. For example, when a user watches a movie on Netflix, user get recommended on what to watch next based on their previous choice which keeps the user engaged in the system increasing their e-commerce business. Collaborative filtering is a state of the art algorithm for recommendation systems. This technique is used widely in recommendation systems. This algorithm works on an assumption that people with similar choices tend to like similar things and people like things that are of similar taste. It is a technique to predict the interests of a user in accordance with the interests of other users who happen to share the same interests. For example, In a movie recommendation system if users A and B both like the movies - "Iron Man" and "The Hobbit" then the system will assume that they both share the same taste. So if user B likes "Harry Potter", the system will recommend "Harry Potter" to user A. Collaborative systems filter identical interests of users from their rating history. The filtering is driven by the idea that we are more comfortable in considering the suggestions of the people with whom we share common interests. A key factor in collaborative filtering is the capacity of obtaining the preferences of the user. It basically depends on the historical data of a user. In this paper, we have applied k-nearest neighbor based collaborative filtering algorithm on amazon kindle store book review dataset [2] to predict the rating of an unknown by a user and also recommend a few items to a user based on highest predicted ratings. Also we have applied a small modification to this algorithm to integrate user expertise in this recommendation.

2 Literature review

Most of the recent work on recommendation systems depends on collaborative filtering [3] [4]. But over time the main focus is on the integration of Deep learning and CF to improve the predictions of the recommendation systems. For instance - DeepMF [5] is a technique to use deep learning for predicting the errors in the matrix factorization and get precise results. Moreover, RNCF [6] proposes a technique which uses three levels of abstraction to improve CF results. Another deep learning approach with CF is used in Autoencoder based CF [7] where the variational autoencoders are used to decompose partial ratings. Expertise recommendation systems are proposed by [8], [9], [10] and [11]. [8] shows a variation CF with the use of explicit expert dataset. On the other hand [9] proposes a content-based CF architecture where experts are found using similarity between keywords and full text index.In the same context to find expert researchers [10] demonstrates an approach which evaluates experts on these - topic relevance, expert quality and researcher connectivity dimensions. Besides, in the literature, [12], [13], [14] have implemented different approaches like domain specific approach, factor based approach, weight based etc. to reduce the dimensionality of the CF algorithm and achieved improvement.

3 Methodology

3.1 Cosine and pearson similarity

We have used a memory-based collaborative filtering algorithm which is based on the user-item k nearest neighbor algorithm. We will use it to predict the ratings of an item by a user based on the ratings of similar users to the user who rated this item before. By using those predictions a list of top K items can be selected which are predicted to be rated higher by a user and are recommended to that user. The algorithm can be described in terms of the equation given below,

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u,v).r_{vi}}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{1}$$

In this equation $\hat{r}_{ui}$ is the predicted rating of user u for an item i and *sim(u,v)* is the similarity score between users u and v. The similarity score between users can be calculated by a few methods. We have used cosine similarity and pearson correlation to calculate the user-to-user similarity.

Cosine similarity is a metric that mathematically measures the similarity between vectors irrespective of their size. It basically quantifies the similarity of the vectors. Mathematically, cosine similarity is the **"cosine"** angle between two vectors. It is the division of the **inner product of two non-zero vectors** by the **production of each vector's magnitude**. For example, if A and B are two vectors and θ is the angle between them then the cosine similarity is -

$$\cos\theta = \frac{A.B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

The value of cosine similarity is bound by a range of **0 to 1** which is same as the cosine angle range. If the angle between the vectors is **90 degrees** then the cosine value is **0**. That means the vectors are perpendicular to each other and have no similarity as they are far from each other. On the other hand, if the angle between the vectors is **10 degrees** then the vectors are closer to each other which means they have many similarities between them. Cosine similarity has multiple uses in various sectors. The recommendation system uses this similarity measure to find out **how similar users, items of the system** are to recommend them correctly. In our work, we have used cosine similarity to measure the similarity between the users.

To compute the cosine similarity between all pairs of users, only users having items in common are taken into consideration here.

$$CosSim(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui}.r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2}\sqrt{\sum_{i \in I_{uv}} r_{vi}^2}} \tag{3}$$

Here, r_{ui} is the ratings of user u for an item i and r_{vi} is the rating of user v for that same item i. According to cosine similarity this simply means how user u

and v rated the common items between them and also how close those ratings are. The similarity is the angle between those ratings. The closer the ratings are, the higher the similarity between users will be.

Pearson's correlation coefficient is calculated by dividing the covariance of the two variables by the product of their standard deviations. It quantifies the degree of similarity between two vectors. It is primarily used in linear regression to assess similarity in linear space. It works by drawing the best fit line between the variables and measuring the distance between the data points and the line.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - x)(y_i - y)}{\sqrt{\sum_{i=1}^{n}(x_i - x)^2}\sqrt{\sum_{i=1}^{n}(y_i - y)^2}} \tag{4}$$

Here,
n is sample size
x_i, y_i are the individual sample points indexed with i
$x = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for y

The value of the Pearson correlation is bound by the range of **-1 to +1**. If the **value is 0** then it means that there is **no correlation** between the two variables. A value **greater than 0** is depicted as a **positive correlation** and **less than 0** is depicted as a **negative correlation**. Despite the positive or negative term, the further the value is from zero the stronger the correlation is. The closer the value is to zero the variation is higher.

The Pearson correlation is a popular similarity measure for recommender systems where it is used to measure how users are similar to each other. It is used in collaborative filtering is based on the idea that **different users** might have **different baselines** of their ratings. What user A considers a three-star rating for an item may be different from what user B considers a three-star rating for that item. It tries to normalize these differences. Instead of measuring similarities between people based on their raw rating values, this approach measures similarities based on the **difference** between a user's rating for an item and their average rating for all items.

$$P(u,v) = \frac{\sum_{i \in I_{uv}}(r_{ui} - u_u)(r_{vi} - u_v)}{\sqrt{\sum_{i \in I_{uv}}(r_{ui} - u_u)^2}\sqrt{\sum_{i \in I_{uv}}(r_{vi} - u_v)^2}} \tag{5}$$

In this equation $P(u,v)$ is the pearson similarity between user u and user v, r_{ui} is the rating of an item i by user u and r_{vi} is the rating of that item i by user v, u_u is the average rating of user u and u_v is the average rating of user v.

After calculating the similarity score between a user and other users, then the algorithm will select the top N users which are similar based on the similarity score calculated using cosine and pearson similarity.

3.2 Selecting top-N users and modified collaborative filtering algorithm

Top-N similar users are selected according to the similarity score of the users in the basic CF algorithm. But we have **modified** this part in our modified collaborative filtering algorithm to collaborate with the expert users to get the improved recommendations. It is based on an **intuition** that the users who are not only **highly similar** but also have a high **expertise** can recommend more precise items to others.

For selecting the most similar expert user, first of all we have selected N most similar users based on cosine and pearson similarity score and then we have selected M users based on the maximum expertise score which is nothing but the ratio between helpful votes of a user and the number of reviews of that users.

$$\textit{expertise score} = \frac{\textit{helpful votes of a user}}{\textit{total no of reviews of that user}} \tag{6}$$

3.3 Calculating Predicted Ratings

It is a simple method that takes the Top-N similar users, multiplies the similarity with their ratings, and averages this value with similarity score for the **not-yet-rated** item for the active user.

$$pred(u, i) = \frac{\textit{Sum}(X \ \times \ Y)}{\textit{Sum}(X)} \tag{7}$$

Here,
pred(u, i) = Predicted Rating of item i for user u
X = Neighbourhood similarity score
Y = Neighbour user rating

3.4 Generating top K recommendations

After predicting the rating now comes the part of the main output where the system needs to generate the recommendations for the active user. From the predicted ratings of all the un-rated items of the system, the system **sorts the items** according to their ratings. Then selects top K items from that sorted list to recommend to the users which can be liked by the users. We generate top K recommendations for each user based on the predicted ratings and evaluate the recommendations using hit rate and coverage.

4 Experiment

4.1 Work Process

In our work process, first of all we collect the dataset and then we preprocess the dataset and select necessary attributes. Then we have applied KNN collaborative filtering using cosine and pearson similarity to calculate the similarity score between the users to select N most similar users for each user and from there

M most expert users are being selected based on their expertise score in the modified algorithm. Finally, we generate top K recommendations for each user based on the maximum predicted ratings and evaluate the model using RMSE, MAE, hit rate and coverage. The values of M, N and K is 10, 20 and 100. Figure 1 shows the diagram of the workflow.

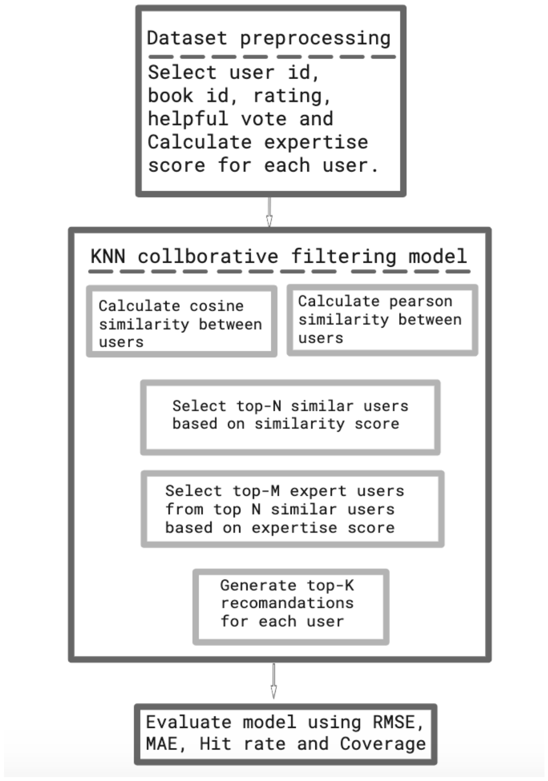
Dataset preprocessing
Select user id, book id, rating, helpful vote and Calculate expertise score for each user.
KNN collborative filtering model
Calculate cosine similarity between users
Calculate pearson similarity between users
Select top-N similar users based on similarity score
Select top-M expert users from top N similar users based on expertise score
Generate top-K recomandations for each user
Evaluate model using RMSE, MAE, Hit rate and Coverage

Fig. 1. Diagram of the workflow

4.2 Dataset

For our experiment, we have used the dataset of Amazon Kindle Store book review dataset [1]. This dataset consists of 5,722,988 reviews from 2,409,262 users on 493,849 books. The dataset was collected from May 1996 to July 2014. This dataset consists of reviews which are user id, book id, review text, ratings and helpful votes and the metadata information of a book. The structure of the dataset is shown in the table 1.

Table 1. Dataset structure

attribute	**value**
overall(rating)	4.0
verified	true
reviewTime	12 29, 2012
reviewerID	A27UD5HYAKBL97
asin(book id)	1423600150
vote(helpful vote)	3
style	Format: Hardcover
reviewerName	Cheryl
reviewText	If you like making salsas this is a great book with different ideas for party. I gave it as a gift
Summary	Great Book
unixReviewTime	1356739200

We preprocess the dataset in such a way that each user has at least 50 reviews and pick attributes which are necessary for our experiment. We have 8,309 users after considering the users having at least 50 reviews. The dataset after preprocess is shown in the table 2.

Table 2. Dataset after preprocess

attribute	**value**
overall(rating)	4.0
reviewerID	A27UD5HYAKBL97
asin(book id)	1423600150
vote(helpful vote)	3

After preprocessing the dataset we have calculated the expertise score by taking the ratio between the number of helpful votes and the total no of reviews of that user. We have used the expertise score to select M expert similar users who have the highest expertise score for rating prediction and recommendation. Further, we keep 20% data separately to test our model.

4.3 Model Evaluation

Finally we evaluated both the baseline KNN collaborative filtering and modified version with the Amazon kindle store book review dataset using RMSE, MAE, hit rate, coverage and compared the results. For RMSE and MAE we compare the true rating with the predicted rating and then we generate top-K recommendations for each user based on the predicted rating to calculate hit rate and coverage.

Root Mean Square Error (RMSE) is the mean squared difference between the values of predicted ratings and the actual ratings. The lower value of RMSE is better which means that predictions are close to actual and error is less.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}} \tag{8}$$

Mean absolute error (MAE) is another method we have used to calculate the error of the predicted rating. It computes the average of all the absolute differences between the predicted rating and real ratings. The lower value of MAE is better.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{9}$$

In these equations, y_i is the predicted rating for item i, x_i is the actual rating for item i and n is the total number of entries.

Another widely used evaluation method for recommendation systems is Hit Rate (HR). Here we generate top K recommendations for all of the users in the test set and check if one of the recommendations in a user's top K recommendations is something they actually rated if then we consider that a hit. To measure Hit Rate we have used **leave one out** cross validation method. In this method one item from each user's top K recommendations is removed from the training dataset and uses that dataset to evaluate the model and then checks if the recommended top K list contains that left out item from the training set. If yes, then it is considered as a hit. A higher hit rate indicates that the predictions are better to be recommended to users.

$$\mathit{Hit\ Rate} = \frac{\mathit{Total\ Number\ of\ Hits}}{\mathit{Total\ Number\ of\ Users}} \tag{10}$$

Coverage is another commonly used method to evaluate a recommendation system. It is the **percentage** of possible recommendations that the system is able to provide. In other terms, it can also be defined by what percentage of users have **at least one** good recommendation. Coverage also shows whether the system can recommend everything from the system catalog or it can recommend the same item to all of the users. For coverage we set a threshold rating of 4.0 that means we will consider a good recommendation if a user has at least one recommendation having a rating of 4.0.

5 Result analysis

As we have discussed earlier, in our experiment we have used RMSE, MAE, hit rate and coverage to evaluate our model and compare the result between the modified and the baseline KNN collaborative filtering algorithm on the Amazon Kindle Book Review Dataset.

Evaluation metrics comparison between the baseline and the modified KNN collaborative filtering algorithm is shown in the table 3.

Table 3. Result comparison between baseline and modified algorithm

	Metrics	Cosine	Pearson
Baseline KNN CF	RMSE	0.85257	0.46058
	MAE	0.44077	0.16078
	Hit Rate	0.00673	0.01059
	Coverage	0.99975	0.95510
Modified Expertise KNN CF	RMSE	0.96750	0.39032
	MAE	0.51136	0.13074
	Hit Rate	0.00818	0.01275
	Coverage	0.99614	0.95101

From the table 3 we can see that the for cosine similarity the baseline KNN collaborative filtering algorithm was working fine but the hit rate of the modified expertise version was better than the baseline algorithm this is because the recommendation was generated from the expert users instead of only considering similar users. For pearson similarity the modified algorithm performed better than the baseline algorithm in terms of RMSE, MAE and hit rate.

6 Conclusion

From our experiment we can see that our modified KNN collaborative filtering were performing better when we consider pearson correlation for user to user similarity calculation also cosine correlation was performing average compared to the baseline algorithm. So we can say that, instead of generating recommendation only based on similarity score between users, if we can find the expert users having more contribution to the system and generate recommendation based on their preferences then the recommendation will become more precious and accurate. Although there are many advantages, there are few disadvantages. One of them is the cold start problem where the system is not able to recommend items using this algorithm if the user has no contribution in the system and another one is popularity bias which is the system is biased towards recommending more popular items.

References

1. Amazon Kindle Store e-book review dataset, Justifying recommendations using distantly-labeled reviews and fined-grained aspects Jianmo Ni, Jiacheng Li, Julian McAuley Empirical Methods in Natural Language Processing (EMNLP), 2019. http://deepyeti.ucsd.edu/jianmo/amazon/index.html
2. 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.
3. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175–186, 1994.

4. R. Lara-Cabrera, Á. González-Prieto, and F. Ortega, "Deep matrix factorization approach for collaborative filtering recommender systems," Applied Sciences, vol. 10, no. 14, p. 4926, 2020.
5. J. Bobadilla, S. Alonso, and A. Hernando, "Deep learning architecture for collaborative filtering recommender systems," Applied Sciences, vol. 10, no. 7, p. 2441, 2020.
6. Y. Ouyang, W. Liu, W. Rong, and Z. Xiong, "Autoencoder-based collaborative filtering," in International Conference on Neural Information Processing, pp. 284–291, Springer, 2014.
7. X. Amatriain, N. Lathia, J. M. Pujol, H. Kwak, and N. Oliver, "The wisdom of the few: a collaborative filtering approach based on expert opinions from the web," in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, pp. 532–539, 2009.
8. J. Protasiewicz, W. Pedrycz, M. Kozłowski, S. Dadas, T. Stanisławek, A. Kopacz, and M. Gał¸e˙zewska, "A recommender system of reviewers and experts in reviewing problems," Knowledge-Based Systems, vol. 106, pp. 164–178, 2016.
9. J. Sun, J. Ma, X. Cheng, Z. Liu, and X. Cao, "Finding an expert: A model recommendation system," 2013.
10. Y. Chung, H.-W. Jung, J. Kim, and J.-H. Lee, "Personalized expert-based recommender system: training c-svm for personalized expert identification," in International Workshop on Machine Learning and Data Mining in Pattern Recognition, pp. 434–441, Springer, 2013.
11. F. Gao, C. Xing, and Y. Zhao, "An effective algorithm for dimensional reduction in collaborative filtering," in International Conference on Asian Digital Libraries, pp. 75–84, Springer, 2007.
12. Z. Zhang, D. Zhang, and Z. Guo, "Improving memory-based collaborative filtering using a factor-based approach," in In Proceedings of AAAI, vol. 7, 2007.
13. S. Ayyaz and U. Qamar, "Improving collaborative filtering by selecting an effective user neighborhood for recommender systems," in 2017 IEEE International Conference on Industrial Technology (ICIT), pp. 1244–1249, IEEE, 2017.
14. A. Bellogín, P. Castells, and I. Cantador, "Improving memory-based collaborative filtering by neighbor selection based on user preference overlap," in Proceedings of the 10th conference on open research areas in information retrieval, pp. 145–148, Citeseer,2013.
15. G. R. Lima, C. E. Mello, A. Lyra, and G. Zimbrao, "Applying landmarks to enhance memory-based collaborative filtering," Information Sciences, vol. 513, pp. 412–428, 2020.