INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Implementation of Nearest Co-cluster Collaborative Filtering using a Novel Similarity Index

Shipra Kataria[1]*, **Usha Batra**[2]

1 PhD Scholar, Department of Computer Science and Engineering, GD Goenka University, Gurugram, India
2 Professor, Department of Computer Science and Engineering, GD Goenka University, Gurugram, India

OPEN ACCESS

Received: 15-03-2022

Accepted: 25-04-2022

Published: 03-08-2023

Citation: Kataria S, Batra U (2023) Implementation of Nearest Co-cluster Collaborative Filtering using a Novel Similarity Index. Indian Journal of Science and Technology 16(29): 2204-2216. https://doi.org/10.17485/IJST/v16i29.597

* **Corresponding author**.

shipra.kataria22@gmail.com

Funding: None

Competing Interests: None

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: To implement Nearest Co-cluster Collaborative Filtering (NCCF) using a novel similarity index and evaluate its performance in sparse environments. **Methods:** An experimental and comparative evaluation of the proposed technique is performed using Python's built – in data analysis packages. As a preliminary step, preprocessing is performed on the explicit book – crossing rating dataset acquired from the GroupLens research lab database. After preprocessing, the nearest co-cluster algorithm based on a novel similarity index is applied to achieve partial matching of users' preferences. The proposed nearest co-cluster algorithm is experimented against two distinct types of co-clustering algorithms: Bimax for constant co-clustering (abbreviated as NCCF – B) and xMotif for coherent co-clustering (abbreviated as NCCF – X). In addition, a co-clustering package CoClust and the toolkit BIDEAL are utilized for co-clusters visualization. **Findings:** Extensive performance evaluation findings of the proposed technique are provided, and the technique has been found to be promising for generating pertinent recommendations despite data sparsity. The technique can significantly reveal the dualism between users and books through co-clustering and partial matching of users' preferences. Compared in terms of precision, recall, and F1 score, NCCF – B and NCCF - X outperformed Item – Based CF (IBCF), User – Based CF (UBCF) and Clustering – Based CF (CBCF). NCCF – B and NCCF – X outperformed with a precision difference of over 30% and a recall difference of 10%. When comparing two co-clustering algorithms, NCCF – X outperformed NCCF – B slightly but consistently. The differences in precision and recall are approximately 3% and 2%, respectively. However, the CBCF technique outperformed the other considered techniques in terms of execution time. **Novelty :** NCCF is robust in sparse environments due to its ability in revealing the dualism between users and books and in capturing the partial matching of users' preferences.

Keywords: Recommender System; Collaborative Filtering; Similarity Index; CoClustering; Nearest Neighbor

1 Introduction

Information Filtering (IF) has recently emerged as a crucial tool to combat the problem of "information overload"[1]. Most of the time, when we are looking for information on a topic (i.e., items, movies, books, etc.), we mostly rely on the opinion of individuals who are more knowledgeable about it. However, the profusion of opinions available on the Internet makes it infeasible to identify the most reliable ones. The strategy is to move away from individual opinions and toward collaborative ones[2]. Collaborative Filtering (CF) utilizes data mining (DM) and information retrieval (IR) technologies to generate recommendations on the basis of opinions of like-minded individuals. CF is an extensively used and successful strategy in a variety of recommender systems (RSs) and e-commerce applications. In the literature, there are two types of CF strategies that have been extensively researched. The first strategy is memory – based or nearest – neighbor based, which computes throughout the entire dataset and recommends items to the target user on the basis of similarities between users or items. Another strategy is model-based, which involves developing a model of specified types of users depending on their rating behaviors and then recommending items to the target user[3].

Among nearest – neighbor CF strategies, a well – established method in research is UBCF, which builds neighborhoods based on user similarity. For a test user, UBCF uses users' similarity to create a neighborhood of his closest ones. The most frequently occurring items in the generated neighborhood are recommended by UBCF to the test user. Another extensively used approach to nearest – neighbor CF is IBCF, which builds neighborhoods based on item similarity[4]. UBCF and IBCF both are one-sided techniques, as they only look at similarities between users or only between items. This characteristic makes them oblivious to the dualism between users and items. Furthermore, because the UBCF and IBCF similarity metrics access a composite collection of users and items, the techniques are unable to detect partial matches of preferences. In a real-time scenario, multiple users may have comparable preferences over a smaller group of items. For example, consider two users who have the same preferences for comedy movies, but have different preferences for other types of movies. In this situation, existing nearest-neighbor CF strategies will miss their partial match for comedy movies, which may help generate viable recommendations for the genre. This is because these strategies compare the similarity of two users across the entire set of items. As a result, their partial match is not captured because the variance in the left over items takes precedence against the subset of items where their preferences align. The same logic applies to IBCF strategies. Content-based filtering (CBF) can help capture partial matches between users and items. However, these strategies use complementary information on the content of items, for example, the features, whereas in this study, only a rating – based CF strategy is considered[5]. On the other hand, model-based strategies offer a high degree of scalability once the model is built. However, the strategies incur the cost of building and updating the model and may not cover as many user preferences as nearest-neighbor CF strategies do. This is because model-based strategies divide users into a restricted number of classes, resulting in limited users' preferences[6]. As a result, it is more challenging to get recommendations for a user with diverse preferences than it is to get recommendations for a user with a limited number of classifications. On the other hand, nearest-neighbor CF strategies will benefit users with diverse interests[7]. Among model-based strategies, clustering is arguably the most extensively utilised strategy. It identifies clusters of users or items with similar preferences. Conventional clustering methods such as k-means and hierarchical clustering place each user in a single cluster. However, when allocating users to a cluster, it is important to note that they usually have a wider range of preferences. As a result, such a user must be assigned to multiple clusters. Conventional clustering methods, which place each user/item in exactly one cluster, are unable to accomplish this. In order to include the user in multiple clusters, overlapping between clusters must be considered. Therefore, if a user has multiple item preferences, overlapping clusters can be used to include him in more clusters to encompass all of his preferences.

To provide solutions to the challenges discussed above, this research work proposes the simultaneous grouping of users and items, resulting in the discovery of co-clusters. Co-clusters refer to groups of users who have a highly associated rating on a group of items. Co-clusters allow to obtain the similarity between a test user and the co-cluster based solely on the items in the co-cluster. As a result, partial preference matching is taken into consideration. In addition, a user can be compared with several closely related co-clusters to obtain recommendations that encompass the entire range of his or her preferences. In the literature, several co-clustering algorithms have been developed, each with its own set of strengths and limitations depending on the situation. In this research work, two distinct ways of co-clustering are tested before selecting the one that produces the optimal results for CF. Co-clusters with constant values and co-clusters with coherent values are the two types that have been investigated. The first category seeks out a constant-value subset of rows and columns, while the second seeks out co-clusters with coherent values. The effectiveness and efficiency of both strategies have been examined in this research. The main contributions of this research work are summarized as follows:

- For the objectives of CF, two distinct types of co-clustering algorithms have been introduced (one with constant values and the other with coherent values). To give more precise suggestions, these algorithms group users and items simultaneously. Unlike the prior related work, which solely employed co-clustering to improve CF scalability, an entirely different strategy is used here to improve CF accuracy.
- A novel nearest co-cluster strategy is proposed, which is based on a new similarity metric that obtains partial matching of users' preferences.
- The impact of two distinct co-clustering strategies is examined.
- Extensive experimental evaluations using benchmark book – crossing rating dataset prove the efficiency and effectiveness of the proposed strategy over existing traditional strategies.

1.1 Related Work

There have been several attempts to use nearest – neighbor CF strategies for the top-M recommendation task. Sinha et al introduced a novel similarity measure, a modified Bhattacharya coefficient to compute user – user similarity in sparse environments and to use it as a weight in the IBCF. The authors showed an improvement in the performance of RS by experimenting the similarity measure on the MovieLens dataset[8]. Fethi Fkih compared thirteen different similarity measures commonly used in CFRS and observed that UBCF and IBCF do not rely on the same similarity measure to provide the best system performance. According to the authors' experimentation, Improved Triangle similarity augmented with user rating preferences (ITR) and Improved PCC weighted with RPB (IPWR) are the top choices for UBCF RSs and Adjusted Mutual Information (AMI) for IBCF RSs[9]. Rao Shen discussed a new approach based on long short – term memory and latent factor models (LSTM – LFM) to generate viable recommendations. The approach works in two key phases: (i) The first phase involves the prediction of user's interest for an item using an LSTM model and, (ii) the second phase involves extracting and characterizing the item's latent factor vector. The authors evaluated and compared the performance of the proposed approach using the MovieLens dataset and found it to outperform the UBCF, IBCF and LFM algorithms in terms of precision and recall[10]. Wang et al introduced a new similarity index using the Hellinger distance of item labels to provide viable recommendations in sparse environments. The effectiveness of the technique in alleviating data sparsity and enhancing the accuracy of RS was demonstrated in an experimental evaluation using two benchmark rating datasets[11]. Jiang et al introduced a hybrid recommender system (HRS) using the implicit feedback dataset as a basic input. The proposed model is based on in-depth emotion analysis and fusion of multiple sources of information. By using multi – source view fusion, the data distribution deviations, if any, can be alleviated in the training dataset, which is used for iterative training. The authors used the twitter dataset for experimental evaluation and proved the proposed model' effectiveness in terms of predictions and recommendations[2].

Some researchers have experimented CF in conjunction with clustering to alleviate data sparsity. Moradi et al proposed a model – based CF technique by employing a graph-based clustering strategy and trust information. In this work, the problem space is first depicted as a graph, and then a sparsest subgraph search method is utilized to find the initial cluster centroids. The proposed graph-based clustering strategy obtains users/items clusters, which are used as neighborhoods to provide diversified recommendations to the target user[12]. Ajoudanian and Abadeh proposed a novel approach to the top-M recommendation task using FCM. The approach adopts the sparsest sub – graph detection algorithm to define the initial clustering centroids. The authors demonstrated the efficacy of the proposed method using case studies on GitHub and showed improvement in recommendations in the presence of data sparsity[13]. Jiangzhou et al devised a k-medoids clustering algorithm based on the item rating probability distribution. The devised technique uses Kullback – Leibler (KL) divergence and distinguishes distinct items while choosing cluster centers. The approach also emphasizes on asymmetric relationships between items. The authors presented the proposed technique's effectiveness in terms of prediction accuracy and handling of data sparsity(14). Koohi and Kiani proposed two approaches based on map – reduce CBCF to provide viable recommendations in sparse environments. Experimental analysis of proposed approaches on the MovieLens and Jester datasets showed improvements in recommendations while handling data sparsity(15). Lee devised a similarity metric based on user genre preferences. Users are then clustered on the basis of genre preferences similarity using FCM, which alleviates the data scalability drawback and reflects the subjectivity of user ratings. To reduce the limitations of FCM, optimization of clusters is performed using the genetic algorithm. The proposed approach experiment on the MovieLens dataset showed improvement against many existing techniques in terms of various metrics(16). Li et al proposed a novel algorithm, namely Soft K – indicators Alternative Projection (SKAP), to solve the soft clustering problem and generate sparse partition matrices and top-M recommendations. Unlike FCM, SKAP is not affected by seed value selection. A detailed performance evaluation proved its superiority over existing CF techniques for the top-M recommendation task(17). Yadav et al employed a cluster-based technique, Cluster-based Diversity Recommendation (Clus-DR), to provide diverse recommendations in sparse environments. The technique clusters users with similar types of diversity and provides superior coverage while maintaining the acceptance level of RS' accuracy(18).

All the techniques discussed above are cluster-based and can handle only one dimension either user or item at a time. Another major limitation of cluster-based techniques lies in their inability to capture drift in users' interests or preferences, which is a key factor in generating pertinent recommendations. Several attempts have been made in this direction by proposing a time – aware and social RSs. Ahmadian et al discussed a recommendation technique using reliability measurements for improving the performance of time – aware RS. The authors used an amalgamation of similarity and trust statements for calculating similarity weights between users, which are utilized for obtaining reliability measurements for prediction. In their experiment work on the benchmark Epinions dataset, the authors proved the proposed technique's effectiveness in enhancing the prediction accuracy of RS while maintaining a good coverage(19). Rezaeimehr et al presented a time – aware RS method based on users' overlapping community structures. The presented method employs time – weighted association rules to provide appropriate recommendations. The authors demonstrated the performance of the method using the MovieLens datasets and revealed improved recommendation accuracy(20). Ahmadian et al proposed a social RS technique using an adaptive neighbor selection method. The proposed technique employs a clustering algorithm to obtain the user's neighborhood, which considers the user's historical ratings and social information. In addition, a confidence model is introduced to eliminate useless users from the obtained neighborhoods and optimize them. This new set of adaptive users is used for rating prediction and item recommendation. Experimental evaluation results on three benchmark rating datasets proved the proposed technique' effectiveness in terms of precision, recall and F1 score against several state-of-the-art techniques(21). Ahmadian et al discussed a social RS based on a temporal clustering strategy, which incorporates the time effects of ratings given by users. To achieve the objective, a temporal similarity metric is discussed, which ascribes more importance to recent ratings than older ones. In addition, the resulting clusters are optimized using an iterative process to obtain quality neighborhoods. Experimental evaluation results using the benchmark Epinions dataset proved the discussed technique's effectiveness in terms of accuracy and other coverage metrics(22). However, in real time, there are instances where multiple users are comparable to one another in terms of a subset of items, or vice versa. To disclose this duality between users and items, some researchers have worked out in this direction using co-clustering. Co-clustering is also known as block clustering, biclustering and two-mode clustering(11). Co-clustering discloses the duality among users and items by clustering them in both dimensions at the same time(2). Candel and Naccache explained that bipartite graphs are representation of datasets with two types of nodes connected through relationships. Therefore, the authors introduced a co-clustering strategy for bipartite graphs. The main feature of this introduced strategy is that it projects in low dimensional space to achieve dimensionality reduction and sparsity alleviation. The proposed algorithm operates only at the cluster-level instead of the item – level and achieves clustering using the mean-shift algorithm. The authors proved the algorithm's effectiveness in terms of recommendation accuracy. However, the challenge of handling scalability still remains(23). Beregovskaya I et al proposed a technique of combined filtering focused on co-clustering and information entropy (CBE – CF) to deal with data sparseness and heterogeneity. Specifically, it uses co-clustering to evaluate compact modules of a rating matrix and then uses a metric of information entropy to determine new user and compact module similarity. Finally, to gratify the recommendations, an equally weighted combination of user-based CF with optimized similarity metric and item-based CF is used(24). Choi S et al proposed a technique using biclustering and reinforcement learning (RL). Biclustering techniques are used to reduce state and action space and improve the quality of recommendations. RL is used for self-learning and further improvement(25). However, the proposed technique did not perform well in finding the optimal biclusters.

In the literature, a number of studies focused on similarity measures, clustering and co-clustering approaches have been discussed, each with its own set of merits and limitations depending on the context. Furthermore, most of the literature employs an implicit feedback dataset as a basic input and does not use the benchmark book-crossing rating dataset for experimental purpose. In the current research work, an initiative has been taken to work on explicit rating datasets and generate recommendations in sparse environments by utilizing co-clustering techniques. The experimental results of the proposed technique were compared against several state-of-the-art algorithms, which proved its effectiveness in terms of accuracy.

2 Methodology

2.1 Proposed technique: Nearest Co-cluster Collaborative Filtering (NCCF)

The proposed technique has three major phases:

Phase 1: Data preprocessing/discretization phase (Optional)
Phase 2: Co-clustering algorithm
Phase 3: Nearest co-cluster algorithm

2.2 Data preprocessing/discretization phase

In the proposed technique, it is assumed that users manifest their rating patterns on a standard scale. However, it is a well-established fact that different users may associate different preferences with different ratings. For example, in training set shown in Table 1, a rating value 6 may exhibit different meanings for different users.

Table 1. Example : Synthetic training and test dataset

Training dataset							
	B1	B2	B3	B4	B5	B6	B7
U1	4	-	8	2	8	6	-
U2	-	6	2	8	-	10	4
U3	8	-	4	-	4	-	10
U4	10	-	4	-	2	-	-
U5	10	2	-	2	-	-	6
U6	-	4	8	4	10	2	-
U7	-	4	10	-	8	2	-
U8	2	8	-	10	8	6	-
Test dataset							
	B1	B2	B3	B4	B5	B6	B7
U9	10	-	8	-	2	-	4

In nearest - neighbor CF techniques, this fact has already been demonstrated in UBCF (Herlocker et al. 1999) and IBCF (Sarwar et al. 2001) as a similarity measure with the Pearson correlation coefficient (shown in Eq. 1) and the adjusted cosine similarity (shown in Eq. 2), respectively. Furthermore, the RS's accuracy should not be affected by negatively rated items, so a positive rating threshold is maintained, and all negative ratings (<=4) are discarded. Since co-clustering techniques reveal the dualism between users and items and cluster them simultaneously, so it is possible to obtain clusters of users who share similar preferences across subset of items. The proposed approach aims to obtain clusters of users who have rated items positively (above the P_t rating threshold). Thus, the dataset can be discretized to binary values using the P_t rating value as the discretization threshold.

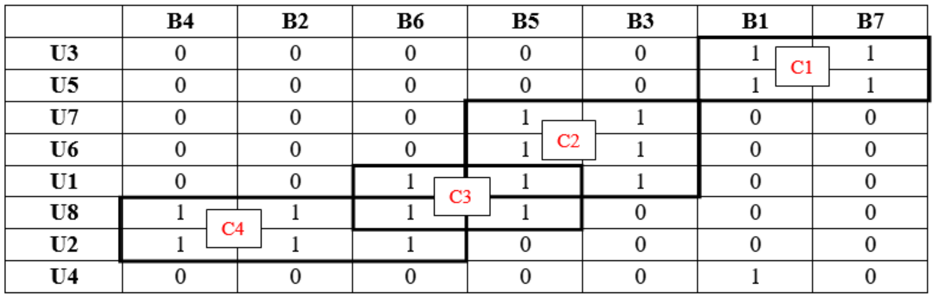
$$\mathrm{Sim}(u,v)=\frac{\sum_{\forall i\in S}\left(r_{u,i}-\bar{r}_u\right)\left(r_{v,i}-\bar{r}_v\right)}{\sqrt{\sum_{\forall i\in S}\left(r_{u,i}-\bar{r}_u\right)^2}\sqrt{\sum_{\forall i\in S}\left(r_{v,i}-\bar{r}_v\right)^2}},S=I_u\cap I_c \tag{1}$$

$$\mathrm{Sim}(i,j)=\frac{\sum_{\forall u\in T}\left(r_{u,i}-\bar{r}_u\right)\left(r_{u,j}-\bar{r}_u\right)}{\sqrt{\sum_{\forall u\in U_i}\left(r_{u,i}-\bar{r}_u\right)^2}\sqrt{\sum_{\forall u\in U_j}\left(r_{u,j}-\bar{r}_u\right)^2}},T=U_i\cap U_j \tag{2}$$

Data discretization is elective and can be excluded in the case where co-clusters are obtained with coherent values on both items and users. In the following section, it can be seen that in the case of the Bimax algorithm, clusters are obtained with constant values and require discretization. However, with the xMotif algorithm, discretization is not required as it obtains clusters with coherent values.

2.3 Co-clustering algorithm

In this phase, two distinct classes of co-clusters have been investigated using co-clustering algorithms: Bimax and xMotif. The primary goal of the Bimax algorithm is to obtain the inclusion maximal co-clusters. Inclusion maximal co-clusters are the those which are not a complete part of any other co-cluster. In the Bimax algorithm, a co-cluster C (Uc, Bc) represents a set of users Uc ∈ U who have positively co-rated a number of books Bc such as Bc ∈ B and generates co-clusters with constant values, as depicted in Figure 1.



	B4	B2	B6	B5	B3	B1	B7
U3	0	0	0	0	0	1	1
U5	0	0	0	0	0	1	1
U7	0	0	0	1	1	0	0
U6	0	0	0	1	1	0	0
U1	0	0	1	1	1	0	0
U8	1	1	1	1	0	0	0
U2	1	1	1	0	0	0	0
U4	0	0	0	0	0	1	0

Fig 1. Co-clusters obtained after applying discretization and Bimax co-clustering algorithm

As input, the minimum number of users in each co-cluster is set to 2 such that Uc >=2, and the minimum number of books in each co-cluster is also set to 2 such that Bc >=2. The obtained co-clustered are described as follows:

C1: Uc1 = {U3, U5}, Bc1 = {B1, B7}

C2: Uc2 = {U6, U7, U1}, Bc2 = {B5, B3}

C3: Uc3 = {U1, U8}, Bc3 = {B6, B5}

C4: Uc4 = {U8, U2}, Bc4 = {B4, B2, B6}

It is observed from Figure 1 that co-clusters C2 and C3 are overlapped for book B5 and co-clusters C3 and C4 are overlapped for book B6. In case, if overlapping is not allowed between co-clusters, then co-cluster C3 will be removed. However, there is a trade-off between the number of clusters in overlapping co-clusters and the RS's performance: (a) A smaller number of co-clusters impact the RS's effectiveness as many other useful clusters will be removed, or (b) a large number of clusters negatively impact the RS's performance with respect to time and computational complexity because of the large number of clusters to be examined. This constraint is investigated further in the paper.

In xMotif algorithm, given a set of users with ratings across a set of items and user-defined parameters α and β such as $0 < \alpha, \beta < 1$, a co-cluster C (Uc, Bc) represents a subset of users Uc ∈ U, who mutually exhibit coherent rating behavior on a number of books Bc ∈ B. In addition, the algorithm should also satisfy the following conditions:

- **Size:** users belonging to Uc should be at least the α- fraction of the total number of users,
- **Conservation:** each book belonging to Bc should be conserved among all the users belonging to Uc, i.e. the book is in the similar state for all users belonging to Uc, and
- **Maximality:** for each book not belonging to Bc, the item should be conserved in at most a β-fraction of users belonging to Uc.

The result of the xMotif algorithm is depicted in Figure 2. Four co-clusters have been obtained for the training dataset given in Table 1.

A threshold value of 2 was set for the minimum number of users and books in a co-cluster. The obtained co-clusters are described as follows:

C1: Uc1 = {U3, U4, U5}, Bc1 = {B1, B7}

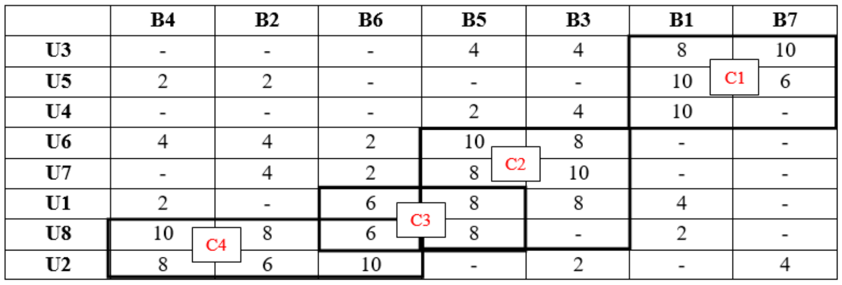
C2: Uc2 = {U6, U7, U1, U8}, Bc2 = {B5, B3}

C3: Uc3 = {U1, U8}, Bc3 = {B6, B5, B3}

C4: Uc4 = {U8, U2}, Bc4 = {B4, B2, B6, B5}

2.4 Nearest co-cluster algorithm

To provide valuable recommendations, there is an overwhelming need to identify co-clusters consisting of users with preferences who exhibit strong partial similarity with the test user. The process is carried out entirely online and comprises two key procedures:



	B4	B2	B6	B5	B3	B1	B7
U3	-	-	-	4	4	8	10
U5	2	2	-	-	-	10	6
U4	-	-	-	2	4	10	-
U6	4	4	2	10	8	-	-
U7	-	4	2	8	10	-	-
U1	2	-	6	8	8	4	-
U8	10	8	6	8	-	2	-
U2	8	6	10	-	2	-	4

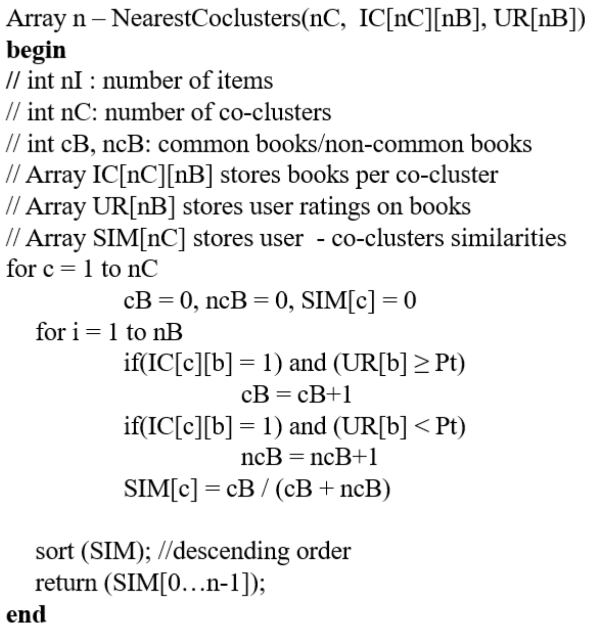
Fig 2. Co-clusters obtained using xMotif co-clustering algorithm

- Creation of test users' neighborhood i.e., locating the n closest co-clusters.
- Generation of top-M recommendation list.

To discover the n nearest co-clusters, the test user's similarity is computed with each obtained co-cluster. The major difference from previous research work is that the similarity between test user and a co-cluster is computed only on the books that are part of the co-clusters, and not on all the books that the test user has rated. This allows the detection of partial preferences. To compute the similarity between test user and each co-cluster, the common set of books is divided by the sum of the books they have in common and do not have in common. The Eq. 3 computes the similarity of test user u and each co-cluster c as follows:

$$Sim\,(u,\,c) = \frac{(B_u \cap B_c|}{(B_u \cap B_c| + \,(B_c \,-\, B_u|} \qquad (3) \qquad (3)$$

Where, the similarity values Sim (u, c) range between [0, 1]. The algorithm for similarity value computation between test user and each co-cluster is shown in Figure 3.



```
Array n – NearestCoclusters(nC,  IC[nC][nB], UR[nB])
begin
// int nI : number of items
// int nC: number of co-clusters
// int cB, ncB: common books/non-common books
// Array IC[nC][nB] stores books per co-cluster
// Array UR[nB] stores user ratings on books
// Array SIM[nC] stores user  - co-clusters similarities
for c = 1 to nC
            cB = 0, ncB = 0, SIM[c] = 0
   for i = 1 to nB
            if(IC[c][b] = 1) and (UR[b] ≥ Pt)
                        cB = cB+1
            if(IC[c][b] = 1) and (UR[b] < Pt)
                        ncB = ncB+1
            SIM[c] = cB / (cB + ncB)

   sort (SIM); //descending order
   return (SIM[0…n-1]);
end
```

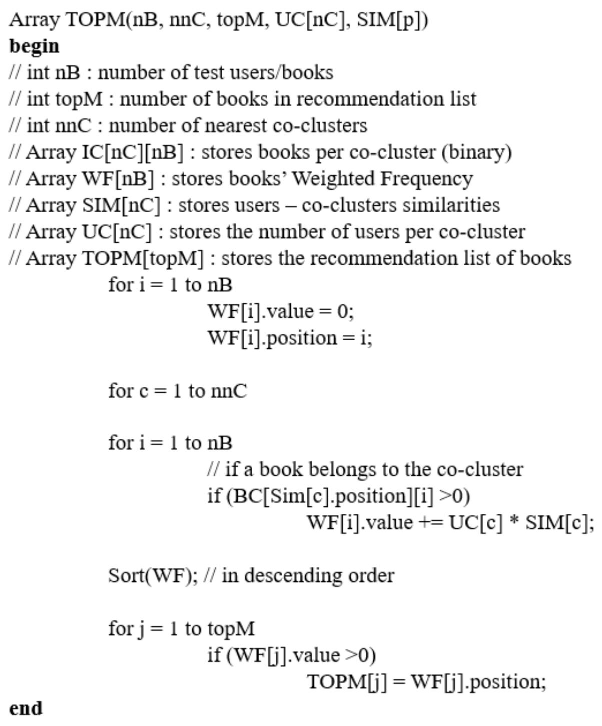
Fig 3. Algorithm for obtaining test user's co-cluster neighborhood

In the following phase, the top-M recommendation list is generated. To accomplish this, the appearance frequency of each item is to be determined and then the top-M most frequent ones are recommended. Eq. 4 shows how weighted frequency (WF) of a book b in a co-cluster c can be calculated as the product between |Uc| and the similarity value sim (u, c) as obtained using

Eq. 3. This way, each co-cluster's contribution is weighted with its size along with its similarity to the test user:

$$WF\,(b,\ c) = Sim\,(u,\ c) * \ (U_c| \qquad (4) \qquad (4)$$

As a final step, the most frequent item recommendation (which suggests the books appearing most frequently in the obtained neighborhood of the test user) procedure is implemented. Then the addition of weighted frequencies of books is performed, sorted and the top-M books are recommended from the resulting list, personalized according to the interests of each test user. Figure 4 depicts the algorithm to generate the top-M recommendation list.



```
Array TOPM(nB, nnC, topM, UC[nC], SIM[p])
begin
// int nB : number of test users/books
// int topM : number of books in recommendation list
// int nnC : number of nearest co-clusters
// Array IC[nC][nB] : stores books per co-cluster (binary)
// Array WF[nB] : stores books' Weighted Frequency
// Array SIM[nC] : stores users – co-clusters similarities
// Array UC[nC] : stores the number of users per co-cluster
// Array TOPM[topM] : stores the recommendation list of books
        for i = 1 to nB
                WF[i].value = 0;
                WF[i].position = i;

        for c = 1 to nnC

        for i = 1 to nB
                // if a book belongs to the co-cluster
                if (BC[Sim[c].position][i] >0)
                        WF[i].value += UC[c] * SIM[c];

        Sort(WF); // in descending order

        for j = 1 to topM
                if (WF[j].value >0)
                        TOPM[j] = WF[j].position;
end
```

Fig 4. Algorithm for generating the top-M recommendation list

In the case of the running example, suppose that all obtained co-clusters are retained, but only the two nearest co-clusters (k = 2) are to be chosen. In that case, it can be seen from Figure 1 that two books (B1, B3) are rated favorably by U9. Further, the similarity to each of the obtained co-clusters is (0.5, 0.5, 0, 0), respectively. Hence, the nearest neighborhood formation of test user comprises first two co-clusters, and the books recommendations are B7 and B5 using Bimax algorithm. For xMotif algorithm, considering the above mentioned assumptions, the similarity of the test user U9 to each of the co-clusters is (0.5, 0.5, 0.33, 0), respectively. It is observed that the similarity values obtained using xMotif differ significantly from the values obtained using Bimax.

3 Results and Discussion

NCCF is implemented and evaluated using python's data analysis packages. For the purpose of experiment, the benchmark book-crossing rating dataset is utilized. The dataset contains 1,149,780 ratings on 271,379 books from 278,858 users. Rating is given on the scale of 1 to 10. Rating information is organized as a series of comma-separated entries, one for each row, with the user ID, ISBN number, and rating of the book.

3.1 Validation

This section compares the performance of the proposed techniques NCCF – B and NCCF – X with existing CF algorithms such as UBCF and IBCF. A CF representative based on clustering, also abbreviated as CBCF (Xue et al. 2005), is employed for comparison purposes. For all these algorithms, their parameters were adjusted to match the original research work. The size of neighborhood (n, default value 20), the recommendation list's size (m, default value 20), and the training dataset's size are all controlled via parameters (default value 75%). All users that do not belong to the training dataset are included in the test dataset. The test set users serve as a source of compution for the metrics in consideration (precision, recall, and F1 score). Ratings of each test user are divided into observed books ratings and held – out books ratings. The test and training users' similarity metrics are calculated using the observed book ratings (default is 10). Held-out books' ratings are used to assess precision, recall, and F1 score. Each test was repeatedly done 20 times (with a distinct training set picked randomly each time), with statistical significance at level 0.05(on the basis of a two-tailed t-test). The CBCF algorithm, a k-means based algorithm requires an input variable in order to calculate an optimal number of user clusters. Consequently, the CBCF algorithm was put to the test with different cluster sizes: 250, 500, 750, and 1,000 clusters. In case of book-crossing data set, the F1 metric findings were best obtained at 500 clusters. Finally, because the data was normalized by subtracting the user's average rating value, hence, P_t has been set to 0.

3.2 Performance Evaluation of NCCF –B

For NCCF – B algorithm, the only input parameter to be considered is the minimum number of users and books in the co-cluster. Consequently, it is critical to adjust these two input parameters to find the best co-clusters. From Figure 5, it can be seen that F1 value is at its best with the value of n (minimum number of users) set to 4. Specifically, for the value of n <= 6, the resultant value of F1 is high. Similarly, from Figure 6, it can be seen that the technique works well when the minimum number of books in the co-cluster are set to 10.

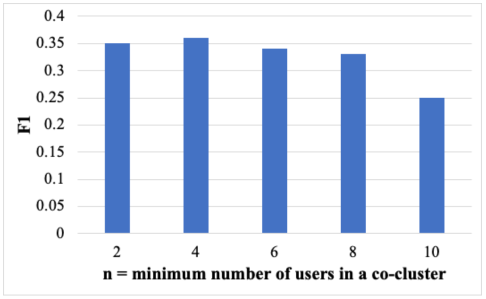


Fig 5. F1 versus tuning number of users

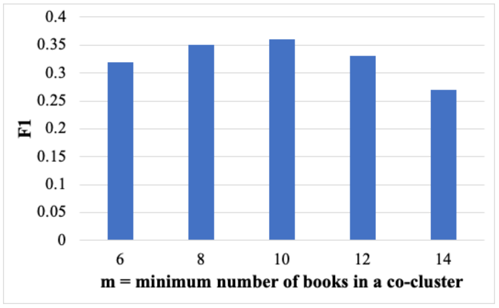


Fig 6. F1 versus tuning number of books

Bimax algorithm results into overlapped co-clusters because of the property of generating inclusion maximal co-clusters. The degree of co-clustering can be adjusted with respect to the number of co-clusters, to avoid resulting in a large number of overlapped co-clusters. Figure 7 depicts the relationship between F1 score and varied degree of overlapping. It is observed that increasing the percentage of overlapping negatively impacts the RS's performance and makes it less efficient. High overlapping percentage results into a greater number of co-clusters, further increasing the time and computational complexity in the system.

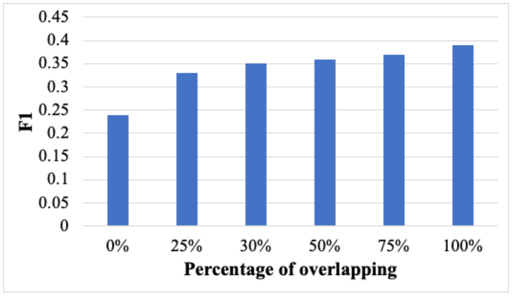


Fig 7. F1 versus tuning of overlapping factor

3.3 Performance Evaluation of NCCF – X

The xMotif algorithm necessitates the initial definition of a number of users as input parameter, which serves as a seed for co-clustering. The algorithm picks n_u (number of users in a co-cluster) with n_b (number of books) for each randomly generated seed. In addition, an input parameter (n_u * n_b) is defined for the total number of generated co-clusters, t. It is essential to adjust n_u and n_b variables to improve the efficacy and efficiency of recommendations.

It can be observed from Figure 8 and Figure 9 that optimal results are obtained at value of n_u as 4 and n_b as 100. Furthermore, to find optimum co-clusters (in terms of efficacy and efficiency), the p – value needs to be adjusted, as it governs co-clusters homogeneity. Thus, the performance of the F1 metric with various p-value parameter is compared and shown in Figure 10. It can be observed that a p-value of 0.001 results in the best F1 value. Finally, since the xMotif co-clustering algorithm does not result in co-clusters with the inclusion-maximal property, it indicates that the co-clusters largely overlap. However, this can be avoided by adjusting the permitted overlapping factor, as shown in Figure 11.

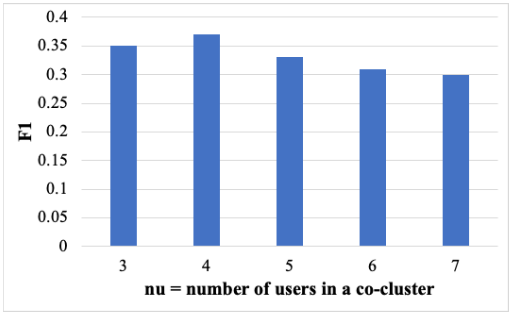


Fig 8. F1 versus tuning number of users

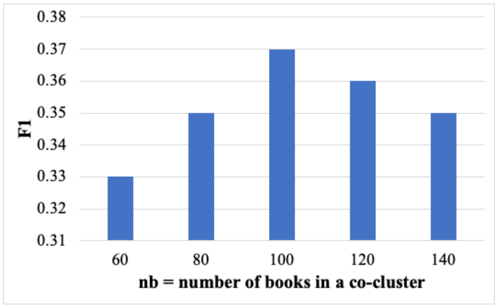


Fig 9. F1 versus tuning number of books

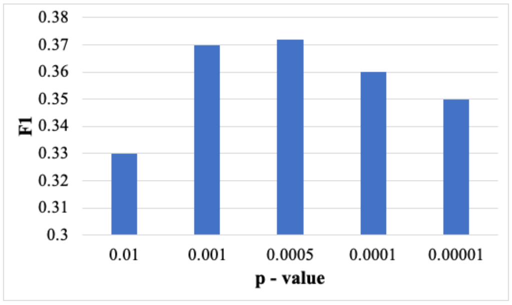


Fig 10. F1 versus tuning number of p-value parameter

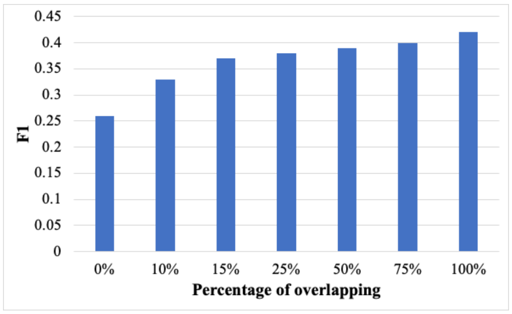


Fig 11. F1 versus tuning of overlapping factor

3.4 Comparative Analysis

A detailed comparative analysis of the proposed techniques NCCF - B and NCCF - X is carried out against the existing UBCF, IBCF and CBCF techniques. Figure 12**(a)** and **(b)** show the precision and recall values versus k findings for the book - crossing rating dataset. As expected, NCCF – B and NCCF – X performed significantly better than other techniques. The precision difference was found to be around 30%, while the recall difference was more than 10%, respectively. Furthermore, NCCF – X was observed to outperform NCCF – B slightly, but consistently due to co-clusters with a diverse range of books. The relationship between the F1 value and the k parameter is shown in Figure 13 .

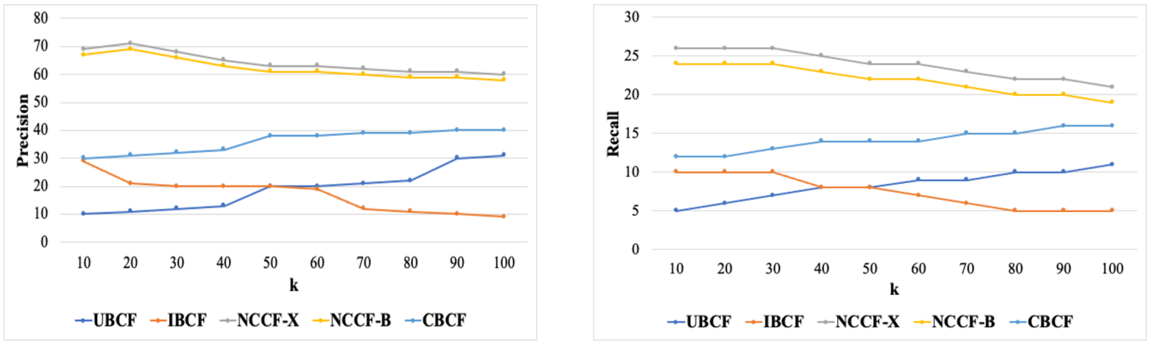


Fig 12. Comparison of UBCF, IBCF, NCCF-X, NCCF-B and CBCF in the context of (a) Precision and (b) Recall

Figure 14 depicts the execution time of each technique in milliseconds. Execution time is the time it takes a RS to provide recommendations to the target user.

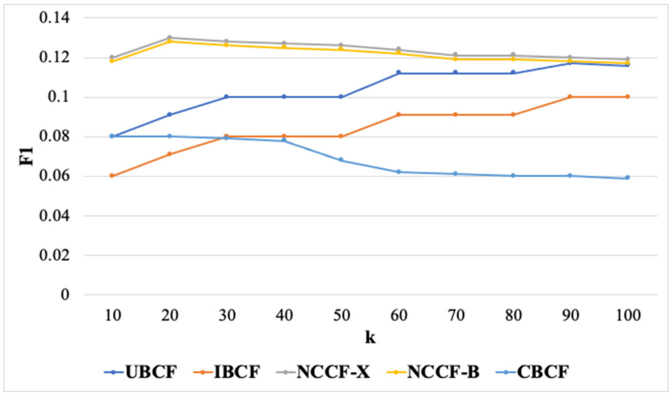


Fig 13. Comparison of UBCF, IBCF, NCCF-X, NCCF-B and CBCF in the context of F1 value

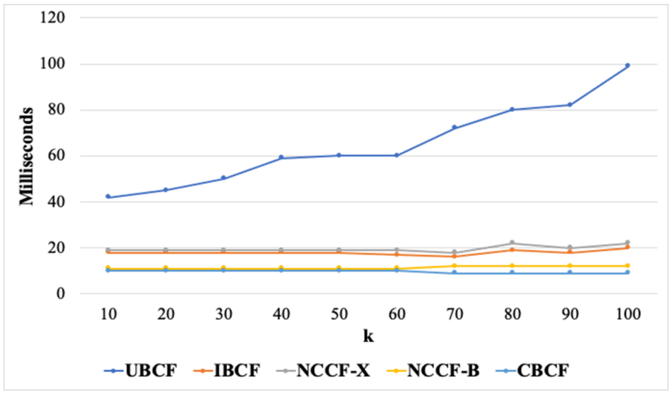


Fig 14. Comparison of UBCF, IBCF, NCCF-X, NCCF-B and CBCF in the terms of execution time

4 Conclusion

In this research work, it is proved that the proposed NCCF algorithm provides superior recommendations in sparse environments. The proposed NCCF algorithm is mainly based on co-clustering and a novel similarity index. The authors proved the effectiveness of co-clustering in dealing with users and items simultaneously. Furthermore, the used novel similarity index well - captured the partial matching of users' preferences and resulted in the formation of personalized co-cluster neighborhoods. This characteristic of the NCCF approach makes it ideal for a wide range of applications. In addition, a WF measure was considered for the top-M recommendation task, which showed an improvement in the RS's accuracy. A detailed experimental evaluation demonstrated the effectiveness of the proposed techniques NCCF – B and NCCF – X over UBCF, IBCF and CBCF with respect to precision, recall and F1 score. However, CBCF outperformed the other considered techniques with respect to execution time. The obtained results validate that the proposed work can be utilized to provide quality recommendations in a wide range of applications where data is sparse and RSs are being used.

References

1) Šegota SB, Anđelić N, Mrzljak V, Lorencin I, Kuric I, Car Z. Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator. *International Journal of Advanced Robotic Systems*. 2021;18(4). Available from: https://doi.org/10.1177/1729881420925283.
2) Jiang L, Liu L, Yao J, Shi L. A hybrid recommendation model in social media based on deep emotion analysis and multi-source view fusion. *Journal of Cloud Computing*. 2020;9(1):1–6. Available from: https://doi.org/10.1186/s13677-020-00199-2.
3) Ahmadian S, Afsharchi M, Meghdadi M. A novel approach based on multi-view reliability measures to alleviate data sparsity in recommender systems. *Multimedia Tools and Applications*. 2019;78(13):17763–17798. Available from: https://doi.org/10.1007/s11042-018-7079-x.
4) Dam NA, Dinh T. A Literature Review of Recommender Systems for the Cultural Sector. *Proceedings of the 22nd International Conference on Enterprise Information Systems*. 2020;p. 715–726. Available from: https://doi.org/10.5220/0009337807150726.
5) Ko H, Lee S, Park Y, Choi A. A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. *Electronics*. 2022;11(1):141. Available from: https://doi.org/10.3390/electronics11010141.

6) Kokkodis M, Ipeirotis PG. Demand-Aware Career Path Recommendations: A Reinforcement Learning Approach. *Management Science*. 2021;67(7):4362–4383. Available from: https://doi.org/10.1287/mnsc.2020.3727.
7) Logesh R, Subramaniyaswamy V, Malathi D, Sivaramakrishnan N, Vijayakumar V. Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. *Neural Computing and Applications*. 2020;32(7):2141–2164. Available from: https://doi.org/10.1007/s00521-018-3891-5.
8) Singh PK, Sinha S, Choudhury P. An improved item-based collaborative filtering using a modified Bhattacharyya coefficient and user–user similarity as weight. *Knowledge and Information Systems*. 2022;64(3):665–701. Available from: https://link.springer.com/article/10.1007/s10115-021-01651-8.
9) Fkih F. Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison. *Journal of King Saud University - Computer and Information Sciences*. 2022;34(9):7645–7669. Available from: https://doi.org/10.1016/j.jksuci.2021.09.014.
10) Shen R, Recommender. A Recommender System Integrating Long Short-Term Memory and Latent Factor. *Arabian Journal for Science and Engineering*. 2022;47(8):9931–9941. Available from: https://link.springer.com/article/10.1007/s13369-021-05933-9.
11) Wang Y, Zhao X, Zhang Z, Zhang LY. A collaborative filtering algorithm based on item labels and Hellinger distance for sparse data. *Journal of Information Science*. 2022;48(6):749–766. Available from: https://doi.org/10.1177/0165551520979876.
12) Moradi P, Ahmadian S, Akhlaghian F. An effective trust-based recommendation method using a novel graph clustering algorithm. 2015. Available from: https://doi.org/10.1016/j.physa.2015.05.008.
13) Ajoudanian S, Abadeh MN. Recommending human resources to project leaders using a collaborative filtering-based recommender system: Case study of gitHub. *IET Software*. 2019;13(5):379–385. Available from: https://doi.org/10.1049/iet-sen.2018.5261.
14) Deng J, Guo J, Wang Y. A Novel K-medoids clustering recommendation algorithm based on probability distribution for collaborative filtering. *Knowledge-Based Systems*. 2019;175:96–106. Available from: https://doi.org/10.1016/j.knosys.2019.03.009.
15) Koohi H, Kiani K. Two new collaborative filtering approaches to solve the sparsity problem. *Cluster Computing*. 2021;24(2):753–765. Available from: https://doi.org/10.1007/s10586-020-03155-6.
16) Lee S. Fuzzy clustering with optimization for collaborative filtering-based recommender systems. *Journal of Ambient Intelligence and Humanized Computing*. 2022;13(9):4189–4206. Available from: https://link.springer.com/article/10.1007/s12652-021-03552-8.
17) Li M, Wen L, Chen F. A novel Collaborative Filtering recommendation approach based on Soft Co-Clustering. *Physica A: Statistical Mechanics and its Applications*. 2021;561:125140. Available from: https://doi.org/10.1016/j.physa.2020.125140.
18) Yadav N, Pal S, Singh AK, Singh KK. Clus-DR: Cluster-based pre-trained model for diverse recommendation generation. *Journal of King Saud University - Computer and Information Sciences*. 2022;34(8):6385–6399. Available from: https://doi.org/10.1016/j.jksuci.2022.02.010.
19) Ahmadian S, Moradi P, Akhlaghian F. An improved model of trust-aware recommender systems using reliability measurements. *2014 6th Conference on Information and Knowledge Technology (IKT)*. 2014;p. 98–103. Available from: https://doi.org/10.1109/IKT.2014.7030341.
20) Rezaeimehr F, Moradi P, Ahmadian S, Qader NN, Jalili M. TCARS: Time-and community-aware recommendation system. 2018. Available from: https://doi.org/10.1016/j.furture.2017.04.003.
21) Ahmadian S, Meghdadi M, Afsharchi M. A social recommendation method based on an adaptive neighbor selection mechanism. *Information Processing & Management*. 2018;54(4):707–725. Available from: https://doi.org/10.1016/j.ipm.2017.03.002.
22) Ahmadian S, Joorabloo N, Jalili M, Meghdadi M, Afsharchi M, Ren Y. A Temporal Clustering Approach for Social Recommender Systems. *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2018;p. 1139–1144. Available from: https://doi.org/10.1109/ASONAM.2018.8508723.
23) Candel G, Naccache D. Co-embedding: Discovering Communities on Bipartite Graphs Through Projection. In: Future of Information and Communication Conference. Springer. 2022;p. 95–113. Available from: https://doi.org/10.48550/arXiv.2109.07135.
24) Beregovskaya I, Koroteev M. Review of Clustering-Based Recommender Systems. 2021. Available from: https://doi.org/10.13140/RG.2.2.34745.90725.
25) Choi S, Ha H, Hwang U, Kim C, Ha JW, Yoon S. Reinforcement learning based recommender system using biclustering technique. 2018. Available from: https://doi.org/10.48550/arXiv.1801.05532.