International Journal of Control and Automation
Vol. 9, No. 12 (2016), pp.395-416
http//dx.doi.org/10.14257/ijca.2016.9.12.33

Research on Improved Collaborative Filtering Recommendation Algorithm on Hadoop

Baojun Tian[1], Xiaojuan Du[1], Peipei Hu[1] and Yila Su[1]

[1]*College of Information Engineering, Inner Mongolia University of Technology, Hohhot, 010080, China*
ngdtbj@126.com

Abstract

As an important one of recommendation technologies, collaborative filtering algorithms have many advantages, and have been very successful in both research and practice. However they also remain fundamental challenges, such as data sparsity, cold-start, dynamic changes of users' preferences and interests, and scalability. For purpose of lessening inaccurate recommendations caused by data sparsity, the pre-filled rating matrix is created by introducing a novel similarity computation method to replace the traditional ones. To address the cold-start issue, we propose a hybrid recommendation method that combines collaborative filtering and content-based filtering exploiting the advantages of both methods. To respond positively to dynamic changes of users' preferences and interests, the improved algorithm takes time factor into consideration as well. Finally we implement parallel execution of the improved algorithm on Hadoop platform, which addresses serious scalability issue when working on big data. The experimental evaluation of our proposed methods took place and the results showed that the improved algorithm has better recommendation quality and real-time performance.

Keywords*: Collaborative Filtering, K-Means, Data Sparsity, Scalability, Hadoop*

1. Introduction

With the era of big data, a massive amount of data results in an information overload, which makes it very challenging for people to find the valuable data in a short time [1-2]. Information filtering is an important part in addressing the challenge. As one of the most important and effective means of making information filtering, recommender systems are applied to various domains, such as movies, books, news articles, jokes, and songs, which apply techniques developed to analyze user data and make recommendations that the user will probably like. Recommendation algorithms include collaborative filtering recommendations, knowledge-based recommendations that use inferences about users' preferences and specific knowledge about the domain and also how the items to be recommended meet the preferences set by the users, content-based recommendation that is based on item metadata and makes recommendations according to the settings that the users supply about their preferences, and hybrid recommendation that is the combination of one or more recommendation methods [3-5].

Collaborative filtering is the most known and widely used technique for providing fast and accurate enough recommendations to users. It relies on a dataset of ratings submitted by each user for items and uses suitable similarity methods in order to provide recommendations to the target user (*i.e.* active user) who makes the request. Moreover it can deal with unstructured data and find user novel interests. Due to its simplicity and high efficiency, therefore, collaborative filtering technique has been widely adopted by many real-world systems, such as Amazon and Netflix *et al*. However, collaborative

ISSN: 2005-4297 IJCA

filtering also has revealed some potential challenges that seriously affect recommendation quality and real-time performance, which mainly include data sparsity, cold-start (*i.e.* new users or new items), dynamic changes of users' preferences and interests, and scalability[6]. In this paper, we propose an improved collaborative filtering algorithm that aims to address these issues. The primary research contributions of the paper are: (1) Pre-filling the user-item matrix with a new similarity computation method for alleviating data sparsity, which attempts to provide high-accuracy recommendations. (2) We propose a hybrid approach that combines collaborative filtering and content-based filtering that employs clustering techniques of K-means algorithm based on users' attributes to find nearest neighbor sets of new users. (3) Introducing a time forgetting function to respond positively to dynamic changes of users' preferences and interests, which can produce more accurate recommendations. (4) Parallel execution of the improved algorithms on Hadoop, which enhances the real-time performance of recommender systems.

By evaluation metrics, we performed extensive experiments using three real datasets. Experimental results confirmed our proposed method is both practical and effective. The quality of the recommendations has been improved when compared to other alternatives.

The rest of the paper is organized as follows: Section 2 is the preliminary part, Section 3 describes the challenges of collaborative filtering and gives the proposed methods, Section 4 explains the experimental evaluation, and Section 5 contains the conclusions.

2. Preliminary

Collaborative filtering technology is based on the assumption that users who have agreed on their previous evaluations would eventually agree in the future. It not only could find the potential interests of target users, but also has high recommendation quality [7-9]. In a typical collaborative filtering scenario, there is a list of n users $U=\{u_1,u_2,\cdots, u_n\}$ and a list of m items $I=\{i_1,i_2,\cdots,i_m\}$. An active user is a distinguished user $u_a \in U$ for whom the task of a collaborative filtering algorithm is to recommend the items. Opinions expressed by a user can be explicitly given as a rating, generally within a certain numerical scale (*e.g.*, in a one-to-five scale), or can be implicitly derived from purchase records.

Collaborative filtering algorithms mainly include two categories: memory-based and model-based. Memory-based collaborative filtering algorithm is divided into user-based and item-based. It utilizes the entire user-item dataset to make predictions. And it employs statistical techniques to find neighbors of users who have a history of agreeing with the target user, which is similarly to the situation where we find neighbors of items in the system. Model-based collaborative filtering algorithm produces recommendations by developing a model of user ratings. The model building process is performed by different machine learning algorithms such as clustering techniques, Bayesian network, and rule-based algorithms *etc* [10]. Algorithms in this category take a probabilistic approach as computing the prediction values [11].

A memory-based collaborative filtering algorithm works in three steps: construct matrix model, select the target user's nearest neighbors, and produce recommendations respectively.

(1) Construct matrix model

The ratings dataset is converted into a user-item ratings matrix, *i.e.* $R_{n\times m}$, (where n is the number of users, m is the number of items), which the collaborative filtering algorithm relies on. The element r_{ij} in $R_{n\times m}$ denotes rating of user i on item j. The ratings matrix is generally high sparsity [12].

(2) Select nearest neighbors

This part is the key of the algorithm. Methods commonly used in computing similarity between users (or items) are cosine similarity, adjusted cosine similarity, and Pearson correlation coefficient. Cosine similarity is shown in Equation (1), adjusted cosine

similarity, and Pearson correlation coefficient are shown in Equation (2) and Equation (3), respectively [13-14]:

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} r_{vi})}{\sqrt{\sum_{i \in I_u} (r_{ui})^2} \sqrt{\sum_{i \in I_v} (r_{vi})^2}} \quad (1)$$

$$Sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u}) \cdot (r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2} \cdot \sqrt{\sum_{i \in I_v} (r_{ui} - \overline{r_u})^2}} \quad (2)$$

$$Sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \overline{r_v})^2}} \quad (3)$$

where r_{ui} is the rating of user u on items i, r_{vi} is the rating of user v on item i, I_{uv} denotes the set of items rated by both users u and v, $\overline{r_u}$ and $\overline{r_v}$ are the mean values of users u and v on items respectively, and I_u and I_v are the rating sets of users u and v respectively.

(3) Make predictions and produce recommendations

In the collaborative filtering algorithms, prediction computation is denoted in Equation (4) or Equation (5). All predicted values are within the same numerical scale (*e.g.*, from one to five) as the opinion values provided to the active user u_a. At present, top-N algorithm is used to produce final recommendations. It produces a recommendation list of top-N items by aggregating the nearest neighbors' preferences.

$$P_{ui} = \overline{r_i} + \frac{\sum_{n \in N_i} Sim(i,n) \times (r_{un} - \overline{r_n})}{\sum_{n \in N_i} |Sim(i,n)|} \quad (4)$$

$$P_{ui} = \overline{r_u} + \frac{\sum_{v \in N_u} Sim(u,v) \times (r_{vi} - \overline{r_v})}{\sum_{v \in N_u} |Sim(u,v)|} \quad (5)$$

Where N_i denotes the nearest neighbor set of the item i, N_u is the nearest neighbor set of the user u respectively.

3. Collaborative Filtering Improvements

Collaborative filtering systems have been very successful in practice, but their widespread uses have revealed some potential challenges. In this section, we study the existing challenges of pure collaborative filtering algorithms, *i.e.* data sparsity, cold start, dynamic changes of users' preferences and interests, and scalability. As to them, in section 3.1, we analyze the existed issues, and propose the corresponding measures. The implementations of the improved algorithms are introduced in section 3.2.

3.1. Proposed Methods

3.1.1. Data Sparsity

In collaborative filtering algorithms, the more ratings are submitted by users, the more accurate recommendations are [15-17]. However in many commercial recommender systems (*e.g.*, Amazon, CDNOW), active users may have purchased well under 1% of the items. The user-item matrix is commonly extremely sparse, items with users' comments account for a small proportion of the total. Under this circumstance, it is very difficult to find similar users accurately, which leads to the bad recommendations directly [18]. On the basis of the improved similarity computation in Equation (6), in order to lessen the negative effect of data sparsity we adopt item-based collaborative filtering algorithm to fill the original ratings matrix, which can increase the number of co-ratings. By this way, the correlation between users and items is considered, *e.g.*, whether the items that the users in the nearest neighbors set of a target user like, or the items in the nearest neighbors set of items a target user likes might be what a target user is interested in.

The similarity computation of pure collaborative filtering algorithms by Peasron correlation coefficient is not reliable. Because it may be this kind of situation: there are few co-ratings between two items, while their Person correlation coefficient is very large,

which indicates they are similar in a sense it is accidental [19]. In order to avoid the occurrence of this situation, we introduce a weighted similarity function, which is stated in Equation (6):

$$Sim(i,j) = \frac{m}{a} \frac{\sum_{i \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{i \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{i \in U_{ij}} (r_{uj} - \bar{r}_j)^2}} \quad (6)$$

where r_{ui} is the rating of user u on item i, $\bar{r}_i$ is mean rating of all users on item i, r_{uj} is the rating of user u on item j, $\bar{r}_j$ is mean rating of all users on item j, U_{ij} is set of users who rate both items i and j, m stands for the size of intersection of users having co-ratings between items i and j, and a is threshold of the size of intersection. Setting m/a is a value in the range (0,1], when m≥a, it is 1.

3.1.2. Cold Start

Cold-start, it is a special case of data sparsity, has two variants: new users or new items [20-22]. Between them, the issue of new users is particularly serious, when a new user login recommendation web site for the first time, because of lacking ratings data collaborative filtering algorithms cannot precisely recommend the items for him or her, which may result in loss of the users for the mistrust of recommender systems. Similarly when the ratings on new items do not reach a certain ratios, they cannot be precisely recommended to users as well, which may cause to lower the freshness of recommendation web site.

In this paper, we propose the improved K-means algorithm, whose initial clustering centers are based on users' attributes shown in Table 1 to Table 4 [23]. After users' attributes are processed with the discrete way shown in Table 5, we exploit Euclidean distance defined in Equation (7) to calculate similarity between users to decide which cluster the new user should belong to.

In n-dimension space, Euclidean distance between two points $a_1(x_1,x_2,\ldots,x_n)$ and $a_2(y_1,y_2,\ldots,y_n)$ is stated below:

$$d = \sqrt{\sum_{i=1,j=1}^{n} (x_i - x_j)^2} \quad (7)$$

In one cluster, we use adjusted cosine similarity defined in Equation (2) to find nearest neighbors of the new user. Then collaborative filtering algorithm recommends the items for the most similar neighbor of the new user on the filled rating matrix. By this way, the issue of the new user cold-start can be addressed. The new items are similarly to the situation.

Table 1. Users' Demographic Data

ID	Age	Gender	Occupation
1	23	Male	technician
2	53	Female	other
3	39	Male	writer
4	25	Male	technician

Table 2. Discrete Age

ageStageNum	ageStage
0	under 18
1	18-24
2	25-34
3	35-44
4	45-49
5	50-55
6	56+

Table 3. Discrete Gender

genderNum	Gender
0	Female
1	Male

Table 4. Discrete Occupation Information

occupationNum	Occupation
0	other or not specified
1	academic/educator
2	artist
3	clerical/admin
4	college/grad student
5	customer service
6	doctor/health care
7	executive/managerial
8	farmer
9	homemaker
10	student
11	lawyer
12	programmer
13	retired
14	sales/marketing
15	scientist
16	self-employed
17	technician/engineer
18	tradesman/craftsman
19	unemployed
20	writer

Table 5. Discrete Users' Demographic Data

ID	Age	Gender	Occupation
1	1	1	17
2	5	0	0
3	3	1	20
4	2	1	17

3.1.3. Dynamic Changes of Users' Preferences and Interests

As time goes on, users' preferences and interests might dynamically change, if recommendation algorithms can't rapidly respond to them, the accuracy of predictions will get worse [24]. So we take the specific time when the users rate the items into consideration [25]. Furthermore, Due to the fact that the forgetting tendency in humans is slow after fast and non-linear, we introduce a non-linear function to describe it, *i.e.* the function is denoted in Equation (8):

$$h(u,i) = 1 - e^{-(t_{u,i}-t_0)} \quad (t_{u,i}>=t_0) \tag{8}$$

here $t_{u,i}$ is the time when the user u rate the item i, t_0 is the earliest rating time.

After introducing the time forgetting function, the adjusted similarity computation is defined in Equation (9):

$$\mathrm{sim}(u,v) = \frac{\sum_{i\in I_{uv}}(r_{ui}\cdot h(u,i)-\overline{r_u})\cdot(r_{vi}\cdot h(v,i)-\overline{r_v})}{\sqrt{\sum_{i\in I_{uv}}(r_{ui}\cdot h(u,i)-\overline{r_u})^2}\cdot\sqrt{\sum_{i\in I_{uv}}(r_{vi}\cdot h(v,i)-\overline{r_v})^2}} \tag{9}$$

where r_{ui} is the rating of user u on items i, r_{vi} is the rating of user v on item i, I_{uv} denotes the set of items rated by both users u and v, $\overline{r_u}$ and $\overline{r_v}$ are the mean values of users u and v on items respectively.

3.1.4. Scalability

With vast amount of available information increasing, the real-time performance of collaborative filtering algorithms confronts serious challenges, especially the system is running on standalone machine with no parallel processing [26]. How to deal with it has become one of the research topics in collaborative filtering algorithms [27].

In this paper we take advantage of the distributed cloud platform Hadoop to address serious scalability issue. Using Hadoop to implement the off-line improved K-means algorithm with MapReduce and the on-line improved collaborative filtering algorithm with MapReduce, which can increase the system's real-time power when processing very large datasets.

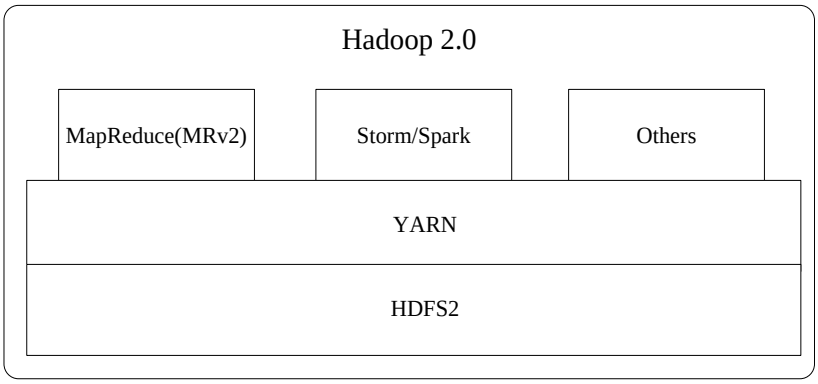

Figure 1. Hadoop 2.0 Framework

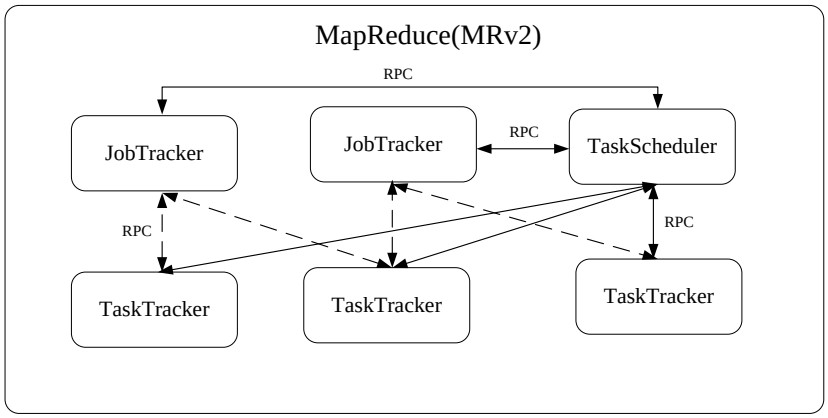

Figure 2. MRv2 Basic Framework

3.2. Implementation of the Improved Algorithms

3.2.1. Weighted Similarity Computation

In this paper, we propose a method of the weighted similarity computation between items to get more accurate ratings. First, we construct a user-item rating matrix R, and take item-based collaborative filtering algorithm utilizing the new measure to produce item's nearest neighbors set, then fill the original ratings matrix to lessen data sparsity. Algorithm 1 is shown below.

Input: R, k, m, a, I /* R is an original ratings matrix, k is the size of the nearest neighbors set, m is the size of intersection of users having co-ratings between items, a is threshold, I is a set of items */

Output: R' /* R' is a filled ratings matrix */

Begin

For each item i in I

For each item j in I-{i}

use the Equation (2) to calculate sim(i,j)

sim(i,j)= (m/a)*sim(i,j)

If (the size of S_i <= k) /* S_i is the nearest neighbors set of the item i */

put item j into S_i

Else

If (sim(i,j) > m /*m is the minimum similarity value between items in S_i*/

put item j into S_i, and remove item k /* similarity between items i and k is the minimum value */

End If

End If

End For

For each user u in S_{ui} /* the user in S_{ui} that does not rated on item i */

Sumsim = 0

SumRating = 0

For each item n in S_i

Sumsim +=|Sim(i, n)|

SumRating += Sim(i, n) ×(r_{un} —$\overline{r_n}$) /* r_{un}, $\overline{r_n}$ and $\overline{r_i}$ are defined in

Equation (4) */

Pre(u,i) =$\overline{r_i}$+ SumRating / Sumsim

fill the Pre(u,i) into matrix R

End For

End For

End

3.2.2. Collaborative Filtering with Time Forgetting Function

In order to consider the dynamic changes of users's interests, we introduce the time forgetting function to adjust the ratings in matrix R' for higher recommendation accuracy, which can compute reasonable prediction ratings at different moments.

In this paper, a nonlinear index forgetting function given in Equation (8) is introduced, then we get a new rating matrix R". After that we use the user-based collaborative filtering algorithm to produce the final recommendations. Algorithm 2 is shown below.

Input: R', k, N, u, U /* R' is a filled ratings matrix, k is the size of the nearest neighbors set, N is the number of recommended items, user u is a target user, U is a set of users */

Output: N items /* the recommended Top-N items */

Begin

adjust R' by time forgetting function, then get R"

For each user v in U-{u}

use Equation (9) to compute sim(u,v)

If (the size of S_u <= k) /* S_u is the nearest neighbors set of user u */

put user v into S_u

Else

If (sim(u,v) > m /*m is the minimum similarity value in S_u*/

put user v into S_u, and remove user k /* similarity between users u and k is the minimum value */

End If

End If

For each item i in S_{iu} /* the item in S_{iu} that does not rated by user u */

Sumsim = 0

SumRating = 0

For each user v in S_u

Sumsim +=|Sim(u, v)|

SumRating += Sim(u, v) ×(r_{vi} —$\overline{r_v}$) /* r_{vi}, $\overline{r_v}$ and $\overline{r_u}$ are defined in Equation (5) */

Pre(u,i) =$\overline{r_u}$+ SumRating / Sumsim

put Pre(u,i) in ratingPre[ ] /* ratingPre[ ] is a prediction ratings list of user u */

End For

sort ratingPre[ ] by the ratings

output top-N items of ratingPre[ ]

End

3.2.3. Improved K-means Algorithm

Under the new users or the new items scenarios, since only few ratings are available, recommendations are still issues, we proposed a commendation method that utilizes a hybrid algorithm, *i.e.* collaborative filtering algorithm combined with K-means algorithm, which aims to address the cold start issue. The method divides users into several clusters, which are based on their attributes. And each of the clustering centers represents the whole cluster. By calculating Euclidean distance defined in Equation (7) between new users and clustering centers, we can determine which clusters new users should belong to, and then use the cosine similarity to calculate the attributes' similarity of users, which can find the most similar target user. Then the items recommended to the target user are what the new user likes. Similarly, new items cold-start issue can be addressed as well.

Since the K-means algorithm is sensitive to the parameter K, in this paper we choice the best one by the experiments. After given clustering parameter K, the traditional K-means algorithm often randomly selects K elements as initial clustering centers, which easily causes the results to fall into local optimum. Therefore we propose a method that uses weka [28] tools to process dataset by the SimpleKMeans algorithm and then get the set C of clustering centers. In general, the elements in set C are not in initial dataset, so it needs to calculate the shortest Euclidean distance between the dataset and the set C, which can get the set U_0, each element of whom certainly belongs to the dataset. It is similarly to the situation where we get I_0 that is initial clustering centers of items. Algorithm 3 is shown below.

Input: U, R", K, N, u_0, U_0, k /* U is a matrix of users' attributes, R" is a adjusted ratings matrix, K is a clustering parameter, N is the number of recommended items, u_0 (id, gender, age, occupation) is a new user, U_0 is set of initial clustering centers, k is the size of the nearest neighbors set */

Output: N items /* recommended Top-N items for user u_0*/

Begin

select U_0 as center_list /* center_list is a set of initial clustering centers of K-means algorithm*/

Do {

For each user u in U- U_0

For (each c in Center_list)

calculate min_distance(u, c); /* min_distance(u,c) is the minimum value by Equation (7) between user u and the clustering center */

End For

put user u in a $cluster_u$ /*the distance between user u and the center of $cluster_u$ is the minimum value */

End For

recalculate the set of clustering centers as U_0

} Until (distortion function of K-means is convergent)

put u_0 in a $cluster_m$ by calculating the minimum distance between attributes

find the most similarly user of u_0, *i.e.*u_m, in $cluster_m$ by Equation (1)

call the algorithm 2 to get top-N recommended items $\{i_1, i_2,\ldots,i_m\}$for user u_m

the set$\{i_1, i_2,\ldots,i_m\}$ is top-N recommended items for use u_0

End

3.2.4. Improved Collaborative Filtering Algorithm with MapReduce

The improved collaborative filtering algorithm with MapReduce is mainly divided into two stages: filling the original ratings matrix stage, producing final recommendations stage.

Filling the original matrix with MapReduce includes four stages [29]: (1) dealing with data format; (2) computing similarity between items; (3) generating the nearest neighbor sets; (4) predicting the ratings.

After that, we propose user-based collaborative filtering algorithm with the time forgetting function to produce final recommendations, which also includes the above four MapReduce stages, so we do not repeat it here.

Below MapReduce process of filling the original ratings matrix is introduced in detail.

(1) The first MapReduce procedure: dealing with the data.

Map procedure:

Input: <offset, (item_id, user_id, rating)>

Output: <item_id, (user_id, rating)>

Steps: Map function firstly uses the FileInputFormat class in MapReduce framework to read data blocks with lines, and forms the <key, value> pairs. Then it partitions each row of data, and passes the data pairs in the form of <item_id, list(user_id, rating)> to Reduce function.

Reduce procedure:

Input: <item_id, list (user_id, rating)>

Output: <item_id, list (user_id, mean-rating)>

Steps: Reduce function summarizes the intermediate results generated by Map function with the same key values, *i.e.* the same item_id values. It calculates the mean values of all items, then form the data pair < item_id, list(user_id, mean-rating)> and saves the results in HDFS.

The MapReduce procedure is shown in Figure 3.

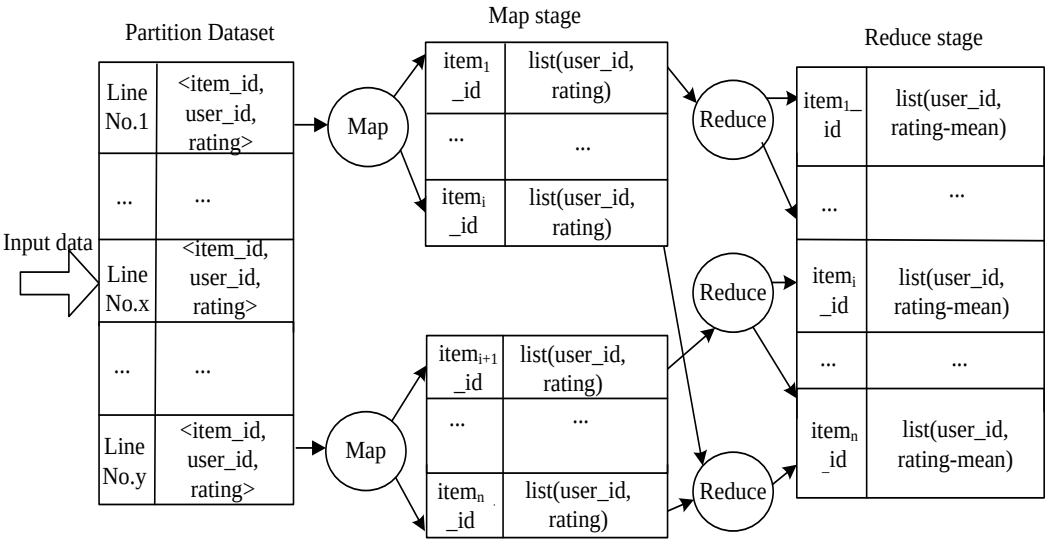

Figure 3. Dealing with Data with MapReduce

(2) The second MapReduce procedure: computing similarity between items.

Map procedure:

Input :<offset,(item_id, user_id, rating-mean)>

Output: <(item$_i$_id, item$_j$_id), similarity >

This stage need not Reduce procedure.

Steps: Map function deals with <offset, (item_id, user_id, rating-mean)> pairs, then they are parsed into <(item$_i$_id, item$_j$_id), similarity > pairs by the Equation (6), and saves the results in HDFS.

The MapReduce procedure is shown in Figure 4.

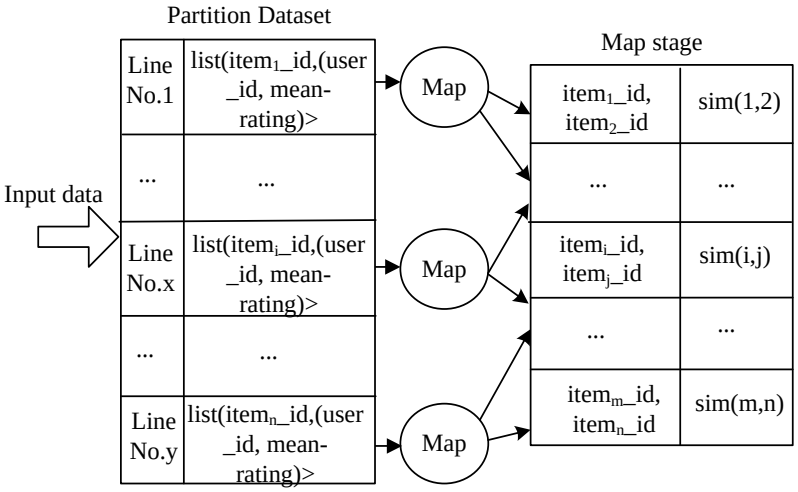

Figure 4. Items Similarity Computation with MapReduce

(3) The third MapReduce procedure: producing the nearest neighbor sets.

Map procedure:

Input: <offset, (item$_i$_id, item$_j$_id), similarity)>

Output: < item$_i$_id, list (item_id, similarity)>

Steps: The <(item$_i$_id, item$_j$_id), similarity)> pairs are parsed into <item$_i$_id, list(item_id, similarity)> pairs, and then they are passed into Reduce.

Reduce procedure:

Input: < item$_i$_id, list (item_id, similarity)>

Output: <item_id, list of nearest neighbour>

Steps: Reduce function analyzes all <item$_i$_id, list(item_id, similarity)> pairs, then they are sorted by similarity, which we can get the nearest neighbor set of each item. The results are saved in HDFS.

The MapReduce procedure is shown in Figure 5.

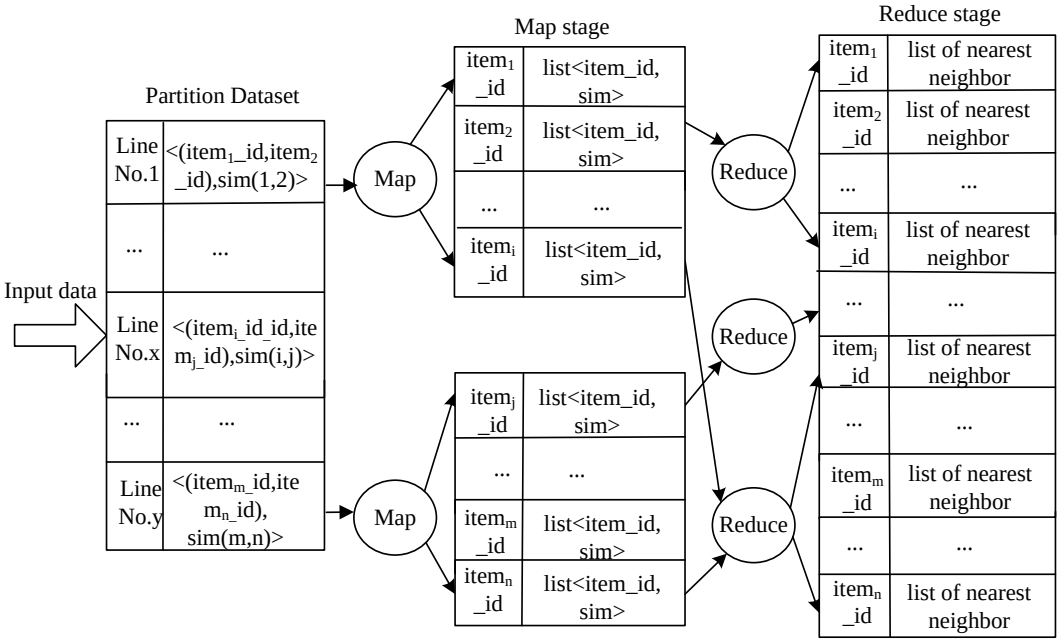

Figure 5. Generating the Nearest Neighbor Sets with MapReduce

(4) The fourth MapReduce procedure: predicting the ratings.

Map procedure:

Input: <offset, list(item_id, user_id, rating-mean)>, <offset, (item_id, list of nearest neighbor)>

Output: < item_id, (user_id, rating-mean)>, <$item_i$_id, list(item_id, similarity)

Steps: The list (item_id, user_id, rating-mean) and the < item_id, list of nearest neighbour> pairs are parsed into <item_id, list(user_id, rating-mean)> pairs and <$item_i$_id,(item_id, similarity)> pairs by Map function.

Reduce procedure:

Input: <item_id, (user_id, rating-mean)>, <$item_i$_id, list(item_id, similarity)

Output: < item_id, list(user_id, prediction ratings)>

Steps: Reduce analyzes all <key, value> pairs with the same key value, searches <item, user> pairs with no ratings in matrix. It predicts the ratings by using Equation (5).

The MapReduce procedure is shown in Figure 6.

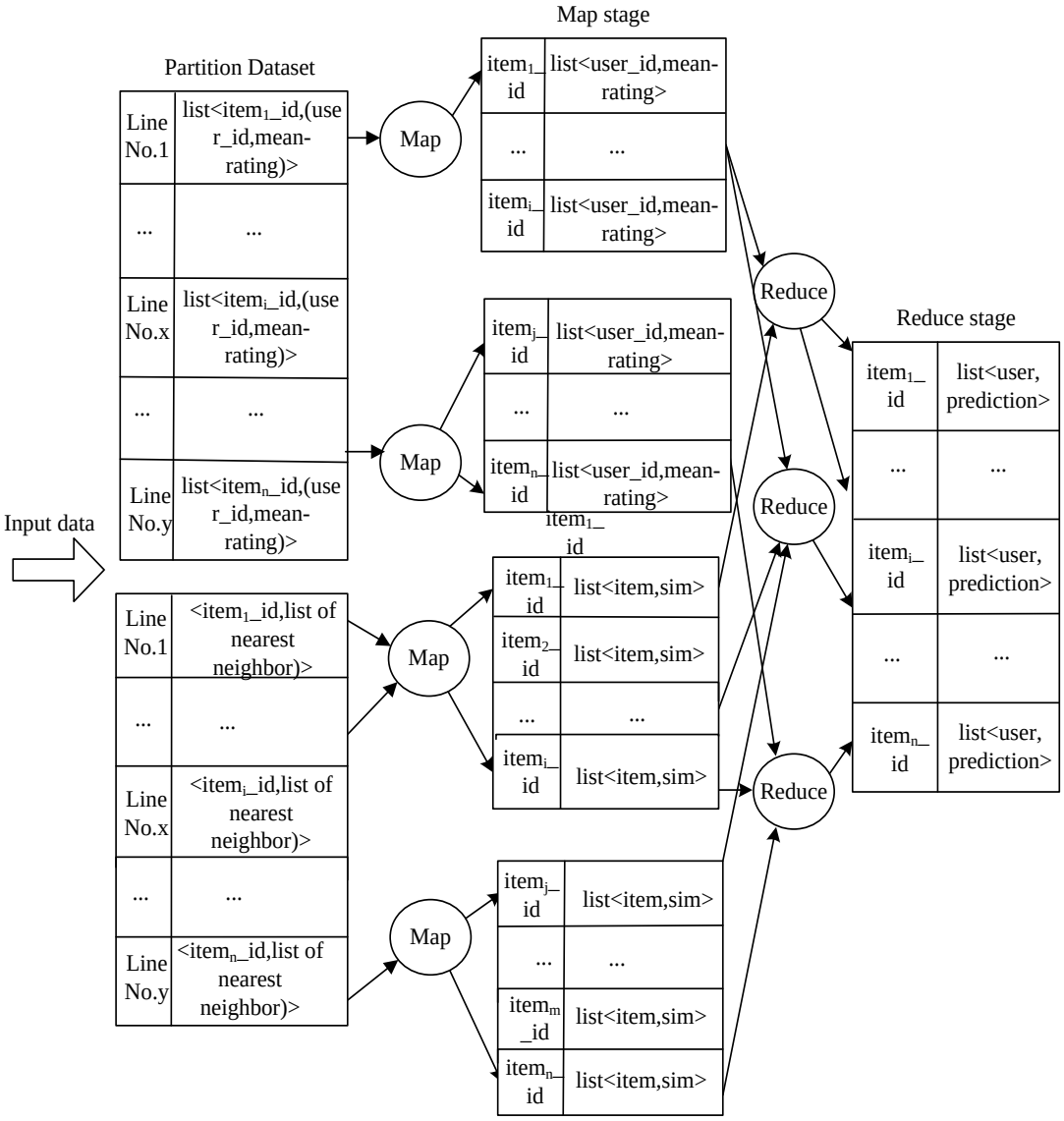

Figure 6. Predicting Ratings with MapRduce

3.2.5. Improved K-means Algorithm with MapReduce

Addressing cold-start issue includes three stages: (1) the generation of clustering models of the users or the items by using K-Means algorithm; (2) finding the clusters that the new users or the new items should belong to and looking for the most similar neighbor in the clusters; (3) producing recommendations.

There are three steps when we implement K-Means algorithm with MapReduce [30,31]. (1) Scanning all the data in the users' attributes matrix and using K users in U_0 (defined in section 3.2.3) as initial clustering centers; (2) each Map node reads the local dataset, produces clustering sets by K-means algorithm, finally gets the new global clustering centers with several clustering sets in Reduce procedure, and repeats the process in the second step until it meets end conditions; (3) classifying all data according to the clustering centers.

4. Experimental Evaluation

In this section, the experimental evaluation of our proposed method based on three real datasets took place and the results were compared under different algorithms.

4.1. Environment Setup

In sequential processing approach, the system was running on standalone machine with 4G memory and 4-core i5 processor, Windows 8 operating system, Eclipse 3.7.1, and JDK 1.7.0_79. The parallel approach in MapReduce was based on Hadoop 2.0 platform, which consisted of eight virtual nodes created by VMware software, *i.e.* one as master node, the others as slave nodes. Each node has 4G memory and 4-core i5 processor, operating system was CentOS 6.5 [32,33]. All the algorithms were implemented in the Java programming language.

4.2. Real Datasets

We conducted our experiments with MovieLens dataset collected by GroupLens in the University of Minnesota [34]. MovieLens is a web-based research recommender system that debuted in fall 1997. The datasets mainly include users' attributes, movies' features, and users' ratings. MovieLens provides three real datasets altogether. MovieLens ml-100k is a real dataset that contains 943 users, 1682 movies, and 100,000 ratings, whose sparsity is 93.7%. MovieLens ml-1M, the real dataset contains 6040 users, 3900 movies, and 1,000,000 ratings, whose sparsity is 95.8%. MovieLens ml-10M, the real dataset contains 71567 users, 10681 movies, and 10,000,000 ratings, whose sparsity is 93.7%. They also contain users' attributes and items' features. The rating data are in the form [userid] [movieid] [rating] [timestamp], whose values are in the scale 1 to 5. The users' attribute data are in the form [userid] [age] [gender] [occupation] [zipcode], the items' features data are in the form [movieid] [movietitle] [generes] [35].

The dataset are split into two groups when experiments performed. An 80% selected part is used for training subset and the remaining 20% is used for test subset. The improved algorithm with training subset computes the recommendations that will be later compared with the original data presented in the test subset.

4.3. Evaluation Metrics

For purpose of measuring the accuracy of the predictions, the widely accepted by the research community Mean Absolute Error (MAE) has been used. It measures the difference between prediction ratings and real user-specified values. The lower the MAE values, the more accurate the predictions. It is computed over all the ratings available in the test subset, defined in Equation (10) [36,37]:

$$MAE = \frac{\sum_{i=1}^{N} |P_i - R_i|}{N} \tag{10}$$

where P_i denotes prediction ratings, R_i refers to real user-specified ratings, N is the number of experimental items.

Speedup is taken as significant criterion to measure our algorithm's power, which is defined as follows [38]:

$$Speedup(N) = \frac{T_{(1)}}{T_{(N)}} \tag{11}$$

where $T_{(1)}$ is standalone's running time, $T_{(N)}$ is average running time with N nodes on Hadoop platform.

4.4. Results Analysis

4.4.1. Size of Nearest Neighbors

The size of the nearest neighbors has significant impact on prediction accuracy in the collaborative filtering algorithms. To determine the sensitivity of this parameter, the experiments were carried out five times. Using the ml-100k as dataset, we adopt the Pearson correlation coefficients to determine it. The mean values of the five experiments are the final results that are shown in Figure 7.

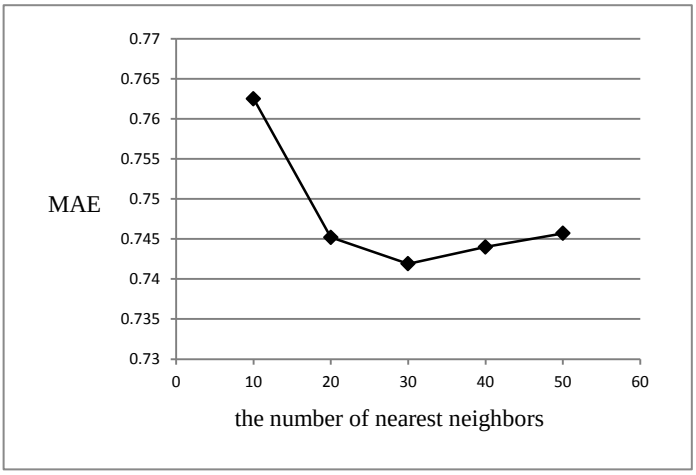

Figure 7. Choice of the Nearest Neighbor Size

From the Figure 7, we can conclude when k<30, the MAE values decrease rapidly with the increase of the k value. But when k>30, the MAE values slightly increase then the curve tends to be flat, which indicates that the recommendation quality cannot be improved with the increasing of the number of the nearest neighbors. Therefore, we select 30 as the optimum value of the size of the nearest neighbors.

4.4.2. Effect of the Threshold

To reduce the accidental errors, we improve the similarity computation by weighting the Pearson correlation coefficient. In Equation (6), a is threshold that has significant impact on the ratings accuracy. We performed experiments five times using ml-100k as the dataset. The mean values of the five experiments are the final results that are shown in Figure 8, and we can observe that when the MAE value is the minimum the specified threshold is 50.

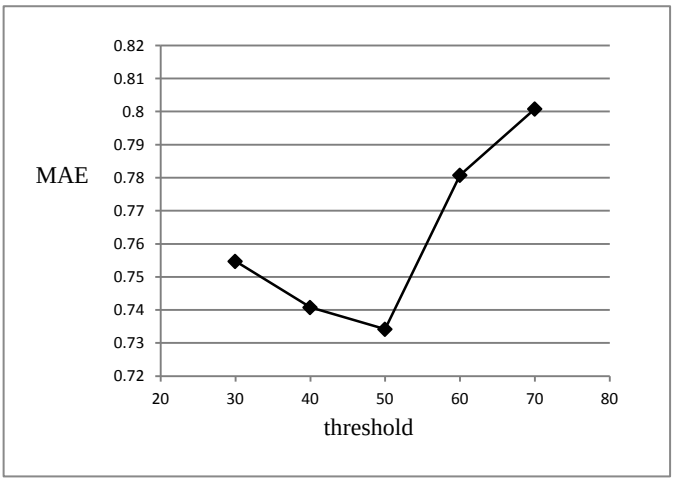

Figure 8. Choice of Threshold

4.4.3. Comparisons of Different K-means Algorithms

We built K-means model to address the cold-start issue confronted by collaborative algorithms. The parameter K is vital to K-means algorithm. We conducted the experiments five times using ml-1M as the dataset under different algorithms. The mean values of the five experiments are the final results shown in Figure 9.

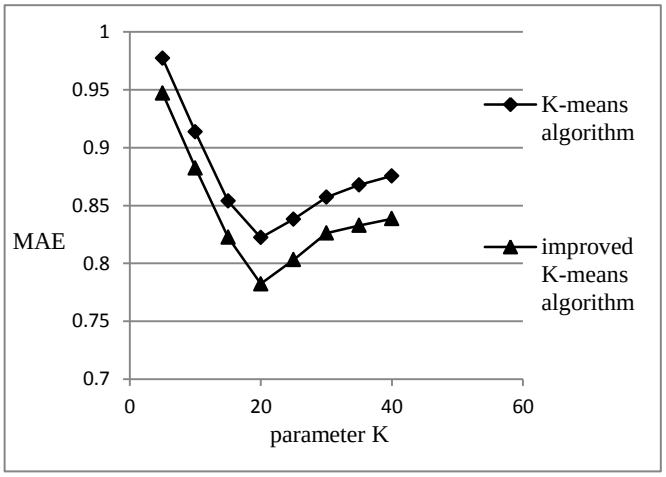

Figure 9. MAE Values Comparison of Different Parameter K

It can be observed from Figure 6 that when the parameter k is 20, the MAE values of two different algorithms are all the smallest. But the MAE value of the improved algorithm is less nearly 0.03 than the algorithm that randomly selects K initial clustering centers.

4.4.4. Comparisons of Different Collaborative Filtering Algorithms

In this paper, the experiments using ml-100k as dataset were performed five times to compare MAE values among the user-based collaborative filtering algorithms, the user-based collaborative algorithm after item-based filling, and the improved collaborative algorithm without considering time factor. The mean values of the five experiments are the final results shown in Figure 10.

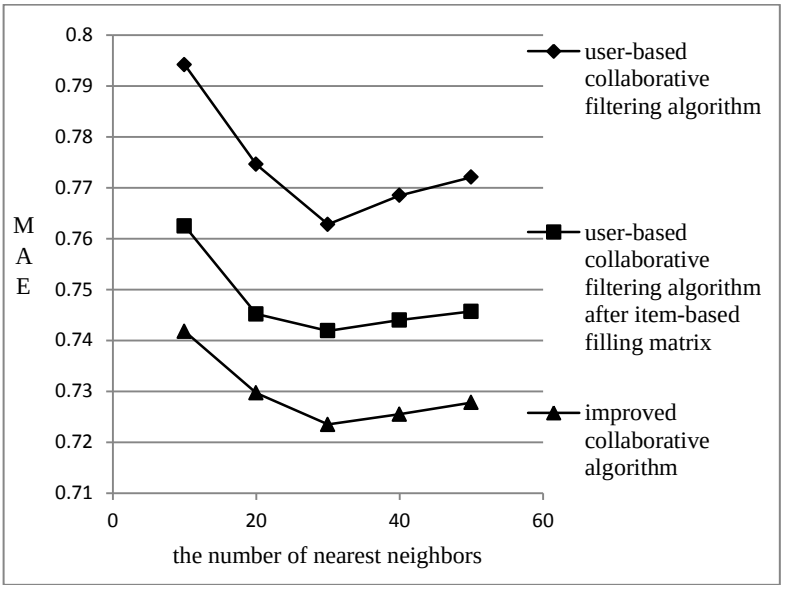

Figure 10. MAE Value Comparison of Different Recommendation Algorithms

In Figure 10, we can see that the MAE values of the improved algorithm is lower nearly 0.04 than the pure user-based collaborative filtering algorithm, and is lower nearly 0.015 than the first item-based filling matrix and then user-based collaborative filtering recommendations, which confirms the proposed algorithm is really effective in improving recommendation quality.

4.4.5. Comparisons with Time Forgetting Function

To respond positively to dynamic changes of users' preferences and interests, the improved algorithm takes time factor into consideration as well. The experiments were performed five times on ml-100k dataset, and the results are shown in Figure 11.

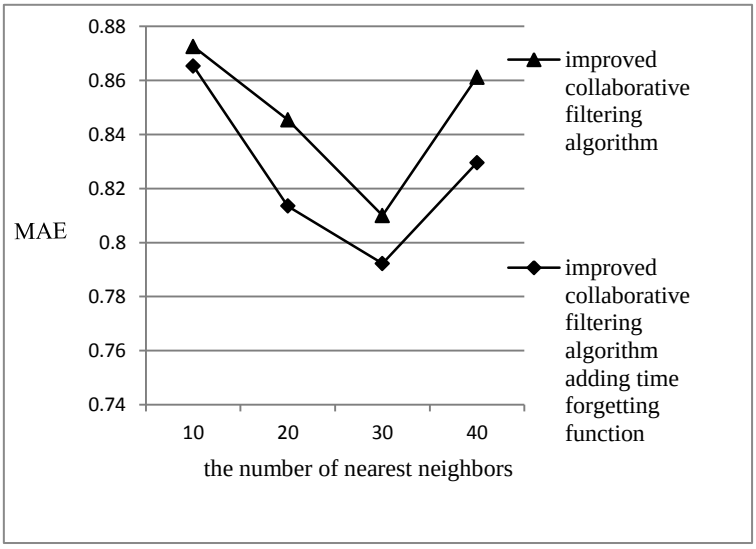

Figure 11. Comparison of the MAE Values after Adding Time Forgetting Function

In Figure 11, we can see the MAE values have been further decreased and the recommendation quality is further improved. For example when the number of the nearest

neighbors is 30, compared to the improved algorithm without considering time factor, the MAE value after adding time forgetting function is less than nearly 0.02.

4.4.6. Comparison of Real-time Performance

The runtime is compared between standalone machine and Hadoop palatform, which is shown in Table 6.

Table 6. Performance Comparison between Standalone Machine and Hadoop Platform

dataset	the number of users	the number of items	the number of ratings	runtime on Standalone Machine (h)	runtime on Hadoop platform with 8 nodes (h)
ml-100k	943	1682	100000	0.27	0.52
ml-1m	6040	3900	1000209	3.49	1.45
ml-10m	71567	10681	10000054	deadlock	30.14

From Table 6, it shows that when the dataset is smaller, the parallel processing mode on Hadoop consumes more time than the sequential one. This is because the smaller datasets might have extra time when the jobs start and communicate among the nodes. Therefore, Hadoop is not suitable for handling the smaller datasets. With the increasing of datasets, the ability of standalone machine processing is getting worse, even deadlock, whose memory and CPU are consumed faster so that it is unable to process large amounts of data quickly. While Hadoop still can maintain higher processing performance.

4.4.7. Speedup

In order to verify the influence of the number of Hadoop's nodes on the collaborative filtering algorithm's power, we use three different datasets to perform the speedup experiments. The results are shown in Figure 12.

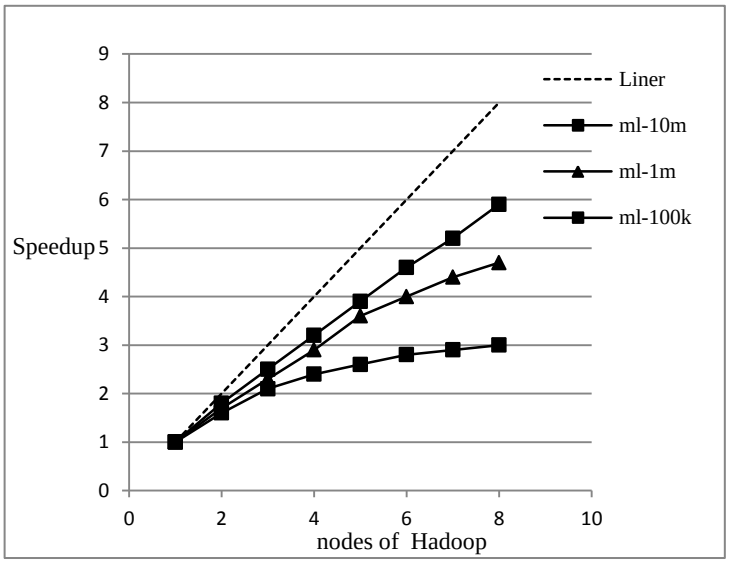

Figure 12. Comparison of Speedup with Different Dataset

In Figure 12, it shows that the speedup is the diagonal line of the axis in theory. However, in fact, with the increasing of the number of the nodes, the time of scheduling job and communicating among nodes will greatly increase, therefore, it will not achieve the values we expect. The speedup curve of ml-100k is relatively flat, while the curve of ml-10M is the best one among the three datasets. The utilization of Hadoop is lower when the datasets are smaller, which further indicates that Hadoop is more suited to process large amounts of data.

5. Conclusions

In this paper, we have a further research on fundamental challenges confronted by the traditional collaborative filtering algorithms, such as data sparsity, cold-start, dynamic changes of users' preferences and interests, and scalability. Exploiting the improved algorithms on Hadoop platform, we addressed the above issues, which negatively affect recommendation quality and real-time performance. The efficiency of the proposed method has been shown by providing the experimental evaluations using three real datasets.

Our improved algorithm is combined with K-means, when addressing the cold-start issue. However, K-means algorithm which randomly selects clustering center is easy to cause that recommendation results fall into local optimum. So, our future work will combine both k-means algorithm and genetic algorithm for global optimum, which can make recommendations more accurate.

Acknowledgments

Funding for this research is provided by the Natural Science Foundation of Inner Mongolia of China under grant No. 2015MS0613, key Natural Science Foundation of Inner Mongolia University of Technology of China under grant No.ZD201317. This work is also supported by National Science Foundation of China under grant No.61363052. We like to thank anonymous reviewers for their valuable comments.

References

[1] Q. Gao, L. Gao and J. Yang, "Weighted Collaborative Filtering Algorithm with Fused Impact Factor", Computer Engineering, vol. 40, no. 8, **(2014)**, pp. 38-42.
[2] L. Li, J. Liao, L. Ou, "Cloud Computing: An Introduction", Application Research of Computers, vol. 27, no. 12, **(2010)**, pp. 119-122.
[3] Y. Ling, "Collaborative Filtering and Its Application in Recommender Systems", Hefei: Hefei University of Technology, **(2013)**.
[4] Y. Ying, Y. Liu and C. Chen, "Personalization recommender system based on cloud-computing technology", Computer Engineering and Applications, vol. 51, no. 13, **(2015)**, pp. 111-117.
[5] A. Bellogín, P. Castells and I. Cantador, "Neighbor Selection and Weighting in User-Based Collaborative Filtering: A Performance Prediction Approach", Acm Transactions on the Web, vol. 8, no. 2, **(2014)**, pp. 12-30.
[6] B. Jeong, J. Lee and H. Cho, "An iterative semi-explicit rating method for building collaborative recommender systems", Expert Systems with Applications, vol. 36, no.3, **(2009)**, pp. 6181-6186.
[7] S. Wei, N. Ye and J. Zhu, " Collaborative Filtering Recommendation Algorithm Based on Item Clustering and Global Similarity", Computer Science, vol. 39, no. 12, **(2012)**, pp. 149-152.
[8] W. Kong, "Research on key problems of collaborative filtering recommender system", Wuhan: Central China Normal University, **(2013)**.
[9] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering", ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 4, no.1, **(2010)**, pp.1-31.
[10] M. Jiang, D. Song and L. Liao, "A Bayesian Recommender Model for User Rating and Review Profiling", Tsinghua Science & Technology, vol.20, no.6, **(2015)**, pp. 634-643.
[11] X. Yu and M. Li, "Effective hybrid collaborative filtering model based on PCA-SOM", Systems Engineering-theory & practice, vol.30, no.10, **(2010)**, pp.1850-1854.
[12] M. Gao, Z. Wu and F. Jiang, "User rank for item-based collaborative filtering recommendation", Information Processing Letters, vol. 111, no. 9, **(2011)**, pp. 440-446.

[13] Y. Xu, C. Chen and Y. Xu, "A collaborative filtering algorithm based on classify", Computer Systems & Applications, no. 1, **(2007)**, pp. 47-50.
[14] X. Bi and W. Jin, "An Improved Collaborative Filtering Similarity Model Based on Neural Networks", Intelligent Transportation, Big Data and Smart City (ICITBS), International Conference on. IEEE, **(2015)**, pp.85-89.
[15] Y. Wu and J. Zheng, "Improved collaborative filtering recommendation algorithm", Computer Engineering and Design, vol. 32, no.9, **(2011)**, pp. 3019-3021.
[16] F. Shi and E. Chen, "Combining the Items' Discriminabilities on User Interests for Collaborative Filtering", Journal of Chinese Computer Systems, vol. 33, no.7, **(2012)**, pp. 1533-1536.
[17] Z. Chen and Z. Li, "Collaborative filtering recommendation algorithm based on user characteristics and item attributes", Journal of Computer Applications, vol.31, no.7, **(2011)**, pp.1748-1750.
[18] A. U. Asuncion and M T. Goodrich, "Nonadaptive Mastermind Algorithms for String and Vector Databases, with Case Studies", IEEE Transactions on Knowledge & Data Engineering, vol. 25, no. 1,**(2013)**, pp. 131-144.
[19] X. Liu, J. Ge and D. Chen, "A Hybrid Recommendation Algorithm Based on Clustering and Collaborative Filtering", Computer Engineering & Science, vol. 32, no. 12, **(2010)**, pp. 125-127,133.
[20] H. Cao, "Research on clustering search method in collaborative filtering recommendation system", Computer Engineering and Applications, vol. 50, no.5, **(2014)**, pp. 16-20, 28.
[21] Q. Liu, "Study on key algorithms in collaborative filtering recommender systems", Zhejiang: Zhejiang University, **(2013)**.
[22] J. Sun and L. Ai, "Collaborative filtering recommendation algorithm based on item attribute and cloud model filling", Journal of Computer Applications, vol. 32, no. 3, **(2012)**, pp. 658-660,668
[23] Y. N. Chen and M. Yu, "A Hybrid Collaborative Filtering Algorithm Based on User-Item", International Conference on Computational and Information Sciences. IEEE Computer Society, **(2010)**, pp. 618-621.
[24] X. Ji, Y. Liu and L. Luo, "Similarity measurement based on user interest in collaborative filtering", Journal of Computer Applications, vol. 30, no.10, **(2010)**, pp. 2618-2621.
[25] Y. Xiao, A. I. Pengqiang and C. H. Hsu, "Time-Ordered Collaborative Filtering for News Recommendation", Communications China, vol. 12, no.12, **(2015)**, pp. 53-62.
[26] G. Huang, Y. Li and J. Gao, "Collaborative filtering recommendation algorithm based on user clustering of item attributes", Computer Engineering and Design, vol. 31, no. 5, **(2010)**, pp. 1038-1041.
[27] M. Jiang, P. Cui and F. Wang, "Scalable Recommendation with Social Contextual Information", IEEE Transactions on Knowledge & Data Engineering, vol. 26, no. 11, **(2014)**, pp. 2789-2802.
[28] M. Yuan, "Data mining and machine learning- WEKA application technology and practice", Beijing: Tsinghua University Press, **(2014)**.
[29] S. Peng, Z. Zhou and G. Wang, "Collaborative Filtering Algorithm Based on Rating Matrix Pre-filling", Computer Engineering, vol. 39, no. 1, **(2013)**, pp. 175-182.
[30] Y. Huang, "Understanding Big Data: big data processing and programming", Beijing: China Machine Press, **(2014)**.
[31] C. Yu, "Research on Parallel Clustering Algorithm Based on Map-Reduce", Xi'an:Xidian University, **(2012)**.
[32] C. Feng, Z. Qin and D. Yuan, "Techniques of Secure for Cloud Data", Chinese Journal of Computers, vol. 38, no. 1, **(2015)**, pp. 151-161.
[33] Y. Ma and X. Meng, "Research on indexing for cloud data management", Journal of Software, vol. 26, no. 1, **(2015)**, pp. 145-166.
[34] G. Lekakos and P. Caravelas, "A hybrid approach for movie recommendation", Multimedia Tools & Applications, vol. 36, no. 1-2, **(2008)**, pp. 55-70.
[35] M. N. Moreno, S. Segrera and V. F. López, "Web mining based framework for solving usual problems in recommender systems. A case study for movies' recommendation", Neurocomputing, vol. 176, **(2015)**, pp. 72-80.
[36] Z. Liu, N. Zhang and J. Li, "One Personal Recommendation Algorithm Based on Collaborative Filtering and Network Structure", Compandlex Systems and Complexity Science, vol. 8, no. 2, **(2011)**, pp. 45-50.
[37] M. Li and D. Xu, "Personalized Recommendation Algorithm Based on the Combination of Item and User", Journal of Chinese Computer Systems, vol. 32, no. 4, **(2011)**, pp. 611-613.
[38] B. Jeong, J. Lee and H. Cho, "An iterative semi-explicit rating method for building collaborative recommender systems", Expert Systems with Applications, vol. 36, no. 3, **(2009)**, pp. 6181-6186.

Authors

Baojun Tian, he received his B.S. degree in 1996 in computational mathematics of Inner Mongolia University, China, M.S. degree in 2003 in control theory and control engineering of Inner Mongolia University of Technology, China. He is currently an associate professor. His research interests include cloud computing, data mining, and recommendation system.