A Recursive Prediction Algorithm for Collaborative Filtering Recommender Systems

Jiyong Zhang
Human Computer Interaction Group,
Swiss Federal Institute of Technology (EPFL),
CH-1015, Lausanne, Switzerland
jiyong.zhang@epfl.ch

Pearl Pu
Human Computer Interaction Group,
Swiss Federal Institute of Technology (EPFL),
CH-1015, Lausanne, Switzerland
pearl.pu@epfl.ch

ABSTRACT

Collaborative filtering (CF) is a successful approach for building online recommender systems. The fundamental process of the CF approach is to predict how a user would like to rate a given item based on the ratings of some nearest-neighbor users (user-based CF) or nearest-neighbor items (item-based CF). In the user-based CF approach, for example, the conventional prediction procedure is to find some nearest-neighbor users of the active user who have rated the given item, and then aggregate their rating information to predict the rating for the given item. In reality, due to the data sparseness, we have observed that a large proportion of users are filtered out because they don't rate the given item, even though they are very close to the active user. In this paper we present a recursive prediction algorithm, which allows those nearest-neighbor users to join the prediction process even if they have not rated the given item. In our approach, if a required rating value is not provided explicitly by the user, we predict it recursively and then integrate it into the prediction process. We study various strategies of selecting nearest-neighbor users for this recursive process. Our experiments show that the recursive prediction algorithm is a promising technique for improving the prediction accuracy for collaborative filtering recommender systems.

Categories and Subject Descriptors

H.1.2 [**Models and Principals**]: User/Machine Systems—*human factors*; H.3.3 [**Information Search and Retrieval**]: Information filtering—*Selection process*.

General Terms

Algorithms, Performance.

Keywords

Recommender systems, collaborative filtering, prediction algorithm, recommendation accuracy.

RecSys'07, October 19–20, 2007, Minneapolis, Minnesota, USA.

1. INTRODUCTION

We are now in an era with an overwhelming amount of information available but individuals are finding it increasingly difficult to discover useful information. Recommendation systems have been regarded as a remedy to overcome the information overload problem and a lot of research effort has been focused on developing highly reliable recommendation techniques. One of the most popular and successful techniques that has been used in recommender systems is known as *collaborative filtering* [4, 10]. The key idea of this approach is to infer the preference of an active user towards a given item based on the opinions of some similar-minded users in the system [1, 3]. Many popular e-commerce web sites – Amazon.com for example – have adopted this technique in making their online shopping system more efficient.

One of the most prevalent algorithms in collaborative filtering approach is based on the *nearest-neighbor* users (called user-based CF approach in this paper). To predict the rating value of a given item for an active user, a subset of neighbor users are chosen based on their similarity to the active user – called nearest-neighbor users – and their ratings of the given item are aggregated to generate the prediction value for it.

The conventional prediction process of the user-based CF approach selects neighbor users using two criteria: 1)they must have rated the given item; 2)they must be quite close to the active user (for instance, only the top K nearest-neighbor users are selected). However, in reality most users in recommender systems are unlikely to have rated many items before starting the recommendation process, making the training data very sparse. As a result, the first criterion may cause a large proportion of users being filtered out from the prediction process even if they are very close to the active user. This in turn may aggravate the data sparseness problem.

To overcome the data sparseness problem and enable more users to contribute in the prediction process, here we propose a recursive prediction algorithm which relaxes the first criterion mentioned above. The key idea is the following: if a nearest-neighbor user hasn't rated the given item yet, we will first estimate the rating value for him/her *recursively* based on his/her own nearest-neighbors, and then we use the estimated rating value to join the prediction process for the final active user. In this way we have more information to contribute to the prediction process and it should be able to improve the prediction accuracy for collaborative filtering recommender systems. The main contribution of this paper is that we relax the constraint that neighbor users must

also have rated the given item. The proposed recursive prediction algorithm enables more flexibility in the prediction process of finding the useful neighbor users. One important issue is exactly how to select those effective nearest-neighbor users for the prediction process. In this paper we will present the recursive prediction algorithm with various ways of determining the nearest-neighbor users and report their performances.

The rest of the paper is organized as follows. The next section provides a brief review of the related work about collaborative filtering recommender systems. Section 3 recalls the general prediction process of the nearest-neighbor based collaborative filtering approach. In Section 4 we describe the recursive prediction algorithm in detail. Section 5 provides experimental results of evaluating the performance of the proposed approach. Finally discussions and conclusions are provided in Section 6 and Section 7 respectively.

2. RELATED WORK

One of the earliest collaborative filtering recommender systems was implemented as an email filtering system called *Tapestry* [2]. Later on this technique was extended in several directions and was applied in various domains such as music recommendation [12] and video recommendation [5]. In this section we briefly review the research literature related to collaborative filtering recommender systems.

Generally speaking, collaborative filtering algorithms can be classified into 2 categories: One is *memory-based*, which predicts the vote of a given item for the active user based on the votes from some other neighbor users. Memory based algorithms operate over the entire user voting database to make predictions on the fly. The most frequently used approach in this category is nearest-neighbor CF: the prediction is calculated based on the set of nearest-neighbor users for the active user (user-based CF approach) or, nearest-neighbor items of the given item (item-based CF approach). The second category of CF algorithms is *model-based*. It uses the user voting database to estimate or learn a probabilistic model (such as cluster models, or Bayesian network models, etc), and then uses the model for prediction. The detail of these methods and their respective performance have been reported in [1].

In this paper we focus on the user-based CF approach [10]. The general prediction process is to select a set of nearest-neighbor users for the active user based on a certain criterion, and then aggregate their rating information to generate the prediction for the given item. More recently, an item-based CF approach has been proposed to improve the system scalability [7, 11]. The item-based CF approach explores the correlations or similarities between items. Since the relationships between items are relatively static, the item-based CF approach may be able to decreases the online computational cost without reducing the recommendation quality. The user-based and the item-base CF approaches are broadly similar, and it is not difficult to convert an implementation of the user-based CF approach into the item-base CF approach and vice versa.

The method of selecting nearest-neighbors is an important issue for the nearest-neighbor CF approach. In [1] it is found that highly correlated neighbors can be exceptionally more valuable to each other in the prediction process than low correlated neighbors. Herlocker et al. [3] have systematically studied various design issues for the nearest-neighbor CF approach. They have studied the correlation-thresholding technique in the neighbor selection process and found it does not give more accurate predictions than the plain non-thresholding method. Also, they have found that selecting a reasonable size of nearest-neighbors (usually range from 20 to 50) produced the lowest prediction error.

The above studies focused on selecting nearest-neighbor users from those who had also rated the given item. The prediction algorithm used in their work is direct and intuitive. However, this algorithm ignores the possibility that some users may be able to make a valuable contribution to the prediction process, despite not having explicitly rated the given item. By comparison, the recursive prediction algorithm proposed in this paper relaxes such condition and could allow the nearest-neighbor users to be selected from a larger range of candidate users. Based on the recursive prediction algorithm, we propose new techniques for selecting nearest-neighbors to improve the prediction accuracy.

3. NEAREST-NEIGHBOR BASED COLLABORATIVE FILTERING

Herlocker et al. [3] have summarized the procedure of applying the nearest-neighbor based CF approach as the following three steps: 1) Measure all users with respect to the similarity with the active user; 2) Select a subset of users to use as a set of predictors for the given item; 3) Compute a prediction value for the given user based on the ratings from the selected neighbors. In this section we recall some of the most relevant prior work on each step and gives formal definition of the key techniques.

3.1 User Similarity

GroupLens [10] introduced the Pearson correlations to measure similarity between users. Let I denotes the set of items which had been rated by both user x and y, the Pearson correlation similarity between user x and y is given by

$$sim(x,y) = \frac{\sum_{i \in I}(R_{x,i} - \bar{R}_x)(R_{y,i} - \bar{R}_y)}{\sqrt{\sum_{i \in I}(R_{x,i} - \bar{R}_x)^2}\sqrt{\sum_{i \in I}(R_{y,i} - \bar{R}_y)^2}} \quad (1)$$

Where $R_{x,i}$ represents user x's rating of item i, and $\bar{R}_x$ is the average rating value of user x.

In this paper we choose the Pearson correlation as the metric for user similarity. There are several other ways of calculating the similarity among users such as the cosine based similarity. The comparison of the similarity metrics is out of the scope of this paper. For more detailed information about this part of research please refer to [1].

3.2 Selecting Neighbors

Often CF recommenders can have a large number of users and it is infeasible to maintain real-time performance if the system adopts rating information from all users in the prediction process. One popular strategy is to choose the K nearest-neighbors of the active user as the subset predictors [10]. Another strategy is to set an absolute similarity threshold, where all neighbor users with absolute similarities greater than a given threshold are selected [12]. In this paper we choose the K nearest-neighbor strategy as the baseline strategy of selecting neighbors.

Rank	Neighbor User (y)	Similarity ($sim(x,y)$)	Rating ($R_{y,i}$)
1	39	1.000	*n.a.*
2	571	1.000	*n.a.*
3	166	1.000	*n.a.*
4	384	1.000	*n.a.*
5	511	1.000	*n.a.*
6	531	1.000	*n.a.*
7	810	1.000	*n.a.*
8	812	1.000	*n.a.*
9	816	1.000	*n.a.*
10	861	0.968	*n.a.*
11	572	0.963	*n.a.*
12	520	0.919	*n.a.*
13	107	0.917	*n.a.*
14	599	0.905	*n.a.*
15	34	0.891	*n.a.*
16	691	0.890	*n.a.*
17	800	0.887	*n.a.*
18	803	0.883	*n.a.*
19	105	0.882	*n.a.*
20	923	0.872	4
21	900	0.868	*n.a.*
22	414	0.868	*n.a.*
23	702	0.866	*n.a.*
24	808	0.866	*n.a.*
25	485	0.866	*n.a.*
26	400	0.866	*n.a.*
27	510	0.863	*n.a.*
28	364	0.853	*n.a.*
29	791	0.845	*n.a.*
30	304	0.845	*n.a.*

Table 1: The nearest-neighbor users for the active user $x = 1$ to predict the rating value of the item $i = 3$.

3.3 Prediction Computation

Once the user similarity and the subset of neighbor users are determined, we need to aggregate their rating information to generate the prediction value. Resnick et al. [10] have proposed a widely-used method for computing predicted ratings. Formally, for the active user x to the given item i, the predicted rating $\hat{R}_{x,i}$ can be calculated as following:

$$\hat{R}_{x,i} = \bar{R}_x + \frac{\sum_{y \in U_x} (R_{y,i} - \bar{R}_y) sim(x,y)}{\sum_{y \in U_x} |sim(x,y)|} \qquad (2)$$

where U_x represents the subset of neighbor users selected in step 2 for the active user x. The similarity value $sim(x,y)$ can be calculated according to Equation 1 and it acts as a weight value on the normalized rating value $R_{y,i} - \bar{R}_y$. In the conventional CF approach, only those neighbor users who have rated the given item explicitly are selected, so the value $R_{y,i}$ can be fetched directly from the training dataset.

4. THE RECURSIVE PREDICTION ALGORITHM

In this section we illustrate the prediction problem further, using the popular MovieLens dataset. After that, we introduce the new neighbor selecting strategies and the recursive prediction algorithm.

4.1 An Illustrative Example

The MovieLens dataset was collected by the GroupLens research project at the University of Minnesota (project website: *http://movielens.umn.edu*). It consists of 100,000 ratings from 943 users on 1682 movies. This dataset has been cleaned up so that users who had less than 20 ratings or did not have complete demographic information were removed from this dataset. The sparsity level of the dataset is 0.9369 [11], which is quite high.

Rank	Neighbor User (y)	Similarity ($sim(x,y)$)	Rating ($R_{y,i}$)
20	923	0.872	4
98	104	0.593	3
120	157	0.569	3
125	714	0.559	5
142	453	0.522	4
146	569	0.510	1
149	276	0.505	3
150	582	0.504	3
190	463	0.457	2
201	246	0.438	2
202	303	0.437	3
208	429	0.431	2
218	267	0.425	4
231	487	0.415	5
237	472	0.412	5
241	268	0.409	1
244	756	0.407	1
249	660	0.403	1
266	500	0.384	4
271	244	0.380	5

Table 2: The top 20 nearest-neighbors that will be selected in the conventional user-based CF approach for the active user $x = 1$ to predict the rating value of the item $i = 3$.

We partitioned the dataset into 2 parts: 80% of the ratings were used in the training set, and the remaining 20% was used as the testing set. Here we investigate one prediction task in the conventional CF approach: predicting the rating value of a given movie $i = 3$ for an active user $x = 1$. Table 1 lists the top 30 nearest-neighbors for the active user $x = 1$. It shows that most of those nearest-neighbors do not provide rating values for the given item $i = 3$. The only neighbor user who provides rating value for the item $i = 3$ in the list is the user with $id = 923$, which has a similarity value of 0.872 with the active user. Table 2 shows those nearest-neighbors that might be selected in the conventional CF approach. We can see that although the active user ($x = 1$) has many close neighbor users, most of them are filtered out from the prediction process because they haven't rated the given item ($i = 3$) yet. Among all the nearest-neighbor users listed in Table 1, only the user with id=923 can be selected to join the conventional prediction process and others are filtered out.

We believe that the MovieLens dataset is quite representative to most datasets used in collaborative filtering recommender systems. In another words, we believe sparsity is a common issue for collaborative filtering recommenders and the above illustrated problem exists commonly in the conventional prediction process.

4.2 The Strategies for Selecting Neighbors

The above example shows that the conventional CF approach excludes a large proportion of similar users from the set of nearest-neighbors just because they have not rated the given item. In this paper we propose a recursive prediction algorithm to relax this constraint. In addition, we also have observed that some neighbors might have only a few com-

mon ratings with the active user, but they could get very high similarity value with the active user by chance. This is because the calculation of the correlation value among users does not take into account the degree of overlaps between users. If two users have only few overlaps, it is unlikely that they are indeed *close-mined* neighbors.

In summary, we propose the following five strategies for selecting the active user's nearest-neighbors:

1. Baseline strategy(BS): selects the top K nearest-neighbors who have rated the given item. This is the conventional strategy for selecting neighbors as illustrated in [10];
2. Baseline strategy with overlap threshold($BS+$): selects the top K nearest-neighbors who have rated the given item and have rated at least φ items that have also been rated by the active user(overlapped with the active user);
3. Similarity strategy(SS): selects the top K' nearest-neighbors purely according to their similarity with the active user;
4. Combination strategy(CS): combines top K nearest-neighbors selected by the baseline strategy(BS) and top K' nearest-neighbors selected by the similarity strategy(SS);
5. Combination strategy with overlap threshold ($CS+$): combines the top K nearest-neighbors selected by baseline Strategy(BS) and top K' nearest-neighbors selected by the similarity strategy (SS). Also, each user must have rated at least φ items overlapped with the active user.

Please keep in mind that for the BS and BS+ strategy, since all those selected neighbors have provided rating values explicitly to the given item, the prediction value can be calculated straightforward without iteration. However, for the other three strategies(SS, CS and CS+), the recursive prediction algorithm must be applied to estimate the intermediate prediction values.

The BS strategy is an extreme case of selecting nearest-neighbor users. It only chooses users among those who have already explicitly rated the given item. The SS strategy is another extreme case, where only those top nearest-neighbors are considered to be useful for the prediction, no matter if they had rated the given item or not. The CS strategy is a compromise of the above two cases. $BS+$ and $CS+$ are the improved version of BS and CS respectively.

4.3 The Recursive Prediction Algorithm

The goal of the recursive prediction algorithm is to include nearest-neighbors who haven't rated the given item in the prediction process. When the process requires a rating value that doesn't exists in the dataset, we can estimate it recursively on the fly, and then use it in the prediction process. The estimated rating values may not be as accurate as those ratings explicitly given by the users. In our algorithm we specify a weight value to distinguish the different contribution of these two types of ratings.

Formally, our recursive prediction algorithm can be represented as the following:

$$\hat{R}_{x,i} = \bar{R}_x + \frac{\sum_{y \in U_x} w_{y,i}(R_{y,i} - \bar{R}_y) sim(x,y)}{\sum_{y \in U_x} w_{y,i} |sim(x,y)|} \qquad (3)$$

```
Configuration Values:
  ζ — the threshold of recursive level;
  λ — the combination weight threshold;
Function Parameters:
  x — the active user;
  i — the given item to be predicted;
  level — the current recursive level;
return: the predicted rating value;
1.  function RecursivePrediction (x, i, level)
2.    if level ⩾ ζ then
3.      return BaselinePrediction(x, i);
5.    endif
6.    U ⟵ SelectNeighbors(x, i);
7.    α = 0.0;
8.    β = 0.0;
9.    for each y in U do
10.     if R_{y,i} is given then
11.       α += (R_{y,i} − R̄_y)sim(x, y);
12.       β += |sim(x, y)|;
13.     else
14.       R̂_{y,i}=RecursivePrediction(y, i, level + 1);
15.       α += λ(R̂_{y,i} − R̄_y)sim(x, y);
16.       β += λ|sim(x, y)|;
17.     endif
18.   endfor
19.   return R̄_x + α/β;
```

Figure 1: The recursive prediction algorithm.

where U_x is the set of neighbor users for the active user x determined by the respective strategy we have mentioned earlier. If $R_{y,i}$ is not given explicitly from the training dataset, we will apply the recursively predicted value $\hat{R}_{y,i}$ instead. Also, we apply a weight value $w_{y,i}$ to the recursively predicted value. The $w_{y,i}$ is determined as the following:

$$w_{y,i} = \begin{cases} 1 & R_{y,i} \texttt{ is given} \\ \lambda & R_{y,i} \texttt{ is not given} \end{cases} \qquad (4)$$

Here the weight threshold λ is a value between $[0, 1]$.

A detailed implementation of the recursive prediction algorithm is shown in Figure 1. The function "SelectNeighbors (x, i)" is the implementation of one of the 5 different selection strategies. Note that if the recursive algorithm has reached the pre-defined maximal recursive level, it will stop the recursive procedure and call the "BaselinePrediction" function instead. The BaselinePrediction function can be implemented by various strategies – we use the conventional baseline strategy(BS). Each of the five neighbor selection strategies can be applied with the recursive prediction algorithm to form a CF approach. In the rest of the paper, we also use the neighbor selection strategy to represent the corresponding CF approach. For instance, the token CS will also represent the CF approach implemented with the recursive prediction algorithm together with the combination strategy for selecting neighbors.

The recursive prediction algorithm is an extension of the conventional nearest-neighbor based prediction algorithm in the CF approach. It is worth pointing out that the recursive prediction algorithm is equivalent to the conventional prediction algorithm if it is applied with the BS strategy. Therefore BS represents the conventional CF approach, which is the baseline CF approach in this paper.

There are a number of important parameters to be decided before a real feasible CF system can be implemented based on the proposed recursive algorithm. One important parameter of the recursive algorithm is the recursive level. As the recursive level increases, so too does the computational cost. For the final level of the algorithm, we use the BS strategy to select nearest-neighbors so that the recursive algorithm could be terminated within a limited computation time. In addition, the neighbor size is an important factor for these strategies and can have a large impact on overall system performance. Furthermore, we must also set the combination weight threshold for the CS and CS+ strategies. Finally, we need to determine the overlap size for the nearest-neighbor users for both the BS+ and CS+ strategy.

5. EVALUATION

5.1 Setup

Our experiments adopt the MoiveLens dataset that we have mentioned in Section 4.1. The full dataset of users and their rating values is divided into a training set and a test set. In our experiments 80% of the data was used as training data and the other 20% for testing data.

Our experiments are executed on a ThinkPad Notebook (model T41P), which has 1GB memory and one CPU of 1.7GHZ under the Linux operating system. We implement both the baseline algorithm and our recursive prediction algorithm in Java based on the *Taste* collaborative filtering open source project [9].

5.2 Evaluation Metrics

Mean Absolute Error (MAE) is a widely used metric for measuring the prediction accuracy between ratings and predictions for collaborative filtering systems. For each rating–prediction pair $< R_t, \hat{R}_t >$, the absolute error between them can be calculated as $|R_t - \hat{R}_t|$. The MAE is the average value of these absolute errors $|R_t - \hat{R}_t|$ for all items in the test dataset. Formally it is calculated as the following:

$$MAE = \frac{\sum_{t=1}^{N} |R_i - \hat{R}_t|}{N} \quad (5)$$

Where N is the size of the test dataset. The lower the MAE, the more accurate the recommendation system predicts the ratings. Compared to other metrics, MAE is easier to measure, and empirical experiments have shown that mean absolute error has high correlations with many other proposed metrics for collaborative filtering. Mean absolute error is the most frequently used metric among collaborative filtering researchers.

In this paper we adopt MAE as the metric to measure the prediction accuracy. Also, we use the task time to measure the computation cost of the each algorithm.

5.3 Experimental Results

There are a number of parameters which can affect the performance of the recursive prediction algorithm. Here we outline these parameters:

1. the neighbor size (the values of K and K')
2. the recursive level (the value of ζ)
3. the combination weight threshold (the value of λ)
4. the overlap size threshold (the value of φ)

In this paper, we first carry out experiments to determine the value of each parameter, and then we compare the overall performance of the CF approach that adopts the recursive prediction algorithm with the baseline CF approach.

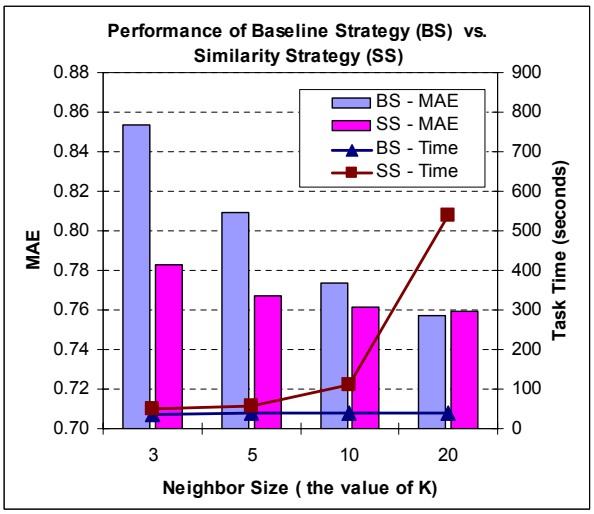

Figure 2: Performance results with various neighbor sizes.

5.3.1 *Neighbor Size for the Similarity Strategy(SS)*

Compared to the baseline strategy(BS) for selecting neighbors, the similarity strategy(SS) is able to adopt those highly correlated neighbor users who haven't rated the given item. Here we compare the prediction accuracy and the task time of completing all predictions in the testing dataset between the two strategies(BS vs SS). The results are shown in Figure 2. For this experiment the recursive level is set to 2.

From the results we can see that the accuracy for both strategies improves as the neighborhood size increases, from 3 to 20. One interesting result is that although the SS strategy includes users who may only have estimated rating values, it can obtain higher prediction accuracy than the BS strategy when the neighbor size is relatively small. For example, if both strategies take only 5 neighbors, the SS strategy reduces the MAE value by 5.2%.

We can also see that the SS strategy requires more computation resources, especially when the neighbor size is large. This is because the SS method needs to estimate the intermediate value recursively. This is a drawback of the SS strategy and it tell us that if we want to improve the prediction accuracy, it is infeasible to use the SS strategy alone. A better approach would be to combine the SS strategy with other neighbor selection techniques.

We noticed that when the neighbor size is 10, the SS strategy can produce low-error predictions, while not being very computationally expensive. In the following experiments we set the neighbor size for the SS strategy as 10.

5.3.2 *Recursive Level*

Recursive level is an important parameter for the recursive prediction algorithm. In this experiment we choose the SS strategy with a neighborhood size of 10. We explore the prediction performance and the algorithm complexity with various recursive levels. Please keep in mind that when the recursive level is zero, the Similarity strategy (SS) is equivalent to the baseline strategy (BS).

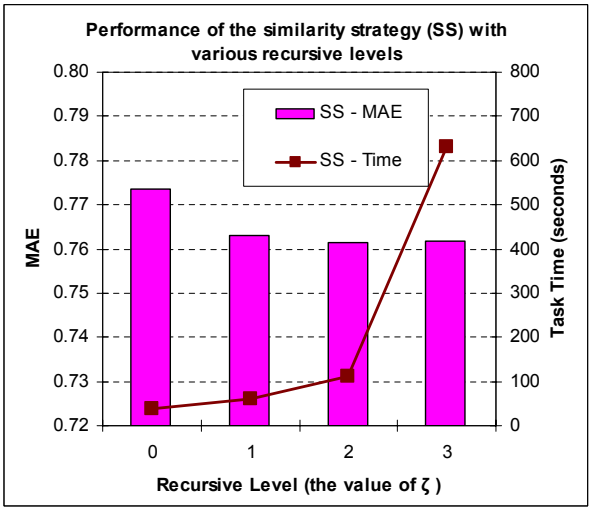

Figure 3: Performance results with various recursive levels.

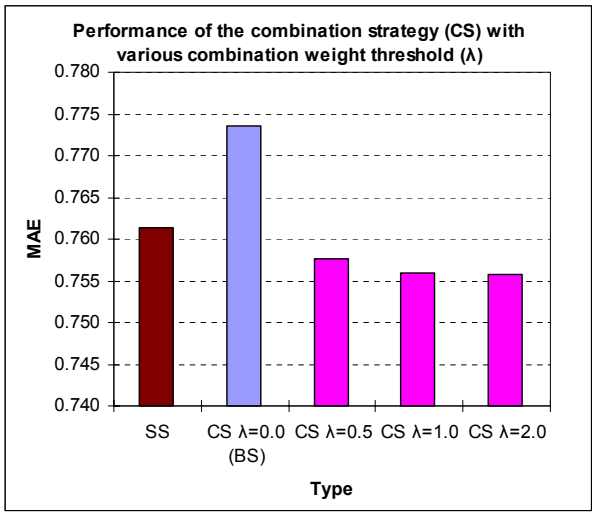

Figure 4: Performance results of the combination strategy(CS) with various combination weight thresholds.

From the results shown in Figure 3 we can see that the prediction performance has an improvement of 1.4% from the non-recursive prediction process to the one with 1 level of recursion. We can also see that when the recursive level is larger than 2, the prediction accuracy doesn't improve significantly, but the task time increases does. To balance the prediction accuracy and the computational cost, we set the recursive level as 2 in our later experiments .

5.3.3 Combination Weight Threshold

As mentioned earlier, the combination strategy (CS) is supposed to benefit from both the conventional baseline strategy (BS) and the similarity strategy (SS). One parameter that we need to determine is the combination weight threshold λ. In this experiment, we set the neighbor size for both the BS strategy and the SS strategy as 10(i.e., $K = 10$ and $K' = 10$), the recursive level is set as 2(i.e. $\zeta = 2$). Please also note that when $\lambda = 0$, The CS strategy is equivalent to the baseline BS strategy.

From Figure 4 we can see that the SS strategy can perform better than the BS strategy. Also, we can get even better performance if we combine them together. For instance, in the experiment when $\lambda = 0.5$, MAE can be reduced by around 1% (compared with the SS strategy) and 3% (compared with the BS strategy) respectively.

5.3.4 Overlap Size Threshold

In this experiment we investigate the relationship between prediction performance and the overlap size threshold for the two strategies: BS+ and CS+. As we can see in Figure 5, increasing the overlap size threshold from 2 to 10 produces better prediction performance for both strategies. However, the prediction performance decreases gradually when the overlap size threshold increases from 10. From this result we can see that a good choice of the overlap size threshold would be around 10. We can also see that in all cases, the CS+ strategy is more accurate than the BS+ strategy given the same overlap size threshold. On average, the CS+ strategy can reduce the MAE by around 2.1% compared to the BS+ strategy when the overlap threshold is set as 10. In the following experiments, we set the overlap size threshold as 10.

5.3.5 Performance Comparison

It is important to know if the recursive prediction algorithm (adopting the SS, CS or CS+ strategy) can lead to substantial performance improvement compared with the traditional direct prediction algorithm (adopting the BS or BS+ strategy) for CF recommender systems. Here we carry out a set of experiments to compare the performance of these strategies for various neighborhood size K. In this experiment, we choose the following parameters for the recursive prediction algorithm: the recursive level ζ is set as 2, and the combination weight threshold λ is set as 0.5. Additionally, the neighborhood size K' is set as 10. The overlap threshold φ for both the BS+ and CS+ strategy is set as 10. The experimental result is shown in Figure 6.

From Figure 6 we can learn several interesting results. Firstly, prediction accuracy increases for all strategies, when the neighborhood size increases from 3 to 50. After that, it levels off. The performance of the baseline strategy (BS) is in line with the results in the earlier literature [3].

Additionally, the BS+ strategy performs better than the BS strategy, especially when the neighborhood size is relatively small. For example, when the neighbor size K is 10, BS+ strategy can reduce the MAE from 0.774 to 0.762 (by 1.4%). Since the only difference between the BS and the BS+ strategy is that BS+ only selects those users with at least 10 overlap items to the active user, we can see that only keeping neighbor users with a high overlap size has a positive influence on the prediction power. However, such improvement is limited with the neighbor size: when the neighbor size is bigger than 50, the BS+ strategy doesn't make any substantial improvement compared with the BS strategy.

Moreover, the CS strategy performs better than the BS strategy, especially when the neighbor size is relatively small. For example, when K=10, the CS strategy can reduce the MAE by 1.9% compared with the BS strategy. This shows

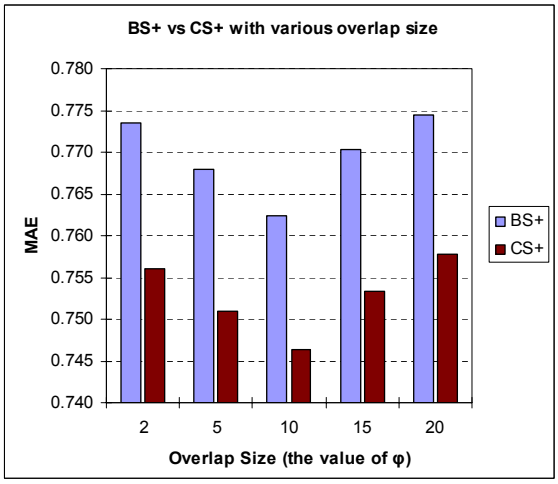

Figure 5: Performance results of the CS+ strategy with various overlap thresholds.

that the proposed recursive prediction algorithm can improve the prediction accuracy. Again, this improvement is not significant when the neighbor size is bigger than 50.

Finally, the results show that the CS+ strategy has the best performance among all strategies in all cases. For example, when the neighbor size is K=10, the CS strategy can reduce the MAE by 3.4% compared with the BS strategy. It is also important to notice that when the neighbor size is large, the CS+ strategy still improves on the BS strategy. For example, when the neighbor size is K=60, the BS strategy reaches its best performance(MAE=0.748). By comparison, the CS+ strategy can further reduce the MAE to 0.742 (0.8% lower than BS strategy).

6. DISCUSSION

To predict the rating value of a given item for an active user, the conventional prediction algorithm in collaborative filtering recommender systems selects neighbor users only from those who have already rated the given item. Because of the dataset sparseness, a large proportion of *nearest-neighbor* users are filtered out without contributing to the prediction process. By comparison, our recursive prediction algorithm is able to keep those *nearest-neighbor* users in the prediction process even though they haven't given ratings to the given item. To the best of our knowledge, this is the first work of improving the prediction accuracy towards the direction of selecting more promising nearest-neighbor users.

The key contribution of this work is that the recursive prediction algorithm enables a larger range of neighbor users to be included in the prediction process of CF recommender systems. Our experimental results show that the proposed prediction algorithm can produce higher prediction accuracy than the conventional direct nearest-neighbor prediction algorithm. When selecting nearest-neighbors with at least a certain number of overlapped rating items with the active user(the CS+ strategy), the recursive prediction algorithm can reduce the prediction error (measured by MAE) by 0.8% compared to the best performance that can be achievable by the conventional user-based prediction algorithm for collaborative filtering recommender systems.

It is worthy to point out that the parameters for the recursive prediction algorithm in our experiments were determined empirically according to the feedback from performance test, and so values are dependent to the current underlying dataset. If the dataset changes, we may need to tune these parameters again to make the algorithm reach its best overall performance on the new dataset.

The work in this paper can be extended in several directions. In this paper we only described our algorithm for the user-based CF recommender systems. This algorithm can be easily applied to the item-based CF recommender systems [7, 11]. This change is intuitive: what we need to do is to change the nearest-neighbor users into nearest-neighbor items. The item-based version of the recursive prediction algorithm is left for future work. Furthermore, trust has been identified as a very useful information in recommender systems [8]. We could also extract trust information from the training dataset and integrate it into our recursive algorithm to improve the prediction accuracy and robustness. For instance, we can apply the trust information as one of the criteria in selecting the nearest-neighbor users in our recursive prediction algorithm to further improve the overall prediction performance. Moreover, recently O'Sullivan et al. [13] has proposed a different approach to ameliorate the data spareness problem. They use data mining technique to increase the similarity coverage among users and could make recommender systems perform better. This technique can also be integrated into the recursive prediction algorithm so to further improve the overall recommendation performance.

One drawback of the current proposed recursive prediction algorithm is that the computational cost is relatively high compared with the conventional prediction algorithm. We noticed that in the recursive prediction process there are a lot of computation redundancies. For example, those intermediate prediction results can be calculated offline, or can be calculated only once and be saved for later use. Thus the online computation complexity could be heavily reduced. Additionally, we could adopt some light computational complexity and low memory consumption algorithm, such as the slope one prediction approach [6], to calculate those intermediate prediction values.

7. CONCLUSIONS

In this paper we proposed a recursive prediction algorithm for CF recommender systems which can predict the missing rating values of the neighbor users, and then apply these missing values to the prediction process for the active user. Our studies show that the recursive prediction algorithm is a promising approach for achieving higher prediction accuracy than the conventional direct prediction algorithm in the nearest-neighbor based CF approach. Specifically, the recursive prediction algorithm together with the CS+ strategy achieved the best prediction performance. When the neighbor size is relatively small (K=10), it can reduce prediction error by 3.4%. When the neighbor size is large (K=60), it can further reduce the prediction error by 0.8% than the best performance that the conventional nearest-neighbor based CF approach could achieve. In future work, we will investigate the soundness of the proposed algorithm on a larger dataset and make it more efficient. We also plan to apply the recursive prediction algorithm on the item-based CF approach to testify its performance.

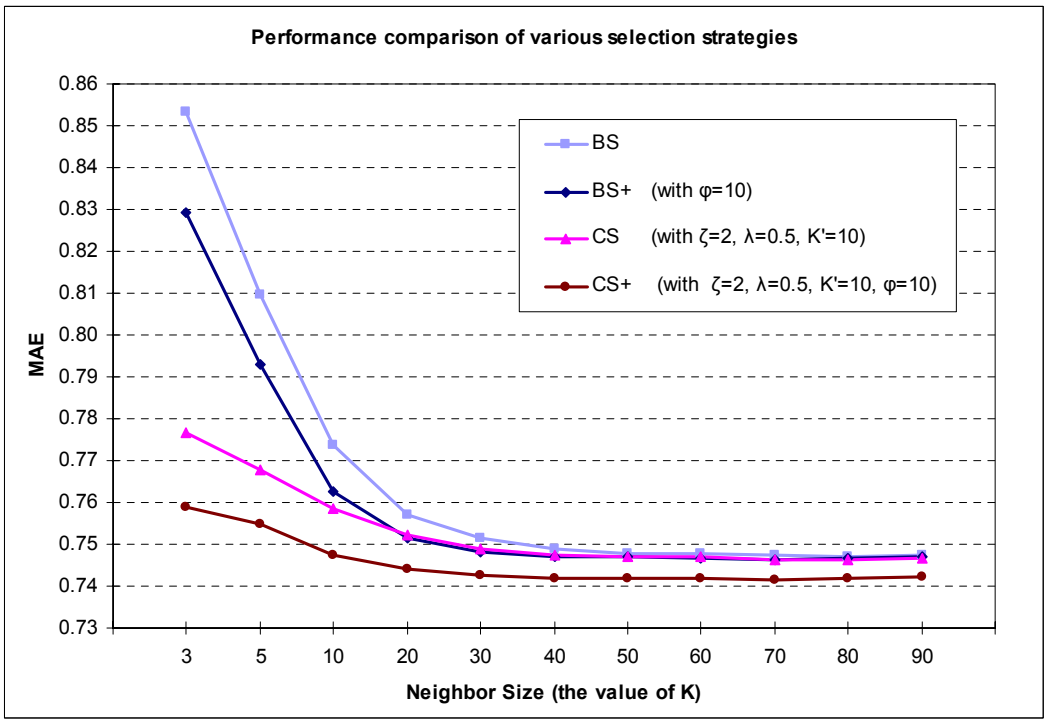

Figure 6: Overall performance comparison of the recursive prediction algorithm with various strategies

8. ACKNOWLEDGMENTS

Funding for this research was provided by Swiss National Science Foundation under grant 200020-111888. We thank James Reilly for proofreading the whole paper.

9. REFERENCES

[1] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, 1998. Morgan Kaufmann.

[2] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

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

[4] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM Press.

[5] W. C. Hill, L. Stead, M. Rosenstein, and G. W. Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI*, pages 194–201, 1995.

[6] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[7] G. D. Linden, J. A. Jacobi, and E. A. Benson. Collaborative recommendations using item-to-item similarity mappings. *US Patent 6,266,649 (to Amazon.com)*, 2001.

[8] J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA, 2005. ACM Press.

[9] S. Owen. *Taste: Open source collaborative filtering for Java: http://taste.sourceforge.net/.*

[10] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM Press.

[12] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *CHI*, pages 210–217, 1995.

[13] D. O. Sullivan, D. Wilson, and B. Smyth. Improving case-based recommendation: A collaborative filtering approach. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 278–291, London, UK, 2002. Springer-Verlag.