An Item-Targeted User Similarity Method for Data Service Recommendation

Cheng Zhang, Xiaofang Zhao
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
zhangcheng@ict.ac.cn; zhaoxf@ict.ac.cn

Jianwu Wang
San Diego Supercomputer Center
University of California, San Diego
San Diego, U.S.A
jianwu@sdsc.edu

***Abstract*—Memory-based methods for recommending data services predict the ratings of active users based on the information of other similar users or items, where the similarity algorithm always plays a key role. In many scenarios, we find that the similarity of two users always show different effectiveness when predicting different ratings. Normal similarity algorithms usually do not count the difference, since they originate from statistic and algebra fields and do not directly aim at recommendations. This paper proposes a novel method to amend the user similarity generated by a normal similarity algorithm to more accurately describe the effectiveness of the similarity on a targeted item. We apply our method to improve the Pearson Correlation Coefficient (PCC) algorithm which is one of the most commonly used similarity algorithms. The experiment results on some practical datasets show that our method is slightly better than the original PCC algorithm for predicting ratings in recommendations.**

Keywords-data services; recommender system; user similarity; collaborative filtering

I. INTRODUCTION

Data service recommendation predicts the interest of data services(items) for an active user by collecting rating information from other similar users or items, and related techniques have been widely employed in some famous web communities [14], such as Amazon[1] and Ebay[2]. The key idea is that the active user will prefer those items that are preferred by similar users.

Memory-based approaches have been deeply studied and widely used in practice, which can be divided in two categories: user-based [4, 8, 10] and item-based [17, 6]. When estimating an unknown test rating, memory-based approaches will firstly collect the similar users or items by some similarity algorithms, such as Pearson Correlation Coefficient (PCC) [16] and Vector Space Similarity (VSS) [4]. Then, the rating information of similar users or items will be handled by some prediction algorithms to produce the estimated value for the test rating.

In many scenarios, we find that the similarity of two users always show different effectiveness when predicting different ratings. For example, user u and v are similar users since they share the preference on items i_1, i_2, i_3, i_4, which are all data services on comedy movies. Item i_5 and i_6 are unknown ratings of user *v*, where i_5 is a data service on comedy movies and i_6 is a data service on science-fiction movie. The ratings of user u on item i_5 and i_6 will be used to predict the rating of i_5 and i_6 for user v respectively according to the similarity between *u* and *v*. The similarity will be more effective to predict the rating of user v on item i_5 than that on item i_6, due to our common experiences that it is unreasonable to infer a user's preference on a science-fiction movie from his/her preference on comedy movies. Similarly, if user *w* and *v* are similar because they both like several action movies, the similarity between user *u* and *v* will be more effective than the similarity between *w* and *v* to predict the rating of user *v* on item i_5. Yet normal similarity algorithms usually do not count such differences, since they generate from more universal statistic principles and do not directly aim at recommendations. So, there is a latent error factor when applying such algorithms into the recommendation tasks directly.

Based on the above observations, we argue that the aspect of the predicted item should be elaborately counted in the user similarity calculation to get better recommendations. This paper proposes a novel method to amend the user similarity generated by a normal similarity algorithm to describe the effectiveness of the similarity on a targeted item more accurately. Given a specific similarity algorithm, our method can work in two ways: one is called *Correction Policy*, which uses item relationships to adjust the user similarity; the other is named *Precision Policy*, which selects the more suitable co-rated items to calculate user similarities.

We note that this paper focuses on the effectiveness description of the user similarity on different items and the influence analyses of the effectiveness on the rating prediction, rather than addressing a combined model to get the optimal recommendation as we have learnt from the Netflix Prize competition [3]. So, we apply our method to improve the Pearson Correlation Coefficient (PCC) algorithm which is one of the most commonly used similarity algorithms. The experiment results on some practical datasets show that our method outperforms the original PCC algorithm for predicting ratings in recommendations.

The rest of the paper is organized as follows. The next section provides a brief review of related work. Section 3 presents our method of the item-targeted user similarity. The

[1] http://www.amazon.com/
[2] http://www.ebay.com/

results of an empirical analysis are presented in Section 4, followed by a conclusion in Section 5.

II. RELATED WORK

A. *Definitions and Notations*

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of items, $U = \{u_1, u_2, ..., u_m\}$ be a set of users, and I_u be the set of items rated by u. The relationship between users and items is denoted by an $M \times N$ matrix, where each entry $r_{u,i}$ represents the rating of item i made by user u. Let $\bar{r}_u$ represent the average rate of user u on all rated items.

B. *Recommendation Methods*

We will first briefly review the development in recommendation. Generally, existed algorithms fall in either of the following two categories: memory-based filtering and model-based filtering [7].

*1)Memory-based Approaches.*Memory-based approaches essentially are heuristics to predict a missing rating by aggregating the rating of k-neighbors who have previously rated the item. To identify the k-neighbors, several algorithms have been proposed to compute the similarity between each pair of users or items. The common-used aggregation is to compute the weighted average of the k-neighbors' ratings on the particular item [1]. Considering the fact that different users may use the rating scale differently, the weighted sum uses their deviations from the average rating of the corresponding neighbor. Two types of memory-based approaches have been studied: user-based [4, 8,10] and item-based [17, 6]. User-based approaches predict the ratings of active users based on the ratings of similar users found, and item-based approaches predict the ratings based on the information of the computed similar items.

2)Model-based Approaches. In contrast to memory-based methods, model-based approaches use the training examples to learn a predefined model, which is then used to make rating predictions. There are various approaches in this category. Typical examples include clustering model [21, 20, 11], aspect models [9, 19] and latent factor models [5]. Dimensionality reduction and matrix reconstruction techniques are often the subject of discussion [2, 18] . Model-based approaches are often time-consuming to build and update, and cannot cover as diverse a user range as the memory-based approaches do [3]. In order to take the advantages of memory-based approaches and model-based approaches, hybrid approaches have been studied recently. For example, Bell et al. combined the output of multiple recommender algorithms to improve performance [12, 2].

3)Similarity Computations. As mentioned above, similarity algorithms are used to collect the similar neighbors in memory-based approaches, and directly affect predictions accuracy. Various approaches have been used to compute the similarity between users or items in recommender systems. We will take user similarities as examples to explain the principal of these approaches. The two most popular approaches are Pearson Correlation Coefficient (PCC) [16] and Vector Space Similarity (VSS) [4]. PCC algorithm [16] measures the similarity between two users based on the items that both users have rated in common:

$$Sim(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}} \quad (1)$$

where $Sim(u, v)$ denotes the similarity between user u and user v, and i belongs to the subset of items co-rated by both users u and v. Obviously, $Sim(u, v)$ is ranging from [0, 1].

VSS algorithm [4] measures the similarity of user u and user v by computing the cosine angles between the rating vectors of them:

$$Sim(u,v) = \cos(\vec{u}, \vec{v}) = \frac{\sum_{i \in I_u \cap I_v} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} r_{u,i}^2} \sqrt{\sum_{i \in I_u \cap I_v} r_{v,i}^2}} \quad (2)$$

where $\vec{u}$ and $\vec{v}$ denotes the rating vectors of user u and v respectively.

Generally, PCC approach is more accurate than VSS approach, since it considers the factor of the differences of user rating styles [14]. The adjusted cosine similarity compensates this drawback by subtracting the corresponding user average from each co-rated pair and has been shown more effective for measuring item similarities [17]. On the other hand, PCC algorithm overestimates the similarities of users who happen to have rated a few items identically, but may not have similar overall preferences [15]. Herlocker etc. proposed to use the following equation to modify the user similarity [15].

$$Sim'(u,v) = \frac{Max(|I_u \cap I_v|, \gamma)}{\gamma} \cdot Sim(u,v) \quad (3)$$

Here γ is a threshold parameter. This method solved the problem when only few items rated but in case that when $|I_a \cap I_b|$ is much higher than γ, the modified similarity will be larger than 1. In order to bound the similarity value to the interval [0,1], Hao etc. proposed another equation [14]:

$$Sim'(u,v) = \frac{Min(|I_u \cap I_v|, \gamma)}{\gamma} \cdot Sim(u,v) \quad (4)$$

Some other researchers address the fact that the similarity algorithms would not work well when there are few items co-rated by both user u and v. It was empirically shown that the rating prediction accuracy could improve if we firstly smooth the missing data of the user-item matrix [14, 21].

III. ITEM-TARGETED USER SIMILARITY

The basic idea of the item-targeted user similarity is to amend the user similarity generated by a normal similarity algorithm to describe the effectiveness of the similarity on a targeted item more accurately. We will explain our method based on the PCC algorithm since it is one of the most commonly used similarity algorithms. Our method can work in two ways to calculate the item-targeted user similarity. In the following two sub-sections, we will discuss them in details.

A. *Correction Policy*

The key idea of Correction Policy is to adjust the original user similarity using a correction factor which measures the similarity effectiveness on the predicted rating, and the original user similarity is calculated by a normal similarity algorithm, such as PCC. The correction factor is as follows:

$$SimQau(u,v,i) = \frac{\sum_{j \in I_U \cap I_v} Sim(j,i)}{|I_u \cap I_v|} \tag{5}$$

where *i* is the predicted rating. Given a specific item *i*, the above equation evaluates the quality of the user similarity according to the similarity relationships between the item *i* and the items which are shared by the user *u* and *v*. We call this factor as *Similarity Quality*. By definition, the value of *Similarity Quality* lies in the range [0,1]. Also, the more similar the shared items and the item *i* are, the larger the *SimQau* value is. Based on *Similarity Quality*, the revised similarity between two users for a specific item *i* is as follows:

$$Sim(u,v,i) = Sim(u,v) * SimQau(u,v,i) \tag{6}$$

Here *Sim*(*u*, *v*) lies in the interval [-1, 1]. *Sim*(*u*, *v*, *i*) will be an order of magnitude lower than *Sim*(*u*, *v*) since the value of *SimQau*(*u*, *v*, *i*) lies in the interval [0, 1]. So the contribution of similar users on the predicted rating will be largely reduced. It is a reasonable inference if we predict the unknown rating using the revised similarity, the predicted rating will be less than it should be. Therefore, a recovery function is required to ensure the revised similarity can keep the effectiveness of the similarity quality while removing the harmful impact mentioned above. So that the similarity is further revised as:

$$Sim(u,v,i) = F(Sim(u,v) * SimQau(u,v,i))$$

$$F(x) = \frac{x}{\sqrt{|x|}}, x \in [-1,1] \tag{7}$$

Obviously, when the value of *x* lies in [-1, 1], the interval of *F(x)* is still in [-1,1] and meanwhile *F(x)* is larger than the absolute value of *x*. In Section 4, we will test the effect of the method on some practical datasets.

B. *Precision Policy*

The key idea of Precision Policy is to select such co-rated items that are similar to the specific item i to calculate the similarity between user u and v. Based on the PCC algorithm, we measure the item-targeted similarity between two users by adding another factor i:

$$Sim(u,v,i) = \frac{\sum_{j \in I_u \cup I_v \wedge Sim(i,j) > \alpha} (r_{u,j} - \bar{r}_u)(r_{v,j} - \bar{r}_v)}{\sqrt{\sum (r_{u,j} - \bar{r}_u)^2} \sqrt{\sum (r_{v,j} - \bar{r}_v)^2}} \tag{8}$$

Where *α* is the item similarity threshold, and *Sim*(*i*, *j*) is computed by the normal PCC algorithm. The selection of the parameter *α* is important since a big value will always cause the shortage of similar users, and a relative small value will not get rid of the impact of the dissimilar items.

In practice, two similar users do not always share such items that are highly similar to the predicted item. In order to adequately utilize the information of such users, the item-targeted similarity is calculated as the follow steps:

- Step 1: To compute the similarity of two users by the Eq.1;
- Step 2: If the similarity is less than β, then it is recalculate by the Eq.8

Where *β* is a value assigned beforehand. The advantage of this method is that we can discover the similarity between users as much as possible, which will provide more useful information for the prediction task. This combined method will be referred to as *c-Precision Policy* and the raw precision policy expressed by the Eq.8 is called *r-Precision Policy*. In Section 4, we will tune the parameter based on our experiments and present the empirical result.

C. *Recommendation Framework*

With the above item-targeted similarity algorithm, there are still several basic problems for recommendations. One is how to select similar users, and another is how to estimate the unknown rating using the similar users' information.

The first question comes from the fact that the similar users with a lower similarity value will decrease the accuracy of the prediction. Although the item-targeted similarity may have an advantage over the original similarity, we still cannot neglect this problem. In this paper, we adopt the method proposed in the paper [14] to deal with this problem. Before making a prediction, a set of k most similar users S(u) toward user u are first generated according to:

$$S(u) = \{u_a | Sim(u_a, u, i) > \theta, u_a \neq u\} \tag{9}$$

where *Sim*(u_a, *u*, *i*) is calculated using Eq.7 or Eq.8 respectively corresponding to the Correction Policy and the Precision Policy, and θ is the similarity threshold. According to Eq.9, the candidate user will be selected as the similar user if the similarity between the candidate user and the active user is larger than θ, which tells us that the selection of θ is important. We will tune the parameter θ based on our experiments and present the empirical results in Section 4.

To solve the second question, a sum of the average rating made by the active user u and a weighted average of the similar users' ratings on the specific item is used to generate the prediction on the unknown rating:

$$r_{u,i} = \bar{r}_u + \frac{\sum_{v \in S(u)} (r_{v,i} - \bar{r}_v) \cdot Sim(u,v,i)}{\sum_{v \in S(u)} Sim(u,v,i)} \quad (10)$$

where $\bar{r}_u$ denotes the average rating made by user u and $Sim(u, v, i)$, which is computed by Eq.7 or Eq.8, is the item-i-targeted similarity between user u and user v. This weighted sum of deviations from the similar users' average rating specially take into account the fact that different users may use rating scale differently.

IV. EXPERIMENTS

We did multiple experiments to evaluate the effectiveness of our method. In particular, we address the following issues:

- How do the recovery function F and the threshold θ affect the accuracy of predictions in the correction policy?
- How do the thresholds α and β affect the accuracy of predictions in the precision policy?
- For the two item-targeted similarity policies, namely *Correction Policy* and *Precision Policy* (including *c-precision policy* and *r-precision policy*), which one more effective?
- Is the item-targeted user similarity better for predictions compared with the user similarity which our method amends?

A. Datasets

Two datasets from movie rating are used in our experiments: MovieLens[3] and Netflix2[4]. We will report the simulation results in details.

TheMovieLens dataset contains 100,000 ratings (1-5 scales) rated by 943 users on 1682 movies, and each user at least rated 20 movies. The density of this datasets is 8.77%. To test on different numbers of training users, we extracted a subset of 500 users with more than 40 ratings. The first 200 users are used for training and we altered the training size to be 50, 100, 200 users. The last 300 users are used for test, and each rating of these users was predicted using an all-but-one policy [4]. The Netflix2 dataset contains over 100 million ratings (1-5 scales) rated by 480189 users on 17770 movies. We randomly extracted a subset of 500 users with more than 80 ratings on 3000 movies. The first 200 users are used for training and we altered the training size to be 100, 200 users. The last 300 users are used for test, and each rating of these users was predicted using an all-but-one policy.

B. Metrics

The major criterion for evaluating rating-oriented recommendation algorithms is the rating prediction accuracy [13]. Several techniques have been proposed to achieve it including Mean Absolute Error (MAE), Mean Squared Error (MSE), Normalized Mean Absolute Error (NMAE), and so on. In this paper, the Mean Absolute Error metrics is selected since it works well for measuring how accurately the algorithm predicts the rating of a randomly selected item [10]. MAE is defined as:

$$MAE = \frac{\sum_{u,i} |r_{u,i} - \hat{r}_{u,i}|}{N} \quad (11)$$

where $r_{u,i}$ denotes the rating that user u gave to item i, and $\hat{r}_{u,i}$ denotes the rating that user u gave to item i which is predicted by our approach, and N denotes the number of tested ratings.

C. Impact of Recovery Function and θ

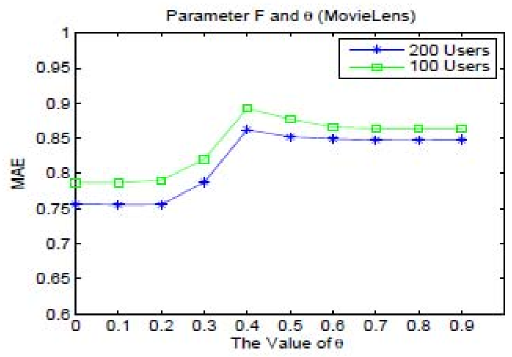

Figure 1. Impact of parameter F and θ on MovieLens dataset.

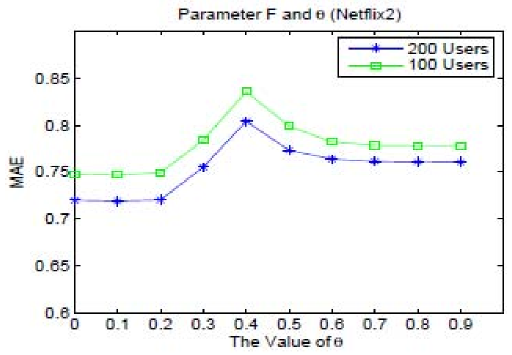

Figure 2. Impact of parameter F and θ on Netflix2 dataset.

As shown in Eq.9, when predicting the ratings under the correction policy, we give a threshold θ to decide the selection of similar users which has been collected. By assigning a different value to θ, the performance of the prediction could be affected. The value of θ is varied from 0 to 1. When setting θ as 0, the algorithm uses all the similar users to complete the prediction. When θ is set as 1, only users that are equal to the active user will be used to predict the unknown rating. We tune the value θ to show the performance on prediction.

Fig.1 and Fig.2 show the performance changes when the threshold θ increases from 0 to 1. We can see that the

[3] http://www.grouplens.org/
[4] http://www.netflixprize.com/

correction policy will achieve the best performance around θ = 0.2 on both the MovieLens dataset and the Netflix2 dataset. The change trends are similar for different numbers of users.

D. Impact of Parameter α

When we adopt the r-precision policy to implement recommendations, parameter α may affect the accuracy of the item-targeted similarity between two users since it is used to decide which items will be selected for computing the similarity value. When setting α as 0, all shared items between two users will be used to compute the similarity, and so the item-targeted similarity algorithm is equal to the normal PCC algorithm. In this experiment, we varied the range of α from 0 to 1 with a step value of 0.1. Fig.3 and Fig.4 show how the parameter α affects the accurcy of predictions on the MovieLens dataset and the Netflix2 dataset respectively.

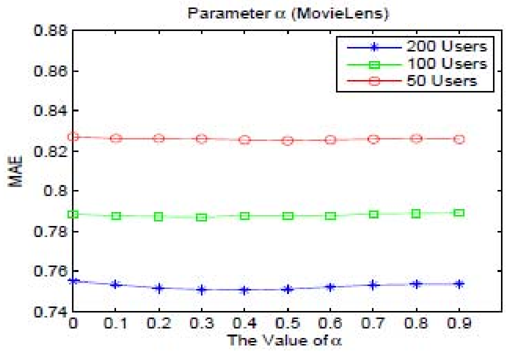

Figure 3. Impact of parameter *F* and θ on Netflix2 dataset.

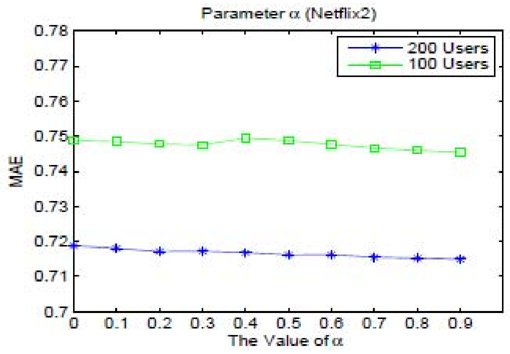

Figure 4. Impact of parameter *F* and θ on Netflix2 dataset.

From Fig.3, we can see that the *r-precision policy* achieves the best performance around α =0.4 on the MovieLens dateset. Fig.4 shows that the r-precision policy achieves the best performance around α=0.8 on the Netflix2 dateset. Although the optimal value of α is different on the two datasets, the change trends are similar for different numbers of users on a certain dataset.

E. Impact of Parameter β

We expect that parameter β can make predictions based on the c-precision policy more accurate in practice since it enables the algorithm to avoid the recalculation for the users with a high similarity value. Based on the above training results, we tested the parameter β on the MovieLens and the Netflix2 datasets. For the MovieLens dataset, we assigned 0.4 to α and selected 50, 100 and 200 training users. For the Netflix2 dataset, we assigned 0.8 to α and selected 100 and 200 training users. In this experiment, we varied the value of β from 0 to 1 with a step value of 0.1. Fig.5 and Fig.6 tell us how the parameter β affects the accuracy of predictions based on the all-but-one policy。

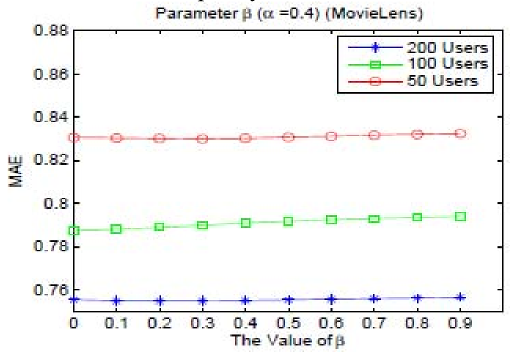

Figure 5. Impact of parameter β on MovieLens dataset

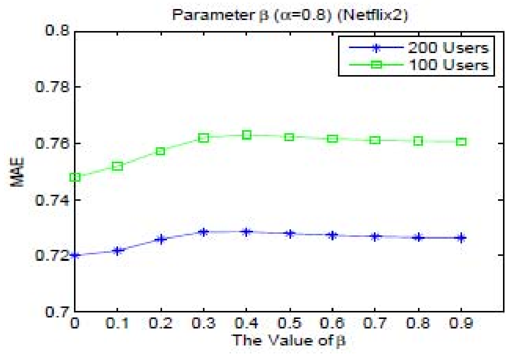

Figure 6. Impact of parameter β on Netflix2 dataset

As showed in Fig.5, given 50, 100 and 200 training users respectively, this *c-precision policy* will achieve the best performance around β=0.1 on the MovieLens dataset. Fig.6 shows that this policy will achieve the best performance around β=0 on the Netflix2 dataset when selecting 100 and 200 training users respectively. Also, the changing tendencies are also similar at different user numbers. This result tells us that the *c-precision policy* have no obvious effects on improving the prediction accuracy.

F. Comparisons

We note that this paper focuses on describing the effectiveness of the user similarity on different items and analyzing the influences of the effectiveness on the rating prediction, rather than addressing a combined model to get the optimal recommendation as we have learnt from the Netflix Prize competition [3]. We discuss how to implement our method by amending a normal algorithm, which will be called original algorithm hereinafter. So, we will compare our method with the original algorithm which is the PCC algorithm in this paper.

Since the *c-precision policy* is a weak method for improving the prediction accuracy, we mainly compare the original PCC algorithm with our *Correction Policy* and *r-Precision Policy* in this experiment. The evaluation on the MovieLens and the Netflix2 datasets is done based on all-but-one policy. For the MovieLens dataset, we set α=0.4, θ =0.2 as suggested by the above results and vary the number of test users from 50 to 200. For the Netflix2 dataset, we set α=0.8, θ=0.2 and vary the number of test users from 100 to 300.

Note that the test users and the training users come from different user sets and the details have been explained in Section 4.1. The results are shown in Tab.1 and Tab.2 respectively.

TABLE I. MAE ON MOVIELENS DATASET FOR THE COMPARED ALGORITHMS

	50 users	**100 users**	**200 users**
PCC algorithm	0.77991	0.75738	0.73821
Correction Policy	0.77708	0.75469	0.73631
r-Precision Policy	0.77155	0.75046	0.73168

TABLE II. MAE ON NETFLIX2 DATASET FOR THE COMPARED ALGORITHMS

	100 users	**200 users**	**300 users**
PCC algorithm	0.76007	0.72306	0.72173
Correction Policy	0.74929	0.72041	0.71883
r-Precision Policy	0.74599	0.71524	0.71766

We note that the performance of different algorithms would improve along with the increase of the test user numbers. This is because, with a larger user database, it will be easier to find sufficient number of neighbors with high similarities to the active user so that his preferences can be estimated more accurately. Also, our method gives better results on the Netflix2 dataset, since this dataset have greater number of movies and it will be easier to find more similar items to calculate the item-targeted similarity. The above test results show that the item-targeted similarity can induce more accurate precisions than the original algorithm.

V. CONCLUSIONS

In this paper, we propose an item-targeted similarity method for data service recommendation based on the observation that the effectiveness of the user similarity is subject to the predicted item. Our method takes into account the relationship of the predicted item and the items used by similarity computations. We select the PCC algorithm as the original algorithm and devise two policies for computing item-targeted similarities: one is the *correction policy* and the other is the *precision policy*. The experiment results on some practical datasets show that our method outperforms the original similarity method when predicting ratings in recommendations.

For future work, we plan to do more research on how to apply our methodology in the calculation of item similarities. We will also try to combine our method with some other techniques and study their usefulness to recommendations

ACKNOWLEDGMENT

This work is supported in part by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under grants 2012BAH19B00.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”. IEEE Trans. on Knowl. and Data Eng., vol. 17(6), pp. 734–749, 2005.

[2] R. Bell, Y. Koren, and C. Volinsky. “Modeling relationships at multiple scales to improve accuracy of large recommender systems”. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 07), pp. 95–104, 2007.

[3] R. M. Bell and Y. Koren. “Lessons from the Netflix prize challenge”. SIGKDD Explor. Newsl., vol. 9(2), pp. 75–79, 2007.

[4] J. S. Breese, D. Heckerman, and C. Kadie. “Empiricalanalysis of predictive algorithms for collaborative filtering”. Proc. the 14th Conference on Uncertainty in Artificial Intelligence (UAI 98), pp.43–52, 1998.

[5] J. Canny. “Collaborative filtering with privacy via factor analysis”. Proc. the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR05), pp. 238–245, 2002.

[6] M. Deshpande and G. Karypis. “Item-based top-n recommendation algorithms”. ACM Trans. Inf. Syst., vol. 22(1), pp. 143–77, 2004.

[7] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. “Using collaborative filtering to weave an information tapestry”. Commun. ACM, vol. 35(12), pp. 61–70, December 1992.

[8] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. “An algorithmic framework for performing collaborative filtering”. Proc. the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 99), pp. 230–237, 1999.

[9] T. Hofmann. “Collaborative filtering via Gaussian probabilistic latent semantic analysis”. Proc. the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (SIGIR 03), pp. 259–266, 2003.

[10] R. Jin, J. Y. Chai, and L. Si. “An automatic weighting scheme for collaborative filtering”. Proc. the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 07), pp. 337–344, 2004.

[11] A. Kohrs and B. Merialdo. “Clustering for collaborative filtering applications”. Computational Intelligence for Modelling, Control and Automation. IOS, 1999.

[12] Y. Koren. “Factorization meets the neighborhood: a multifaceted collaborative filtering model”. Proc. the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 08), pp.426–434, 2008.

[13] N. N. Liu and Q. Yang. “Eigenrank: A ranking-oriented approach to collaborative filtering”. Proc. the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR08), pp. 83–90, 2008.

[14] H. Ma, I. King, and M. R. Lyu. “Effective missing data prediction for collaborative filtering”. Proc. the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR07), pp. 39–46, 2007.

[15] M. R. McLaughlin and J. L. Herlocker. “A collaborative filtering algorithm and evaluation metric that accurately model the user experience”. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR04), pp. 329–336, 2004.

[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. “Grouplens: An open architecture for collaborative filtering of netnews”. Proc. ACM 1994 Conference on Computer Supported Cooperative Work (CSCW94), pp. 175–186, 1994.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. “Item-based collaborative filtering recommendation algorithms”. Proc. the 10th international conference on World Wide Web (WWW01), pp. 285–295, 2001.

[18] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. “Application of dimensionality reduction in recommender system - a case study”. In ACM WebKDD Workshop, 2000.

[19] L. Si and R. Jin. “Flexible mixture model for collaborative filtering”. Proc. the Twentieth International Conference on Machine Learning (ICML 03), pp. 704–711, 2003.

[20] L. Ungar, D. Foster, E. Andre, S. Wars, F. S. Wars, D. S. Wars, and J. H. Whispers. “Clustering methods for collaborative filtering”. In Workshop on Recommendation Systems, 1998.

[21] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. “Scalable collaborative filtering using cluster-based smoothing”. Proc.the 28th annual international conference on Research and development in information retrieval (SIGIR05), pp.114–121, 2005.