Knowledge Push Algorithm Based on Hot Item Punishment and User Interest Change

Haihui Xiao and Hongming Wang*

Changzhou Textile Garment Institute, No. 5 Gehu Road Changzhou, Jiangsu, 213164, China

Abstract: In order to improve the accuracy of user similarity calculation and knowledge push, this paper puts forward a collaborative filtering push algorithm based on hot item punishment and user interest change. Firstly, this algorithm is utilized to cluster knowledge items into several classes; and then in each class, the user interest degree function is introduced to predict the rating value of unrated knowledge items; and in user similarity calculation of each class, hot item weight coefficient is introduced to punish the influence of hot items on user similarity; and finally weight coefficient of user interest change with time is introduced to push. The experiment also uses Movie Lens data set to test the algorithm, and the results show that the improved algorithm is better than traditional collaborative filtering algorithm in terms of push accuracy.

Keywords: Collaborative filtering, hot item punishment, knowledge item clustering, user interest change.

1. INTRODUCTION

With the popularization of the internet and the rapid development of e-commerce, information and knowledge are generated explosively. To seek content that users really need from massive information and provide for users timely, a lot of e-commerce enterprises have developed their own individualized knowledge push systems on websites and the core of such systems is the push algorithm [1]. At present, traditional collaborative filtering (CF) push algorithm is the most widely used [2, 3], and its main ideas are: to find the nearest neighbors of target users by calculating similarities of target users to other users, predict rating of knowledge items to be pushed based on rating of knowledge items by the nearest neighbors, and then select the top *N* knowledge items with the highest rating for *Top-N* push. This algorithm is advantageous as it doesn't require consideration of representation of knowledge content or the similarity of knowledge content to user interest, but pushes according to similarities among users and discovers new resources of interest for users. However, it has also some obvious deficiencies, such as data sparseness and scalability problems in push system, no consideration of changes of user interest with time, and the possible influence of hot items on the calculation of user similarity, all of which would lead to deviation of pushed information from users' actual needs [4].

Thus, many scholars put forward rating prediction methods for knowledge items to reduce sparseness, improve real-time response ability of push system by clustering, and introduce a user interest degree model based on time weight to improve the accuracy of similarity calculation and target user push. Xue *et al.* [5] put forward a collaborative filtering algorithm based on clustering, and the introduction of clustering is conducive to improving the real-time response ability of push system. VozalisM *et al.* [6] put forward a collaborative filtering algorithm based on singular value decomposition (SVD) to improve the accuracy of push algorithm from the perspective of singular value. Ding *et al.* [7] put forward a collaborative filtering algorithm based on time weight to analyze in detail the influence of time factor in collaborative filtering algorithm. Wang Qian *et al.* [8] put forward a collaborative filtering algorithm based on user preference attribute value for specific analysis of user attribute factor. Liu Fangxian *et al.* [9] introduce exponential time function to collaborative filtering push algorithm and carry out rating prediction of unrated knowledge items, which can improve the accuracy of push system effectively and alleviate data sparseness of system etc. Zheng Xianrong *et al.* [10] introduce a nonlinear gradual forgetting mechanism to user forgetting ability model to alleviate the influence of change of user interest with time on push accuracy.

The above improved push algorithms based on collaborative filtering can improve push accuracy from some aspects, but most of them haven't considered about the influence of hot items on user similarity or the role of time weight in data filling. Based on existing research findings, this paper improves significantly the push accuracy of improved algorithm by introducing hot item punishment function to user similarity calculation and introducing time weight to prediction of unrated knowledge items and final rating push of users.

2. TRADITIONAL CF ALGORITHMS

Traditional collaborative filtering algorithms can be divided into two categories which are based on users and knowledge items respectively [11]. Collaborative filtering algorithms based on users are to get the nearest neighbors of

Table 1. User-knowledge item rating matrix.

User	**Knowledge Item**			
	Knowledge Item 1	**Knowledge Item 2**	**…**	**Knowledge Item *n***
User 1	r_{11}	r_{12}	…	r_{1n}
User 2	r_{21}	r_{22}	…	r_{2n}
…	…	…	…	
User *m*	r_{m1}	r_{m2}	…	r_{mn}

target users through analysis of similarities among users and then conduct push prediction by different algorithms combined with evaluation of neighbor users. Collaborative filtering algorithms based on knowledge items are to study on similarities among knowledge items, get the nearest neighbors of target knowledge items by similarity algorithm, predict users' rating of target knowledge items according to neighbors' rating of knowledge items, and then select the top *k* knowledge items for push. This paper will focus on collaborative filtering algorithms based on knowledge items.

2.1. Traditional CF Algorithms

Traditional collaborative filtering algorithms have generally three steps, namely: data representation; seek the nearest neighbors; and generate a push data set [12].

(1) Data representation. Firstly, get users' rating of commodities, establish an $m \times n$ user-knowledge item rating matrix, as shown in Table **1**. *m* is the number of users, *n* is the number of knowledge items, matrix element $r_{i,j}$ denotes rating of user *i* for knowledge item *j*, the value of $r_{i,j}$ is generally 1~5 to denote users' preference degree for knowledge items, and this value shall be 0 if users don't rate.

(2) Seek the nearest neighbors. Calculate similarities among users in the user-knowledge item rating matrix by similarity calculation method, arrange according to similarities to target users, and take the top *k* users with great similarities as a set of the nearest neighbors. Similarity sim(*u*, *v*) among users can be calculated by Pearson similarity method with formula as follows:

$$\operatorname{sim}(u,v) = \frac{\sum_{i \in I_{uv}} (R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)}{\sum_{i \in I_{uv}} (R_{u,i} - \bar{R}_u)^2 \sum_{i \in I_{uv}} (R_{v,i} - \bar{R}_v)^2} \quad (1)$$

where, I_{uv} denotes a set of knowledge items rated jointly by user *u* and user *v*, $R_{u,i}$ and $R_{v,i}$ denote ratings of user *u* and user *v* for knowledge item *i* respectively, $\bar{R}_u$ and $\bar{R}_v$ denote mean ratings of user *u* and user *v* for rated knowledge items respectively.

(3) Generate a push data set. After getting a set of the nearest neighbors for target users, conduct weighted prediction of ratings of target users for unrated knowledge items according to similarities of the nearest neighbors to target users, select the top *N* knowledge items with the highest ratings to push to users, and thus generate a *Top-N* push set. Traditional target users' prediction of knowledge items can be calculated by the following formula:

$$P_{u,i} = \bar{R}_u + \frac{\sum_{v=1}^{n} (R_{v,i} - \bar{R}_v) \cdot \operatorname{sim}(u,v)}{\sum_{v=1}^{n} |\operatorname{sim}(u,v)|} \quad (2)$$

2.2. Analysis of Deficiencies in Traditional CF Algorithms

Traditional CF push algorithms have deficiencies mainly including the following three aspects [13]:

(1) The set of the nearest neighbors of target knowledge items shall be sought over the entire knowledge item space, which is of great time complexity and poor real-time response ability of system. Meanwhile, the proportion of user rating knowledge items to the total number of knowledge items is generally 1%~2% in a high-dimension matrix, making matrix sparseness a main influencing factor of push quality.

(2) Generally, similarity between two users is calculated based on all knowledge items rated jointly by these two users, while the factor that whether knowledge items are correlated is not considered. Meanwhile, a large number of users would rate hot items, which will affect calculation accuracy of user similarity, and this hasn't been improved in traditional algorithms.

(3) In reality, user interest changes frequently with time. However, traditional CF algorithms consider equally users' ratings of knowledge items at different time, resulting in inaccuracy of neighbor users and reduction of system push quality.

3. A COLLABORATIVE FILTERING ALGORITHM BASED ON HOT ITEM PUNISH MENT AND USER INTEREST CHANGE

3.1. Knowledge Item Clustering

At present, the numbers of users and knowledge items of large e-commerce websites increase rapidly. As similarities among knowledge items are relatively more stable than those among users, knowledge items can be clustered in a high-dimension matrix by K-Means algorithm [14, 15], and knowledge items with high similarities shall be clustered into one category. In this way, when seeking the nearest neighbors of target knowledge items, most of them can be found merely by looking up in the candidate set consisting of several clusters with high similarities to target knowledge items without having to look up in the entire knowledge item space. In addition, users have a variety of interests which are corresponding to different nearest neighbors. The nearest neighbors of users corresponding to different interests can be sought conveniently and accurately in corresponding knowledge item categories through clustering; meanwhile, only categories with new knowledge items shall be calculated if the rating matrix shall be updated, which improves system timeliness and scalability substantially [16].

Knowledge item clustering algorithm is basically as follows:

Input: user-knowledge item rating matrix $R_{m\times n}$ and the number of clusters *k*.

Output: *k* similar knowledge item clusters that meet the minimum variance.

Step 1: retrieve all *n* knowledge items from user rating database and denote as set $I=\{i_1,i_2,\cdots,i_n\}$.

Step 2: select arbitrarily *k* knowledge items, take ratings of these *k* knowledge items in user rating database as the initial clustering centers, and denote as set $M=\{m_1,m_2,\cdots,m_k\}$.

Step 3: initialize *k* initial categories $J_1,J_2,\cdots,J_k$ to be empty, and denote as set $J=\{J_1,J_2,\cdots,J_k\}$.

Step 4: for each knowledge item $i_i(i=1,2,\cdots,n)$ and clustering center m_i, calculate their similarities $\mathrm{sim}(i_i,m_t)=\max\{\mathrm{sim}(i_i,m_1),sim(i_i,m_2),\cdots,sim(i_i,m_k)\}$, and according to the maximum similarity, knowledge item i_i will be clustered into category $t(t=1,2,\cdots,k)$, *i.e.* $J_t=J_t\cup i_i$.

Step 5: recalculate each clustering center (mean clustering) with changes until being stable.

Step 6: after clustering of knowledge items, divide user-knowledge item rating matrix into *k* categories of matrixes, *i.e.* $R_{m\times n}=R_1\mathrm{U}R_2\mathrm{U}\cdots\mathrm{UR}_k$, where R_j is the $m\times s_j(s_j<n,j=1,2,\cdots,k)$-order rating matrix of category *j*, and one knowledge item may belong to multiple categories.

3.2. Rating Prediction of Unrated Knowledge Items

To make rating data denser after forming k clustering matrixes R_k, predict ratings of unrated knowledge items in each rating matrix and consider about the factor of user interest change with time unlike traditional collaborative filtering algorithms. Thus, exponential time function $f(t_{ui})$ based on user interest change is introduced to traditional collaborative filtering algorithms, *i.e.*:

$$f(t_{ui})=\frac{1}{1+e^{-t_{ui}}} \tag{3}$$

Formula (3) can be used to highlight weights of latest user interests and reduce weights of past preference, in order to realize more accurate real-time push. Where, t_{ui} denotes the interval between the time when user u becomes interested in knowledge item i and time of being used in system, and $f(t_{ui})$ denotes time weight of user u's rating of knowledge item i. It can be seen that the value of $f(t_{ui})$ increases with the increase of users' rating time of knowledge items, and the closer of rating time to current time, the larger value of exponential time function.

Rating prediction filling algorithm for unrated knowledge items is as follows:

Input: k user-knowledge item clustering matrixes R_k and exponential time function $f(t_{ui})$.

Output: k dense user-knowledge item rating matrixes R_k.

Step 1: in user-knowledge item rating matrix R_k after clustering, calculate similarities of each unrated target item i to other knowledge items in category of i by formula (4):

$$\mathrm{sim}(i,j)=\frac{\sum_{u\in U_{ij}}[(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)]}{\sqrt{\sum_{u\in U_{ij}}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U_{ij}}(R_{u,j}-\bar{R}_j)^2}} \tag{4}$$

where, U_{ij} denotes a set of users who have rated target knowledge item i and other knowledge item j in the same category, $\bar{R}_i$ and $\bar{R}_j$ denote mean ratings of knowledge items i and j by all users in clustering matrix R_k respectively, $R_{u,i}$ and $R_{u,j}$ denote ratings of knowledge items i and j by user u in the category respectively.

Step 2: set a similarity threshold value *SV* according to similarity of target knowledge item i and other items, take knowledge item set with similarities greater than threshold value *SV* as the nearest neighbor set of target knowledge items and denote as $M_i=\{I_{i1},I_{i2},\cdots,I_{im}\}(i=1,2,\cdots,n;m$ have values changing with i), make $i\notin M_i$, and I_{i1} has the highest similarity to i, followed by knowledge item I_{i2}, showing a decreasing trend.

Step 3: based on existing rating information of unrated item i, rating information of the nearest neighbor M_i and exponential time function $f(t_{uj})(j\in M_i)$, improve formula (2) in rating prediction:

$$P_{u,i} = \bar{R}_i + \frac{\sum_{j\in M_i} [(R_{u,j} - \bar{R}_i)\cdot \mathrm{sim}(i,j)\cdot f(t_{uj})]}{\sum_{j\in M_j} [\mathrm{sim}(i,j)\cdot f(t_{uj})]} \tag{5}$$

where, $P_{u,i}$ is the predictive rating of general user u for unrated knowledge item i, $\bar{R}_i$ and $\bar{R}_j$ denote mean ratings of knowledge item i and j by all users in clustering matrix R_k respectively, $\mathrm{sim}(i,j)$ denotes the similarity of knowledge item i to j, $P_{u,i}$ denotes rating of unrated knowledge item i by user u.

Step 4: after rating prediction of each unrated knowledge item in sparse rating matrix R_k, select the top N items with the biggest predictive values to form a push set, combine push sets generated by k matrixes, select the top N predictive values from the big set to form a final push rating set, fill these N predictive ratings in each sparse matrix R_k, and thus reduce data sparseness.

Step 5: when knowledge item i belongs to multiple categories, calculate rating of knowledge item i by user u in each category, and then take the mean value of predictive values of all categories as the final predictive rating $P_{u,i}$ of knowledge item i by user u.

3.3. User Similarity Calculation Based on Hot Item Punishment

Traditional user similarity calculation methods only consider about ratings of knowledge items evaluated jointly by two users, while fail to consider about the influence of the hot degrees of knowledge items on user similarity. For instance, if two users have bought Xinhua Dictionary, this doesn't mean that they have a similar interest because the majority of Chinese people have once bought Xinhua Dictionary. However, if two users have bought Supply Chain Management, it can be said that they have a similar interest because only talents who study on supply chain management would buy this book, indicating that two users have interest similarity if they have had common behaviors in less popular items [17-19]. Thus, hot item weight coefficient is introduced to reduce the influence of hot items on user similarity in the list of common interests of two users.

The improved user similarity calculation method is as follows:

Step 1: determine rating matrix R_k of knowledge item i to be pushed to target user u.

Step 2: introduce hot item weight coefficient $w_i = \frac{1}{\lg(1+N_i)} (i \in I_{uv})$, where N_i denotes the number of users who like item i rated jointly by two users among all users. This formula punishes the influence of hot items in the common interest list of user u and user v on their similarity, and reduces the similarity possibility of hot items to many items.

Step 3: to calculate the final user similarity, introduce hot item weight coefficient to Pearson correlation similarity measure formula to obtain a more accurate user similarity $\mathrm{sim}(u,v)$, and the calculation formula is as follows:

$$sim(u,v) = \frac{\sum_{i\in I_{uv}} [(R_{u,i} - \bar{R}_u)(R_{v,i} - \bar{R}_v)\cdot w_i]}{\sum_{i\in I_{uv}} (R_{u,i} - \bar{R}_u)^2 \cdot w_i \ \sum_{i\in I_{uv}} (R_{v,i} - \bar{R}_v)^2 \cdot w_i} \tag{6}$$

where, I_{uv} denotes a set of knowledge items rated jointly by user u and user v, $R_{u,i}$ and $R_{v,i}$ denote ratings of knowledge item i by user u and v respectively, $\bar{R}_u$ and $\bar{R}_v$ denote mean ratings of rated knowledge items by user u and v respectively.

After knowledge item clustering and rating prediction of unrated knowledge items, a user-knowledge item matrix that applies to collaborative filtering calculation can be obtained. In clustering matrix R_k of knowledge items to be pushed, calculate similarities $\mathrm{sim}(u,v)$ based on hot item weight between user u and other users according to formula (6), select the top N users with the highest similarities as the nearest neighbor set of target users, *i.e.* $M_u = \{U_{u1}, U_{u2}, \cdots, U_{un}\}(u = 1,2,\cdots,m, n$ have values changing with u), make $u \notin M_u$, and user U_1 is the most similar to u, followed by user U_2, showing a decreasing trend.

3.4. Calculation of Push Item Set

Traditional push algorithms only consider about ratings of push knowledge items by the nearest neighbors of users, and have weighted calculation of target users' ratings of push knowledge items according to similarities of the nearest neighbors to target users. However, these algorithms don't consider about constant changes of user interest preference, as knowledge items rated earlier play a smaller role in users' current interests while those rated later play a greater role in users' current interests. Here, an exponential function which decays with time is introduced to all knowledge items to be rated by users, giving greater weights to recent ratings and smaller weights to past ratings, *i.e.* introduce formula (3) to predictive value calculation formula, as shown below:

$$P_{u,t} = \bar{R}_u + \frac{\sum_{v\in M_u} [(R_{v,t} - \bar{R}_v)\cdot \mathrm{sim}(u,v)\cdot f(t_{vt})]}{\sum_{v\in M_u} [\mathrm{sim}(u,v)\cdot f(t_{vt})]} \tag{7}$$

where, $P_{u,t}$ is the predictive rating of target user u for target knowledge item t, $\bar{R}_u$ and $\bar{R}_v$ are mean ratings of user u and v for rated knowledge items respectively, $f(t_{vt})$ refers to the time weight of neighbor user $v(v \in M_u)$ for knowledge item i to be pushed, and $\text{sim}(u,v)$ is the similarity of user u to v.

According to rating values $P_{u,t}$ of knowledge items to be pushed, sort in a descending order, and select the top N items to form a $Top-N$ push item set and push to users.

3.5. Knowledge Push Algorithm Based on Hot Item Punishment and User Interest Change

Push algorithm based on hot item punishment and user interest change is as follows:

Input: user-knowledge item rating matrix $R_{m\times n}$, neighbor size n, cluster number k.

Output: $Top-N$ push item set of target user u.

Step 1: establish a user-knowledge item rating matrix $R_{m\times n}$, m is the number of users, n is the number of knowledge items, and $R_{u,i}$ denotes rating of knowledge item i by user u. Cluster all knowledge items by K-Means clustering algorithm to form k categories of matrixes.

Step 2: calculate n knowledge items with the highest similarities in the category of knowledge item i as the nearest neighbors $M_i = \{I_{i1}, I_{i2}, \cdots, I_{im}\}$, make $i \notin M_i$, and I_{i1} is the most similar to i, followed by knowledge item I_{i2}, showing a decreasing trend.

Step 3: introduce an exponential time function $f(t_{ui})$ based on user interest change to each rating matrix after clustering, and use unrated knowledge item filling algorithm for data prediction and filling according to ratings of the nearest neighbors of target users.

Step 4: introduce hot item weight coefficient w_i to each rating matrix after filling, calculate similarities among users in each category by formula (6) and form a user similarity matrix.

Step 5: calculate n users with the highest similarities to target user u in the category of target knowledge item i as the nearest neighbors of user u in terms of corresponding interests. $M_u = \{U_{u1}, U_{u2}, \cdots, U_{un}\}$ makes $u \notin M_u$, and U_i is the most similar to u, followed by user U_2, showing a decreasing trend.

Step 6: introduce an exponential time function $f(t_{ut})$ based on user interest change to each rating matrix after filling, and use algorithm in formula (7) to calculate the predictive value $P_{u,t}$ of target knowledge item based on ratings of target user u and its nearest neighbor M_u.

Step 7: sort in a descending order according to values of $P_{u,t}$, and select the top N items to generate a $Top-N$ push set.

4. SIMULATION EXPERIMENT RESULTS AND ANALYSIS

4.1. Data Set and Evaluation Standards

Data set applied in this experiment is provided by Movielens website , and this data set contains 100K public data established by Group Lens Research Group of U.S. Minnesota University. This data set contains rating records of 1,682 films by 943 users, and stipulates that registered users can only use this system normally after rating at least 20 films, and thus each user has rated at least 20 films. This experiment selects 56,770 records from the data set as experimental data set containing 500 users and 1,549 films, user rating data sparseness: 1-56,770/(500×1,549)=0.926, user rating range: 1~5. Greater ratings indicate that users are more interested in a film. Select 70% of all data as a training set, and the rest 30% as a test set. Mean Absolute Error (MAE) is used as the evaluation standard of this experiment, and is mainly used to calculate the absolute difference between users' actual rating and value predicted by push algorithm in test set, and the smaller MAE, the higher system push quality. To test whether an algorithm could track users' latest interests, the selected predictive knowledge items have been rated by users recently. Assume that uses' predictive knowledge item rating set is: $\{p_1, p_2, \cdots, p_N\}$ while users' actual knowledge item rating set is: $\{r_1, r_2, \cdots, r_N\}$, the MAE formula is:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{7}$$

4.2. Experimental Results and Analysis

(1) The experiment clusters knowledge items by K-Means algorithm. As specification of cluster number is very crucial and too large cluster number would result in insufficient real-time response ability of system while too small cluster number would result in many items in each cluster, this experiment sets different cluster numbers (KM) for clustering knowledge items into: 14, 16, 18, 20, 22 and 24 respectively. This experiment also detects the influence of cluster number changes on MAE, and experimental results are shown in Fig. (**1**).

As this experiment is based on data provided by Movielens website and there are generally 15~25 categories of films, cluster numbers lower than 14 or higher than 24 would result in relatively low system push efficiency and accuracy. Thus, cluster number k is specified to be 18 in this experiment.

(2) After clustering of knowledge items, introduce collaborative filtering algorithm based on knowledge items and user interest degrees to each category, and fill knowledge items with high predictive values in matrixes, in order to reduce sparseness. Sparseness degrees selected in this experiment are 0.942, 0.794 and 0.754 respectively, and experimental results are shown in Fig. (**2**).

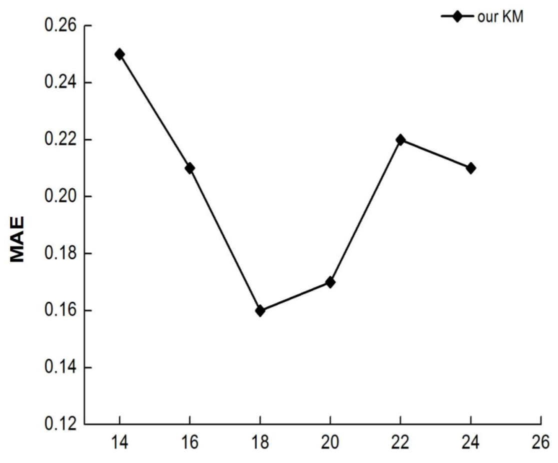

Fig. (1). The influence of different cluster numbers on push accuracy.

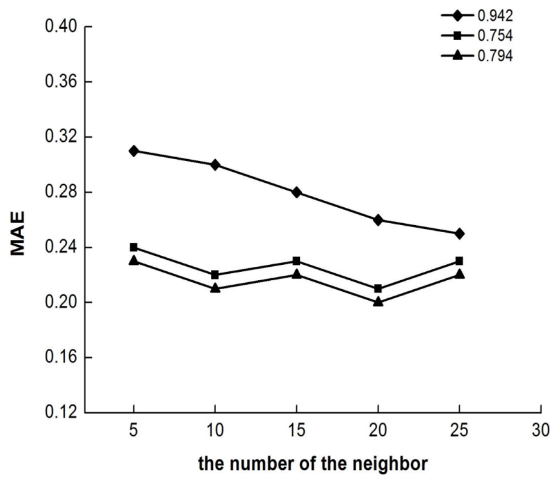

Fig. (2). Comparison of push effects among different sparseness degrees.

As can be known from Fig. (**2**), quite dense matrixes obtained by data filling will facilitate more accurate push of push system. However, it doesn't mean that lower sparseness would result in better push quality, and the minimum MAE and the best system push quality are obtained under 0.794 of sparseness. Thus, appropriate sparseness is vital for matrixes.

(3) To better calculate similarities among users under 0.794 of sparseness, introduce hot item punishment function to detect the influence on push system performance, and compare it with traditional cosine similarity and Pearson correlation coefficient method under the same conditions with results shown in Fig. (**3**).

According to Fig. (**3**), in user similarity calculation after introduction of hot item punishment function, MAEs under different numbers of neighbors are smaller than those obtained by traditional similarity calculation methods. Thus, user similarity calculation accuracy can be effectively improved after reduction of hot item weight, and the knowledge item push set obtained is also more effective.

(4) This experiment combines improved algorithms including knowledge item clustering, data filling based on user interest and hot item punishment etc, and introduces user interest degree function to final user push to further improve

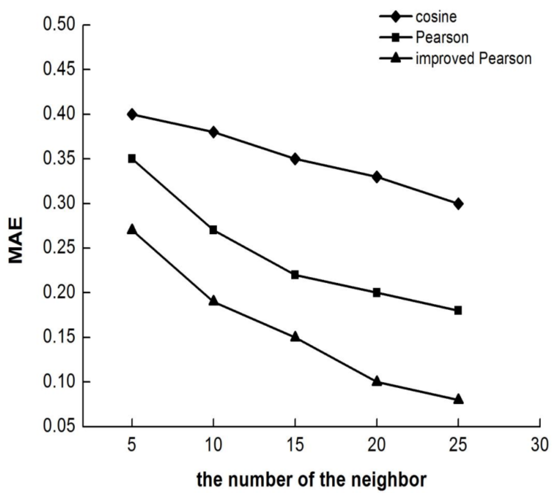

Fig. (3). Comparison of three similarity measure methods in terms of performance.

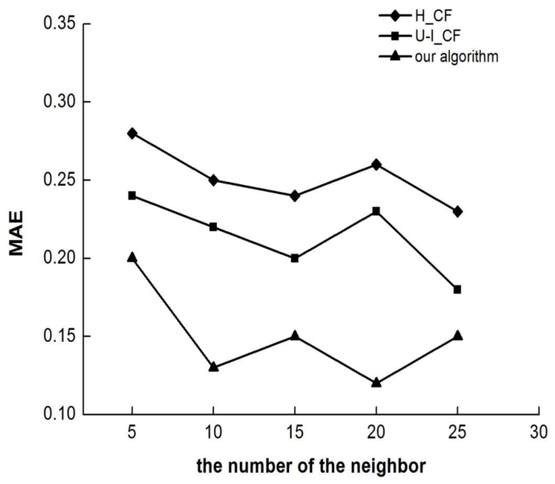

Fig. (4). Comparison of three push algorithms in terms of accuracy.

users' knowledge item push accuracy, and compares the performance between collaborative filtering algorithm based on knowledge item clustering (H_CF) and collaborative filtering algorithm based on user interest degree (U-I_CF), and experimental results are shown in Fig. (**4**).

The experimental results show that algorithm proposed in this paper has obviously higher accuracy than collaborative filtering algorithms in comparison under different numbers of neighbors. This is because this experiment introduces not only technologies including knowledge item clustering, user interest degree and sparse matrix filling etc, but also hot item punishment function, which alleviates effectively problems in traditional collaborative filtering algorithms and further improves system reliability and push accuracy.

CONCLUSION

With respect to problems in traditional collaborative filtering algorithms including failure in timely reflection of user interest change, possible influence of hot items on user similarity as well as data sparseness and scalability etc, this paper puts forward an improved algorithm mainly based on time weight function and hot item punishment function, introduces function based on time weight to knowledge item rating prediction, and applies hot item punishment function to user similarity calculation. This improved algorithm can make up for deficiencies of traditional algorithms effectively. Based on the application of time weight function and hot item punishment function, experimental comparison shows that algorithm push effects will be greatly improved with appropriate parameters and algorithm performance is also improved to a certain degree by the application of clustering and data filling of sparse matrixes. However, as different users have their respective interests currently, a parameter can be given to interest change law of each user to denote the speed of interest change for the sake of more accurate individualized knowledge push, while determination of parameters will be a key of future researches.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] K. J. Kim, and N. H. Ah, "A Recommender System Using GAK-means Clustering in An Online Shopping Market," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1200-1209, 2008.

[2] H. Maofeng, N. Weiwei, W. Jiajun and H. Xiao, "Clustering-oriented logarithmic spiral data disturbance method", *Journal of Computer*, vol. 11, pp. 2275-2282, 2012.

[3] B. Sarwar, J. Karypis, J. konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," In: *10th International Conference on World Wide Web, Hong Kong*, pp. 285-295, 2001.

[4] Y. Cao, and W. He, "mixed recommendation model based on user interest," *System Engineering*, vol. 27, no. 6, 2009, pp. 68-72.

[5] G. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zheng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press*, 2005, pp. 114-121.

[6] G. VozalisM, and K. G. Margaritis, "Applying SVD on item-based filtering," In: *Proceedings of the 5th International Conference on Intelligent System Design and Applications. Washington, D. C.: IEEE Computer Society Press*, 2005, pp. 464-469.

[7] Y. Ding, and X. Li, "Tmie weight collaborative filtering," In: *Proceedings of the 14th ACM International Conference on Information and Know ledge Management. New York: A CM Press*, 2005, pp. 485-492.

[8] Q. Wang, L. Yang, and D. Yang, "Collaborative filtering algorithm based on rating distribution of attributes faced user preference," *Journal of Systems Engineering*, vol. 25, no. 4, pp. 561-568, 2010.

[9] F. Liu, and S. Song, "Improved collaborative filtering algorithm," *Computer Engineering and Applications*, vol. 47, no. 8, pp. 72-75, 2011.

[10] X. Zheng, Z. Tang, and X. Cao, "Nonlinear gradual forgetting collaborative filtering algorithm capable of adapting to users' Drifting Interest," *Computer Aided Engineering*, no. 2, pp. 69-73, 2007.

[11] R. Hu, and Y. Lu, "A hybrid user and item-based collaborative filtering with smoothing on sparse data," In: *Proceedings of the 16th International Conference on Artificial Reality and Telexistence-Workshops. Washington, D. C.: IEEE Computer Society Press*, 2006, pp. 184-189.

[12] Z. Xiong, Q. Liu, Y. Zhang, W. Li, "Improved collaborative filtering algorithm based on knowledge item clustering," *Application Research of Computers*, no. 2, pp. 493-496, 2012.

[13] Y. Wang, B. Xu, X. Wei, and H. Ling, "Website navigation systems based on user accessing sequences clustering," *Systems Engineering-Theory & Practice*, vol. 30, no. 7, pp. 1305-1311, 2010.

[14] G. Linden, and H. Wang, "Amazon. com Recommendations: Item-to-item Collaborative Filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, 2003.

[15] P. Dan, and A.W. Moore, "X-means: extending k-means with efficient estimation of the number of clusters," In: *Proceedings of the 17th International Conference on Machine Learning. Morgan Kaufmann Publishers Inc.*, 2000.

[16] C. Xing, F. Gao, S. Zhan, and L. Zhou, "Collaborative filtering push algorithm capable of adapting to user interest change," *Journal of Computer Research and Development*, vol. 44, no. 2, pp. 296-300, 2007.

[17] L. Xiang, "*Recommendation System Practice*," Beijing: Posts & Telecom Press, 2012.

[18] L. Zhang, and M. Li, "Real preference gaussian mixture model for collaborative filtering," *Journal of Systems Engineering*, vol. 22, no. 6, pp. 613-619, 2007.

[19] A. Albadvi, and M. A. Shahbaze, "Hybrid recommendation technique based on product category attributes," *Expert System with Applications*, vol. 36, no. 9, pp. 11480-11488, 2009.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015