Knowl Inf Syst
DOI 10.1007/s10115-014-0771-x

REGULAR PAPER

Biclustering neighborhood-based collaborative filtering method for top-*n* recommender systems

Faris Alqadah · Chandan K. Reddy · Junling Hu · Hatim F. Alqadah

Received: 24 July 2013 / Revised: 27 February 2014 / Accepted: 14 July 2014

Abstract We propose a novel collaborative filtering method for top-*n* recommendation task using bicustering neighborhood approach. Our method takes advantage of local biclustering structure for a more precise and localized collaborative filtering. Using several important properties from the field of Formal Concept Analysis, we build user-specific biclusters that are "more personalized" to the users of interest. We create an innovative rank scoring of candidate items that combines local similarity of biclusters with global similarity. Our method is parameter-free, thus removing the need for tuning parameters. It is easily scalable and can efficiently make recommendations. We demonstrate the performance of our algorithm using several standard benchmark datasets and two paypal (in-house) datasets. Our experiments show that our method generates better recommendations compared to several state-of-the-art algorithms, especially in the presence of sparse data. Furthermore, we also demonstrated the robustness of our approach to increasing data sparsity and the number of users.

Keywords Recommender systems · Collaborative filtering · Biclustering · Top-*n* recommendation · Implicit feedback · Formal concept analysis

F. Alqadah
PayPal, San Jose, CA, USA
e-mail: falqadah@paypal.com

C. K. Reddy (✉)
Department of Computer Science, Wayne State University, Detroit, MI, USA
e-mail: reddy@cs.wayne.edu

J. Hu
Samsung Research, San Jose, CA, USA
e-mail: junlinghu@gmail.com

H. F. Alqadah
QM Scientific, Cincinnati, OH, USA
e-mail: hatim@qmscientific.com

1 Introduction

Recommender systems have seen a wide variety of applications in E-commerce, online games, display advertising and mobile applications. In spite of extensive work over the past fifteen years, building efficient algorithms for improving the accuracy of the recommender systems still remains to be an active area of research [24]. A key application is ranking a list of items for users to see. The more relevant a recommended list is, the more likely a user will click on it and make a purchase. This means increased revenue for companies and has huge commercial implication. The research problem is formulated as top-n recommendation problem, where performance is measured by number of hits in the top-n ranked list. One solution is to use the collaborative filtering (CF) technology which will enable the system to rank the items. While several methods have been proposed in the literature to solve this problem, they either suffer from computational speed and scalability issues, or the requirement of model tuning (specifically for model-based approaches).

In this paper, we propose a bicluster neighborhood-based approach for making top-n recommendations. It explores bicluster similarity in addition to the standard item similarity. A bicluster is a subset of users and items which forms a dense submatrix, where every user has an interaction with every item [18]. In other words, it represents a dense neighborhood of the user. By exploring the hierarchical relationship between one bicluster with other biclusters, we construct bicluster similarity to capture local proximity between a recommended item and a user. We then combine this local measure with a global distance measure to create a balanced ranking score for every item. We applied our algorithm to several datasets with *implicit feedback* which is typically used to evaluate most of the top-n recommender systems. Implicit feedback data are commonly seen in E-commerce applications where the user feedback is in the form of purchase, website browsing activities or search activities. Early recommender systems were built on *explicit feedback data* such as user ratings, but such data are normally hard to gather from the users.

We list the primary advantages of the proposed bicluster neighborhood algorithm:

1. *Parameter-free*: As opposed to several existing model-based approaches, our algorithm does not require any parameters to produce the recommendations.
2. *Scalability*: Since there is no model-building necessary for our approach, the recommendations for each user can be performed independent of each other. Moreover, since the recommendations for each user are independent of each other and can be performed in an embarrassingly parallel manner, our approach is extremely scalable to large-scale datasets.
3. *Interpretability*: Biclustering offers a solution to the problem of curse of dimensionality in large datasets in addition to comprehensibility [22]. Utilizing our framework we can clearly traceback the source of recommendations, without utilizing the "blackbox" of latent space.
4. *Flexibility*: Our algorithm is flexible in modeling the user preferences using both local and global similarities.

Our experiments on several publicly available datasets and two large PayPal datasets show that our algorithm outperforms all existing state-of-the-art algorithms in sparse datasets. Most importantly, our algorithm scales better with respect to the number of users. The organization of this paper is as follows: In Sect. 2, we review existing works. The theoretical basis of our methodology is described in Sect. 3. Section 4 details the framework, while Sect. 5 presents the results of experiments. Finally, Sect. 6 offers concluding remarks, shortcomings, and avenues for future work.

2 Related work

Existing collaborative filtering algorithms can be broadly classified into two categories:

- *Memory-based algorithms*, which typically make the recommendations based on the preferences of the nearest neighbors in the data. Item-based CF algorithms (*Item CF*) are the popular choice in this category and usually outperform user-based algorithms [7]. Since their development more than a decade ago, several advanced methods have been proposed, including instance based methods [9] for feature weighting and instance selection. Nevertheless, they form a strong baseline method to comparatively evaluate any new approach for this problem.
- *Model-based algorithms*, on the other hand, typically make recommendations by first developing a model of user ratings. Regression [20], matrix factorization algorithms [14] and Bayesian methods [19] are the popular ones that fall into this category. These methods either explore a 'latent space" or build a model to capture the relationship. Recently, proposed sparse linear methods for top-*n* recommendation problem also fall into this category [16]. The authors have demonstrated that their approach can simultaneously produce high-quality recommendations in quick time. By solving L_1-norm and L_2-norm regularized optimization problems, sparsity is induced into the final recommendations. This makes it suitable for real-time applications. We also compare the performance of this approach (*SLIM*) to our work in the experimental results section.

A few excellent surveys that provide more detailed descriptions of the prominent collaborative filtering algorithms are also available [1,21]. Several other works [13,23] combine the advantages of memory-based and model-based collaborative filtering of approaches by introducing a smoothing-based method.

Various approaches have been proposed to handle implicit feedback data. One of the first works that use implicit feedback in the context of recommender systems is [17]. Then, item-based top-*n* algorithm [7] was proposed. Some of the challenges of using implicit feedback for recommender systems are well described in [12]. In this paper, Hu et al. proposed a Weighted Regularized Matrix Factorization (*WRMF*) method which scales linearly with the size of the data. While this approach outperforms the item-based approaches in ranking results, it has the same computational problem. Both methods need to process the entire matrix, which can be very expensive for handling large-scale datasets. E-commerce datasets could easily contain millions of rows and millions of columns. Thus, a more local approach (or scalable) would be desirable.

There are few works that proposed to use biclusters for collaborative filtering [8,11,15,22]. These methods aim to simultaneously cluster the users and items together and then recommend items. However, due to their static nature of obtaining the initial biclusters, they are inferior in performance for many practical purposes. For example, in [22], the authors developed an approach similar in philosophy to what we propose here. The major difference between the two approaches is that in [22] biclusters are computed offline and not "on demand"; in turn, this forces the authors to match the nearest biclusters to a user based solely on the item vector of a single user. On the other hand, in our approach we map a user to a bicluster on demand; hence, we match the nearest biclusters based on the interactions of the original user and similar users. This is a key difference, as matching the personalized bicluster, as opposed to a single users' interaction vector allows for exploiting the overall bicluster structure in the data. Moreover, computing the biclusters off-line may require potentially quadratic or exponential storage space and does not allow for easy incremental updates to the model. In summary, *contrary to the existing local approaches, in our approach, we obtain more personalized biclusters that are specific to the users of interest.*

Finally, the rank score we employ is based on novel bicluster similarity measures combined with a traditional global similarity. We leverage on our recent work on query-specific biclustering which provides a theoretical framework for building query-specific biclusters using several properties from Formal Concept Analysis (FCA) [2]. We build on that framework and use it to build user-specific biclusters that are "more personalized" to the users of interest. Using such biclusters, the final top-n recommendations are made by combining the items from the bicluster and the global similarity of the items.

3 Preliminaries

In this section, we will provide several important definitions that are needed to comprehend our algorithm: biclusters, neighboring biclusters, siblings, and smallest bicluster.

3.1 Biclusters

Let $\mathbf{K}$ denotes a matrix with rows corresponding to users and columns corresponding to items. If a user u has no interaction with item i (such as purchasing, browsing, rating etc), then $\mathbf{K}(u, i) = 0$. If there is any interaction between user u and item i, $\mathbf{K}(u, i) > 0$. In this paper, as with previous works on implicit feedback recommendations, we consider the binary case, in which all the real values are simply converted to '1's. *The top-n recommendation problem is to produce a ranked list of items for user u from those items that u does not currently interact with.* Let U and I denote subsets of users and items respectfully, we define a bicluster as the following:

Definition 1 A **bicluster** is a pair (U, I) such that $\mathbf{K}(u, i) > 0 \quad \forall u \in U, \; i \in I$ and the submatrix $\mathbf{K}[U, I]$ is **maximal**. Maximal indicates that the addition of any additional user or item will introduce a zero element.

Note that u and i need not to be adjacent on the matrix in order to form a bicluster. Intuitively, biclusters represent small clusters of users who are similar only with respect to a small cluster of items and vice versa. Thus, biclusters capture the notion of local or conditional similarity of users and items. Note that, under this definition of a bicluster, all formulations apply dually to rows and columns [10]; hence, all statements made in the sequel apply dually to item recommendations and user recommendations.

3.2 Neighboring biclusters

Formal Concept Analysis (FCA) [10] further stipulates a complete mathematical framework for ordering and structuring sets of biclusters. The set of biclusters in matrix $\mathbf{K}$ are ordered by the **hierarchial order**; under this ordering the set of biclusters form a complete lattice. Using this formulation identifying neighborhoods of closely related biclusters is a well-defined task and several efficient algorithms exist to identify these neighborhoods [5] in a local fashion.

Definition 2 Given biclusters (U_1, I_1) and (U_2, I_2), then $(U_1, I_1) < (U_2, I_2)$ if and only if $U_1 \subset U_2$ and $I_1 \supset I_2$. This ordering relation is referred to as the **hierarchical** ordering. A bicluster (U_1, I_1) is a **minimum bicluster** if there does not exist (U_2, I_2) such that $(U_1, I_1) > (U_2, I_2)$. A bicluster (U_1, I_1) is a **maximum bicluster** if there does not exist (U_2, I_2) such that $(U_1, I_1) < (U_2, I_2)$.

(a)

	i_1	i_2	i_3	i_4	i_5
u_1	0	0	1	0	1
u_2	0	1	0	1	0
u_3	0	0	1	1	0
u_4	1	1	0	0	1
u_5	0	0	1	1	0
u_6	1	1	0	0	0
u_7	0	0	1	1	0
u_8	1	1	0	0	0
u_9	1	0	1	0	1
u_{10}	1	1	0	0	0

(b)

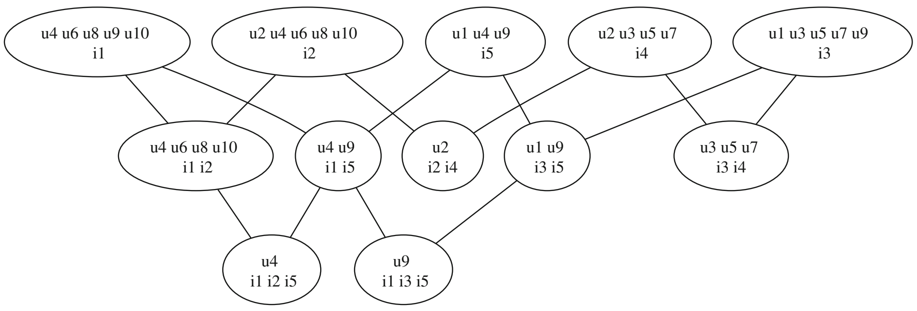


Fig. 1 Running example: **a** a simple synthetic dataset containing 10 users and 5 items, **b** a lattice of biclusters corresponding to the synthetic dataset

Please note that the size of the users and the size of the items go in the opposite direction in the hierarchy. A bicluster in the higher order has more users and fewer items. We further define the upper and lower neighbors of a bicluster.

Definition 3 A bicluster (U_1, I_1) is an **upper neighbor** of (U_2, I_2) if $(U_1, I_1) > (U_2, I_2)$, and there doest not exist bicluster (U_3, I_3) satisfying $(U_1, I_1) > (U_3, I_3) > (U_2, I_2)$.

Definition 4 A bicluster (U_1, I_1) is a **lower neighbor** of (U_2, I_2) if $(U_1, I_1) < (U_2, I_2)$, and there doest not exist bicluster (U_3, I_3) satisfying $(U_1, I_1) < (U_3, I_3) < (U_2, I_2)$.

Note that an upper (or lower) neighbor of a bicluser is not unique. There can be several upper neighbors as illustrated in Fig. 1.

3.3 Sibling and smallest bicluster

We will now define the siblings of a bicluster and the smallest bicluster.

Definition 5 A bicluster (U_1, I_1) is a **sibling** of (U_2, I_2) if there exists a bicluster (U_3, I_3) such that (U_3, I_3) is an upper (or lower) neighbor of both (U_1, I_1) and (U_2, I_2).

We are interested in mapping a user to the smallest bicluster that contains this user.

Definition 6 Given user u and $u \in U_1$, a bicluster (U_1, I_1) is the **smallest bicluster for user u** if there does not exist a bicluster (U_2, I_2) such that $(U_2, I_2) < (U_1, I_1)$ and $u \in U_2$.

The smallest bicluster for a user u corresponds to the bicluster that contains user u and the greatest number of items. It is easy to prove that the smallest bicluster for a given user is unique and will contain the fewest number of users. But note that the smallest bicluster may not be a minimum cluster. For example, in Fig. 1, the smallest bicluster for user u_6 is $(\{u_4, u_6, u_8, u_{10}\}, \{i_1, i_2\})$. But it is not a minimum cluster.

Intuitively, neighboring biclusters are assumed to be similar. This intuition has been formalized and exploited to extract knowledge in both binary and real-valued [2,3,25] contexts. In the next section, we will illustrate how exploring bicluster neighborhoods yields suitable candidates for top-n recommendation.

4 Bicluster neighborhood collaborative filtering

The bicluster neighborhood framework (*BCN*) for collaborative filtering consists of three basic steps:

1. Given user u, map u to the smallest bicluster $C = (U, I)$ that contains u.
2. Identify candidate set I of items for recommendation by exploring the bicluster neighborhood of I.
3. Rank items in I by combining global and bicluster neighborhood similarity.

The proposed approach initially maps each user $u \in \mathbf{U}$ to a bicluster C. Then, more similar users to u are identified by exploring the bicluster neighborhood of C while simultaneously building up a candidate set of items for recommendation. This is done by appending those items in the neighboring biclusters to the candidate set. Finally, the candidate set of items are ranked by combining bicluster similarity and global similarity measures. These steps are illustrated below using the sample synthetically generated dataset and the bicluster lattice depicted in Fig. 1.

4.1 Mapping users to biclusters

We start by mapping a user u to the smallest bicluster (U, I) that contains u. This is achieved via a two-step procedure: First, the set of all items I that u interacts with are identified. Next, the set of users U that all $i \in I$ jointly interact with are computed via set intersection. For example, user u_3 in Fig. 1 is mapped to the bicluster $(\{u_3, u_5, u_7\}, \{i_3, i_4\})$.

4.2 Creating candidate set

As shown in Proposition 1, the items most similar to the items in I with respect to the user set U are located in the lower bicluster neighborhood of C. As a result, items of the lower bicluster neighborhood form suitable candidates for top-n recommendation.

Proposition 1 *Given the smallest bicluster $C = (U, I)$ for user u, and lower neighbor biclusters (U_l, I_l), let J denote the Jaccard index* **defined over user set** *U. Then, for all $i \in I$ and all $i' \in \mathbf{I} \backslash I$:*

$$\arg\max_{i'} J(i, i') \subseteq \bigcup_{(U_l, I_l)} I_l \tag{4.1}$$

where all (U_l, I_l) *are lower neighbor biclusters of* (U, I).

Proof Two cases are possible: (1) C is a minimal bicluster or (2) C is not a minimal bicluster.

1. In this case, all items i' have no interactions with users in U; equivalently $\mathbf{K}(u, i') = 0$ for any $u \in U$. This holds true by definition of a bicluster and the fact that C in minimal. As result, for any item i', $J(i, i') = 0$. Hence, the proposition is vacously true.
2. We will prove this case by contradiction. Let I_{max} denote $\arg\max_{i'} J(i, i')$ and I_{cand} denote $\bigcup_{(U_l, I_l)} I_l$. For any $i* \in I_{max}$, assume that $i* \notin I_{cand}$. Recall that $J(i, i*)$ defined over the user set U is defined as:
$$\frac{|U_i \cap U_{i*} \cap U|}{|(U_i \cup U_{i*}) \cap U|} \tag{4.2}$$
where U_i is the set of all users who interact with i, and U_{i*} be the set of users who interact with $i*$. By the definition of a bicluster and hierarchical order, U_i is a superset of U; therefore, the denominator of the Jaccard index is always $|U|$. As a result, computing $J(i, i*)$ can be reduced to computing $|U_i \cap U_{i*} \cap U|$. However, by definition of lower neighbors, the item(s) that maximize $|U_i \cap U_{i*} \cap U|$ are only contained in the lower neighbor(s) of C. Hence, $i*$ cannot be in I_{max}. □

In the special case when a bicluster C is a minimum bicluster, we cannot use lower neighbors for getting additional items to make recommendation.

Hence, we need to take the extra step of computing the lower neighbors of the upper neighbors of C, or the siblings of C.

Example 1 Consider the example in Fig. 1. If we want to make recommendation to u_3, the smallest bicluster for u_3 is $(\{u_3, u_5, u_7\}, \{i_3, i_4\})$. Since this bicluster is also a minimum bicluster, the sibling biclusters $(\{u_1, u_9\}, \{i_3, i_5\})$ and $(\{u_2\}, \{i_2, i_4\})$ reveal items $\{i_5\}$ and $\{i_2\}$ as candidate items.

The procedure outlined above may be applied recursively to each neighboring bicluster to generate a greater number of candidates. As will be discussed in the experimental results section, we found that it is typically sufficient to apply at most two levels of candidate generation.

4.3 Ranking candidate items

In the final step of the framework, the candidate items are ranked by combining global and bicluster neighborhood similarity. Three principles guide the ranking process:

1. The global similarity of each candidate item to the initial items should be considered.
2. The similarity of the minimum bicluster to the neighborhood biclusters containing a candidate item should be considered.
3. The ranking score should be monotonic with respect to the number of biclusters shared between candidate items and initial items.

The first criterion reflects the guiding principle of traditional collaborative filtering methods; global similarity refers to similarity computed across the entire data matrix. The second principle captures local similarities between locally similar users and items. Finally, notice that each candidate item may in fact belong to several of the neighborhood biclusters. Naturally, the more biclusters that initial items and candidate items share should, at a minimum, not decrease the rank score of a candidate item.

Abiding by the principles outlined above we propose three simple ranking functions. Let $C = (U, I)$ be the smallest bicluster and C be the set of all neighboring biclusters $C' = (U', I')$ from which the candidate items are drawn. The ranking score of a candidate item i with respect to user u is computed generally as

$$r(u, i') = g(u, i') \times l(u, i') \tag{4.3}$$

where $g(\cdot)$ calculates the average global distance between user u and item i', and $l(\cdot)$ returns the local distance based on bicluster similarity. Computing bicluster similarity is an integral part of ranking the candidate items. We will now elaborate more on these different similarity measures and provide implementation details along with some analysis.

4.3.1 Global similarity

We are only interested in items belonging to biclusters. Let (U, I) and (U', I') be two biclusters, and item $i \in I$ and item $i' \in I'$. The global distance between user u and item i' is defined as:

$$g(u, i') = \frac{\sum_{i \in I} J(i, i')}{|I|} \tag{4.4}$$

where $J(i, i')$ is the Jaccard index, which is defined over all users who interact with i and those who interact with i'. Let U_i be the set of all users who interact with i, and $U_{i'}$ be the set of users for i'

$$J(i, i') = \frac{|U_i \cap U_{i'}|}{|U_i \cup U_{i'}|} \tag{4.5}$$

4.3.2 Bicluster similarity

Bicluster similarity measures have been proposed earlier in our previous work [4]. Let $C = (U, I)$ and $C' = (U', I')$ be two different biclusters.

We define the union of these two biclusters as $D = (U \cup U', I \cup I')$, then the **zeros-induced** similarity measure is

$$b(C, C') = 1 - \frac{zeros(D)}{|D|} \tag{4.6}$$

where $|D| = |U \cup U'| * |I \cup I'|$ and $zeros(D)$ is the number of zeros occurring in the submatrix D. Clearly, the fewer the zeros the more similar C and C' are deemed to be. Assume that C is our initial cluster, and C' is a lower neighbor or sibling of C. By definition, at least one zero must be in D.

4.3.3 Local similarity

The rank of a candidate item is computed by aggregating the bicluster similarity of all biclusters in which i occurs to the minimum bicluster. More rigorously, we propose utilizing the three aggregating functions of summation, average, and maximum. Mathematically,

$$l(u,i') = \sum_{C' \in Z} b(C,C') \tag{4.7}$$

$$l(u,i') = \frac{1}{|Z|} \sum_{C' \in Z} b(C,C') \tag{4.8}$$

$$l(u,i') = \max_{C' \in Z} b(C,C') \tag{4.9}$$

The aggregator functions for global similarity are dually defined. The guiding principles hold true for all three aggregating functions, with the exception that the average aggregating function is not guaranteed to be monotonic with respect to the number of biclusters shared between candidate items and initial items. Selection of the appropriate aggregating function is explored in the experimental results section.

Example 2 We want to make recommendation to user u_6. The smallest bicluster for u_6 is $(\{u_4, u_6, u_8, u_{10}\}, \{i_1, i_2\})$, which has a lower neighbor $(\{u_4\}, \{i_1, i_2, i_5\})$. One candidate item is i_5. Then, ranking score for i_5 is:

$$\begin{aligned} r(u_6, i_5) &= g(u_6, i_5) \times \max_{C' \in Z} b(C, C') \\ &= \left(\frac{0.142 + 0.33}{2}\right) \times \left(1 - \frac{3}{12}\right) \\ &= 0.1773 \end{aligned}$$

4.4 Implementation and analysis

```
Input: $u_k$: user to perform recommendation for
Input: $\mathbf{K}$: full data matrix
Input: $n$: desired num of recommendations
Result: Return a size $n$ list of recommended items
1  begin
2      $C \leftarrow (U, I) \leftarrow$ Smallest bicluster of $u_k$ $\mathrm{C_l} \leftarrow LowerNeighbors(C)$ ;
3      $\mathrm{C_s} \leftarrow Siblings(C)$ ;
4      $cands \leftarrow \emptyset$ ;
5      for $C_l \in \mathrm{C_l}$ do
6          $cands \leftarrow cands \cup I_l \backslash I$
7      for $C_s \in \mathrm{C_s}$ do
8          $cands \leftarrow cands \cup I_s \backslash I$
9      for $i \in cands$ do
10         Compute $r(u, i)$
11     Return top $n$ items in $cands$ ranked by $r(u, i)$
12 end
```

Algorithm 1: The *BCN* framework for top-*n* recommendation

The bicluster neighborhood *BCN* framework for recommendations is depicted in Algorithm 1. The major computational burden of *BCN* is computing the lower and sibling biclusters of the minimum bicluster. Given minimum bicluster $C = (U, I)$, let $|U| = n$, $|I| = m$, and let $|\mathbf{U}| = N$, $|\mathbf{I}| = M$, the algorithm described in [5] prescribes a method to compute the neighboring biclusters in

$$O(M \times N) + O((N - n) \times m) \times M + M \times (N + M) \tag{4.10}$$

In the worst case scenario, this may amount to $O(M^2 + N)$; in this case the number of neighbors is assumed to be $O(M)$ and the number of interactions per item is assumed to

Table 1 Real-world datasets used for evaluating the performance of the proposed *BCN* algorithm

Dataset	No. of users	No. of items	No. of transactions	Density (%)
Paypal_big	15,000	156,992	674,223	0.000286
Paypal_small	30,000	5,576	134,059	0.0008
delic_bookmarks	1,867	69,226	104,799	0.0081
lastfm_friends	1,476	2,100	21,852	0.07
lastfm	1,883	18,745	92,834	0.33

be in $O(N)$. Clearly, in sparse data, this is not the case. Hence, the complexity is typically $O(N \times n \times m^2)$. Computing the global similarity of candidate items amounts to $O(N \times m)$. Therefore, it does not effect the overall computational complexity. Computing bicluster similarity is an $O(1)$ operation due to the properties of the bicluster lattice. For a given bicluster $C = (U, I)$ and lower neighbor $C_l = (U_l, I_l)$, from the definition of a bicluster the sub-matrix $\mathbf{K}[U_l \backslash U, I \backslash I_l]$ only contains zeros. As these set differences are computed while the bicluster neighbors are identified, the bicluster similarity requires no additional computation; the number of zeros in the union of the bicluster is simply the product of the user and item set differences.

4.4.1 Scalability

Bicustering neighborhood is scalable with respect to number of users and individual recommendation time. Notice that, computing the recommendation list for each user can be accomplished independently from all the other users. Initially, computing the recommendation for each user requires access to the entire data matrix; however, once the minimum bicluster is computed, only a small submatrix of the original matrix is required for each user. This fact allows *BCN* to be implemented in an embarrassingly parallel manner when computing recommendations for a large set of users. Moreover, since *BCN* is not a model-based approach, and bicluster computation is performed "on-demand", the incremental change in time to perform a single recommendation is also scalable. As more users are added to the data matrix, *BCN* still takes advantage of this new information without requiring the costly step of retraining a model. This computational aspect is investigated more closely in the experimental section.

5 Experimental results

5.1 Real-world datasets

The performance of the *BCN* framework was evaluated on five real-world datasets on implicit feedback; the characteristics of these are shown in Table 1. The *Paypal* datasets correspond to the user buying patterns at different online stores. The *lastfm* dataset represents user listening preferences with respect to different artists, while *lastfm_friends* contains friendship connections between users[1] [6]. Finally, *delic_bookmarks* contains users along with urls that they have bookmarked[2] [6].

5.2 Evaluation methodology

Five-time Leave-One-Out cross validation (LOOCV) is used to evaluate the *BCN* framework. Each dataset was split into a training and testing set by randomly selecting one of the nonzero entries of each user and placing it into the testing set. The recommendation algorithm is applied to the training dataset, and n recommendations are produced. The results we report here correspond to $n = 10$ and $n = 1$.

Recommendation quality is measured by the Hit Rate (HR) and the Average Reciprocal Hit-Rank ($ARHR$). If an item in the test set occurs in the n-size recommendation list of a user u_i, then a "hit" occurs. Using this definition, the hit rate is defined as the proportion of total hits to total users. Formally, let N be the number of users, and h be the number of hits,

$$HR = \frac{h}{N} \tag{5.1}$$

$ARHR$ is a weighted version of HR that rewards hits that occur closer to the top of n-size recommendation list. Let p_i be the position of a hit for user u_i, then

$$ARHR = \frac{\sum_{i=1}^{h} \frac{1}{p_i}}{N} \tag{5.2}$$

5.3 Experimental results

The performance of the *BCN* framework was compared with three algorithms: *SLIM* [16], *WRMF* [12], and basic *Item CF* [7], as described in the related work section. For *SLIM*, we used the source code provided by the authors of the original paper. For *WRMF*, we implemented a version based on the description provided in the corresponding paper. For *Item CF*, the mahout implementation was utilized (https://cwiki.apache.org/MAHOUT/recommender-documentation.html). *BCN* was implemented in C++ utilizing the three different similarity combination approaches described previously (*BCN-Sum*, *BCN-Avg*, *BCN-Max*) and was set to explore two levels of neighbors. All experiments were run on a Linux-based 6-core I7 Intel PC with 24 GB of main memory.

The results of the experiments are displayed in Table 2. With the exception of the *lastfm* dataset, *BCN* approaches outperform all other approaches. Traditional Item-based collaborative filtering performed significantly worse than all three methods; we believe this is due to the effect of sparsity in the data. Conversely, *BCN* methods performed best in sparse datasets and comparable to *SLIM* and *WRMF* in denser datasets. Notice that *paypal* and *delicious* datasets are each at least one order of magnitude sparser than the *lastfm* datasets. Additionally, in each of these sparser datasets, the best performing *BCN* method was *BCN-Sum*.

We find that the number of shared biclusters among the items played an important role in producing better recommendations. In sparse data, the number of neighboring biclusters is small. On the other hand, in dense datasets, the number of neighboring biclusters to initial biclusters tended to be quite large and this actually played a negative role in recommendation. The large number of neighboring bicluster seemed to cloud over significantly similar items by biasing the algorithm only toward frequently occurring items in neighbors. For this reason,

[1] http://lastfm.com.

[2] http://delicious.com.

Table 2 Experimental results on real-world datasets demonstrating recommendation quality using top 10 HR, $ARHR$ and top 1 HR

Dataset	Algorithm	HR-Top 10 (%)	ARHR	HR-top 1 (%)
paypal_big	Item CF	3.59	1.33	1.65
	WRMF	4.46	1.07	0.77
	SLIM	6.68	3.02	1.7
	BCN-Sum	**8.12**	**3.47**	**1.89**
	BCN-Avg	8.03	3.32	1.83
	BCN-Max	8.01	3.29	1.81
paypal_small	Item CF	1.17	0.94	0.2
	WRMF	4.01	1.2	0.64
	SLIM	5.24	**2.28**	1.26
	BCN-Sum	**6.66**	1.47	**2.77**
	BCN-Avg	6.18	1.3	2.5
	BCN-Max	6.13	1.25	2.34
delic	Item CF	1.31	0.16	0.35
	WRMF	1.05	0.17	0.42
	SLIM	1.05	0.53	0.87
	BCN-Sum	**5.38**	2.15	3.27
	BCN-Avg	5.01	**2.42**	3.44
	BCN-Max	5.32	2.39	**3.46**
lastfm_friends	Item CF	3.59	1.23	1.33
	WRMF	17.08	6.66	**3.23**
	SLIM	13.8	5.88	3.14
	BCN-Sum	17.38	**6.91**	2.59
	BCN-Avg	16.75	6.35	2.85
	BCN-Max	**17.63**	6.70	2.99
lastfm	Item CF	0.26	0.005	0.001
	WRMF	21.4	9.81	5.59
	SLIM	**22.0**	**10.8**	**6.82**
	BCN-Sum	19.85	10.22	6.46
	BCN-Avg	21.37	10.58	6.79
	BCN-Max	20.45	10.28	6.56

The best results are highlighted in bold

we believe that the best *BCN* scores in these datasets came from *BCN-Max*. Interestingly, setting the *BCN* framework to explore only a single level of neighboring biclusters performed on par with two-level search in the sparse datasets.

However, in dense datasets, there was a significant decline in performance. In the dense datasets, users were mapped to initial biclusters that contained a large number of items. In turn, this implied that no lower neighbors were available for exploration and that sibling biclusters also contained large numbers of sparsely related items. In effect that advantage of exploring localized biclusters was eliminated. Exploring one additional level of the lattice quickly filtered out the non-similar biclusters leading to much improved recommendation quality.

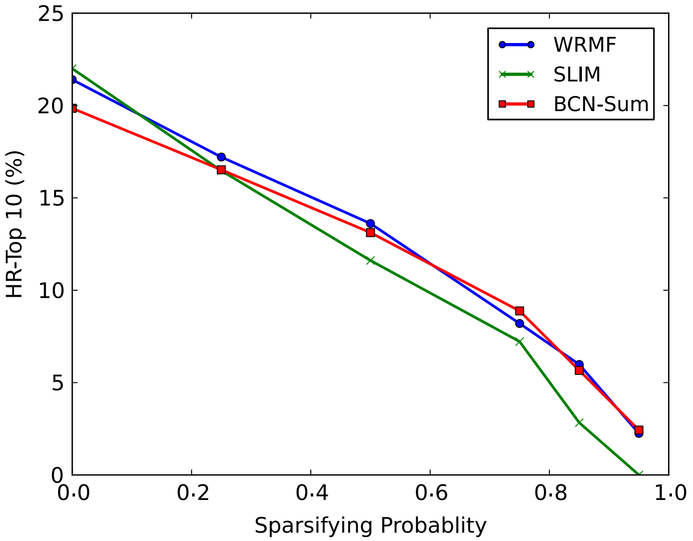


Fig. 2 The effect of sparsity on the recommendation quality

5.4 Effect of data sparsity

In order to further examine the effect of sparsity on BCN, we synthetically “sparsified” the *lastfm* dataset. Sparsity was added by randomly removing transactions with probability p, where p ranges from 0 to 1. Figure 2 illustrates the results of this experiment. We can see that as sparsity in the data increases, the performance of the once dominant *SLIM* method deteriorates at a quicker rate, while *WRMF* and *BCN* methods tend to be on equal footing. Additionally, we also find that (not shown in the chart), *BCN-Sum* takes over as the dominant *BCN* method among the 3 *BCN* methods as sparsity increases. This result reaffirms the observations made in the previous section.

5.5 Incremental scalability

An important aspect of *BCN* is its ability to scale-up incrementally. In other words, as real-world datasets grow dynamically overtime, it is desirable for the total CPU running time to make a single recommendation to scale-up in a reasonable manner. The actual total time to make a single recommendation between *BCN* framework and *SLIM* is not comparable (see Fig. 3a); clearly, *BCN* is orders of magnitude faster as there is no training involved for a single recommendation even as the dataset grows. On the other hand, *SLIM* or other matrix factorization methods would have to re-train their models.

In light of this, experiments were conducted to assess the growth rate of the total CPU time with respect to the growth in the number of users. The *paypal_large* dataset was scaled up incrementally by factors of $2\times$ users while maintaining the same sparsity level; this represents the real-world situation of more users joining the system but the overall distribution of data not changing. The final metric we measured was the ratio of total CPU time to make a single recommendation in the modified datasets to total CPU time in the original dataset. This metric was computed after applying *BCN* with level 1 and level 2 neighbors and the *SLIM* algorithm. Figure 3b illustrates the incremental growth rate of the CPU times. While all the

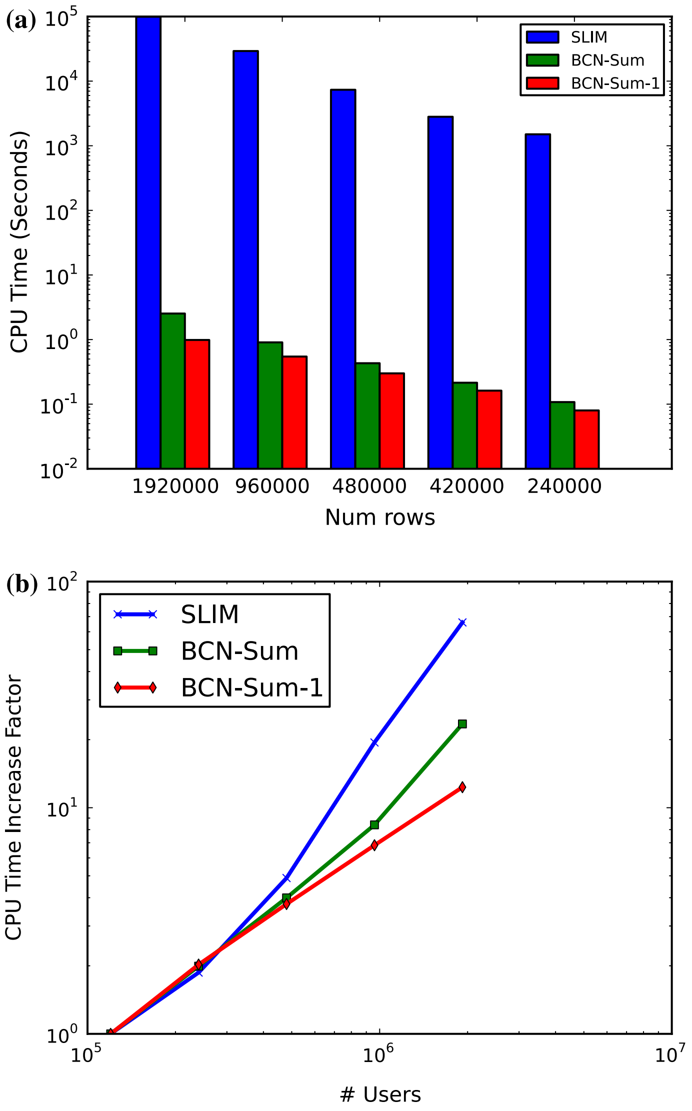


Fig. 3 Experimental results on the scalability of different algorithms on real-world datasets. **a** Actual total CPU time to make a single recommendation. **b** Total CPU time scaling with respect to the user growth

three algorithms tend to have steep growth rates, the *BCN* methods increased at most by a factor of 4.21, while *SLIM* increased by a factor of 10.4 with respect to number of users.

6 Conclusions

We proposed a novel collaborative filtering method based on BCN framework. The framework is designed for implicit feedback data and tested for the top-*n* recommendation task. Our framework takes advantage of the biclustering neighborhood of the smallest bicluster associated with each user to perform localized collaborative filtering and combines it with global similarity measure. Borrowing ideas from the field of Formal Concept Analysis, we build user-specific biclusters that are "more personalized" to the users of interest. Our method is easily scalable and efficiently makes recommendations in large-scale datasets. Our experiments show that this method generates better recommendation than the state-of-the-art algorithms, especially in sparse data. Furthermore, the scalability of the *BCN* method with respect to the number of users and incremental changes in the dataset was illustrated both theoretically and experimentally. The weakness of our *BCN* method is with dense datasets, where we were slightly outperformed by existing *SLIM* method. However, our method remains strong relative to other traditional collaborative filtering methods.

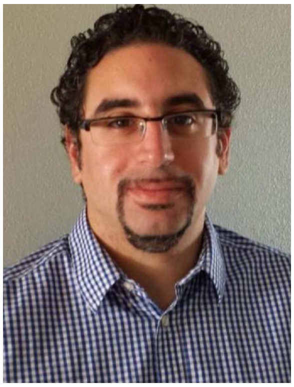
In our future work, we want to address the cold start problem in which a user has only a few interactions. In such cases, the smallest bicluster may not have enough neighbors to form suitable candidates. One possibility is to incorporate additional data sources from an information network such as social media. A second direction is to extend this algorithm to real-valued data by developing novel bicluster similarity measures.

Acknowledgments We would like to thank Santosh Kabbur and Xia Ning for providing us with implementations of *WRMF* and *SLIM* respectively. This work was supported in part by the National Cancer Institute of the National Institutes of Health under Award Number R21CA175974 and the US National Science Foundation Grants IIS-1231742 and IIS-1242304. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH and NSF.

References

1. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
2. Alqadah F, Bader JS, Anand R, Reddy CK (2012) Query-based biclustering using formal concept analysis. In: SDM
3. Alqadah F, Bhatnagar R (2009) Discovering substantial distinctions among incremental bi-clusters. In: SDM'09
4. Alqadah F, Bhatnagar R (2011) Similarity measures in formal concept analysis. Ann Math Artif Intell 61:245–256
5. Berry A, Bordat J-P, Sigayret A (2007) A local approach to concept generation. Ann Math Artif Intell 49:117–136
6. Cantador I, Brusilovsky P, Kuflik T (2011) 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: RecSys 2011
7. Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22:143–177
8. de Franca FO, Ferreira HM, Von Zuben FJ (2007) Applying biclustering to perform collaborative filtering. In: Proceedings of the seventh international conference on intelligent systems design and applications, ISDA '07, pp 421–426
9. Yu K, Xu X, Ester M, Kriegel HP (2003) Feature weighting and instance selection for collaborative filtering. An information-theoretic approach. Knowledge and Information Systems 5(2):201–224
10. Gamter B, Wille R (1999) Formal concept analysis: mathematical foundations. Springer, Berlin
11. George T, Merugu S (2005) A scalable collaborative filtering framework based on co-clustering. In: ICDM '05: Proceedings of the fifth IEEE international conference on data mining. IEEE computer society, pp 625–628
12. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proceedings of the 2008 eighth IEEE international conference on data mining
13. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 426–434
14. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37

15. Leung CW, Chan SC (2006) A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowl Inf Syst 10:357–381
16. Ning X, Karypis G (2011) Slim: sparse linear methods for top-n recommender systems. In: ICDM 2011
17. Oard D, Kim J (1998) Implicit feedback for recommender systems. In: Proceedings of the AAAI workshop on recommender systems, pp 81–83
18. Odibat O, Reddy CK (2014) Efficient mining of discriminative co-clusters from gene expression data. Knowl Inf Syst (KAIS). http://link.springer.com/article/10.1007%2Fs10115-013-0684-0
19. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence
20. Slobodan Vucetic ZO (2005) Collaborative filtering using a regression-based approach. Knowl Inf Syst 7:1–22
21. Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. Adv Artif Intell 4:2–4:2
22. Symeonidis P, Nanopoulos A, Papadopoulos A, Manolopoulos Y (2008) Nearest-biclusters collaborative filtering with constant values. Inf Retr 11:51–75
23. Xue G-R, Lin C, Yang Q, Xi W, Zeng H-J, Yu Y, Chen Z (2005) Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '05, pp 114–121
24. Yu H-F, Hsieh C-J, Si S, Dhillon IS (2013) Parallel matrix factorization for recommender systems. Knowl Inf Syst 1–27. http://link.springer.com/article/10.1007/s10115-013-0682-2
25. Zaki MJ, Ogihara M (1998) Theoretical foundations of association rules. In: 3rd SIGMOD'98 workshop on research issues in data mining and knowledge discovery (DMKD)



Faris Alqadah received his B.S. in computer science in 2005 and his Ph.D. in computer science in 2010, both from the University of Cincinnati. His primary research interests are large-scale data mining, machine learning, network science and data management. He is currently a senior data scientist at PayPal developing classification, regression and clustering algorithms for e-commerce applications that operate efficiently at the terabyte/petabyte scale. Prior to joining PayPal he was a Post-Doctoral Fellow at the Johns Hopkins School of Medicine where he applied his research to High Throughput Biology with applications in drug discovery and genetic network interaction mapping. He has published several papers in leading peer-reviewed conferences and journals including SIGKDD, CIKM, and SDM and was awarded Best Doctoral Forum Poster Award at SDM 2010 and twice nominated for Best Paper Awards at SDM 2010 and 2012.

Chandan K. Reddy is an Associate Professor in the Department of Computer Science at Wayne State University. He received his Ph.D. from Cornell University and M.S. from Michigan State University. His primary research interests are in the areas of data mining and machine learning with applications to healthcare, bioinformatics, and social network analysis. His research is funded by the National Science Foundation, the National Institutes of Health, the Department of Transportation, and the Susan G. Komen for the Cure Foundation. He has published over 50 peer-reviewed articles in leading conferences and journals. He received the Best Application Paper Award at the ACM SIGKDD conference in 2010 and was a finalist of the INFORMS Franz Edelman Award Competition in 2011. He is a senior member of IEEE and a member of ACM.

Junling Hu is the director of Data Mining at Samsung research, where she leads the data mining R&D for content recommendation and advertising for smart TVS. Before joining Samsung, Junling was a Senior Manager of Data Science at PayPal, leading R&D effort building recommender systems and predictive modeling for business units. Prior to PayPal, she led a data mining team at eBay, working on large-scale data mining on structured and unstructured data. She also worked as a manager of data mining group at Robert Bosch research. She is a recipient of CAREER award from National Science Foundation. She received her Ph.D. of computer science from The University of Michigan in 1999, and her M.S. in economics from Florida State University in 1993.

Hatim F. Alqadah received the B.S. degree in computer engineering in 2005, M.S. degrees in electrical engineering and mathematics in 2007 and 2010 respectively, and a Ph.D. degree in electrical engineering in 2011 all from the University of Cincinnati. During his graduate studies he was supported by a Dayton Area Graduate Research Institute (DAGSI) fellowship where he worked on developing sparse regularization techniques for inverse scattering and radar imaging problems. He is currently a National Research Council (NRC) postdoctoral research associate at the U.S. Naval Research Laboratory physical acoustics division. His research interests include 3D electromagnetic/acoustic image reconstruction, model-based signal processing, and computer vision.