Similarity Measure and Instance Selection for Collaborative Filtering*

Chun Zeng, Chun-Xiao Xing, Li-Zhu Zhou
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China
bobofu00@mails.tsinghua.edu.cn, {xingcx, dcszlz}@tsinghua.edu.cn

ABSTRACT
Collaborative filtering has been very successful in both research and applications such as information filtering and E-commerce. The k-Nearest Neighbor (KNN) method is a popular way for its realization. Its key technique is to find k nearest neighbors for a given user to predict his interests. However, this method suffers from two fundamental problems: sparsity and scalability. In this paper, we present our solutions for these two problems. We adopt two techniques: a matrix conversion method for similarity measure and an instance selection method. And then we present an improved collaborative filtering algorithm based on these two methods. In contrast with existing collaborative algorithms, our method shows its satisfactory accuracy and performance.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *information filtering*; H.5.3 [**Information Interfaces and Presentation (I.7)**]: Group and Organization Interfaces – *collaborative computing*.

General Terms
Algorithms, Performance, Experimentation.

Keywords
Collaborative Filtering, Similarity Measure, Instance Selection

1. INTRODUCTION
Nowadays World Wide Web has become an important way to get information. With the rapid growing of the Web, people have to spend more and more time to search what they need. To deal with this difficulty, many tools have been developed. Search engines, such as Alta Vista (www.altavista.com), Yahoo! (www.yahoo.com) and Google (www.google.com), are the most popular way. These retrieval systems satisfy users' need to a great extent. However, because of their all-purpose characteristics, they can't satisfy any query requests from different background, with different intention and at different time. Personalization is a way to address this problem by providing different services for different users. It can actively recommend items to users according to their interests and behaviors.

Personalization systems for the Web can be classified into three types: rule-based systems, content-based filtering systems and collaborative filtering systems [1]. Rule-based systems, such as Broadvision (www.broadvision.com), allow Website administrators to specify rules used to determine the contents served to individual users. Content-based filtering systems, such as Letizia [2], Personal WebWatcher [3], Syskill & Webert [4], associate every user with profiles to filter contents. Collaborative filtering systems, such as GroupLens [5], WebWatcher [6] and Let's Browse [7], utilize the similarity among profiles of users to recommend interesting materials.

Collaborative filtering has been very successful in both research and applications such as information filtering and E-commerce. The k-Nearest Neighbor (KNN) method is a popular memory-based way for its realization. Its key technique is to find k nearest neighbors for a given user to predict his interest. However, this method suffers from two fundamental problems: sparsity and scalability. Sparsity refers to the fact that most users do not rate most items and hence a very sparse user-item rating matrix is generated. As a result the accuracy of the method will be poor. In scalability aspect, the nearest neighbor algorithm suffers serious scalability problems in failing to scale up its computation with the growth of both the number of users and the number of items.

To solve the first problem, Balabanovic et al [8] and Claypool et al [9] put forward a content-based collaborative filtering method, which utilizes the contents browsed by users to compute the similarity among users. Sarwar et al [10] uses Latent Semantic Indexing (LSI) to capture the similarity among users and items in a reduced dimensional space. Yu et al [11] uses a feature-weighting method to improve the accuracy of collaborative filtering algorithms.

To solve the second problem, many studies bring forward model-based methods that use the users' preferences to learn a model, which is then used for predications. Breese et al [12] utilizes clustering and Bayesian network approaches. Its results show that the clustering-based method is the most efficient but suffering from poor accuracy. And the Bayesian networks may prove practical for environments in which knowledge of user preferences changes slowly with respect to the time needed to build the model but are not suitable for environments in which user preference models must be updated rapidly or frequently. Sarwar et al [13] put forth an item-based collaborative filtering algorithm utilizes a pre-computed model of item similarity to increase the online scalability of item-based recommendations. Sarwar et al [14] presents a rule-based approach using association rule discovery algorithms to find association between relevant items and then generates item recommendation based on the strength of the association between items. Yu et al [15] solves the second problem from another point of view. It adopts a technique of instance selection to remove the irrelevant and redundant instances from the training set. Its results show that this method improves not only the accuracy but also the performance of the

* Supported by the National Natural Science Foundation of China under Grant No. 60221120146; the National Grand Fundamental Research 973 Program of China under Grant No.G1999032704.

WWW 2003, May 20-24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

collaborative filtering algorithm.

In this paper, we present a matrix conversion method for similarity measure to improve the accuracy of the collaborative filtering algorithm. Furthermore, we introduce a new instance selection method to reduce the size of training set and improve the performance of prediction without influencing the accuracy.

The rest of the paper is organized as follows. The next section provides a brief background in collaborative filtering algorithms. Section 3 discusses our method for similarity measure. In section 4, we describe the technique of instance selection in detail. Section 5 describes an improved collaborative filtering algorithm based on these two methods. Section 6 describes our experimental work including details of our data sets, evaluation metrics, methodology and results of different experiments and discussion of the results. The final section provides some concluding remarks and directions for future research.

2. COLLABORATION FILTERING

The task of collaborative filtering is to predict the preference of an active user to a target item based on user preferences. There are two general classes of collaborative filtering algorithms: memory-based methods and model-based methods [12].

2.1 Memory-based Algorithm

The memory-based algorithm [16] is the most popular prediction technique in collaborative filtering applications. Its basic idea is to compute the active user's predicted vote of an item as a weighted average of the votes given to that item by other users. The user-based collaborative filtering algorithm is a memory-based algorithm. It can be summarized in the following steps:

1. Similarity between users is measured as the Pearson correlation between their rating vectors.
2. Select *n* neighboring users that have the highest similarity with the active user.
3. Compute a prediction from a combination of the selected neighboring users' ratings.

In step 1, similarity between users is computed using Pearson correlation coefficient, defined by

$$w_{a,b} = \frac{\sum_{j=1}^{M}(v_{a,j} - \bar{v}_a)(v_{b,j} - \bar{v}_b)}{\sqrt{\sum_{j=1}^{M}(v_{a,j} - \bar{v}_a)^2 \sum_{j=1}^{M}(v_{b,j} - \bar{v}_b)^2}} \quad (1)$$

Where M is the number of items, $v_{a,j}$ is the rating given to item j by user a, $\bar{v}_a$ is the mean rating given by user a, $w_{a,b}$ is the similarity between user a and user b.

Pearson correlation only measures the overlapping items between users. To devalue the correlations based on few co-rated items we multiply the correlation by a significance-weighting factor [17]:

$$w'_{a,b} = w_{a,b} \times \frac{N_{common}}{50} \quad (2)$$

Where N_{common} is the number of co-rated items. If two users have more than 50 co-rated items, then we leave the correlation unchanged.

In step 3, predictions are computed as the weighted average of deviation from the neighbor's mean:

$$p_{a,j} = \bar{v}_a + \frac{\sum_{b=1}^{N}(v_{b,j} - \bar{v}_b) \times w'_{a,b}}{\sum_{b=1}^{N} w'_{a,b}} \quad (3)$$

Where N is the number of users, $p_{a,j}$ is the prediction of the active user a on the target item j.

Memory-based methods have the advantages of being able to rapidly incorporate the most up-to-date information and relatively accurate prediction, but they suffer from poor scalability for large numbers of users because the search for all similar users will be much slower when the database becomes large. If the size of the training set is N, then the time complexity of the memory-based method is O(NM).

2.2 Model-based Algorithm

The model-based collaborative filtering algorithm uses the users' preferences to learn a model, which is then used for predications. The model can be built off-line over several hours or days. The resulting model is very small, very fast, and essentially as accurate as memory-based methods [12]. Model-based methods may prove practical for environments in which user preferences change slowly with respect to the time needed to build the model. Model-based methods, however, are not suitable for environments in which user preference models must be updated rapidly or frequently.

In this paper, we will focus on memory-based algorithms and present some new methods to improve the accuracy and the scalability.

3. SIMILARITY MEASURE

There exists relationship between items. Yu et al [11] computes dependencies between items using mutual information method and presents a feature weighting method to compute the similarity between users. And Sarwar et al [13] computes similarities between items and uses them straightforward for prediction.

We observe that items can be classified into different types. There exists strong relevancy between these items belonging to the same class. So we first classify all items. For instance, for those text-formatting items, we can use the Naive Bayes method to carry out classification [18]. And then we convert the user-item matrix into a user-class matrix. The value of an element of the user-class matrix is the total rating of items belonging to the same class:

$$v_{a,k} = \sum_{\text{item } j \text{ belonging to class } c_k} v_{a,j} \quad (4)$$

Where $c_k \in C=\{c_1, \cdots, c_K\}$, $v_{a,k}$ is the rating of user a on class c_k.

Figure 1 illustrates the process of matrix conversion from user-item matrix to user-class matrix. Because the number of classes is far less than items, the density of data of matrix increases greatly. Moreover, the similarity measure between users will be more accurate for users are more correlated among classes.

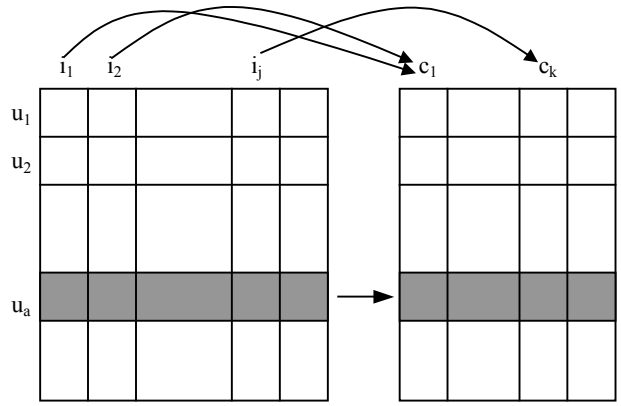

Figure 1. Matrix conversion from user-item matrix to user-class matrix.

The similarity between users based on the user-class matrix can still be measured by computing Pearson correlation. We observe that the target item is relevant to some classes. So we adopt a weighting method:

$$w_{a,b} = \frac{\sum_{k=1}^{K} w_k^2 (v_{a,k} - \bar{v}_a)(v_{b,k} - \bar{v}_b)}{\sqrt{\sum_{k=1}^{K} w_k^2 (v_{a,k} - \bar{v}_a)^2 \sum_{k=1}^{K} w_k^2 (v_{b,k} - \bar{v}_b)^2}} \quad (5)$$

Where K is the number of classes, w_k represents $P(c_k|i_j)$, which is the probability of the target item j belonging to class c_k. $\bar{v}_a$ is the mean rating given by user a on all classes.

4. INSTANCE SELECTION

Under some circumstances, the user-class matrix is not enough to improve the accuracy of the collaborative filtering algorithm for full of irrelevant information. It is an interesting problem to remove irrelevant information from user-class matrix.

On the other hand, how to improve the scalability of the collaborative filtering algorithm? To respond to these challenges, we present an instance selection method. Different from other random sampling or data focusing techniques [19], the method can not only reduce the size of training set but also improve the accuracy of prediction.

4.1 Relevancy between Users and Items

Hofmann [20] proposes an aspect model, a latent class statistical mixture model, for associating word-document co-occurrence data with a set of latent variables. Hofmann et al [21] applies the aspect model to user-item co-occurrence data for collaborative filtering. In the aspect model, users $u \in U=\{u_1, ..., u_N\}$, together with the items they rate $i \in I=\{i_1, ..., i_M\}$, form observation (u, i), which are associated with one of the latent class variable $c \in C=\{c_1, ..., c_K\}$. The key assumption made is that u and i are independent, conditioned on c. The joint probability distribution over users, classes and items can be written as:

$$P(u,i) = \sum_{c \in C} P(u \mid c) P(i \mid c) P(c) \quad (6)$$

Where $P(u|c)$ and $P(i|c)$ are class-conditional multi-nominal distributions and $P(c)$ are the class prior probabilities.

By using the identity $P(c)P(i|c) = P(i, c) = P(i)P(c|i)$ and the identity $P(u, i) = P(u)P(i|u)$, the model yields:

$$P(i \mid u) = \frac{P(i)}{P(u)} \sum_{c \in C} P(u \mid c) P(c \mid i) \quad (7)$$

And then according to the identity $P(u|c) / P(u) = P(c|u) / P(c)$, the equation (7) can be written as:

$$P(i \mid u) = P(i) \sum_{c \in C} \frac{P(c \mid u) P(c \mid i)}{P(c)} \quad (8)$$

The equation (8) depicts the relevancy of user u to item i based on a latent class model.

4.2 Instance Selection Method

An important step of the user-based collaborative filtering algorithm is to search neighbors for the active user. The traditional method is to search the whole database. Apparently this method suffers from poor scalability when more and more users and items are added in the database. To handle this problem, indexing techniques might be considered. However recent results [22] show that a simple sequential scan of the whole data outperforms any of the indexing methods when the dimensionality exceeds around 10. Usually, the dimension of the data of the collaborative filtering algorithm is more than 1000, so the method of indexing is impractical. The practical method is instance selection that selects appropriate instances from the whole database for the filtering algorithm.

According to the above analysis, we use the equation (8) to rank the relevancy of users to a target item. A noticeable point is that $P(i)$ is insignificant for the ranking of the relevancy between users and the target item.

A problem comes to mind, how to compute the $P(c|u)$, $P(c|i)$ and $P(c)$. First let's show how to compute $P(c|i)$. For those text-formatting items, we can use the Naive Bayes method to carry out computation. Hofmann et al [21] presents an Expectation Maximization (EM) algorithm for maximum likelihood estimation of these parameters. However, EM algorithm is not suitable here because the knowledge of classification of items obtained from the user-item matrix is different from the content-based classification of items. Moreover, we must explicitly obtain the knowledge of classification of items to create user-class matrix.

We utilize the rating of user u on class c divided by the total rating of the user on all classes to represent $P(c|u)$, which can be written as:

$$P(c \mid u) \approx \frac{v_{a,c}}{\sum_{c_k \in C} v_{a,k}} \quad (9)$$

Where $v_{a,c}$ is the rating of user a on class c.

$P(c)$ is represented as the number of items belonging to class c divided by the total number of items belonging to all classes:

$$P(c) \approx \frac{N_c}{\sum_{c_k \in C} N_k} \quad (10)$$

Where N_k is the number of items belonging to class c_k.

Because the instance selection method can remove irrelevant instance and reduce the size of training set, the accuracy and the scalability of the collaborative filtering algorithm are improved.

5. AN IMPROVED COLLABORATIVE FILTERING ALGORITHM

On the basis of these two methods mentioned above, we introduce an improved collaborative filtering algorithm, called class-based algorithm, as shown in Figure 2. To improve the accuracy and the performance further of the algorithm, we introduce a data preprocessing method. We observe that not all of the items are significant for computation. So it is reasonable removing some items using feature selection methods. Yang et al [23] carries out a comparative study between several feature selection methods for text categorization. Its results show that the document frequency threshold is an effective way to select features. So we adopt an analogous method to select items, which is termed the user frequency threshold. The idea is that rare rated items are either non-informative, or not influential in global performance. So we remove those items whose user frequency is less than a threshold. Improvement of accuracy is also possible if rare rated items happen to be noise. Despite that this algorithm can improve performance, its time complexity is still equal to the traditional memory-based algorithm.

Algorithm: Class-based collaborative filtering

Input: training set, test set, a target item, a given user frequency threshold and a given selection rate k

Output: mean absolute error of prediction

Method:

1. Preprocess training data set and remove rare rated items whose user frequency is less than the given threshold.
2. Convert the user-item matrix into the user-class matrix and select top k% instances relevant to the target item.
3. For each instance in the test set, search neighbors from the training set and then compute a prediction for the target item.
4. Compute the mean absolute error of prediction and output.

Figure 2. An improved collaborative filtering algorithm.

6. EXPERIMENTAL EVALUATION

In this section, we describe the dataset, metrics and methodology for the comparison between different prediction algorithms, and present the results of our experiments.

6.1 Data Set

We use EachMovie data set [24] to evaluate the performance of improved algorithm. The EachMovie data set is provided by the Compaq System Research Center, which ran the EachMovie recommendation service for 18 months to experiment with a collaborative filtering algorithm. The information they gathered during that period consists of 72,916 users, 1,628 movies, and 2,811,983 numeric ratings ranging from 0 to 5. To speed up our experiments, we only use a subset of the EachMovie data set. For every experiment, we randomly select 5000 users who had rated 30 or more items and then divide them into a training set (90% users) and a test set (10% users). We have about 380,000 ratings for all movies, and the data density is 4.8%. All movies have already been classified manually into 10 types by the EachMovie recommendation service: action, animation, art or foreign, classic, comedy, drama, family, horror, romance, and thriller. Moreover, a movie might belong to more than one class.

6.2 Metrics and Methodology

The metrics for evaluating the accuracy of a prediction algorithm can be divided into two main categories: statistical accuracy metrics and decision-support metrics. Statistical accuracy metrics evaluate the accuracy of a predictor by comparing predicted values with user-provided values. Decision-support accuracy measures how well predictions help users select high-quality items. We use Mean Absolute Error (MAE), a statistical accuracy metrics, to report prediction experiments for it is most commonly used and easy to understand:

$$MAE = \frac{\sum_{a \in T} | v_{a,j} - p_{a,j} |}{| T |} \qquad (11)$$

Where $v_{a,j}$ is the rating given to item j by user a, $p_{a,j}$ is the predicted value of user a on item j, T is the test set, $|T|$ is the size of the test set.

As in [12], we also employ two protocols, All but 1, and Given K. In the first class, we randomly withhold a single randomly selected vote for each test user, and try to predict its value given all the other votes the user has voted on. The All but 1 experiments measure the algorithms' performance when given as much data as possible from each test user. The second protocol, Given K, randomly select K votes from each test user as the observed votes, and then attempts to predict the remaining votes. It looks at users with less data available, and examines the performance of the algorithms when there is relatively little known about an active user.

6.3 Experimental Results

We design several experiments for evaluating our algorithm and analyze the effect of various factors by comparison.

6.3.1 Effect of Data Preprocessing

We compared the accuracy of prediction before and after data preprocessing. The user frequency thresholds are 0, 30, 50, 70, and 90. The data density of training set increases as the threshold increases. The initial data density is 4.8% when the threshold is 0 and the size of training set is 5000. When the threshold is 30, 50, 70, and 90, the data density becomes 8.2%, 9.2%, 10.3%, and 11.1% respectively. Figure 3 shows the experimental results. It can be observed from the results that the data preprocessing method has a clear advantage, as the MAE of those thresholds 50, 70 and 90 are lower than that of the threshold 0, which represents the traditional algorithm. We also observe that the collaborative filtering algorithm can't carry out prediction when the size of training set is 1000 and the threshold is 70 and 90. Because the

accuracy of the threshold 50 is better than others on average, we select this threshold for the rest of our experiments.

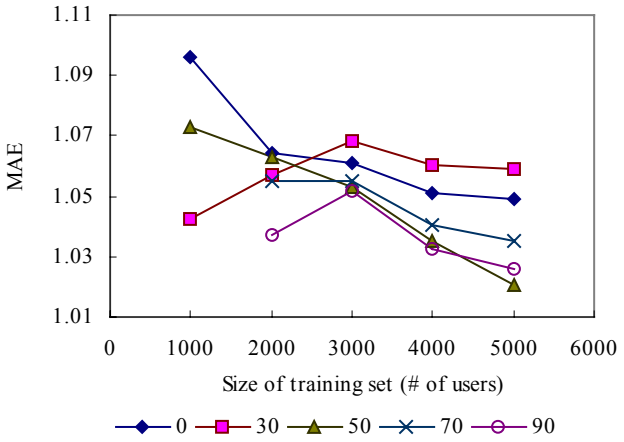

Figure 3. All but 1 results of data preprocessing.

6.3.2 Impact of Matrix Conversion

To determine the impact of matrix conversion, we carried out an experiment to compare the accuracy of prediction before and after matrix conversion. The experiments were conducted for training set with 1000, 2000, 3000, 4000 and finally 5000 users. The data density of the training set increases after matrix conversion. For instance, when the size of training set is 5000, the initial data density is 4.8% before data preprocessing. After data preprocessing and matrix conversion, the data density becomes 91.8%. Figure 4 shows the experimental results, we observe that the proposed method outperforms the traditional algorithm. However, we also observe that the data preprocessing has negative influence on the prediction of algorithm when the size of training set is less than 3000. The main reason is that the user frequency threshold is always 50, which is not the most appropriate threshold for prediction when the size of training set is less than 3000.

In addition, we also carried out an experiment under the protocols of Given 5, 10, 15, 20, 25, 30 when the size of training set is 5000 as shown in Figure 5. It can be observed that our method has a clear advantage. The results indicate our method can keep the quality of prediction even if the user only votes few items during the startup period.

6.3.3 Sensitivity of Instance Selection

Those irrelevant instances have a significant impact on the prediction quality [11]. We evaluated the quality of our method of selecting relevant instances. The outcomes are given in Figure 6. We sort users in descending order of their relevancy to each movie, and select highly relevant users for the prediction in different selection rates of 2.5%, 5%, 10%, 20%, 40%, 60%, 80% and 100%. The results are compared with the outcomes of random sampling. The proposed method outperforms random sampling in accuracy, and the combination with other methods results in further 6-8% improvement of accuracy. We also observe that the accuracy of instance selection without using other two techniques outperforms that of instance selection using other two techniques when the selection rate is less than 30%. However, these techniques are effective even if the selection rate is equal to 5%.

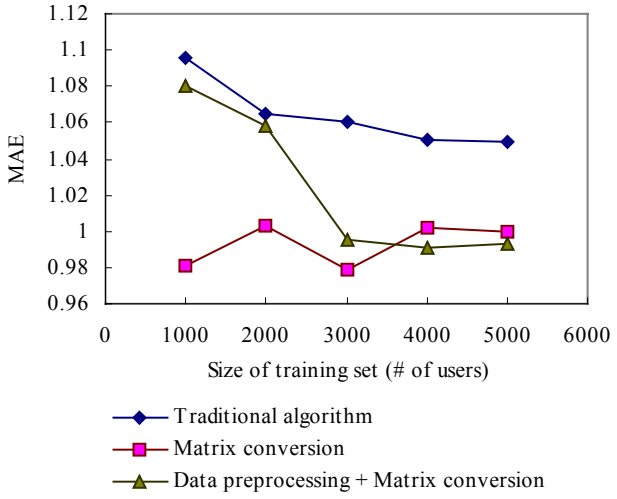

Figure 4. All but 1 results of matrix conversion.

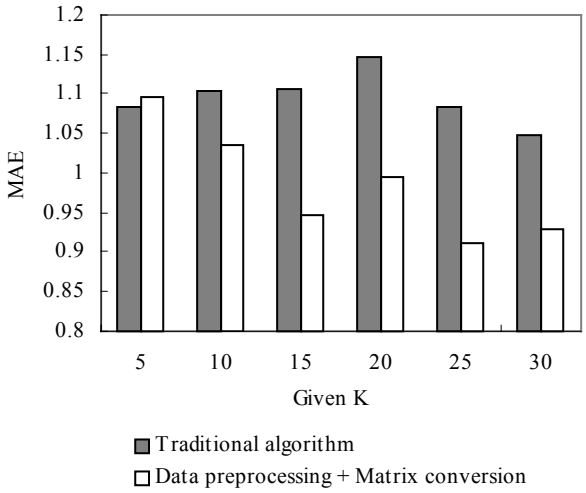

Figure 5. Given K results of matrix conversion.

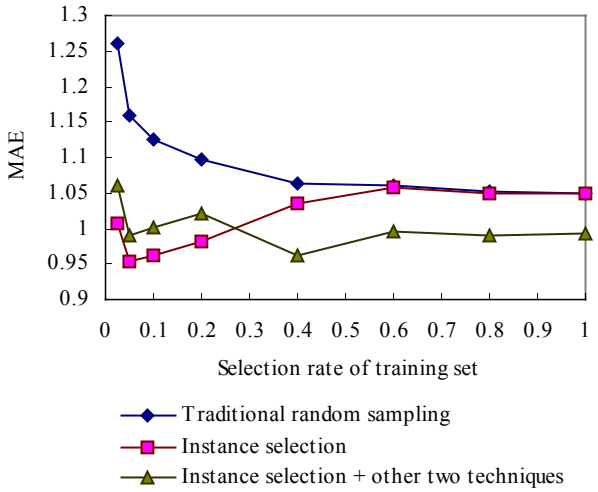

Figure 6. All but 1 results of different selection rates.

6.3.4 *Comparison Experiments*

We implemented five different collaborative filtering algorithms: user-based algorithm, item-based algorithm [13], cluster-based algorithm [12], feature-weighting algorithm [11], and our algorithm that is class-based algorithm. The results of comparison of accuracy between these five algorithms are shown in Figure 7. Our algorithm shows its satisfactory results. The selection rate of our algorithm is 5%. If we take the user-based algorithm as the baseline, then we can see that the clustering-based algorithm is the worst, and the class-based algorithm is better than both the feature-weighting algorithm and the item-based algorithm.

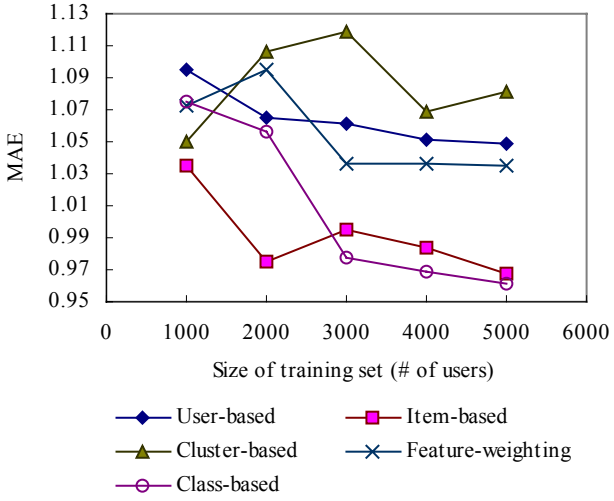

Figure 7. All but 1 comparison of accuracy between five algorithms.

6.3.5 *Performance Results*

We also carried out an experiment to compare the performance of these five algorithms. Figure 8 shows the results of comparison of training time between these five algorithms. In this figure, we observe that the training time of the cluster-based algorithm is more than that of other four algorithms on average. The training time of the feature-weighting algorithm is also large because the algorithm spends a long time on computing the mutual information between the target item and other items. The training time of the class-based algorithm is small because the time of data preprocessing and matrix conversion is small. The training time of the user-based algorithm and that of the item-based algorithm are small because these two algorithms only load data into memory without any processing.

Figure 9 shows the results of comparison of predicting time between these five algorithms. We observe that the predicting time of the user-based algorithm is more than that of other four algorithms on average and is subject to the number of users. The predicting time of the cluster-based algorithm is little because the algorithm finds neighbors from clusters. The predicting time of the item-based algorithm is small and doesn't change significantly because the algorithm is based on items and isn't subject to the number of users. The predicting time of the feature-weighting algorithm and that of the class-based algorithm are small because these two algorithms select appropriate user instances from the training set as a new smaller set.

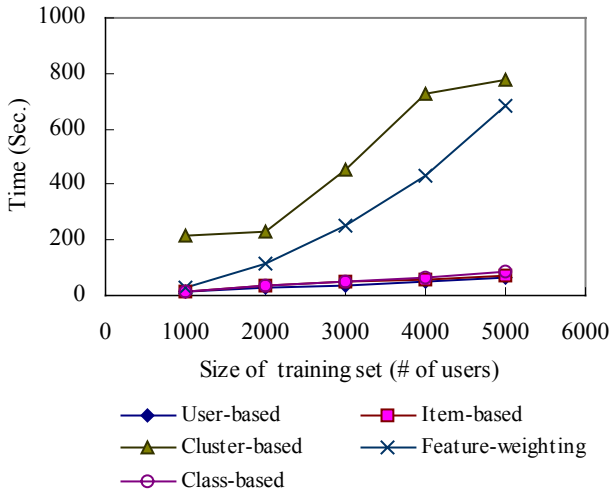

Figure 8. All but 1 comparison of training time between five algorithms.

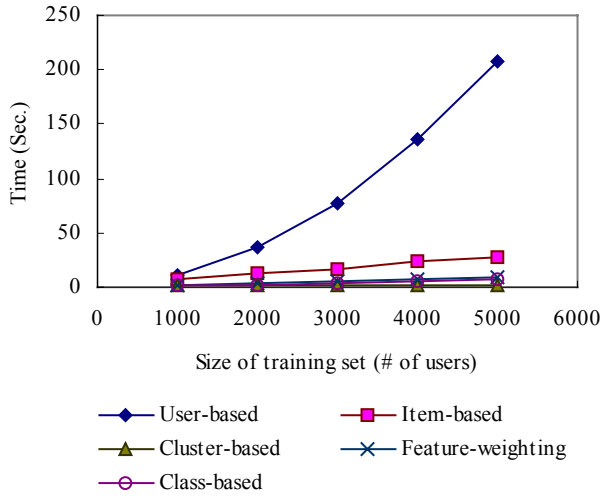

Figure 9. All but 1 comparison of predicting time between five algorithms.

7. CONCLUSION

Collaborative filtering has been very successful in both research and applications such as information filtering and E-commerce. Many recommender systems are developed based on the collaborative filtering technology. These systems help users find items they want to buy from a business. On the other hand, they also help business sell more items. New technologies are needed that can dramatically improve the performance of recommender systems.

In this paper, we focus on the study of memory-based collaborative filtering. Memory-based methods have the advantages of being able to rapidly incorporate the most up-to-date information and relatively accurate prediction, but they suffer from two problems. This paper discusses these two problems and provides our solutions. The experimental results show that the accuracy and the performance of our methods are quite satisfactory. In the future, we will integrate the contents of items

into the collaborative filtering algorithm to improve the accuracy further.

8. REFERENCES

[1] Mobasher, B., Colley, R., and Srivastava, J. Automatic personalization based on Web usage mining. Communications of the ACM, Aug. 2000, 43(8): 142-152.

[2] Lieberman, H. Letizia: an agent that assists web browsing. In Proceedings International Conference on AI, Montreal, Canada, Aug. 1995. 924-929.

[3] Mladenic, D. Personal WebWatcher: design and implementation. Technical Report IJS-DP-7472, J. Stefan Institute, Department for Intelligent Systems, Ljubljana, 1998.

[4] Pazzani, M., Muramatsu, J., and Billsus, D. Syskill & Webert: identifying interesting web sites. In Proceedings of the 13th National Conference on Artificial Intelligence, 1996. 54-61.

[5] Konstan, J., Miller, B., Maltz, D., et al. GroupLens: Applying collaborative filtering to Usenet news. Communications of the ACM, Mar. 1997, 40(3): 77-87.

[6] Joachims, T., Freitag, D., and Mitchell, T. WebWatcher: A Tour Guide for the World Wide Web. In Proceedings of the International Joint Conference on Artificial Intelligence, Aug. 1997. 770-777.

[7] Lieberman, H., Dyke, N. van, and Vivacqua, A. Let's browse: a collaborative Web browsing agent, In Proceedings of the International Conference on Intelligent User Interfaces, Jan. 1999. 65-68.

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

[9] Claypool, M., Gokhale, A., Miranda, T., etc. Combining Content-Based and Collaborative Filters in an Online Newspaper. In ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, Aug. 1999.

[10] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. Application of Dimensionality Reduction in Recommender System - A Case Study. In ACM WebKDD Workshop, 2000.

[11] Yu, K., Wen, Z., Xu, X., Ester, M. Feature Weighting and Instance Selection for Collaborative Filtering. In Proceedings of the 2nd International Workshop on Management of Information on the Web - Web Data and Text Mining, 2001.

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

[13] Sarwar, B., Karypis, G., Konstan, J., et al. Item based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference, 2001. 285-295.

[14] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. Analysis of Recommendation Algorithms for E-Commerce. In Proceedings of the ACM EC'00 Conference, Minneapolis, MN. 2000. 158-167.

[15] Yu, K., Xu, X., Tao, J., et al. Instance Selection Techniques for Memory-Based Collaborative Filtering. In Proceedings of the 2nd SIAM International Conference on Data Mining, 2002.

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

[17] Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. An Algorithmic Framework for Performing Collaborative Filtering. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Aug. 1999. 230-237.

[18] Joachims, T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In Proceedings of the 14th International Conference on Machine Learning. San Francisco: Morgan Kaufmann Publishers, 1997. 143-151.

[19] Ester, M., Kriegel, H.P., and Xu, X. Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification, In Proceedings of the 4th International Symposium On Large Spatial Databases, Portland, ME, 1995, 67-82.

[20] Hofmann, T. Probabilistic latent semantic analysis. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, 1999, 289-296.

[21] Hofmann, T., and Puzicha, J. Latent class models for collaborative filtering. In Proceedings of the 16th International Joint Conference on Artificial Intelligence, 1999, 688-693.

[22] Weber, R., Schek, H. J., Blott, S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Proceedings of the International Conference on Very Large Databases, 1998. 194-205.

[23] Yang, Y. and Pedersen, J. O. A comparative study on feature selection in text categorization. In Proceedings of the 14th International Conference on Machine Learning, 1997, 412-420.

[24] EachMovie data http://research.compaq.com/SRC/eachmovie.