Scalable Collaborative Filtering Using Cluster-based Smoothing*

Gui-Rong Xue[1], Chenxi Lin[1], Qiang Yang[3], WenSi Xi[4], Hua-Jun Zeng[2], Yong Yu[1], Zheng Chen[2]

[1]Computer Science and Engineering
Shanghai Jiao-Tong University
Shanghai 200030, P.R.China
{grxue, linchenxi, yyu}@sjtu.edu.cn

[2]Microsoft Research Asia
5F, Sigma Center, 49 Zhichun Road
Beijing 100080, P.R.China
{hjzeng, zhengc}@microsoft.com

[3]Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
qyang@cs.ust.hk

[4]Computer Science
Virginia Polytechnic Institute and State University
Virginia, U.S.A
xwensi@vt.edu

ABSTRACT

Memory-based approaches for collaborative filtering identify the similarity between two users by comparing their ratings on a set of items. In the past, the memory-based approaches have been shown to suffer from two fundamental problems: data sparsity and difficulty in scalability. Alternatively, the model-based approaches have been proposed to alleviate these problems, but these approaches tends to limit the range of users. In this paper, we present a novel approach that combines the advantages of these two kinds of approaches by introducing a smoothing-based method. In our approach, clusters generated from the training data provide the basis for data smoothing and neighborhood selection. As a result, we provide higher accuracy as well as increased efficiency in recommendations. Empirical studies on two datasets (EachMovie and MovieLens) show that our new proposed approach consistently outperforms other state-of-the-art collaborative filtering algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval – *Information Filtering*

General Terms: Algorithms, Performance, Experimentation

Keywords: Collaborative Filtering, Sparsity Data, Smoothing, Clustering.

1. INTRODUCTION

Collaborative filtering predicts the interest of items for an active user based on the aggregated rating information of the like-minded users in a historical database. The key idea is that the active user will prefer those items that like-minded people prefer, or even that dissimilar people don't prefer. Two types of algorithms for collaborative filtering have been studied: memory-based and model-based. Memory-based algorithms perform the computation on the entire database to identify the top *K* most similar users to the active user from the training database in terms of the rating patterns and then combines those ratings together. Notable examples include the Pearson-Correlation based approach [16], the vector similarity based approach [4], and the extended generalized vector-space model [20]. These approaches focused on utilizing the existing rating of a training user as the features. However, the memory-based method suffers from two fundamental problems: data sparsity and inability to scale up. Data sparsity refers to the difficulty that most users rate only a small number of items and hence a very sparse user-item matrix is available. As a result, the accuracy of the method is often quite poor. As for computational scalability, algorithms based on memory-based approaches often cannot cope well with the large numbers of users and items.

In contrast to the memory-based approaches, model-based approaches group different users in the training database into a small number of classes based on their rating patterns. In order to predict the rating from an active user on a particular item, these approaches first categorize the active user into one or more of the predefined user classes and use the rating of the predicted classes on the targeted item as the prediction. Algorithms within this category include Bayesian network approach [4], clustering approach [13][21] and the aspect models [12].The model-based approaches are often time-consuming to build and update, and cannot cover as diverse a user range as the memory-based approaches do.

In this paper, we propose a novel framework for collaborative filtering which combines the strengths of memory-based approaches and model-based approaches in order to enable recommendation by groups of closely related individuals. Our method uses the clusters as the computed groups and smoothes the unrated data for individual users. The use of clusters for smoothing permits the integration of the advantages from both the memory-based and model-based approaches. By using the rating information from a group of closely related users, unrated items of the individual user in a group can be predicted; this allows the

*This work was conducted while the first author was doing internship at Microsoft Research Asia.

missing values to be filled in. Moreover, assuming that the nearest neighbor should be also in Top N most similar clusters to the active user, we need only select the nearest neighbors in the set of the Top N clusters. This enables the system to be scalable.

In the rest of the paper, we provide a brief description of several major approaches for collaborative filtering and the related work. In section 3, we propose general framework for collaborative filtering. The results of empirical studies are presented in Section 4, followed by conclusion in Section 5.

2. BACKGROUND

In this section, we review several major approaches for collaborative filtering.

2.1 Collaborative Filtering

2.1.1 Memory-based Approaches

The memory-based approaches [4] are among the most popular prediction techniques in collaborative filtering. The basic idea is to compute the active user's predicted vote of an item as a weighted average of votes by other similar users or K nearest neighbors (KNN). Two commonly used memory-based algorithms are the Pearson Correlation Coefficient (PCC) algorithm [16] and the Vector Space Similarity (VSS) algorithm [4]. These two approaches differ in the computation of similarity. As described in [4], the PCC algorithm generally achieves higher performance than vector-space similarity method.

2.1.2 Model-based Approaches

Two popular model-based algorithms are the clustering for collaborative filtering [13][21] and the aspect models [12]. *Clustering techniques* work by identifying groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Some clustering techniques represent each user with partial participation in several clusters. The prediction is then an average across the clusters, weighted by the degree of participation.

The *aspect model* [12] is a probabilistic latent-space model, which considers individual preferences as a convex combination of preference factors. The latent class variable is associated with each observation pair of a user and an item. The aspect model assumes that users and items are independent from each other given the latent class variable.

2.1.3 Hybrid Model

Pennock et al. [15] proposed a hybrid memory- and model-based approach. Given a user's preferences for some items, they compute the probability that a user belongs to the same "personality diagnosis" by assigning the missing rating as a uniform distribution over all possible ratings [15]. Previous empirical studies have shown that the method is able to outperform several other approaches for collaborative filtering [15], including the PCC method, the VSS method and the Bayesian network approach. However, the method neither takes the whole aggregated information of the training database into account nor considers the diversity among users when rating the non-rated items. From our point of view, the clustering-based smoothing could provide more representative information for the rating.

2.2 OTHER RELATED WORK

Several other related methods have also been proposed to deal with the sparsity problem. The dimension-reduction method aims to reduce the dimensionality of the user-item matrix directly. A simple strategy is to form clusters of users or items and then use these clusters as basic units in making recommendation. Principle Component Analysis (PCA) [8] and information retrieval techniques such as Latent Semantic Indexing (LSI) [7][18] are also proposed. Zeng [23] proposed to compute the users' similarity by a matrix conversion method for similarity measure. The dimensionality-reduction approach addresses the sparsity problem by removing unrepresentative or insignificant users or items so as to condense the user-item matrix. However, potentially useful information might be lost during this reduction process. By considering the association between users and items, transitive associations of the associative-retrieval technique [11] are proposed to iteratively reinforce the similarity of the users and the similarity of items.

Content-boosted CF [1][5] approaches require additional information regarding items as well as a metric to compute meaningful similarities among them. In [17], A. Popescul et al. also proposed a unified probabilistic model for integrating content information to solve the sparse-data problem. Most previous studies have demonstrated significant improvement in recommendation quality. However, in practice, such item information may be difficult or expensive to acquire.

Sarwar et al. [19] proposed an item-based approach to addressing both the scalability and sparsity problems. Given an item, similar items rated by the active user in the past are identified and then used for recommendation. Item similarities are computed as the correlations between the corresponding column (item) vectors.

3. CLUSTER-BASED COLLABORATIVE FILTERING FRAMEWORK

```
Algorithm: Cluster-Smoothed CF
  ● Preprocess: create user clusters $C$
      (we use a K-means algorithm; see below)
  ● Given an active user $u_a$ and $i$ rated items, an item $t$ and
    an integer $K$, the number of nearest neighbors:
    1. Choose $s$ users into $G$ from groups that are most similar to $u_a$.
    2. Calculate similarity $sim(u_a, u)$ for each $u$ in $G$ in which
       $u$'s rating is the combination of the $R_u(t)$ and $R_{Cu}(t)$
    3. Select the top-$K$ most similar users as the nearest neighbors
    4. Predict the rating of a particular item $t$ for $u_a$ by the
       behaviors of the $K$ nearest neighbors.
```

Figure 1. Clustering-based smoothing collaborative filtering

We first define the notations that are used throughout this paper. Let $T=\{t_1, t_2, \ldots, t_m\}$ be a set of items, $U=\{ u_1, u_2, \ldots, u_n \}$ be a set of users in the database, and u_a be an active user – the user for whom we need to provide recommendations for items that the user has not seen before. Let $\{(u_{(1)}, i_{(1)}, r_{(1)}), \ldots, (u_{(k)}, i_{(k)}, r_{(k)})\}$ be all the ratings found in the training database. Each triple $((u_{(i)}, t_{(i)}, r_{(i)})$ indicates that the item $t_{(i)}$ is rated as $r_{(i)}$ by the user $u_{(i)}$. For each user u, $R_u(t)$ denotes the rating of item t by user u, and

$\overline{R_u}$ denotes his average rating. The rating scale goes from 1 to r_{max}.

Our clustering-based smoothing algorithm is shown in Figure 1. The framework is fairly general to include both the memory-based and the model-based approaches. In the algorithm, the choice of a cluster (u_a) corresponds to selecting a similar user group. Scoring step and prediction step integrate the smoothing operation with recommendation.

3.1 Clustering Algorithms

There are many algorithms that can be used to create clusters. In this paper, a *K*-means algorithm is selected as the basic clustering algorithm. The number *k* is an input to the algorithm that specifies the desired number of clusters. In the first pass, the algorithm takes the first *k* users as the centroids of *k* unique clusters. Each of the remaining users is then compared to the closest centroid. In the following passes, the cluster centroids are re-computed based on cluster centroids formed in the previous pass and the cluster-membership is re-evaluated. The running time of this algorithm for each pass is linear in the total number (*N*) of users to be clustered; i.e. the computational time is O(k^2N).

Assuming that users could be clustered into *N* groups, clustering results of the users $U=\{u_1,u_2,\ldots,u_n\}$ are represented as $\{C_u^1, C_u^2, \cdots, C_u^k\}$.

We take the Pearson correlation-coefficient function as a similarity measure function. The similarity between user *u* and user *u'* is defined as:

$$sim_{u,u'} = \frac{\sum_{t \in T(u) \wedge T(u')} (R_u(t) - \overline{R_u}) \cdot (R_{u'}(t) - \overline{R_{u'}})}{\sqrt{\sum_{t \in T(u) \wedge T(u')} (R_u(t) - \overline{R_u})^2} \sqrt{\sum_{t \in T(u) \wedge T(u')} (R_{u'}(t) - \overline{R_{u'}})^2}} \quad (1)$$

3.2 Data Smoothing

As we discussed above, data sparsity is a fundamental problem for collaborative filtering. To fill the missing values in data set, we make explicit use of clusters as smoothing mechanisms. Cluster-based smoothing technique for nature language processing [3] is successful to estimate probability of the unseen term by using the topic (cluster) of the term belongs to, which motivate us to examine the sparsity problem on collaborative filtering.

Based on the clustering results, we apply the smoothing strategies to the unseen rating data. We first define a special rating value as follows:

$$R_u(t) = \begin{cases} R_u(t) & \text{if user } u \text{ rate the item } t \\ \hat{R}_u(t) & \text{else} \end{cases} \quad (2)$$

where $\hat{R}_u(t)$ denotes the smoothed value for user *u's* rating towards an item *t*.

Given a user *u*, $C_u \in \{C_u^1, C_u^2, \cdots, C_u^k\}$ refers to the cluster the user belong to. By considering the diversity of the individual, we propose to use the following equation to calculate $\hat{R}_u(t)$.

$$\hat{R}_u(t) = \overline{R_u} + \Delta R_{C_u}(t) \quad (3)$$

where $\Delta R_{C_u}(t)$ is average deviations rating for all users in cluster C_u to the item *t*, which is defined as:

$$\Delta R_{C_u}(t) = \sum_{u' \in C_u(t)} (R_{u'}(t) - \overline{R_{u'}}) / |C_u(t)| \quad (4)$$

where $C_u(t) \in C_u$ is the user set that the users in cluster C_u who have rated item *t*. $|C_u(t)|$ is the number of users in cluster C_u who have rated the item *t*.

3.3 Neighbor Pre-Selection

An important step of collaborative filtering algorithm is to search neighbors of an active user. Traditional method is to search the whole database. Apparently this method suffers from poor scalability when more and more new users and new items are added into the database. By using the concept of a cluster, we can do better. The feature of the group of users in a cluster is represented by the centroid of the cluster. This centroid is represented as an average rating over all users in the cluster. To compute a similar set of users in a cluster, the similarity between group *C* of users and the active users is also calculated based on the following function:

$$sim_{u_a,C} = \frac{\sum_{t \in T(u_a) \wedge T(C)} \Delta R_C(t) \cdot (R_{u_a}(t) - \overline{R_{u_a}})}{\sqrt{\sum_{t \in T(u_a) \wedge T(C)} (\Delta R_C(t))^2} \sqrt{\sum_{t \in T(u_a) \wedge T(C)} (R_{u_a}(t) - \overline{R_{u_a}})^2}}. \quad (5)$$

After calculating the similarity between each group and the active user, we take the users in the most similar groups as the candidates. From the process, the cluster can help speed up the computation of similarity calculation as well as remove some irrelevant information.

3.4 Neighbor Selection

After pre-selection, we also need to re-calculate the similarity between user in the candidate set and the active user on the smoothed rating.

After smoothing by the cluster information, user's rating value contains two parts: original rating and group rating. In this paper, the different weight is considered between user's original rating and group rating when calculating the similarity between the user in the candidate set and the active user. That is, we set w_{ut} as confidential weight for the user *u* to the items *t*.

$$w_{ut} = \begin{cases} 1-\lambda & \text{if user } u \text{ rate the item } t \\ \lambda & \text{else} \end{cases} \quad (6)$$

where λ is the parameter for tuning the weight between original rating and group rating. The value of λ is varied from 0 to 1.

Then, we select the Top *K* most similar users based on the following similarity function:

$$sim_{u_a,u} = \frac{\sum_{t \in T(u_a)} w_{ut} \cdot (R_u(t) - \overline{R_u}) \cdot (R_{u_a}(t) - \overline{R_{u_a}})}{\sqrt{\sum_{t \in T(u_a)} w_{ut}^2 \cdot (R_u(t) - \overline{R_u})^2} \sqrt{\sum_{t \in T(u_a)} (R_{u_a}(t) - \overline{R_{u_a}})^2}} \qquad (7)$$

By assigning different value to λ we can adjust the weights of different rating in the overall similarity. For example, when λ is set to 0, the algorithm is the basic PCC algorithm that only uses the rated information for similarity computation and prediction. While if λ is set to 1, the algorithm is the basic cluster-based collaborative filtering algorithm which just uses the average rating of clustering for similarity computation and prediction.

3.5 Prediction

In making a prediction, a subset of K most similar users is chosen based on their similarity to the active user, and a weighted aggregate of their ratings is used to generate predictions for the active user as follows. The predictions are computed as the weighted average of deviations from the neighbor's mean:

$$R_{u_a}(t) = \overline{R_{u_a}} + \frac{\sum_{i=1}^{K} w_{ut} \cdot sim_{u_a,u} \cdot (R_u(t) - \overline{R_u})}{\sum_{i=1}^{K} w_{ut} \cdot sim_{u_a,u}} \qquad (8)$$

where $sim_{u_a,u}$ is the similarity between the active user u_a and the training user u, and K is the number of users in the neighborhood.

As shown in Table 1, our framework is very flexible by combine neighbor pre-selection and smoothing.

Table 1. Algorithms Specifications

	No Pre-selection	Pre-selection
No smoothing	PCC	SPCC
Smoothing	CBPCC	SCBPCC
Clustering	CBCF	---------

Here PCC is Pearson Correlation Coefficient algorithm, CBCF is cluster-based collaborative filtering. CBPCC is cluster-based Pearson Correlation Coefficient algorithm which just utilizes the cluster for smoothing. SPCC is scalable Pearson Correlation Coefficient algorithm, which use the cluster for neighbor pre-selection. SCBPCC is the algorithm that uses the cluster for neighbor pre-selection and for smoothing.

4. EXPERIMENTS

We conduct a set of experiments to examine the effectiveness of our new scheme for collaborative filtering in terms of scalability and recommendation quality. In particular, we address the following issues:

1) How does the confidence parameter affect the performance of prediction? As we discussed in Section 3, tuning the parameter could leverage the degree of smoothing. In this paper, we conduct experiments to show the accuracy of prediction on different parameter values.

2) How does the neighbor-selection method affect the performance of predication and speed up the calculation of the algorithm? Experiments are conducted to examine the accuracy of cluster-based neighbor pre-selection and the efficiency of the algorithm.

3) How do the clusters found influence the prediction accuracy? As described above, the number of clusters and the clustering methods tend to affect the performance of prediction. Experiments are conducted to examine the impact of clustering methods on the final performance of collaborative filtering.

4) How do the approaches under the newly proposed framework compare with existing collaborative filtering approaches? We compared them to standard collaborative filtering approaches including Pearson Correlation Coefficient (PCC), Vector similarity (VS), Aspect Model (AM), and Personality Diagnosis (PD).

4.1 Dataset

Two datasets from movie rating are used in our experiments: MovieLens (http://www.cs.umn.edu/Research/GroupLens/) and EachMovie [1]. For MovieLens dataset, we extracted a subset of 500 users with more than 40 ratings. For EachMovie dataset, we extracted a subset of 10,000 users with more than 40 ratings. The global statistics of these two datasets as used in our experiments are summarized in Table 2. To compare algorithms thoroughly, we experimented with several different configurations. For MovieLens we altered the training size to be the first 100, 200, and 300 users, which are denoted as ML_100, ML_200, and ML_300, respectively. For EachMovie we used the first 500, 2000 and 6000 users for training, which are denoted as EM_500, EM_2000 and EM_6000, respectively. For different training size, the test set we used is fixed to same size. For MovieLens, we use the last 200 users for testing, and for EachMovie we used the last 4000 users.

Table 2. Characteristics of MovieRating and EachMovie

	MovieLens (ML)	EachMovie (EM)
Number of Uses	500	10000
Number of Items	1000	1682
Avg. # of rated Items/User	87.7	101.1
Density of data	8.77%	6.01%
Number of Ratings	5	6

4.2 Metrics and Methodology

We use the Mean Absolute Error (MAE) [10], a statistical accuracy metrics, to measure the prediction quality metric:

$$MAE = \frac{\sum_{u \in T} |R_u(t_j) - \tilde{R_u}(t_j)|}{|T|} \qquad (9)$$

[1] EachMovie dataset is provided by the Compaq Systems Research Center. For more information see http://www.research.digital.com/SRC/EachMovie/.

where $R_u(t_j)$ is the rating given to item t_j by user u, $\tilde{R}_u(t_j)$ is the predicted value of user u on item t_j, T is the test set, and $|T|$ is the size of the test set.

For each active user, we varied the number of rated items provided by the active user from 5, 10, to 20, which named *Given5*, *Given10* and *Given20*, respectively.

4.3 Performance

As we mentioned above, our algorithm could alleviate two fundamental problems: data sparsity and scalability. In this sub-section, we will conduct an experiment to show the performance of our proposed framework.

4.3.1 Overall Performance

In order to show the performance of our approach to collaborative filtering, we compare our algorithm cluster-based Pearson Correlation Coefficient (SCBPCC) with the state-of-art algorithms for collaborative filtering: Pearson Correlation Coefficient (PCC), Personality Diagnosis (PD), Aspect Model (AM) and Cluster-based collaborative filtering (CBCF).

Several parameters for our experiments need be set in the following experiments i.e. λ=0.35, the cluster number K=50 for the EachMovie dataset and the cluster number K=20 for the MovieRating dataset. The percentage of pre-selected neighbors is about 30% of whole users. The number of nearest number is set to 20.

Table 3 and Table 4 summarize the results for these five methods. Clearly, SCBPCC outperforms other three methods in all configurations. By utilizing the clusters as a smoothing method for the missing data, our new smoothing scheme is found to be effective in improving the prediction accuracy for collaborative filtering.

Table 3. MAE on MovieLens for different algorithms. (A small value means a better performance)

Training Set	Methods	Given5	Given10	Given20
ML_100	PCC	0.874	0.836	0.818
	PD	0.849	0.817	0.808
	AM	0.963	0.922	0.887
	CBCF	0.924	0.896	0.890
	SCBPCC	**0.848**	**0.819**	**0.789**
ML_200	PCC	0.859	0.829	0.813
	PD	0.836	0.815	0.792
	AM	0.849	0.837	0.815
	CBCF	0.908	0.879	0.852
	SCBPCC	**0.831**	**0.813**	**0.784**
ML_300	PCC	0.849	0.841	0.820
	PD	0.827	0.815	0.789
	AM	0.820	0.822	0.796
	CBCF	0.847	0.846	0.821
	SCBPCC	**0.822**	**0.810**	**0.778**

Table 4. MAE on EachMovie for different algorithms. (A small value means a better performance.)

Training Set	Methods	Given5	Given10	Given20
EM_500	PCC	1.157	1.075	1.048
	PD	1.148	1.145	1.140
	AM	1.157	1.082	1.057
	CBCF	1.207	1.132	1.089
	SCBPCC	**1.105**	**1.041**	**1.004**
EM_2000	PCC	1.124	1.052	1.020
	PD	1.120	1.087	1.043
	AM	1.125	1.078	1.054
	CBCF	1.187	1.113	1.063
	SCBPCC	**1.085**	**1.014**	**0.973**
EM_6000	PCC	1.118	1.039	0.988
	PD	1.101	1.063	1.051
	AM	1.117	1.069	1.046
	CBCF	1.197	1.111	1.060
	SCBPCC	**1.073**	**1.001**	**0.956**

4.3.2 Performance on Sparsity Data

The density of a rating matrix can have a significant impact on the performance of collaborative filtering. To show the performance of our proposed approach, we conducted an experiment to simulate the phenomenon of the sparseness of rating matrix and compare the performance about four algorithms: PCC, PD, CBCF and AM. In Figure 2, we empirically analyze how MAE evolves with the density of rating matrix. In this experiment, we randomly select 20%, 40%, 60%, 80% and 100% of whole rating data on the dataset EM_6000 to represent different degrees of density of the rating matrix. The number of nearest neighbors is set to 20 while the number of rated items for active user is set to 20.

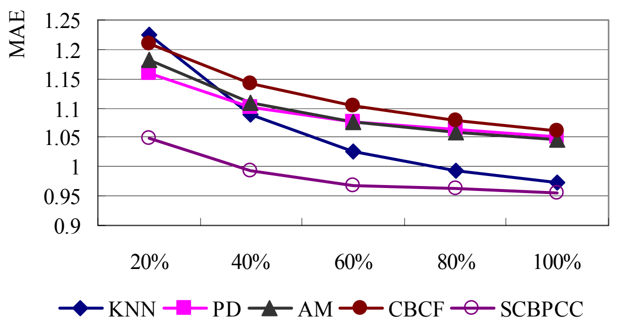

Figure 2. MAE on different density of EachMovie data (A small value means a better performance)

The results show that indeed the density has a great effect on the performance of different algorithms. When the rating matrix becomes denser, all algorithms tend to achieve higher performance. As seen from Figure 2, the MAE curve of our algorithm is below that of the other algorithms, which means that the sparseness has the least impact on our proposed algorithm.

4.3.3 Scalability with Neighbor Selection

Generally, memory-based approaches online-compute the similarity of the active user with all the training users in the database to select *K* nearest neighbor (KNN). The efficiency will be affected by the number of users. In our framework, we perform the KNN computation on a subset which is pre-selected by computing the similarity between the active users and a cluster of users. The subset is selected from the most similar clusters to the active user. Thus, we could speed up the computation. We conduct an experiment on the data set EM_6000 and the subset of the pre-selected neighbor is increased from 10% to 100%. The number of nearest neighbors is set to 20 while the number of rated items for active user is set to 20. We conduct the experiments by using scalable Pearson Correlation Coefficient algorithm (SPCC), scalable cluster-based Pearson Correlation Coefficient (SCBPCC) and Pearson Correlation Coefficient algorithm (PCC).

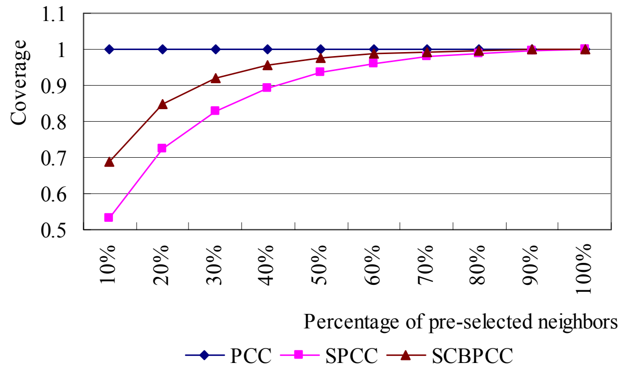

Figure 3. Coverage on different percentage of pre-selected neighbors on EM_6000

The first experiment shows the coverage of Top 20 neighbors generated by our smoothed SCBPCC method within a subset of the pre-selected neighbors. The X-axis is the percentage of per-selected users from the most similar clusters. The Y-axis shows the coverage ratio of 20 nearest neighbors in the subset. As shown in Figure 3, the coverage will increase when the number of pre-selected neighbors is increased. When the size of the pre-selected subset is about 40% of the whole dataset, about 90%~95% of the top *K* nearest neighbors will be found in the pre-selected subset. Furthermore, the curve of our algorithm is above the SPCC algorithm, showing the effect of smoothing by clustering.

The second experiment shows the performance as a function of the pre-selected neighbor. As shown in Figure 4, SPCC and SCBPCC algorithms achieve relatively stable performance when the percentage of pre-selected neighbors is about 30%. SPCC will achieve higher performance when the percentage neighbors is about 20% ~30%, which can verify that the neighbor pre-selection can remove dissimilar users and to improve the performance.

As we discussed in Section 3, the online execution time for finding similar users is time-consuming. The general PCC algorithm needs to scan the entire database. By using the cluster for neighbors pre-selection, the execution time can be reduced. The computation of our proposed framework consists two parts: (1) calculating the active user's similarity with the clusters and the active user; (2) calculating the active user's similarity with pre-selected neighbors. The whole execution time is the sum of the two parts. We conduct the experiments to compare PCC and SPCC algorithm on the EM_6000 data. Here we cluster the users into 50 clusters.

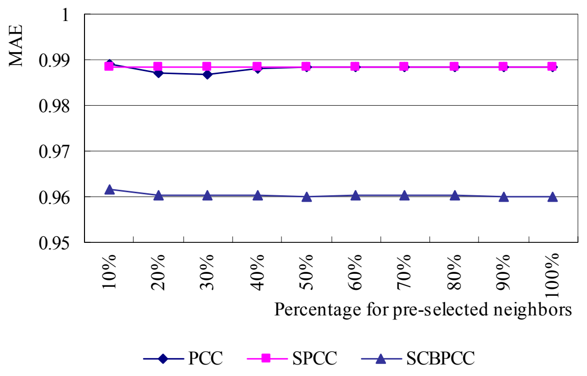

Figure 4. MAE on different percentage of pre-selected neighbors on EM_6000 (A small value means a better performance)

The execution time for scalable Pearson Correlation Coefficient algorithm SPCC and Pearson Correlation Coefficient algorithm (PCC) are shown in Figure 5. With increase of the pre-selected neighbors, the execution time will increase quickly. According to the MAE value in Figure 4, if we use the cluster as a neighbor pre-selection and select 30% of the whole users as the candidates, most of the execution time could be saved.

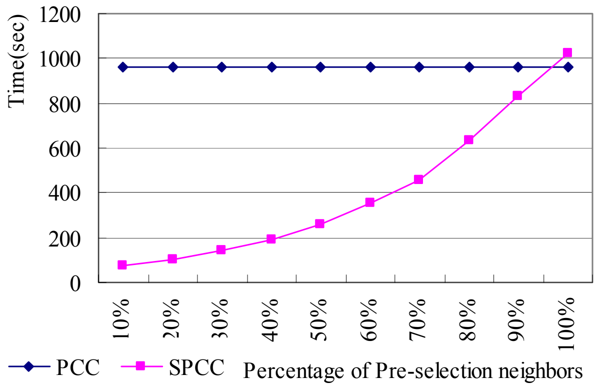

Figure 5. Running Time on different percentage of pre-selected neighbors on EM_6000

4.4 Parameters Tuning

In this section, we conduct the experiments to relate the smoothing parameters and the performance of the algorithms. In the following experiments, we perform the experiment by using the cluster-based Pearson Correlation Coefficient algorithm (CBPCC).

4.4.1 Smoothing Parameter Selection

As shown in Equation 6, we give a confidence weight when we use the cluster information to smooth the unrated data for the training users. By assigning a different confidential value to w_{ij} to show how much the training user relies on the clusters, the performance will be affected.

The value of λ is varied from 0 to 1. When setting λ to 0, the algorithm only uses the rated information for similarity computation and prediction. When λ is set to 1, the algorithm just use the average rating of clustering for similarity computation and prediction. We tune the value λ to show the performance on prediction.

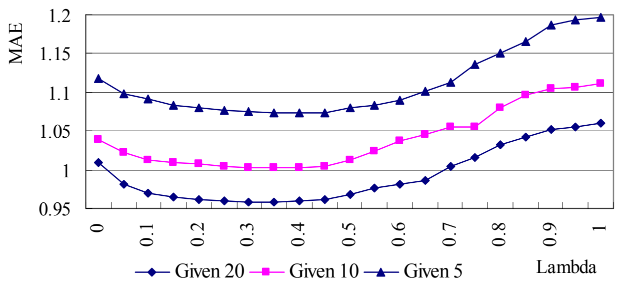

Figure 6. MAE of different λ on EM_6000 (A small value means a better performance)

We first vary the value of λ by different protocols on EM_6000 to show the performance of a given different protocol. We have also conducted tests on two datasets ML_100 and EM_6000 by Given20. As shown in Figure 6 and Figure 7, the algorithm can achieve the best performance on different protocols and on different datasets when λ is set to 0.35. When λ is higher than 0.35, which means that we rely heavily on the cluster information, the performance will decrease. When λ is less than 0.35, which means that we rely less on the cluster information, the data sparseness will cause the lower performance.

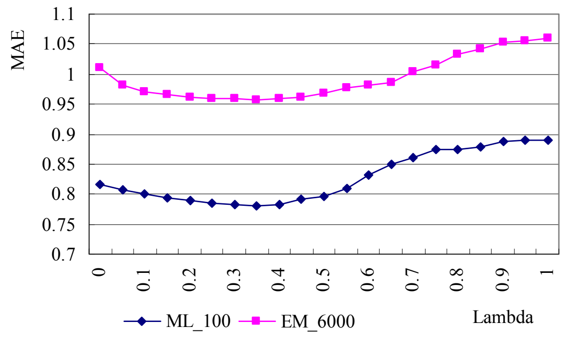

Figure 7. MAE of different λ on EM_6000 and MR_100 (A small value means a better performance)

4.4.2 Clustering Number Selection

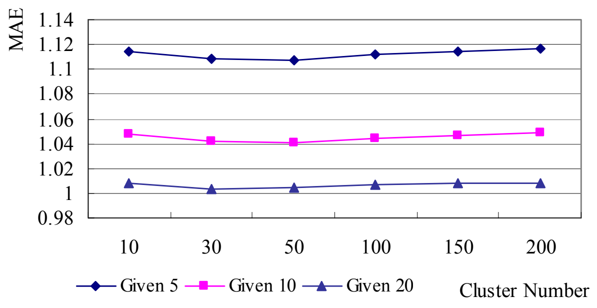

Figure 8. MAE of different cluster number on EM_6000 (A small value means a better performance)

In order to select a suitable number of clusters, we use the EM_6000 dataset for training. We apply the algorithms to group users into clusters based on six different values of K (10, 30, 50, 100, 150, and 200). We then perform collaborative filtering on each group of clusters. As shown in Figure 8, MAE of the best performance is about 50 for dataset EM_6000.

As shown in Figure 7, the number of clusters does have an effect on the performance of our algorithms. The large number of clusters makes the cluster information more specific while the small ones cause the cluster information too general to represent difference among the dissimilar users.

4.4.3 Different Smoothing Strategies Comparison

It is natural to consider that the average rating of all the users on an item that could be used for smoothing the missing data. In order to compare smoothing based on all users and smoothing based on clusters, we conducted experiments on three dataset EM_500, EM_2000 and EM_6000 by *Given5*, *Given10* and *Given20*, respectively.

Table 5. MAE on different smoothing methods (A small value means a better performance)

	EM_500		EM_2000		EM_6000	
	Global	Cluster	Global	Cluster	Global	Cluster
5	1.121	**1.105**	1.113	**1.085**	1.104	**1.073**
10	1.059	**1.041**	1.048	**1.014**	1.035	**1.001**
20	1.027	**1.004**	1.012	**0.973**	0.998	**0.956**

As shown in Table 5, the performance of using all users' data (Global) for smoothing cannot achieve the best result. This is because the data from all users correspond to using only one global cluster, which performs smoothing at too coarse a level.

5. CONCLUSION

In this paper, we have proposed a novel framework for collaborative filtering. By integrating the advantages of memory- and model-based collaborative filtering into a single framework, our approach targets two fundamental problems: data sparsity and scalability. We used clusters to provide smoothing operations to solve the missing-value problems. Experimental results show that our proposed framework can significantly improve the accuracy of predication as well as solve the scalability problem.

For future work, we have begun to investigate how to automatically learn the smoothing parameters according to the features of the users. For example, the rating number, the confidence of the rating, etc. We also want to develop a principled probabilistic interpretation of the framework we have proposed. Furthermore, we wish to find an automatic method such that the estimated optimal number of clusters would produce more accurate predictions.

6. REFERENCES

[1] M. Balabanovic, Y. Shoham. Fab: Content-based, Collaborative Recommendation. Communication of the ACM, Mar. 1997, 40(3): 66-72.

[2] D. Billsus and M. J. Pazzani. Learning Collaborative Information Filters. In Proc. 15th International Conf. on

Machine Learning, pages 46--54. Morgan Kaufmann, San Francisco, CA, 1998.

[3] P. F. Brown, V. J. D. Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based N-gram Models of Natural Language. Computational Linguistics, 18(4):467-479, 1992.

[4] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of Predictive Algorithms for Collaborative Filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998. 43-52.

[5] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining Content-based and Collaborative Filters in an Online Newspaper. In Proceedings of ACM SIGIR Workshop on Recommender, August 1999.

[6] M. O. Conner and J. Herlocker. Clustering Items for Collaborative Filtering. In Proceedings of the ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, August 1999.

[7] D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas, and R, Vuduc. SWAMI: a Framework for Collaborative Filtering Algorithm Development and Evaluation. In Proceedings of the 23rd Annual International Conference on Researech and Development in Information Retrieval (SIGIR), 2000.

[8] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval, 4(2):133--151, 2001.

[9] J. Herlocker, J. Konstan, A. Brochers, and J. Riedl. An Algorithm Framework for Performing Collaborative Filtering. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 1999.

[10] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems 22 (2004), ACM Press, 5-53.

[11] Z. Huang, H. Chen, D. Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. ACM Transactions on Information Systems, 22 (1), 2004.

[12] T. Hofmann and J. Puzicha, Latent Class Models for Collaborative Filtering. In Proceedings of the 16th International Joint Conference on Artificial Intelligence, 1999, 688-693.

[13] A. Kohrs and B. Merialdo. Clustering for Collaborative Filtering Applications. In Proceedings of CIMCA'99. IOS Press, 1999.

[14] J. Konstan, B. Miller, D. Maltz. GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM, Mar. 1997, 40(3): 77-87.

[15] D. M. Pennock, E. Horvitz, S. Lawrence and C. L. Giles. Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach, in Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI), 2000.

[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of the ACM Conference on Computer Supported Cooperative Work, pages 175--186, 1994.

[17] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence. Probabilistic Models for Unified Collaborative and Content-based Recommendation in Sparse-data Environments. In 17th Conference on Uncertainty in Artificial Intelligence, 2001.

[18] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl. Application of Dimensionality Reduction in Recommender Systems: a Case Study. In ACM WebKDD Workshop,2000.

[19] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In: Proceedings of the 10th International World Wide Web Conference. Hong Kong (2001)

[20] I. M. Soboroff and C. Nicholas. Collaborative Filtering and the Generalized Vector Space Model. In Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval (SIGIR), 2000.

[21] L. H. Ungar and D. P. Foster. Clustering Methods for Collaborative Filtering. In Proc. Workshop on Recommendation Systems at the 15th National Conf. on Artificial Intelligence. Menlo Park, CA: AAAI Press.

[22] K. Yu, Z. Wen, X.W Xu, and M. Ester. Feature Weighting and Instance Selection for Collaborative Filtering, in Proceedings of 2nd International Workshop on Management of Information on the Web: Web Data and Text Mining (MIW'01), 2001.

[23] C. Zeng, C.-X. Xing, L.-Z. Zhou. Similarity Measure and Instance Selection for Collaborative Filtering. In Proceedings. of 12th International World Wide Web conference, 2003.