Int. j. inf. tecnol. (June 2022) 14(4):1725–1731
https://doi.org/10.1007/s41870-022-00913-0

ORIGINAL RESEARCH

Co-clustering neighborhood—based collaborative filtering framework using formal concept analysis

Shipra Kataria[1] · **Usha Batra**[1]

Received: 18 November 2021 / Accepted: 10 March 2022 / Published online: 7 April 2022
© The Author(s), under exclusive licence to Bharati Vidyapeeth's Institute of Computer Applications and Management 2022

Abstract In recent times information on the web is growing exponentially and reaching to the point where humans can no longer deal with it manually or with conventional techniques. Recommender systems (RS) act as an effective tool in this situation and are being utilized in a variety of web-based applications and social media platforms. Collaborative filtering (CF) has been researched as one of the most extensively and successfully used techniques in RS. However, CF techniques encounter a significant number of challenges. One such challenge is data sparsity, which has a detrimental impact on the performance of a recommender system. In this paper, a novel CF technique is proposed for generating viable recommendations in sparse environment. The technique is based on co-clustering and optimized by formal concept analysis (FCA). The proposed research work is also focused on providing prospective recommendations ranking based on both local and global similarity index. This work is carried upon using benchmark book-crossing rating dataset. The proposed technique outperforms the existing sparse linear methods (SLIM) and item-based CF (IBCF) techniques in terms of parameters such as hit ratio (HR) and mean reciprocal hit value (MRHV). The results obtained from the research work finds its application in various recommender systems even in the presence of data sparsity.

Keywords Recommender system · Collaborative filtering · Co-clustering · Formal concept analysis

✉ Shipra Kataria
shipra.kataria22@gmail.com

Usha Batra
dr.ushabatra@gmail.com

[1] G D Goenka University, Gurugram, India

1 Introduction

In today's scenario, recommender systems, a tool for generating automatic and personalized information, have been widely employed in a number of applications including online gaming, mobile applications, advertising, e-commerce etc. Despite substantial research in this area over the last fifteen years, developing efficient algorithms for enhancing the recommender systems' accuracy still remains a hot topic in academia [1, 2]. An important application of recommender systems is ranking a list of items for users to see. A user is more likely to click on a recommended list and make a purchase if it is found to be relevant. As a result, companies will earn more revenue and has significant business implication. Hence, the research problem can be formulated as top-m recommendation problem, where performance is measured by calculating HR and MRHV in the top-m ranked list of items. Researchers have proposed a number of techniques to solve the above-mentioned problem, but they either suffer from scalability, and computational cost issues, or model tuning (in case of model-based approaches). Nearest-neighbor/memory-based CF techniques act as the most prominent and widely used solution for this problem [3, 4]. Nearest-neighbor CF techniques are further categorized into user-based CF (UBCF) and item-based CF (IBCF) techniques [5]. However, both the techniques rely on sparse rating datasets as an input to propose recommendations. Sparse dataset describes a circumstance in which the ratio of ratings to be projected to the previously acquired ratings is extremely high, i.e., predicting more than 90% of ratings based on less than 10% of existing ratings [6, 7]. As a result, providing top-m recommendations using nearest-neighbor CF techniques become a very challenging task. In order to overcome the data sparsity issue and provide

quality recommendations using nearest-neighbor CF techniques, a number of research has been carried out. Few researchers have employed clustering techniques while performing collaborative filtering. Gong [8] presented a strategy based on user and item clustering. In this strategy, the data smoothing process is utilized for clustering of users. Afterwards, item clustering is employed to generate the suggestions. Katarya and Verma [9] used a hybrid method based on Particle Swarm Optimization (PSO), K-means (KM) and Fuzzy C-means (FCM) to enhance the prediction accuracy with more personalized suggestions. Koohi and Kiani [10] proposed a UBCF technique based on FCM and evaluated its performance by comparing it with Self-organizing map (SOM) and KM clustering algorithms. Clustering is a process which handles only one dimension at a time either users or items. However, there are circumstances where a number of users are similar to each other with respect to only a subset of items or vice versa. To reveal this dualism between users and items, few research work in this direction is carried out using co-clustering techniques. Co-clustering is also named as two-mode clustering, biclustering and block clustering [11]. Co-clustering techniques reveal the dualism between users and items, by clustering them in both dimensions simultaneously [12]. According to the best of the authors' knowledge, Cheng and Church [13] were the first to demonstrate the utility of co-clustering technique on gene expression data. Furthermore, the technique was found to be effective in predicting recommendations with greater accuracy. Symeonidis et al. [14] proposed a novel nearest bicluster algorithm based on a new similarity measure and used it in combination with a co-clustering algorithm—Bimax to deal with constant values. In addition, the authors evaluated its performance against two well-known CF algorithms, such as UBCF and IBCF and found it with improved precision value by more than 30% and recall by more than 10%, respectively. Further, Symeonidis et al. [15] extended the work by experimenting the nearest bicluster algorithm with xMotif biclustering algorithm to deal with coherent values. The algorithm achieved better results in terms of effectiveness and efficiency as compared to the existing CF algorithms. Pablo et al. [16] described an immune-inspired algorithm, namely artificial immune network for Biclustering (BIC–aiNet) to provide several diverse and high quality biclusters. In further work, Pablo et al. [17] compared BIC-aiNet with Pearson Correlation, Probabilistic memory-based CF (PMCF), Bayes, Nearest—Biclusters and MultiLayer Perceptron (MLP), and found it performing better than other techniques in terms of root mean square error (RMSE), mean absolute error (MAE) and accuracy. Coelho et al. [18] proposed a multi-objective multipopulation immune inspired algorithm (MOM -aiNet) for generating diverse biclusters and improving the dataset coverage. De França et al. [19] compared MOM-aiNet to item-based k- Nearest Neighbor algorithm in terms of MAE and showed improvements. Alqadah et al. [1] proposed a bicluster neighborhood (BCN) framework for implicit feedback dataset. The authors employed several fundamental features from the field of formal concept analysis to generate more personalized user-specific biclusters. All the above—discussed co-clustering techniques cluster users and items simultaneously and recommend items. However, these techniques are not suitable for many real-life applications because of their static nature of obtaining initial set of co-clusters. The authors of [1] proposed a technique that is similar to the one proposed in this research work. But the major difference between the two techniques is that in the former technique, implicit feedback dataset is used for generating recommendations and clusters are computed offline. On the other hand, in the proposed technique, explicit rating dataset is used as an input, and clusters are generated on demand. Further, the recommendations ranking mechanism is based on both nearest co-cluster similarity and global similarity index values. Moreover, the literature also lacks the utilization of benchmark book-crossing rating dataset for experimental purpose.

Further, it is witnessed that the utilization of co-clustering algorithm with FCA is researched only once, and that too for implicit feedback dataset. In the current research work, an attempt has been made to work upon explicit rating dataset and generate recommendations in sparse environment by employing co-clustering techniques. In addition, FCA is used for co-clusters optimization. Experimental evaluation is performed by considering benchmark book-crossing rating dataset. The dataset is obtained from grouplens organization. It contains 278,858 users, providing 1,149,780 ratings on 271, 379 books. Rating is given on the scale of 1–10. In order to generate recommendations, firstly, discretization is applied to preprocess the dataset. After that, co-clusters are formed by applying suitable co-clustering algorithm. Further, to perform optimization on co-clusters, FCA conventional features are utilized to explore hierarchical relationship among co-clusters and result into more personalized user-specific co-clusters. Finally, the ranking score is calculated for items using nearest bicluster similarity and global similarity index values. The entire experimental evaluation was performed in python environment using sklearn library and inbuilt co-clustering package CoClust. The BIDEAL toolbox was also utilized for co-clusters visualization. Furthermore, a comparison of the proposed technique is done with SLIM and basic IBCF methods.

2 Preliminaries

This section covers a few key definitions that are required to comprehend the proposed method: co-clusters, neighboring co-clusters, minimum co-cluster, maximum co-cluster, smallest co-cluster and siblings.

2.1 Co-clusters

Let R (U, I) denotes a user-item rating matrix with rows representing users and columns representing items. If a user $u \in U$ has provided a rating value to an item $i \in I$, then $R(u, i) > 0$. Else, no rating value is denoted as 0 in matrix R i.e., $R(u, i) = 0$. In this paper, the work is done with explicit feedback dataset i.e., ratings provided by users on items. In addition, working with only constant values is considered. Further, it is assumed that only positive rating values were kept in the rating matrix by discarding all negative rating values from the data during pre-processing stage by employing a positive rating threshold value $P_t \geq 5$ and converted them to 1 by applying discretization. After that, Bimax co-clustering algorithm is employed for generating co-clusters with constant values. Now, the goal of the top-m recommendation problem is to create a ranked list of items for user u amongst those items with whom the user u has not yet interacted. Let Uc and Ic represent the users and items subsets respectively i.e., $Uc \in U$ and $Ic \in I$, then, a co-cluster can be defined as follows:

Definition 1 A co-cluster is a pair (Uc, Ic) such as $R(u, i) > 0$, for all $u \in Uc$, $i \in Ic$, and the submatrix R (Uc, Ic) is maximal. Maximal specifies that adding more user or item will result into a zero element in the matrix.

Here, it is important to note that u and i need not to be adjacent in original rating matrix to be a part of the same co-cluster. Further, a co-cluster denotes small subset of comparable users. Comparable users are those users who have similarity in terms of a smaller subset of items. Co-clusters thereby incorporate the notion of conditional or local proximity between users and items.

2.2 Neighboring co-clusters

Formal concept analysis specifies a comprehensive mathematical framework to organize co-clusters in a hierarchical manner. Within this hierarchical configuration, the set of co-clusters form a full lattice. And, using this lattice structure, determining minimum co-cluster, maximum co-cluster and neighbors of closely associated co-clusters is a well- defined task.

Definition 2 If (Uc1, Ic1) and (Uc2, Ic2) are two given co-clusters, then $(Uc1, Ic1) < (Uc2, Ic2)$ iff $Uc1 \subset Uc2$ and $Ic1 \supset Ic2$. Such ordering arrangement is described as hierarchical ordering. A co-cluster (Uc1, Ic1) is referred to as a minimum co-cluster if there is no other co-cluster (Uc2, Ic2) such that $(Uc1, Ic1) > (Uc2, Ic2)$. A co-cluster (Uc1, Ic1) is referred to as a maximum co-cluster if there is no other co-cluster (Uc2, Ic2) such that $(Uc1, Ic1) < (Uc2, Ic2)$.

Figure 1 shows the full lattice structure corresponding to the given dataset in Table 1. Note that in the obtained hierarchical structure in Fig. 1, the number of users and the number of items appear in the opposite order. A lower-order co-cluster has more items but fewer users. Further, the top and bottom neighbors for a co-cluster can be defined as:

Definition 3 A co-cluster (Uc1, Ic1) is a top neighbor of (Uc2, Ic2) if $(Uc1, Ic1) > (Uc2, Ic2)$, and there is no other co-cluster (Uc3, Ic3) satisfies $(Uc1, Ic1) > (Uc3, Ic3) > (Uc2, Ic2)$.

Definition 4 A co-cluster (Uc1, Ic1) is a bottom neighbor of (Uc2, Ic2) if $(Uc1, Ic1) < (Uc2, Ic2)$, and there is no other co-cluster (Uc3, Ic3) satisfies $(Uc1, Ic1) < (Uc3, Ic3) < (Uc2, Ic2)$.

It is important to note that a co-cluster's top and bottom neighbors are not unique. As seen in Fig. 1, there might be multiple top and bottom neighbors.

2.3 Smallest co-cluster and siblings

The primarily concern of the proposed work is to map the target user in the smallest co-cluster that contains that user. The smallest co-cluster can be defined as:

Definition 5 Given a user u such as $u \in Uc1$, a co-cluster (Uc1, Ic1) is defined as the smallest co-cluster for user u if there is no other co-cluster (Uc2, Ic2) such that $(Uc2, Ic2) < (Uc1, Ic1)$ and $u \in Uc2$.

The smallest co-cluster for a user u represents the co-cluster containing user u and has maximum count of items in it. Further, it's important to note that the smallest co-cluster of a user will always be unique and have fewest number of users. However, the smallest co-cluster might not always be a minimum cluster. As an example, it can be seen in Fig. 1 that {(U1, U9), (I3, I5)} is the smallest co-cluster for user U1, but it is not the minimum one.

Definition 6 A co-cluster (Uc1, Ic1) is considered a sibling of (Uc2, Ic2) if there exists a co-cluster (Uc3, Ic3) such that (Uc3, Ic3) is a top (or bottom) neighbor of both (Uc1, Ic1) and (Uc2, Ic2).

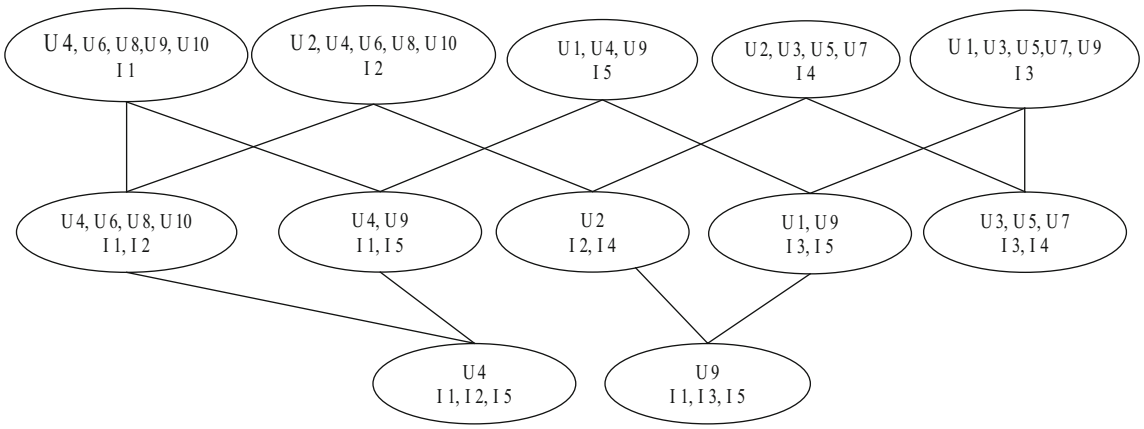

Fig. 1 A lattice/hierarchical structure of co-clusters, corresponding to the given dataset in Table 1

Table 1 A Synthetic running example: a rating dataset with 10 users and 5 items

	I1	I2	I3	I4	I5
U1	0	0	1	0	1
U2	0	1	0	1	0
U3	0	0	1	1	0
U4	1	1	0	0	1
U5	0	0	1	1	0
U6	1	1	0	0	0
U7	0	0	1	1	0
U8	1	1	0	0	0
U9	1	0	1	0	1
U10	1	1	0	0	0

Further, in the next section, a detailed description is given on examining co-clusters neighborhoods and generating viable candidate items for top-m recommendations.

3 Proposed algorithm

For collaborative filtering, the co-cluster neighborhood framework (CCN) involves three main stages:

- Given a user u, find the smallest co-cluster C = (Uc, Ic), where u ∈ Uc.
- Examine the co-cluster neighborhood of C to determine the potential set of candidate items for recommendation. The potential set of candidate items are the ones which are similar to the items belonging to Ic.
- Rank the obtained set of candidate items, by combining global and co-cluster neighborhood similarity.

These stages are demonstrated further by using a synthetically created dataset, as shown in Table 1 and the co-cluster lattice structure as shown in Fig. 1.

3.1 Finding smallest co-cluster for target user u

To find out the smallest co-cluster (Uc, Ic) containing the test user u, a three-step process is followed:

1) First of all, identify an itemset i i.e., i ∈ I that the target user u has interacted with.
2) Next, identify the user set u i.e., u ∈ U, who have conjointly interacted with any of the items i such as i ∈ I with which the test user u interacted.
3) Perform set intersection on the set of users obtained in step 2. And the resultant cluster would be the smallest co-cluster for the test user u.

As an example, in Fig. 1, it can be seen that the smallest co-cluster for test user U1 is {(U1, U9), (I3, I5)}.

3.2 Generating candidate items from nearest co-clusters

According to the Proposition 1 [1], the most similar items with items in Ic in terms of the user set Uc can be best obtained from the bottom co-cluster neighborhood of C, and it can be seen and understood well from lattice structure in Fig. 1. As a result, the items belonging to the bottom co-cluster neighborhood which are not a part of C are the best candidate items for top- m recommendation.

One significant point to note here is that there are some cases where the obtained smallest co-cluster is the minimum one and does not have any bottom co-clusters. In such cases, one additional step has to be performed by locating the bottom neighbors of the top neighbors of C or to find out its siblings. After that the items belonging to these neighborhoods which are not part of C, form the best candidate items for top-m recommendation.

For example, consider Fig. 1. The task is to make recommendations for the user U1, who belongs to the smallest co-cluster {(U1, U9), (I3, I5)}. However, this cluster is not

the minimum one and it has bottom neighborhood {(U9), (I1, I3, I5)}. So, the best recommendation for U1 is I1. On the other hand, in order to make recommendations for U3, the smallest co-clusterfoundis {(U3, U5, U7), (I3, I4)}. In this case, the smallest co-cluster is the minimum one and does not have any bottom neighborhood. In such cases, siblings need to be found. From Fig. 1, it can be identified that the siblings for this co-cluster are {(U1, U9), (I3, I5)} and {(U2), (I2, I4)}. From these siblings, the best recommendation items identified for user U3 are I2 and I5.

3.3 Ranking candidate items

As a final step, candidate items will be ranked using the combination of co-cluster neighborhood similarity and global similarity, as shown in Eq. (1).

For the co-cluster neighborhood similarity, the similarity of the smallest co-cluster (obtained in stage 1) to the neighboring co-clusters (obtained in stage 2) including candidate item should be considered. This will reflect the local or conditional similarity between local similar users and items. On the other hand, global similarity depicts the similarity calculated on complete data matrix.

Let C = (Uc, Ic) represents the smallest co-cluster and C′ = (Uc′, Ic′) represents the set of all co-clusters neighborhoods wherein the candidate items are chosen from. Candidate item ic′ ranking score in relation to user uc is calculated as:

$$r\left(uc, ic^{'}\right) = g\left(uc, ic^{'}\right) \times l\left(uc, ic^{'}\right) \tag{1}$$

where (uc, $ic^{!}$) represents the average global similarity of user uc and item ic′, and $l(uc, ic^{!})$ represents the local similarity (calculated using co-clusters neighborhood similarity) between user uc and item ic′. In further sections, these similarity calculations are discussed in detail.

3.3.1 Co-cluster neighborhood similarity

For computing co-cluster neighborhood similarity, a zero-induced similarity measure (discussed in [1]) is used, as shown in Eq. (2). Let C = (Uc, Ic) and C′ = (Uc′, Ic′) be two co-clusters. First of all, we consider the union of these two co-clusters as $K = (Uc\ U\ Uc^{!}, Ic\ U\ Ic^{!})$ then the similarity measure zero-induced is used as:

$$n\left(C, C^{'}\right) = 1 - \frac{zeros(K)}{|K|} \tag{2}$$

where $|K| = |Uc\ U\ Uc'| * |Ic\ U\ Ic'|$ and $zero(K)$ represents the total number of zeros in submatrix K. It is to note that fewer zeros will depict more similarity between C and C′. However, it is clear by definition that there must be at least one zero in K.

3.3.2 Local similarity

The final rank value using local similarity requires combining co-cluster similarity values of all the co-clusters in which Ic′ occurs to the smallest co-cluster. Three mathematical aggregating functions (shown in Eq. 3, 4 and 5) are proposed for this, such as sum, mean and maximum. These functions can be represented in mathematics as:

$$l\left(uc, ic^{'}\right) = \sum_{c^{'} \in X} n\left(C, C^{'}\right) \tag{3}$$

$$l\left(uc, ic^{'}\right) = \frac{1}{|X|} \sum_{c^{'} \in X} n\left(C, C^{'}\right) \tag{4}$$

$$l\left(uc, ic^{'}\right) = {}_{c^{'} \in X}{}^{max} n\left(C, C^{'}\right) \tag{5}$$

In the experimental results section, a discussion is given on how to choose the right aggregating function.

3.3.3 Global similarity

Here, one should note that co-cluster items are the only ones that we are interested in. Let (Uc, Ic) and (Uc′, Ic′) are two co-clusters, and item ic ∈ Ic and item ic′ ∈ Ic′. In this context, the global similarity of user uc and item ic′ can be calculated using Eq. (6):

$$g\left(uc, ic^{'}\right) = \frac{\sum_{ic \in Ic} J(ic, ic^{'})}{|Ic|} \tag{6}$$

where (ic, $ic^{!}$) represents Jaccard index, defined across all users who interacted with both items ic and ic′. Let Uci represents the user set who interact with ic and Uci′ represents the user set for ic′, then $J(ic, ic^{!})$ can be defined using Eq. (7):

$$J(ic, ic') = \frac{|Uci \cap Uci'|}{|Uci \cup Uci'|} \tag{7}$$

As an example, consider we have to provide recommendation for user U1. The smallest co-cluster for U1 is {(U1, U9), (I3, I5)}, which has a bottom neighbor {(U9), (I1, I3, I5)}. One potential recommendation for U1 is I1. Then, the ranking score for I1 can be calculated using Eq. (8) as:

$$\begin{aligned} r(U1, I1) &= g(U1, I1) \times {}^{\max}_{c' \in X}\, n(C, C') \\ &= \left(\frac{0.111 + 0.333}{2}\right) \times \left(1 - \frac{1}{6}\right) \\ &= 0.963. \end{aligned} \tag{8}$$

4 Co-clustering neighborhood (CCN) algorithm

Algorithm 1 depicts the above-discussed co-cluster neighborhood collaborative filtering framework:

Input: ut: test user for whom recommendation is to be performed
Input: R: complete data matrix
Input: n: required number of recommendations
Output: top – n items recommendation list
Begin
1. Map ut to the smallest co-cluster (Uc, Ic), such as ut ∈ C (Uc, Ic).
2. Determine smallest co-cluster's neighborhood by identifying bottom neighborhoods or siblings (in case of minimum co-cluster) :
   Cb <- Bottom_Neighbors (C) or
   Cs <- Siblings (C)
3. Generating candidate items:
   Initially, *candidate_items* = Φ
   - For Cb ∈ Cb do

   candidate_items <- *candidate_items* U Ib\Ic
   - For *Cs* ∈ Cs do

   candidate_items <- *candidate_items* U Is\Ic
4. Ranking candidate items
   - For i ∈ *candidate_items* do

   Compute r (ut, i): using combination of co-cluster neighborhood similarity and global similarity.
5. Recommend top – n items from *candidate_items* ranked by r (ut, i).

End

Algorithm 1: The CCN framework for collaborative filtering

5 Results

5.1 Dataset

Experimental evaluation is done using benchmark book-crossing dataset. It contains 278,858 users, with 1,149,780 ratings on 271, 379 books. Rating is given on the scale of 1 to 10. The rating information is organized as a series of comma-separated entries, one for each line, with user ID, ISBN number, and rating of the book.

5.2 Performance metrics

For performance evaluation, the Leave-One-Out cross validation (LOOCV) is used for examining the CCN framework. The dataset was divided into training set and test set by arbitrarily choosing any one of each user's non-zero entries from the original dataset and putting it to the test set. The proposed recommendation method is then tested on the training dataset, and top-m recommendation list is generated. Here, the value of m is set to 10.

To access quality of recommendation, two parameters HR and MRHV are considered. HR is defined as the ratio of total obtained hits (HT) in the test dataset to the total count of users (UT). So, if UT represents the total count of users and HT represents the total obtained hits in the test dataset, then HR is mathematically formulated, as shown in Eq. (9):

$$HR = \frac{H_T}{U_T} \tag{9}$$

MRHV refers to the weighted form of HR that favors hits appearing towards the top of the m-size recommendation list. If li represents the location of a hit for user ut, then, MRHV can be formulated mathematically as shown in Eq. (10):

$$MRHV = \frac{\sum_{i=1}^{H_T} \frac{1}{l_i}}{N} \tag{10}$$

The performance of the proposed CCN algorithm has been compared to two algorithms: SLIM, and basic IBCF, as discussed in the background work. To evaluate SLIM, the original paper's source code has been utilized, as specified by its authors. For IBCF, the mahout implementation was used. CCN implementation was done using CCN-Sum, CCN-Mean and CCN-Max. Further, the level for finding co-cluster neighborhood was set to 2.

Experimental results are shown in Table 2. It can be seen that the baseline IBCF algorithm is performing worst. And the reason for this worst performance is data sparsity. On the other side, the proposed CCN-Sum variant is giving best results in case of sparse datasets, and it is comparable to the SLIM algorithm in terms of HR value.

A detailed comparative analysis of proposed CCN framework variants with SLIM and IBCF can be seen in Fig. 2. Further, it is important to note that the proposed CCN-Sum variant works best only in case of sparse datasets. In case of dense dataset, number of co-cluster neighborhoods to the obtained smallest co-cluster would be very large and that will negatively impact the recommendation performance. It will bias towards the most frequently occurring items in neighbors and this will limit diversification in recommendations. In such cases of dense datasets, we expect CCN-Max to score best.

Table 2 Experimental results on book-crossing rating dataset

Dataset	Algorithm	HR (Top 10%)	MRHV
Book-crossing rating dataset	IBCF	3.55	1.25
	SLIM	6.70	1.05
	CCN-Sum	8.35	3.58
	CCN-Mean	8.05	3.26
	CCN-Max	8.02	3.25

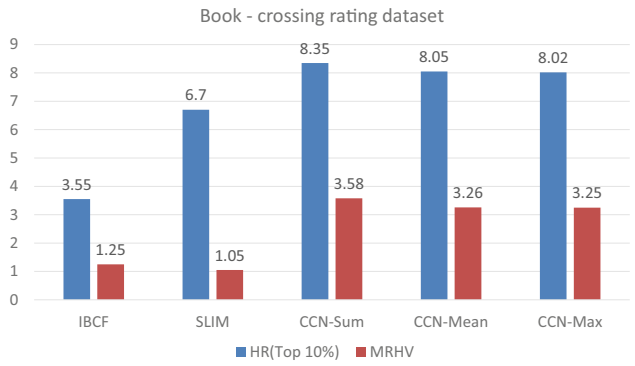

Fig. 2 A comparative analysis of proposed CCN framework variants with SLIM and IBCF

6 Conclusion

In this research work, a novel CF technique based on co-clustering neighborhood (CCN) framework is proposed. The technique has been tested on top-m recommendation task and is specifically built for explicit rating datasets. The ranking process utilized in the top-m recommendation task is focused on co-clustering neighborhood of the smallest co-cluster associated with the target user to obtain local or conditional similarity value and combined it with global similarity index value. The resultant co-clusters were optimized using formal concept analysis. In this research work, three variants of the proposed technique CCN-Sum, CCN-Mean and CCN-Max are compared with SLIM and IBCF techniques. CCN-Sum found to be outperforming all other techniques with HR value of 8.35 and MRHV value of 3.58. The proposed work can be utilized in a wide area of applications where the recommender systems are being used.

References

1. Alqadah F, Reddy CK, Hu J, Alqadah HF (2015) Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. Knowl Inf Syst 44(2):475–491
2. Yu HF, Hsieh CJ, Si S, Dhillon IS (2014) Parallel matrix factorization for recommender systems. Knowl Inf Syst 41(3):793–819
3. Bedi P, Agarwal SK (2011) Managing security in aspect oriented recommender system. In: 2011 International conference on communication systems and network technologies, IEEE, pp 709–713
4. Nazemian A, Gholami H, Taghiyareh F (2012) An improved model of trust-aware recommender systems using distrust metric. In: 2012 IEEE/ACM international conference on advances in social networks analysis and mining, IEEE, pp 1079–1084
5. Shi Y, Larson M, Hanjalic A (2014) Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput Surv 47(1):1–45
6. Kant S, Mahara T (2018) Nearest biclusters collaborative filtering framework with fusion. J Comput Sci 25:204–212
7. Frisch G, Leger JB, Grandvalet Y (2021) Co-clustering for fair recommendation. HAL Id: hal-03239856. https://hal.archives-ouvertes.fr/hal-03239856v1
8. Gong S (2010) A collaborative filtering recommendation algorithm based on user clustering and item clustering. J Softw 5(7):745–752
9. Katarya R, Verma OP (2016) A collaborative recommender system enhanced with particle swarm optimization technique. Multimedia Tools Appl 75(15):9225–9239
10. Koohi H, Kiani K (2016) User based collaborative filtering using fuzzy C-means. Measurement 91:134–139
11. Govaert G, Nadif M (2013) Co-clustering: models, algorithms and applications. Wiley
12. Kużelewska U (2020) Effect of dataset size on efficiency of collaborative filtering recommender systems with multi-clustering as a neighbourhood identification strategy. In: International conference on computational science, Springer, Cham, pp 342–354
13. Cheng Y, Church GM (2000) Biclustering of expression data. ISMB 8(2000):93–103
14. Symeonidis P, Nanopoulos A, Papadopoulos A, Manolopoulos Y (2006) Nearest-biclusters collaborative filtering with constant values. In: International workshop on knowledge discovery on the web, Springer, Berlin, Heidelberg, pp 36–55
15. Symeonidis P, Nanopoulos A, Papadopoulos AN, Manolopoulos Y (2008) Nearest-biclusters collaborative filtering based on constant and coherent values. Inf Retr 11(1):51–75
16. De Castro PA, de França FO, Ferreira HM, Von Zuben FJ (2007) Applying biclustering to perform collaborative filtering. In: Seventh international conference on intelligent systems design and applications (ISDA 2007), IEEE, pp 421–426
17. De Castro PA, de França FO, Ferreira HM, Von Zuben FJ (2007) Evaluating the performance of a biclustering algorithm applied to collaborative filtering-a comparative analysis. In: 7th International conference on hybrid intelligent systems (HIS 2007), IEEE, pp 65–70
18. Coelho GP, de França FO, Von Zuben FJ (2009) Multi-objective biclustering: When non-dominated solutions are not enough. J Math Model Algorithms 8(2):175–202
19. De França FO, Coelho GP, Von Zuben FJ (2009) Coherent recommendations using biclustering. In Proceedings of the XXX Congresso Ibero-Latino-Americano de Métodos Computacionais em Engenharia (CILAMCE), pp 1–15