Dynamic item-based recommendation algorithm with time decay

Conference Paper · August 2010

DOI: 10.1109/ICNC.2010.5582899 · Source: DBLP

Xiaohong Jiang
Zhejiang University
37 PUBLICATIONS **643** CITATIONS

Yu Zhang
Zhejiang University
23 PUBLICATIONS **367** CITATIONS

Dynamic Item-Based Recommendation Algorithm with Time Decay

Chaolun Xia[1], Xiaohong Jiang[2], Sen Liu[2], Zhaobo Luo[2], Zhang Yu[2]
College of Computer Science
Zhejiang University,
Hangzhou, China
[1]Email: sasnzy12@163.com
[2]Email: {jiangxh, liusen, sixi, yzh}@zju.edu.cn

***Abstract*—Nowadays, the customers involved in e-commercial business are increasing rapidly. To meet their needs, many famous companies, like Amazon and Netflix, place building and optimizing e-commerce recommender systems as their priority. Recommender systems aim to provide personalized advice through mining and discovering the interests and consuming patterns of customers. Generally speaking, recommender systems use two strategies to provide recommendations, namely content-based and collaborative filtering (CF). Furthermore, two primary approaches, namely user-based and item-based, are widely used to build the CF-based top-*N* recommender systems. Item-based approaches have been empirically proved to provide comparable or even better recommendations than those provided by user-based approaches. In this paper, we first introduce the concept of "time decay" by giving its mathematical definition and redefine the item-to-item similarity function based on time decay. Then we study three patterns of time decay and show their effects on recommendations. Based on the above work, finally we present the dynamic item-based top-*N* recommendation algorithm that uses time decay to build models and provide recommendations. Our experiments on real data show that the proposed algorithm provides better recommendations.**

Keywords:* *collaborative filtering, item-based, top-N recommendation, dynamic similarity, time decay function

I. INTRODUCTION

As e-commercial business thrives, recommender systems [1] become a research hotspot. Famous e-commerce companies, like Amazon, Netflix, and Alibaba, take recommender systems as their important research jobs. Recommender systems aim to provide personalized advice through mining and discovering the interests and consuming patterns of users [2].

Generally speaking, recommender systems use two primary strategies to provide recommendations, namely the content-based approach and the collaborative filtering(CF) approach [3]. The content-based approaches profile each user or product, allowing programs to associate users with matching products [4]. By contrast, CF approaches do not require the explicit profiles of users or items, whereas they rely on historical data of user behaviors.

For top-*N* recommendation problems, there are two primary approaches to build the CF-based top-*N* recommender systems [7], namely user-based and item-based [5]. User-based approaches rely on the fact that each user belongs to a certain group whose members share the same interests or have similar behaviors. Different from user-based approaches, item-based approaches build item-to-item similarity matrix and then recommend users the items that are similar to what they have already purchased. Besides the difference of building model, item-based approaches are empirically proved to be capable of providing comparable or even better recommendations than the best available user-based approaches [7,11].

In this paper, we propose the dynamic item-based top-*N* recommendation algorithm that utilizes time decay to build the models and provide recommendations. The contributions of this paper are as follows: (a) it interprets that time factor is one of the factors that have effects on the computations of similarity between items, and the data of time factor can be easily obtained from e-commerce databases; (b) to utilize time factor and analyze its effect on recommendations, this paper introduces the concept of "time decay" by giving its mathematical definition; (c) this paper also redefines the item-to-item similarity function based on time decay, and interprets that the redefined similarity function does not increase the computational complexity of building item-to-item similarity matrix; (d) this paper gives two primary constraints on the choice of time decay functions, and studies three patterns of time decay by illustrating their effects on recommendations; (e) Finally, based on the above studies, this paper presents the dynamic item-based top-*N* recommendation algorithm with time decay and evaluates the proposed algorithm by real e-commerce data. The experimental evaluation shows that the proposed algorithm provides better recommendations, thus the effect of time decay on item-based top-*N* recommendations is positive.

The rest of this paper is organized as follows. After the introduction, Section 2 briefly introduces several recommendation algorithms and describes the basic algorithm of computing item-to-item similarity matrix. Section 3 presents the proposed dynamic item-based top-*N* recommendation algorithm with time decay by redefining the dynamic similarity function and building three time decay functions. Section 4 evaluates the proposed algorithm by real e-commerce data. Section 5 gives the concluding remarks and points out our future work.

II. Related Work

In general, user-based top-*N* recommendation algorithms use three steps to recommend items to a specific user [6,10]. However, as one of the most successful CF-based recommendation algorithms, user-based approaches cannot avoid serious scalability problems because the complexity of computing the user-to-item matrix grows linearly when the number of users and items increases [7]. To reduce their effects, [12] focuses on instance selection to decrease the training set size. By contrast, [7,11] presents empirical evidence which shows item-based approaches can provide better computational performance than traditional user-based approaches, and at the same time, item-based algorithms are capable of providing comparable or even better quality than the best available user-based algorithms [2]. Besides, [11] uses the correlation-based and cosine-based techniques to build similarities matrix between items and obtain the ratings from them. After that, this idea has been further extended in [7] for the top-*N* item recommendations. The rest of this section describes the process of computing item-to-item matrix in item-based top-*N* recommendation algorithms.

```
Algorithm 1 Build_Model($R$, $k$)
For $i \leftarrow 1$ to $m$ do
Begin
    For $j \leftarrow 1$ to $m$ do
        If $i \neq j$ then
            $M_{ij} \leftarrow \mathrm{sim}(R_{*,i}, R_{*,j})$          (*)
        Else
            $M_{ij} \leftarrow 0$
    For $j \leftarrow 1$ to $m$ do
        If $M_{ij}$ is not among the top-$k$ largest in $M_{i,*}$ then
            $M_{ij} \leftarrow 0$
End
Return ( $M$ )
```

In Algorithm 1 [7], R is an $n \times m$ user-item matrix as the input. Usually, R stores either rating or binary values. If R stores explicit ratings, R_{ij} denotes the score of item j given user i. Whereas, if R stores binary values, $R_{ij} = 1$ means user i has viewed or purchased item j and $R_{ij} = 0$ means the opposite. The parameter k means to each item, only k most similar items(or nearest neighbours) are stored in memory for following computations. k is pre-given and the choice of k's value has been fully discussed in [7]. In our method k is set by 20. As the output of this algorithm, M is an $m \times m$ item-to-item similarity matrix. In M, the ith row stores the k most similar items to item i. Usually, M_{ij} is the value of similarity between item i and item j, and if $M_{ij} > 0$, it means the jth item is one of the k most similar items to item i.

The effectiveness of the item-based top-*N* recommendation algorithms largely depends on $sim(R_{*,i}, R_{*,j})$ in Algorithm 1. $sim(R_{*,i}, R_{*,j})$ is to compute the similarity between item i and item j and we briefly review several ways to build it. The first way is based on the cosine of two m dimensional vectors corresponding to the ith and jth row of maxtrix R. For convenience, we use $sim(i,j)$ to substitute for $sim(R_{*,i}, R_{*,j})$.

$$sim(i,j) = \cos(\vec{R}_{*,i}, \vec{R}_{*,j}) = \frac{\vec{R}_{*,i} \cdot \vec{R}_{*,j}}{\left\|\vec{R}_{*,i}\right\|_2 \left\|\vec{R}_{*,j}\right\|_2} \tag{1}$$

The cosine similarity funtion leads to symmetric similarity between items. Another way to compute similarity between any two items is based on conditional probability.

$$sim(i,j) = P(i \mid j) = \frac{Num(i \cap j)}{Num(j)} \tag{2}$$

$P(i|j)$, the probability that the users purchase item i under the condition that they have purchased item j. In particular, $Num(X)$ denotes the number of the users who purchased items in the set X. However, it is obvious that $P(i|\ j)$ is not equal to $P(j|\ i)$ in general, which leads to asymmetric similarity. There are several ways proposed to adjust the asymmetric similarity. For example, [8] multiplies $P(i|j)$ by $log_2P(i)$; [9] divides $P(i|j)$ by $P(i)$; [7] gives Equation 3 to compute similarity between each item:

$$sim(i,j) = \frac{Num(i \cap j)}{Num(i) \times (Num(j))^{\chi}} \tag{3}$$

In Equation 3, the parameter χ takes a value between 0 and 1. When $\chi = 0$, Equation 3 is identical to Equation 2, whereas if $\chi = 1$, sim(i,j) become a symmetric similarity function.

However, the motheds mentioned above to compute item-to-item similarity ignore the time factor of purchase records that can be obtained easily from most of e-commerce databases. Thus in Section 3 we propose a method that utilizes time factor of purchase records to compute item-to-item similarity matrix.

III. Dynamic Item-based Top-*N* recommendation Algorithms with Time Decay

A. Problems

One of the primary motivations behind item-based recommendation algorithms depends on the fact that the two items purchased by the same user are likely related to each other[7]. However, this motivation does not mention the fact the similarity between the two purchased items varies according to the time interval between the two purchase behaviors. For example, it is reasonable to understand that the similarity between the two items that are purchased by the same user on the same day may differ from the similarity between the two items that are purchased by the same user in the same year. This phenomenon is caused by the time factor and we call it "time decay".

In this section, we first introduce the concept of time decay by give its mathematical definition and redefine the dynamic item-to-item similarity function based on time decay. Then we study three patterns of time decay and use the concave, convex and linear functions to simulate them respectively. Based on the above study, we present the dynamic item-based top-*N* algorithm with time decay.

In our method, it is assumed that a record from database is a triple <*t, u, i*> which represents user *u* purchased item *i* on the *t*-th day. The rest of this section consists of two parts, building the dynamic model though defining dynamic similarity function based on time decay, and applying the model.

B. Dynamic Similarity Function with time decay

To quantify and utilize the effect of time decay, we give a new symmetric definition of *sim(i,j)* and name it dynamic similarity:

$$sim(i,j) = \sum_{k=1}^{n} [R_{ki} \cdot R_{kj} \cdot \theta(|T_{ki} - T_{kj}|)] \quad (4)$$

In Equation 4, *n* is number of users. R_{ki} denotes a binary value and if user *k* has purchased item *i*, R_{ki} is set by 1, otherwise by 0. T_{ki} and T_{kj} are two algebraically calculable timestamps(e.g., Unix second code) that respectively record the time when user *k* purchased item *i* and item *j* respectively. $|T_{ki}$-$T_{ki}|$ denotes the number of basic time interval units between T_{ki} and T_{kj} (e.g., we can use one day, three days or one month as the basic time interval unit). $\theta(x)$ named Time Decay Function(TDF), quantifies and computes the effect of time decay on the similarity. To see more clearly, the variable *x* denotes the value of time interval that is counted as the number of basic time interval unit. Note that we do not divide *sim(i,j)* by a normalizing factor (e.g., $\sum_{k=1}^{n} R_{ki} \sum_{k=1}^{n} R_{kj}$) simply because it reduces the recommendation precision in experiments.

C. Time Decay Functions

On the principle of time decay's effect, there are two primary constraints on the choice of $\theta(x)$:

- $\forall x$, $\theta(x) \in [0,1]$ (i)
- $\forall x, x'$, if $x \geq x'$ then $\theta(x) \leq \theta(x')$ (ii)

Constraint (i) ensures every $\theta(x)$ is a "decaying" function and Constraint (ii) ensures that the larger time interval is, the stronger the effect of time decay is. In practice, if there are more basic time interval units between any two purchase behaviors, then the two purchased items are less similar. Our method implements $\theta(x)$ in three ways.

1) Concave Time Decay Function

Here we use a concave function to stimulate the time decay as follows.

$$\theta(x) = \alpha^{x} \quad (5)$$

In Equaion 5, the parameter $\alpha \in [0,1]$ is named concave Time Decay Coefficient(TDC) in our method. Generally speaking, the smaller α is, the stronger of the time decay's effectiveness is. In particular, When $\alpha = 1$, $\theta(x)$ is always equal to 1 and reflects no effectiveness of time decay, whereas if $\alpha = 0$ (for convenience, here we assume 0^0 is 1), it means if the time interval between two purchase behaviors is larger than or equal to one basic time interval unit, the similarity contributed by the purchaser will not be taken into consideration ($\forall\ |T_{ki} - T_{kj}| > 0, \theta_{\alpha=0}(|T_{ki} - T_{kj}|) \equiv 0$).

2) Convex Time Decay Function

We use a convex function to stimulate the time decay.

$$\theta(x) = 1 - \beta^{t-x} \quad (6)$$

In Equaion 6, parameter *t* represents the value of the largest time interval in the training dataset. For example, among all the records, if the earliest purchase record is <$timestamp_{earlist}$, u_1, i_1> and the latest purchase record is <$timestamp_{latest}$, u_2, i_2>, *t* is equal to $|timestamp_{latest} - timestamp_{earliest}|$. The parameter $\beta \in (0,1)$ is the convex TDC. The larger β is, the stronger of the time decay's effectiveness is.

3) Linear Time Decay Function

We use the linear function to stimulate the time decay.

$$\theta(x) = 1 - \frac{x}{t} \cdot \gamma \quad (7)$$

In Equation 7, the parameter $\gamma \in [0,1]$ is the linear TDC. The larger γ is, the stronger of the time decay's effectiveness is. The parameter *t* is defined identically in Equation 6.

D. Applying the Dynamic Model

The process of producing recommendation results to a centain user, is shown in Algorithm 2.

```
Algorithm 2 Apply_Model($M,U,N$)
T ← Zero-Vector
For $i$ ← 1 to $m$ do
    If $U_i$ = 1 then
    Begin
        For $j$ ← 1 to $m$ do
            If $M_{ij} > 0$ then
                $T_j \leftarrow T_j + M_{ij}$
    End
For $i$ ← 1 to $m$ do
    If $T_i$ is not among the top-$N$ largest in $T$ then
        $T_i \leftarrow 0$;
Return ( $T$ )
```

In particular, *U* is an *m*×*1* vector that stores the purchase informations of a specific user. U_i = 1 means the specific user has purchased item *i* and U_i = 0 means the opposite. *N* is the number of recommendations the algorithm provides and it is pre-given. The *m* × *1* vector *T*, as the output, stores the personalized top-N recommendations provided by our algorithms to the specific user. If $T_i > 1$, the algorithm recommends the specific user item i as one of his/her top-N items. We will evaluate the recommendation results by real data in Section 4.

E. Computational Complexity

The computational complexity of proposed algorithm depends on the amount of time required to build the model and apply the model and it does not increase after redefining the dynamic similarity function based on time decay. The upper bound on the overall computational complexity is $O(m^2n)$. However, the worst case will occur rarely because the item-to-item similarity matrix M is always sparse. In our dataset, the M is more than 99 % sparse, so instead of matrices, we use linked lists to minimize unnecessary calculation.

IV. EXPERIMENTS AND RESULTS

A. Experimental Dataset

The dataset we use to evaluate the proposed method is a real e-commerce data released by Alibaba China. It is available at *http://legendcode.alibaba-inc.com/intro-project.jsp#c*. The historical dataset consists of 838,174 view records from 51367 users and 62120 items. A view record is a triple <*t, u, i*> that represents user u viewed item i on the t-th day. All the users viewed all the items in past eight consecutive days that are represented chronologically as $\{d_1,...,d_8\}$ and we use "one day" as the basic time interval unit for Time Decay Function. We use all the data and divide them into two parts, D_A and D_B. D_A consists of 424,688 view records in which t is among the first 4 days ($t \in \{d_1,...,d_4\}$) and we use it as training data to compute the item-to-item similarity matrix M and then produce the top-N recommendation vector T for each user. D_B consists of the other 413,486 view records in which t is among the last 4 days ($t \in \{d_5,...,d_8\}$) and we use it as test data to evaluate the algorithms.

B. Evaluation Method

We use precision to evaluate our method. First we define Customer Satisfaction Index (CSI) as:

$$CSI(i) = \frac{|T_i \cap V_i|}{Min(|T_i|,|V_i|)} \times 100\% \tag{8}$$

In particular, *CSI(i)* denotes the satisfaction degree of user i on the recommendations provided by algorithms. V_i is the set of items user i viewed in the last 4 day(recorded in D_B). T_i is the set of the items that algorithms recommended to user i. Note that $|T_i| = N$, and N is the number of recommendations provided by algorithms to each user. Then we define precision as:

$$precision = \frac{1}{n}\sum_{i=1}^{n} CSI(i) \tag{9}$$

Based on *CSI* that reflects a user's satisfaction degree, the precision function reflects the average satisfaction degree of all the users. Most important of all, it reflects every user's satisfaction degree equally.

C. Comparation of Algorithms

First we use the concave time decay function in Equation 5 to compute the dynamic similarity. We build the model in Algorithm 1, apply the model in Algorithm 2 to produce top-N recommendations with N = 10, and then get the precision of recommendation results. Figure 1 shows the precision curve of concave TDC α varying from 0 to 1. When $\alpha = 0.8$, the model reaches its highest precision, 0.191%.

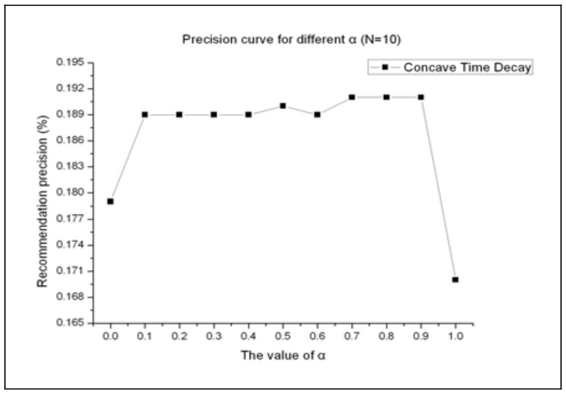

Figure 1. Precision curve of concave time decay function

In Figure 2, we use the convex time decay function in Equation 6 to compute the dynamic similarity and get its precision curve. When $\beta = 0.6$, the model reaches its highest precision 0.191%. Note that $\beta \in (0,1)$, so we replace 0 by 0.01 and 1 by 0.99 in actual experiments.

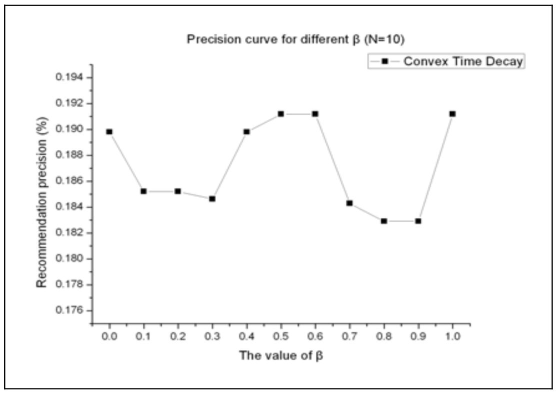

Figure 2. Precision curve of convex time decay function

In Figure 3, we use the linear time decay function in Equation 7 to compute the dynamic similarity and get its precision curve. When $\gamma = 0.7$, the model reaches its highest precision 0.191%. In all the above experiments, we set N to 10.

In Figure 4, we compare the effects of the three time decay functions on recommendation precision. As Figure 4 shows, the highest recommendation precision of all is 0.191%, and all the three time decay functions can reach it.

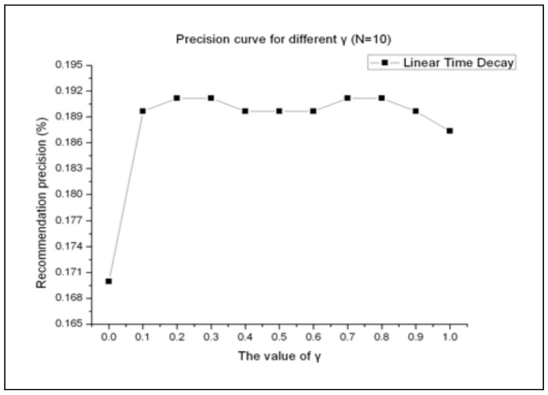

Figure 3. Precision curve of linear time decay function

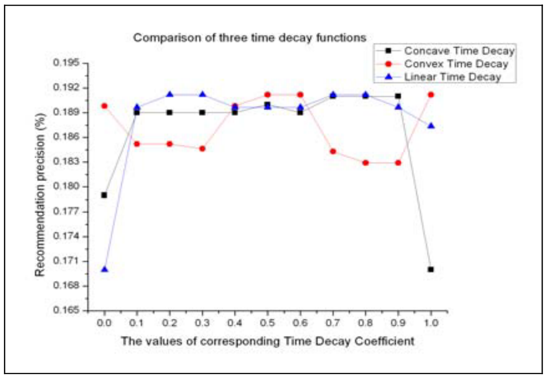

Figure 4. Precision curves of three time decay functions

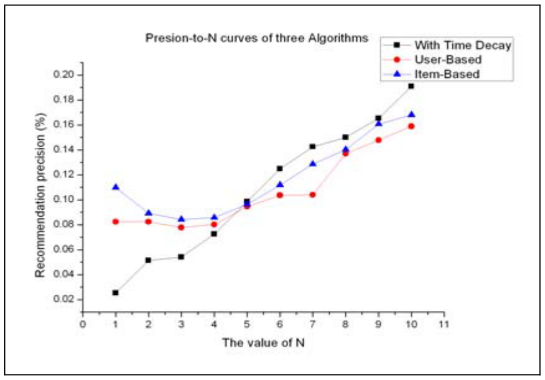

Figure 5. Precision-to-*N* curve

In Figure 5 we compare the precision-to-*N* curves of user-based algorithm [7], item-based algorithm [7], and the proposed item-based algorithm with time decay. We vary *N* from 1 to 10. The black line with square symbols represents the precision-to-*N* curve of the proposed item-based algorithm with time decay where we implement concave time decay function and set $\alpha = 0.8$ (Because when using concave time function, $\alpha = 0.8$ makes the model reach highest precision). The red line with round symbols represents the precision-to-*N* curve of the user based algorithm described in [7]. The blue line with triangular symbols represents the precision-to-*N* curve of the item based algorithm [7] where we implement Algorithm 1 and the cosine-based similarity function in Equation 1. It can be learned from Figure 5 that when *N* is greater than 4, the time decay function improves the recommendation precision and the recommendation precision grows as *N* increases.

D. *Result Analysis*

As a summary, Table 1 shows the highest precision of three time decay functions and the value of the corresponding TDC when the funtions reach the highest precision.

TABLE I. THREE TIME DECAY FUNCTION

Time Decay Functions	Highest Precision (%)	The value of TDC
Concave	0.191	$\alpha = 0.8$
Convex	0.191	$\beta = 0.6$
Linear	0.191	$\gamma = 0.7$

According to Table 1, we can learn that the proposed algorithm with any time decay function can reach the recommendation precision at 0.191% if we proper choose the corresponding TDC's value.

TABLE II. PRECISIONS OF TOP-*N* RECOMMENDATION ALGORITHMS

Top-*N* recommendation algorithms	Precision (%)
Random	≈0.01
User-Based	0.159
Item-Based	0.168
Item-Based With Time Decay	0.191

Table 2 shows the precisions of four top-*N* recommendation algorithms (*N*=10). The precision of random top-*N* recommendation algorithm that recommends each user *N* item randomly is approximately equal to 0.01%. The precision of the user-based approach is 0.159%. The precision of the item-based top-*N* recommendation algorithm is 0.168%. The precision of the proposed item-based algorithm with time decay is 0.191%, the highest of all.

V. CONCLUSION & FUTURE WORK

In this paper, we present the dynamic item-based top-*N* recommendation algorithm with time decay. First, we introduce the concept of "time decay" by giving its mathematical definition and redefine the dynamic item-to-item similarity function based on time decay. Then we study three patterns of time decay and show their effects on recommendations. As the experiment results Table 1 and Table 2 both illustrate the effects of three time decay fucntions on recommendations and show the proposed algorithm with time decay provides better recommendations when we propoerly choose the value of the correspoding time decay coefficient. Besides, the precision curves in Figure 1, 2 and 3 also show the fact that as long as not against Constraint (i) and (ii), even if we use any of the three time decay funtions and then arbitrarily choose its time decay coefficient's value, the recommendations provided by the proposed algorithm will not be worse than the basic item-based top-*N* algorithm. The fact shows time decay has a postive effect on recommendations.

Since we have chosen a time decay funtion and set its time decay coefficient to a proper value, it is reasonable to infer that if the time interval of any two purchase records is very large, the two purchase records will contribute little to the similarity of the two items they recorded. For example, if there are two records r_1=<2010-1-15, $user_i$, $item_1$> and r_2=<2010-3-15, $user_i$, $item_2$> and we choose the concave time decay fuction in Equation 5, set $\alpha = 0.8$, and use "one day" as the basic time interval unit, the dynamic similarity increment between $item_1$ and $item_2$ both purchased by user i is 0.8^{59}. Obviously it is too small to be considered. Therefore, when using the proposed algorithm, it is not necessary to compute the dynamic similarity between each two items. Instead, we can (i)sort the records chronologically and (ii)then only compute the dynamic similarity increment caused by two records that were recorded within a pre-given time interval. Under this assumption, the computational comlexity drops and the quality of recommendations remains the same. In future work, we will exame the assumption by analysis and experiments.

ACKNOWLEDGMENT

Thanks to the professors and students in Service Science & Grid Computing Lab at Zhejiang University. This work is supported by China 863 Project (Grant No. 2006AA01A123) and the Key Technological Project (No.2009AA011900).

REFERECES

[1] P. Resnick and H. R. Varian. Recommender systems – introduction to the special section. Commun. ACM, 40(3):56–58, 1997.

[2] G. Adomavicius and A. Tuzhilin, "Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", IEEE Transactions on Knowledge and Data Engineering 17, 634–749, 2005.

[3] D. Goldberg, D. Nichols, B.M. Oki and D. Terry. "Using Collaborative Filtering to Weave an Information Tapestry". Communications of the ACM 35, 61–70, 1992.

[4] R. M. Bell and Y. Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. Proc. IEEE International Conference on Data Mining (ICDM'07), 2007.

[5] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems. 210−217, 1995.

[6] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. Communications of the ACM 40, 3, 77−87, 1997.

[7] M. Deshpande and G. Karypis, "Item-Based Top-N Recommendation Algorithms," ACM Trans. Information Systems, vol. 22, no. 1, pp. 143-177, 2004.

[8] G. Salton. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley. 1989.

[9] B. Kitts, D. Freed, and M. Vrieze. Cross-sell: A fast promotion-tunable customer-item recommendation method based on conditional independent probabilities. In Proceedings of ACM SIGKDD International Conference. 437−446, 2000.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In Proceedings of ACM E-Commerce. 2000.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. "Item-Based Collaborative Filtering Recommendation Algorithms,". Proc. 10th Int'l WWW Conf.. 2001.

[12] K. Yu, X. Xu, J. Tao, M. Ester, and H. Kriegel Instance Selection Techniques for Memory-Based Collaborative Filtering In Proc. of the Second Siam Intl. Conf. on Data Mining, (SDM) 2002.