TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/11 pp318-327
Volume 18, Number 3, June 2013

TST: Threshold Based Similarity Transitivity Method in Collaborative Filtering with Cloud Computing

Feng Xie, Zhen Chen*, Hongfeng Xu, Xiwei Feng, and Qi Hou

Abstract: Collaborative filtering solves information overload problem by presenting personalized content to individual users based on their interests, which has been extensively applied in real-world recommender systems. As a class of simple but efficient collaborative filtering method, similarity based approaches make predictions by finding users with similar taste or items that have been similarly chosen. However, as the number of users or items grows rapidly, the traditional approach is suffering from the data sparsity problem. Inaccurate similarities derived from the sparse user-item associations would generate the inaccurate neighborhood for each user or item. Consequently, its poor recommendation drives us to propose a Threshold based Similarity Transitivity (TST) method in this paper. TST firstly filters out those inaccurate similarities by setting an intersection threshold and then replaces them with the transitivity similarity. Besides, the TST method is designed to be scalable with MapReduce framework based on cloud computing platform. We evaluate our algorithm on the public data set MovieLens and a real-world data set from AppChina (an Android application market) with several well-known metrics including precision, recall, coverage, and popularity. The experimental results demonstrate that TST copes well with the tradeoff between quality and quantity of similarity by setting an appropriate threshold. Moreover, we can experimentally find the optimal threshold which will be smaller as the data set becomes sparser. The experimental results also show that TST significantly outperforms the traditional approach even when the data becomes sparser.

Key words: cloud computing; recommender systems; big data; collaborative filtering; data mining; similarity transitivity; machine learning; mapReduce; android applications

1 Introduction

Information overload problem stemmed from the fact that the increasing amount of data (also called Big Data) makes users harder and take more time to find their preferred items. This situation has promoted the development of recommender systems[1,2], which is one of the most promising information filtering technologies that match users with the most appropriate items by learning about their preferences.

Different from content based recommender approaches[3,4], Collaborative Filtering (CF)[5-7] is domain free, which can address data aspects that are often elusive and difficult to profile using content filtering. Nowadays, CF has been successfully implemented to recommend movies[8,9], TV shows[10,11], and Web pages[12] relying only on past

• Feng Xie and Xiwei Feng are with Department of Automation, Research Institute of Information Technology and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China. E-mail: xief10@mails.tsinghua.edu.cn; fxw10@mails.tsinghua.edu.cn.
• Zhen Chen is with Research Institute of Information Technology and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China. E-mail: zhenchen@tsinghua.edu.cn.
• Hongfeng Xu is with Department of Computer Science and Technologies and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China. E-mail: xhf10@mails.tsinghua.edu.cn.
• Qi Hou is with Department of Electronic Engineering and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China. E-mail: houq10@mails.tsinghua.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2013-4-15; revised: 2013-5-15; accepted: 2013-5-15

user behaviors, for example, previous transactions or item ratings.

Generally, CF can be classified into similarity based methods[13,14] and model based methods[15-17]. Due to its simple algorithm and good interpretation for recommendations compared to model based methods, similarity based methods have been widely applied, which predict a user's interest for an item based on the weighted combination of ratings of the similar users on the same item or the user on the similar items. The similar users are other users who tend to give similar rating on the same item, while the similar items are the items that tend to get similar rating from the same user. Therefore, the recommendation quality would mainly depend on the accuracy of similarity measurement for users and items.

However, as the system scale becomes large with millions of users and items recently, similarity based CF methods are facing more and more serious data sparsity problem[18-20]. The sparse data depresses the accuracy of similarity measurement and poor recommendations may generate through these inaccurate similarities[7]. Besides, such methods tend to recommend popular items which are usually chosen by similar users or are similar to those previously chosen by users, thus, the recommendation diversity would be low. Furthermore, the computational complexity is quadratic in the number of users or items, therefore, similarity based methods also suffers from the limitation of system scalability.

Recently, many approaches have been proposed to alleviate the data sparsity problem. The most representative approach is the one using dimensionality reduction techniques, such as Singular Value Decomposition (SVD)[21] and Principle Component Analysis (PCA)[22], to remove unrepresentative or insignificant users or items to reduce the dimensionalities of the user-item matrix, then, the similarity between two users is measured by the representation of the users in the reduced space. This approach can deal with scalability problem and quickly generate good quality recommendations especially for the incremental SVD CF algorithm[23] , but useful information may be lost after the dimensionality reduction and recommendation quality may be degraded finally[13,18]. Moreover, clustering CF algorithms[24,25] can address the scalability problem by firstly clustering users into different groups and then choosing similar users for recommendation only from each group not the entire set of users, but there are still tradeoffs between scalability and prediction performance. Several graph-based recommendation algorithms[26-28] have been demonstrated to improve diversity of recommendation, but the algorithmic computational complexity is very sensitive to the statistic properties of the data sets[29].

Since collaborative filtering has been extensively applied in real-world systems, it is meaningful to find other ways to improve its algorithmic performance. Therefore, we propose a Threshold based Similarity Transitivity (TST) method, in which the similarity between two users is not directly computed if their intersection is less than the set threshold and will be replaced by the transitivity similarity. Figure 1 shows an illustration of the user intersection network, where there is only one commonly selected item between users B and C, obviously, the similarity measured directly from the insufficient intersection might be inaccurate. An alternative method is to derive the similarity between users B and C from the similarity between users A and B, and the one between users A and C with similarity transitivity. Statistically speaking, it is unreliable to identify whether two users are similar or not when less intersection between them.

Therefore, we can improve the quality[1] of similarities by setting a proper intersection threshold, and increase the similarity quantity benefiting from similarity transitivity. The experimental results on the public data set and the real-world data set show that the TST method is much more accurate and provides more diverse recommendations especially on the sparser data set. Moreover, the TST method is developed to be scalable with MapReduce[30] , which is a programming paradigm that comes with a framework to provide to the programmers an easy way for parallel and distributed computing.

2 TST method

A recommender system always comprises users and items which are denoted as the user set $U = \{u_1, u_2, \cdots, u_m\}$ and the item set $I = \{i_1, i_2, \cdots, i_n\}$, and the user-item associations can be fully described by an adjacent matrix $A = \{a_{ij}\} \in R^{m\times n}$, where $a_{ij} = 1$ if user u_i has chosen item i_j , otherwise $a_{ij} = 0$. The system sparsity level is the proportion

[1]Quality is defined as the accuracy of the similarity, while the subsequent quantity describes the number of similarities.

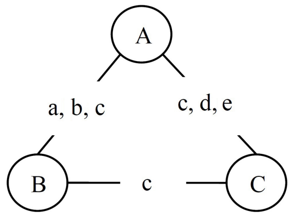


Fig. 1 An example of the user intersection network (Letters on the line are the item labels that are commonly selected by both ends of users, for example, user A and user C have three commonly selected items, c, d, e).

of zero elements in the matrix and the bigger the value is, the sparser the system will be. More specially, the sparsity level[18] can be expressed as Eq. (1) using aforementioned notations:

$$\text{sparsitylevel} = 1 - \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}}{m \cdot n} \tag{1}$$

2.1 Traditional user based CF

The traditional user based CF directly measures the similarity between users u_i and u_j using the well-known cosine distance method[2,31] :

$$s_{ij}^{1} = \frac{\sum_{k=1}^{n} a_{ik} \cdot a_{jk}}{\sqrt{\sum_{k=1}^{n} a_{ik} \cdot \sum_{k=1}^{n} a_{jk}}} \tag{2}$$

When more commonly selected items between two users and fewer items are chosen by each user, the value of s_{ik}^{1} would be bigger and both of users are more similar. After all other users' similarities to certain user u_i are calculated, the prediction p_{ij} of this user on her unselected item i_j (i.e., $a_{ij} = 0$) is formulated as[32] :

$$p_{ij} = \frac{\sum_{k=1}^{m} s_{ik}^{1} a_{kj}}{\sum_{k=1}^{m} s_{ik}^{1}} \tag{3}$$

The recommendations for user u_i are those items which have high predictions.

2.2 TST method

As described above, the similarity directly calculated using Eq. (2) is inaccurate when the intersection is little due to sparse data. We therefore set an intersection threshold t in TST method. If the intersection between two users u_i and u_j is not less than the threshold, the similarity between them is computed directly using Eq. (2); otherwise, the similarity is formulated as:

$$s_{ij}^{2} = \frac{1}{|U_i \bigcap U_j|} \sum_{k \in U_i \bigcap U_j} \left(s_{ik}^{1} \cdot \frac{s_{kj}^{1}}{\sum_{o \in U_k} s_{ko}^{1}} \right) \tag{4}$$

where U_i is the set of users who share at least t commonly selected items with user u_i , and whose similarities to the given user are calculated directly using Eq. (2). U_j is defined similarly. Therefore, $|U_i \bigcap U_j|$ is the number of users who simultaneously share at least t commonly selected items with users u_i and u_j. If $|U_i \bigcap U_j|$ equals to zero, the similarity between users u_i and u_j cannot be derived from similarity transitivity and zero will be replaced in this situation. Therefore, the unified expression for similarity measurement in TST can be depicted as:

$$s_{ij} = \begin{cases} s_{ij}^{1}, & j \in U_i; \\ s_{ij}^{2}, & j \notin U_i \text{ and } U_i \bigcap U_j \neq \Phi; \\ 0, & j \notin U_i \text{ and } U_i \bigcap U_j = \Phi \end{cases} \tag{5}$$

The prediction process is similar to the one introduced in traditional similarity based CF, thus we evaluate the predicted score p_{ij} for the user u_i on her unselected item i_j is given as:

$$p_{ij} = \frac{\sum_{k=1}^{m} s_{ik} a_{kj}}{\sum_{k=1}^{m} s_{ik}} \tag{6}$$

Obviously, the traditional user based CF and TST have similar process, but the latter filters out inaccurate similarities to enhance similarity quality and increases similarity quantity through similarity transitivity (see Eq. (4)).

2.3 Scalable TST method

As each user has to be compared with every other user for similarity measurement, the complexity of the user based CF approach is quadratic in the number of users. In order to improve the scalability of TST, it needs to modify TST to be a parallel algorithm so that the runtime of the similarity computation process can approximatively speedup proportional to the number of machines in the cluster. First, we denote A_i as the rating vector of user u_i , where A is the adjacency matrix of the user-item associations.

$$A_i = (a_{i1}, a_{i2}, \cdots, a_{in}), i = 1, 2, \cdots, m \tag{7}$$

Next, the function is defined to compute the sum of all elements from a vector:

$$\text{num}(A_i) = \sum_{k=1}^{n} a_{ik} \tag{8}$$

Followed by the function of the dot product of two vectors:

$$\text{dot}(A_i, A_j) = A_i \cdot A_j \tag{9}$$

Finally, if the intersection between users u_i and u_j is not less than the threshold, namely, $\text{dot}(A_i, A_j) \geqslant t$, the function $\text{sim}^1()$ will be used to compute the similarity between them:

$$\begin{aligned} s_{ij}^1 &= \text{sim}^1(\text{num}(A_i), \text{num}(A_j), \text{dot}(A_i, A_j)) \\ &= \frac{\text{dot}(A_i, A_j)}{\sqrt{\text{num}(A_i) \cdot \text{num}(A_j)}} \end{aligned} \tag{10}$$

Inspired by these functions, we partition A by its rows (the users) and store it in the Hadoop Distributed File System (HDFS). Each map function reads a row-pair out of $\frac{m(m-1)}{2}$ pairs which is named as one job, computes the similarity with aforementioned functions, and returns the result, then the reduce function simply has to aggregate the results from different computation nodes. It generates an initiate similarity matrix $S^1 = \{s_{ij}^1\} \in R^{m \times m}$ after all jobs are finished, where s_{ij}^1 equals to zero when the intersection between users u_i and u_j is less than the threshold, otherwise it is computed with Eq. (10).

Actually, the similarity transitivity process in TST can similarly be designed for MapReduce framework. We define S_i^1 to be the similarity vector of user u_i , where S^1 is the initial similarity matrix.

$$S_i^1 = (s_{i1}^1, s_{i2}^1, \cdots, s_{im}^1), i = 1, 2, \cdots, m \tag{11}$$

The sum of the similarities for each user to other users in S^1 can be computed using Eq. (8):

$$\text{num}(S_i^1) = \sum_{k=1}^{m} s_{ik}^1 \tag{12}$$

We preprocess vector S_i^1 to be $\widehat{S}_i^1$, where:

$$\widehat{S}_{ik}^1 = \frac{S_{ik}^1}{\text{num}(S_k^1)}, k = 1, 2, \cdots, m \tag{13}$$

Furthermore, the function intersection() is used to count the number of users whose initial similarities to the input users are non-zeros, simultaneously:

$$\begin{aligned} &\text{intersection}(S_i^1, \widehat{S}_j^1) \\ &= \text{size}\{S_{ik}^1 \neq 0, \widehat{S}_{jk}^1 \neq 0 | k = 1, 2, \cdots, m\} \end{aligned} \tag{14}$$

Then, similarity derived from the similarity transitivity process is expressed as:

$$s_{ij}^2 = \frac{\text{dot}(S_i^1, \widehat{S}_j^1)}{\text{intersection}(S_i^1, \widehat{S}_j^1)} \tag{15}$$

Therefore, if we define the row-pair for map function in this process to be two vectors S_i^1 and $\widehat{S}_j^1$, where $s_{ij}^1 = 0$. The map and reduce functions in the initial similarity computation process can be reused.

3 Evaluation

Experiments are conducted on the cloud computing platform, which is based on Apache Hadoop and Mahout. There are about 30 physical servers, and the storage reaches about 40 TB, which is also used in Internet forensic analysis[33]. The map and reduce functions has been introduced in the previous section. In this section, the data sets and metrics are described. The experimental results are shown as follows.

3.1 Data sets

The publicly available data set MovieLens[2] and a real-world data set from AppChina[3] are used. The former consists of 100 000 ratings of 943 users on 1682 movie items. Each rating is an integer values ranging from 1 to 5. Every user has rated more than 20 movies. However, in the real-world situation, users are usually reluctant or forgetful to give ratings after buying a cloth, seeing a movie or listening to a piece of music. To address this, a more commonly used way is to infer whether an item (i.e. a cloth, a movie, or a piece of music) is chosen by a user or not from the user's abundantly implicit records. More specially, a movie is set to be chosen by a user only if the given rating is not less than 3. Therefore, the MovieLens data set is preprocessed and 85 250 user-item associations remain.

The other data set is obtained from AppChina, a company aims to make users download Android applications conveniently through its Android software installation tool. Once a user runs this assistance tool, his/her operations (i.e., installation, upgrade and deletion) on applications are recorded. We collect about 1 TB logs (zipped, and the original files above 10 TB) during the three-month period from May 1st, 2012 to July 31st, 2012. Totally there are about 200 K active users and 10 K Android applications. Then a model is built to infer whether an application is chosen by a user or not (The online A/B test results of the most

[2](http://www.grouplens.org/)
[3](http://www.appchina.com/)

Table 1 Statistical properties of MovieLens and AppChina data sets.

	MovieLens	AppChina
#Users	943	2395
#Items	1682	2486
#Total associations	85 250	95 803
#Per user	90	40
#Per item	51	39
#Sparsity level	94.6%	98.4%

relevant application recommendation on the website, appchina.com, verify the efficiency of this model, which will be discussed in another paper).

Finally, we extract a section of data set including 95 803 user-item associations with 2395 users and 2486 application items. The statistical properties of these two data sets are summarized in Table 1. #Per user is the average number of items chosen by per user, while #Per item represents that per item would be chosen by the average number of users. #Sparsity level is defined in Eq. (1).

3.2 Evaluation metrics and methodologies

Both of the data sets are randomly divided into the training set with 80% of the data, and the test set with the remaining 20% of the data. The algorithmic accuracy is measured by two well-known metrics, precision and recall, while the coverage and popularity are used as the metrics for the measurement of algorithmic diversity. For a top-N recommendation, each user will get N most relevant items. The most relevant items are those which have not been chosen by him/her before and are predicted to be rated high by the given user. If a user-item association in the test set is included in the recommendations, there is a *hit*. Then, the overall precision[34-36] is defined as the ratio of all *hits* to recommendations:

$$\text{precision} = \frac{\#\text{hits}}{\#\text{recommendations}} = \frac{\#\text{hits}}{N\#\text{Users}} \tag{16}$$

The overall recall[34-36] represents how many associations in test set can be recommended to users, which is the ratio of all hits to all user-item associations for testing:

$$\text{recall} = \frac{\#\text{hits}}{\#\text{Test associations}} \tag{17}$$

Precision and recall are usually used to evaluate the algorithmic accuracy in area of information retrieval. The high precision and recall values are expected.

The coverage[37,38] is the overall ratio of recommended individual items to all items in the system, which corresponds to the percentage of items the system is able to recommend.

$$\text{coverage} = \frac{\#\text{Recommended items}}{\#\text{Items}} \tag{18}$$

It is meaningful to measure algorithmic capability to recommend unpopular items. Thus, the overall popularity[37,38] represents the average #Per item of recommended items.

The higher coverage and lower popularity values show that the algorithm can recommend diverse items to users and these items are surprising. These two metrics can be used to evaluate the algorithmic diversity.

3.3 Experimental results

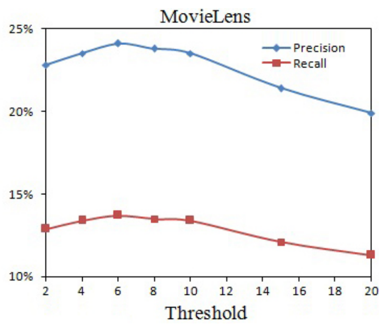
In order to analyze the effectiveness of the proposed TST method, extensive evaluation experiments have been conducted on two data sets and comparision has been made with the state-of-art solution user based CF in four quality metrics.

3.3.1 Influence of threshold

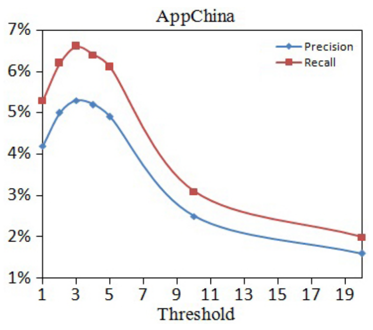
Since different thresholds will produce different qualities and quantities of similarity, it is valuable to experimentally find the optimal one. Intuitively, if the threshold is set to be low, the inaccurate similarities cannot be efficiently filtered out, therefore, the algorithmic accuracy will be affected by those low-quality ones. Besides, high threshold will result in fewer similarities for the similarity transitivity process, thus, low performance will derive from low quantity of similarities. In the real-world situation, 10 quality recommendations to users are enough. Therefore we use top-10 recommendation to test the influence of threshold.

The experimental results in Fig. 2 show that an appropriate threshold (i.e. 6 for MovieLens and 3 for AppChina) can be found to obtain the highest accuracy. It also infers that conservative and radical thresholds are not effective in TST. Besides, the optimal threshold value will be lower as the data set becomes sparser, for example, the optimal threshold is 3 for AppChina, while it is 6 for MovieLens.

Obviously, when the threshold is set to 6 for MovieLens, TST does not achieve the best performance in coverage and popularity (see Table 2). Analogously, TST gets lower coverage and higher popularity with threshold 3 than some of the other thresholds (i.e., threshold 4). Although the property derived from the precision and recall measurements does not appear



(a) Precision and recall of TST on MovieLens



(b) Precision and recall of TST on AppChina

Fig. 2 Recommend 10 items to each user with TST.

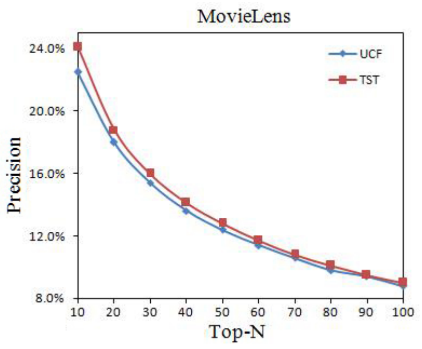


Fig. 3 The precision comparison on MovieLens data.

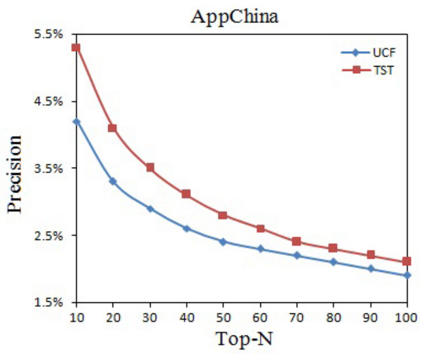


Fig. 4 The precision comparison on AppChina data set.

Table 2 Summarization of the coverage and popularity of TST on two data sets with variable thresholds.

MovieLens			AppChina		
threshold	Coverage /%	Popularity	threshold	Coverage/%	Popularity
2	5.3	5.04	1	11.5	4.38
4	7.2	5.01	2	22.1	4.22
6	9.8	4.97	3	37.5	4.03
8	12.3	4.94	4	62.2	3.82
10	14.0	4.92	5	58.1	3.73
15	14.4	4.84	10	46.0	3.19
20	14.8	4.76	20	12.9	3.19

in the coverage and popularity measurements, TST still has comparable performance when the optimal threshold for the highest accuracy (i.e. 6 for MovieLens and 3 for AppChina). Moreover, high coverage and low popularity are meaningful only when high accuracy is achieved. Actually, there is a tradeoff between accuracy and diversity. Therefore, the subsequent experiments are based on the optimal threshold in TST.

3.3.2 Comparison with user based CF

As mentioned above, we set the threshold as 6 on MovieLens and 3 on AppChina. Extensive experiments are conducted to compare the performance of TST to User based CF (UCF) in accuracy and diversity metrics with the number of items recommended to each user varying from 10 to 100.

Figures 3 and 4 illustrate that TST outperforms UCF in precision on both of data sets. The accuracy of prediction affects the rank of recommendation list for each user. The precision will be sensitive to the rank when few items are recommended to each user, intuitively, high precision is achieved if all items related to a user in the test set are ranked high. On the contrary, a long recommendation list may contain all items which appear in user's test list, although they are ranked behind. In this situation, the algorithmic precision may not be affected by the accuracy of prediction. This is the reason why the difference tends to be inconspicuous as top-N increases. Moreover, the precision degrades with the increasing top-N values because the number of recommendations increases much faster than the number of *hits* does in Eq. (16).

Figures 5 and 6 show that TST also outperforms UCF in recall on both of data sets. Different from precision, the recall grows as top-N becomes big. This is because of that more *hits* will derive from bigger top-N while the test associations are static in Eq. (17).

TST can recommend much more individual items to users with higher coverage than UCF in Figs. 7 and 8. TST provides more diverse recommendation. Obviously, the coverage increases when more items recommended to each user just as the illustratration in aforementioned figures.

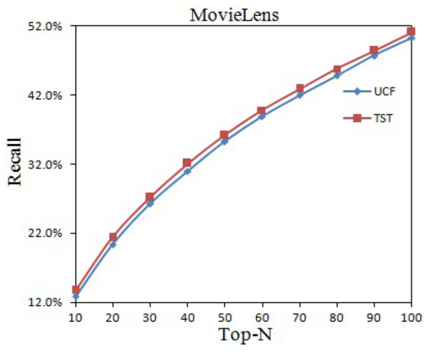


Fig. 5 The recall comparison on MovieLens data set.

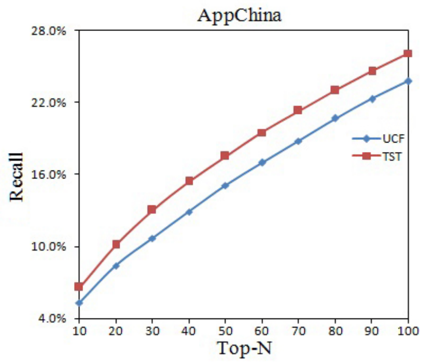


Fig. 6 The recall comparison on AppChina data set.

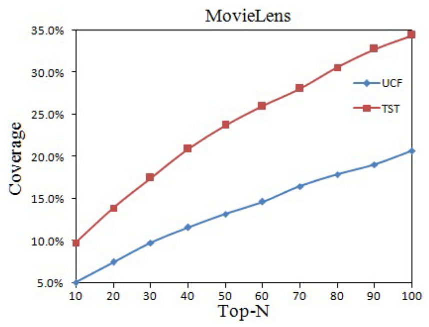


Fig. 7 The coverage comparison on MovieLens data set.

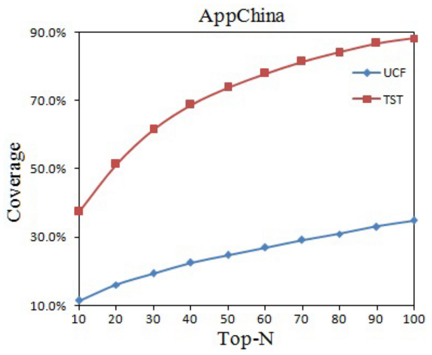


Fig. 8 The coverage comparison on AppChina data set.

Figures 9 and 10 illustrate that the average popularity of items recommended by TST is lower than the one of UCF. It infers that TST has the capability to recommend

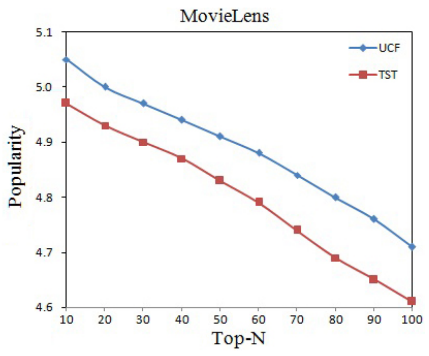


Fig. 9 The popularity comparison on MovieLens data set.

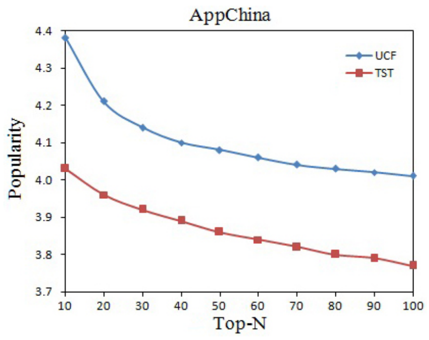


Fig. 10 The popularity comparison on AppChina data set.

surprising items to users. It also suggests that TST improves the diversity of recommendation. The extensive experiments introduced above conclude that TST outperforms UCF in accuracy and diversity. Especially, the improvement is more notable on AppChina which is sparser. Therefore, it can also conclude that TST can cope with data sparsity problem to a certain extent. Besides, it is worthwhile to note that high diversity is meaningful and expected only when the algorithmic accuracy is high. In real-world case, there always exists a tradeoff between accuracy and diversity.

4 Conclusion and Future Work

Similarity based collaborative filtering makes recommendation by finding similar users or items with similarity computing, which possesses simple and efficient characteristics. But directly using insufficient intersection between two users to compute similarity will result in inaccurate result. In this paper, we proposed a threshold based similarity transitivity method to filter out those low-quality similarities and replaced them with transitivity similarities to increase similarity quantity. TST is evaluated in the well-known data set MovieLens and an Android application market AppChina data set. The significant performance improvement on two data sets demonstrates that the TST method can well balance the tradeoff between quality and quantity of similarity. Moreover, the TST method has been implemented with MapReduce which enhances the algorithmic scalability. In the future work, it would be attractive to theoretically find the optimal threshold value, although we have experimentally inferred that it must exist. Intuitively, the optimal threshold will become smaller as the data set becomes sparser.

Acknowledgements

The authors would like to thank Prof. Jun Li of NSLAB from RIIT for his careful guidance about the paper's structure and writing. We are also grateful to Prof. Junwei Cao from RIIT, Dr. Zihong Huang and Xiaoping Feng from Electronic Engineering Department for their help.

This work is supported by Ministry of Science and Technology of China under the National Key Basic Research and Development (973) Program of China (Nos. 2012CB315801 and 2011CB302805), the National Natural Science Foundation of China A3 Program (No. 61161140320) and the National Natural Science Foundation of China (No. 61233016). This work is also supported by Intel Research Council with the title of Security Vulnerability Analysis based on Cloud Platform with Intel IA Architecture.

References

[1] P. Resnick and H. R. Varian, Recommender systems, *Communications of the ACM*, vol. 40, no. 3, pp. 56-58, 1997.

[2] G. Adomavicius and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, 2005.

[3] M. Balabanovic and Y. Shoham, Fab: content-based, collaborative recommendation, *Communications of the ACM*, vol. 40, no. 3, pp. 66-72, 1997.

[4] M. J. Pazzani and D. Billsus, Content-based recommendation systems, *The Adaptive Web*. Heidelberg: Springer Berlin, 2007, pp. 325-341.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, vol. 35, no. 12, pp. 61-70, 1992.

[6] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, Collaborative filtering recommender systems, *The Adaptive Web*. Heidelberg: Springer Berlin, 2007, pp. 291-324.

[7] X. Su and T. M. Khoshgoftaar, A survey of collaborative filtering techniques, *Advances in Artificial Intelligence*, vo. 2009, pp. 1-19.

[8] C. Christakou, S. Vrettos, and A. Stafylopatis, A hybrid movie recommender system based on neural networks, *International Journal on Artificial Intelligence Tools*, vol. 16, no. 5, pp. 771-792, 2007.

[9] B. Yang, T. Mei, X. S. Hua, L. Yang, S. Q. Yang, and M. Li, Online video recommendation based on multimodal fusion and relevance feedback. in *Proceedings of the 6th ACM international conference on Image and video retrieval*, Amsterdam, Netherlands, 2007, pp. 73-80.

[10] M. Van Setten, M. Veenstra, A. Nijholt, and B. van Dijk, Prediction strategies in a TV recommender system-method and experiments. in *Proceedings of the Second IADIS International Conference WWW/Internet*, Algarve, Portugal, 2003, pp. 203-210.

[11] J. Park, S. J. Lee, S. J. Lee, K. Kim, B. S. Chung, and Y. K. Lee, Online video recommendation through tag-cloud aggregation, *IEEE MultiMedia*, vol. 18, no. 1, pp. 78-86, 2011.

[12] M. Balabanovic, Exploring versus exploiting when learning user models for text recommendation, *User Modeling and User-Adapted Interaction*, vol. 8, no. 1-2, pp. 71-102, 1998.

[13] G. Linden, B. Smith, and J. York, Amazon. com recommendations: Item-to-item collaborative filtering, *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, 2003.

[14] T. Hofmann, Latent semantic models for collaborative filtering, *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89-115, 2004.

[15] K. Miyahara, and M. J. Pazzani, Collaborative filtering with the simple Bayesian classifier, *PRICAI 2000 Topics in Artificial Intelligence*. Heidelberg: Springer Berlin, 2000, pp. 679-689.

[16] X. Su and T. M. Khoshgoftaar, Collaborative filtering for multi-class data using belief nets algorithms, in *Proceedings of 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, Washington DC, USA, 2006, pp. 497-504.

[17] G. Shani, D. Heckerman, and R. I. Brafman, An MDP-based recommender system, *Journal of Machine Learning Research*, vol. 6, no. 2, pp. 1265-1295, 2006.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, Analysis of recommendation algorithms for e-commerce, in *Proceedings of the 2nd ACM conference on Electronic commerce*, Minneapolis, MN, USA, 2000, pp. 158-167.

[19] H. Ma, T. C. Zhou, M. R. Lyu, and I. King, Improving recommender systems by incorporating social contextual information, *ACM Transactions on Information Systems (TOIS)*, vol. 29, no. 2, pp. 1-23, 2011.

[20] F. Xie, M. Xu, and Z.Chen, RBRA: A simple and efficient rating-based recommender algorithm to cope with sparsity in recommender systems, in *Procedings of 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Fukuoka, Japan, 2012, pp. 306-311.

[21] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, Application of dimensionality reduction in recommender systems-a case study, in *Proceedings of 6th SIGKDD Workshop on Web Mining and Web Usage Analysis (WebKDD'00)*, Boston, MA, USA, 2000.

[22] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Eigentaste: A constant time collaborative filtering algorithm, *Information Retrieval*, vol. 4, no. 2, pp. 133-151, 2001.

[23] Sarwar B, Karypis G, Konstan J, and J. Riedl, Incremental singular value decomposition algorithms for highly scalable recommender systems, in *Procedings of Fifth International Conference on Computer and Information Science*, 2002.

[24] L. H. Ungar, and D. P. Foster, Clustering methods for collaborative filtering, in *Procedings of AAAI Workshop on Recommendation Systems*, Madison, isconsin, USA, 1998.

[25] S. H. S. Chee, J. Han, and K. Wang, Rectree: An efficient collaborative filtering method, *Data Warehousing and Knowledge Discovery, Springer Berlin Heidelberg*, pp. 141-151, 2001.

[26] Z. Huang, D. Zeng, and H. Chen, A comparative study of recommendation algorithms in e-commerce applications, *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 68-78, 2007.

[27] T. Zhou, J. Ren, M. Medo, and Y. C. Zhang, Bipartite network projection and personal recommendation, *Physical Review E*, vol. 76, no. 4, 046115, 2007.

[28] X. Li, and H. Chen, Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach, *Decision Support Systems*, vol. 54, no. 2, pp. 880-890, 2012.

[29] J. G. Liu, T. Zhou, H. A. Che, B. H. Wang, and Y. C. Zhang, Effects of high-order correlations on personalized recommendations for bipartite networks, *Physica A: Statistical Mechanics and its Applications*, vol. 389, no.4, pp. 881-886, 2010.

[30] J. Dean, and S. Ghemawat, MapReduce: simplified data processing on large clusters, *Communications of the ACM*, vol. 51, no. 1, pp. 107-113, 2008.

[31] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl, Evaluating collaborative filtering recommender systems, *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5-53, 2004.

[32] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, An algorithmic framework for performing collaborative filtering, in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, Berkeley, CA, USA, 1999, pp. 230-237.

[33] Z. Chen, F. Y. Han, J. W. Cao, X. Jiang, and S. Chen, Cloud computing-based forensic analysis for collaborative network security management system, *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 40-50, 2013.

[34] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, *The Journal of Machine Learning Research*, vol. 10, pp. 2935-2962, 2009.

[35] H. Steck, Training and testing of recommender systems on data missing not at random, in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington DC, USA, 2010, pp. 713-722.

[36] H. Steck, Item popularity and recommendation accuracy, in *Proceedings of the fifth ACM conference on Recommender systems*, Chicago, USA, 2011, pp. 125-132.

[37] P. Castells, S. Vargas, and J. Wang, Novelty and diversity metrics for recommender systems: choice, discovery and relevance, in *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*, Chicago, USA, 2011, pp. 29-37.

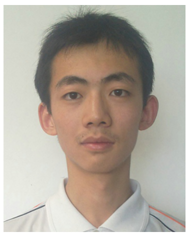
[38] G. Adomavicius, and Y. O. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 896-911, 2012.

Zhen Chen is an associate professor in Research Institute of Information Technology in Tsinghua University. He received his BEng and Ph.D. degrees from Xidian University in 1998 and 2004. He once worked as postdoctoral researcher in Network Institute of Department of Computer Science in Tsinghua University during 2004 to 2006. He is also a visiting scholar in UC Berkeley ICSI in 2006. He joined Research Institute of Information Technology in Tsinghua University since 2006. His research interests include network architecture, computer security, and data analysis. He has published around 80 academic papers.

Feng Xie now a PhD candidate in Department of Automation at Tsinghua University. He received his bachelor degree in School of Electronic and Information Engineering from Beijing Jiaotong University in 2010. His research interests include big data, recommender system and social network.

Hongfeng Xu is now a master candidate in Department of Computer Science and Technology, Tsinghua University. He received his bachelor degree in Computer Science in 2010 from Beihang University. His research interests include recommender system and content centric network.

Xiwei Feng is an undergraduate student working in Department of Autiomation at Tsinghua University. His research interests include network security, data mining and machine learning.



Qi Hou is an undergraduate student working in Department of Electronic Engineering at Tsinghua University. His research interests include network security and data analysis.