An Efficient Collaborative Filtering Approach Using Smoothing and Fusing

Daqiang Zhang[†,‡], Jiannong Cao[‡], Jingyu Zhou[†], Minyi Guo[†], Vaskar Raychoudhury[‡]
[†]Department of Computer Science, Shanghai Jiao Tong University
[‡]Department of Computing, The Hong Kong Polytechnic University
{csdqzhang, csjcao, csvray}@comp.polyu.edu.hk, {zhou-jy, guo-my}@cs.sjtu.edu.cn

***Abstract*—Collaborative Filtering (CF) has achieved widespread success in recommender systems such as Amazon and Yahoo! music. However, CF usually suffers from two fundamental problems - data sparsity and limited scalability. Among the two broad classes of CF approaches, namely, memory-based and model-based, the former usually falls short of the system scalability demands, because these approaches predict user preferences over the entire item-user matrix. The latter often achieves unsatisfactory accuracy, because they cannot capture precisely the diversity in user rating styles.**

In this paper, we propose an efficient Collaborative Filtering approach using Smoothing and Fusing (CFSF) strategies. CFSF formulates the CF problem as a local prediction problem by mapping it from the entire large-scale item-user matrix to a locally reduced item-user matrix. Given an active item and a user, CFSF dynamically constructs a local item-user matrix as the basis of prediction. To alleviate data sparsity, CFSF presents a fusion strategy for the local item-user matrix that fuses ratings of the same user makes on similar items, and ratings of like-minded users make on the same and similar items. To eliminate diversity in user rating styles, CFSF uses a smoothing strategy that clusters users over the entire item-user matrix and then smoothes ratings within each user cluster. Empirical study shows that CFSF outperforms the state-of-the-art CF approaches in terms of both accuracy and scalability.

I. Introduction

Collaborative Filtering (CF) is a process that automatically predicts user preferences from large-scale item-user matrices. It is capable of offering users with the most appropriate items in order to enhance user satisfaction and loyalty to electronic retailers and content providers [1]. In general, CF approaches can be classified into two categories: memory-based and model-based. It is achieving increasing success in recommender systems such as Amazon [2], Google News Personalization [3] and Yahoo! Music [4]. Despite its wide acceptability, CF suffers from two fundamental problems - data sparsity and limited scalability [5], [6].

Memory-based CF approaches achieve high accuracy by exploiting similarity functions of items and users, but they are unable to scale up [7], [8]. They predict user preferences by identifying similar items or like-minded users over the entire item-user matrix. This process tends to be time-consuming, because operations are performed on the entire large-scale item-user matrix. Previous work has been much more concerned with accuracy than with scalability, though the size of item-user matrices continues to grow at an unprecedented speed in recommender systems.

In contrast, model-based CF approaches obtain good scalability but relatively poor accuracy [2], [7]. They cluster all items or users into classes using machine learning algorithms, e.g., linear classifiers [9] and Bayesian network [10]. Then, they use these classes to predict unrated data for users. They narrow down the search space for similar items or like-minded users; and thus, become scalable. Intuitively, they seem like desirable CF approaches, but they are limited by data sparsity. Commercial item-user matrices in recommender systems are quite sparse and the density of available ratings may be less than 1% [11]. Due to data sparsity, many related ratings are not available when model-based CF approaches predict user preferences. Moreover, most of them do not exploit the ratings that like-minded users make on similar items. Thus, unreliable recommendation quality prevents them from practical applications. Apart from these two primary CF approaches, other CF approaches are also proposed [5], [12], [13]. Most of them focus on either reducing the dimensionality of data or combining memory-based and model-based CF approaches. These approaches, however, only partially solve the problems of CF.

In this paper, we propose an efficient Collaborative Filtering approach using Smoothing and Fusing (CFSF), which formulates the CF problem as a local prediction problem. It is achieved by mapping the CF problem over the entire large-scale item-user matrix to a locally reduced item-user matrix. CFSF divides the CF process into two phases - offline and online. In the offline phase, CFSF computes a global item similarity matrix and sorts the result in descending order. Then, it clusters users with a smoothing strategy to eliminate the diversity in user ratings styles. In the online phase, CFSF constructs a local item-user matrix by picking up the top M similar items from the global item similarity matrix and the top K like-minded users from user clusters. Finally, CFSF predicts user preferences over the local item-user matrix by fusing the ratings of the same users made on similar items, and like-minded users make on the same and similar items. Empirical study on public datasets shows that CFSF efficiently addresses the two fundamental problems of CF - data sparsity and scalability.

The rest of this paper is organized as follows. Section II is an overview of related work. Section III introduces the background of CF. Section IV describes our approach in detail. Section V reports our empirical study and Section VI

0190-3918/09 $26.00 © 2009 IEEE
DOI 10.1109/ICPP.2009.16

concludes our work by pointing to future directions.

II. Related work

In this section, we briefly review previous work, including memory-based, model-based and other CF approaches.

A. Memory-based CF approaches

Memory-based CF approaches can be further classified into item-based, user-based and UI-based approaches according to the rating source used in prediction [7], [11]. Item-based CF approaches predict user preferences from the ratings made on similar items by the same user [2]. User-based CF approaches exploit the idea that like-minded users may go for the same item. Alternatively, UI-based CF approaches predict user preferences not only from the ratings used by item-based and user-based CF approaches, but also from the ratings that the like-minded users make on similar items.

Representative memory-based CF approaches are SF [8] and EMDP [6]. SF is one type of UI-based CF approach that unifies the item-based and user-based approaches. SF considerably improves prediction accuracy, but it is slow due to a search on similar items and like-minded users over the whole item-user matrix. SF predicts user preferences without capturing the diversity in user ratings. This diversity is incorporated in EMDP, which calculates the items and users whose similarities exceed certain thresholds. Thus, EMDP improves the accuracy and returns exact results. EMDP is based on a set of different thresholds for each item and user. This is a computer-intensive job, due to the large quantity of items and users in recommender systems. Moreover, inappropriate thresholds may lead to few results, leaving users expectant to receive suggestions from the feedback system. EMDP tends to select the most similar items and like-minded users for prediction, which is easily achieved using CFSF.

B. Model-based CF approaches

Model-based CF approaches learn a model of user ratings in an item-user matrix, and then use it to predict the scores of unrated items for an active user. They use a variety of techniques to create the model, such as Bayesian network [10], linear classifiers [9] and Principal Component Analysis (PCA) [2]. Typical examples for model-based CF approaches are clustering [14] and aspect model [15].

Usually, model-based CF approaches are fast to predict user preferences because they narrow down the search space for identifying similar items and like-minded users. They tend, however, to suffer from time-consuming training and updating processes. Furthermore, most of them do not consider the ratings made on similar items by like-minded users [7], [16].

C. Other CF work

Several other types of CF approaches have also been proposed [1], [13], [17], [18], [19]. In [20], [12], matrix factorization is explored to improve prediction accuracy of CF. In [7], SCBPCC is a cluster-based CF approach that introduces a smoothing strategy among user clusters and efficiently solves the data sparsity problem. It achieves high levels of accuracy, but doesn't capture the ratings of like-minded users on similar items. Moreover, SCBPCC could be further improved in scalability because it identifies the similar items over the entire item-user matrix each time.

In contrast, content-based CF approaches, used in Google and Yahoo!, predict user preferences based on the classifications of item contents, rather than item sources or other criteria [5], [21]. They are, however, limited by two assumptions: 1) users are able to express their particular preferences or information requirements with respect to intrinsic features of items, and 2) systems are capable of thoroughly comprehending item contents and accurately extracting their features [16].

III. Background

Recommender systems aim at predicting the rating of active item i_a made by active user u_b from user profiles. These profiles are represented as a $Q \times P$ item-user matrix X, where Q and P are the sizes of X. In this section, we introduce notations for CF. Let

- $\mathcal{I} = \{i_1, i_2, \ldots, i_Q\}$ and $\mathcal{U} = \{u_1, u_2, \ldots, u_P\}$ be the sets of items and users in X,
- $\{C_u^1, C_u^1, \ldots, C_u^L\}$ be L user clusters, and users in each cluster share some similar tastes,
- $I\{u\}$, $I\{C_u\}$ and $U\{i\}$ be the set of items rated by user u, the set of items rated by user cluster C_u, and the set of users who have rated item i,
- r_{u_b,i_a} denote the score that user b rates item a, $\overline{r_{i_a}}$ and $\overline{r_{u_b}}$ represent the average ratings of item i_a and user u_b,
- SI, SU and SUI be the sets of the similar items, like-minded users, and similar items and like-minded users,
- SIR, SUR and $SUIR$ denote predicting user preferences over the entire item-user matrix from the ratings of the same user make on the similar items, the like-minded users make on the same item, and the like-minded users make on the similar items,
- SR represent predicting user preferences from all the ratings, i.e., SIR, SUR and $SUIR$,
- SIR', SUR', $SUIR'$ and SR' be the counterparts of SIR, SUR, $SUIR$ and SR, but they are calculated over the local item-user matrix.

Then, the item vector of the matrix X is:

$$X_i = [i_1, i_2, \cdots, i_Q], i_q = [r_{1,q}, \cdots, r_{P,q}]^T,$$

where $q \in [1, Q]$. Each column vector i_m corresponds to the ratings of a particular item m by P users. Matrix X can also be represented by user vectors illustrated as:

$$X_u = [u_1, u_2, \cdots, u_P]^T, u_p = [r_{p,1}, \cdots, r_{p,Q}]^T,$$

where $p \in [1, P]$. Each row vector u_p^T indicates a user profile that represents a particular user's item ratings. Item-based CF approaches, illustrated as Fig. 1a, find the similar items among item vectors and then use their ratings made by the same user to predict his or her preferences. For example, given an active item i_a and a user u_b, Eq. 1 denotes the mechanism of item-based CF approaches, where sim_{i_a,i_c} is the similarity of items

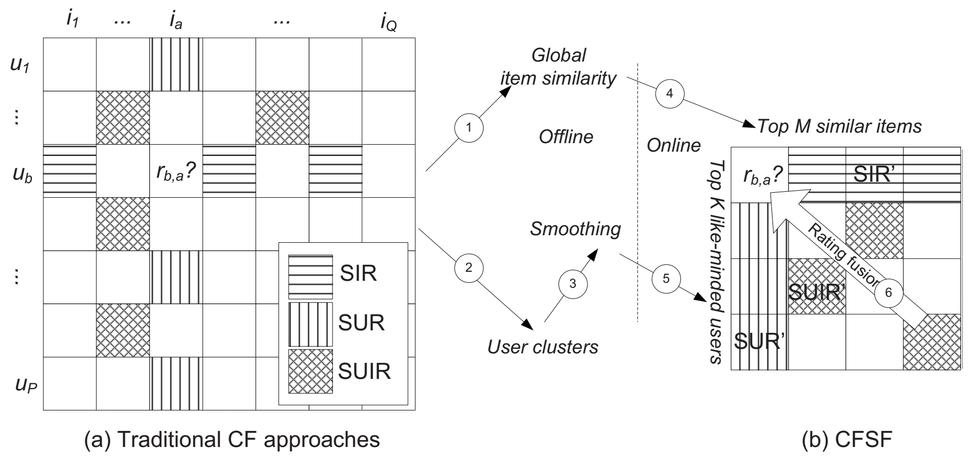

Fig. 1. Deriving CFSF from traditional CF approaches

i_a and i_c, and is usually computed by Pearson Correlation Coefficient (PCC) or Vector Space Similarity (VSS).

$$SIR : \widehat{r_{u_b,i_a}} \longleftarrow \frac{\sum\limits_{i_c \in SI} sim_{i_a,i_c} \cdot r_{u_b,i_c}}{\sum\limits_{i_c \in SI} sim_{i_a,i_c}} \tag{1}$$

Alternatively, user-based CF approaches take advantage of the similar motivation to predict user preferences, where the ratings of like-minded users make on the active item are used. Eq. 2 shows the mechanism of user-based CF approaches, where sim_{u_b,u_c} is similarity of users u_b and u_c.

$$SUR : \widehat{r_{u_b,i_a}} \longleftarrow \frac{\sum\limits_{u_c \in SU} sim_{u_b,u_c} \cdot r_{u_c,i_a}}{\sum\limits_{u_c \in SU} sim_{u_b,u_c}} \tag{2}$$

Both item-based and user-based CF approaches do not consider $SUIR$ that is heuristic for accuracy improvement. Let i be a similar item to i_a and u be a like-minded user to u_b, $SUIR$ is calculated as Eq. 3.

$$SUIR : \widehat{r_{u_b,i_a}} \longleftarrow \frac{\sum\limits_{u,i \in SUI} sim_{(i,i_a),(u,u_b)} \cdot r_{u,i}}{\sum\limits_{u,i \in SUI} sim_{(i,i_a),(u,u_b)}}, \tag{3}$$

where $sim_{(i,i_a),(u,u_b)}$ is the weight for the rating user u makes on item i, denoting how much the rating $r_{u,i}$ is considered in prediction. It is defined as Eq. 13 in CFSF.

UI-based CF approaches [6], [8] have been proposed to combine SIR, SUR and $SUIR$, given as Eq. 4.

$$SR : \widehat{r_{u_b,i_a}} \longleftarrow \pounds\{SIR, SUR, SUIR\} \tag{4}$$

where $\pounds$ is a fusion function that fuses the ratings from SIR, SUR and $SUIR$, whose mechanisms are illustrated as Fig. 1a. Due to the time-consuming search for active items and users over the entire item-user matrix, all memory-based CF approaches achieve limited scalability.

IV. CFSF APPROACH

In this section, we propose an efficient Collaborative Filtering approach using Smoothing and Fusing strategies (CFSF). We overview it briefly and then describe it in detail.

A. CFSF overview

Fig. 1 explains how to derive CFSF from traditional CF approaches. First, CFSF creates a global item similarity matrix (GIS) as performed in the memory-based manner (i.e., search over the entire item-user matrix). Then, CFSF classifies users into clusters, within each of which unrated ratings are smoothed. These two steps significantly reduce the influences of ratings diversity and accelerate the selection of like-minded users. Based on the request for an active user from the recommender system, CFSF picks up the top M similar items from GIS, the top K like-minded users from user clusters and extracts related ratings to create the local item-user matrix. In the end, it predicts user preferences by fusing the ratings SIR', SUR' and $SUIR'$ over the local item-user matrix. Fig. 1 also shows that CFSF significantly cuts down the number of users and items involved in prediction.

In view of the increasing number of items and users in recommender systems, CFSF divides the CF process into offline and online phases. The offline phase involves the first three steps - creating the GIS, clustering users and smoothing user ratings. This phase is a time-consuming process. The online phase includes the latter three steps - selecting items and users, constructing a local item-user matrix and predicting preferences. For each user, CFSF requires him or her to rate a certain number of items and then inserts a record in the item-user matrix.

Let M be the number of similar items and K be the number of like-minded users. Algorithm 1 illustrates the process of CFSF. Sections IV-B, IV-C and IV-D describe the offline phase and Section IV-E describes the online phase.

B. Creating $GIS \leftarrow I$

In this step, CFSF creates GIS to save the item similarity matrix and to eliminate the diversity in item ratings. Items that are popular tend to get higher ratings than unpopular items. PCC, rather than Pure Cosine Similarity (PCS), is selected as the item similarity function, because PCS does not consider the diversity in item ratings. Given items i_a, i_b and U=$U\{i_a\}\wedge$

Algorithm 1 CFSF algorithm

```
1:  procedure CFSF
2:      input: Item set: $\mathcal{I}$, User set: $\mathcal{U}$
3:      output: $r_{u_b,i_a}$: rating of item $i_a$ by user $u_b$

4:      Offline
5:          Creating $GIS \leftarrow I$
6:          Clustering users $C_i \leftarrow U$
7:          Smoothing user ratings within each $C_i$
8:      End Offline
9:      // Constructing a local $M \times K$ matrix
10:     Online
11:         Selecting top $M\ items \leftarrow GIS$
12:         Selecting top $K\ users \leftarrow C_i$
13:         Computing $SIR'$, $SUR'$ and $SUIR'$
14:         Calculating $SR' \leftarrow \pounds\{SIR', SUR', SUIR'\}$
15:     End Online
16: end procedure
```

$U\{i_b\}$, the similarity between i_a and i_b is defined as Eq. 5:

$$sim_{i_a,i_b} = \frac{\sum\limits_{u \in \mathcal{U}} (r_{u,i_a} - \overline{r_{i_a}}) \cdot (r_{u,i_b} - \overline{r_{i_b}})}{\sqrt{\sum\limits_{u \in \mathcal{U}} (r_{u,i_a} - \overline{r_{i_a}})^2} \cdot \sqrt{\sum\limits_{u \in \mathcal{U}} (r_{u,i_b} - \overline{r_{i_b}})^2}} \quad (5)$$

Given the large number of items, we set thresholds for Eq. 5 to filter less important items. Then, the size of GIS will be greatly reduced.

C. Clustering users $C_i \leftarrow U$

In order to eliminate the diversity in user ratings, CFSF uses K-means to cluster users and then smoothes ratings within each user cluster. The K-means method trains the data iteratively and assigns every user to a cluster whose centroid is closest to him or her. The time complexity of each iteration is linear in the size of dataset. Compared with other clustering methods, K-means is simple, fast and accurate. Its primary objective is minimizing $\sum_{i=1}^{k} \sum_{u_j \in C_i} sim|u_j - \overline{u}|$, where $\overline{u}$ is the centroid of all users that belong to C_i. Similarity $sim|u_j - \overline{u}|$ is defined as Eq. 6 based on PCC similarity function, where u_a and u_b are users, and I=$I(u_a) \wedge I(u_b)$, denoting an item set that both users u_a and u_b have rated.

$$sim_{u_a,u_b} = \frac{\sum\limits_{i \in \mathcal{I}} (r_{u_a,i} - \overline{r_{u_a}}) \cdot (r_{u_b,i} - \overline{r_{u_b}})}{\sqrt{\sum\limits_{i \in \mathcal{I}} (r_{u_a,i} - \overline{r_{u_a}})^2} \cdot \sqrt{\sum\limits_{i \in \mathcal{I}} (r_{u_b,i} - \overline{r_{u_b}})^2}} \quad (6)$$

Thus, user clusters are generated for smoothing user ratings and selecting the top K like-minded users as was done in Sections IV-D and IV-E2.

D. Smoothing user ratings within C_i

There are two reasons for smoothing unrated data. First, users in the same cluster share similar tastes but have dissimilar rating styles; and so, the same item is rated differently by users belonging to the same cluster. This diversity in user ratings negatively affects the prediction accuracy. Second, rating data is quite sparse because users prefer not to rate items and also cannot rate all items due to the overwhelming number of items in recommender systems. Consequently, the prediction accuracy is unsatisfactory without capturing such diversity. CFSF uses a smoothing strategy similar to SCBPCC [7] to fill unrated data. The smoothing function is defined as Eq. 7.

$$r_{u,i} = \begin{cases} r_{u,i}, & if\ u\ rates\ i \\ \overline{r_u} + \Delta r_{C_{u',i}} & otherwise \end{cases} \quad (7)$$

where $\Delta r_{C_{u',i}}$ is the deviation of average rating of item i in $C_{u',i}$ that is a set of users who rate the item i in user cluster $C_{u'}$. $\Delta r_{C_{u',i}}$ is given as Eq. 8:

$$\Delta r_{C_{u',i}} = \sum_{u \in C_{u',i}} (r_{u,i} - \overline{r_u})/|C_{u',i}|, \quad (8)$$

where $|C_{u',i}|$ is the size of $C_{u',i}$.

After smoothing, CFSF creates $iCluster$ for each user to store its similarity to each user cluster. These $iCluster$s are sorted in descending order and used for selecting the top K like-minded users. As a result, they accelerate selection efficiency considerably. The feature of a user cluster is denoted as a centroid that represents an average rating over all users in the cluster. Given an item set $\mathcal{I} = I\{u_a\} \wedge I\{C_{u'}\}$, the similarity between user u_a and cluster $C_{u'}$ is defined as Eq. 9. Thus, we get the $iCluster$ for each user. For instance, the $iCluster$ for user u_a is $\{C_0, C_1, C_7, C_6, C_2, C_3, C_5, C_4\}$.

$$sim_{u_a,C_{u'}} = \frac{\sum\limits_{i \in \mathcal{I}} \Delta r_{C_{u',i}} \cdot (r_{u_a,i} - \overline{r_{u_a}})}{\sqrt{\sum\limits_{i \in \mathcal{I}} \Delta (r_{C_{u',i}})^2} \cdot \sqrt{\sum\limits_{i \in \mathcal{I}} (r_{u_a,i} - \overline{r_{u_a}})^2}} \quad (9)$$

So far, we have described all the steps in the offline phase that are often computer-intensive; and hence, performed in the backend. The online phase of CFSF focuses on responding to requests, including constructing a local item-user matrix and fusing the ratings for prediction.

E. Constructing a local $M \times K$ item-user matrix

In general, user preferences are most likely derived from the most similar items and like-minded users. CFSF creates the local item-user matrix containing the most related users and items; and thus, yields significant savings in CPU, bandwidth, latency, etc. When a request comes, CFSF will pick up the top M similar items from GIS, the top K like-minded users from user clusters C, and extract related ratings from the original item-user matrix.

1) Selecting top M similar items: Recall that CFSF sorts the result as GIS in descending order when it creates a global item similarity matrix. Consequently, CFSF can directly pick up the top M similar items from GIS.

2) Selecting top K like-minded users: User preferences are often scattered into several user clusters. Take movie for example, user u may like action, fantasy and crime types. To cover user preferences as much as possible, CFSF selects a user candidate set and then selects the top K like-minded users. To create a user candidate set, CFSF selects users from clusters in $iCluster$ one by one, e.g., from C_0 to C_4 mentioned in Section IV-D. To select the top K like-minded users, CFSF varies the similarity function according to two types of ratings: original and smoothed ratings. It introduces a parameter w to differentiate these ratings. Given user u and active user u_a, the similarity function of selecting the top K like-minded users is defined as:

$$sim_{u_a,u} = \frac{\sum\limits_{f} w_{u,i} \cdot (r_{u,i} - \overline{r_u}) \cdot (r_{u_a,i} - \overline{r_{u_a}})}{\sqrt{\sum\limits_{f} {w_{u,i}}^2 (r_{u,i} - \overline{r_u})^2} \cdot \sqrt{\sum\limits_{f} (r_{u_a,i} - \overline{r_{u_a}})^2}} \tag{10}$$

where f denotes $i \in I\{u_a\}$ and w is the coefficient, defined as Eq. 11. Depending on the rating is original or smoothed rating, the weighting coefficient w varies.

$$w : w_{u,i} = \begin{cases} \varepsilon & if\ u\ rates\ i \\ 1 - \varepsilon & otherwise \end{cases} \tag{11}$$

Compared with previous methods, CFSF reduces the computation overhead by selecting the like-minded users from iCluster rather than the entire item-user matrix. Moreover, CFSF is capable of setting thresholds for Eq. 10, which will further reduce the computation overhead. After the top M similar items and the top K like-minded users are selected, CFSF will extract related ratings from the original item-user matrix to fill the local item-user matrix.

F. Fusing SIR', SUR' and $SUIR'$

CFSF defines SIR', SUR' and $SUIR'$ as prediction of user preferences over the local item-user matrix from the ratings of the same user makes on similar items, the like-minded users make on the same and similar items. Because M and K are much less than Q and P, SIR', SUR' and $SUIR'$ are computed much faster than SIR, SUR and $SUIR$ that are calculated over the entire large-scale item-user matrix. Given active item i_a and user u_b, Eq. 12 illustrates these definitions.

$$\begin{aligned} SIR' &= \frac{\sum_{s=1}^{M} w \cdot sim_{i_s,i_a} \cdot r_{u_b,i_s}}{\sum_{s=1}^{M} w \cdot sim_{i_s,i_a}} \\ SUR' &= \frac{\sum_{t=1}^{K} w \cdot sim_{u_t,u_b} \cdot (r_{u_t,i_a} - \overline{r_{u_t}})}{\sum_{t=1}^{K} w \cdot sim_{u_t,u_b}} + \overline{r_{u_b}} \quad , \\ SUIR' &= \frac{\sum_{t=1}^{K} \sum_{s=1}^{M} w \cdot sim_{(i_s,i_a),(u_t,u_b)} \cdot r_{u,i}}{\sum_{t=1}^{K} \sum_{s=1}^{M} w \cdot sim_{(i_s,i_a),(u_t,u_b)}} \end{aligned} \tag{12}$$

where w is defined as Eq. 11 and $sim_{(i,i_a),(u,u_b)}$ is defined by Euclidean distance as Eq. 13, denoting the weight for the rating of the similar item i by the like-minded user u.

$$sim_{(i_s,i_a),(u_t,u_b)} = \frac{sim_{i_s,i_a} \cdot sim_{u_t,u_b}}{\sqrt{sim^2_{i_s,i_a} + sim^2_{u_t,u_b}}} \tag{13}$$

CFSF selects SUR' as the major prediction tool and SIR' and $SUIR'$ as supplementary when it predicts user preferences. Because SIR', SUR' and $SUIR'$ have different impact on accuracy, CFSF introduces two parameters λ and δ to balance them. The fusion function of CFSF is defined as Eq. 14:

$$\begin{aligned} SR' : \widehat{r_{u_b,i_a}} &= \pounds\{SIR', SUR', SUIR'\} \\ &= (1-\delta) \cdot (1-\lambda) \cdot SIR' \\ &+ (1-\delta) \cdot \lambda \cdot SUR' \\ &+ \delta \cdot SUIR', \end{aligned} \tag{14}$$

where $\pounds$ is a function, and λ and δ are between 0 and 1.

To summarize, CFSF contains two phases. In the offline phase, its time complexity is very high, and determined by the creation of the global item similarity and K-means. To reduce computation overhead, CFSF sets thresholds to filter items. In the online phase, its time complexity is $O(MK)$, where M and K are the number of similar items and like-minded users. Considering that M and K are much less than the original sizes of an item-user matrix, CFSF is scalable.

V. Evaluation

In order to evaluate our proposed approach, we carried out a series of experiments. In particular, we tried to answer the following questions:

- What is the overall performance of CFSF? Does it work better than traditional item-based, user-based CF approaches, and the state-of-the-art CF approaches?
- How do the two fundamental problems of CF (sparsity and scalability) affect the performance of CFSF?
- How do the parameters influence the performance of CFSF? Several parameters are involved such as similarity fusion parameters λ and δ.

A. Dataset

The proposed approach is evaluated with MovieLens dataset [1]. This dataset is from the University of Minnesota, which is one of the most well-respected datasets for collaborative filtering. Most existing work is evaluated across it.

We randomly extracted 500 users from MovieLens, where each user rated at least 40 movies. We changed the size of the training set by selecting the first 100, 200 and 300 users, denoted as ML_100, ML_200 and ML_300. We selected the last 200 users as the testset. We varied the number of items rated by active users from 5, 10 to 20, denoted as Give5, Given10 and Given20. Table I summarizes the statistical features of the datasets used in our experiments.

TABLE I
Statistics of the datasets

	MovieLens
No. of Users	500
No. of Items	1000
Average no. of rated items per user	94.4
Density of data	9.44%
No. of ratings	5

[1]G. Lab. MovieLens. http://www.grouplens.org/

TABLE II
MAE ON MOVIELENS FOR THE SIR, SUR AND CFSF

Training set	Methods	Given5	Given10	Given20
ML_300	CFSF	**0.743**	**0.721**	**0.705**
	SUR	0.838	0.814	0.802
	SIR	0.870	0.838	0.813
ML_200	CFSF	**0.769**	**0.734**	**0.713**
	SUR	0.843	0.822	0.807
	SIR	0.855	0.834	0.812
ML_100	CFSF	**0.781**	**0.758**	**0.746**
	SUR	0.876	0.847	0.811
	SIR	0.890	0.801	0.824

B. Metrics

In order to maintain consistency with experiments reported in the literature [7], [8], [11], [22], [23], we chose the same MAE as the evaluation metric, defined as:

$$MAE = \frac{\sum_{u \in T} |r_{u,i} - \overline{r_{u,i}}|}{|T|}, \quad (15)$$

where $r_{u,i}$ denotes the rating that user u rates item i, T represents the testset and $|T|$ is the test size. The smaller the value of MAE, the better the performance.

C. Accuracy

1) Overall performance: We carried out experiments from two aspects to evaluate the performance of CFSF. One aspect is to compare CFSF with traditional memory-based CF approaches: an item-based approach using PCC (SIR) and a user-based approach using PCC (SUR). For MovieLens, the parameters of CFSF are set as follows: $C = 30$, $\lambda = 0.8$, $\delta = 0.1$, $K = 25$, $M = 95$ and w=0.35. Table II illustrates the results, showing that CFSF considerably outperforms the SUR and SIR with respect to prediction accuracy.

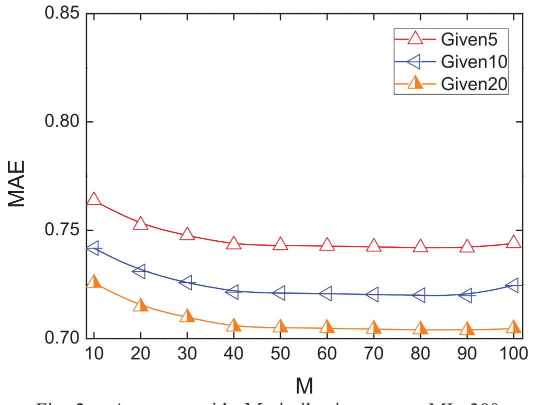

Fig. 2. Accuracy with M similar items over ML_300

The other aspect is to compare CFSF with the other state-of-the-art CF approaches, i.e., AM [16], EMDP [6], PD[19], SCBPCC [7] and SF [8]. We varied the item number that each user was required to rate on all the testsets for MovieLens dataset. The result is shown as Table III. As the testset increases, the MAEs of all approaches show a downward trend. As the number of rated items for each user increases from 5 to 20, a similar trend is observed. Among them, CFSF achieves the best accuracy. This is because CFSF can select the most similar items and like-minded users.

TABLE III
MAE ON MOVIELENS FOR THE STATE-OF-THE-ART CF APPROACHES

Training set	Methods	Given5	Given10	Given20
ML_300	CFSF	**0.743**	**0.721**	**0.705**
	AM	0.820	0.822	0.796
	EMDP	0.788	0.754	0.746
	SCBPCC	0.822	0.810	0.778
	SF	0.804	0.761	0.769
	PD	0.827	0.815	0.789
ML_200	CFSF	**0.769**	**0.734**	**0.713**
	AM	0.849	0.837	0.815
	EMDP	0.793	0.760	0.751
	SCBPCC	0.831	0.813	0.784
	SF	0.827	0.773	0.783
	PD	0.836	0.815	0.792
ML_100	CFSF	**0.781**	**0.758**	**0.746**
	AM	0.963	0.922	0.887
	EMDP	0.807	0.769	0.765
	SCBPCC	0.848	0.819	0.789
	SF	0.847	0.774	0.792
	PD	0.849	0.817	0.808

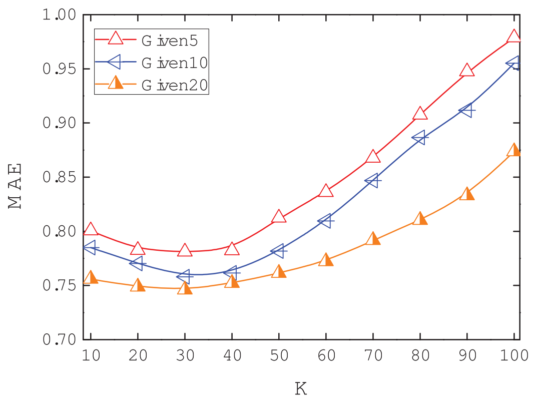

Fig. 3. Accuracy with K similar items over ML_300

2) Accuracy with M similar items: The similar items M, the like-minded users K and the user clusters C conspicuously affect the accuracy of CFSF. In order to figure out their influence on CFSF, we conducted experiments over Given5, Given10 and Given20 on all the training sets for all datasets.

Fig. 2 shows the accuracy of CFSF with M over ML_300. CFSF achieves higher scalability as M increases. When M is less than 50, the similar items to the active item are not many, leading to a high MAE. When M is greater than 60, CFSF collects enough ratings so that it achieves a low MAE.

3) Accuracy with K like-minded users: In order to check the accuracy with K like-minded users, we did experiments over all the training sets with varying the value of K from 10 to 100 at Given5, Given10 and Given20 for all datasets.

Fig. 3 shows the results. When K is between 20 and 40, CFSF gets a low MAE. When K is larger than 40, it gets a high MAE. This is because the ratings from less related users are considered too much for recommendation.

4) Accuracy with C user clusters: CFSF uses the smoothing strategy within user clusters to eliminate the diversity in user ratings. Therefore, the number of user clusters affects the performance of CFSF. We conducted experiments for all the training sets by varying C from 10 to 100.

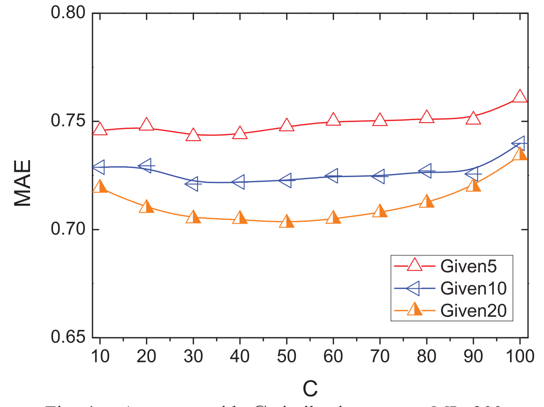

Fig. 4. Accuracy with C similar items over ML_300

Fig. 4 illustrates the accuracy with the user clusters C for ML_300. When C is less than 30, CFSF is incapable of getting a low MAE due to the diversity in user ratings. When C is larger than 90, CFSF cannot properly eliminate the diversity due to too many user clusters. Note that the MAE of CFSF at Given20 increases much faster due to the large possibility of coincidence ratings among users.

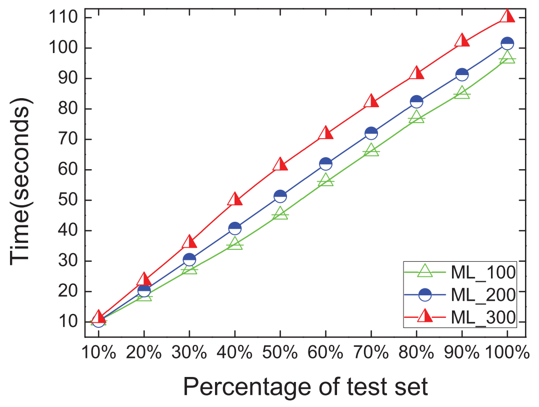

Fig. 5. Response time at Given20 on MovieLens

D. Support for scalability

Scalability is extremely important for CF approaches that are used in larger-scale recommender systems. In this section, we conducted experiments to check the scalability of CFSF by varying the testset and training sets over the MovieLens dataset. We randomly selected 10%, 20% and up to 100% of the last 200 users as testsets, and selected ML_100, ML_200 and ML_300 as training sets. We selected response time as a metric of scalability and ran our program with Windows XP with 1 GB RAM and 2.4 GHz CPU.

Fig. 5 shows the response time of CFSF for online prediction. As the testset grows, the response time increases in a linear fashion, indicating that CFSF is highly scalable. The maximum response time for ML_300 with 100% percentage of the testset is 110 seconds, while SCBPCC spent around 260 seconds. CFSF achieves this by using the locally reduced item-user matrix and caching intermediate results.

E. Sensitivity of parameters

CFSF involves several parameters, such as λ, δ and w. To evaluate their influence, we conducted a series of experiments.

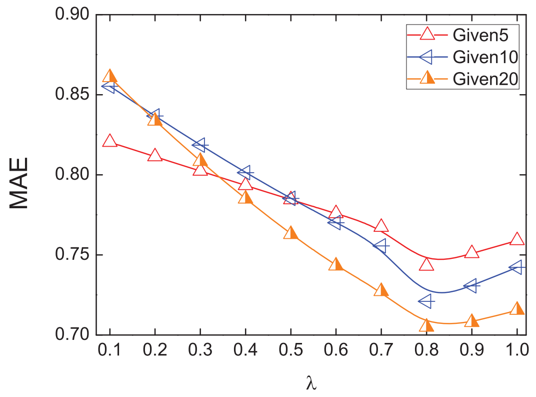

Fig. 6. Sensitivity of λ over ML_300

1) Sensitivity of λ: CFSF incorporates SIR' and SUR' for high accuracy, which have different influence on the prediction accuracy. Therefore, CFSF introduces λ to differentiate them. When λ is set to 0, SUR' does not influence the prediction; while λ is set to 1, SIR' is not considered at all. We evaluated λ for all the training sets on MovieLens.

Fig. 6 shows the sensitivity of λ. As λ increases from 0.1 to 1, the MAE for CFSF first decreases and later increases for ML_300. The minimum MAE is obtained when λ equals to 0.8, which is the default value of λ in our experiments, which means SUR' is more important than SIR'.

2) Sensitivity of δ: CFSF incorporates the $SUIR'$ that exerts a lesser influence on prediction than SIR' and SUR'. Consequently, CFSF introduces δ to differentiate $SUIR'$ from SIR' and SUR'. When δ is set to 0, $SUIR'$ has no impact on the prediction accuracy; while δ is set to 1, $SUIR'$ solely controls the prediction accuracy.

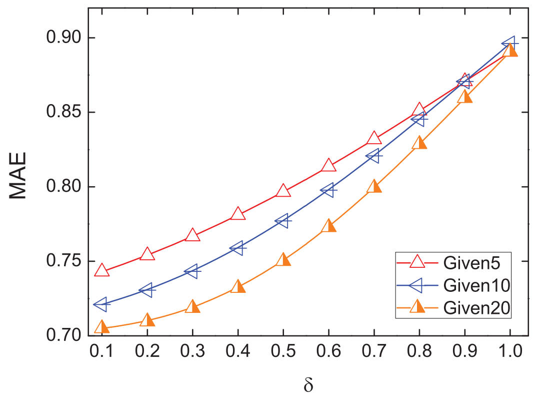

Fig. 7. Sensitivity of δ over ML_300

Fig. 7 illustrates the sensitivity of δ. The MAE of CFSF continuously rises when δ increases from 0.1 to 1. Its mini-

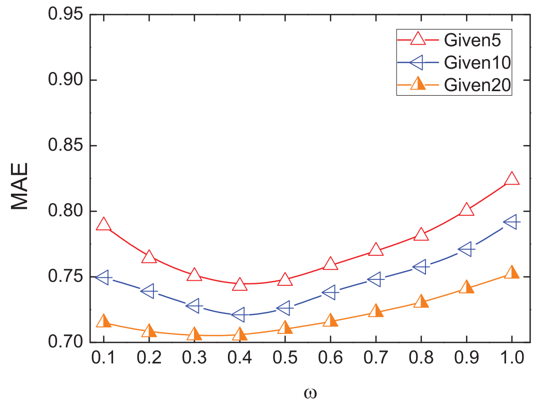

Fig. 8. Sensitivity of w over ML_300

mum for ML_300 is obtained when δ is 0.1. This denotes that $SUIR'$ improves the MAE for CFSF, but not significantly.

3) Sensitivity of w: CFSF introduces w to differentiate smoothed ratings and original ratings. We conducted experiments to check how much parameter w affects the CFSF.

Fig. 8 shows the sensitivity of w. When the value of w is between 0.2 and 0.4, CFSF achieves a high level of accuracy. Otherwise, CFSF achieves poor accuracy because it considers either original or smoothed ratings too much.

VI. Conclusions

In this paper, we have proposed an efficient Collaborative Filtering approach using Smoothing and Fusing (CFSF) to solve two primary problems of CF - data sparsity and limited scalability. The contributions of CFSF are two-fold. First, it offers a mechanism to formulate the CF problem as a local prediction problem, by mapping CF problem from the entire large-scale item-user matrix to a locally reduced item-user matrix. This mechanism significantly decreases the scale of CF problem. Second, for the locally reduced item-user matrix, CFSF presents smoothing and fusing strategies that enable it to achieve high levels of accuracy and scalability.

Currently, CFSF could be further improved. We will capture more aspects of the data for improvement in accuracy, such as dates associated with the ratings and attributes of items and users, which may reflect shifts of user preferences. Moreover, we will study how CFSF can improve its scalability in a parallel manner, as well as how it can keep GIS up-to-date.

Acknowledgements

We would like to thank Weiliang Qian and Xiangyang Liu for helping me develop and test the codes. We also thank Nicole Kwoh and Joanna Izabela Siebert for their technical discussions and proofreading.

This work is supported by the National High Technology Research and Development Program (863 Program) of China (Grant No. 2006AA01Z172 and 2008AA01Z106), the National Natural Science Foundation of China (Grant No. 60533040, 60725208 and 60773089), and Shanghai Pujiang Program (Grant No. 07pj14049).

References

[1] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proc. of the 14th ACM SIGKDD*, 2008, pp. 426–434.
[2] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.
[3] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proc. of the 16th Int. Conf. on World Wide Web*, 2007, pp. 271–280.
[4] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," in *Proc. of the 23rd Conf. on Uncertainty in Artificial Intelligence*, 2007.
[5] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowl. and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
[6] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proc. of the 30th ACM SIGIR*, 2007, pp. 39–46.
[7] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proc. of the 28th ACM SIGIR*, 2005, pp. 114–121.
[8] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proc. of the 29th ACM SIGIR*, 2006, pp. 501–508.
[9] T. Zhang and V. S. Iyengar, "Recommender systems using linear classifiers," *J. Machine Learning Research*, vol. 2, pp. 313–334, 2002.
[10] Y. Zhang and J. Koren, "Efficient bayesian hierarchical user modeling for recommendation system," in *Proc. of the 30th ACM SIGIR*, 2007, pp. 47–54.
[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. of the 10th Int. Conf. on World Wide Web*, 2001, pp. 285–295.
[12] R. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in *Proc. of the 13th ACM SIGKDD*, 2007, pp. 95–104.
[13] J. Kleinberg and M. Sandler, "Using mixture models for collaborative filtering," *J. of Comp. and Sys. Sci*, vol. 74, no. 1, pp. 49–69, 2008.
[14] M. Connor and J. Herlocker, "Clustering items for collaborative filtering," in *Proc. of SIGIR'01 Workshop on Recommender Systems*, 2001.
[15] K. Cheung, K. Tsui, and J. Liu, "Extended latent class models for collaborative recommendation," *IEEE Transactions on Systems, Man, and Cybernatics. Part A: Systems and Humans*, vol. 34, pp. 143–148, 2004.
[16] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.
[17] R. Jin, L. Si, and C. Zhai, "A study of mixture models for collaborative filtering," *Information Retrieval*, vol. 9, no. 3, pp. 357–382, 2006.
[18] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," *IEEE Int. Conf. on Data Mining*, pp. 43–52, 2007.
[19] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach," in *Proc. of 16th Conf. on Uncertainty in Artificial Intelligence*, 2000, pp. 473–480.
[20] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proc. of the 22nd Int. Conf. on Machine Learning*, 2005, pp. 713–719.
[21] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proc. of the 5th ACM Conf. on Digital Libraries*, 2000, pp. 195–204.
[22] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.
[23] R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in *Proc. of the 27th ACM SIGIR*, 2004, pp. 337–344.