Inf Retrieval (2008) 11:51–75
DOI 10.1007/s10791-007-9038-4

Nearest-biclusters collaborative filtering based on constant and coherent values

Panagiotis Symeonidis · Alexandros Nanopoulos · Apostolos N. Papadopoulos · Yannis Manolopoulos

Received: 8 August 2006 / Accepted: 14 November 2007 / Published online: 1 December 2007

Abstract Collaborative Filtering (CF) Systems have been studied extensively for more than a decade to confront the "information overload" problem. Nearest-neighbor CF is based either on similarities between users or between items, to form a neighborhood of users or items, respectively. Recent research has tried to combine the two aforementioned approaches to improve effectiveness. Traditional clustering approaches (*k*-means or hierarchical clustering) has been also used to speed up the recommendation process. In this paper, we use biclustering to disclose this duality between users and items, by grouping them in both dimensions simultaneously. We propose a novel nearest-biclusters algorithm, which uses a new similarity measure that achieves partial matching of users' preferences. We apply nearest-biclusters in combination with two different types of biclustering algorithms—Bimax and xMotif—for constant and coherent biclustering, respectively. Extensive performance evaluation results in three real-life data sets are provided, which show that the proposed method improves substantially the performance of the CF process.

Keywords Nearest neighbor · Collaborative filtering · Biclustering · Clustering

This work is an extension of a preliminary version presented in WebKDD 2006 Workshop. In this paper, extensive material has been added, which compares new classes of biclustering algorithms over more real data sets. Moreover, model-based approaches (in addition to the memory-based approaches) are under examination, too.

P. Symeonidis (✉) · A. Nanopoulos · A. N. Papadopoulos · Y. Manolopoulos
Department of Informatics, Aristotle University, Thessaloniki 54124, Greece
e-mail: symeon@delab.csd.auth.gr

A. Nanopoulos
e-mail: alex@delab.csd.auth.gr

A. N. Papadopoulos
e-mail: apostol@delab.csd.auth.gr

Y. Manolopoulos
e-mail: manolopo@delab.csd.auth.gr

1 Introduction

Information Filtering has become a necessary technology to attack the "information overload" problem. In our everyday experience, while searching on a topic (e.g., products, movies, etc.), we often rely on suggestions by others, more experienced on it. In the Web, however, the plethora of available suggestions pauses difficulties in detecting the trustworthy ones. The solution is to shift from individual to collective suggestions. Collaborative Filtering (CF) applies information retrieval and data mining techniques to provide recommendations based on suggestions of users with similar preferences. CF is a very popular method in recommender systems and e-commerce applications.

Two families of CF algorithms have been studied extensively in the literature:

- Memory-based algorithms, which perform the computation on the entire database to identify the top *k* similar users to a target user (Xue et al. 2005).
- Model-based algorithms, which recommend items to a target user after they have first computed a model of predefined classes of users based on their rating patterns (Xue et al. 2005).

Regarding memory-based algorithms, a well-studied approach in research is user-based (UB) CF (Breese et al. 1998; Resnick et al. 1994), which forms neighborhoods based on similarities between users. In particular, for a test user, UB employs users' similarities to form a neighborhood of his nearest users. Then, UB recommends to the test user, the most frequent items in the formed neighborhood. Recently, another algorithm was proposed (Sarwar et al. 2001), denoted as item-based (IB) CF, which forms item neighborhoods based on similarities between items. IB is usually assigned to the memory-based family.[1] Both UB and IB are one-side approaches, in the sense that they examine similarities either only between users or only between items, respectively. This way, they ignore the clear duality that exists between users and items. Furthermore, UB and IB algorithms cannot detect partial matching of preferences, because their similarity measures consider the entire set of items or users, respectively. However, two users may share similar preferences only for a subset of items.

As an example, consider two users that share similar preferences for science-fiction books and differentiate in all other kinds of literature. In this case, their partial matching for science-fiction, which can help to provide useful recommendations for this kind of books, will be missed by existing nearest neighbor CF approaches. The reason is that these approaches measure similarity between two users with respect to the entire set of items. Thus, they miss their partial matching, since the differences in the rest items prevail over the subset of items in which their preferences match. Analogous reasoning applies for the IB approaches. Content-Based Filtering (CBF) (Balabanovic and Shoham 1997; Salter and Antonopoulos 2006; Melville et al. 2002) can help towards partial matching (such approaches are detailed in Sect. 2). However, these approaches exploit additional information about the content of items (e.g., in the form of features), whereas in this paper we focus on CF based solely on ratings.

To disclose the duality between users and items, we propose the generation of groups of users *and* items at the same time. The simultaneous clustering of users and items discovers *biclusters*, which correspond to groups of users which exhibit highly correlated ratings on groups of items. Biclusters allow the computation of similarity between a test

[1] Since, in its off-line part, IB learns relationships between items according a model, it could be considered as a model-based algorithm as well.

user and a bicluster *only* on the items that are included in the bicluster. Thus, partial matching of preferences is taken into account. Moreover, a user can be matched with several nearest biclusters, to receive recommendations that cover the range of his various preferences.

Regarding Model-based algorithms, once they have build the model, they present good scalability. However, they have the overhead to build and update the model, and they cannot cover as diverse user ranges as the nearest-neighbor algorithms do (Xue et al. 2005). The reason is that model-based approaches group users into small number of classes, resulting to reduced range of variation for the users' preferences. Thus, it is more difficult for a user with eclectic taste compared with a small number of classes to get recommendations. In contrast, nearest-neighbor algorithms provide good recommendations to users with eclectic tastes. Clustering is probably the most widely method used in model-based approaches (Xue et al. 2005; Hofmann 2004; Kohrs and Merialdo 1998). It finds groups of users or items that have similar preferences. Traditional clustering approaches, such as *k*-means and hierarchical clustering, put each user in exactly one cluster. However, the fact that a user usually has various different preferences, should be taken into account for the process of assigning him to clusters. Therefore, such a user should be included in more than one clusters. This cannot be achieved by the traditional clustering algorithms, which place each user/item in exactly one cluster.

To include a user in more than one clusters, we allow a degree of overlap between biclusters. Thus, if a user presents different item preferences, by using overlapping biclusters, he can be included in more clusters in order to cover all his different preferences.

Several biclustering algorithms have been proposed in the literature, each of which has strengths and weaknesses depending on different scenarios. For this reason, we try two different approaches and choose an algorithm that delivers the best results for purposes of CF. Two main bicluster classes have been proposed: (a) biclusters with constant values and (b) biclusters with coherent values. The first category looks for subsets of rows and subsets of columns with constant values, while the second is interested for biclusters with coherent values. In this paper, we examine the effectiveness and efficiency of both schemes. The contributions of this paper are summarized as follows:

- We introduce the application of two classes of biclustering algorithms (one with constant and one with coherent values) for the purposes of CF. These algorithms perform simultaneous clustering of users and items to provide more accurate recommendations. In contrast to previous related work (George and Merugu 2005), that used biclusters only to improve the scalability of CF, we follow an entirely different approach and aim to additionally improve the accuracy of CF.
- We propose a novel nearest-biclusters algorithm, which uses a new similarity measure that achieves partial matching of users' preferences.
- We compare the impact of two different bicluster algorithms.
- Our extensive experimental results in three real-life data sets illustrate the effectiveness and efficiency of the proposed algorithm over existing state-of-the-art approaches.

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Sect. 3 contains the analysis of the CF issues. The proposed approach is described in Sect. 4. Experimental results are given in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Related work

Collaborative Filtering (CF) is based on common users preferences. In 1994, the GroupLens system (Resnick et al. 1994) proposed an CF algorithm, known as user-based (UB) CF, because it employs users' similarities for the formation of the neighborhood of nearest users. Since then, many improvements of user-based algorithm have been suggested, e.g. (Breese et al. 1998; Herlocker et al. 1999; Mobasher et al. 2001). Herlocker et al. (2002) proposed the latest variation, by weighting the significance of nearest neighbors based on the number of common ratings between them and the target user. According to Significance Weighting (SW), if variable *sim* denotes the similarity between two users, then it can be weighted based on a γ parameter, which is a threshold for the required number of co-rated items between the two users. The similarity *sim* between the two users is multiplied with a weighting factor $\frac{\max(c,\gamma)}{\gamma}$, where c is the number of co-rated items. In 2001, another CF algorithm was proposed. It is based on the items' similarities for a neighborhood generation (Sarwar et al. 2001; Karypis 2001; Deshpande and Karypis 2004). This approach is known as item-based CF algorithm, because it employs items' similarities for the formation of the neighborhood of nearest users. Additionally, Lemire and Maclachlan (Lemire and Maclachlan 2005) proposed three related slope one schemes with predictors of the form $f(x) = x + b$, which precompute the average difference between the ratings of one item and another for users who rated both. Finally, Wang et al. (2006) proposed the Similarity Fusion (SF) between the UB and IB methods, using also data from a third source (ratings of other similar users on other similar items). In particular, the final rating predictions are estimated by fusing predictions from three sources: (i) predictions based on UB, (ii) predictions based on IB and, (iii) predictions based on data from similar users rating other similar items.

Compared with memory-based algorithms, the basic idea of model-based algorithms is to cluster items or training users into classes explicitly, and predict ratings of a test user by using the ratings of classes that fit best with the test user (Breese et al. 1998; Hofmann 2004; Jin et al. 2006). Several different probabilistic models have been proposed. Hofmann (2004) proposed a statistical modeling technique that introduces latent class variables to discover user communities and prototypical interest profiles. Jin et al. (2006) compared five different probabilistic mixtures models. They claimed that the Decoupled Model (DM) outperforms the other models, because it satisfies all three desired properties for a mixture model: (i) separate clustering of users and items (ii) flexibility for a user/item to be in multiple clusters and, (iii) decoupling of user preferences from its rating patterns. There are also other model-based algorithms which are based on pure cluster-based methods. For instance, Kohrs and Merialdo (1998) proposed clustering methods for CF. Xue et al. (2005) proposed scalable CF using cluster-based smoothing. Once clusters are created, predictions for a test user can be made by averaging the opinions of the other users in the cluster that he participates. Some clustering techniques represent each user with partial participation in several clusters. The prediction is then an average across the clusters, weighted by the degree of participation (Xue et al. 2005). In contrast, traditional clustering approaches such as k-means and hierarchical clustering put each user in exactly one cluster.

In contrast to CF algorithms, CBF assumes that each user operates independently. As a result, CBF exploits solely information derived from document or item features (e.g., terms or attributes). There have been several attempts to combine CBF with CF. The

Fab System (Balabanovic and Shoham 1997), measures similarity between users after first computing a profile for each user. This process reverses the CinemaScreen System (Salter and Antonopoulos 2006) which runs CBF on the results of CF. Melville et al. (2002) used a bayesian content-based predictor to enhance existing user data, and then to provide personalized suggestions though collaborative filtering. In this paper, we focus on the CF systems and the information derived solely from the user-item ratings matrix.

The concept of biclustering has been used in (Mirkin1996) to perform grouping in a matrix by using both rows and columns. However, biclustering has been used previously in (Hartigan 1972) under the name *direct clustering*. Recently, biclustering (also known as *co-clustering*, *two-sided clustering*, *two-way clustering*) has been exploited by many researchers in diverse scientific fields, towards the discovery of useful knowledge (Cheng and Church 2000; Dhillon 2001; Dhillon and Mallela 2003; Long et al. 2005; Murali and Kasif 2003). One of these fields is bioinformatics, and more specifically, microarray data analysis. The results of each microarray experiment are represented as a data matrix, with different samples as rows and different genes as columns. Among the proposed biclustering algorithms we highlight the following: (i) Cheng and Church's algorithm (Cheng and Church 2000) which is based on a mean squared residue score, (ii) the Bimax algorithm, an exact biclustering algorithm based on a divide-and-conquer strategy, that is capable of finding all maximal bicliques in a corresponding graph-based matrix representation (Prelic et al. 2006), and (iii) the xMOtif algorihm, an iterative search method which seeks biclusters by extracting conserved gene expression motifs from gene expression data (Murali and Kasif 2003). Finally, Shafiei and Milios (2005) proposed a co-clustering model able to find co-clusters in an input-data matrix, where some data points could belong to more than one bicluster. This approach does not focus on recommender systems and CF in particular, thus can be treated as a model for creating bi-clusters.

In the CF area, Madeira and Oliveira (2004) have reported in their survey, the existence of works that have used two-sided clustering. In these models (Ungar and Foster 1998; Hofmann and Puzicha 1999), there is a hidden variable for each user and item, respectively, that represents the cluster of that user or item. For each user-item pair, there is a variable that denotes their relation. The existence of the relation depends on the cluster of the person, and the cluster of item, hence the notion of two-sided clustering. These are latent class models using statistical estimation of the model parameters and clustering is performed separately for users and items. Finally, George and Merugy (2005) used co-clustering in CF to build an efficient real-time CF framework. In particular, they proposed a method for incremental update of the biclusters as new users and new ratings are being continuously updated. As they claimed, their algorithm achieves the same accuracy as the other CF algorithms. In contrast, our approach is based on the application of specific biclustering algorithms[2] that perform simultaneous clustering of users and items to provide more accurate recommendations. We test our method, with other state-of-the-art algorithms (Herlocker and Shoham 2002; Wang et al. 2006; Jin et al. 2006). The experimental results demonstrate that our algorithm outperforms significantly the other methods in terms of accuracy.

[2] For implementation issues, we use the Bimax and xMotif biclustering algorithms, however any other algorithm can be used equally well, as our approach is independent of the specific biclustering algorithm that is used.

3 Examined issues

In this section, we provide details for the issues we examine about CF algorithms and simple clustering. Table 1 summarizes the symbols that are used in the sequel.

Scalability: Scalability is important, because in real-world applications the number of users/items is very large. As the number of users/items grows, CF algorithms face performance problems. There are many model-based approaches that have been used to confront this scalability problem. These approaches first create a model of users' ratings. hierarchical clustering or k-means are examples of traditional clustering algorithms, which have been extensively used to create clusters of items or users. CF algorithms should be evaluated in terms of their responding time in providing recommendations.

Similarity measure: The most extensively used similarity measures are based on correlation and cosine-similarity (Herlocker et al. 2002; McLauglin and Oliveria 2004; Breese et al. 1998; Sarwar et al. 2001). Specifically, user-based CF algorithms mainly use Pearson's Correlation (Eq. 1), whereas for item-based CF algorithms, the Adjusted Cosine Measure is preferred (Eq. 2). The Adjusted Cosine Measure is a variation of the simple cosine formula, that normalizes bias from subjective ratings of different users. As default options, for user-based CF we use the Pearson Correlation (McLauglin and Oliveria 2004), whereas for item-based we use the Adjusted Cosine Similarity (Sarwar et al. 2001), because they presented the best behavior overall.

Table 1 Symbols and definitions

Symbol	Definition
k	Number of nearest neighbors or biclusters
N	Size of recommendation list
P_τ	Threshold for positive ratings
$\mathcal{I}$	Domain of all items
$\mathcal{U}$	Domain of all users
u, v	Some users
i, j	Some items
I_u	Set of items rated by user u
U_i	Set of users who rated item i
$r_{u,i}$	The rating of user u on item i
$\bar{r}_u$	Mean rating value for user u
$\bar{r}_i$	Mean rating value for item i
n	Minimum allowed number of users in a bicluster
m	Minimum allowed number of items in a bicluster
$\mathcal{B}$	Set of all biclusters
b	A bicluster
I_b	Set of items of bicluster b
U_b	Set of users of bicluster b
t	Total number of created biclusters
n_s	Number of users which act as seeds
n_u	Maximum number of users which can be selected for each seed
n_i	Maximum number of items which can be selected for each seed
p-Value	Percentage of a bicluster's homogeneity

$$\mathrm{sim}(u,v) = \frac{\sum_{\forall i \in S}(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{\forall i \in S}(r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_{\forall i \in S}(r_{v,i} - \overline{r}_v)^2}}, S = I_u \cap I_v. \tag{1}$$

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

Neighborhood size: The number, k, of nearest neighbors used for the neighborhood formation is important, because it can affect substantially the system's accuracy. In most related works (Herlocker et al. 1999; Sarwar et al. 2000, 2001; Breese et al. 1998), k has been examined in the range of values between 10 and 100. The optimum k depends on the data characteristics (e.g., sparsity). Therefore, CF algorithms should be evaluated against varying k, in order to tune it.

Positive rating threshold: Recommendation for a test user is performed by generating the top-N list of items that appear most frequently in his formed neighborhood (this method is denoted as Most-Frequent item-recommendation). Nevertheless, it is evident that recommendations should be "positive", as it is not success to recommend an item that will be rated with, e.g., 1 in 1–5 scale. Thus, "negatively" rated items should not contribute to the increase of accuracy. We use a rating-threshold, P_τ, to recommended items whose rating is not less than this value. If we do not use a P_τ value, then the results become misleading.

Training/Test data size: There is a clear dependence between the training set's size and the accuracy of CF algorithms (Sarwar et al. 2001, 2000). Through our experimental study we verified this conclusion. Though most related research uses a size around 80%, there exist works that use significantly smaller sizes (McLauglin and Oliveria 2004). Therefore, CF algorithms should be evaluated against varying training data sizes.

Recommendation list's size: The size, N, of the recommendation list poses a tradeoff: With increasing N, the absolute number of relevant items (i.e., recall) is expected to increase, but their ratio to the total size of the recommendation list (i.e., precision) is expected to decrease. (Recall and precision metrics are detailed in the following.) In related work (Karypis 2001; Sarwar et al. 2001), N usually takes values between 10 and 50.

Evaluation Metrics: Our performance study is mainly focused on widely accepted metrics from information retrieval. For a test user that receives a top-N recommendation list, let R denote the number of *relevant recommended items* (the items of the top-N list that are rated higher than P_τ by the test user). We define the following:

- *Precision* is the ratio of R to N.
- *Recall* is the ratio of R to the total number of relevant items for the test user (all items rated higher than P_τ by him).

Notice that with the previous definitions, when an item in the top-N list is not rated at all by the test user, we consider it as *irrelevant* and it counts negatively to precision (as we divide by N). Moreover, this precision metric (where non-rated items counted as non relevant) is also proposed by (McLauglin and Oliveria 2004) denoted as modified precision. In the following we also use F_1, because it combines both the previous metrics:

$$F_1 = 2 \cdot \text{recall} \cdot \text{precision}/(\text{recall} + \text{precision}).$$

F_1 is used because precision and recall follow a reverse behavior with respect to the size of the top-*N* list. By increasing *N*, recall increases but precision decreases, and vise versa. Therefore, F_1 considers both of them at the same time.

On the other hand, several other metrics have been used for the evaluation of CF algorithms, for instance the Mean Absolute Error (MAE) or the Receiving Operating Characteristic (ROC) curve (Herlocker et al. 2002, 2004). In order to compare our work with other algorithms, we also include experiments with MAE, which represents the absolute differences between the real and the predicted values. In particular, MAE measures, for all test users u_t, the average absolute deviation of predictions between their actual ratings $r_{u_t,i}$ and the predicted ratings $\hat{r}_{u_t,i}$. The definition of MAE is given in Eq. 3:

$$\text{MAE} = \frac{\sum_{u_t,i} |r_{u_t,i} - \hat{r}_{u_t,i}|}{L}, \tag{3}$$

where L denotes the total number of examined ratings.

From our experimental study (Sect. 5) we understood that MAE is able to characterize the accuracy of prediction, but is not indicative for the accuracy of recommendation. Since in real-world recommender systems the experience of users mainly depends on the accuracy of recommendation, MAE may not be the preferred measure.

4 Nearest-biclusters approach

4.1 Outline of the proposed approach

Our approach consists of three stages.

- *Stage 1*: the data preprocessing/discretization step (optional).
- *Stage 2*: the biclustering process.
- *Stage 3*: the nearest-biclusters algorithm.

The proposed approach, initially, applies a data preprocessing/discretization step, which is optional. The motivation is to preserve only the positive ratings. Consequently, we proceed to the biclustering process, where we create groups consisting of users and items in a single-step. Finally, we implement the *k* nearest-biclusters algorithm. We calculate similarity between each test user and the generated bicluster. Thus, we create the test users' neighborhood, consisted of the *k* nearest biclusters. Then, we provide for each test user a top-*N* recommendation list based on the most frequent items in his neighborhood.

To ease the discussion, we will use the running example illustrated in Fig. 1, where I_{1-7} are items and U_{1-9} are users. As shown, the example data set is divided into training and test set. The null cells (no rating) are presented with dash.

4.2 Data preprocessing/discretization step

In our approach so far, we assume that all users express their rating behavior similarly on a common scale. However, it is known that different users may associate subjectively different preferences with specific ratings. For instance, a 4 in a rating scale (Balabanovic and

Fig. 1 Running example: (**a**) training set; (**b**) test set

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
U_1	5	-	2	-	1	-	-
U_2	2	-	4	1	4	3	-
U_3	4	-	2	-	2	-	5
U_4	-	3	1	4	-	5	2
U_5	-	2	4	2	5	1	-
U_6	5	1	-	1	-	-	3
U_7	-	2	5	-	4	1	-
U_8	1	4	-	5	4	3	-

(a)

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
U_9	5	-	4	-	1	-	2

(b)

Shoham1997; Barkow et al. 2006; Breese et al. 1998; Cheng and Church 2000; Deshpande and Karypis 2004) may mean different things for different people. Similarly, some items may get higher ratings than their real value, because they have been rated by users with positive intent. In memory-based methods, this is taken into account in the similarity measures, such as Pearson Coefficient in UB CF (Herlocker et al. 1999) and Adjusted Cosine Similarity in IB CF (Sarwar et al. 2001). Particularly, in Eqs. 1 and 2, this is taken into account by subtracting the mean rating value for each user/item, respectively. To attain the same result in our approach, similarly to (Wang et al. 2006), we normalize the user-item rating matrix as follows: for each rating of a user, we subtract the user's mean rating value. (In our running example, this preprocessing step is presumed.)

According to the positive rating threshold, introduced in Sect. 3, recommendations should be "positive", as it is meaningless to recommend an item that will be rated with, e.g., 1 in 1–5 scale. Thus, "negatively" rated items should not contribute to the increase of accuracy. This is the reason that we are interested only in positive ratings, as shown in Fig. 2.

Furthermore, as biclustering groups items and users simultaneously, it allows to identify sets of users sharing common preferences across subsets of items. In our approach, the main goal is to find subsets of users that have rated positively (above P_τ rating threshold)

Fig. 2 Training set with rating values $\geq P_\tau$

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
U_1	5	-	-	-	-	-	-
U_2	-	-	4	-	4	3	-
U_3	4	-	-	-	-	-	5
U_4	-	3	-	4	-	5	-
U_5	-	-	4	-	5	-	-
U_6	5	-	-	-	-	-	3
U_7	-	-	5	-	4	-	-
U_8	-	4	-	5	4	3	-

Fig. 3 Binary discretization of the training set

	I_1	I_2	I_3	I_4	I_5	I_6	I_7
U_1	1	0	0	0	0	0	0
U_2	0	0	1	0	1	1	0
U_3	1	0	0	0	0	0	1
U_4	0	1	0	1	0	1	0
U_5	0	0	1	0	1	0	0
U_6	1	0	0	0	0	0	1
U_7	0	0	1	0	1	0	0
U_8	0	1	0	1	1	1	0

items. Therefore, the problem can be discretized to binary values by setting as discretization threshold the P_τ rating threshold. The discretized data are shown in Fig. 3.

Discretization of data is optional and can be omitted, in case we use a biclustering algorithm which discovers biclusters with coherent values on both users and items. In our case, as it is shown in the next section, we use Bimax algorithm which finds clusters with constant values and the discretization step is required, and we also use xMotif algorithm, which finds biclusters based on coherent values, and this step can be omitted.

4.3 The biclustering process

The biclustering process on a data matrix involves the determination of a set of clusters taking into account both rows and columns. Each bicluster is defined on a subset of rows and a subset of columns. Moreover, two biclusters may overlap, which means that several rows or columns of the matrix may participate in multiple biclusters. Another important characteristic of biclusters is that each bicluster should be maximal, i.e., it should not be fully contained in another determined bicluster.

For the biclustering step, we have adopted two different algorithms as representatives of the main two bicluster classes that have been proposed: (a) biclusters with constant values and (b) biclusters with coherent values. The first category looks for subsets of rows and subsets of columns with constant values, while the second is interested for biclusters with coherent values.

The first algorithm is a simple binary inclusion-maximal biclustering algorithm denoted as Bimax (Prelic et al. 2006). It is an exact biclustering algorithm based on a divide-and-conquer strategy that is capable of finding all maximal biclusters in a corresponding graph-based matrix representation.

The second algorithm is the xMotif algorihm, an iterative search method which seeks biclusters by extracting conserved gene expression xMotifs from gene expression data [Murali and Kasif 2003]. An xMotif is a conserved gene expression motif followed from a subset of genes that is simultaneously conserved across a subset of samples.

According to (Prelic et al. 2006), the run-time complexity of Bimax is $O(nm\beta)$, where n is the number of users, m is the number of items, and β is the number of the resulting biclusters. Also, according to (Murali and Kasif 2003], for xMotif, the run-time complexity to identify the largest motif, is $O(nm^{O(\log(1/\alpha)\ \log(1/\beta))})$, where α and β are users parameters in the range [0, 1] that conserve the size of motif. For more information, see the original papers (Prelic et al. 2006; Murali and Kasif 2003). However, have in mind that in our method, both algorithms are executed off-line. Thus, these run-time complexities are adequate.

4.3.1 The Bimax algorithm

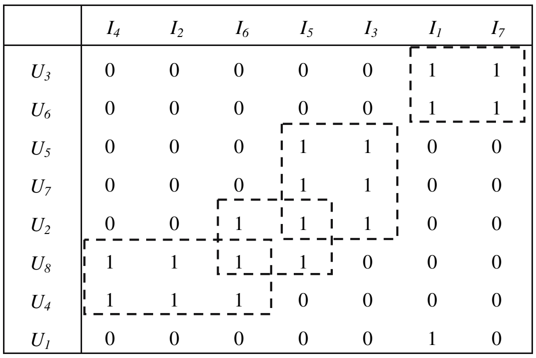
For the Bimax algorithm, a bicluster $b(U_b, I_b)$ corresponds to a subset of users $U_b \subseteq \mathcal{U}$ that jointly present positively rating behavior across a subset of items $I_b \subseteq \mathcal{I}$. In other words, the pair (U_b, I_b) defines a submatrix for which all elements equal to 1.

The main goal of the Bimax algorithm is to find all biclusters that are *inclusion-maximal*, i.e., that are not entirely contained in any other bicluster. The required input to Bimax is the minimum number of users and the minimum number of items in a bicluster. It is obvious that the Bimax algorithm finds a large number of overlapping biclusters. To avoid this we can perform a secondary filtering procedure to reduce this number to the desired overlapping degree. In particular, a criterion to remove overlap is to maintain each time those biclusters that are larger in size, by giving also a percentage of the desired overlap.

In Fig. 4, we have applied the Bimax algorithm to the running example. Four biclusters are found (depicted with dashed rectangles), with minimum number of users equal to 2 (i.e., $|U_b| \geq 2$) and the minimum number of items equal to 2 (i.e., $|I_b| \geq 2$). These bilcusters are summarized as follows:

$$\begin{array}{lll}
b_1: & U_{b_1} = \{U_3, U_6\}, & I_{b_1} = \{I_1, I_7\} \\
b_2: & U_{b_2} = \{U_5, U_7, U_2\}, & I_{b_2} = \{I_5, I_3\} \\
b_3: & U_{b_3} = \{U_2, U_8\}, & I_{b_3} = \{I_6, I_5\} \\
b_4: & U_{b_4} = \{U_8, U_4\}, & I_{b_4} = \{I_4, I_2, I_6\}
\end{array}$$

Notice that there is overlap between biclusters, specifically between biclusters 2 and 3 in item I_5. Also, we have an overlap between biclusters 3 and 4 in item I_6. We can allow this overlapping (it reaches 16, 6%) or we can forbid it. If we forbid it, then we will abolish the existence of the third bicluster because it is smaller than the other two. In order not to miss important biclusters, we allow overlapping. However, overlapping introduces a trade-off: (a) with few biclusters the effectiveness reduces, as several biclusters may be missed; (b) with a high number of biclusters efficiency reduces; as we have to examine many possible matchings. In our experimental results we show the tuning of the allowed overlapping factor.



	I_4	I_2	I_6	I_5	I_3	I_1	I_7
U_3	0	0	0	0	0	1	1
U_6	0	0	0	0	0	1	1
U_5	0	0	0	1	1	0	0
U_7	0	0	0	1	1	0	0
U_2	0	0	1	1	1	0	0
U_8	1	1	1	1	0	0	0
U_4	1	1	1	0	0	0	0
U_1	0	0	0	0	0	1	0

Fig. 4 Applying the Bimax algorithm to the training set

4.3.2 The xMotif algorithm

For the xMotif algorithm, given a set of users whose ratings are across a set of items and the user-defined parameters $0 < \alpha$, $\beta < 1$, a bicluster $b(U_b, I_b)$ corresponds to a subset of users $U_b \subseteq \mathcal{U}$ that jointly present coherent rating behavior across a subset of items $I_b \subseteq \mathcal{I}$, which also satisfies the following conditions:

- *Size*: the number of users in U_b is at least an
- α-fraction of all the number of users,
- *Conservation*: every item in I_b is conserved across all the users in U_b, i.e., the item is in the same state in all the users in U_b, and
- *Maximality*: for every item not in I_b, the item is conserved in at most a β-fraction of the users in U_b.

The maximality condition enforces a balance between the number of items in the motif and the number of users matching the motif. If we add a extra item to the motif, then the number of users matching the new motif will decrease by a fraction of at least β, a cost we may not be willing to pay. Given this definition, the user-item data matrix may contain many xmotifs. We can allow a user or item to appear in more than one motif, modelling the possibility that the user may have different preferences or an item can belong in many genres. In Fig. 5, we have applied the xMotif algorithm to the running example (without the data discretization stage). There are four biclusters consisting of at least of 2 users and 2 items.

These bilcusters are summarized as follows:

$$\begin{array}{lll}
b_1: & U_{b_1} = \{U_3, U_6, U_1\}, & I_{b_1} = \{I_1, I_7\} \\
b_2: & U_{b_2} = \{U_5, U_7, U_2, U_8\}, & I_{b_2} = \{I_5, I_3\} \\
b_3: & U_{b_3} = \{U_2, U_8\}, & I_{b_3} = \{I_6, I_5, I_3\} \\
b_4: & U_{b_4} = \{U_8, U_4\}, & I_{b_4} = \{I_4, I_2, I_6, I_5\}
\end{array}$$

The xMotif algorithm permits the generation of xMotifs (biclusters) which can embody users or items in a more flexible and complete way. Firstly, it finds biclusters with coherent values. Secondly, in contrast to Bimax, which defines a submatrix for which all elements are equal to 1 (above the P_τ threshold), xMotif algorithm allows a user or item to be included in a bicluster, even if there exists a fraction of ratings which cannot not be defined as interesting ones (ratings under the P_τ threshold or null values). The aforementioned characteristic is controlled by a p-value parameter which is user-defined. This p-value

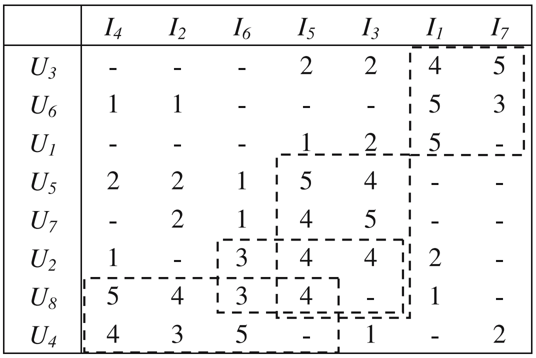


Fig. 5 Applying the xMotif algorithm to the training set

	I_4	I_2	I_6	I_5	I_3	I_1	I_7
U_3	-	-	-	2	2	4	5
U_6	1	1	-	-	-	5	3
U_1	-	-	-	1	2	5	-
U_5	2	2	1	5	4	-	-
U_7	-	2	1	4	5	-	-
U_2	1	-	3	4	4	2	-
U_8	5	4	3	4	-	1	-
U_4	4	3	5	-	1	-	2

parameter defines how a range of the expression values of a bicluster is statistically significant (Murali and Kasif 2003). Thus, through the p-value parameter we can control the generation of biclusters which are homogeneous or biclusters with more diversity. The required input for the xMotif algorithm is an initial n_s number of users which will act as seeds for the biclusters. For each randomly selected seed, we select n_u sets of users with n_i number of items. This means that the number of biclusters is defined as an input parameter ($n_s * n_u$).

Finally, as it is shown in Fig. 5, there is overlapping between biclusters. We can again allow this overlapping or we can forbid it as in the Bimax case. We have to mention that the xMotif algorithm does not create *inclusion-maximal* biclusters. But this can be avoided also through the tuning of the allowed overlapping factor.

4.4 The nearest-biclusters algorithm

In order to provide recommendations, we have to find the biclusters containing users with preferences that have strong partial similarity with the test user. This stage is executed on-line and consists of two basic operations:

- The formation of test users' neighborhood, i.e., to find the k nearest biclusters.
- The generation of the top-N recommendation list.

To find the k nearest biclusters, we measure the similarity of the test user against each of the biclusters. The central difference with past work is that we are interested for the similarity of test user and a bicluster *only* on the items that are included in the bicluster and not on all items that he has rated positively (above P_t threshold).[3] As described, this allows for the detection of partial similarities. The similarity between a test user u and each bicluster b is calculated as given by Eq. 4:

$$\mathrm{sim}(u,b) = \frac{|I_u \bigcap I_b|}{|I_b|} \tag{4}$$

It is obvious that similarity values range between [0, 1].

In the next phase, we proceed to the generation of the top-N recommendation list. For this purpose, we have to find the appearance frequency of each item and recommend the N most frequent ones. In Eq. 5, we define as *Weighted Frequency* (WF) of an item i in a bicluster b, the product between $|U_b|$ and the similarity $sim(u, b)$. This way, we weight the contribution of each bicluster with its size in addition to its similarity with the test user:

$$WF(i,b) = sim(u,b) * |U_b| \tag{5}$$

Finally, we apply the Most Frequent Item Recommendation (proposing those items that appear most frequently in the test user's formed neighborhood). Thus, we add the item weighted frequencies, we sort them, and propose the top-N items in the constructed list, which is customized to each test user preferences.

In our running example, for the Bimax case, assume that we keep all four biclusters (allow overlapping) and we are interested for 2 nearest biclusters ($k = 2$). As it is shown, U_9 has rated positively only two items (I_1, I_3). So, his similarity with each of the biclusters is (0.5, 0.5, 0, 0), respectively. Thus, test user's nearest neighbors come from the first two

[3] In future work we plan to investigate the role of negatively rated items.

biclusters, and the recommended items for him will be items I_7 and I_5. For the xMotif algorithm, taking into account the aforementioned assumptions, the similarity of U_9 with each of the biclusters is 0.5, 0.5, 0.33, 0, respectively. As we can see, the similarity values are different than those resulted by Bimax.

5 Experimental evaluation

In this section, we compare the performance of our nearest-bicluster CF approach, using the xMotif (denoted as NBCF-X) and Bimax (denoted as NBCF-B) algorithms, against existing CF algorithms such as user-based algorithm denoted as UB (Breese et al. 1998) and item-based algorithm denoted as IB (Sarwar et al. 2001). For comparison reasons, we also used a clustering-based CF representative denoted as CBCF (Xue et al. 2005). All algorithms were implemented in C++ and their parameters were tuned according to the original papers. In particular for the generation of biclusters, we used the Biclustering Analysis Toolbox (BicAT) (Barkow et al. 2006). Our experiments were performed on a 3 GHz Pentium IV, with 1 GB of memory, running Windows XP.

The factors that are treated as parameters, are the following: the neighborhood size (k, default value 20), the size of the recommendation list (N, default value 20), and the size of training set (default value 75%). The test set consists of all remaining users, i.e., those not in the training set. Users in the test set are the basis for measuring the examined metrics (precision, recall, and F_1). Each test user's ratings have been split into *observed* items and *held-out* items. The ratings of *observed* items (default is 5) are input for the similarity measures between the test user and the training users. The *held-out* items are used for measuring precision, recall and F_1. Each experiment has been repeated 20 times (each time a different training set is selected at random) and the presented measurements, based on two-tailed t-test, are statistically significant at the 0.05 level.

We performed experiments with three real-life data sets that have been used as benchmark in prior work. In particular, we examined the following data sets: (i) the MovieLens 100 K data set with 100,000 ratings (1–5 scale) assigned by 943 users on 1,682 movies, denoted as 100 K data set, (ii) the Movielens 1 M data set with about 1 million ratings for 3,592 movies by 6,040 users, denoted 1 M data set and (ii) the EachMovie data set where we extracted a subset of 2,000 users with ratings (1–6) scale on 1,628 movies denoted as EachMovie.

For the CBCF algorithm, which is based on k-means algorithm, it is required an input parameter for specifying the desired number of clusters of users. Thus, CBCF algorithm evaluated against different desired number of clusters: 250 clusters, 500 cluster, 750 clusters and 1,000 clusters. For the 100 K data set, the best results for F_1 metric was with 500 clusters. For the 1 M data set, the input parameter for CBCF was set to 750 clusters, while for the EachMovie data set was set to 500. Finally, since we have normalized our data sets by subtracting the user's mean rating value, P_τ is set to 0 for all data sets.

5.1 Sensitivity analysis for NBCF-B

As already discussed in Sect. 4.3.1, the only input to the Bimax algorithm is the minimum allowed number of users in a bicluster, n, and the minimum allowed number of items in a bicluster, m. In order to discover the best biclusters (in terms of effectiveness and

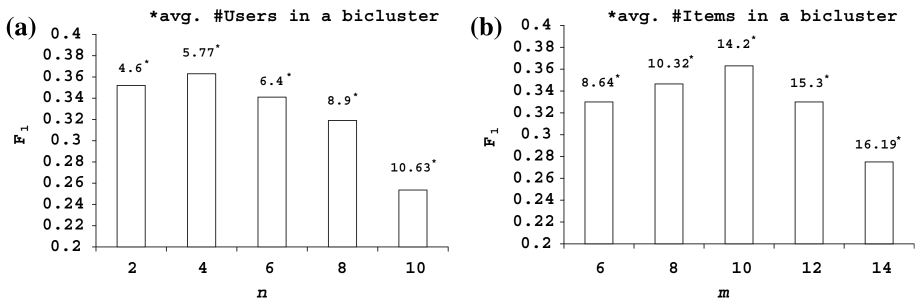


Fig. 6 F_1 versus tuning number of (**a**) users, (**b**) items

efficiency), it is important to fine-tune these two input variables. For the 100 K data set, we examine the performance of F_1 metric versus different values for n and m.

Figure 6a illustrates F_1 for varying n (in this measurement we set $m = 10$). As n is the minimum allowed number of users in a bicluster, Fig. 6a also depicts (through the numbers over the bars) the average numbers of users in a bicluster, which as expected increase with increasing n. As shown, the best performance is attained for $n = 4$. In the following, we keep this as the default value. Nevertheless, notice that performance is, in general, robust against varying n. In particular, for $n \leq 6$ the resulting F_1 is high. In contrast, for higher n, F_1 decreases. The reason is that with higher n we result with an inadequate number of biclusters to provide qualitative recommendations. Thus, small values for n are preferred, a fact that eases the tuning process. Finally, in Fig. 7, we present F_1 for varying n and m parameters simultaneously to (i) show that there is interaction between parameters n and m, and (ii) to verify the findings in Fig. 6a and b. Figure 7 shows that the best F_1 is attained by setting $n = 4$ and $m = 10$, as we have already found in Fig. 6a and b.

Similarly, we examined F_1 for varying m. The results for F_1 are depicted in Fig. 6b ($n = 4$). As previously, in the same figure we also illustrate the resulting average numbers of items in a bilcuster. The best performance is attained for $m = 10$ (henceforth kept as default value), whereas F_1 decreases for higher or lower m values. The reason is as follows: for very small values of m, there are not enough items in each bicluster to capture the similarity of users' preferences (i.e., matching is easily attained), thus the quality of recommendation decreases; on the other hand, for very large values of m, the number of discovered biclusters is not adequate to provide recommendations.

In Sect. 4.3, we mentioned that Bimax finds all biclusters that are not entirely contained in any other bicluster. It is obvious that this characteristic generates overlapping biclusters. The number of overlapping biclusters can be enormously large. To avoid this, we can perform a secondary filtering procedure to reduce the number of biclusters with respect to the desired overlapping degree. In Fig. 8 we can see F_1 versus varying overlapping degree (given as a percentage of common items/users between the biclusters). The figure also depicts (numbers over the bars) the resulting number of biclusters for each overlapping degree. With decreasing overlapping degree, F_1 decreases too. On the other hand, by keeping a high level of overlap between the biclusters, we harm efficiency—in terms of execution time—of the Nearest Biclusters algorithm (for its on-line part). As shown, by permitting 100% of overlapping, the number of generated biclusters is 85, 723. It is obvious that this number impacts the efficiency of the recommendation process. The best combination of effectiveness and efficiency can be attained by having an overlapping equal

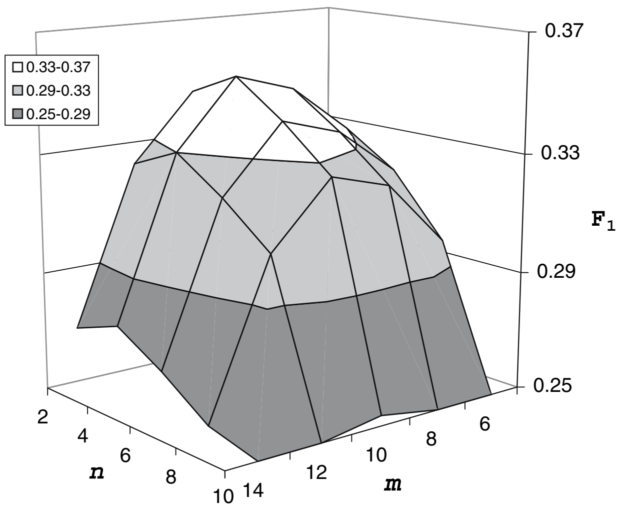


Fig. 7 F_1 versus varying n and m

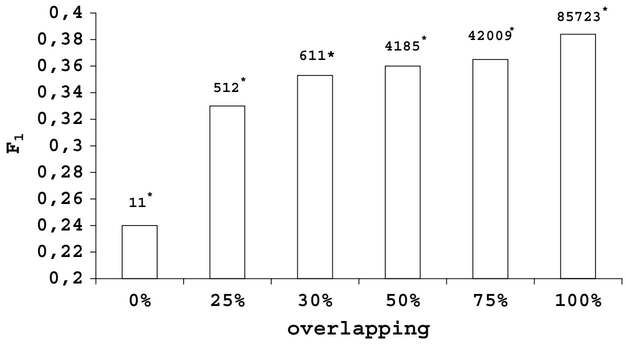


Fig. 8 F_1 versus overlapping percentage between the biclusters

to 30% (results to 611 biclusters), where the resulting F_1 is 0.35 (very close to the 100% overlapping result).

For the 1 M data set, we follow the same tuning procedure and we resulted to the following values which have the best results in our experiments in terms of F1 measure: $n = 3$, $m = 6$, overlapping = 10%, which result to 2126 biclusters. For the EachMovie data set, the input parameters are set as follows: $n = 7$, $m = 12$, overlapping = 20%, which resulted to 1,022 biclusters.

5.2 Sensitivity analysis for NBCF-X

In Sect. 4.3 the required input for the xMotif algorithm is an initial n_s number of users, which will act as seeds for the biclusters. For each randomly selected seed, the algorithm selects n_u users with n_i number of items in a bicluster. Moreover, the total number of created biclusters, t is defined as an input parameter ($n_s * n_i$). In order to improve our

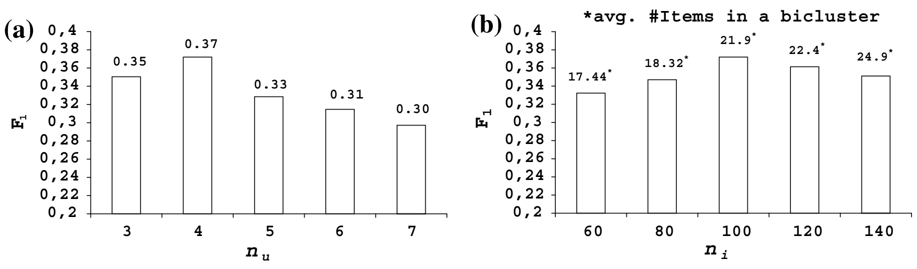


Fig. 9 F_1 versus tuning number of (**a**) users, (**b**) items

recommendations in terms of effectiveness and efficiency, it is important to fine-tune n_u and n_i variables. For the 100 K data set, we examine the performance of F_1 metric versus different values for n_u. Figure 9a illustrates F_1 for varying n_u values (in this measurement we set $n_s = 10$, $n_i = 100$). As shown, the best performance is attained for $n_u = 4$. In the following, we keep this as the default value. Nevertheless, notice that performance is, in general, robust against varying n_u. Figure 9b illustrates F_1 for varying n_i values which is the number of items in a bicluster(in this measurement we set $n_s = 10$, $n_u = 4$). As shown, the best performance is attained for $n_i = 100$. In the following, we keep this as the default value.

Moreover, xMotif algorithm allows a user or item to be included in a bicluster, even if, there exist a fraction of their ratings values which cannot not defined as interesting ones(ratings under the P_τ threshold or null values). Thus, in order to discover the best biclusters (in terms of effectiveness and efficiency), it is also important to fine-tune the p-value parameter which controls a bicluster's homogeneity. So, we examine the performance of F_1 metric versus different values of the p-value parameter. Figure 10a illustrates F_1 for varying p-values (in this measurement we set $n_s = 10$ and $n_u = 4$ $n_i = 100$). As shown, the best performance is attained for p-value =0.001 where the average number of items in biclusters is 21.9. This average has at least 7 items more than the Bimax case. This is the reason we succeed more diversity in our biclusters. In the following, we keep this as the default value.

Finally, note that xMotif does not create *inclusion-maximal* biclusters. This means that there is extend overlapping between biclusters. Nevertheless, this can be avoided through the tuning of the allowed overlapping factor. In Fig. 10b we observe F_1 versus varying overlapping degree (given as a percentage of common items between the biclusters). The figure also depicts (numbers over the bars) the resulting number of biclusters for each

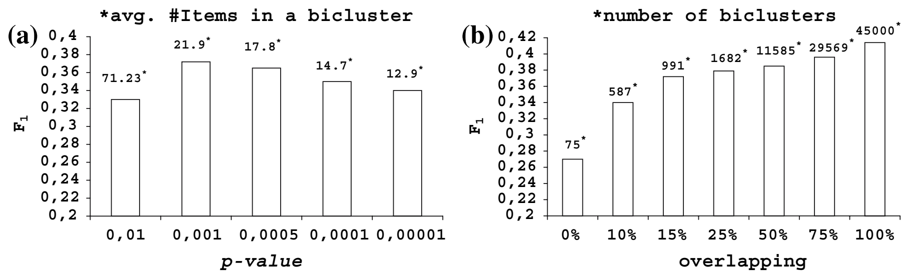


Fig. 10 F_1 versus tuning number of (**a**) p-value parameter, (**b**) overlapping factor

overlapping degree. With decreasing overlapping degree, F_1 decreases too. On the other hand, by keeping a high level of overlap between the biclusters, we harm efficiency—in terms of execution time—of the NBCF-X algorithm (for its on-line part). As shown, by permitting 15% of overlapping, the number of remaining biclusters is 991. It is obvious that this number impacts the efficiency of the recommendation process.

For the 1 M data set, we follow the same tuning procedure and we resulted to the following values which have the best results in our experiments in terms of F1 measure: $n_s = 30$, $n_u = 3$, $n_i = 120$, p-value $= 0.001$, overlapping $= 5\%$, which results to 2,690 biclusters. For the EachMovie data set the input parameters are set as follows: $n_s = 20$, $n_u = 6$, $n_i = 110$, p-value $= 0.005$, overlapping $= 15\%$, which resulted to 1,356 biclusters.

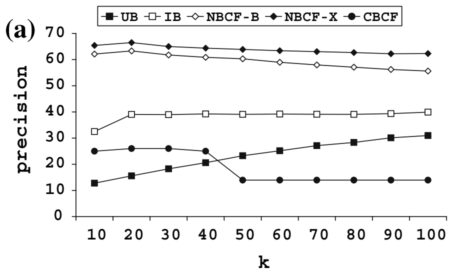
5.3 Comparative results for effectiveness

We proceed to the comparison of NBCF-B, NBCF-X with the UB, IB and CBCF algorithm. The results for precision and recall versus k for 100 K data set are displayed in Fig. 11a and b, respectively.

As shown, UB performs worst than IB for small values of k. The performance of the two algorithms converges to the same value as k increases. The reason is that with a high k, the resulting neighborhoods for both UB and IB are similar, since they include almost all items. Thus, the top-N recommendation lists are about the same, as they are formed just by the most frequent items. In particular, both UB and IB reach an optimum performance for a specific k. In the examined range of k values, the performance of UB and IB increases with increasing k and outside this range (not displayed), it stabilizes and never exceeds 40% precision and 15% recall. Regarding CBCF algorithm, it does not exceed 26% for small values of k, collapses for big values of k and presents the worst performance.

As expected, biclustering algorithms significantly outperform UB and IB. The difference in precision is almost 30%, whereas with respect to recall, it exceeds 10% (we refer to the optimum values resulting from the tuning of k). The reason is that the two Biclustering algorithms take into account partial matching of preferences between users and the possibility of overlapping between their interests. In contrast, UB and IB are based on individual users and items, respectively, and do not consider the aforementioned characteristics.

Comparing the two biclustering algorithms, we observe that NBCF-X outperforms slightly but constantly NBCF-B. The reason is that it includes biclusters with more diversity in items. The difference is 3% in terms of precision and 2% in terms of recall.



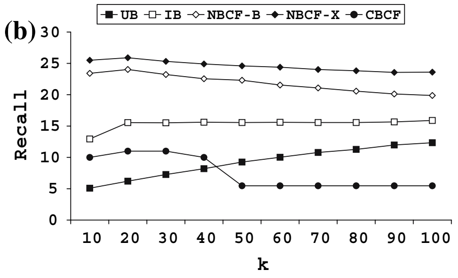


Fig. 11 For the 100 K data set comparison between UB, IB, NBCF-X, NBCF-B and CBCF algorithms in terms of (**a**) precision (**b**) recall

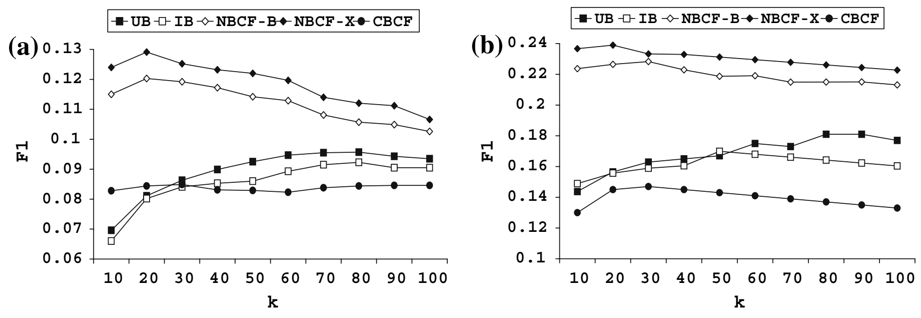


Fig. 12 Comparison between UB, IB, NBCF-X, NBCF-B and CBCF algorithms in terms of F1 measure for (**a**) 1 M data set (**b**) EachMovie data set

For the 1 M data set, the results for F1 measure versus *k* are displayed in Fig. 12a. Evidently, the difference between NBCF-B and NBCF-X increases for the 1 M data set. The reason is that the number of items is double in relation to 100 K data set and the sparsity is larger. Therefore, finding compact biclusters using NBCF-B becomes more difficult. On the other hand, NBCF-X, by exploiting diversity in constructing biclusters (through the *p*-value parameter), can overcome smoothly the aforementioned problems.

For the EachMovie data set, the results for F1 measure versus *k* are displayed in Fig. 12b. As we can see, the difference between NBCF-B and NBCF-X is decreased but NBCF-X is always better than NBCF-B. Both approaches outperform all other three methods. The results for all algorithms are better than 1 M because sparsity of data is smaller in EachMovie data set.

5.4 Comparative results for efficiency

Regarding efficiency, we measured the wall-clock time for the on-line parts of UB, IB, NBCF-B, NBCF-X and CBCF algorithms. The on-line parts concern the time it takes to create a recommendation list, given what is known about a user. Notice that there is an off-line part for the IB, NBCF-X, NBCF-B and CBCF algorithms, which demands additional computational time, needed to build the items' similarity matrix and find the biclusters or clusters, respectively. However, since these computations are executed off-line, we do not count them in the recommendation time. The results versus *k* are presented in Fig. 13. In particular, we present the average time in milliseconds that takes to provide recommendations to a test user.

As expected, IB requires less time to provide recommendations than UB. The required time for IB to provide recommendations for a test user is almost stable, whereas the time for UB increases by increasing *k*. The reason is that UB finds, firstly, user neighbors in the neighborhood matrix and then counts presences of items in the user-item matrix. In contrast, with IB, the whole task is completed in the item-neighborhood matrix, which is generated off-line. Thus, in terms of execution time, IB is superior to UB.

In all cases, NBCF-B algorithm needs less time than IB and NBCF-X to provide recommendations. The reason is that the number of biclusters (611) in our experiment is less than two numbers: (i) the number of items (1,682) in the similarity matrix of IB, and (ii) the number of biclusters (991) in the NBCF-X case. NBCF-X is worse than IB, because the average number of items inside biclusters increases due to diversity. As it is already

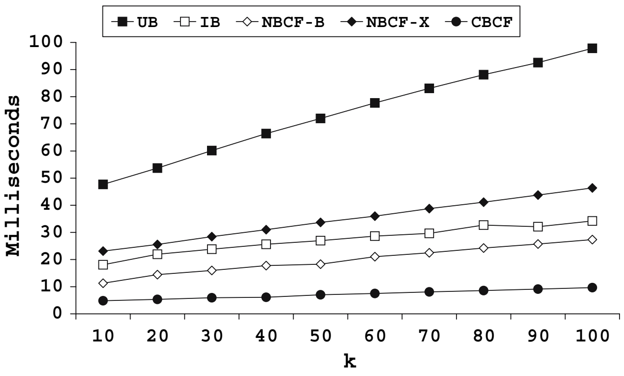


Fig. 13 For the 100 K data set, comparison between UB, IB, NBCF-B, NBCF-X and CBCF in terms of execution time

presented in Sect. 5.2, by decreasing the percentages of overlap between biclusters, accuracy decreases too. On the other hand, by keeping a high level of overlap between biclusters, we harm efficiency. Thus, to combine effectiveness and efficiency, a pre-requisite is a fine-tuning of the degree of overlap between biclusters.

Finally, as expected, CBCF algorithm presents the best performance in terms of execution time. The reasons are the following: (i) there is a small number of created clusters (750), and (ii) it does not take into account the existence of users. But as we have seen, this performance is not followed by effective results in terms of accuracy. For 1 M and EachMovie data sets, our experiments have analogous results for the execution times of the aforementioned algorithms, confirming the 100 k data set results.

5.5 Examination of additional factors

In this section we examine the impact of additional factors. In our measurements we again consider UB, IB, NBCF-B, NBCF-X, and the CBCF algorithms.

Recommendation list's size: We examine the impact of *N*. The results of our experiments are depicted in Fig. 14a and b.

As expected, by increasing *N*, recall increases and precision decreases. The best performance of UB and IB corresponds to the worst performance of biclustering algorithms. The relative differences between the algorithms are coherent with those in our previous

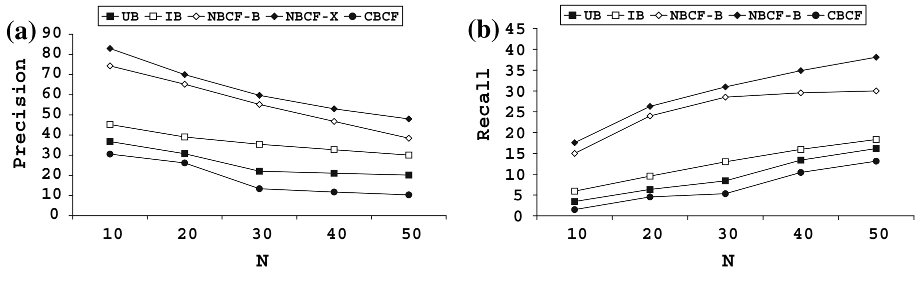


Fig. 14 For the 100 K data set, comparison versus *N* in terms of (**a**) precision, (**b**) recall

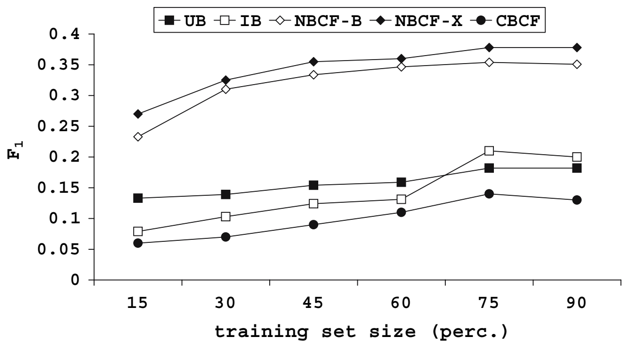


Fig. 15 For the 100 K data set, comparison of F_1 versus training set size

measurements. In real-life applications, *N* should be kept low, because it is impractical for a user to see all recommendations when their number is large.

Training/Test data size: We test the impact of the size of the training set, which is expressed as percentage of the total data set size. The results for F_1 are given in Fig. 15.

As expected, when the training set is small, performance decreases for all algorithms. Therefore, we should be careful when we evaluate CF algorithms so as to use an adequately large training set. Similarly to the previous measurements, NBCF-X and NBCF-B are better than IB and UB in all cases. The performance of both UB and IB reaches a peak around 75%, after which it reduces. It is outstanding that Biclustering algorithms trained with the 15% of the data set, attain much better F_1 than UB and IB, which have been trained with 75%. Also, we see that above a threshold of the training set size, the increase in accuracy is less steep. However, the effect of overfitting is less significant compared to general classification problems. In contrast, low training-set sizes negatively impact accuracy. Therefore, the fair evaluation of CF algorithms should be based on adequately large training sets. Regarding the two biclustering algorithms, as expected, NBCF-X outperforms NBCF-B algorithm in all cases. The reason is that it includes biclusters with more diversity in items. On the other hand, NBCF-B biclusters are more compact and can speed up the recommendation process.

For the 1 M and EachMovie data sets, the results for F_1 are given in Fig. 16a and b, respectively. As expected, the relative differences between the algorithms are coherent with those in our previous measurement. As a result, based on our experiments, we can claim that a 15% of the training set is adequate to provide accurate results.

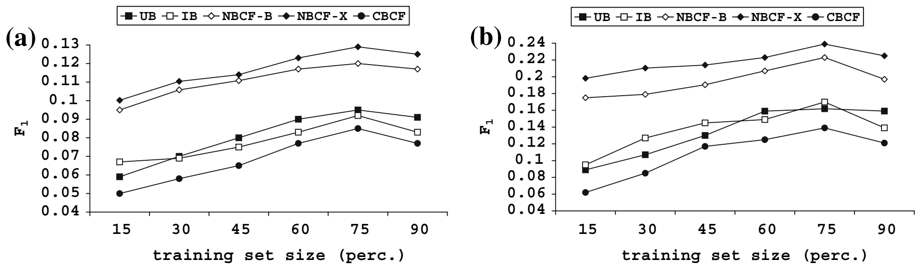


Fig. 16 Comparison of F_1 versus training set size for: (**a**) 1 M data set, (**b**) EachMovie data set

5.6 Comparison to other methods

In this Section, we compare NBCF-X (the algorithm that attained the best performance in previous experiments) against the following state-of-the-art methods:

- Wang et al. (2006) proposed an algorithm based on Similarity Fusion between the UB and IB methods. This algorithm is denoted as SF.
- Jin et al. (2006) proposed the Decoupled Model. This method is denoted as DM.
- Herlocker et al. (2002) proposed the Significance Weighting, denoted as SW.

The parameters we used to evaluate the performance of SF, DM and SW, are identical to those reported in the original papers. However, for data sets that were not used in these papers, we tuned the parameters so as to get the best results for these methods. SF and DM have focused on predicting ratings, not on recommending a list of top-*N* items. Their evaluation has been performed in terms of Mean Absolute Error (MAE), which evaluates algorithms on a per prediction basis.

Previous research (McLauglin and Herlocher 2004] has clearly indicated that this measure biases performance results towards algorithms which predict all items well. However, this does not necessarily hold for the recommendation of top-*N* items. More precisely, MAE works well for measuring how accurately the algorithm predicts the rating of a randomly selected item, but fails to evaluate whether an algorithm will provide a good user experience (McLauglin and Herlocher 2004). Therefore, Precision/Recall evaluation techniques, in addition to MAE, are able to better measure the quality of recommendations. Since, (Wang et al. 2006) and (Jin et al. 2006) lack a method to generate the top-*N* recommendation list for SF and DM, respectively, the most reasonable way of generating the recommendations is as follows: After predicting the ratings for the test user, we rank the predicted ratings and propose the items with the highest predicted rating.

Correspondingly, in order to measure MAE for NBCF-X and SW, we predict the rating of test user u_t on item i as shown in Eq. 6 (McLauglin and Herlocher 2004). Note that for the NBCF-X, $r_{v,\ i}$ is the average rating across the users of the bicluster.

$$\hat{r}_{u_t,i} = \bar{r}_{u_t} + \frac{\sum_{v \in U} \mathrm{sim}(u_t, v)(r_{v,i} - \bar{r}_{u_t})}{\sum_{v \in U} |\mathrm{sim}(u_t, v)|} \tag{6}$$

Initially, we measured MAE for all the examined algorithms for the 100 K Movielens and EachMovie data sets. The results are summarized in Table 2a and b, respectively. In these measurements, following (Jin et al. 2006; Wang et al. 2006), MAE is given against test users with 5 and 10 given items (the set of *observed* items). By varying the number of given items we can test the robustness of the prediction procedure.

As shown, DM attains the lowest MAE values. NBCF-X is second best for the 100 K MovieLens data set, whereas SF is second best for the EachMovie data set. Moreover, as the number of given items increases (in the tables, from 5 to 10), the differences between MAE of all algorithms significantly decreases.

Next, we measure precision versus recall for all four algorithms. The results for the 100 K Movielens and EachMovie data sets are depicted in Fig. 17a and b, respectively. For both data sets, NBCF-X and SW significantly outperform DM and SF, whereas NBCF-X attains the best performance in all cases. These results are in accordance with the conclusions in (McLauglin and Herlocher 2004; Symeonidis et al. 2006): good results on MAE, like those of DM and SF, cannot fully characterize users' experience in web-based recommender systems (such as Amazon, eBay, etc.), which propose a top-*N* ranked list of

Table 2 Comparison between NBCF-X, SW, SF and DM algorithms in terms of MAE for (a) 100 K Movielens and (b) EachMovie data sets

Algorithms	5 Items given	10 Items given
(a)		
DM	**0.84**	**0.79**
NBCF-X	**0.84**	0.83
SW	0.88	0.84
SF	0.89	0.87
(b)		
DM	**1.05**	**1.03**
SF	1.06	1.04
SW	1.09	1.08
NBCF-X	1.08	1.04

A smaller value (bold values) means a better performance

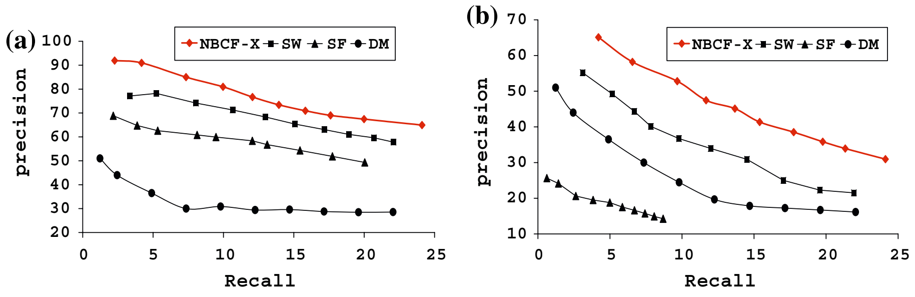


Fig. 17 Comparison between NBCF-X, SW, SF and DM algorithms in terms of precision-recall curve for (**a**) 100 K data set (**b**) EachMovie data set

items to a user. The basic reason is that an error of size ε has the same impact on MAE regardless of where that error places the item in a top-N ranking (McLauglin and Herlocher 2004; Symeonidis et al. 2006). In contrast, results on Precision/Recall can better characterize users' experience in the aforementioned systems. Therefore, since DM and SF has been designed with emphasis on optimizing MAE, are outperformed by SW and NBCF-X. Moreover, compared to SW, the similarity measure of NBCF-X is based on nearest-biclusters, and thus, being able to detect partial matching of users' preferences, can provide accurate recommendations. This is the reason why SW is outperformed by NBCF-X.

6 Conclusions

We proposed the application of two different classes of biclustering algorithms (i) with constant values (Bimax) and (ii) with coherent values (xMotif) in the CF area, to disclose the duality between users and items and to capture the range of users' preferences. In addition, we propose a novel nearest-biclusters algorithm, which uses a new similarity measure that achieves partial matching of users' preferences and allows overlapping interests.

We performed an extensive experimental comparison of NBCF-B and NBCF-X algorithms against state-of-the-art CF algorithms over three real-life data sets (100 k and 1 M Movielens data sets and EachMovie). Our experimental results illustrate the effectiveness and efficiency of the proposed algorithms over the existing CF approaches.

We highlight the following conclusions from our examination:

- Our approach shows significant improvements over existing CF algorithms, in terms of effectiveness, because it exploits the duality of users and items through biclustering and partial matching of users' preferences.
- Our approach shows good improvement over existing nearest neighbor algorithms, in terms of efficiency. The Nearest-biclusters algorithm needs even less time than IB approach to provide recommendations.
- In our experiments, we have seen for 3 different data set, that only a 15% of the training set in our approach is adequate to provide accurate results.
- We introduced a similarity measure for the biclusters' neighborhood formation and proposed the Weighted Frequency for the generation of the top-*N* recommendation list of items.
- In our experiments, NBCF-X algorithm outperforms slightly the NBCF-B algorithm, in terms of accurate recommendations. The reason is that it includes biclusters with more diversity in items. The difference is 3% in terms of precision and 2% in terms of recall.
- Comparing the two biclustering algorithms in terms of efficiency the NBCF-B algorithm needs less time than NBCF-X to provide recommendations. This is due to the fact that the biclusters are with smaller average number of items. Thus, they are more compact and homogeneous.

Summarizing the aforementioned conclusions, we see that, the proposed nearest-biclusters algorithms, by using the partial matching of users' preferences, achieve better results in terms of effectiveness and efficiency than the existing CF algorithms. For this reason, in our future work we will examine our more special classes of biclustering algorithms. Furthermore, we will examine different similarity measures between a user and a bicluster and more top-*N* generation list algorithms by taking also into account the non-positive ratings. Finally, based on the approach of (George and Merugu 2005) we will consider the problem of incremental updating the bi-clusters, in order to speed-up the off-line part of our algorithm.

References

Balabanovic, M., & Fab, S. Y. (1997). Content-based, collaborative recommendation. *ACM Communications, 40*(3), 66–72.

Barkow, S., Bleuler, S., Prelic, A., Zimmermann, P., & Zitzler, E. (2006). BicAT: A biclustering analysis toolbox. *Bioinformatics, 22*(10), 1282–1283.

Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Uncertainty in Artificial Intelligence Conference* (pp. 43–52).

Cheng, Y., & Church, G. (2000). Biclustering of expression data. In *Proceedings of the ISMB Conference* (pp. 93–103).

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems, 22*(1), 143–177.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the ACM SIGKDD Conference*.

Dhillon, I. S., & Mallela, D. S., & Modha, S. (2003). Information theoretic co-clustering. In *Proceedings of the ACM SIGKDD Conference*.

George, T., & Merugu, S. (2005). A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the IEEE ICDM Conference*.

Hartigan, J. A. (1972). Direct clustering of a data matrix. *Journal of the American Statistical Association, 67*(337), 123–129.

Herlocker, J., Konstan, J., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *Proceedings of the ACM SIGIR Conference* (pp. 230–237).

Herlocker, J., Konstan, J., & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval, 5*(4), 287–310.

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems, 22*(1), 5–53.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems, 22*(1), 89–115.

Hofmann, T., & Puzicha, J. (1999). Latent class models for collaborative filtering. In *Proceedings of the IJCAI Conference*.

Jin, R., Si, L., & Zhai, C. (2006). A study of mixture models for collaborative filtering. *Information Retrieval, 9*(3), 357–382.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the ACM CIKM Conference* (pp. 247–254).

Kohrs, A., & Merialdo, B. (1998). Clustering for collaborative filtering applications. In *Proceedings of the CIMKA Conference*.

Lemire, D., & Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining Conference*.

Long, B., Zhangm, Z., & Yu, P. S. (2005). A formal statistical approach to collaborative filtering. In *Proceedings of the ACM SIGKDD Conference*.

Madeira, S., & Oliveira, A. (2004). Biclustering algorithms for biological data analysis: A survey. *ACM Transactions on Computational Biology and Bioinformatics, 1*, 24–45.

McLauglin, R., & Herlocher, J. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the ACM SIGIR Conference* (pp. 329–336).

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Proc. AAAI conf.* (pp. 187–192).

Mirkin, B. (1996). *Mathematical classification and clustering*. Kluwer Academic Publishers: Dordrecht.

Mobasher, B., Dai, H., Luo, T., & Nakagawa, M. (2001). Improving the effectiveness of collaborative filtering on anonymous web usage data. In *Proceedings of the Workshop Intelligent Techniques for Web Personalization* (pp. 53–60).

Murali, T., & Kasif, S. (2003). Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the Pacific Symposium on Biocompomputing Conference* (Vol. 8, pp. 77–88).

Prelic, A., et al. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics, 22*(9), 1122–1129.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering on netnews. In *Proceedings of the Computer Supported Collaborative Work Conference* (pp. 175–186).

Salter, J., & Antonopoulos, N. (2006). Cinemascreen recommender agent: Combining collaborative and content-based filtering. *Intelligent Systems Magazine, 21*(1), 35–41.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the ACM Electronic Commerce Conference* (pp. 158–167).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the WWW Conference* (pp. 285–295).

Shafiei, M., & Milios, E. (2005). Model-based overlapping co-clustering. In *Proceedings of the IEEE SDM Conference*.

Symeonidis, P., Nanopoulos, A., Papadopoulos, A., & Manolopoulos, Y. (2006). Collaborative filtering process in a whole new light. In *Proc. IDEAS conf.* (pp. 29–36).

Ungar, L., & Foster, D. (1998). A formal statistical approach to collaborative filtering. In *Proceedings of the CONALD Conference*.

Wang, J., Vries, A., & Reinders, M. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the SIGIR Conference* (pp. 501–508).

Xue, G., Lin, C., & Yang, Q., et al. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the ACM SIGIR Conference* (pp. 114–121).