Instance Selection Techniques for Memory-Based Collaborative Filtering *

Kai Yu[1, 2], Xiaowei Xu[1], Jianhua Tao[3], Martin Ester[2] and Hans-Peter Kriegel[2]

Abstract: Collaborative filtering (CF) has become an important data mining technique to make personalized recommendations for books, web pages or movies, etc. One popular algorithm is the memory-based collaborative filtering, which predicts a user's preference based on his or her similarity to other users (instances) in the database. However, the tremendous growth of users and the large number of products, memory-based CF algorithms results in the problem of deciding the right instances to use during prediction, in order to reduce executive cost and excessive storage, and possibly to improve the generalization accuracy by avoiding noise and overfitting. In this paper, we focus our work on a typical user preference database that contains many missing values, and propose four novel instance reduction techniques called TURF1-TURF4 as a preprocessing step to improve the efficiency and accuracy of the memory-based CF algorithm. The key idea is to generate prediction from a carefully selected set of relevant instances. We evaluate the techniques on the well-known EachMovie data set. Our experiments showed that the proposed algorithms not just dramatically speed up the prediction, but also improved the accuracy.

1 Introduction

The tremendous growth of information gathered in E-commerce has motivated the use of information filtering and personalization technology. One main problem that the users face is how to find the product they like from millions of products. For the vendor, again, it is crucial to find out the user preferences. Collaborative filtering (CF) based recommender systems have emerged in response to these problems [Breese et al., 1998; Resnick et al., 1994; Shardanand and Maes, 1995]. CF-based recommender system accumulates a database of consumers' product (or item) preferences, and then uses them

* The work was performed in Cooperate Technology, Siemens AG. The contact author is Xiaowei Xu.

[1] Cooperate Technology, Siemens AG, Munich, Germany, {Xiaowei.Xu|Kai.Yu.external}@mchp.siemens.de.
[2] University of Munich, {Yu_k|Ester|Kriegel}@dbs.informatik.uni-muenchen.de.
[3] Tsinghua University, Beijing, China, , Tao@media.cs.tsinghua.edu.cn.

to predict the preference of a particular user (the active user) for the target items such as music CDs, books, web pages, or movies. The intuition behind the algorithm is that the active user will prefer those items that the like-minded people prefer. So far, two general classes of CF algorithms have been widely investigated [Breese et al., 1998]. *Memory-based* CF, which is the most prevalent approach, operates over the entire user preference database to make predictions. In contrast *model-based* algorithms use the preference database to infer a model, which is then applied for predictions.

CF algorithms have been very successful in both research and practice. However, there still remain important research questions in overcoming two fundamental challenges for CF [Sarwar et al., 2000]. The first challenge is to improve the scalability and efficiency of CF algorithms. Existing CF algorithms can deal with thousands of consumers within a reasonable time, but the demand of modern E-Commerce systems is required to scale millions of users. Efficiency is another intimately related issue. Prediction time of a request must be less than 1 second and prediction engines must often support throughput of several hundred requests per second [Herlocker et al., 1999]. The second challenge is to improve the quality of the recommendations for the users. Users need recommendations they can trust to help them find products they will like. If a user trusts a recommender system, purchases a product, but finds out he does not like the product, the user will be unlikely to use the recommender systems again. Much work has been conducted on this issue.

In this paper, we focus our work on memory-based CF algorithms and address the problem of deciding which instances to use during prediction, in order to reduce time complexity, and to improve the accuracy by avoiding noise and overfitting. Two cases of instance removal are considered, the first is to remove redundant instances whose preference pattern has been already carried by other instances, and the second is to remove irrelevant instances whose preference profile is hard to generalize for prediction. Four novel instance selection techniques called TURF1–TURF4 are proposed as a preprocess step for memory-based CF. The key idea is to speed up predictions by generating predictions over relevant and informative instances, instead of operating over the entire database. Our first algorithm works in an incremental manner, which starts with a few training instances, and adds those instances with novel profile into training set. The second algorithm works in a filtering manner, those instances with a strongly rational profile are included into the training set. In the third algorithm, we combine the two former algorithms to pick up those instances with a novel and rational profile. In the fourth algorithm, we try to explore the potential of storage reduction. Our experiment on a real-world preference database confirms the efficiency and accuracy of our approaches.
The rest of this paper is organized as follows. Section 2 introduces related work, covering CF algorithms and instance selection methods in memory-based learning. In section 3, we review the scenario of memory-based CF algorithms over preference database and then we describe the four proposed instance selection algorithms one by one. Section 4 gives empirical results that indicate how well our proposed algorithm works in practice. Finally in section 5, the paper is finished with a conclusion and some interesting future work.

2 Related Work

The task in CF is to predict the preference of a particular user (active user) based on a database of users' preferences. There are two general classes of CF algorithms: memory-based methods and model-based methods [Breese et al., 1998].

Memory-based algorithm [Breese et al., 1998, Resnick et al., 1994, Shardanand and

Maes, 1995] is the most popular prediction technique in CF applications. The basic idea is to compute the active user's vote on a target item as a weighted average of the votes given to that item by other like-minded users. The Pearson correlation, which was first introduced in [Resnick et al., 1994], has been widely and successfully used as a similarity measure between users. Memory-based methods have the advantages of being able to rapidly incorporate the most up-to-date information and relative accurate prediction [Breese et al., 1998], but they suffer from the poor scalability for large number of users.

Model-based CF, in contrast, uses the user's preference database to learn a model, which is then used for predictions. The model can be built off-line over a matter of hours or days. The resulting model is very small, very fast, and essentially as accurate as memory-based methods. The reported model-based CF algorithms include Bayesian networks [Breese et al., 1998], clustering techniques [Breese et al., 1998; Fisher et al.,2000] singular value decomposition with neural network classification [Billsus and Pazzani, 1998], induction rule learning [Basu et al, 1998], and Personality Diagnosis [Pennock et al, 2000]. Model-based methods may prove impractical for environments in which knowledge of consumer preference changes, since the training time is very long.

Approaches to improve the accuracy of CF have been widely investigated in literatures. However, relatively fewer work addresses the issue of efficiency so far. Breese et al compared the running time and learning time performance between memory-based algorithm and model-based algorithm (Bayesian network) when the training movie data contains 5000 users [Breese et al., 1998]. They found although model-based methods were approximately 4 times as fast as the memory-based methods, the learning time for Bayesian network was up to 8 hours. In [Billsus and Pazzani, 1998], similarly, the algorithm need to compute the SVD (singular value decomposition) and learning models for each item, the cost of training is expensive too. In contrast, memory-based methods have no training time, But it suffers from the neighbor search during prediction. Suppose given millions of users, performing CF can be exhausting. There are two possible ways to speed up the neighborhood search in memory-based CF. One is to build a data structure. However efficient similarity search through index structure in a high-dimension space still remains a challenge. In [Han et al., 2001], a tree-like structure named RecTree was introduced to improve the efficiency of memory-based CF. A hierarchical clustering algorithm was performed to construct the tree. As a result, the search for neighborhood will be faster than scanning the entire database. Obviously, RecTree didn't avoid the problem of noise, data redundancy and overfitting. The other way is to reduce the training data. In [Sarwar et al., 2000], SVD(singular value decomposition) was applied to reduce the dimensionality. But that work focused on the accuracy performance rather than efficiency. So far, little work was reported on reducing the number of training users for collaborative filtering. We believe data reduction can be combined with other methods in order to obtain a maximum performance of efficiency and accuracy.

In this paper, our study on CF focuses on selecting training users (instances), which is also a topic related to instance-based learning algorithms. It is necessary to decide what instances to store for generalization in order to reduce excessive storage and time complexity, and possibly to improve accuracy. Therefore instance selection has become an important topic in instance-based learning [Aha et al., 1991; Wilson and Martinez, 2000; Pradhan and Wu, 1999; Wilson and Martinez, 2000]. Some algorithms seek to select representative instances, which could be border points [Aha et al., 1991] or central points [Zhang, 1992]. For almost all the instance selection algorithms mentioned above, the classifier was applied at least once in each step of removing or adding an instance, so the computational complexity is somewhat expensive. Our work is unique because we

consider the nature of user preference database in which different instances have different attributes and any attribute can be a possible target to be predicted (The details will be given in the next section.). Specifically, we studied two cases: (1) There are lots of irrelevant instances whose profile is not clearly described by their attributes, thus these instances are hard to be generalized and even lead to overfitting; (2) The other case arises when some preference patterns have been redundantly carried by many instances, the redundant instances may delay the prediction and thus should also be reduced. In the following sections, we will study the techniques of instance selection for CF in details.

3 Instance Selection for Memory-Based Collaborative Filtering

Before our introduction to the proposed work, for a better understanding, it is necessary to review the scenario of memory-based CF algorithm (in the rest of this paper, if not specified, CF indicates the memory-based CF). And we also briefly give our solution to improve the memory-based CF algorithms in the first subsection. Then we give the details of the proposed instance selection techniques one by one in following subsections.

	Superman	Titanic	Dances with Wolves	Batman
Jason	5			5
Karen			3	4
Fred	2	5		2
Sophia	2	5	4	2
Julia	?	4	?	2
Tom	4	3	4	?

If we want to predict Julia's vote on *Dances with Wolves*, then corresponding instances include Karen and Sophia.

Figure 1. A list of people and their votes on a list of movies (The vote is on a discrete scale of 0~5, a higher score indicating a higher preference.)

3.1 Memory-Based Collaboration Algorithm

As shown in Figure 1, the historical preference data is a user-item matrix, with each entry $v_{u,i}$ indicating the vote of user u for item i. An important characteristic is that many empty entries exist in the matrix, since users only rated a small portion of items. In memory-based algorithm [Breese et al., 1998, Resnick et al., 1994, Shardanand and Maes, 1995] , the prediction $P_{a,i}$ of active user a on target item i is given by:

$$P_{a,i} = \bar{v}_a + k \sum_{b \in \text{neighborhood}(a,\, T_i)} r(a,b)(v_{b,i} - \bar{v}_b) \qquad (3.1.1)$$

where T_i , the instance set for item i, is the set of all the users who have rated item i, and *neighborhood*(a, T_i) represents neighbors of active user a in T_i , here neighbors can be defined as all the users in T_i [Breese et al., 1998], or results of KNN(k-nearest neighbor) query [Herlocker et al., 1999] or range query[Shardanand and Maes, 1995]. $\bar{v}_u$ is the mean vote for user u, and k is a normalizing factor such that the absolute values of the weights sum to unity. The Pearson correlation coefficient [Resnick et al., 1994] has been

widely and successfully used as a similarity measure between users *a* and user *u* :

$$r(a,b)=\frac{\sum_{j\in \text{overlap}(a,b)}\left(v_{a,j}-\overline{v}_a\right)\left(v_{b,j}-\overline{v}_b\right)}{\sqrt{\sum_{j\in \text{overlap}(a,b)}\left(v_{a,j}-\overline{v}_a\right)^2\sum_{j\in \text{overlap}(a,b)}\left(v_{b,j}-\overline{v}_b\right)^2}} \tag{3.1.2}$$

where *overlap*(*a*, *b*) indicates that the similarity between two users is computed over the common voted items. [Shardanand and Maes, 1995] claimed better performance by computing similarity using a *constrained* Pearson correlation coefficient, where a user's mean vote was replaced by a constant, the midpoint of the rating scale. In our experiment, where discrete scale is 0-5, we choose the midpoint to be 3 and form the neighborhood by range query, where users whose similarity to active user is higher than 0 are selected.

As indicated by eq. (3.1.1), a single prediction is generated by the weighted sum of votes on the target item *i* over active user's neighborhood in the instance space of T_i. Since it is almost impractical to build data structure in high dimensional space, the search of neighborhood is very computationally expensive when the training set T_i is very large. An obvious critique to memory-based learning methods is their high sensitivity to noises and redundancy. In this section we will present four techniques of Training User Reduction for Collaborative Filtering, TURF1-TURF4, to remove noise and redundancy for reducing the computational complexity while trying to maintain or even improve the accuracy. The basic approach is to get reduced training sets T'_i for every target item *i*, where $T_i' \subseteq T_i$, and then perfom collaborative filtering Eq. (3.1.1) over T'_i instead of T_t.

3.2 TURF1 – Selecting Users with Novel Profile

The first selection technique TURF1, whose idea is borrowed from Condensed NN rule [Hart, 1968] and IB2 [Aha et al., 1991], addresses the redundancy of training set. Consider the Example 1 again, Fred and Sophia have shown very close preference profiles for the target item *Batman*, if we remove one of them, the lost of preference patterns in training set may be not considerable in a sense, while the execution can be accelerated by a factor of 4/3. For a target item *i*, TURF1 starts from a small randomly selected training set T'_i from T_i, and incrementally processes other training users in T_i, following the simple rule: User *u* is judged to be with novel profile and added into the training set T'_i, if user *u*'s vote on target item *i* is incorrectly predicted by CF over current instance set T'_i. The algorithm proceeds as the following pseudo code:

```
TURF1( Training preference database T )
    For each target item i in T
        If ( number of users in T_i who voted on i > INITIAL_SIZE )
            Randomly seed the initial training set T'_i with INITIAL_SIZE users in T_i
            For each remaining user u in T_i
                If u's vote on i isn't correctly predicted by CF using T'_i
                    Add u into T'_i
                Endif
            Endfor
        Endif
    Endfor
End
```

After instance selection is finished, every target item *i* will has a smaller training set T'_i,

and the CF prediction performed over the reduced training set will be more efficient. The algorithm TURF1 is quite similar to Condensed NN rule [Hart, 1968] and IB2 algorithm [Aha et al., 1991], except TURF1 seeds T'_i with a small number of users by random selection and is applied for numeric prediction, instead of classification. In our study, we will consider explicit user votes on a discrete scale from 0 to 5. If the prediction error is less than 0.5, which indicates the right rounded value, we judge the vote to be correctly predicted. And the initial size of T'_i is set to be 150.

[Aha et al., 1991] indicated the IB2 algorithm retained border points in training instance space while eliminating internal points that are surrounded by members of the same class. In addition, we have other reasons to perform TURF1 for CF regression:

1. Put more strength on sharply changing regions. Consider instances (or users) are distributed in an attribute space, with the vote on the target item serving as the instance's value and its votes on other items serving as the instance's attributes. In some regions where instance's value is sharply changing, since the values of instances in the neighborhood don't consist with each other, the prediction error is always greater than that in other smooth regions. TURF1 follows this nature and retains more training users in this kind of regions and removes more training users in smooth regions, while maintaining the same prediction accuracy.
2. Avoid the prediction bias caused by dense regions. Redundancy of training set not only slow down the prediction, but also caused potential bias of prediction. As shown in Figure 2, in a neighborhood from a range query, there is a dense region containing many similar training users, we call them redundant users. Although the distance from the dense region to query user is large and hence each single redundant user contributes a little to the prediction, however, the combination of their effects might be greater than those closer users and biases the predicted value. Obviously, TURF1 can reduce the redundancy in dense regions by removing correctly predicted users. From this point of view, TURF1 may improve the accuracy in some cases. In fact, our experiments confirmed this. While in [Aha et al. 1991], IB2 sacrificed classification accuracy. An explanation is that, IB2 applied 1-NN algorithm and thus involves no dense region in neighborhood, while in CF algorithm the size of neighborhood is always much larger (tens or hundreds), since it is meaningless to find several nearest neighbors in a space with dimensionality over 1000 [Hinneburg et al. 2000].

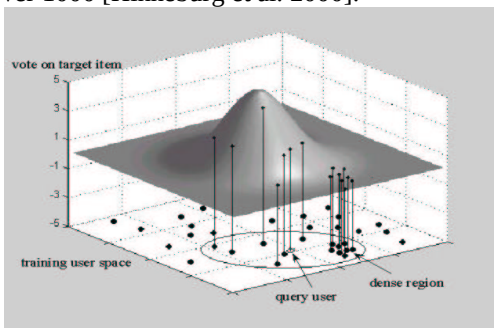

Figure 2. Illustration of a CF prediction biased by a dense region

3. Incrementally accommodate new types of preference patterns. With the growth of business, recommender system must be able to handle new preference patterns. The users with novel pattern may be distributed in a remote and empty region in the instance space.

If a new profile can't be predicted by current training set by using CF, it is indicated the system encounters a new profile that should be added into the training set.

Our experiment showed TURF1 can reduced the size of training set for each target item in training phase, and then speeded up the CF and improved the accuracy. However, this method remains some problem. As IB2 [Aha et al. 1991], TURF1 can be also prone to accept exceptional training users, whose vote to the target item can not be reasonably explained by his/her vote to other items and thus act like noise. Those irrelevant instances may misleading the CF and cause a high computational cost. In next parts of this section, we will discuss this point in details. The other problem is the cost of training. Since each judgement on a training user in T_i needs a run of CF prediction over current T'_i, it can be very expensive when the database is very large. If the average number of users who voted on each item is n, each user have voted on average m items, and k is the average resulting selection rate of training size, the expected complexity for TURF1 is $O(kn^2m/2)$.

3.3 TURF2 – Selecting Users with Rational Profile

In this section, we continue to propose the second algorithm TURF2, which removes the irrelevant instances. Our scope is focused on the question: *Given a training user, has his or her profile been well described by his or her preference data in the database?* For convenience, assuming to predict user votes for item i, we will look a training user as an instance u with its value $v_{u,i}$—u's vote on item i. People's votes on item i are denoted by V_i , with value v_i . The set of u's other voted items is the u's descriptive item set, noted by $F(u, i)$. Then people's votes on the descriptive item set are $V_{F(u,i)}$ with value $v_{F(u,i)}$.

Definition 3.3.1 Given an instance u in T_i, if entropy $H(V_i)$ is a prior uncertainty of people's votes for the target item i, then the ***rationality of instance u with respect to target item i*** is the uncertainty reduction of V_i given knowledge of u's descriptive item set $V_{F(u,i)}$. Its value is calculated by

$$R_{u,i} = H(V_i) - H\left(V_i \middle| V_{F(u,i)}\right) \tag{3.3.1}$$

According to above definition, two instances have different rationalities if their descriptive item sets are different. It follows the nature of CF, since actually each user normally rated a rather small portion of items (e.g. about 3% in our used database). The definition of *rationality* points out how sufficiently an instance's description represents itself. In an extreme case, if the uncertainty of target concept is reduced to zero, the given description is the most sufficient.

Theorem 3.3.1 Given an instance u in T_i, if each item j in u's descriptive item set $F(u, i)$ is independent of each other no matter given V_i or not, then the following conclusion holds:

$$R_{u,i} = \sum_{j \in F(u,i)} I(V_i; V_j) \tag{3.3.2}$$

where $I(V_i ; V_j)$ represents the mutual information between V_i and V_j .
Proof: (see appendix)

Theorem 3.3.1 provides a way to calculate the rationality under some assumptions. The independence and conditional independence assumption have been widely adopted in literatures on feature selection [Blum and Langley, 1997] and naive Bayesian classifier [Mitchell, 1997]. And an investigation in our experiment also showed that most items are nearly irrelevant with each other. Since mutual information is always positive, the theorem 3.3.1 indicate the more the item voted, the higher the rationality is. However, does a larger descriptive item set really mean a better relevance? The other question, is logical sufficiency the only issue in identifying a good instance? The answers to the two questions are both No. Consider in a nearest neighbor classification as shown in Figure 3, suppose we already have a good feature X to classify all the instances, if we introduce another irrelevant feature Y such that $I(C;Y)=0$, although instance rationality remains the same, however, the distances between instances have been seriously biased by Y. The performance of 1-NN classifier is greatly degraded. Therefore besides the sufficiency of descriptions, we use a simple heuristic to panelize the instance with long $F(u, i)$. The ***strength of rationality of an instance u*** is defined as follows:

$$R_{u,i}^{strength} = \frac{1}{|F(u,i)|} R_{u,i} = \frac{1}{|F(u,i)|} \sum_{j \in F(u,i)} I(V_i; V_j) \tag{3.3.3}$$

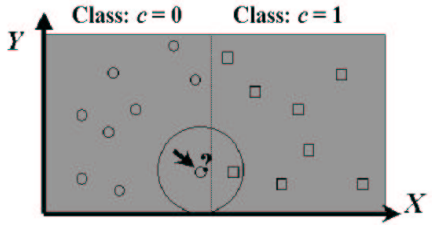

Figure 3. Illustration of NN classifier biased by an irrelevant feature Y

Based on above discussion, we can set up to interpret our training user selection algorithm TURF2. Given a set of training users, who are described by different sets of item-vote pairs, as well as their vote on the target item, the scheme of TURF2 is to select strong and rational training users by comparing the rationality and strength of descriptions between users. A possible way is to select users with enough rationality, and from rational users we pick out relatively stronger ones. However this approach seems complicated in practice. TURF2 applied a simplified approach: since every training user has voted on a fairly large number of items, if someone's rationality is low, he/she could not be with a high strength of rationality. Thus we would like to select users only based on the strength of descriptions. In summary, our algorithm for user selection and prediction proceeds in the following steps:

1. Estimate the mutual information between votes on each pair of items.
2. For each target item i, compute the strength of rationality for all the users in T_i, then select upper $\min(MIN\_SIZE, |T_i| \cdot r)$ users according to a sampling rate r, where MIN_SIZE is set to avoid over-reduction. And thus for each target item i, we create an index table of selected users set T'_i.
3. In prediction phase, calculate the constrained Pearson correlation between query user and every selected training user and find all the neighbors, then make prediction by computing the weighted average of neighbors' votes on target item i.

Later, our experiment will show TURF2 improves the accuracy and leads to a more significant instance reduction. And thus the efficiency is dramatically improved. On the other hand, TURF2 has a better performance than TURF1 in training cost. If we have n users and m items in training set T, the computational complexity of training phase (step 1 and step 2) is $O(nm^2)+O(nm)+O(nlogn)$. Normally, n is much larger than m.

3.4 TURF3 – Selecting Users with a Rational and Novel Profile

As indicated before, TURF1 can reduce the redundancy of training set and speed up the CF runtime, but it suffers from its sensitivity to noisy or imperfect training users and the high computational selection complexity. The drawbacks can be overcome through a noise-filtering pass before TURF1. Our third algorithm TURF3 takes this way to integrate the first two algorithms: eliminating users with unclear profile by TURF2 and then employ TURF1 to further reduce the training users. Compared with other two schemes, the new approach has the following advantages:

1. The high computational complexity of TURF1 is dramatically decreased.
2. The TURF1' sensitivity to noise has been removed.
3. TURF3 leads to a further accuracy and efficiency improvement then TURF2.

In our study, the experimental evaluation showed TURF3 outperforms the former two algorithms in both prediction efficiency and prediction accuracy. And it also significantly reduces the complexity of TURF1.

3.5 TURF4 – Reducing the Storage Requirement of Training Data

In former algorithms, we mainly addressed the problem of CF's efficiency. Now we propose the fourth user selection approach – TURF4, to explore the potential of reducing the number of total users, as well as reduction of storage requirement. A common result of TURF1–TURF3 is that, for a target item, we create a new table containing the index list of new selected training users. In predicting phase, CF engine searches the reduced training user space by reading the index lists. Therefore, TURF1–TURF3 don't actually decrease the total number of users in the training preference database. As a result of TURF1–TURF3, a user I may serve as a selected training user for a number of target items.

Definition 3.5.1 (User utility) For a user I in the training user preference database T, If user I serves as training users for N target items in T, then N is the utility of user I, denoted by $U(I) = N$. For a training user preference database T, its total utility is measured by the sum of utilities of all the users.

Based on the above definition, TURF4 reduces the actual size of database by removing the users with low utility. TURF4 calculates all the users' utilities based on the result of TURF3, and removes the lower ones to ensure the total utility lost of remaining database is less than 10%. The threshold is pictured by point P in Figure 4. Figure 4 also shows the utility of training users in our experiment before and after TURF3 was applied. It shows TURF3 expressively suppress the utility of many users.

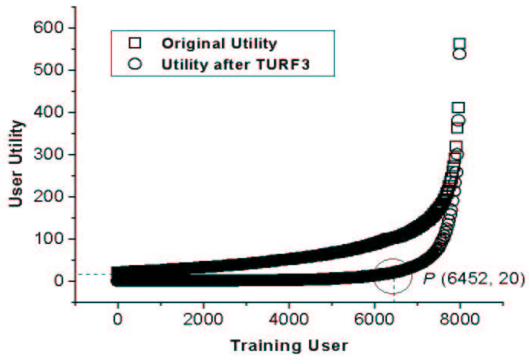

Figure 4. Training user utilities before and after TURF4 was applied

4 Empirical Analysis

In this section we report the results of the experimental evaluation of our proposed algorithms. We describe the data set used, the experimental metrics and the methodology, as well as performance with respect to prediction speed, accuracy, learning speed, and storage reduction. All of our experiments were performed on a Toshiba Notebook (Pentium III 450 MHz, 64 M main memory and Microsoft Window 98).

4.1 The EachMovie Database

We ran experiments using the EachMovie data set, which was part of a research project at the Systems Research Center of Digital Equipment Corporation[1]. The database contains ratings from 72,916 users on 1,628 movies. User ratings were recorded on a numeric six-point scale (We transfer it to 0, 1, 2, 3, 4, and 5). Although 72,916 users are available, we restrict our analysis to 35,527 users who gave at least 20 votes over the totally 1623 movies. For those users whose vote number is less than 20, since their profiles are unclear, they are hard to be used as instances. Moreover, to speed up our experimental time, we randomly select 10,000 users from 35,527 users and divide them into training set (8000 users) and test set (2000 users).

4.2 Metrics and Protocols

As applied in [Breese et al., 1998], we also employ two protocols, *All but One*, and *Given K*. In the first class, we randomly hide an existing vote for each test user, and try to predict its value given all the other votes the user has given. The *All but One* experiments are indicative of what might be expected of the algorithms under steady state usage where the database has accumulated a fair amount of data about a particular user. The second protocol, *Given K*, randomly select *K* votes from each test user as the observed votes, and

[1] For more information see http://www.research.digital.com/SRC/EachMovie/.

then attempts to predict the remaining votes. Its results show the performance when a user is new to a particular collaborative filtering recommender.

We use *mean absolute error* (MAE), where the error is the difference between the actual vote and the predicted vote, to evaluate the accuracy of proposed algorithms. This metric has been widely used in previous work [Breese et al., 1998; Herlocker et al., 1999; Resnick et al., 1994; Shardanand and Maes, 1995]. In addition, [Pennock et al. 2000; Shardanand and Maes, 1995] argue that CF accuracy is most crucial when predicting extreme (very high or very low) votes for items. Intuitively, since the goal is to provide recommendations, high accuracy on the best and worst items is most important. Therefore, we introduce *modified mean absolute error* (MMAE) to measure the accuracy, in which each single prediction error is weighted by the true vote value's deviation from the middle value (here set to be 2.5). For both MAE and MMAE, a lower value indicates a better performance.

In our experiments, we evaluate five different instance selection approaches for CF, as well the basic memory-based CF without instance selection, and compare their performances in efficiency, accuracy, and the storage reduction:

(1) *Baseline Approach*. The memory-based algorithm without instance selection (see 3.1 section).
(2) *Random Sampling.* According to a selection rate r = 0.03125, 0.625, 0.125, 0.25, 0.5 or 1.0, we randomly select 250, 500, 1000, 2000, 4000, or all the users to serve as selected training instances, and the corresponding experimental results are averaged over 4 times.
(3) *TURF1*. We use TURF1 to reduce the redundancy of training set. The only parameter is the initial size of training set, which is set to be 150.
(4) *TURF2, TURF3 and TURF4.* These three algorithms both perform at a selection rate r = 0.03125, 0.625, 0.125, 0.25 0.5 and 1.0 We set up a minimum size of training set to avoid over sampling. For example, for a target movie only 100 users voted on it, it is not necessary to perform further sampling, and hence we set 250 as minimum size of training set.

4.3 Experimental Results

4.3.1 Speed-up of Collaborative Filtering

Our empirical experiments have shown the proposed algorithms all outperformed the baseline algorithm in terms of prediction time. The results are shown in Figure 5 and Table 1. The prediction time of baseline approach is 222 ms per vote. After reducing redundancy of instance set by TURF1, the prediction time was reduced to 122 ms by a factor of 1.82. And TURF2 in different r demonstrated a better improvement of efficiency with speed-up factors varying from 1.80 to 10.1. The third algorithm TURF3, which is the combination of the first two ones, even achieved a better performance than TURF1 and TURF2. The highest speed-up factor reached 12.3 in the case of r = 0.03125. Finally, TURF4 achieves the shortest mean prediction time 16 ms, which indicates the speed-up factor of 13.9. From the Figure 5, we can also observe that the runtime of memory-based CF linearly scales to the number of training users. It confirms that our training user selection techniques can significantly improve the efficiency of CF algorithm.

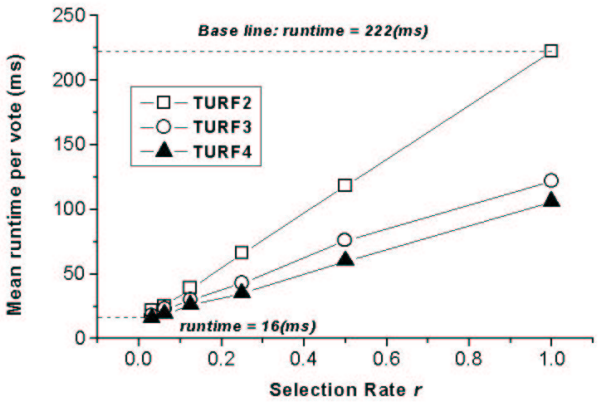

Figure 5. Illustration of mean prediction runtime per vote (Note: TURF2 in case of r=1 is equivalent to baseline approach; Turf3 in case of r=1 is equivalent to Turf1.)

4.3.2 Accuracy for Prediction

	RUN TIME		***MAE***		***MMAE***	
	(ms)	*Speedup factor*	*Value*	*Error reduction*	*Value*	*Error reduction*
Baseline	222	1	0.982	0%	1.174	0%
Random r=0.125	30	7.4	1.008	-2.65%	1.179	-0.43%
Turf1	122	1.8	0.959	2.3%	1.130	3.7%
Turf2 r=0.125	39	5.7	0.959	2.3%	1.130	3.7%
Turf3 r=0.125	30	7.4	0.947	3.6%	1.102	6.1%
Turf4 r=0.125	26	8.5	0.954	2.8%	1.120	4.6%
Turf4 r=0.03125	16	13.9	0.962	2.0%	1.138	3.0%

Table 1. Efficiency and accuracy of memory-based CF algorithms (All but one).

	Given 5		***Given 10***		***Given 20***	
	MAE	*MMAE*	*MAE*	*MMAE*	*MAE*	*MMAE*
Baseline	1.041	1.253	1.022	1.230	1.004	1.206
Turf1	1.035	1.233	1.010	1.202	0.988	1.172
Turf2 r=0.125	1.035	1.238	1.007	1.203	0.985	1.172
Turf3 r=0.125	1.034	1.230	1.000	1.188	0.976	1.153
Turf4 r=0.125	**1.042**	1.246	1.010	1.205	0.984	1.168
Turf4 r=0.03125	**1.045**	**1.255**	1.014	1.214	0.988	1.179

Table 2. Accuracy of memory-based CF algorithms (Given K).

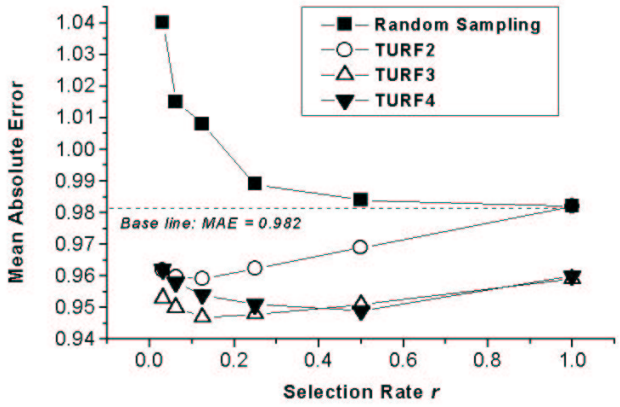

Figure 6. All-But-One mean absolute error of different algorithms (Note: TURF2 in case of *r*=1 is equivalent to baseline approach; Turf3 in case of *r*=1 is equivalent to Turf1.)

Table 2 and Figure 6 show our results of *All but One* test. In all the cases studied, our proposed algorithms outperformed random sampling method, as well as baseline CF which is equivalent to the case of random sampling with a selection rate of 1.0. In Figure 6, the baseline, where MAE accuracy equals to 0.982, is given by the CF over the entire training database. The performance of random sampling shows the fewer training users we select, the poor accuracy we get. In contrast, TURF1 achieved a lower prediction error, with MAE =0.959. This result shows the reduction of redundancy not only reduces the prediction time, but also improves the accuracy. It is consistent with our analysis on TURF1 in section 3.1. Moreover, another interesting point happened in TURF2's results. With the training size getting smaller, the prediction error is even getting lower until *r* =0.125, which indicates TURF2 successfully suppressed the impact of irrelevant instances in the training database. And our results show TURF3 further outperformed TURF1 and TURF2 with corresponding prediction errors MAE =0.947 and 0.953 when *r* =0.125 and 0.03125 respectively. TURF3's results show it successfully combined the first two algorithms: filtering irrelevant instances and then removing redundant instances. For the fourth algorithm, TURF4, although its performance in accuracy is not as good as TURF3, it still outperforms the baseline and TURF1 – 2 . We also used MMAE to evaluate the various algorithms. In contrast to MAE, the improvements in MMAE are more significant, e.g. in the case of TURF3 with r =0.125, decrease of MAE from baseline is 3.56% while that of MMAE is 6.10%. This result is very positive since it showed a better precision for those extreme preferences. Furthermore, Our results of *Given K* (*K*=5, 10 or 20) also present similar results. Table 4 shows the detailed results, in which only 3 results are slightly worse than baseline CF's and have been marked out by bold fonts.

4.3.3 Training Time and the Storage Requirement

Although our proposed algorithms can suppress the prediction error and computational cost, they also introduce additional training cost into CF recommender systems. Table 3 shows the learning time of different algorithms applied for our training database. As we

pointed out in section 3, the learning cost for TURF1 is very prohibitive, while this problem has been dramatically overcome by combining it with TURF2. Our proposed algorithms mainly focus on the speed-up of the prediction process, by performing CF over reduced training sets. Only TURF4 addresses the problem of actual reduction of database. The resulting size of reduced database as well as the size of the original preference database are shown in Table 4.

Since business is ever-growing, recommender systems have to be able to deal with new added data. Obviously, Only TURF1 is incremental among the proposed methods. But in future we believe we can extend them to meet this requirement by the threshold of rationality strength of instances instead of the selection rate *r*.

	TURF1 (hour)	***TURF2 (hour)***	***TURF3 (hour)***	***TURF4 (hour)***
***r*=0.0625**		0.52	0.73	0.73
***r*=0.125**		0.56	0.93	0.94
***r*=0.25**		0.58	1.28	1.28
***r*=0.5**		0.59	2.86	2.86
	7.77			

Table 3. Training time for different algorithms

	TURF4 r=0 .03125	***TURF4 r= 0.0625***	***TURF4 r = 0.125***	***TURF4 r = 0.25***	***TURF4 r = 0.5***	***Original database***
# of users	1548	1609	1731	2325	3788	8000
Storage (MB)	1.79	1.83	1.88	2.22	3.06	4.54

Table 4. TURF4's performance in reduction of the database

5 Conclusions and Future Work

In this paper we have presented four novel instance selection methods, TURF1 – TURF4, to meet the challenge of efficiency for the widely used memory-based collaborative filtering algorithms. Our key idea is to reduce the prediction time by performing neighborhood search in a carefully selected subset of the whole preference database. In TURF1, an incremental scheme is applied to reduce the training set by removing redundant preference patterns from the preference database. While TURF2 reduces the size of training set by filtering the irrelevant preference patterns. In this approach, the sparsity of the preference database is exploited. In the third algorithm TURF3, we combine the first two algorithms: filtering the noises and then reducing the redundancy to overcome TURF1's sensitivity to irrelevant patterns and high computational cost. In the fourth approach, we further explore the potential of actual database reduction by measuring the user's utility for collaborative filtering. Our empirical evaluation on a real-world database shows all the proposed algorithms significantly speed up the prediction time and improve the accuracy, e.g. in the protocol of All But One, TURF4 with a selection rate of 0.03125 improved the efficiency by a factor of 13.9, and the accuracy was improved by 2.0% (MAE) and 3.0% (MMAE) respectively. Among the four algorithms, TURF3 achieves the best performance in accuracy, while TURF4 achieves the best overall performance, since it results in the best efficiency, has a good accuracy, and further reduces the size of database.

We hope this work will pave the way for further improvement of efficiency and accuracy of memory-based collaborative filtering, and also help the model-based

algorithms to address the problem of learning from large databases. Our empirical evaluation is on a real data set collected from an operational movie prediction site. The data set is a typical preference database with explicit votes, and is still the most popular data set used for the research of recommender systems. We believe our work in this paper can be generalized to other application domains, such as videos, books, or web pages. In the near future, we plan to extend this work to the databases in other domains. In addition, we should also develop new incremental CF algorithm to meet the challenges of ever-growing databases.

6 Acknowledgements

We would like to thank the System Research Center of Digital Equipment Corporation for making the EachMovie database available for research.

7 Appendix

Theorem 3.3.1 Given an instance u in T_i, if each item j in u's descriptive item set $F(u, i)$ is independent of each other no matter given V_i or not, then the following conclusion holds:

$$R_{u,i} = \sum_{j \in F(u,i)} I\left(V_i; V_j\right) \tag{3.3.2}$$

where $I(V_i ; V_j)$ represents the mutual information between V_i and V_j .

Proof : According to definition 3.3.1, we have :

$$\begin{aligned} R_{u,i} &= H\left(V_i\right) + H\left(V_{F(u,i)}\right) - H\left(V_i, V_{F(u,i)}\right) \\ &= H\left(V_i\right) - H\left(V_i \mid V_{F(u,i)}\right) \\ &= H\left(V_{F(u,i)}\right) - H\left(V_{F(u,i)} \mid V_i\right) \end{aligned}$$

Since each item $j \in F(u,i)$ is independent of each other no matter given V_i or not, then

$$\begin{aligned} R_{u,i} &= \sum_{j \in F(u,i)} H\left(V_j\right) - \sum_{j \in F(u,i)} H\left(V_j \mid V_i\right) \\ &= \sum_{j \in F(u,i)} I\left(V_i; V_j\right) \end{aligned}$$

Therefore the conclusion eq.(3.3.2) holds. □

8 Reference

[Aha et al., 1991] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based Learning Algorithms", *Machine Learning*, 6: 37-66, 1991.

[Basu et al., 1998] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as Classification: Using Social and Content-based Information in Recommendation", In *Proceedings of the 1998 Workshop on Recommender Systems*, pages 11-15, AAAI Press, August 1998.

[Billsus and Pazzani, 1998] D. Billsus and M. J. Pazzani, "Learning Collaborative Information Filters", In *Proceedings of the International Conference on Machine Learning*, 1998.

[Blum and Langley, 1997] A. L. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning", Artificial Intelligence, 97:245-272, 1997.
[Breese et al., 1998] 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.
[Fisher et al., 2000] D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas and R. Vuduc, "SWAMI: A Framework for Collaborative Filtering Algorithm Development and Evaluation", SIGIR'2000, Athens, Greece, 2000. (short paper)
[Chee et al., 2001] S. H. S. Chee, J. Han, and K. Wang, "RecTree: An Efficient Collaborative Filtering Method", DaWak2001, Munich, Germany, 2001.
[Hart, 1968] P. E. Hart, "The Condensed Nearest Neighbor Rule", IEEE Transactions on Information Theory, 14, pp. 515-516, 1968.
[Herlocker et al., 1999] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering", in *Proceedings of the Conference on Research and Development in Information Retrieval*, 1999.
[Hill et al., 1995] W. Hill, L. Stead, M. Rosenstein and G. Furnas, "Recommending and evaluating choices in a virtual community of use", in *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 194-201, 1995.
[Hinneburg et al. 2000] A. Hinneburg, C. C. Aggarwal, D. A. Keim, "What is the nearest neighbor in high dimensional spaces?", Proceedings of the 26th VLDB conference, Cairo, Egypt, 2000.
[Mitchell, 1997] T. M. Mitchell, "Machine Learning", McGraw-Hill, 1997.
[Pennock et al, 2000] 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*, pp. 473-480, Morgan Kaufmann, San Francisco, 2000.
[Pradhan and Wu, 1999] S. Pradhan and X. Wu, "Instance Selection in Data Mining", Technical Report, 1999.
[Resnick et al., 1994] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews", In *Proceedings of the 1994 Computer Supported Collaborative Work Conference*.
[Sarwar et al., 2000] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Analysis of Recommender Algorithms for E-Commerce", In *Proceedings of ACM E-Commerce 2000 Conference*.
[Sarwar et al., 2000] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Application of Dimensionality Reduction in Recommender System – A Case Study", In *Proceedings of ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, August 20, 2000, Boston, MA.
[Shannon, 1948] C. E. Shannon, "A Mathematical Theory of Communication", *Bell Sys. Tech. Journal*, vol. 27, 1948
[Shardanand and Maes, 1995] U. Shardanand, and P. Maes, "Social Information filtering Algorithms for Automating 'Word of Mouth'", In *Proceedings of CHI'95*.
[Wilson and Martinez, 2000] D. R. Wilson and T. R. Martinez, "Reduction Techniques for Instance-Based Learning Algorithms", *Machine Learning*, 38-3, pp. 257-286, 2000.
[Zhang, 1992] J. Zhang, "Selecting Typical Instances in Instance-Based Learning", in *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, Scotland: Morgan Kaufmann, pp. 470-479.