Scalable Collaborative Filtering based on Latent Semantic Indexing*

Panagiotis Symeonidis and **Alexandros Nanopoulos** and **Apostolos Papadopoulos** and **Yannis Manolopoulos**
Aristotle University, Department of Informatics, Thessaloniki 54124, Greece
symeon, alex, apostol, manolopo@delab.csd.auth.gr

Abstract

Nearest-neighbor collaborative filtering (CF) algorithms are gaining widespread acceptance in recommender systems and e-commerce applications. User ratings are not expected to be independent, as users follow trends of similar rating behavior. In terms of Text Mining, this is analogous to the formation of higher-level concepts from plain terms. In this paper, we propose a novel CF algorithm which uses Latent Semantic Indexing (LSI) to detect rating trends and performs recommendations according to them. We perform an extensive experimental evaluation, with two real data sets, and produce results that indicate its superiority over existing CF algorithms.

Introduction

The "information overload" problem affects our everyday experience while searching for valuable knowledge. To overcome this problem, we often rely on suggestions from others who have more experience on a topic. In Web case, this is more manageable with the introduction of Collaborative Filtering (CF), which provides recommendations based on the suggestions of users who have similar preferences.

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. Related research has reported that memory-based algorithms (a.k.a. nearest-neighbor algorithms) present excellent performance, in terms of accuracy. Their basic drawback is that they cannot handle scalability and sparsity. This means that they face performance problems, when the volume of data is extremely big and sparse.

Latent Semantic Indexing (LSI) has been extensively used in informational retrieval, to detect the latent semantic relationships between terms and documents. LSI constructs a low-rank approximation to the term-document matrix. As a result, it produces a less noisy matrix, which is better than the original one. Thus, higher level concepts are generated from plain terms. In CF, this is analogous to the formation of users' trends from individual preferences.

*This work is conducted while the first two authors were scholars of the State Scholarships Foundation of Greece (IKY).

In this paper, we propose a new algorithm that is based on LSI to produce a condensed model for the user-item matrix. This model comprises a matrix that captures the main user trends and presents a two-fold advantage: (i) it removes noise by focusing on main rating trends and not on particularities of each individual user, (ii) its size is much smaller than the original matrix, thus it can speedup the searching for similar users/items.

Our contribution and novelty are summarized as follows: (i) based on Information Retrieval, we include the pseudo-user concept in order to compare it with our processed data. This differs our method from related work (Sarwar *et al.* 2000b), where Singular Value Decomposition (SVD) methods have used only to summarize the user-item matrix for dimensionality reduction. (ii) We implement a novel algorithm, which tunes the number of principal components according to the data characteristics. (iii)We generalize the recommendation procedure for both user- and item-based CF methods. (iv) We generate predictions based on the users' neighbors and not based on the test user itself, as it has been reported in related work so far. (v) We propose a new top-N generation list algorithm based on SVD and the Highest Prediction Rated items.

The rest of this paper is organized as follows. We summarize the related work and analyze the CF factors. We describe the proposed approach and give experimental results. Finally, we conclude this paper.

Related work

In 1992, the Tapestry system (Goldberg *et al.* 1992) introduced Collaborative Filtering (CF). In 1994, the GroupLens system (Resnick *et al.* 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. In 2001, another CF algorithm was proposed. It is based on the items' similarities for a neighborhood generation of nearest items (Sarwar *et al.* 2001; Karypis 2001) and is denoted as item-based CF algorithm.

All aforementioned algorithms are memory-based. Their basic drawback is that they cannot handle scalability. 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,

Heckerman, & Kadie 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 et al. (Furnas, Deerwester, & 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 and the documents. SVD is a well-known factorization technique that factors a matrix into three matrices. Berry et al. (Berry, Dumais, & 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[2] model is less noisy than the original data.

Sarwar et al. (Sarwar *et al.* 2000b; 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. (Sarwar *et al.* 2000b; 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 et al. (Goldberg *et al.* 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 (Hofmann 2004) proposed a model-based algorithm which relies on latent Semantic and statistical models.

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. Table 1 summarizes the symbols that are used in the sequel.

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. In 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 *et al.* 2004).

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

[2]We use the term "Dimension" as it is defined in Linear Algebra.

Symbol	Definition
k	number of nearest neighbors
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
$\mathcal{I}_u$	set of items rated by user u
$\mathcal{U}_i$	set of users rated item i
$r_{u,i}$	the rating of user u on item i
$\overline{r}_u$	mean rating value for user u
$\overline{r}_i$	mean rating value for item i
$p_{u,i}$	predicted rate for user u on item i
c	number of singular values
A	original matrix
U	Left singular vectors of A
S	Singular values of A
V'	Right singular vectors of A
A^*	Approximation matrix of A
$\mathbf{u}$	user vector
$\mathbf{u_{new}}$	inserted user vector
n	number of training users
m	number of items

Table 1: Symbols and definitions.

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. For this reason dimensionality reduction techniques have been proposed.

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). From our study we concluded that an adequate training-set size should be chosen in order to provide safe results.

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.* 2000a), 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.

Similarity measure: The most extensively used similarity measures are based on correlation and cosine-similarity (Herlocker, Konstan, & Riedl 2002; Sarwar *et al.* 2001). Specifically, user-based CF algorithms mainly use Pearson's Correlation (Equation 1), whereas for item-based CF algorithms, the Adjusted Cosine Measure is preferred (Equation 2) (McLauglin & Herlocher 2004; Sarwar *et al.* 2001). 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, whereas for item-based we use the Adjusted Cosine Similarity, because they presented the best behavior overall.

$$\text{sim}(u,v) = \frac{\sum_{\forall i \in S}(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{\forall i \in S}(r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_{\forall i \in S}(r_{v,i} - \overline{r}_v)^2}}, S = \mathcal{I}_u \cap \mathcal{I}_v. \quad (1)$$

$$\text{sim}(i,j) = \frac{\sum_{\forall u \in T}(r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{\forall u \in \mathcal{U}_i}(r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_{\forall u \in \mathcal{U}_j}(r_{u,j} - \overline{r}_u)^2}}, T = \mathcal{U}_i \cap \mathcal{U}_j. \quad (2)$$

Pearson's Correlation (Equation 1) takes into account only the set of items, S, that are *co-rated* by both users. Moreover, in Equation 1 mean ratings $(\overline{r}_v, \overline{r}_u)$ in numerator are computed only from co-rated items. Similarly, the Adjusted Cosine Measure (Equation 2) considers in the numerator only the set of users, T, that co-rated both the examined pair of items, whereas means are taken over all ratings for a user, not a subset of ratings shared with any other user. In contrast, the denominator of Equation 2 does not consider only the users that co-rated both the items.

Second stage factors

Recommendation list's size: The size, N, of the recommendation list is important for the quality of the system's accuracy. In related work (Karypis 2001; Sarwar *et al.* 2001), N usually takes values between 10 and 50. Actually, N should depend on the average number of rates given by each user. That is, N is application dependent.

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.* 2000a). Henceforth, this technique is denoted as Most-Frequent item recommendation (MF). MF can be applied to both user-based and item-based CF algorithms.

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.

Third stage factors

Evaluation Metrics: For the evaluation of CF algorithms several metrics have been used in related work (Herlocker, Konstan, & Riedl 2002; Herlocker *et al.* 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. For this reason we focus on widely accepted metrics from information-retrieval. For a test user that receives a list of N recommended items (top-N list), the following are defined:

- *Precision* is the ratio of the number of relevant items in the top-N list (i.e., those in the top-N list that rated positively by the test user) to N.
- *Recall* is the ratio of the number of relevant items in the top-N list to the total number of relevant items (all items rated positively by the test user).

In the following we also use F_1, which is a metric that combines both the aforementioned ones.

Proposed Method

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 Figure 1 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.

	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

(a)

	I_1	I_1	I_1	I_1
U_4	1	4	1	0

(b)

Figure 1: (a) Training Set ($n \times m$), (b) Test Set.

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 Equation 3 and Figure 2 show:

$$A_{n\times m} = U_{n\times n} \cdot S_{n\times m} \cdot V'_{m\times m} \quad (3)$$

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

$V'_{m\times m}$

Figure 2: 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).

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 Equation 4 and Figure 3:

$$A^*_{n\times m} = U_{n\times c} \cdot S_{c\times c} \cdot V'_{c\times m} \quad (4)$$

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

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

-0.61	0.28
-0.29	-0.95
-0.74	0.14

$U_{n\times c}$

8.87	0
0	4.01

$S_{c\times c}$

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

$V'_{c\times m}$

Figure 3: 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^*).

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.

Inserting a test user in the c-dimensional space

Related work (Sarwar *et al.* 2000b) 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 Equation 5 (Furnas, Deerwester, & Dumais 1988). In the current example,we insert U_4 into the 2-dimensional space, as it is shown in Figure 4:

$$\mathbf{u_{new}} = \mathbf{u} \cdot V_{m\times c} \cdot S^{-1}_{c\times c} \quad (5)$$

-0.23	-0.89

$\mathbf{u_{new}}$

1	4	1	0

$\mathbf{u}$

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

$V_{m\times c}$

0.11	0
0	0.25

$S^{-1}_{c\times c}$

Figure 4: Example of: $\mathbf{u_{new}}$ (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).

In Equation 5, $\mathbf{u_{new}}$ 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 $\mathbf{u_{new}}$ vector should be added in the end of the $U_{n\times c}$ matrix which is shown in Figure 3. 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 Figure 1. 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.

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.

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.

Generating the Recommendation List

As it is mentioned, in second stage of CF factors analysis, 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 Equations 6 and 7, 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.

$$p_{u,i} = \overline{r}_u + \frac{\sum_{v\in U} \text{sim}(u,v)(r_{v,i} - \overline{r}_v)}{\sum_{v\in U} |\text{sim}(u,v)|} \quad (6)$$

$$p_{u,i} = \overline{r}_i + \frac{\sum_{j\in \mathcal{I}} \text{sim}(i,j)(r_{u,j} - \overline{r}_j)}{\sum_{j\in \mathcal{I}} |\text{sim}(i,j)|} \quad (7)$$

Therefore, we sort (in descending order) the items according to predicted rating value, and recommend the first N of them.[3] 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.

Evaluation of the CF process

Related work (Sarwar *et al.* 2000b) proposed the Equation 8 for the generation of predictions.

$$p_{u,i} = \overline{r}_u + U_{n\times c} \cdot \sqrt{S_{c\times c}}\sqrt{S_{c\times c}} \cdot V'_{c*m} \quad (8)$$

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 Equations 6 and 7 for prediction, to exploit information from other users or items. Thus, we use these predicted values for the calculation of MAE.

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

Experimental configuration

In the sequel, we study the performance of the described SVD dimensionality reduction techniques against existing CF algorithms, by means of a thorough experimental evaluation. Both user-based and item-based algorithms are tested. Several factors are considered, like the number of considered dimensions, 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),and the size of training set (default value 75%). The metrics we use are recall, precision, and F_1.

We perform experiments with two real data sets that have been used as benchmarks in prior work. In particular, we examined two MovieLens data sets(the default set is the former): (i) the first one with 100,000 ratings assigned by 943 users on 1,682 movies, and (ii) the second one with about 1 million ratings for 3,592 movies by 6,040 users. The range of ratings is between 1(bad)-5(excellent) of the numerical scale. The performance of the former has been verified with the results of the other two (sparse) real data sets. Finally, P_τ is set to 3 and the value of an unrated item is considered equal to zero.

Results for user-based CF algorithm

Firstly, we study the performance of SVD dimensionality reduction in user-based approach. Each time, we preserve a different fraction of principal components of SVD model. More specifically, we preserve 90%, 70%, 50%, 30% and 10% of the total information of initial user-item matrix. The results for precision and recall vs. k are displayed in Figure 5a 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 Figure 6a 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. 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. SVD50 performs a little better than SVD10, as it uses more information. In contrast, the latter SVD reduction uses only 11 dimensions instead

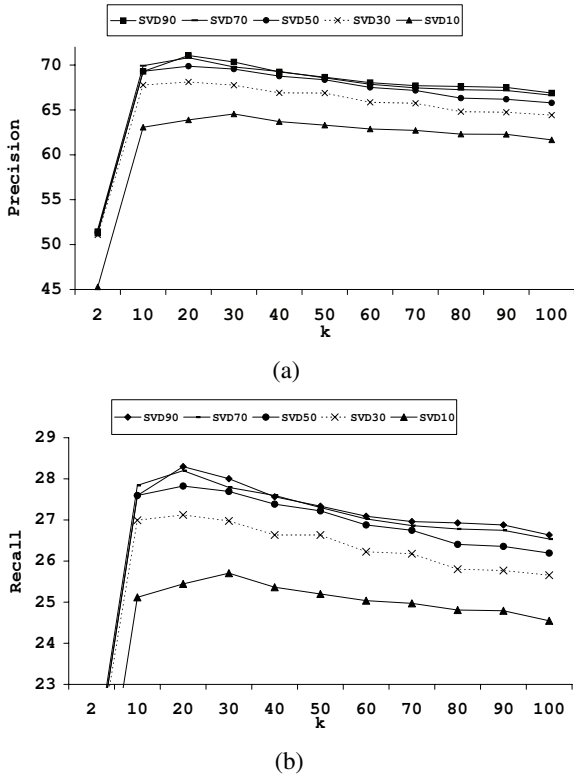

Figure 5: Performance of SVD dimensionality reduction of user-based CF vs. k: (a) precision, (b) recall.

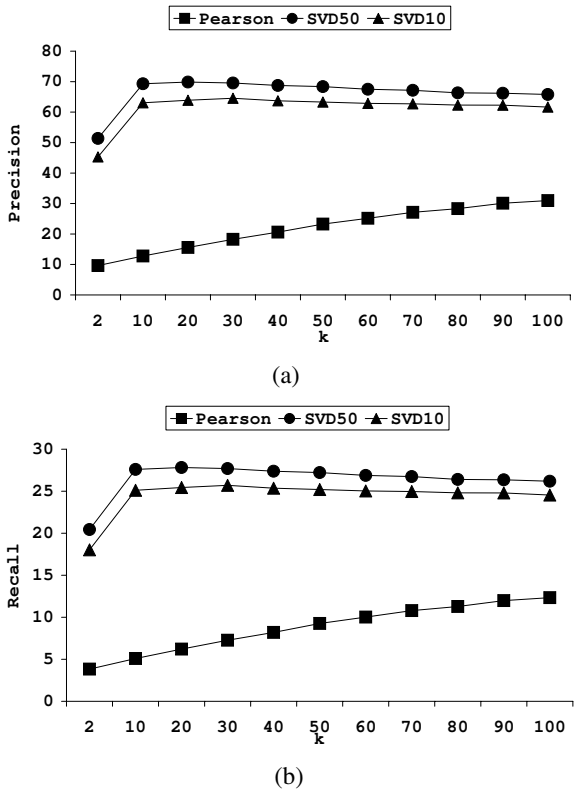

Figure 6: Performance of user-based CF vs. k: (a) precision, (b) recall.

of 157 of the former. This means that the performance of the latter is faster than the former one. Both SVD50 and SVD10 reach 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 40% precision and 11% recall. This illustrates that SVD50 and SVD10 reach their best performance for much smaller values of k compared to Pearson.

We now examine the MAE metric. Results are illustrated in Figure 7a. As expected, Pearson yields the lowest MAE values. This fact supports our conviction that MAE is indicative only for the evaluation of prediction and not of recommendation, as Pearson measure did not attain the best performance in terms of precision and recall.

To consider the impact of scalability, we also examine the 1M data set. The results for the F_1 metric are depicted in Figure 7b. In this case, the relative difference is more extent 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 708 (all training users) 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 HPR top-N list generation algorithm. The results for precision and recall are given in Figure 8. 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 708 for the latter.

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 SVD50 and SVD10 for the item-based case. The results are depicted in Figure 9 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.

Next, we compare Adjusted Cosine, SVD50 and SVD10 against MAE. The results are illustrated in Figure 10a. Differently from the case of user-based CF, all measures have

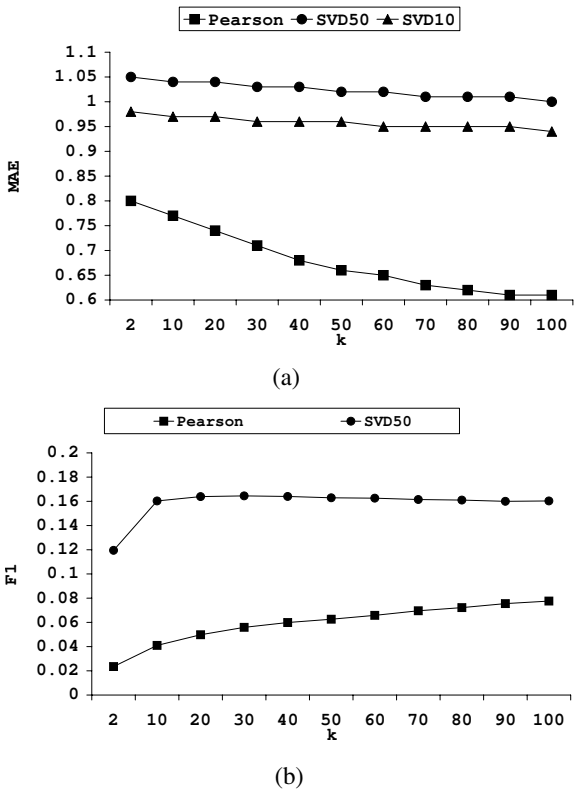

Figure 7: Performance of user-based CF vs. k: (a) MAE, (b) F_1 for 1M Movielens data set.

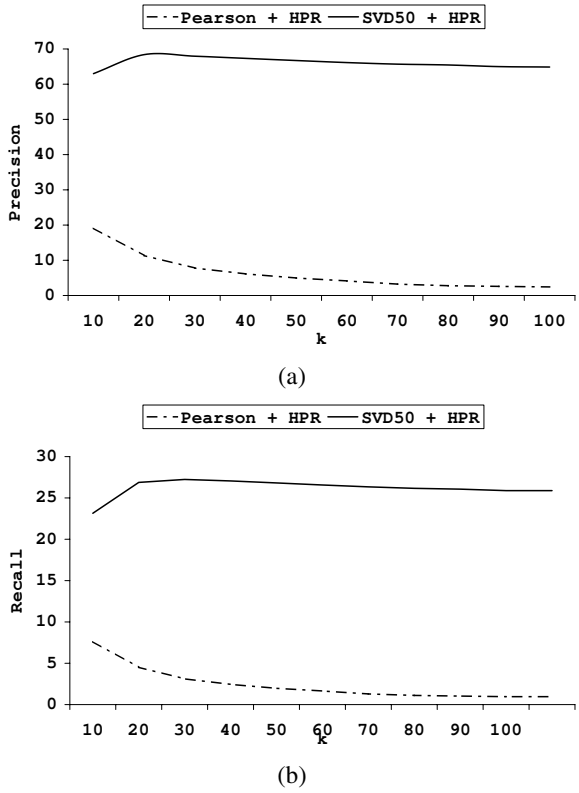

Figure 8: Comparison HPR criteria for the generation of top-N list for user-based CF vs. k (a) precision, (b) recall

similar MAE. 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 Equation 2). This improves its performance for the task of recommendation and worsens the performance of prediction.

Regarding the examination of the larger data set (1M Movielens), the results for the F_1 metric are illustrated in Figure 10b. As we can see, SVD50 is better in terms of precision but the difference is smaller than it was in user-based algorithm. The reason is that the item based CF algorithm has been used to deal with sparse data. So, in situations where the sparsity level is greater, it performs better than the user based CF.

Finally, we test the described criteria HPR for the top-N list generation by item-based CF algorithms. Similarly to the user-based case, HPR in combination with SVD50 performs satisfactory and clearly outperforms the Adjusted Cosine combined with HPR.

Comparative results

In this section, we compare user-based and item-based SVD50 reduction algorithms. With respect to the criterion for the generation of the top-N list, for both of them we use MF as generation list algorithm. The results for precision and recall are displayed in Figure 11a and b, respectively.

These results demonstrate that user-based SVD50 compares favorably against item-based SVD50. The difference in precision is larger than 10%, whereas with respect to recall, it exceeds 5%.

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 vs. k are presented in Figure 12a, whereas the results for varying percentage of preserved information, are depicted in Figure 12b (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 neigborhood matrix. So, in terms of execution item based approach is superior over user based.

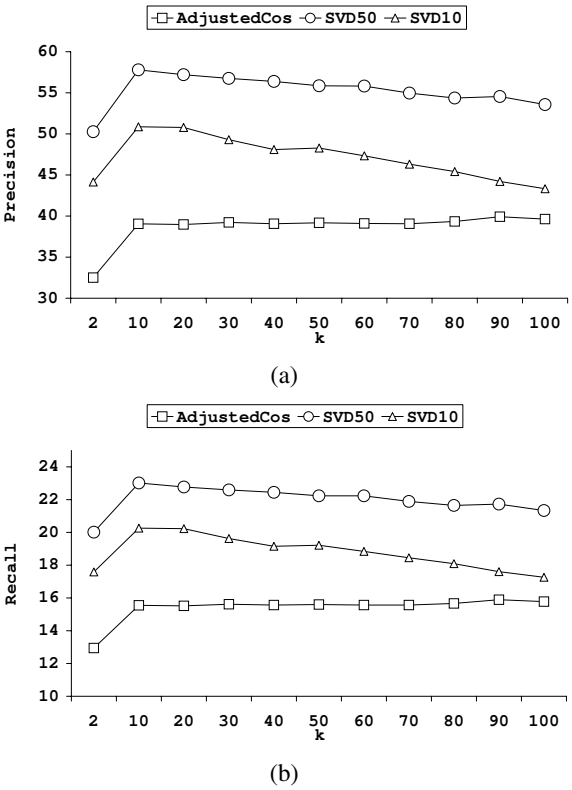

Figure 9: Performance of item-based CF vs. k: (a) precision, (b) recall.

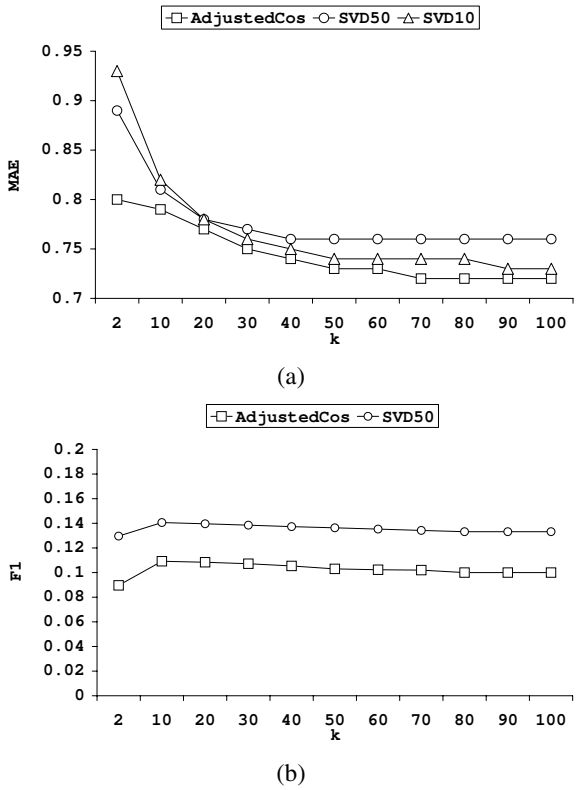

Figure 10: Performance of item-based CF vs. k: (a) MAE, (b) F_1 for 1M Movielens data set.

Conclusions

We perform experimental comparison of the proposed method against well known CF algorithms, like user-based or item-based methods (that do not consider trends), with real data sets.

- Our method shows significant improvements over existing CF algorithms, in terms of accuracy (measured through recall/precision), because 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 dramatically reduced.
- In our experiments we have seen that only a 10% of the original matrix is adequate to provide accurate results.
- The generation of the top-N list with a ranking criterion, is a significant problem that worths further consideration. The proposed HPR criterion in combination with SVD can compete the existing MF, which has been used by the majority of related work.
- Our results showed that, by applying SVD in user-based and item-based CF, the former compares favorably to the latter in terms of precision and recall.
- We should notice that the process of inserting test users rated in the c-dimensional space, is omitted in the item based approach. Thus, this process is more applicable in real applications in terms of complexity even if its accuracy results seems to be smaller than the user's based approach.

Summarizing, on one hand, item-based algorithms are more applicable for off-line computations and, thus, better in on-line response. On the other hand, user-based algorithms produce more accurate recommendations with a small fraction of initial information. For this reason, in our future work we will consider the issue of an approach that would combine high accuracy recommendations in the minimum responding time.

References

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

Breese, J.; Heckerman, D.; and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, 43–52.

Furnas, G.; Deerwester, S.; and Dumais, S. e. a. 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proc. ACM SIGIR Conf.*, 465–480.

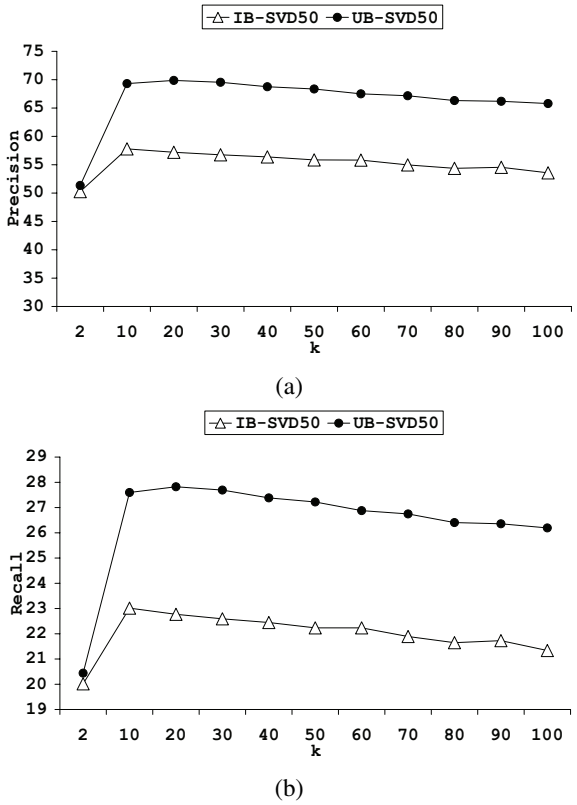

Figure 11: Comparison between item-based and user-based CF in terms of precision vs. k.

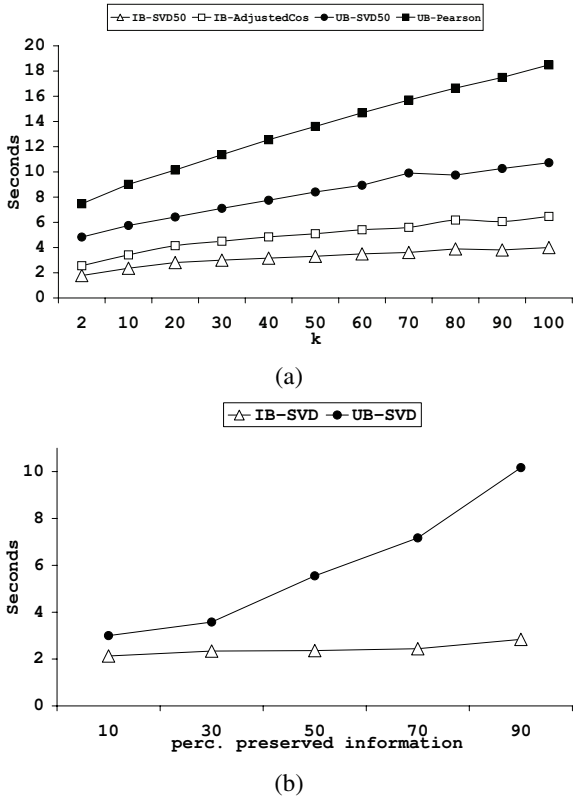

Figure 12: Comparison between item-based and user-based CF in terms of execution time: (a) time vs. k, (b) time vs. percentage of preserved information.

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

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

Herlocker, J.; Konstan, J.; Borchers, A.; and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In *Proc. ACM SIGIR Conf.*, 230–237.

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

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

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

Karypis, G. 2001. Evaluation of item-based top-n recommendation algorithms. In *Proc. ACM CIKM Conf.*, 247–254.

McLauglin, R., and Herlocher, J. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proc. ACM SIGIR Conf.*, 329–336.

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

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering on netnews. In *Proc. Conf. Computer Supported Collaborative Work*, 175–186.

Sarwar, B.; Karypis, G.; Konstan, J.; and J., R. 2000a. Analysis of recommendation algorithms for e-commerce. In *Proc. ACM Electronic Commerce Conf.*, 158–167.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2000b. Application of dimensionality reduction in recommender system-a case study. In *ACM WebKDD Workshop*.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW Conf.*, 285–295.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2002. Incremental singular value decomposition algorithms for higly scalable recommender systems. In *Proc. Computer and Information Technology*.