Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications xxx (2007) xxx–xxx
Expert Systems with Applications
www.elsevier.com/locate/eswa

Collaborative recommender systems: Combining effectiveness and efficiency

Panagiotis Symeonidis *, Alexandros Nanopoulos, Apostolos N. Papadopoulos, Yannis Manolopoulos

Department of Informatics, Aristotle University, 54124 Thessaloniki, Greece

Abstract

Recommender systems base their operation on past user ratings over a collection of items, for instance, books, CDs, etc. Collaborative filtering (CF) is a successful recommendation technique that confronts the ''information overload'' problem. Memory-based algorithms recommend according to the preferences of nearest neighbors, and model-based algorithms recommend by first developing a model of user ratings. In this paper, we bring to surface factors that affect CF process in order to identify existing false beliefs. In terms of accuracy, by being able to view the ''big picture'', we propose new approaches that substantially improve the performance of CF algorithms. For instance, we obtain more than 40% increase in precision in comparison to widely-used CF algorithms. In terms of efficiency, we propose a model-based approach based on latent semantic indexing (LSI), that reduces execution times at least 50% than the classic CF algorithms.

Keywords: Recommender system; Collaborative filtering; Nearest neighbors

1. Introduction

The ''information overload'' problem affects our everyday experience while searching for knowledge on a topic. To overcome this problem, we often rely on suggestions from others who have more experience on the topic. However, in Web case where there are numerous suggestions, it is not easy to detect the trustworthy ones. Shifting from individual to collective suggestions, the process of recommendation becomes controllable. This is attained with the introduction of CF, which provides recommendations based on the suggestions of users who have similar preferences. Since CF is able to capture the particular preferences of a user, it has become one of the most popular methods in recommender systems.

Two types of CF algorithms have been proposed in the literature: memory-based algorithms, which recommend according to the preferences of nearest neighbors, and model-based algorithms, which recommend by first developing a model of user ratings. Both practical experience and related research have reported that memory-based algorithms (a.k.a. nearest-neighbor algorithms) present excellent performance, in terms of accuracy, for multi-value rating data. On the other hand, model-based algorithms are efficiently handle scalability to large data sets.

1.1. Motivation

Nearest-neighbor CF is influenced by several factors. Related research on CF, during the past decade, approached some of these factors. However, existing approaches may not be considered complete, because they examine the various factors only partially. More specifically, existing CF algorithms and their experimental evaluation focus only on parts of the CF process and do not

* Corresponding author.
E-mail addresses: symeon@csd.auth.gr (P. Symeonidis), ananopou@csd.auth.gr (A. Nanopoulos), papadopo@csd.auth.gr (A.N. Papadopoulos), manolopo@csd.auth.gr (Y. Manolopoulos).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.05.013

handle it as a whole. For the aspects that these partial considerations do not examine, they usually make choices, which our study demonstrates that can be misleading. Through our study we are also able to confirm that there exist dependencies between the factors affecting CF. Therefore, we have to perform an evaluation of the entire CF process in order to produce reliable conclusions.

Moreover, to handle scalability, we have to extend our findings for nearest-neighbor CF algorithms through a model-based approach. This approach will combine the effectiveness of the nearest-neighbor CF algorithms in terms of accuracy, with the efficiency in terms of execution time. Towards this direction, latent semantic indexing (LSI) is a technique that has been extensively used in informational retrieval. LSI detects latent relationships between documents and terms. In CF, LSI can be used to form users' trends from individual preferences, by detecting latent relationships between users and items. Therefore, with LSI, a higher level representation of the original user-item matrix is produced, which presents a three-fold advantage: (i) it contains the main trends of users' preferences, (ii) noise is removed, (iii) it is much more condensed than the original matrix, thus it favors scalability.

1.2. Contributions

In this work, first, we provide a thorough analysis of the factors involved in CF. Notably, we examine several similarity measures, various criteria for generating the recommendation list, the appropriateness of evaluation metrics, and the impact of CF in real-world applications, which is considered through user's satisfaction (measured with the popularity of recommended items) and the division of the ratings of the test user in past and future sets. During the analysis we identify choices that have been incorrectly adopted and new issues that have not been considered so far. As a result, we propose several extensions and new approaches, which substantially improve the entire CF process. Moreover, we propose a new model-based CF approach, which is based on LSI to produce a condensed model for the user-item matrix that handles the factor of scalability.

Our contributions are summarized as follows:

- The proposed approach examines the factors involved through the entire CF process. This helps to: (a) reveal fallacies in existing beliefs for several of them, (b) better analyze their impact and provide insights, and (c) synthesize a novel method, which substantially improve the effectiveness of CF in terms of accuracy.
- The previous findings are extended with an approach based on LSI. This way, execution times for CF are significantly reduced. Moreover, improvement is possible in effectiveness too, because the proposed model identifies the main trends and removes noise. Notice that differently from similar approaches in related work, we introduce the notion of pseudo-user in order to fold-in the vectors of target users in the produced model (related work incorrectly assumes the knowledge of target users during the building of the model).
- We carried out extensive experimental evaluation, which, to our knowledge, considers all the involved factors for the first time. Our experimental results demonstrate the superiority of the proposed methods (more than 40% improvements in terms of precision over widely-used CF algorithms and 50% in terms of execution times).

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 contains the analysis of the CF factors. The proposed approaches are described in Sections 4 and 5. Experimental results are given in Section 6. Finally, Section 7 concludes this paper.

2. Related work

In 1992, the Tapestry system (Goldberg, Nichols, Brian, & Terry, 1992) introduced collaborative filtering (CF). In 1994, the GroupLens system (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994) implemented a CF algorithm based on common users preferences. Nowadays, it is known as user-based CF algorithm, 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, Heckerman, & Kadie, 1998; Herlocker, Konstan, Borchers, & Riedl, 1999).

In 2001, another CF algorithm was proposed. It is based on the items' similarities for a neighborhood generation of nearest items (Karypis, 2001; Sarwar, Karypis, Konstan, & Riedl, 2001) and is denoted as item-based CF algorithm.

Most recent work followed the two aforementioned directions (i.e., user-based and item-based). Herlocker, Konstan, and Riedl (2002) weight similarities by the number of common ratings between users/items. Deshpande and Karypis (2004) apply item-based CF algorithm combined with conditional-based probability similarity and cosine-similarity. Xue, Lin, and Yang (2005) suggest a hybrid integration of aforementioned algorithms (memory-based) with model-based algorithms.

All aforementioned algorithms are memory-based. Their efficiency is affected from scalability of data. This means that they face performance problems, when the volume of data is extremely big. To deal with this problem, many model-based algorithms have been developed (Breese et al., 1998). However, there are two conflicting challenges. If an algorithm spends less execution time, this should not worse its quality. The best result would be to improve quality with the minimum calculation effort.

Furnas, Deerwester, and Dumais (1988) proposed latent semantic indexing (LSI) in information retrieval area to deal with the aforementioned challenges. More specifically, LSI uses SVD to capture latent associations between the terms

Please cite this article in press as: Symeonidis, P. et al., Collaborative recommender systems: Combining effectiveness ..., *Expert Systems with Applications* (2007), doi:10.1016/j.eswa.2007.05.013

and the documents. SVD is a well-known factorization technique that factors a matrix into three matrices. Berry, Dumais, and O'Brien (1994) carried out a survey of the computational requirements for managing (e.g., folding-in[1]) LSI-encoded databases. He claimed that the reduced-dimensions model is less noisy than the original data.

Sarwar, Karypis, and Konstan (2000, 2002) applied dimensionality reduction for the user based CF approach. He also used SVD for generating predictions. In contrast to our work, Sarwar et al. (2000, 2002) do not consider two significant issues: (i) Predictions should be based on the users' neighbors and not on the test (target) user, as the ratings of the latter are not apriori known. For this reason we rely only on the neighborhood of the test user. (ii) The test users should not be included in the calculation of the model, because they are not known during the factorization phase. For this reason, we introduce the notion of pseudo-user in order to include a new user in the model (folding in), from which recommendations are derived. Other related work also includes Goldberg, Roeder, Gupta, and Perkins (2001), who applied Principal Components Analysis (PCA) to facilitate off-line dimensionality reduction for clustering the users, and therefore, manages to have rapid on-line computation of recommendations. Hofmann (2004) proposed a model-based algorithm which relies on latent semantic and statistical models.

3. Factors affecting the CF process

In this section, we identify the major factors that critically affect all CF algorithms. Our analysis focuses on the basic operations of the CF process, which consists of three stages.

- *Stage 1*: formation of user or item neighborhood, where objects inside the neighborhood have similar ratings and behavior.
- *Stage 2*: top-N list generation with algorithms that construct a list of best items recommendations for a user.
- *Stage 3*: quality assessment of the top-N list.

In the rest of this section we elaborate on the aforementioned factors, which are organized with respect to the stage that each one is involved. The examined factors,which are detailed in the following, are described in Table 1. Table 2 summarizes the symbols that are used in the sequel. To ease the discussion, we will use the running example illustrated in Fig. 1 where U_{1-10} are users I_{1-6} are items. As shown, the example data set is divided into a training and test set. The null cells (no rating) are represented as zeros.

Note that related work has identified some few more factors, like the impact of subjectivity during the rating or issues concerning the preprocessing of data (Breese et al., 1998; Mobasher, Dai, Luo, & Nakagawa, 2001; Sarwar et al., 2000). Nevertheless, we do not examine these factors, because their effect is less easily determinable.

[1] Folding in terms or documents is a simple technique that uses existing SVD to represent new information.

Table 1
Factors affect CF algorithms

Factor name	Short description	Stage
Sparsity	Limited percentage of rated products	1
Scalability	Computation increase by the number of users and items	1
Train/test data size	Data are divided to training and evaluation or test	1,3
Neighborhood size	Number of neighbors used for the neighborhood formation	1
Similarity measure	Measures that calculate proximity of two objects	1
Recommendation list size	Number of top-N recommended items	2
Generation of recommendation list	Algorithms for the top-N list generation	2
Positive rating-threshold	Positive and negative ratings segregation	2,3
Evaluation metrics	Metrics that evaluate the quality of top-N list	3
Setting a baseline method	A simple method against which performance is compared	3
Past/future items	The segregation between apriori known and unknown items	3

Table 2
Symbols and definitions

Symbol	Definition	Symbol	Definition
k	Number of nearest neighbors	$\bar{r}_i$	Mean rating value for item i
N	Size of recommendation list	$p_{u,i}$	Predicted rate for user u on item i
$NN(u)$	Nearest neighbors of user u	$\|T\|$	Size of the test set
$NN(i)$	Nearest neighbors of item i	c	Number of singular values
P_τ	Threshold for positive ratings	A	Original matrix
$\mathscr{I}$	Domain of all items	U	Left singular vectors of A
$\mathscr{U}$	Domain of all users	S	Singular values of A
u, v	Some users	V'	Right singular vectors of A
i,j	Some items	A^*	Approximation matrix of A
$\mathscr{I}_u$	Set of items rated by user u	$\mathbf{u}$	User vector
$\mathscr{U}_i$	Set of users rated item i	$\mathbf{u}_{\text{new}}$	Inserted user vector
$r_{u,i}$	The rating of user u on item i	n	Number of training users
$\bar{r}_u$	Mean rating value for user u	m	Number of items

3.1. First stage factors

Sparsity: In most real-world cases, users rate only a very small percentage of items. This causes data sets to become sparse. The problem of sparsity is extensively studied. In

a

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4	1	1	4	0	0
U_2	1	4	4	0	0	4
U_3	2	1	4	0	0	4
U_4	1	2	1	1	0	0
U_5	4	1	2	0	1	0
U_6	1	5	4	0	4	0

b

	I_1	I_2	I_3	I_4	I_5	I_6
U_7	2	1	4	0	1	0
U_8	1	2	1	0	2	5
U_9	4	1	2	1	1	0
U_{10}	1	4	1	0	4	0

Fig. 1. (a) Training set ($n \times m$) and (b) test set.

such cases, the recommendation engine cannot provide precise proposals, due to lack of sufficient information. A similar problem of CF algorithms is the one of cold-start (O'Mahony, Hurley, Kushmerick, & Silvestre, 2004).

In few existing works (Mobasher et al., 2001; Sarwar et al., 2000), there is a preprocessing step that fills missing values. Our experimental examination indicated that this approach incurs the loss of valuable information. For the same reasons, the aforementioned direction has not been followed by forthcoming works.

Several recent works (Herlocker et al., 2002; Karypis, 2001; Sarwar et al., 2001) focus only on very sparse data. Also, related work provides benchmark data sets with different sparsity, e.g., the Jester data set (Goldberg et al., 2001) is dense; in contrast the Movielens data sets (Herlocker et al., 1999) are relatively sparse. The degree of sparsity, however, depends on the application type. To provide complete conclusions, someone has to experiment with the amount of sparsity as well.

Scalability: Scalability is important, because in real-world applications the number of users/items is very large. Related work (Sarwar et al., 2000) has proposed the use of dimensionality reduction techniques, which introduce a trade-off between the accuracy and the execution time of CF algorithms. In Section 5, we will propose a model to confront with the scalability problem.

Train/test data size: There is a clear dependence between the training set's size and the accuracy of CF algorithms (Sarwar et al., 2001). Through our experimental study we verified this conclusion. Additionally, we saw that after an upper 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. Though most related research uses a size around 80%, there exist works that use significantly smaller sizes (McLauglin & Herlocher, 2004). From our experimental results we concluded that an 75% training set size corresponds to an adequate choice. But we have to notice that training/test size should not be data set independent. (In the running example, we set training size at 60%).

Neighborhood size: The number, k, of nearest neighbors used for the neighborhood formation produces a trade-off: a very small k results to low accuracy, because there are not enough neighbors to base prediction. In contrast, a very large k impacts precision too, as the particularities of user's preferences can be blunted due to the large neighborhood size. In most related works (Herlocker et al., 1999; Sarwar, Karypis, Konstan, & Riedl, 2000), 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(In the running example, we set $k = 3$).

Similarity measure: Related work (Herlocker et al., 2002; McLauglin & Herlocher, 2004; Mobasher et al., 2001; Sarwar et al., 2001) has mainly used Pearson correlation and cosine-similarity. In particular, user-based (UB) CF algorithms use the Pearson correlation (Eq. (1)),[2] which measures the similarity between two users, u and v. Item-based (IB) CF algorithms use a variation of adjusted cosine-similarity (Eq. (2)),[3] which measures the similarity between two items, i and j, and has been proved more accurate (McLauglin & Herlocher, 2004; Sarwar et al., 2001), as it normalizes bias from subjective ratings.

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

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

Herlocker et al. (2002) proposed a variation of the previous measures, which henceforth is denoted as Weighted Similarity (WS). If sim is a similarity measure (e.g., Pearson or cosine), then WS is equal to $\frac{\max(c,\gamma)}{\gamma} \cdot \mathrm{sim}$, where c is the number of co-rated items.

Eq. (1) takes into account only the set of items, S, that are *co-rated* by both users. This, however, ignores the items rated by only one of the two users. The number of the latter items denotes how much their preferences differ. Especially for the case of sparse data, by ignoring these items we discard significant information. Analogous reasoning applies for Eq. (2), which considers (in the numerator) only the set of users, T, that co-rated both the examined pair of items, and for WS, which is based on Eqs. (1) or (2). To

[2] Means $\bar{r}_u$, $\bar{r}_v$ are the mean ratings of u and v over their co-rated items.
[3] Means $\bar{r}_u$, $\bar{r}_v$ are taken over all ratings of u and v.

address the problem, in the following, we will examine alternative definitions for S and T.

Another issue that has not been precisely clarified in related work, is whether we include in the neighborhood a user or item with negative similarity. In order to improve accuracy, we suggest keeping only the positive similarities for the neighborhood formation, even if less than the specified number k of neighbors remain. This approach is also followed in several works (Mobasher et al., 2001). Our experiments have shown that, by including users or items with negative similarities (just to maintain a constant neighborhood size), can impact the accuracy.

Moreover, we have to mention that, for the generation process of item similarities matrix, this can be made off-line. In contrast, the user similarities matrix should be created on-line. More specifically, for the user similarities matrix we use the test data, while for the item based similarities table we use only training data.

The application of Pearson Correlation (Eq. (1)) to the running example is depicted in Fig. 2a and the resulting k-nearest neighbors (k-NN) are given in Fig. 2b. Respectively, the similarities between items, calculated with the adjusted cosine measure (Eq. (2)), are given in Fig. 2c and d depicts the nearest neighbors. As only positive values of similarities are considered during the neighborhood formation, the items have different neighborhood size.

3.2. Second stage factors

Recommendation list's size: The size, N, of the recommendation list corresponds to a trade-off: 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. (In the running example, we set $N = 2$).

Positive rating threshold: It is evident that recommendations should be "positive". It is not success to recommend an item that will be rated with 1 in scale 1–5. Nevertheless, this issue is not clearly defined in several existing works. We argue that "negatively" rated items should not contribute to the increase of accuracy, and we use a rating-threshold, P_τ, to recommended items whose rating is no less than this value. If not a P_τ value is used, then the results can become misleading, since negative ratings can contribute to the measurement of accuracy.

Generation of recommendation list: The most often used technique for the generation of the top-N list, is the one that counts the frequency of each item inside the found neighborhood, and recommends the N most frequent ones (Sarwar et al., 2000). Henceforth, this technique is denoted as Most-Frequent item recommendation (MF). MF can be applied to both user-based and item-based CF algorithms. For example, assume that we follow the aforementioned approach for the test user U_7, for $k = 3$ and $N = 2$. For the case of user-based recommendation, the top-2 list includes items I_3, I_6. In contrast, for the case of item-based recommendation, the top-2 list includes two items of I_6 or I_2 or I_5 because all three have equal presence. Fig. 3 describes the corresponding two algorithms (for the user and item-based CF, respectively).

We have to mention that these two algorithms, in contrast to the past work, include, in addition, the concept of positive rating-threshold ($P\tau$). Thus, "negatively" rated items do not participate to the top-N list formation. Moreover, it is obvious that the generation of top-N list for the user-based approach is more complex and time consuming. The reason is that the former algorithm finds, firstly, user neighbors in the neighborhood matrix and then counts presences of items in the user-item matrix. In contrast, with the item-based approach the whole work is completed in the item neighborhood matrix.

Karypis (2001) reports another technique, which additionally considers the degree of similarity between items. This takes into account that the similarities of the k neighbors may vary significantly. Thus, for each item in the neighborhood, this technique admeasures not just their number of appearances, but the similarity of neighbors as well. The N items with the highest sum of similarities are finally recommended. Henceforth, this technique is denoted

a

	U_1	U_2	U_3	U_4	U_5	U_6
U_7	-0.19	0.19	1	-0.76	0.33	-0.14
U_8	-0.5	0.44	0.38	1	-0.82	0.67
U_9	0.41	-0.94	0.14	-0.47	1	-0.95
U_10	-0.5	0.5	-0.76	1	-0.82	0.67

b

	$1^{st}NN$	$2^{st}NN$	$3^{st}NN$
U_7	$U_3(1)$	$U_5(0.33)$	$U_2(0.19)$
U_8	$U_4(1)$	$U_6(0.67)$	$U_2(0.44)$
U_9	$U_5(1)$	$U_1(0.41)$	$U_3(0.14)$
U_10	$U_4(1)$	$U_6(0.67)$	$U_2(0.5)$

c

	I_1	I_2	I_3	I_4	I_5	I_6
I_1	-	-0.65	-0.66	0.36	-0.68	-0.42
I_2	-0.65	-	0.18	-0.51	0.50	-0.36
I_3	-0.66	0.18	-	-0.66	0.10	0.67
I_4	0.36	-0.51	-0.66	-	0	0
I_5	-0.68	0.50	0.10	0	-	0
I_6	-0.42	-0.36	0.67	0	0	-

d

	$1^{st}NN$	$2^{st}NN$	$3^{st}NN$
I_1	$I_4(0.36)$	-	-
I_2	$I_5(0.50)$	$I_3(0.18)$	-
I_3	$I_6(0.67)$	$I_2(0.18)$	$I_5(0.10)$
I_4	$I_1(0.36)$	-	-
I_5	$I_2(0.50)$	I3(0.10)	-
I_6	$I_3(0.67)$	-	-

Fig. 2. (a) Users' similarities matrix, (b) users' nearest neighbors in descending similarity matrix, (c) items' similarities matrix and (d) items' nearest neighbors in descending similarity matrix.

Please cite this article in press as: Symeonidis, P. et al., Collaborative recommender systems: Combining effectiveness ..., *Expert Systems with Applications* (2007), doi:10.1016/j.eswa.2007.05.013

a

```
Vector GenTopMF(u, NN(u), I, N)
//User u
//Set NN(u)
//int I,N
begin
    for i = 1 to I
        f[i] = 0;
    foreach ν ∈ NN(u)
        for i = 1 to I
            if r[ν][i] ≥ Pτ
                f[i] = f[i] + 1;
    sort(f); //descending order
    n = 0; i = 1;
    while (n < N and f[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

b

```
Vector GenTopMF(u, NN(i), I, N)
//User u
//Set NN(i)
//int I,N
begin
    for j = 1 to I
        f[j] = 0;
    for j = 1 to I
        if r[u][j] ≥ Pτ
            foreach ν ∈ NN(i)
                f[ν] = f[ν] + 1;
    sort(f); //descending order
    n = 0; i = 1;
    while (n < N and f[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

Fig. 3. Generation of top-N list based on most frequent algorithm for (a) user-based algorithm and (b) item-based algorithm.

as Highest-Sum-of-Similarities item recommendation (HSS). HSS is applicable only to item-based CF. For our example, the top-2 list based on this algorithm includes the items I_6, I_2 because they have the greater sum of similarities. Note that for both techniques, we have to remove from the recommendation items, these that are already rated from the test user.

Based on the aforementioned rationale, we wanted to perform an examination of other additional criteria against MF (used in the majority of existing works), in order to examine if this direction is promising for future work, because besides (Karypis, 2001), very few works elaborate on this issue.

3.3. Third stage factors

Evaluation metrics: For the evaluation of CF algorithms several metrics have been used in related work (Herlocker et al., 2002; Herlocker, Konstan, Terveen, & Riedl, 2004), for instance the Mean Absolute Error (MAE) or the Receiving Operating Characteristic (ROC) curve. Although MAE has been used in most of related works, it has received criticism as well (McLauglin & Herlocher, 2004). From our experimental study we understood that MAE is able to characterize the accuracy of prediction, but is not indicative for the accuracy of recommendation, as algorithms with worse MAE many times produce more accurate recommendations than others with better MAE. Since in real-world recommender systems the experience of users mainly depends on the accuracy of recommendation, MAE may not be the preferred measure. Other extensively used metrics are *precision* and *recall*. These metrics are simple, well known, and effective to measure the accuracy of recommendation procedure.

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). In the following we also use $F_1 = 2 \cdot \text{recall} \cdot \text{precision}/(\text{recall} + \text{precision})$. F_1 is used because it combines both the previous metrics.

Recently, (McLauglin & Herlocher, 2004) has proposed a modified precision metric for evaluating user experience to address obscure recommendations that cannot been identified with the MAE measure. To keep comparisons with prior work clear, we focused on the traditional precision/recall metrics. However, we also consider the issue of items' popularity in order to test if the examined algorithms recommend obscure items or not.

Setting a baseline method: Existing experimental evaluations lack the comparison against a baseline algorithm. The baseline algorithm has to be simple and to indicate what can be attained with as little effort as possible. Through a baseline, we can see the actual improvement due to existing CF algorithms.

Past/future data: In real-world applications, recommendations are derived only from the currently available ratings of the test user. However, in most of related works, all the ratings of each test user is considered apriori known. For a more realistic evaluation, recommendations should consider the division of items of the test user into two sets (Huang, Chen, & Zeng, 2004): (i) the past items of the test user and (ii) the future items of the test user. As most existing works ignore this division, their reported performance corresponds to the best case, because they exploit apriori known information. To make it more clear, let's see an example. Assume that a test user has rated 10 items. According to most previous works, all the 10 ratings are

used to calculate his similarity with each training user. In contrast, we want to measure how algorithms behave when we use a smaller number of items than those he will finally rate. This simulates the real-world applications, where the test user gradually rates items and receives recommendations before he provides all his ratings. In the previous example, if we use two past-items, we have to compute similarities based only on the provided two ratings and we have to predict the remaining eight items (those in the future set). If only a fraction of items is included in the past set, then the accuracy is expected to decrease compared to the best case. For this reason, we study the effect of this issue.

4. Proposed extensions for memory-based algorithms

In the sequel, we describe in more detail our proposed extensions. These extensions are based on the "big picture" of the CF process, which was obtained from the investigation of factors described in Section 3.

4.1. First stage extension : the UNION similarity measures

We first examined the two most important factors that are involved in the first stage: sparsity and the similarity measure. As mentioned, for the formation of sets S and T (see Eqs. (1) and (2)), past work (Herlocker et al., 2004; McLauglin & Herlocher, 2004; Sarwar et al., 2001) takes into account only the items that are co-rated by both users.

Example a: In Fig. 4, Example a depicts the ratings of four users, U_1–U_4, over six items. Comparing the similarity of U_1 and U_2 users, when only co-rated items are considered, then the similarity measure will be computed based only on the ratings for I_1 and I_3.

In case of sparse data, we have a very small amount of provided ratings to compute the similarity measure. By additionally constraining S and T with co-rated items only, we reduce further the effective amount of used information. To avoid this, we consider alternative definitions for S and T, given in Eq. (3):

$$S = I_u \cup I_v, \quad T = U_i \cup U_j \tag{3}$$

According to Eq. (3), S includes items rated by at least one of the users. In the Example a of Fig. 4, except the ratings for I_1 and I_3, the ratings for I_2 and I_4 will be considered too (the issue of how to treat items rated by only one user, will be discussed in the following). Similar reasoning is followed for the set T, in the case of IB CF. By combining the definitions of S and T given in Eq. (3) with the Pearson correlation and adjusted cosine-similarity measures, we get two reformed measures: UNION Pearson correlation (for UB) and UNION adjusted cosine (for IB), respectively.[4] Notice that in case of UNION Pearson correlation, user mean ratings correspond to the average user ratings over all rated items. To further understand why should not base similarity only on co-rated items, consider the following example.

Example b: In Fig. 4, Example b depicts the items rated positively (i.e., higher than P_τ) by a test user U_{test} and two users, U_1 and U_2, belonging in the training set. U_{test} and U_1 have co-rated items I_1 and I_2. Assume that U_{test} and U_1 rated I_1 with 5 in the 1–5 scale, whereas I_2 have been rated by both of them with 4. Nevertheless, items $I_3 - I_9$ are rated only by U_{test} or U_1, and not by both. In this case, the Pearson measure of Eq. (1), which is based on co-rated items only, results to the maximum possible similarity value (i.e., equal to 1) between U_{test} and U_1. However, this is based only on the two co-rated items and ignores the seven items that are rated only by one of them. On the other hand, assume that U_2 rated I_1 and I_2 with 5 and 4, respectively. As previously, the Pearson measure of Eq. (1) results to the maximum possible similarity between U_{test} and U_2, whereas U_{test} and U_2 differ in three items rated by only one of them. This example reflects the impotence of Eq. (1) to capture the actual notion of similarity: despite the fact that U_{test} and U_1 differ at seven items and U_{test} and U_1 differ at three, it assigns the same similarity value in both cases.

In the previous example, if we designate U_1 as neighbor of U_{test}, we ignore two issues: (i) Items I_3 and I_4, will not be recommended to U_{test} by U_1, as U_1 has not rated them; this fact harms recall. (ii) Items I_5–I_9 will be recommended by U_1, but as they have not been rated by U_{test}, this will harm precision. It follows that a desirable property from a similarity measure is to maximize the number, x, of items that are co-rated by the test user and each of his neighbors, relatively to the number, y, of items that are rated only by one of them (in the example of Fig. 4b, for U_{test} and U_1, $x = 2$ and $y = 7$). In the best case, the ratio $x/(x + y)$ has value equal to 1 and in the worst 0.

To evaluate the previously described argument, we compared Pearson correlation against UNION by performing the following measurement. We used the MovieLens 100K data set and for each test user we computed its k nearest neighbors (k was set to 10) from the training set. Next, we measured x and y between each test user and each of his k neighbors. Fig. 5a illustrates for each x, the resulting ratio $x/(x + y)$. Fig. 5a clearly presents that Pearson

a

	I_1	I_2	I_3	I_4	I_5	I_6
U_1	4	-	1	4	-	-
U_2	1	4	4	-	-	4
U_3	2	1	4	-	-	4
U_4	1	2	1	1	-	-

b

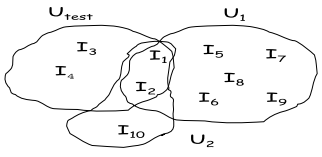

Fig. 4. (a) Example of the ratings of four users over six items (dash denotes an empty value) and (b) example of items rated positively by a test user U_{test} and two other users U_1 and U_2.

[4] Henceforth, when it is clearly understood from the context whether we discuss about UB or IB, we use only the name UNION.

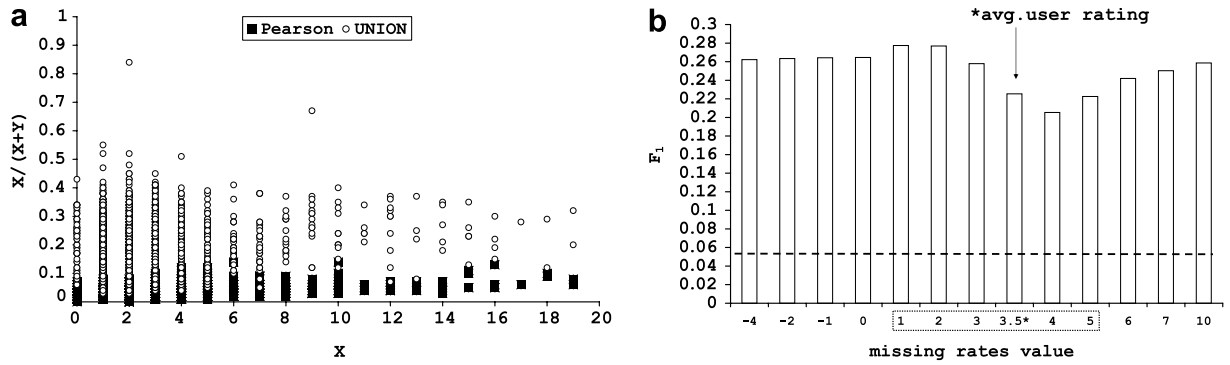

Fig. 5. (a) Measuring the ratio $x/(x+y)$ and (b) impact of assigned value for unrated items.

measure results to significantly lower ratios than UNION. This explains why UNION compares favorably to Pearson correlation in terms of precision and recall, as will be presented experimentally in Section 6. (Due to lack of space we do not present the analogous comparison between adjusted cosine and UNION.)

4.1.1. Assigning a value to unrated items

To calculate the UNION Pearson correlation between two users U_1 and U_2, we have to assign a rating by U_1 to an item that is rated only by U_2 (e.g., I_2 in the example of Fig. 4a) and vice-versa. The same requirement holds for the UNION adjusted cosine measure in the IB case. Notice that this problem cannot be effectively solved with existing techniques for filling missing values, because the sparsity of user-item matrices severely hinders this task (more than 99.9% missing values). For this reason we assign the same rating value to all the required cases. There could be several options for this value. For instance, in a 1–5 scale, we can assign the 0 value, to reflect that the absence of a rating for an item denotes indifference. Assuming that user u did not rate item i, another option is to assign the average value of the provided ratings on other items by u, or the average of the provided ratings on i by all users.

To examine the impact of the selected value, we measured F_1 versus the assigned value, which is depicted in Fig. 5b for the MovieLens 100K data set (it uses 1–5 scale). The dashed line in Fig. 5b corresponds to F_1 of the Pearson correlation (it is independent from the assigned value). As shown, values between the positive threshold (in this case P_τ was set to 3) and the maximum rating of the scale, result to reduced F_1 (notice that this range also includes the user average rating value). The reason is that these values impinge the ability to distinguish the assigned values from the actual positive ratings. However, when we assign values smaller than P_τ or *outside* the rating scale, F_1 is not affected. The reason is that with such assigned values we do not miss the ability to distinguish the assigned values from the actual positive ratings (as the latter are always within the provided scale). Thus, we conclude that UNION is not significantly affected by the selection for the assigned ratings, as all values outside the rating scale or below P_τ result to about the same F_1. Even more, the values between P_τ and the upper limit of the scale result to significantly higher F_1 than Pearson measure. Henceforth, we assume that the assigned value is equal to 0.

4.2. Second stage extensions: the HPR and HSR algorithms

In Section 3.2, we described the two criteria, MF and HSS, that have been proposed so far for the generation of the top-N list. The ranking criteria are important, as they can significantly affect the performance of CF. For this reason we examined additional criteria, to see if this direction of research worths investigation.

As a first extension of the existing ranking criteria, someone could use the predicted values for each item to rank them. Predicted values (McLauglin & Herlocher, 2004) are computed by Eqs. (4) and (6), for the cases of user-based and item-based CF, respectively. These equations have been used in related work for the purpose of MAE calculation, whereas we use them for generation of top-N list.

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in U} \mathrm{sim}(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in U} |\mathrm{sim}(u,v)|} \quad (4)$$

$$p_{u,i} = \bar{r}_i + \frac{\sum_{j \in I} \mathrm{sim}(i,j)(r_{u,j} - \bar{r}_j)}{\sum_{j \in I} |\mathrm{sim}(i,j)|} \quad (5)$$

$$\mathrm{MAE} = \frac{\sum_{u=1}^{T} |r_{u,i} - p_{u,i}|}{|T|} \quad (6)$$

Therefore, we sort (in descending order) the items according to predicted rating value, and recommend the first N of them.[5]

This ranking criterion, denoted as highest predicted rated item recommendation (HPR) is influenced by the good accuracy of prediction that existing related work reports through the MAE. HPR opts for recommending the items that are more probable to receive a higher rating. Nevertheless, as already mentioned, MAE is not indicative for the accuracy of recommendation. As our experimental results will demonstrate, HPR presents poor performance. This fact is another indication that MAE alone cannot

[5] If less than N items have positive ratings (i.e., not less than P_τ), then less than N items remain in the list.

characterize the performance of CF algorithms. In Fig. 6a, we represent the aforementioned criterion for the user-based approach.

As another criterion, which resulted from observations during our experimental investigation, we sum the positive rates of the items in the neighborhood, instead of just counting their frequency. This criterion is denoted as highest sum of rates item recommendation (HSR). The top-N list consists of the N items with the highest sum. The intuition behind HSR is that it takes into account both the frequency (as MF) and the actual ratings, because it wants to favor items that appear most frequently in the neighborhood *and* have the best ratings. Assume, for example, an item i that has just a smaller frequency from an item j. If i is rated much higher than j, then HSR will prefer it from i, whereas MF will favor j. In the running example, for test user U_8, for the user-based approach with $k = 3$ and $N = 1$, using this criterion for the export of the top-N list, we recommend finally item I_2 because its sum of rates in the neighborhood is $9(4 + 5)$ instead of item I_3 where its sum of rates is $8(4 + 4)$. If we had used the most frequent top-N list generation these two items would have the same presences (2) in the neighborhood. So, both of them could be proposed. In Fig. 6b, we represent the aforementioned criterion for the user-based approach.

4.3. Third stage extension : a baseline algorithm

For the third stage, we considered all the issues that are described in Section 3.3. One issue that needs further explanation is the definition of a baseline algorithm. We propose the one that recommends the N items that are most frequently rated positively in the entire training set. This algorithm is denoted as BL. BL is very simple and, as will be shown in our experimental results, it is quite effective. For instance, our experiments with Movielens-100K data set have shown that, with BL, when we simply propose the $N = 20$ most positively rated movies (20 most popular movies), precision reaches 40%. We examined these movies and verified that they are not obscure at all, as one can recognize in Fig. 7. Therefore, the most frequently rated items are very probable to be selected by the majority of the users. For the aforementioned reasons, BL is a tool to clearly evaluate the actual improvement of existing CF

a

```
Vector GenTopHPR(u, NN(u), I, N)
//User u
//Set NN(u)
//int I,N
begin
    for i = 1 to I
        f[i] = 0;
    foreach ν ∈ NN(u)
        for i = 1 to I
            if r[ν][i] ≥ Pτ
                nom = nom + sim(u,ν)*(r[ν][i]-r̄_ν);
                denom = denom + abs(sim(u,ν));
                p[i]= r̄_u + nom / denom;
    sort(p); //descending order
    n = 0; i = 1;
    while (n < N and p[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

b

```
Vector GenTopHSR(u, NN(u), I, N)
//User u
//Set NN(u)
//int I,N
begin
    for i = 1 to I
        Sum[i] = 0;
    foreach ν ∈ NN(u)
        for i = 1 to I
            if r[ν][i] ≥ Pτ
                sum[i] += r[ν][i];
    sort(sum); //descending order
    n = 0; i = 1;
    while (n < N and sum[i] > 0)
        A[n] = i;
        n = n + 1;
    return A;
end.
```

Fig. 6. Generation of top-N list based on (a) highest predicted rated (HPR) algorithm and (b) highest sum of rates (HSR).

	Title	Rates			Rates
1	Star Wars (1977)	558	11	Silence of the Lambs, The (1991)	374
2	Fargo (1996)	476	12	Liar Liar (1997)	365
3	Return of the Jedi(1983)	476	13	Twelve Monkeys (1995)	352
4	Contact (1997)	454	14	Pulp Fiction (1994)	351
5	Toy Story (1995)	417	15	Empire Strikes Back, The (1980)	350
6	Raiders of the Lost Ark (1981)	403	16	Jerry Maguire (1996)	342
7	English Patient, The (1996)	398	17	Independence Day (ID4) (1996)	335
8	Scream (1996)	390	18	Rock, The (1996)	332
9	Godfather, The (1972)	384	19	Star Trek: First Contact (1996)	328
10	Air Force One (1997)	380	20	Titanic (1997)	327

Fig. 7. The 20 most positively rated movies of the Movielens 100K dataset.

Please cite this article in press as: Symeonidis, P. et al., Collaborative recommender systems: Combining effectiveness ..., *Expert Systems with Applications* (2007), doi:10.1016/j.eswa.2007.05.013

algorithms. We will show experimentally that, as we increase the k nearest neighbors, the performances of Pearson and adjusted cosine become equivalent to BL.

5. Proposed method for an LSI-based algorithm

Our approach, initially, applies Singular Value Decomposition (SVD) over the user-item matrix A. We tune the number, c, of principal components (i.e., dimensions) with the objective to reveal the major trends. The tuning of c is determined by the information percentage that is preserved compared to the original matrix. Therefore, a c-dimensional space is created and each of the c dimensions corresponds to a distinctive rating trend. Next, given the current ratings of the target user u, we enter pseudo-user vector in the c-dimensional space. Finally, we find the k nearest neighbors of pseudo-user vector in the c-dimensional space and apply either user- or item-based similarity to compute the top-N recommended items. Conclusively, the provided recommendations consider the existence of user rating trends, as the similarities are computed in the reduced c-dimensional space, where dimensions correspond to trends.

To ease the discussion, we will use the running example illustrated in Fig. 8 where I_{1-4} are items and U_{1-4} are users. As shown, the example data set is divided into training and test set. The null cells(no rating) are presented as zeros.

5.1. Applying SVD on training data

Initially, we apply SVD on training data $n \times m$ matrix A that produces three matrices. These matrices obtained by SVD can give by performing multiplication the initial matrix as the following Eq. (7) and Fig. 9 show:

a

	I_1	I_1	I_1	I_1
U_1	4	1	1	4
U_2	1	4	2	0
U_3	2	1	4	5

b

	I_1	I_1	I_1	I_1
U_4	1	4	1	0

Fig. 8. (a) Training set ($n \times m$) and (b) test set.

$$A_{n\times m} = U_{n\times n} \cdot S_{n\times m} \cdot V'_{m\times m} \tag{7}$$

5.2. Preserving the principal components

It is possible to reduce the $n \times m$ matrix S to have only c largest singular values. Then, the reconstructed matrix is the closest rank-c approximation of the initial matrix A as it is shown in Eq. (8) and Fig. 10:

$$A^*_{n\times m} = U_{n\times c} \cdot S_{c\times c} \cdot V'_{c\times m} \tag{8}$$

We tune the number, c, of principal components (i.e., dimensions) with the objective to reveal the major trends. The tuning of c is determined by the information percentage that is preserved compared to the original matrix. Therefore, a c-dimensional space is created and each of the c dimensions corresponds to a distinctive rating trend. We have to notice that in the running example we create a 2-dimensional space using 83% of the total information of the matrix (12,88/15,39). In our experiments we have seen that only a 10% is adequate to provide accurate results.

5.3. Inserting a test user in the c-dimensional space

Related work (Sarwar et al., 2000) has studied SVD on CF considering the test data as apriori known. It is evident that, for user-based approach, the test data should be considered as unknown in the c-dimensional space. Thus a specialized insertion process should be used. Given the current ratings of the test user u, we enter pseudo-user vector in the c-dimensional space using the following Eq. (9) (Furnas et al., 1988). In the current example,we insert U_4 into the 2-dimensional space, as it is shown in Fig. 11:

$$\mathbf{u}_{\text{new}} = \mathbf{u} \cdot V_{m\times c} \cdot S^{-1}_{c\times c} \tag{9}$$

In Eq. (9), $\mathbf{u}_{\text{new}}$ denotes the mapped ratings of the test user $\mathbf{u}$, whereas $V_{m\times c}$ and $S^{-1}_{c\times c}$ are matrices derived from SVD. This $\mathbf{u}_{\text{new}}$ vector should be added in the end of the $U_{n\times c}$ matrix which is shown in Fig. 10. Notice that the inserted vector values of test user U_4 are very similar to these of U_2 after the insertion. This is reasonable, because these two users have similar ratings as it is shown in Fig. 8.

4	1	1	4
1	4	2	0
2	1	4	5

$A_{n\times m}$

-0.61	0.28	-0.74
-0.29	-0.95	-0.12
-0.74	0.14	0.66

$U_{n\times n}$

8.87	0	0	0
0	4.01	0	0
0	0	2.51	0

$S_{n\times m}$

-0.47	-0.28	-0.47	-0.69
0.11	-0.85	-0.27	0.45
-0.71	-0.23	0.66	0.13
-0.52	0.39	-0.53	0.55

$V'_{m\times m}$

Fig. 9. Example of: $A_{n\times m}$ (initial matrix A), $U_{n\times m}$ (left singular vectors of A), $S_{n\times m}$ (singular values of A), $V'_{m\times m}$ (right singular vectors of A).

2.69	0.57	2.22	4.25
0.78	3.93	2.21	0.04
3.17	1.38	2.92	4.78

$A^*_{n\leftarrow i}$

-0.61	0.28
-0.29	-0.95
-0.74	0.14

$U_{n\leftarrow c}$

8.87	0
0	4.01

$S_{c\leftarrow c}$

-0.47	-0.28	-0.47	-0.69
0.11	-0.85	-0.27	0.45

$V'_{c\leftarrow m}$

Fig. 10. Example of: $A^*_{n\times m}$ (approximation matrix of A), $U_{n\times c}$ (left singular vectors of A^*), $S_{c\times c}$ (singular values of A^*), $V'_{c\times m}$ (right singular vectors of A^*).

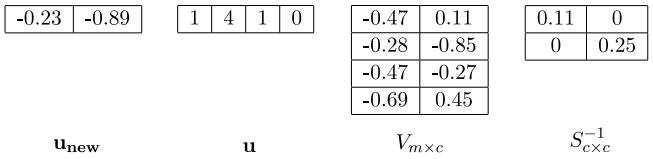

Fig. 11. Example of: $\mathbf{u}_{\text{new}}$ (inserted new user vector), $\mathbf{u}$ (user vector), $V_{m\times c}$ (two left singular vectors of V), $S^{-1}_{c\times c}$ (two singular values of inverse S).

Note that this process is omitted in the item-based approach, which means that this process is more applicable in real applications in terms of complexity. In our experiments, we will present this drawback of the user-based approach in terms of execution time.

5.4. Generating the neighborhood of users/items

Having a reduced dimensional representation of the original space, we form the neighborhoods of users/items in that space.

For the user based approach, we find the k nearest neighbors of pseudo-user vector in the c-dimensional space. The similarities between training and test users can be based on cosine similarity. First, we compute the matrix $U_{n\times c}\cdot S_{c\times c}$ and then we perform vector similarity. This $n\times c$ matrix is the c-dimensional representation for the n users. Note that in our experiments, we generate – as a second approach – the similarity matrix taking into account only $U_{n\times c}$ matrix. As we will see, results are quite impressive, but have to do only with the user- based approach.

For the item based approach, we find the k nearest neighbors of item vector in the c-dimensional space. First, we compute the matrix $S_{c\times c}\cdot V_{c\times m}$ and then we perform vector similarity. This $c\times m$ matrix is the c-dimensional representation for the m items.

5.5. Generating the recommendation list

As it is mentioned in Section 3.2, existing ranking criteria, such as MF, are used for the generation of the top-N list in classic CF algorithms. We propose a ranking criterion that uses the predicted values of a user for each item. Predicted values are computed by Eqs. (4) and (6), for the cases of user-based and item-based CF, respectively. These equations have been used in related work for the purpose of MAE calculation, whereas we use them for generation of top-N list.

Therefore, we sort (in descending order) the items according to predicted rating value, and recommend the first N of them.[6] This ranking criterion, denoted as highest predicted rated item recommendation (HPR), is influenced by the good accuracy of prediction that existing related work reports through the MAE. HPR opts for recommending the items that are more probable to receive a higher rating. As our experimental results will demonstrate, HPR presents poor performance for the classic CF algorithms, but dramatically spectacular results when it is used in combination with SVD. The reason is that in the latter it is based only on the major trends of users.

[6] If less than N items have positive ratings (i.e., not less than P_τ), then less than N items remain in the list.

5.6. Evaluation of the CF process

Related work (Sarwar et al., 2000) proposed Eq. (10) for the generation of predictions, where $\oplus$ means the addition of a scalar with a matrix.

$$p_{u,i} = \bar{r}_u \oplus U_{n\times c}\cdot\sqrt{S_{c\times c}}\sqrt{S_{c\times c}}\cdot V'_{c*m} \tag{10}$$

We test this equation and find out that the predicted values were calculated not from other users but from the user himself. So, the information of an inserted in c-dimensional space test user was used to predict his own real rates. These predicted values were so close to the real ones. Therefore, although the produced MAE may be good, this procedure is not prediction, because the test user is considered apriori known. In contrast, we use Eqs. (4) and (6) for prediction, to exploit information from other users or items. Thus, we use these predicted values for the calculation of MAE.

6. Performance study

In the sequel, we study the performance of the described extensions against existing CF algorithms, by means of a thorough experimental evaluation. Both user-based and item-based algorithms are tested. Moreover, we study the performance of the described SVD dimensionality reduction techniques against existing CF algorithms. Several factors are considered, like the sparsity, the similarity measures, and criteria for generating the top-N list.

The additional factors, that are treated as parameters, are the following: the neighborhood size (k, default value 10), the size of the recommendation list (N, default value 20), the size of training set (default value 75%), and the division between past/future data. Regarding WS, the γ value was set to 5. Evaluation is performed with the precision and recall metrics (given as percentages). We also use F_1 metric.

We perform experiments with four real data sets that have been used as benchmarks in prior work. In particular, we examined two MovieLens data sets: (i) the first one with 100,000 ratings assigned by 943 users on 1682 movies (This is the default data set. Any use of other data sets is categorically defined), and (ii) the second one with about 1 million ratings for 3592 movies by 6040 users. The range of ratings is between 1(bad)-5(excellent) of the numerical scale. Moreover, we ran our experiments on the EachMovie data set (McJones & DeTreville, 1997), which contains 2,811,983 ratings entered by 72,916 for 1628 different movies, and the Jester data set, which contains 4.1 million ratings of 100 jokes from 73,496 users. Finally, in all data sets, we normalized the rating scale in the range 1–5, whereas P_τ

is set to 3 and the value of an unrated item is considered equal to zero.

6.1. Results for user-based CF algorithms

First, we examine user-based CF algorithms and compare the existing Pearson similarity and WS measures against the proposed UNION method. We also include the baseline (BL) algorithm. The results for precision and recall vs. k are displayed in Fig. 12a and b, respectively.

As shown, the existing Pearson measure, which is based on co-rated items, performs worst than BL. This result is surprising, as BL is very simple. WS improves Pearson, because the disadvantage of Pearson, due to co-rated items, is downsized by the weighting with the number of common items. UNION clearly outperforms all other measures. The reason is that the MovieLens data set is sparse and these measures better exploit the available information. UNION reaches an optimum performance for a specific k. In contrast, in the examined range of k values, the performance of Pearson increases with increasing k. Outside the examined k range (not displayed), it stabilizes and never exceeds BL. We have to notice here, that as we increase the number of k nearest neighbors Pearson measure is literally transformed to BL.

We now examine the MAE metric. Results are illustrated in Fig. 13a (BL is only for recommendation, not prediction, thus omitted). As expected, Pearson yields the lowest MAE values, whereas WS is second best. This fact supports our explanation that MAE is indicative only for the evaluation of prediction and not of recommendation, as these measures did not attain the best performance in terms of precision and recall.

To consider the impact of density, we also examine the Jester data set. The results for the F_1 metric are depicted in Fig. 13b. In this case, the relative differences are smaller than for the case of sparse data. The reason is that dense data have a sufficient amount of information, thus there is less need to exploit information in the way UNION does.

Finally, we test the described criteria for the top-N list generation: MF, HPR, and HSR. We used the UNION similarity measure, because it presented the best performance in the previous measurements. The results for precision and recall are given in Fig. 14a and b, respectively.

In Fig. 14a it is shown that HPR is clearly outperformed by the other two criteria, a fact that furthermore illustrates the unsuitability of the MAE metric to characterize the task of recommendation. On the other hand, MF and HSR present similar precision. Fig. 14b presents the results on recall (to make comparison between MF and HSR more clear, we omit HPR). HSR is constantly better than MF, though slightly.

6.2. Results of LSI application on user-based CF algorithms

Regarding the performance of SVD dimensionality reduction in user-based approach, we preserve, each time, a different fraction of principal components of SVD model. More specifically, we preserve 90%, 70%, 50%, 30% and

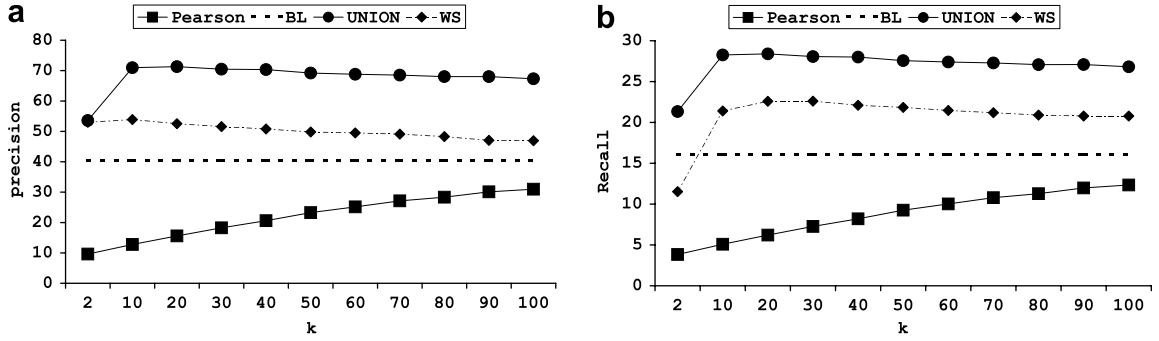

Fig. 12. Performance of user-based CF vs. k: (a) precision and (b) recall.

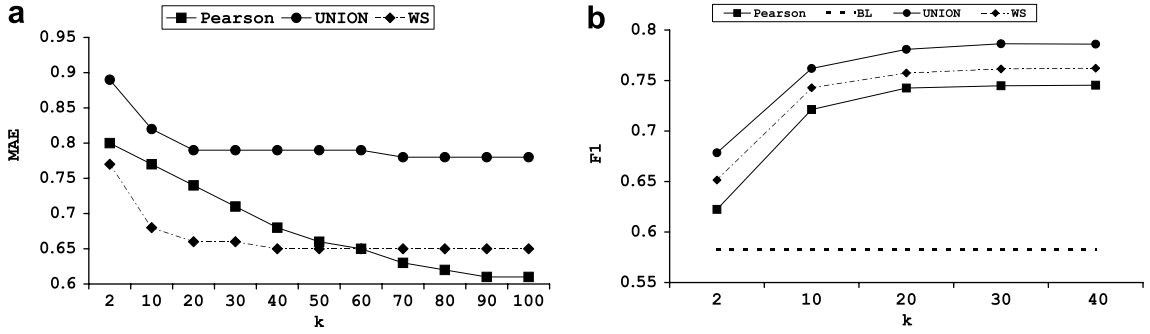

Fig. 13. Performance of user-based CF vs. k: (a) MAE and (b) F_1 for dense data.

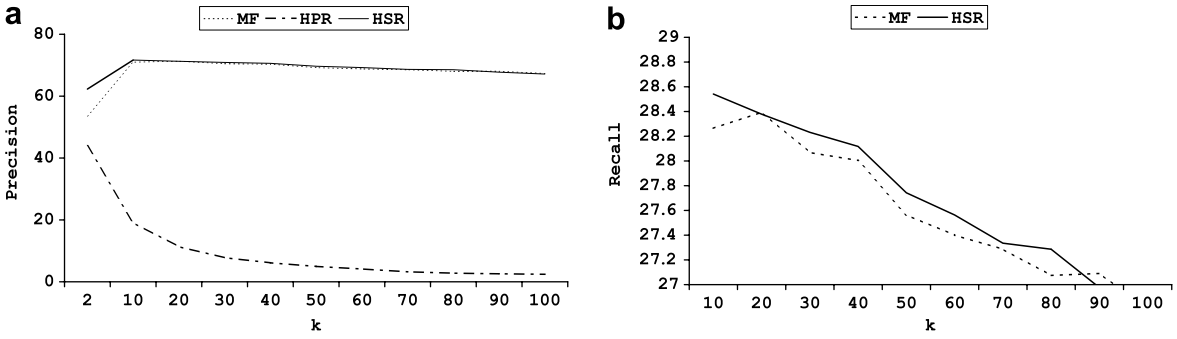

Fig. 14. Comparison of criteria for the generation of top-N list for user-based CF vs. k: (a) precision and (b) recall.

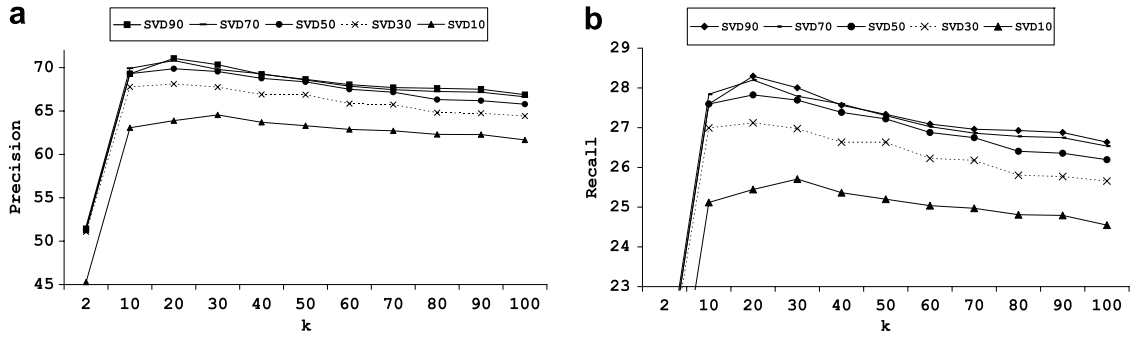

Fig. 15. Performance of SVD dimensionality reduction of user-based CF vs. k: (a) precision and (b) recall.

10% of the total information of initial user-item matrix. The results for precision and recall vs. k are displayed in Fig. 15a and b, respectively.

As we can see, the performance of SVD50, SVD70, SVD90 are similar in terms of precision and recall. The reason is that SVD50 is adequate for producing a good approximation of the original matrix. Thus, we will continue our study with two representative SVD models, SVD50 and SVD10. These choices are indicative of the behavior of the SVD model.

We now move on to comparison of existing user-based CF algorithm that uses Pearson similarity against the two representative SVD reductions. The results for precision and recall vs. k are displayed in Fig. 16a and b, respectively.

As shown, the existing Pearson measure, which is based on co-rated items, performs worst than SDV reductions. The two proposed reductions clearly outperform Pearson measure. We have to notice that SVD50 perfroms equally with the UNION algorithm. The reason is that the MovieLens data set is sparse and relatively large (high n value). The SVD reductions reveal the most essential dimensions and filter out the outliers and misleading information. So, with less information, we have almost the same result. SVD50 performs a little better than SVD10, as it uses more information. In contrast, the latter SVD reduction uses only 11 dimensions instead of 157 of the former. This means that the performance of the latter is faster than the former one.

Now, we compare the SVD50 and UNION algorithms with the variation we described in Section 5.4 denoted as SVDU50. As we can see, SVDU50 performs even better than the UNION algorithm. We propose it, as a second way for producing the users' similarity matrix. The position of the points between $U_{n\times c} \cdot S_{c\times c}$ matrix and only the

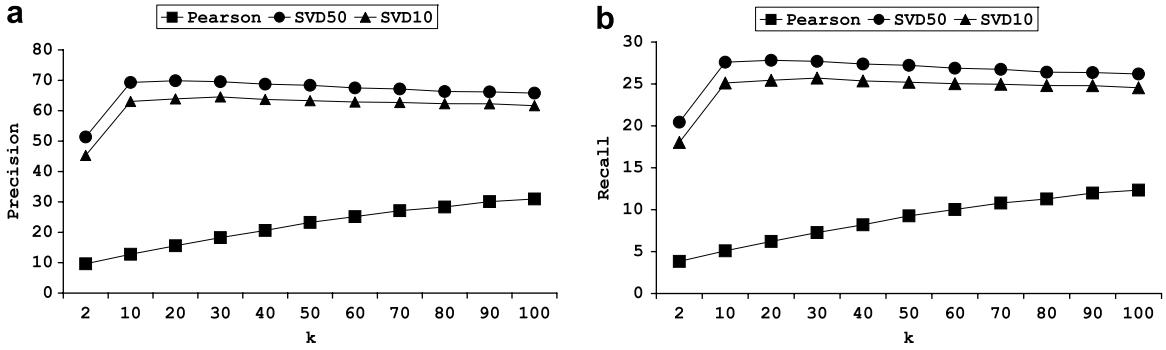

Fig. 16. Performance of user-based CF vs. k: (a) precision and (b) recall.

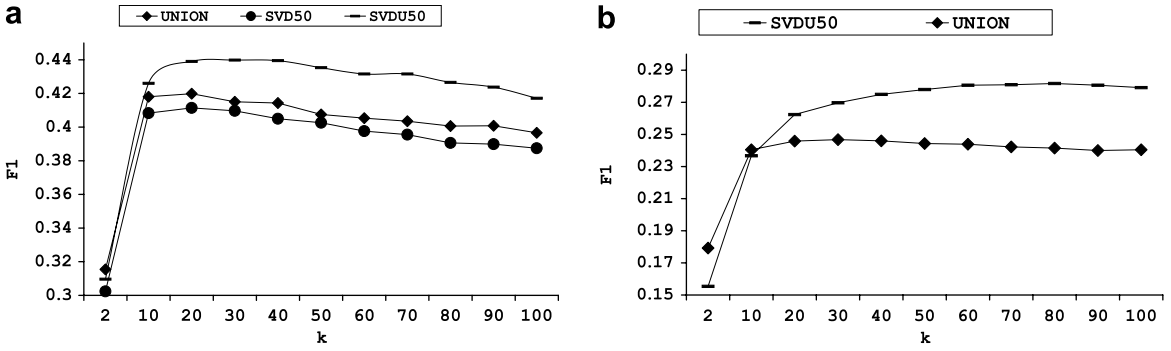

Fig. 17. Performance of user-based CF vs. k: (a) F_1 for 100k Movielens data set and (b) F_1 for 1M Movielens data set.

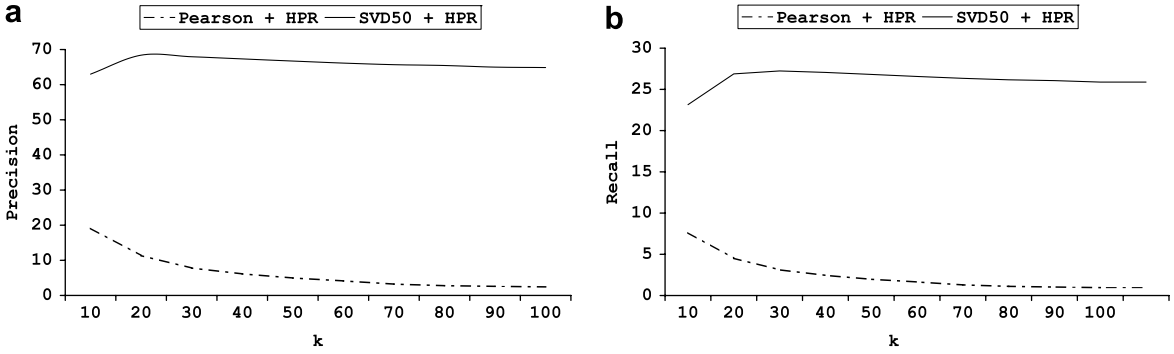

Fig. 18. Comparison HPR criteria for the generation of top-N (a) precision and (b) recall list for user-based CF vs. k

$U_{n\times c}$ matrix are the same except that the former is stretched or shrunk in proportion to the corresponding diagonal elements of $S_{c\times c}$ matrix.

To consider the impact of scalability, we also examine the 1M data set. The results for the F_1 metric are depicted in Fig. 17b. In this case, the relative difference broadens as k increases, than the case of 100K data set. The reason is that 1M data set is more sparse (the percentage of non rated items exceeds 95%) than 100K data set (the corresponding percentage is 93%). Moreover, the rank (i.e., number of independent dimensions) of the 1M data set is 3010 instead of 943 for the 100K. This means, that there are many dependent dimensions in the former that can be compressed.

Finally, we test the described criteria for the HSR top-N list generation algorithm. The results for precision and recall are given in Fig. 18. As shown, the combination of the SVD similarity measure with HPR as list generation algorithm, clearly outperforms the Pearson with HPR. This is due to the fact that in the former the remaining dimensions are the determinative ones and outliers users have been rejected. Note that in the SVD50 we preserve only 157 basic dimensions instead of 943 for the latter.

6.3. Results for item-based CF algorithms

We perform similar measurements for the case of item-based CF. Thus, we first examine the precision and recall for the existing adjusted cosine measure (considers co-rated items) against UNION for the item-based case (that consider not only co-rated items through the extended definitions of T set). The results are depicted in Fig. 19 and are analogous to those of the user-based case. UNION clearly outperforms adjusted cosine and WS. Again, it is surprising to find the item-based CF algorithm with the adjusted cosine measure looses out by BL.

Next, we compare adjusted cosine, UNION and WS against MAE. The results are illustrated in Fig. 20a. Differently from the case of user-based CF, all measures have similar MAE, and for larger k values (where all of them reach the optimum MAE value) UNION and the adjusted cosine are equal. The reason that adjusted cosine does not present better MAE, is that in its denominator it considers all items and not just the co-rated ones (see Eq. (2)). This improves its performance for the task of recommendation and worsens the performance of prediction.[7]

Regarding the examination of the dense data set (Jester), the results for the F_1 metric are illustrated in Fig. 20b. Since item-based CF has been designed to suit the needs of sparse data, we find out that for dense data all item-based algo-

[7] We have examined a variation of adjusted cosine that uses only the co-rated items in the denominator. As expected, it resulted to worse precision and recall, but to better MAE.

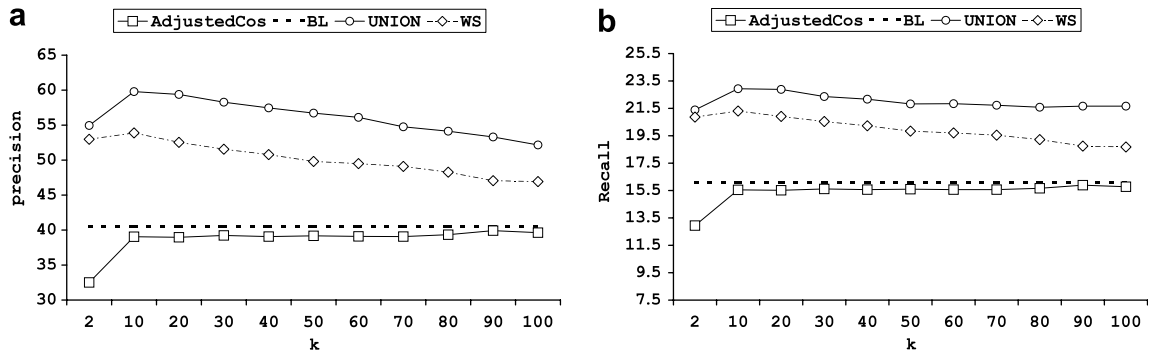

Fig. 19. Performance of item-based CF vs. k: (a) precision and (b) recall.

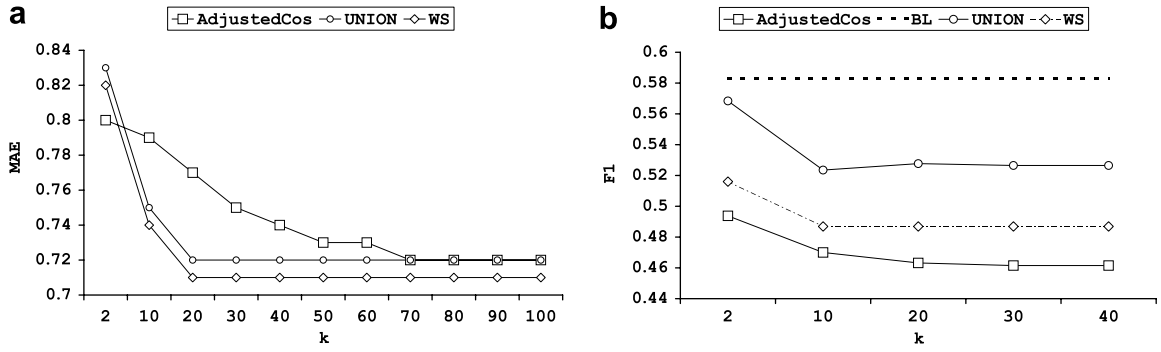

Fig. 20. Performance of item-based CF vs. k: (a) MAE and (b) F_1 for dense data.

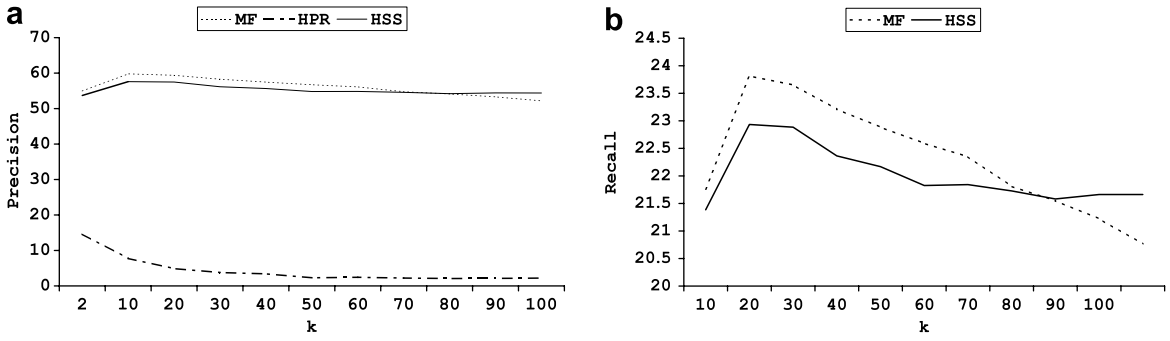

Fig. 21. Comparison of criteria for the generation of top-N list for item-based CF vs. k: (a) precision and (b) recall.

rithms are outperformed by BL. This is the case even for UNION, although it performs better than adjusted cosine. This result clarifies the need to examine CF algorithms for all the involved factors, in this case density, in order to draw more complete conclusions.

Finally, we test the three described criteria, MF, HPR, and HSS, for the top-N list generation by item-based CF algorithms. The results for precision are illustrated in Fig. 21a. Similarly to the user-based case, HPR performs very poorly. Fig. 21b shows the resulting recall (HPR is omitted to make comparisons more clear). MF performs a little better than HSS. However, no clear winner can be identified, because for the other data sets HSS performed a little better.

6.4. Results of LSI application on item-based CF algorithms

Regarding the performance of SVD dimensionality reduction in item-based approach, we first examine the precision and recall for the existing adjusted cosine Measure (considers co-rated items) against SVD50 and SVD10 for the item-based case. The results are depicted in Fig. 22 and are analogous to those of the user-based case. SVD50 and SVD10 clearly outperform adjusted cosine. Notice that unlike the user-based case, the difference between SVD50 and SVD10 is greater. This mean that item based algorithm cannot preserve accuracy in a satisfactory way when we decrease the percentage of dimension's information.

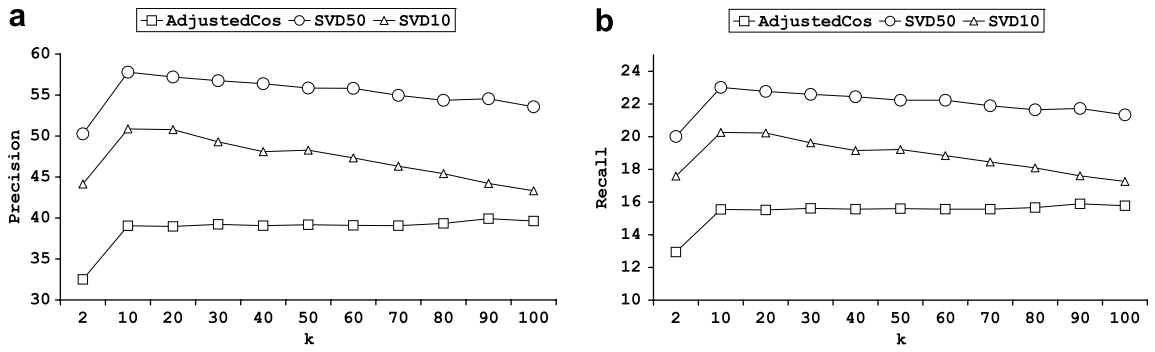

Fig. 22. Performance of item-based CF vs. k: (a) precision and (b) recall.

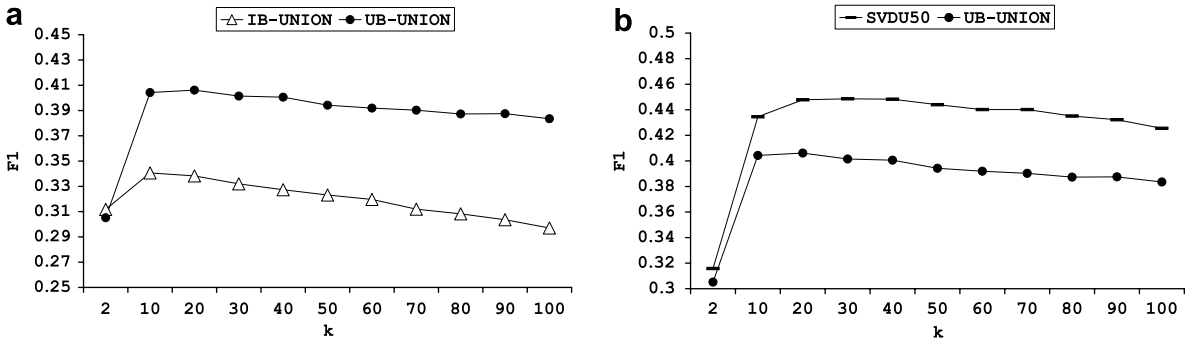

Fig. 23. Comparison between item-based and user-based CF vs. k.

6.5. Comparative results in terms of effectiveness

In this section, we compare user-based and item-based CF using the best options as they have resulted from the previous measurements. Therefore, for user-based and item-based we use the UNION similarity measure. With respect to the criterion for the generation of the top-N list, for user-based we use HSR, whereas for the item-based we use MF. The result for $F1$ metric are displayed in Fig. 23a.

These results demonstrate that user-based CF, when using the best options, compares favorably against item-based CF, even when the latter uses the best options. The difference in $F1$ is larger than 0.3. The difference with respect to precision is larger of 10%, whereas with respect to recall, it exceeds 5% (we refer to the optimum values resulting from the tuning of k). This conclusion contrasts the existing one, that item-based CF is preferable than user-based for sparse data. The reason is that user-based CF is more focused towards the preferences of the target user. In contrast, with item-based CF, the recommended items may have been found similar by transactions of users with much different preferences than the ones of the target user. Thus, they may not directly reflect the preferences of the latter. The differences that have been reported in prior work can only be explained by the fact that the algorithms did not use the best possible options. In Fig. 23b, we pres-

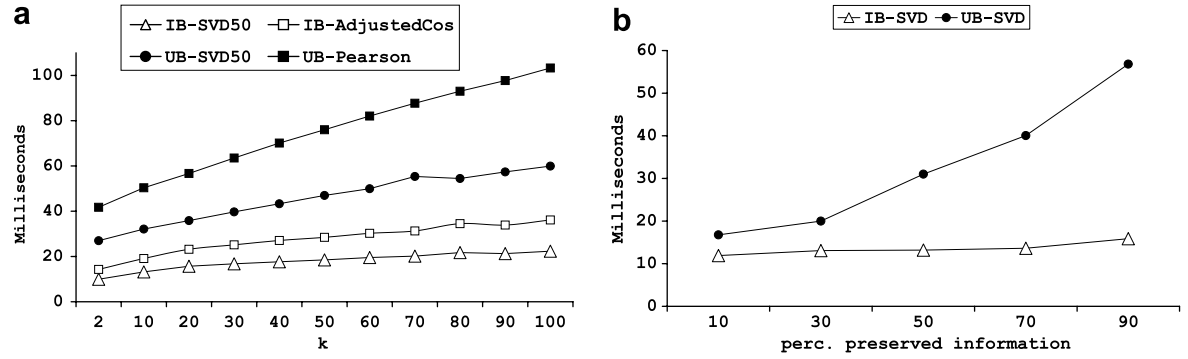

Fig. 24. Comparison between item-based and user-based CF in terms of execution time: (a) time vs. k and (b) time vs. percentage of preserved information.

ent again -for reasons of clearness- the comparison between UNION algorithm with SVDU50. As we can see, SVDU50 performs favorably against UNION algorithm. The reason is that the SVD reduction reveals the most essential dimensions and filters out the outliers and misleading information. So, by applying LSI in user-based algorithm we attain the best possible result in terms of accuracy.

6.6. Comparative results in terms of efficiency

Regarding the execution time, we measured the wall-clock time for the on-line parts of the user-based and item-based algorithms. The results regarding the time needed for a user recommendation vs. k are presented in Fig. 24a, whereas the results for varying percentage of preserved information, are depicted in Fig. 24b (in the latter measurement we set $k = 10$).

As already mentioned, item based CF needs less time to provide recommendations than user-based CF. This holds for both the aforementioned measurements. The reason is that a user-rate vector in user-based approach has to be inserted in the c-dimensional space. Moreover, we have to mention that the generation of top-N list for the user-based approach further burdens the CF process. The reason is that the algorithm finds, firstly, user neighbors in the neighborhood matrix and then counts presences of items in the user-item matrix. In contrast, with the item-based approach the whole work is completed in the item neighborhood matrix. So, in terms of execution item based approach is superior over user based.

6.7. Examination of the additional factors

In this section we examine the impact of the additional factors. In our measurements we consider the existing two cases, that is, user-based CF with Pearson similarity and item-based CF with adjusted cosine (both the two existing cases consider co-rated items).

First, we examine the impact of N (recommendation list's size). The results are depicted in Fig. 25. As expected, with increasing N, recall increases and precision decreases. The relative differences between the algorithms are coherent with those in our previous measurements.

Next, 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 the F_1 metric are given in Fig. 26a (to make the graph more clear, we show results only for the two best user-based and item-based algorithms). As expected, when the training set is small, performance downgrades for both algorithms. Therefore, we should be careful enough when we evaluate CF algorithms and use adequately large training sets. Similar to the previous measurements, in all cases user-based UNION is better than item-based UNION. The performance of both reaches a peak around 75%, after which it reduces. The reason for this is the overfitting that results from very large training sets.

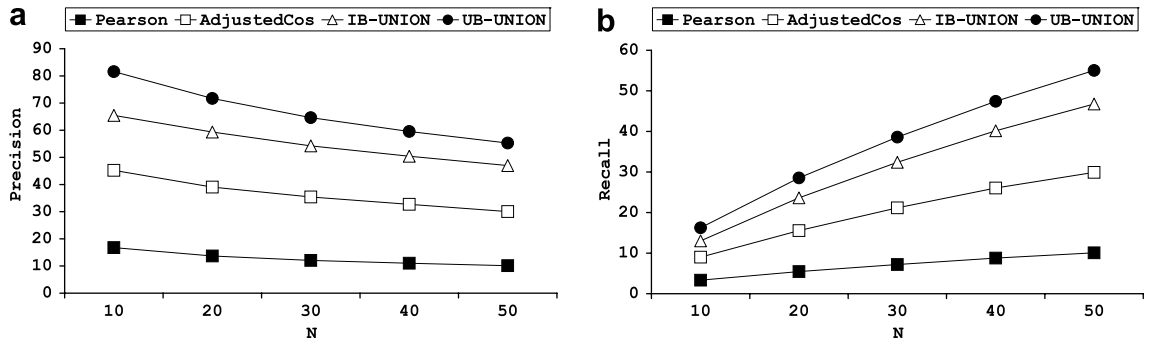

Fig. 25. Comparison vs. N: (a) precision and (b) recall.

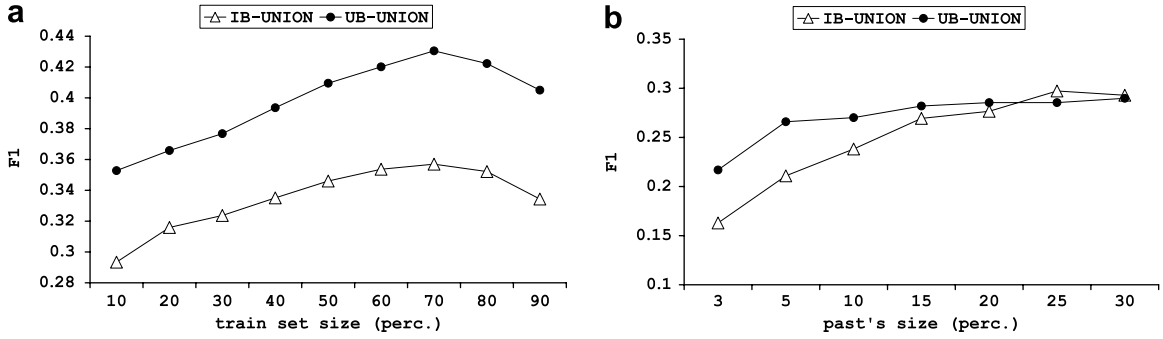

Fig. 26. (a) Comparison vs. training set size and (b) comparison vs. past's size.

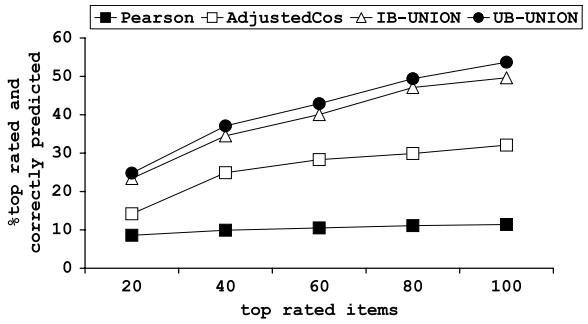

Fig. 27. Measuring the percentage of items that are correctly predicted and top rated.

We also examine the performance when considering the division between past and future data. For each user's transaction in the test set we keep the 70% as future data and use a varying number of ratings from the rest 30% as past data. This way, we examine the impact of past's size. We compare user-based UNION with item-based UNION using the F_1 metric. The results are illustrated in Fig. 26b. As the size of the considered past reduces, the performance of both algorithms reduces. This result demonstrates the need to evaluate CF algorithms using the division between past and future data, because this division is more indicative about the actual performance in real-world CF applications. Nevertheless, user-based UNION manages to keep a higher F_1 value for small past size, whereas both algorithms converge to the same point for larger past. The merit of user-based UNION is evident, as in real-world applications the past size usually takes very small values (users do not have to wait long enough before start receiving recommendations).

Finally, we test the examined algorithms with respect to the quality of the provided recommendations. The results are presented in Fig. 27. In particular, we are interested in determining whether these algorithms provide obscure recommendations or not. For this reason, we sort the items according to the number of positive ratings (i.e., higher than P_τ) they received. Each time we keep a varying number of these top rated items (e.g., the top 20, 40, etc.). These top rated items are considered as more popular, thus they do not comprise obscure recommendations. We measure the percentage of the top rated items that are *correctly* predicted by each CF algorithm.[8] This measurement indicates how much of the correct predictions belong to the most rated items.

As the number of considered top rated items increases, the Pearson similarity measure (which is based on co-rated items) does not increase the percentage of the correctly predicted items that belong to these top rated items. This means that the recommendations of this algorithm tend to be outside the range of the top rated items. Thus, Pearson recommends more obscure items. The adjusted cosine similarity measure presents a small increase, which means that it presents the latter problem to a smaller degree. In contrast, both user-based UNION and item-based UNION clearly present the best percentage and their resulting slope is high. This means that these algorithms tend to avoid obscure items during their recommendations and concentrate on the first positions of the top rated items.

[8] The alternative of just counting the predictions, regardless if they are correct, could provide misleading results and, thus, it is avoided.

7. Conclusions

We have performed a thorough study of neighborhood-based CF, which brought out several factors that have not been examined carefully in the past. Based on our observations, we proposed extensions of similarity measures, new criteria for generating the recommendation list, and new ways to evaluate the quality of the recommendation result. Additionally to these extensions, we proposed an LSI-based approach combining simultaneously effectiveness and efficiency of CF algorithms. We performed experimental comparison of our proposed method against well known CF algorithms, like user-based or item-based methods, with real data sets. Our experimentation reforms several existing beliefs and provides new insights.

In particular, we highlight the following conclusions from our examination:

- In contrast to what is reported in majority of related work, MAE is not indicative for the accuracy of the recommendation process. It is, though, useful to characterize the quality of the similarity measure (as reflected in the process of prediction).
- Constraining similarity measures with co-rated items, weaknesses the measure. Though it is somewhat useful to consider the number of co-rated items (as WS does), the strict constraining inside the formulae for similarity measures is not suitable.
- The proposed extensions that do not use co-rated items only, substantially improve the performance of CF, especially for sparse data, because they exploit more information. Actually, the existing approaches based on co-rated items are outperformed by a simple baseline algorithm we developed.
- Our results showed that, following the best options for user-based and item-based CF, the former compares favorably to the latter. This contrasts with existing results in related work, because until now, comparison did not follow the best options we describe.
- Our results have shown that item-based CF is not appropriate for dense data.
- Our LSI-based significantly outperforms existing CF algorithms in terms of accuracy (measured through recall/precision). Moreover, in some cases, it can slightly improve UNION. The reason is that it is able to identify more clearly the correct recommended items by focusing on trends and isolating noisy users (e.g., outliers). In terms of execution times, due to the use

of smaller matrices, execution times are substantially reduced.
- In our experiments we have seen that only a 10% of the original matrix is adequate to provide accurate results.
- The proposed HPR criterion in combination with SVD can compete the existing MF, which has been used by the majority of related work.
- The process of folding-in is omitted in the item-based approach. Thus, item-based algorithms are faster than user-based ones. For this reason they may be more appropriate for real-world applications, despite their worse accuracy, since execution time is sine-qua-non in realistic deployments.

Summarizing the aforementioned conclusions, we see that, on one hand, item-based algorithms are more appropriate for off-line computations and, thus, better in on-line response. On the other hand, user-based algorithms can produce more effective recommendations with a small fraction of initial information and efficient responding time. With the proposed approaches we combine effectiveness and efficiency in contrast to the classic CF item and user-based algorithms for each approach individually. In our future work we will consider the issue of an approach that would unify the best characteristics of these two cases in an integrated approach.

References

Berry, M., Dumais, S., & O'Brien, G. (1994). Using linear algebra for intelligent information retrieval. *SIAM Review, 37*(4), 573–595.

Breese, J., Heckerman, D. & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the conference on uncertainty in artificial intelligence* (pp. 43–52).

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

Furnas, G., Deerwester, S. & Dumais, S., et al. (1988). Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the ACM SIGIR conference* (pp. 465–480).

Goldberg, D., Nichols, D., Brian, M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *ACM Communications, 35*(12), 61–70.

Goldberg, K., Roeder, T., Gupta, T., & Perkins, C. (2001). Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval, 4*(2), 133–151.

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.

Huang, Z., Chen, H., & Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems, 22*(1), 116–142.

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

McJones, P. & DeTreville, J. (1997). Each to each programmer's reference manual. *Tech. Rep. Systems Research Center*, 1997-023.

McLaughlin, 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).

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).

O'Mahony, M., Hurley, N., Kushmerick, N., & Silvestre, G. (2004). Collaborative recommendation: a robustness analysis. *ACM Transactions on Internet Technology, 4*(4), 344–377.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering on netnews. In *Proceedings of the conference computer supported collaborative work* (pp. 175–186).

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

Sarwar, B., Karypis, G., Konstan, J. & Riedl J. (2000). Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD workshop*.

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

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2002). Incremental singular value decomposition algorithms for higly scalable recommender systems. In *Proceedings of the computer and information technology*.

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).

Please cite this article in press as: Symeonidis, P. et al., Collaborative recommender systems: Combining effectiveness ..., *Expert Systems with Applications* (2007), doi:10.1016/j.eswa.2007.05.013