A New Parallel Item-Based Collaborative Filtering Algorithm Based on Hadoop

Qun Liu, Xiaobing Li*

School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China.

* Corresponding author. Tel: +8618875081258; email: chrislee8610@foxmail.com.

Manuscript submitted June 30, 2014; accepted October 8, 2014.
doi: 10.17706/jsw.10.4.416-426

Abstract: With the appearance of big data's era, some problems caused in recommendation systems are needed to solve immediately. So it is very useful to design parallel recommendation algorithms. An improved parallel item-based collaborative filtering (IP_Item-basedCF) algorithm based on Hadoop is proposed in this paper. In order to consider the influence of user's activity, a new parameter called IUF is introduced that can give the active users soft punishment. And the user's rating is also considered in prediction model. Finally, we evaluate the performance of our approach by using two real datasets – MovieLens and Douban. The experimental results show that this new parallel algorithm outperforms the algorithms existed and has a good scalability and speedup.

Key words: Collaborative filtering, Hadoop, co-occurrence matrix, parallel algorithm.

1. Introduction

Recommendation system is the system that satisfies the user's interests and provides personalized recommendations for products suiting a user's taste [1]. The emergence of recommendation system aims to win-win between information producers and consumers.

In the field of data analysis, the MapReduce paradigm and its open-source implementation Hadoop have also been widely adopted in industry and academia [2]. It is suitable to process big data. Though many collaborative filtering (CF) algorithms [3]-[8] used by recommendation system obtain the excellent predict accuracy, their training models still lead to high computational complexity, which are unfit for applying on the large-scale dataset. As we all know, a complex computational problem can be processed parallel. With the coming of the big data, the researches on efficient parallel CF algorithms have become more and more important in real applications. In Ref. [9]. a parallel item-based collaborative filtering (P_Item-basedCF) algorithm based on co-factor in Mahout was proposed. Zhou *et al.* [10]. described a parallel algorithm called Alternating-Least-Squares with Weighted-λ-Regularization (ALS-WR). Jiang *et al.* [11]. proposed the efficient partition strategies not only to enable the parallel computation in each Map-Reduce phase but also to maximize data locality to minimize the communication cost. Luo *et al.* [2]. proposed the parallel RMF (P-RMF) model by applying the Alternating Stochastic Gradient Solver (ASGD) solver to deal with the parameter training process. Karydi *et al.* [13]. implemented two parallel versions of the collaborative filtering algorithm Slope One, which owned advantages such as the efficiency and the ability to update data dynamically. Xu *et al.* [14] proposed SingCF approach aiming at improving the recommendation accuracy, which attempted to incorporate multiple singular ratings and to implement collaborative filtering. Li *et al*. [15] proposed parallel algorithms for SimRank computation on Map-Reduce framework. However, these algorithms did not consider the impacts of user's activities on recommendation results. And most of them evaluated algorithms only on one or two perspectives, not on comprehensive perspective such as classification accuracy, prediction accuracy, Coverage etc. But these metrics are very important for recommendation system.

To our best knowledge, most of recommendation algorithms are serial algorithms. With the increasing of the amount of data, parallel algorithms have attracted more and more attentions. As the best distributed framework, Hadoop can be used to process large data sets. Hence, parallel recommendation algorithms based on Hadoop have far-reaching implications.

In this paper, we focus on designing an efficient parallel Item-based CF algorithm for recommendation system on large-scale rating dataset based on Hadoop. An improved parallel item-based collaborative filtering (IP_Item-basedCF) algorithm based on Hadoop is proposed. In order to consider the influence of user's activity, a new parameter called IUF is introduced that can give the active users soft punishment. In addition, the user's rating is also considered in our predicting model.

The rest of this paper is organized as follows. Section 2 presents the preliminary study. Section 3 introduces our model that overcomes the weakness of existing distributed item-based CF algorithm. The experimental analyses and results are reported in Section 4. In the last section, conclusions have been given and researches in the future have been put forward.

2. Preliminaries

This section makes a brief description of the notations used in our paper firstly. Then we analyze the data processing based on MapReduce and the feasibility of traditional CF algorithm based on Hadoop.

2.1. Notations

Table 1 lists the descriptions of notations used in the paper. Each notation will be defined and explained detailed in Section 3.

Table 1. Notations

Notations	Descriptions
U	The number of users
I	The number of items
I_n	The Nth item
R_{ai}	Rating of user a on item i
R_{ai}^{p}	The predicted rating of user a on item i
$\bar{R}_i$	Average of all $R_{ai} \forall a \in U$
$N(a)$	The set of items that user a has rated
$N(i)$	The set of users that like item i
w_{ij}	Similarity between item i and item j
$\mathbf{T}$	The co-occurrence matrix
$\mathbf{P}_a$	the preference vector of user a
$\mathbf{P}_a'$	Predicted user's preference vector
t_{nn}	Co-factors between item N and item N
n	The number of recommendations

2.2. The Feasibility Analysis of Traditional CF Algorithm Based on Hadoop

2.2.1. Mapreduce review

MapReduce [2], [16] is a parallel programming model proposed by Google for parallel processing of large data sets. MapReduce includes two main processes: Map and Reduce. Map divided input file into several data blocks, and produces a set of output key/value pairs. Reduce merges together these key/value pairs. Obviously, key/value pairs are fundamental operations of MapReduce programming model. The computing process of MapReduce is as follows in Fig. 1.

$$\begin{gathered} Map :< K1, V1 > \to < K2, V2 > \\ Shuttle : Partitioner(\mathrm{K}\,2) \to \mathrm{Reduce} \\ \mathrm{Re}\,duce :< K2, list(\mathrm{V}\,2) > \to < K2, V3 > \end{gathered}$$

Fig. 1. The computing process of MapReduce.

In Map phase, each Split, which input data is divided into, has a corresponding Map. After computing, it gets zero or more <K2, V2>, which may be different data type. In Shuffle phase, middle output results are transported to Reduce. In general, Partitioner decides which key/value pairs are mapped to one certain Reduce by Hash value of K. In Reduce phase, final results are written into HDFS.

2.2.2. Feasibility analysis

The feature of MapReduce programming model is to divide the file into blocks, then to compute them parallelly. Map function just calculates on a single row or other particular amount of data.

When computing the Item-User matrix in the phase of Map, we take the itemId as K values, the (userId, itemId, rating) as input values V and the (userId, preference) as output values V in our algorithm according to MapReduce model. Then in the phase of Reduce, we merge the V values by the same K values, i.e. the same itemId. The output values <K, V> are equal to the each row of Item-User matrix. However, the entire Item-User matrix, that is the entire file, will be involved while calculating the similarity between items based on Item-User matrix. Traditional ItemSimilarity computation need calculate all corresponding user's rating, for example, a single row data just presents one user's rating record, so the calculation must use several rows information in the matrix and the whole matrix needs to be loaded into the memory. When the data size of file is small, it is enough to be loaded into RAM. Obviously, it is unsuitable for the Big Data.

Hence, the traditional CF algorithms are infeasible for MapReduce programming model [17]. directly, which requires researchers to design new parallel CF algorithms based on Hadoop.

3. Parallel Item-Based Collaborative Filtering Recommendation Algorithm

In Section II, we know that the traditional CF algorithms are not suitable for MapReduce programming model. Based on the parallel algorithm in Apache Mahout, this section emphasizes on discussion of our parallel CF algorithm.

3.1. Parallelization Design for Item-Based Collaborative Filtering (P_Item-Basedcf) Algorithm

Apache Mahout is a new open source project by the Apache Software Foundation (ASF) based on Hadoop. Parallel item-based collaborative filtering in Mahout is a three-step algorithm which is as following.

3.1.1. Constructing user's preference vectors

In order to convert serial CF algorithm into distributed programming model, user's preference on items can be noted as a vector $[R_{a1}\ R_{a2} \cdots R_{an-1}\ R_{an}]^T$, named user vector. Each dimension indicates an item and the value represents the preference value. The scale of preference value is from 0 to 5, 0 indicates user has no preference on the item. User vector is sparse because lots of users are just interested in few items.

3.1.2. Computing co-occurrence matrix

Parallel recommendation algorithms depend on ItemSimilarity implementation as well as Traditional Item-based CF recommendation algorithms. Firstly, when designing parallel algorithm, the co-factor which means the number of users who rated the same items simultaneously in their preferences lists is computed to replace the traditional similarity calculation such as Pearson correlation coefficient, cosine similarity. The matrix consisted of co-factors is called co-occurrence matrix. Obviously, co-occurrence matrix is a symmetric matrix, because the similarity between items i and j is the same as the similarity between items j and i. Table II shows a simple example for co-occurrence matrix.

From Table 2, if there are 7 users who express some preference for both Item1 and Item2, then Item1 and Item2 co-occurred 7 times, which means their co-factor is 7. If two items never appear together in any user's preferences list, their co-factor is 0. Conceptually, each item co-occurs with itself every time if any user expresses a preference for it, its co-factor is defined as 0 because this value won't be useful in co-occurrence matrix [9]. Co-occurrence matrix is much like similarity matrix, the more times two items appears together, the more similar they possibly are. So co-occurrence matrix plays a role like ItemSimilarity in the serial item-based CF algorithm.

Table 2. The Co-Occurrence Matrix for the Simple Example

	Item1	Item2	...	ItemN-1	ItemN
Item1	0	7	...	5	1
Item2	7	0	...	3	2
...	...	...	...	...	...
ItemN-1	5	3	...	0	3
ItemN	1	2	...	3	0

3.1.3. Recommendation result

In order to compute recommendations for user a, the user vector, as a column vector, will be multiplied with the co-occurrence matrix, as shown in (1).

$$\begin{array}{c} \mathbf{T} \\ I_1 \\ I_2 \\ \vdots \\ I_n \end{array}\begin{array}{c} \begin{array}{cccc} I_1 & I_2 & \cdots & I_n \end{array} \\ \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nn} \end{pmatrix} \end{array} \bullet \begin{array}{c} \mathbf{P}_a \\ \begin{pmatrix} R_{a1} \\ R_{a2} \\ \vdots \\ R_{an} \end{pmatrix} \end{array} = \begin{array}{c} \mathbf{P}'_a \\ \begin{pmatrix} R^p_{a1} \\ R^p_{a2} \\ \vdots \\ R^p_{an} \end{pmatrix} \end{array} \tag{1}$$

Equation (1), T denotes the co-occurrence matrix, $\mathbf{P}_a$ denotes the user vector for a, $\mathbf{P}_a^{'}$ is the recommendation result for user a. After sorting from R^p_{a1} to R^p_{an}, the top n items can be recommended.

3.2. Disadvantage of P_Item-Basedcf

The shortcomings of above P_Item-basedCF algorithm mainly include two parts as follows.

1) As we have already discussed above, the preference list of each user makes a contribution to similarity between items. Therefore, co-factors give an equal weight to each of user's contribution. It will reduce the accuracy of recommendation.
2) While predicting preference for one user, P_Item-basedCF adds up all values after doing multiplication between the co-occurrence matrix and the user vector. After sorting by each component in P_a, the top N items are recommended to user a. But the values in P_a cannot be described a predicted preference value well because they are too large.

3.3. Improved Parallel Item-Based Collaborative Filtering (IP_Item-Basedcf) Algorithm

To aim for solving both of disadvantages above, an Improved Parallel Item-based Collaborative Filtering (IP_Item-basedCF) algorithm is proposed from two aspects.

3.2.2. Parallel similarity calculation

Co-occurrence matrix ignores the influence of user's activity degree, which is just defined according to the total number of items rated by each user. Assumed that 100 items are rated by user a, the activity degree of user a is 100. The more items are rated by one user, the bigger activity degree he has. But this cannot represent high activities of one user.

John S. Breese [18]. proposed an index called IUF (Inverse User Frequence), which can be used to illustrate the user' s activity degree. He considered the contributions to similarity which active user made should be less than inactive user. He thought that IUF should be joined in the formula for computing ItemSimilarity.

$$w_{ij} = \frac{\sum_{u \in N(i) \cap N(j)} 1}{\sqrt{|N(i)||N(j)|}} \tag{2}$$

Equation (2) is original formula for computing ItemSimilarity. Where $N(i)$ denotes the set of users who like item i. After adding $IUF = \frac{1}{\log(1+|N(u)|)}$, (2) can be rewritten as follows.

$$w_{ij} = \frac{\sum_{u \in N(\mathrm{i}) \cap N(\mathrm{j})} \frac{1}{\log(1+|N(u)|)}}{\sqrt{|N(i)||N(j)|}} \tag{3}$$

where $u \in N(i) \cap N(j)$ is equivalent to co-factors in co-occurrence matrix. In fact, IUF gives the active users soft punishment. In the previous section, we have already discussed that co-occurrence matrix played a role of ItemSimilarity. So in our model, we apply the IUF into the co-occurrence matrix. After modifying, the new co-occurrence matrix is as follows.

$$\begin{pmatrix} t'_{11} & t'_{12} & \cdots & t'_{1n} \\ t'_{21} & t'_{22} & \cdots & t'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t'_{n1} & t'_{n2} & \cdots & t'_{nn} \end{pmatrix} \tag{4}$$

$$t'_{nn} = \sum_{u \in N(i) \cap N(j)} \frac{1}{\log(1+|N(u)|)}$$

Fig. 2 and Fig. 3 describe the parallel process of modifying co-occurrence matrix.

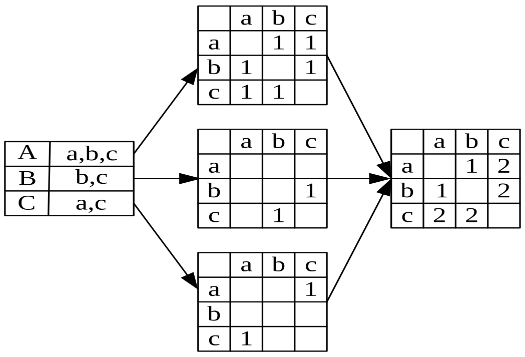


Fig. 2. The process of computing co-occurrence matrix.

Leftmost grid presents preference list of user A, B and C, in which each line denotes the set of items that one user preferred. For the each line item, every two items are set 1 in the new matrices, in the same way, three new matrices are produced as shown in the middle grids in Fig. 2. The value of each dimension depends on the occurrence times of each pair respectively. At last, after adding up all three matrices in the middle, the co-occurrence matrix is obtained as the rightmost. Fig. 3 shows the new co-occurrence matrix obtained after the original co-occurrence matrix multiplies the IUF.

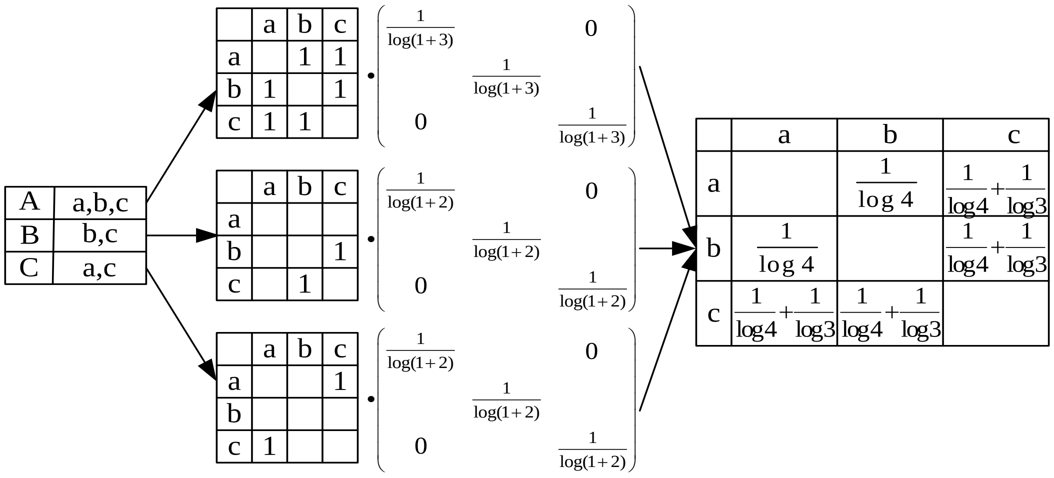


Fig. 3. The process of modifying co-occurrence matrix by multiplying IUF.

3.2.3. Predicting preference

Let us review the predicting formula in traditional item-based CF algorithm firstly, it is defined as follow.

$$R_{ai}^{p} = \bar{R}_i + \frac{\sum_{j \in S(i,K) \cap N(a)} w_{ij}(R_{aj} - \bar{R}_j)}{\sum_{j \in S(i,K) \cap N(a)} |w_{ij}|} \tag{5}$$

where $S(i,K)$ denotes the set of top K items similar with i, $N(a)$ denotes the set of items that user a has rated, $\bar{R}_i$ denotes average of all $R_{ai} \forall a \in U$. Due to the similarities between (1) and (5), we can improve prediction model in (1) by introducing the idea in (5). After improving, each component in $\mathbf{P}_a^{'}$ can be a predicted rating.

1) When computing predicted rating, predicting model needs the average rating of each item. Then the average rating vector of all items can be described as follows:

$$\bar{\mathbf{R}} = \begin{pmatrix} \bar{R}_1 & \bar{R}_2 & \cdots & \bar{R}_n \end{pmatrix}^T \tag{6}$$

where $\bar{R}_i$ denotes the average rating for item $i(i = 1,2,\cdots,n)$.

2) After adding average vector, (1) can be rewritten as follows.

$$\mathbf{A} \bullet \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nn} \end{pmatrix} \bullet \begin{pmatrix} R_{a1} - \bar{R}_1 \\ R_{a2} - \bar{R}_2 \\ \cdots \\ R_{an} - \bar{R}_n \end{pmatrix} + \begin{pmatrix} \bar{R}_1 \\ \bar{R}_2 \\ \cdots \\ \bar{R}_n \end{pmatrix} = \begin{pmatrix} R_{a1}^{p} \\ R_{a2}^{p} \\ \vdots \\ R_{an}^{p} \end{pmatrix} \tag{7}$$

$$\mathbf{A} = diag(\frac{1}{\sum_{k-1}^{n} t_{1k}}, \frac{1}{\sum_{k-1}^{n} t_{2k}}, \ldots, \frac{1}{\sum_{k-1}^{n} t_{nk}})$$

3) At last, co-occurrence matrix is replaced with an improved matrix according to (4), and the final prediction model can be received in (8).

$$\mathbf{A}' \bullet \begin{pmatrix} t'_{11} & t'_{12} & \cdots & t'_{1n} \\ t'_{21} & t'_{22} & \cdots & t'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t'_{n1} & t'_{n2} & \cdots & t'_{nn} \end{pmatrix} \bullet \begin{pmatrix} R_{a1} - \bar{R}_1 \\ R_{a2} - \bar{R}_2 \\ \cdots \\ R_{an} - \bar{R}_n \end{pmatrix} + \begin{pmatrix} \bar{R}_1 \\ \bar{R}_2 \\ \cdots \\ \bar{R}_n \end{pmatrix} = \begin{pmatrix} R^p_{a1} \\ R^p_{a2} \\ \vdots \\ R^p_{an} \end{pmatrix} \quad (8)$$

$$\mathbf{A}^{'} = diag(\frac{1}{\sum_{k-1}^{n} t^{'}_{1k}}, \frac{1}{\sum_{k-1}^{n} t^{'}_{2k}}, \ldots, \frac{1}{\sum_{k-1}^{n} t^{'}_{nk}})$$

4. Experiments

In this section, we conduct experiments for evaluating IP_Item-basedCF proposed in Section III. Meanwhile the comparison results are also shown between IP_Item-basedCF and P_Item-basedCF through the experiments.

4.1. Dataset and Evaluation

In order to evaluate the performance accuracy of our approach, we adopt two real datasets, MovieLens and Douban [19]. MovieLens dataset collected 10,000,054 anonymous rating of approximately 10,681 movies made by 71,567 MovieLens users who joined MovieLens in 2000. Douban dataset contains 129,490 unique users and 58,541 unique movie items. Their scales of the ratings are from 1 to 5.

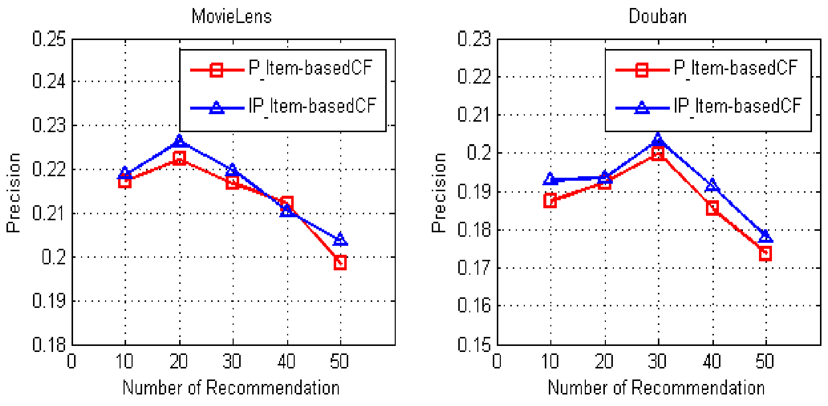
There are many metrics on performance accuracy in recommendation system [20], [21]. These metrics can be roughly classified into two categories: classification accuracy and prediction accuracy. In this paper, we use Precision, Recall as classified accuracy, MAE as prediction accuracy. In addition, Coverage metric describes the ability of a recommendation system to discover long tail items. Speedup evaluates the efficient and scalable performances for parallel algorithms. So we use Precision, Recall, MAE, coverage, speedup as our performance metrics.

4.2. Experimental Results and Analysis

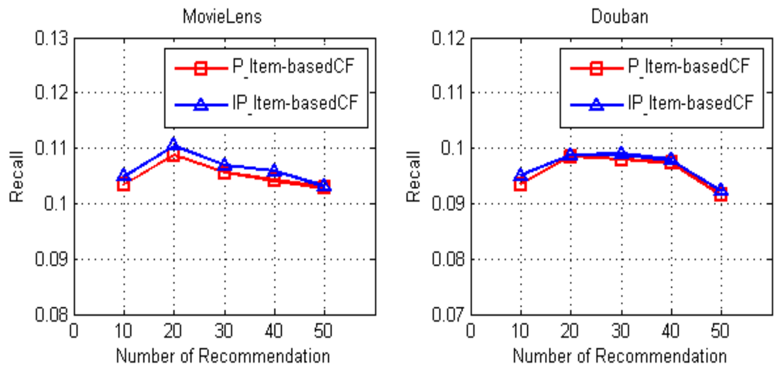
We do all experiments on a Hadoop cluster which includes 4 nodes, one node as Namenode and Jobtracker, and other three nodes as Datanode and Tasktracker. Before doing experiments, ninety percent of user's data were used as training set, and the remaining data were used as test set. In order to evaluate our algorithm, three groups of experiments have been done from the perspective of classification accuracy, prediction accuracy and Coverage on both datasets.

From Fig. 4, we can report that the correlation between classification accuracy and the number of recommendations is neither positive nor negative on precision and recall. So the number of recommendations has great effect on the classification accuracy. But the performance of two algorithms don't have great difference, our algorithm has slightly better accuracy than P_Item-basedCF.

As aforementioned contents in Section 3, estimated values in $\mathbf{P}^{'}_{a}$ cannot be represented predicted ratings accurately because they are too large and the values in $\mathbf{P}^{'}_{a}$ are just weight value. In order to do a comparison with P_Item-basedCF, we do a normalization processing for co-occurrence matrix. As shown in Fig.5, our algorithm exhibits better performance than P_Item-basedCF when the number of recommendation is 40, 20 respectively in MovieLens dataset and Douban dataset. Both of them received the lowest MAE when the number of recommendation is 50. So prediction accuracy of our algorithm is better than that of P_Item-basedCF.



(a) Precision



(b) Recall

Fig. 4. Comparison in classification accuracy between P_Item-basedCF and our algorithm.

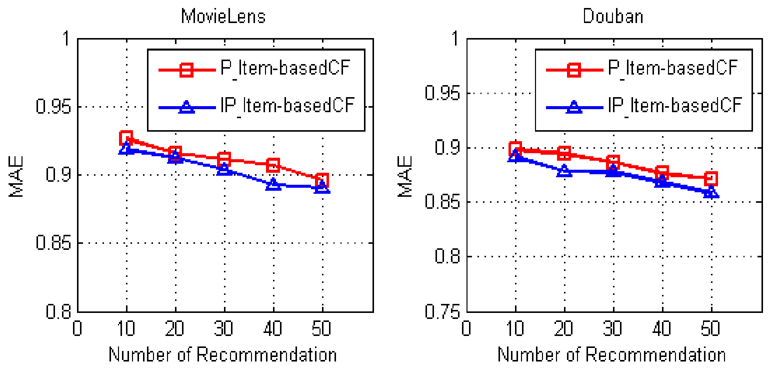


Fig. 5. Comparison in MAE between P_Item-basedCF and our algorithm.

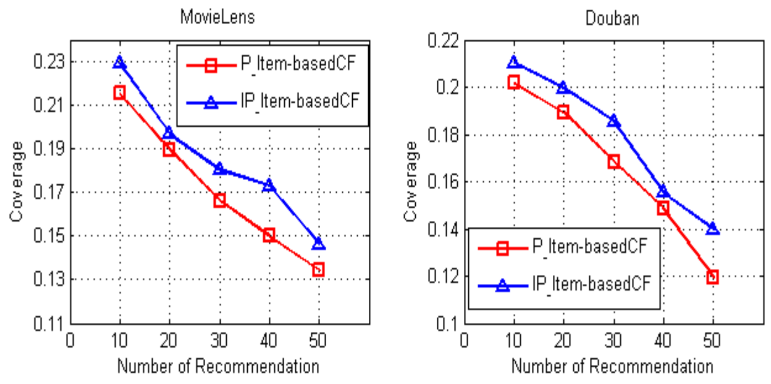


Fig. 6. Comparison in coverage between P_Item-basedCF and our algorithm.

Coverage metric describes the ability of digging long tail items in recommendation system. With the number of recommendations increasing, the distribution of items gradually appeared to be normal distribution. That means, the more the number of recommendation has, the more popular items are likely to be recommended. On the contrary, long tail items are gradually ignored. However, long tail items are significant in the recommendation systems, which reflect the personality requirements of users. So Coverage is also quite important metric for recommendation system. As is shown in Fig. 6, comparing to P_itme-basedCF, our algorithm has an outstanding performance on Coverage metric, the reason is that our algorithm does a punishment on active user.

According to the above comparison and analysis, although the classification accuracy on precision and recall of both algorithms are nearly similar, our algorithm has received obvious improvement on prediction accuracy and an outstanding performance on Coverage has been obtained especially. Generally, our algorithm has been proved effectively through the results of experiments.

4.3. Efficiency and Scalability

Speedup is an important metric on the aspect of scalability in recommendation system. As is shown in Fig. 7, when the number of cluster's node is greater than 2, speedup is growing fast. So the greater the scale of dataset is, the bigger the speedup is. Those shows that parallel recommendation algorithms based on Hadoop have a good speedup and scalability. It's verified the performance of cluster will show better with the increase of the dataset and the number of nodes in the cluster.

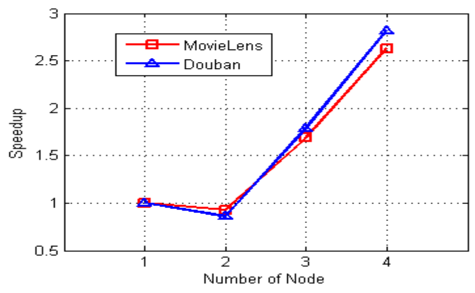


Fig. 7. The speedup of two datasets.

5. Conclusions

Collaborative filtering is the most widely used recommendation technology in the personalized recommendation system. However, studies about parallel recommendation algorithms are far fewer than those of serial traditional CF algorithms. Distributed item-based CF recommendation algorithms existed ignore the user's activity and rating. This paper propose a new parallel item-based collaborative filtering algorithm to improve the performance of recommendation based on modified co-occurrence matrix by introducing punishments on active user. We also conduct a series of experiments to compare parallel item-based collaborative filtering algorithm in Mahout with our proposed recommendation algorithm. According to the results of experiments, our algorithm outperforms the existed algorithms. Recently, Context-aware Recommendation System [22] (CARS) has been widely studied, and many companies start incorporating some contextual information into recommendation engines. The user's context information [23], [24] such as time information [25], [26] can be considered in our future works, which may obtain better performance.

Acknowledgment

This paper is sponsored by the National Natural Science Foundation of China under Grant No. 61379114, the CSC Foundation and the Natural Science Foundation of Chongqing under Grant No. CSTC2014jcyjA40047, KJ1400403.

References

[1] Adomavicius, G., & Tuzhilin, A. (2005). 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, 17(6),*

734-749.

[2] Apache™ Hadoop®. URL. Retrieved, from http://hadoop.apache.org/

[3] Sen, S., Vig, J., & Riedl, J. (2009, April). Tagommenders: connecting users to items through tags. *Proceedings of the 18th International Conference on World Wide Web* (pp. 671-680).

[4] Choi, K., & Suh, Y. (2013). A new similarity function for selecting neighbors for each target item in collaborative filtering. *Knowledge-Based Systems*, *37*, 146-153.

[5] Töscher, A., Jahrer, M., & Bell, R. M. (2009). The bigchaos solution to the netflix grand prize. Netflix prize documentation.

[6] Yang, X., Guo, Y., Liu, Y., & Steck, H. (2014). A survey of collaborative filtering based social recommender systems. *Computer Communications*, *41*, 1-10.

[7] Gao, M., Fu, Y., Chen, Y., & Jiang, F. (2012). User-weight model for Item-based recommendation systems. *Journal of Software*, *7(9)*, 2133-2140.

[8] Sun, H., Peng, Y., Chen, J., Liu, C., & Sun, Y. (2011). A new similarity measure based on adjusted Euclidean distance for memory-based collaborative filtering. *Journal of software, 6(6)*, 993-1000.

[9] Anil, R., Dunning, T., & Friedman, E. (2011). *Mahout in Action*. Manning.

[10] Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. *Algorithmic Aspects in Information and Management*, 337-348.

[11] Jiang, J., Lu, J., Zhang, G., & Long, G. (2011). Scaling-up item-based collaborative filtering recommendation algorithm based on Hadoop. *Proceedings of the 2011 IEEE World Congress on in Services* (pp. 490-497).

[12] Luo, X., Liu, H., Gou, G., Xia, Y., & Zhu, Q. (2012). A parallel matrix factorization based recommender by alternating stochastic gradient decent. *Engineering Applications of Artificial Intelligence, 25(7)*, 1403-1412.

[13] Karydi, E., & Margaritis, K. G. (2012, October). Parallel implementation of the slope one algorithm for collaborative filtering. *Proceedings of the 2012 16th Conference on Pan-Hellenic* (pp. 174-179).

[14] Xu, R., Wang, S., Zheng, X., & Chen, Y. (2014). Distributed collaborative filtering with singular ratings for large scale recommendation. *Journal of Systems and Software, 95*, 231-241.

[15] Li, L., Li, C., Chen, H., & Du, X. (2013, June). Mapreduce-based SimRank computation and its application in social recommender system. *Proceedings of the 2013 IEEE International Congress on Big Data*(pp. 133-140).

[16] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51(1)*, 107-113.

[17] Xiao, Q., Zhu Q. H. *et al.* (2013). Design and implementation of distributed collaborative filtering algorithm on hadoop. *New Technology of Library and Information Service,* 83-89.

[18] Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 43-52). Morgan Kaufmann Publishers Inc.

[19] Ma, H., Zhou, D., Liu, C., Lyu, & King, I. (2011). Recommender systems with social regularization. *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (pp. 287-296).

[20] Liang, X. (2012). *Recommendation System in Action*. Manning.

[21] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS), 22(1)*, 5-53.

[22] Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. *Recommender systems handbook* .

[23] Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., & Duval, E. (2012). Context-aware recommender systems for learning: a survey and future challenges. *Learning Technologies, 5(4)*, 318-335.

[24] Mo, Y., Chen, J., Xie, X., Luo, C., & Yang, L. T. (2014). Cloud-based mobile multimedia recommendation System with user behavior information. *IEEE Systems Journal*, *8*, 184-193.

[25] Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM*, *53(4)*, 89-97.

[26] Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., & Sun, J. (2010). Temporal recommendation on graphs via long-and short-term preference fusion. *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 723-732).

Qun Liu was born in 1969 and she is a professor at the School of Computer and Science Technology, Chongqing University of Posts and Telecommunications. She received her bachelor degree in engineering dynamics from Xi'an Jiaotong University in 1991. She received her master degree in computer technology application from Wuhan University in 2002, and she received her the PhD degree in computer software theory from Chongqing University in 2008. Her research interests are in the data mining, intelligent information processing, and complex network.

Xiaobing Li was born in 1989, He is a graduate in the Department of Computer and Science Technology, Chongqing University of Posts and Telecommunication, Chongqing, China. He received his bachelor degree in computer and science technology from Henan Polytechnic University in 2012. His main research interest is personalized recommendation.