The Journal of Systems and Software xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Distributed collaborative filtering with singular ratings for large scale recommendation

Ruzhi Xu[a,b], Shuaiqiang Wang[a,b,*], Xuwei Zheng[b], Yinong Chen[c]

[a] Department of Information Engineering, Qilu University of Technology, 58 Sangyuan Road, Jinan 250100, China
[b] School of Computer Science and Technology, Shandong University of Finance and Economics, 7366 2nd East Ring Road, Jinan 250014, China
[c] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, P.O. Box 878809, Tempe, AZ 85287-8809, USA

ARTICLE INFO

Article history:
Received 7 January 2014
Received in revised form 24 April 2014
Accepted 25 April 2014
Available online xxx

Keywords:
Collaborative filtering
Recommender systems
Distributed framework

ABSTRACT

Collaborative filtering (CF) is an effective technique addressing the information overloading problem, where each user is associated with a set of rating scores on a set of items. For a chosen target user, conventional CF algorithms measure similarity between this user and other users by utilizing pairs of rating scores on common rated items, but discarding scores rated by one of them only. We call these comparative scores as *dual ratings*, while the non-comparative scores as *singular ratings*. Our experiments show that only about 10% ratings are dual ones that can be used for similarity evaluation, while the other 90% are singular ones. In this paper, we propose SingCF approach, which attempts to incorporate multiple singular ratings, in addition to dual ratings, to implement collaborative filtering, aiming at improving the recommendation accuracy. We first estimate the unrated scores for singular ratings and transform them into dual ones. Then we perform a CF process to discover neighborhood users and make predictions for each target user. Furthermore, we provide a MapReduce-based distributed framework on Hadoop for significant improvement in efficiency. Experiments in comparison with the state-of-the-art methods demonstrate the performance gains of our approaches.

1. Introduction

The Web has been flooded with an overwhelming amount of information, which represents one of today's major challenges of Web-based computing and information mining (Chen and Tsai, 2014). As an effective technique addressing the problem, recommender systems attempt to make predictions and give recommendations based on a large collection of users' historical behaviors and ratings. They have become a de facto standard and must-owned tools for e-commerce to promote business and to help customers discovering new products (Sarwar et al., 2000). Prominent examples include those systems used in Amazon (www.amazon.com), eBay (www.ebay.com/), Facebook (www.facebook.com/), LinkedIn (www.linkedin.com), and Netflix (www.netflix.com/).

Collaborative filtering (CF) (Chen et al., 2013; Herlocker et al., 2004; Su and Khoshgoftaar, 2009) is one of the most successful methods of building recommender systems. CF algorithms are based on the assumption that users will rate and act on other items similarly if they have rated items similarly or had similar behaviors (Goldberg et al., 1992; Resnick et al., 1994). CF utilizes the user–item rating matrix to make predictions and recommendations, avoiding the need of collecting extensive information about items and users. In addition, CF can be easily adopted in different recommender systems without requiring any domain knowledge (Liu and Yang, 2008).

Given the effectiveness and convenience, many CF methods have been proposed, which fall into two categories: memory-based (Deshpande and Karypis, 2004; Herlocker et al., 1999; Liu and Yang, 2008; Resnick et al., 1994; Sarwar et al., 2001; Wang et al., 2012) and model-based (Liu et al., 2009; Rendle et al., 2009; Shani et al., 2005; Si and Jin, 2003; Sun et al., 2012; Weimer et al., 2007). Memory-based methods make predictions based on similarities between users or items, while model-based methods make predictions based on a mathematical model.

In this study, we focus on the memory-based CF. Memory-based approach has been shown to be easy to implement, have strong robustness, and are effective (Hofmann, 2004). They take

* Corresponding author at: School of Computer Science and Technology, Shandong University of Finance and Economics, 7366 2nd East Ring Road, Jinan 250014, China. Tel.: +86 18253179913.
E-mail addresses: xrzpuma@gmail.com (R. Xu), shqiang.wang@gmail.com (S. Wang), xuwei_zheng@gmail.com (X. Zheng), yinong@asu.edu (Y. Chen).

http://dx.doi.org/10.1016/j.jss.2014.04.045
0164-1212/

the advantage of big data analysis with instance-based learning to make predictions. The impact of the noise data can be significantly reduced in the averaging process of large amount of data. This approach is particularly promising in commercial environments where large amount of data are available, and thus many commercial systems, such as Amazon.com, are memory-based (Linden et al., 2003).

In the existing memory-based CF algorithms, each user is associated with a set of scores on rated objects, either based on rated items or based on user preferences. These algorithms start with measuring the similarity between users by utilizing pairs of rating scores on commonly rated objects, but discarding scores rated by one of them only. In this paper, we called these comparative rating scores as *dual ratings*, while the non-comparative scores as *singular ratings*.

Our experimental results on two movie rating datasets show that only about 10% of ratings are dual ones that can be used for similarity evaluation in CF algorithms, while the other 90% are singular ones and are discarded in these approaches. Indeed, the singular rating data are less relevant to a target user. However, the amount matter in big data analysis. The more data we have, the more relevancies can be discovered. We believe that it is an important issue to explore a practical way of making full use of the majority of data resources, by adding the singular ratings into dual rating studies.

For this reason, we propose SingCF, which takes the advantage of the unused singular ratings to improve the accuracy of CF algorithms. We first process the unrated scores of singular ratings, find the correlations, and transform them into dual ones. Then we perform a CF process to discover neighborhood users and make predictions for each target user. In addition, we study the equivalence of the similarity measure and the prediction formula between rating-oriented and ranking-oriented CF algorithms. We prove that they are equivalent in principle by showing that the ranking-oriented algorithms can be obtained from the rating-oriented techniques.

In this paper, we implement two versions of SingCF for validation purpose, a rating-oriented and a ranking-oriented. Experimental results in comparison demonstrate that our approach outperform the existing methods.

All CF algorithms have a time complexity n^2, where n is the number of users, as the similarities between each pair of users need to be computed to discover potential neighborhood users for each target user to make predictions (Dean and Ghemawat, 2004). Furthermore, the users and data amount of the recommender systems increase rapidly, and most of them are stored in a distributed storage system for the scalability consideration (Lee and Chang, 2013). The solutions to the big data problem are the utilization of the distributed framework for CF algorithms.

In light of this requirement, in this paper, we also investigate the distributed framework for SingCF based on MapReduce in Hadoop, aiming to significantly improve the efficiency of SingCF.

Now, we discuss why SingCF can achieve better recommendation accuracy than conventional CF. The main difference between the two approaches is that SingCF uses additional singular ratings data. In fact, these data are valuable and useful, because 50% of the scores of these additional singular data are ground truth scores rated by users, and the other 50% of data are estimated scores. The mean errors of these data are quite low. Our study shows that the mean average errors of these estimated scores are only about 20% higher than that of the final predictions. As the result, SingCF achieves more accurate prediction than the dual CF algorithms.

The effectiveness of SingCF can be understood from another perspective. The unrated ratings of the singular ones are the missing values. Data cleaning is an essential technique in data mining, and one possible step is to attempt to fill in the missing values automatically with a measure of central tendency (Han et al., 2011).

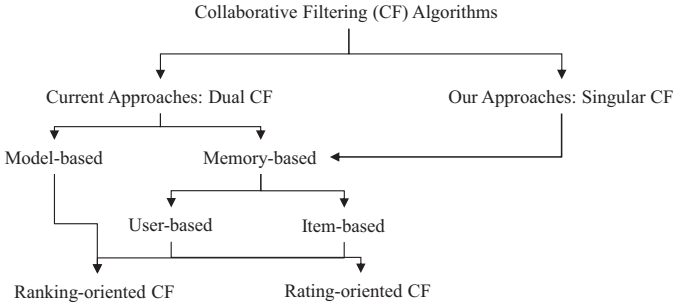

Fig. 1. Spectrum of different CF algorithms.

SingCF proposes an effective method to fill in the missing values using singular ratings, instead of pure speculation. It improves the prediction accuracy, which in turn improves the recommendation accuracy.

Our main contributions in this research include:

(1) We proposed SingCF, a collaborative filtering algorithm that incorporates singular ratings for improvement in recommendation accuracy, where we firstly estimated the unrated scores of singular ratings and transform them into dual ones as additional data for training and prediction.
(2) We proved that they were equivalent by showing that the ranking-oriented algorithms could be obtained from the rating-oriented techniques. Based on the same framework, we implemented two versions of SingCF for validation, a rating-oriented and a ranking-oriented for verification purpose.
(3) We provided DSingCF, a MapReduce-based distributed SingCF algorithm on Hadoop, aiming at significantly improving the efficiency of SingCF. Experiments in comparison with the state-of-the-art methods demonstrated the performance gains of our approaches.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 presents the preliminaries on memory-based collaborative filtering. Section 4 proposes the SingCF algorithm. Section 5 provides the distributed SingCF on Hadoop. Section 6 reports the experimental results. Section 7 concludes the paper.

2. Related work

2.1. Collaborative filtering

Fig. 1 gives the spectrum of different CF algorithms, as well as the position of the proposed SingCF algorithms in the spectrum.

Model-based CF algorithms use a mathematical model, a statistical model, or a learning model to analyze data and to make predictions on what a target user may purchase. Memory-based algorithms make predictions based on similarities of scores given by neighboring users and given to the related items. Many commercial systems, such as Amazon.com, use memory-based CF, as the approach is relatively easy to implement and results in good performance in recommendation.

The memory-based CF algorithms can be further categorized into two types: user-based and item-based (Wang et al., 2012). The user-based CF algorithms estimate the unknown ratings of a target user based on the ratings given by a set of neighboring users who tend to rate terms similarly, and thus, what they purchased may apply to the target user. On the other hand, in the item-based CF algorithms, item–item similarities are used to select a set of neighboring items that have been rated by the target user. In particular, given a target user/item, each user/item can be determined as her

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

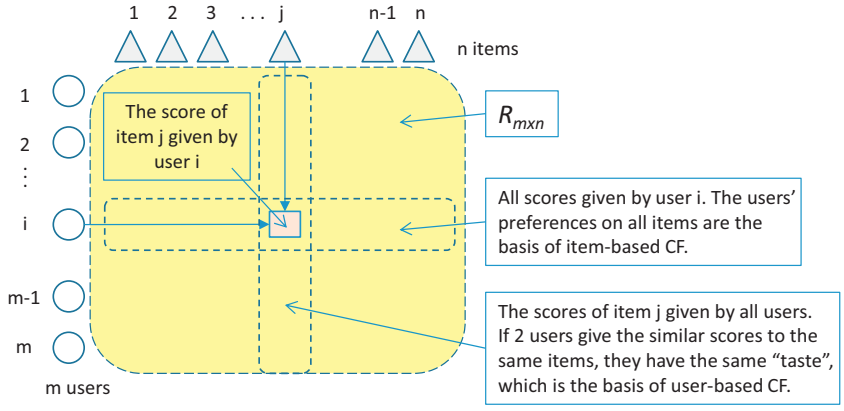

Fig. 2. The $R_{m\times n}$ matrix, with the scores of items given by users.

neighbor if the similarity between them is no less than a threshold. The ratings on the unrated items are predicted based on the user's ratings on the neighboring items.

Both rating-oriented CF and ranking-oriented algorithms can be memory-based or model-based. Rating-oriented CF considers the actual values of the rating scores, while ranking-oriented CF (McNee et al., 2006) ignores particular values of rating scores but maintains the ordered preference relationships between the items, which is more natural for recommendation tasks.

The difference between the two CF strategies can be explained through an example. Suppose the ground truths of the rating scores of items i_1 and i_2 from user u are 2 and 3, respectively. A rating-oriented CF algorithm could be evaluated as poor in performance if it predicts 4 and 5 for i_1 and i_2. However, from the point of view of ranking-oriented techniques, the predictions is very accurate because the preference relationship between i_1 and i_2, i.e., $i_1 < i_2$ is same as the ground truth.

Ranking-oriented CF generally utilizes Kendall tau correlation coefficient (Kendall, 1938) to measure similarity between each pair of users for prediction based on two users' preferences on the same set of items (Liu and Yang, 2008; Wang et al., 2012).

2.2. The MapReduce framework

MapReduce is a scalable parallel algorithm in distributed frameworks, which has found many applications in cloud computing and big data environments. It was first introduced by Google for counting the words in documents (Dean and Ghemawat, 2004), and currently it is regarded as one of the most crucial techniques in distributed frameworks such as Hadoop. Moreover, MapReduce has been successfully applied in traditional CF algorithms which require a large amount of parallel computing, leading to significant efficiency gain (Jiang et al., 2011; Lee and Chang, 2013; Schelter et al., 2013; Adomavicius and Tuzhilin, 2005; Chakravarthy et al., 2013).

3. Memory-based collaborative filtering

Generally, the memory-based CF can be either rating-oriented or ranking-oriented. They work in the following two phases: (I) discovery of neighborhood users and (II) prediction for recommendation. For each user, Phase I discovers a set of most similar users as the neighborhood users, and then, Phase II predicts rating scores or preferences on items for recommendation purpose.

For example, Pearson correlation coefficient (Herlocker et al., 2002; Resnick et al., 1994), a widely used similarity measure in rating-oriented CF, is based on two users' rating scores on the set of common items (Deshpande and Karypis, 2004; Herlocker et al., 1999; Resnick et al., 1994; Sarwar et al., 2001).

As our study is based on memory-based collaborative filtering (CF) approach, we present in this section the existing algorithms. These algorithms are studied formally, and they can be classified into two categories: rating-oriented and ranking-oriented, as categorized in Fig. 1.

3.1. Rating-oriented CF

Let U be a set of users and I be a set of items. In a recommender system, each user $u \in U$ gives scores on a set of items $I_u \subseteq I$, and each item $i \in I$ is rated by a set of users $U_i \subseteq U$. The scores can be represented as a matrix. Let $R_{m\times n}$ be a user–item rating matrix with m users and n items, where each element $r_{u,i}$? IN is the rating score of the item i with respect to u, and IN is the natural number set indicating different relevance scores. $\bar{r}_u$ is the mean score of user u over all the items rated by u. For a target user u, a set of neighborhood users $N_u \subseteq U$ are selected according to their similarity to u, on which the rating scores of the unrated items are predicted.

Fig. 2 illustrates the basis of correlating users in the user-based and item-based paradigms, rows are scores given by the same users, and columns are scores of the same item given by different users.

Rating-oriented CF recommends items based on historical rating scores of items. User-user paradigm is a widely used model for rating-oriented CF, where the Pearson correlation coefficient is used to evaluate the similarity $s_{u,v}$ between two users u and v with their normalized ratings on the set of common items $I_{u,v} = I_u \cap I_v$:

$$s_{u,v} = \frac{\sum_{i\in I_{u,v}}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I_{u,v}}(r_{u,i}-\bar{r}_u)^2\sum_{i\in I_{u,v}}(r_{v,i}-\bar{r}_v)^2}} \quad (1)$$

The rating of item i for user u can be predicted by the scores of i rated by a set of neighborhood users N_u of u:

$$r_{u,i}^{(P)} = \bar{r}_u + \frac{\sum_{v\in N_u} s_{u,v}(r_{v,i}-\bar{r}_v)}{\sum_{v\in N_u} s_{u,v}} \quad (2)$$

Item–item paradigm is an alternative model for rating-based CF, where the similarity between items can also be measured with Pearson correlation coefficient. The rating of item i for user u can be predicted by the scores of a set of neighborhood items I_u of i rated by u:

$$r_{u,i}^{(P)} = \frac{\sum_{j\in I_u} s_{i,j} r_{u,i}}{\sum_{j\in I_u} s_{i,j}} \quad (3)$$

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

3.2. Ranking-oriented CF

Ranking-oriented CF predicts users' preference ranking on items to make recommendations based on their relative preferences on common pairs of items (Liu and Yang, 2008; Wang et al., 2012).

In ranking-oriented CF, Kendall tau correlation coefficient (Kendall, 1938) is a common measure for evaluating similarity between two users based on users' preference scores on the same set of the pairwise preferences on items:

$$s_{u,v} = \frac{N_c - N_d}{(1/2)k(k-1)} \tag{4}$$

where N_c and N_d are the numbers of the concordant pairs and discordant pairs, respectively, and $N_c + N_d = (1/2)k(k-1)$.

Let $p_{u,(i,j)} \in \{+1, -1\}$ be the preference function, where $p_{u,(i,j)} = +1$ indicates user u prefers item i to j while $p_{u,(i,j)} = -1$ indicates u prefers j to i, formally:

$$p_{u,(i,j)} = \begin{cases} +1 & \text{if } r_{u,i} > r_{u,j} \\ -1 & \text{if } r_{u,i} < r_{u,j} \end{cases} \tag{5}$$

Let $N_{u,(i,j)}$ is the set of neighborhood users of u who hold certain preferences on the pair of items i and j. For user u, the preference on a pair of items (i, j) can be predicted with a preference function $\Psi_u(i, j)$ as follows:

$$\Psi_u(i,j) = \frac{\sum_{v \in N_{u,(i,j)}} s_{u,v} p_{u,(i,j)}}{\sum_{v \in N_{u,(i,j)}} s_{u,v}} \tag{6}$$

Based on the predicted pairwise preferences, a total ranking of items τ for user u can be obtained by optimizing the following problem:

$$\max \sum_{\forall (i,j): \tau(i) < \tau(j)} \Psi_u(i,j) \tag{7}$$

A greedy strategy can be made for this optimization problem, which has a time complexity of $O(n^2)$, where n is the number of the items. It can be proved to have an approximation ratio of 2, i.e., $\Lambda(\tau) \geq 1/2\Lambda(\tau^*)$ (Cohen et al., 1999).

4. The SingCF algorithm

In this section, we present our approach. We first investigate the equivalence between the two paradigms of CF algorithms, rating-oriented and ranking-oriented. We develop our algorithm SingCF based on rating-oriented CF. The ranking-oriented version of SingCF can be derived from the equivalence between two CF paradigms.

4.1. Ranking-oriented CF vs. rating-oriented CF

In ranking-oriented CF, each user u is represented as a set of relative preferences on pairs of items. In this study we propose another representation based on user–(item–item) preference matrix, which is defined as follows:

Definition 1. Let $R_{m \times n}$ be the user–item score matrix with m users and n items. The matrix $P_{m \times n(n-1)}$ is called a user–(item–item) preference matrix with m users and $n(n-1)$ preferences on all possible pairs of items, where each element $p_{u,(i,j)} = \{+1, -1\}$ indicates the preference of the target user u on the pair of items i and j, as shown in Eq. (5).

According to the definition of the user–(item–item) preference matrix $P_{m \times n(n-1)}$, the following corollary is satisfied:

Corollary 1. *The average rating scores of each user is 0, formally*:

$$\forall u \in U : \bar{p}_u = 0$$

Since for each user $u \in U$ who has rated a pair of items i and j, $p_{u,(i,j)} + p_{u,(j,i)} = 0$, and thus Corollary 1 holds.

Theorem 1. *The ranking-oriented similarity measure of the Kendall tau correlation coefficient based on the common preference sets is equivalent to the Pearson correlation coefficient based on the representation of the user–(item–item) preference matrix.*

The proof is provided in Appendix.

Theorem 2. *The ranking-oriented prediction formula based on the common preference sets is equivalent to the rating-oriented prediction formula based on the representation of the user–(item–item) preference matrix.*

The proof is provided in Appendix.

In this section, we mainly introduce our algorithm SingCF based on the rating-oriented CF. The ranking-oriented version can be easily implemented based on the equivalence of the similarity measure and the prediction formula between ranking-oriented and rating-oriented CF algorithms.

4.2. Estimation for singular rating

In existing memory-based CF algorithms, the similarity between users is evaluated by dual ratings without considering any singular one. The dual ratings are pairs of scores on certain items rated by a pair of users, and the singular ratings are scores rated by either of two users.

Definition 2. Let I_u and I_v be two sets of items rated by user u and v, respectively. Let $I_{u,v} = I_u \cap I_v = \{i_1, i_2, \ldots, i_k\}$ be the common set of items rated by users u and v. All pairwise rating scores rated by u and v on $I_{u,v}$ are called dual ratings, i.e., $\{(r_{u,i_1}, r_{v,i_1}), (r_{u,i_2}, r_{v,i_2}), \ldots, (r_{u,i_k}, r_{v,i_k})\}$. The rating scores rated by u on items $I_u/I_{u,v}$ and those by v on $I_v/I_{u,v}$ are called singular ratings.

Example 1. Let $\{i_1, i_2, i_3, i_4\}$ be four items. Let $\{u, v\}$ be two users, where their rating scores are listed as follows:

$$\begin{cases} r_{u,i_1} = 0, r_{u,i_2} = 1, r_{u,i_3} = 2 \\ r_{v,i_2} = 3, r_{v,i_3} = 4, r_{v,i_4} = 5 \end{cases}$$

According to our definitions, $(r_{u,i_2}, r_{v,i_2}) = (1, 3)$ and $(r_{u,i_3}, r_{v,i_3}) = (2, 4)$ are dual ratings. $r_{u,i_1} = 0$ and $r_{v,i_4} = 5$ are singular ones.

For incorporating singular ratings into CF, the most straightforward way is to estimate their corresponding unrated scores and transform the singular ratings into dual ones. In Example 1, r_{v,i_1} and r_{u,i_4} are corresponding unrated scores of r_{u,i_1} and r_{v,i_4}, respectively.

In doing this, accurate estimation is very important in SingCF, which can avoid introducing plenty of noises into the rating matrix R.

In this study, we fully consider the historical rating scores assigned by the target user u to items I_u, and the scores assigned by users U_i to the target item i:

$$\hat{r}_{u,i} = \hat{r}_u + \frac{\sum_{v \in U_i} (r_{v,i} - \overline{r_v})}{|U_i|} \tag{8}$$

$$\tilde{r}_{u,i} = \frac{\sum_{i \in I_u} r_{u,i}}{|I_u|} = \bar{r}_u \tag{9}$$

In particular, Eqs. (8) and (9) are special cases of Eqs. (2) and (3), respectively, where all users U_i who have rated i are considered as the neighborhood users of u, all items I_u rated by u as the neighborhood items of i, and all similarities are 1.

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

In SingCF, we linearly combine $\hat{r}_{u,c}$ and $\tilde{r}_{u,c}$ to estimate the unrated ratings for singular data:

$$r_{u,i}^{(E)} = \alpha \cdot \hat{r}_{u,i} + (1-\alpha)\cdot \tilde{r}_{u,i} = \alpha \cdot \left(\bar{r}_u + \frac{\sum_{v \in U_i}(r_{v,c} - \overline{r_v})}{|U_i|} \right)$$
$$+ (1-\alpha)\cdot \bar{r}_u = \bar{r}_u + \alpha \cdot \frac{\sum_{v \in U_i}(r_{v,i} - \bar{r}_v)}{|U_i|} \quad (10)$$

where U_i is the set of users who has rated i, α is a real number and $0 \le \alpha \le 1$.

An interesting observation is that Eq. (10) is similar to Eq. (2), where the first part measures the average scores assigned by u, and the second part evaluates the quality of i, whether its rating is higher than the average score or lower. These two equations are the same when the following conditions hold:

- The similarity between u and each user who has rated i is "1" Formally, $\forall v \in U_i : s_{u,v} = 1$.
- Each user who has rated i is considered as a neighborhood user of u. Formally, $\forall v \in U_i : v \in N_u$, and $N_u \equiv U_i$.
- The parameter α in Eq. (10) is "1".

The first two conditions are easily understood. If we have no prior knowledge on similarity between users, we have to assign each to "1". Thus each user who has rated i is considered as a neighbor of u.

Since not all users in U_i are similar to the target user u, we lack of full confidence on the quality estimation on i, and α can be regarded as a confidence rate.

Thus if the first two conditions are satisfied, Eq. (10) is a generalization of (2), and they are the same if the confidence rate $\alpha = 1$.

Algorithm 1: Rating-oriented SingCF

Input: A set of users U, a set of items I, an incomplete rating score matrix R, a similarity threshold θ
Output: Estimated scores for each unknown ratings in R
Begin
1 **for each** $u \in U$
2 **for each** $i \in R$
3 **if** $r_{u,i}$ is unknown **then** calculate $r_{u,i}^{(E)}$ with Eq. (10)
4 **end for**
5 **end for**
6 **for each** $u \in U$
7 $N_u \leftarrow \varnothing$
8 **for each** $v \in U \backslash \{u\}$
9 Calculate $s_{u,v}$ with Eq. (1)
10 **if** $s_{u,v} \ge \theta$ **then** $N_u \leftarrow N_u \cap \{v\}$
11 **end for**
12 **end for**
13 **for each** $u \in U$
14 **for each** $i \in R$
15 **if** $r_{u,i}$ is unknown **then** calculate $r_{u,i}^{(P)}$ with Eq. (2)
16 **end for**
17 **end for**
End

The pseudocode of rating-oriented SingCF algorithm is shown in Algorithm 1. In particular, lines 1–5 estimate the unrated scores $r_{u,i}$ of singular ratings with Eq. (10) and transform them into dual ones. Then lines 6–17 performs a rating-oriented CF process. In particular, lines 6–12 discover a set of neighborhood users N_u for each target user u based on their similarities to user u, where the similarity is measured by Eq. (1). Lines 13–17 predict each unknown rating score $r_{u,i}^{(P)}$ on the item i by the user u with Eq. (2). The time complexity of lines 1–5 is $O(mn)$, and the time complexity of 6–17 is $O(m^2 n)$ thus the time complexity of Algorithm 1 is $O(m^2 n)$, where m and n are the numbers of users and items, respectively.

Similarly, for ranking-oriented CF, let $U_{(i,j)}$ be is the set of users who hold certain preferences on items i and j, i.e., $U_{(i,j)} \subseteq U_i \cap U_j \wedge \forall u \in U_{(i,j)} \rightarrow p_{u,(i,j)} \neq 0$. According to Corollary 1, for each user, $\overline{p}_u = 0$. The estimating formula of Eq. (10) can be rewritten as follows:

$$f(U_{(i,j)}) = \alpha \cdot \frac{\sum_{v \in U_{(i,j)}} p_{v,(i,j)}}{|U_{(i,j)}|}$$

where, $f(U_{(i,j)})$ returns a real number while the preference on i, j should be +1 or −1. Let θ be a threshold where $0 \le \theta \le 1$. The preference on (i,j) can be estimated as follows:

$$p_{u,(i,j)}^{(E)} = \begin{cases} +1 & \text{if } f(U_{(i,j)}) > \theta \\ -1 & \text{if } f(U_{(i,j)}) < -\theta \end{cases} \quad (11)$$

Then, these pairwise preferences can be aggregated into a total ranking of items based on greedy algorithm, random walking (Liu and Yang, 2008), and the Schulze method (Wang et al., 2012).

The pseudocode of ranking-oriented SingCF algorithm is shown in Algorithm 2. In particular, lines 1–7 estimate the unrated scores $p_{u,(i,j)}$ of singular preferences with Eq. (11) and transform them into dual ones. Then lines 8–29 performs a ranking-oriented CF process. In particular, for each user u, lines 8–14 transform the rating scores into the relative preferences on pairs of items with Eq. (5). Lines 15–21 discover a set of neighborhood users N_u for each target user u based on their similarities to user u, where the similarity is measured by Eq. (4). Lines 22–27 predict each unknown relative preference $p_{u,(i,j)}$ on the pair of items (i,j) by the user u with Eq. (6). Line 28 predicts a total ranking τ on the set of items I for u by optimize the problem defined by Eq. (7). Obviously, the time complexity is $O(m^2 n^2)$, where m and n are the numbers of users and items, respectively. It is easy to see that the time complexity of ranking-oriented CF is n times larger than rating-oriented CF.

Algorithm 2: Ranking-oriented SingCF

Input: A set of users U, a set of items I, an incomplete rating score matrix R, a similarity threshold θ
Output: Estimated scores for each unknown ratings in R
Begin
1 **for each** $u \in U$
2 **for each** $i \in R$
3 **for each** $j \in R \backslash \{i\}$
4 **if** $p_{u,(i,j)}$ is unknown **then** calculate $p_{u,(i,j)}^{(E)}$ with Eq. (11)
5 **end for**
6 **end for**
7 **end for**
8 **for each** $u \in U$
9 **for each** $i \in R$
10 **for each** $j \in R \backslash \{i\}$
11 Calculate $p_{u,(i,j)}$ with Eq. (5)
12 **end for**
13 **end for**
14 **end for**
15 **for each** $u \in U$
16 $N_u \leftarrow \varnothing$
17 **for each** $v \in U \backslash \{u\}$
18 Calculate $s_{u,v}$ with Eq. (4)
19 **if** $s_{u,v} \ge \theta$ **then** $N_u \leftarrow N_u \cap \{v\}$
20 **end for**
21 **end for**
22 **for each** $u \in U$
23 **for each** $i \in R$
24 **for each** $j \in R \backslash \{i\}$
25 **if** $p_{u,(i,j)}$ is unknown **then** calculate $\Psi_u(i,j)$ with Eq. (6)
26 **end for**
27 **end for**
28 Output τ for u by optimize the problem defined by Eq. (7)
29 **end for**
End

4.3. Discussion

The effectiveness of SingCF can be understood from the following two perspectives.

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

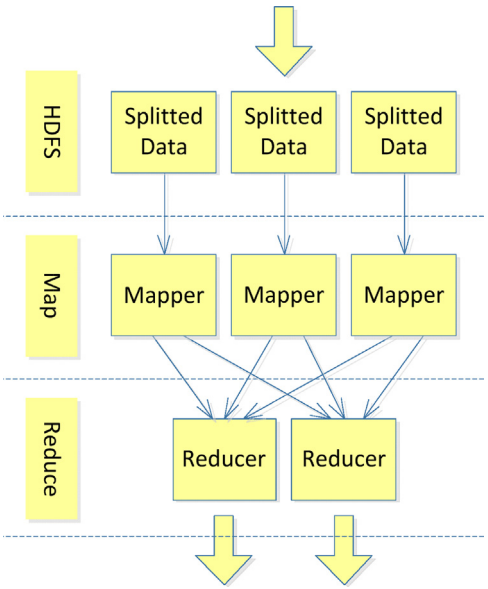

Fig. 3. The MapReduce framework.

- *Utilizing additional high-quality data*. Among the additional data introduced by SingCF, one half of them are the ground truth scores rated by users, while the mean errors of the estimated scores in the other half group are quite low, which are only about 20% higher than that of the final predictions. As the result, SingCF achieves more accurate prediction than dual CF.
- *Data cleaning*. The unrated ratings of the singular ones are the missing values. Data cleaning is an essential technique in data mining, and one possible step is to attempt to fill in the missing values automatically with a measure of central tendency (Han et al., 2011). SingCF proposes an effective method to fill in the missing values for singular ratings and thus improving predicting accuracy, which in turn improving recommendation accuracy.

SingCF works in two phases: (I) data cleaning, and (II) collaborative filtering. Phase I estimates the unrated rating scores and transforms singular ratings into dual ones. Then for each user, Phase II discovers a set of most similar users as the neighborhood users to make predictions and recommendations.

5. The DSingCF algorithm

5.1. MapReduce and Hadoop

MapReduce is a built-in model in Hadoop for distributed computing, which was initially applied by Google (Dean and Ghemawat, 2004) for counting the number of the terms that occur in documents. It perfectly fits the situation where data are already stored in a distributed file system, which allows performing computations locally on each data node. Fig. 3 shows the MapReduce framework used for such purposes.

First, all of the data are split into a collection of datasets, which are stored in the Hadoop Distributed File System (HDFS). MapReduce consists of the *Map* components and the *Reduce* components. Both components utilize the $\langle key, value\rangle$ structure as their inputs and outputs, in both *Map* and *Reduce* phases, where *key* is a unique ID used for partitioning data (values) and value is a list values identified by the key. More specifically, assume the job is partitioned into x pieces, for each piece p, where $p = 0, 1, 2, \ldots, x-1$.

- One Map node takes a key-value pair $\langle k_{p1}, v_{p1}\rangle$ and generates a new pair $\langle k_{p2}, v_{p2}\rangle$. All Map nodes will generate a list $\langle k_{p2}, v_{p2}\rangle$, where $p = 0, 1, 2, \ldots, x-1$

$$\textbf{Map}: \quad \langle k_{p1}, v_{p1}\rangle \xrightarrow{\text{yields}} \langle k_{p2}, v_{p2}\rangle$$

- One Reduce node takes n outputs from n Map nodes and generates a reduced (smaller) list: list $\langle k_{q3}, v_{q3}\rangle$, where $q = 0, 1, 2, \ldots, y-1$, $y < x$.

$$\textbf{Reduce}: \quad \langle k_{p2}, v_{p2}\rangle \xrightarrow{\text{yields}} \text{list}\langle k_{q3}, v_{q3}\rangle$$

In the rest of the discussion on MapReduce, we will not explicitly use the indices p and q to simplify the notation.

5.2. The distributed SingCF on Hadoop

In this section, we present DSingCF, the MapReduce-based distributed SingCF algorithm on Hadoop, aiming at significantly speedup the execution of SingCF.

As mentioned in Section 4.3, SingCF works in two phases: (I) data cleaning and (II) collaborative filtering. Accordingly, we introduce two MapReduce-based distributed phases in DSingCF. The framework of the DSingCF is shown in Fig. 4.

Phase I of DSingCF tries to estimate the unrated rating scores and transforms singular ratings into dual ones. The *Map* and *Reduce* execution is organized as follows. There are two MapReduce procedures. The first procedure (Map-1 and Reduce-1) calculates the average rating score of each user $\bar{r}_u$. The second procedure (Map-2 and Reduce-2) estimates the unrated rating scores for each user–item pair $r_{u,i}^{(E)}$ using Eq. (10), where the average scores of the target user u and her neighborhood users are provided from the first procedure.

- **Map-1**: As shown in Fig. 4(a), each *Map* node accepts a tuple $\langle(u,i),r_{u,i}\rangle$ as the input, where the pair (u, i) of the user u and the item i form the *key* part, and the rating score $r_{u,i}$ is the *value* part. Then it outputs $\langle u,(i,r_{u,i})\rangle$ for the *Reduce* component, where the user u is the *key* part, and the pair $(i, r_{u,i})$ is the *value* part.
- **Reduce-1**: Each *Reduce* node accepts the outputs $\langle u,(i,r_{u,i})\rangle$ of the *Map* nodes, and lists them together as $\langle u,\text{list}(i,r_{u,i})\rangle$, where the user u is the *key* part, and the list of the pairs $(i, r_{u,i})$ is the *value* part. Then it estimates the average rating scores $\bar{r}_u$ for each user u.
- **Map-2**: Each *Map* node accepts a tuple $\langle(u,i),r_{u,i}\rangle$ as the input, where the pair (u, i) of the user u and the item i is the *key* part, and the rating score $r_{u,i}$ is the *value* part. Then it outputs $\langle i,(u,r_{u,i})\rangle$ for the *Reduce* component, where the user i is the *key* part, and the pair $(u, r_{u,i})$ is the *value* part.
- **Reduce-2**: Each *Reduce* node accepts the outputs $\langle i,(u,r_{u,i})\rangle$ of the *Map* nodes, and lists them together as $\langle i,\text{list}(u,r_{u,i})\rangle$, where the user i is the *key* part, and the list of the pairs $(u, r_{u,i})$ is the *value* part. Then it calculates each unrated rating score $r_{u,i}^{(E)}$ using Eq. (10) to obtain output.

Phase II of the DSingCF discovers a set of most similar users as the neighborhood users to make predictions and recommendations for each user. First, it collects the triples $\langle u,i,r_{u,i}\rangle$ of each user u to form a user vector u. Based on the vector representation, the *Map* and *Reduce* strategies are listed as follows.

- **Map**: As shown in Fig. 4(b), the *Map* component accepts each pair of user vectors $\langle(u,v),(u,i)\rangle$ as the input, where the pair of the user

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

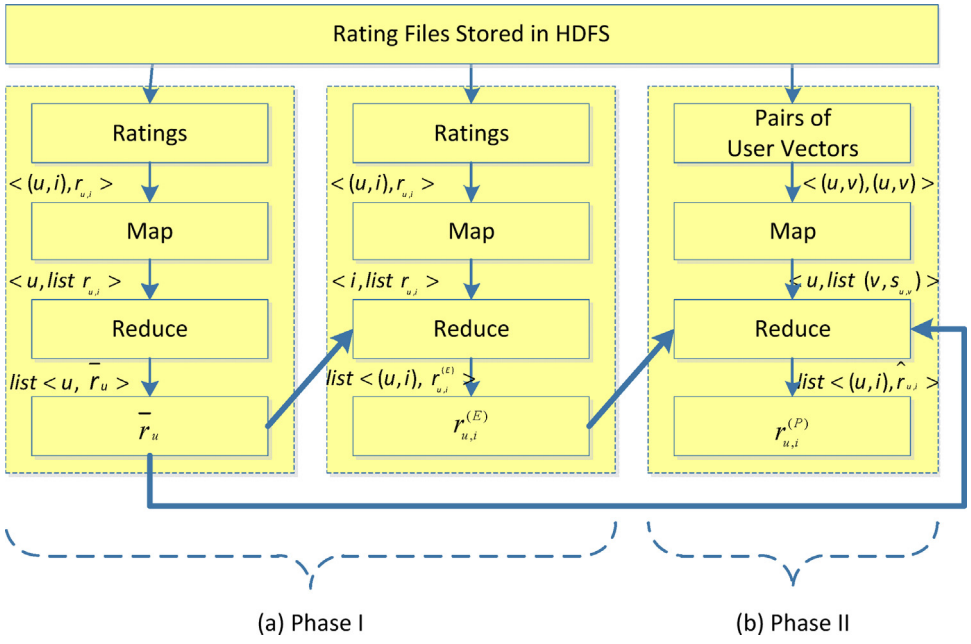

Fig. 4. The MapReduce-based distributed SingCF on Hadoop.

index (u, v) is the *key* part, and the pair of their vectors (u, i) is the *value* part. Then it computes the similarity $s_{u,v}$ between u and v and outputs $\langle u,(v,s_{u,v})\rangle$ for the *Reduce* component, where the user u is the *key* part, and the pair $(v,s_{u,v})$ is the *value* part.

- **Reduce**: Each *Reduce* component accepts the outputs $\langle u,(v,s_{u,v})\rangle$ of the *Map* components, and lists them together as $\langle u,\text{list}(v,s_{u,v})\rangle$, where the user u is the *key* part, and the list of the pairs $(v,s_{u,v})$ is the *value* part. Then it predicts each unrated rating score $r^{(P)}_{u,i}$ using Eq. (2) to obtain output.

Note that the above presentations are for the rating-oriented CF implementation of the DSingCF. For the ranking-oriented CF implementation, in the Phase I, the input of the *Map* component is $\langle(u,i,j),p_{u,(i,j)}\rangle$, and the output is $\langle u(i,j,p_{u,(i,j)})\rangle$; the input of the *Reduce* component is $\langle u,list(i,j,p_{u,(i,j)})\rangle$, and the output is the estimation of each preference $p^{(E)}_{u,(i,j)}$ using Eq. (11). In the Phase II, the input of the *Map* component is $\langle(u,v),(u,i)\rangle$, where the vector of each user is composed of the pairwise preferences of the items. The output of the *Map* component is $\langle u,(v,s_{u,v})\rangle$; The input of the *Reduce* component is $\langle u,\text{list}(v,s_{u,v})\rangle$, and the output is the estimation of each preference $p^{(P)}_{u,(i,j)}$ using Eq. (7).

6. Experiments

We use MovieLens (http://www.grouplens.org/datasets/movielens/), a real movie rating benchmark in our experiments. There are two datasets: the smaller one names ML-100K, comprising around 100,000 rating scores and the bigger one names ML-1M, comprising around 1,000,000 rating scores. In the experiments, we randomly selected 80% rated items for training and used the remaining 20% for testing the accuracy of our algorithms. In other words, we assume that these 20% of data are unknown, and we try to predict and make recommendation. In order to make sure that there are adequate percentage of common rating items between each neighborhood user and the target user, we filtered out those users who have rated less than 30 items. Our Hadoop platform is constructed running on 64-bit Linux Ubuntu 12.04 operation system, where the master has an Intel i5 3.2 GHz processor and 8G memory, and the slaver has an Intel e7500 2.94 GHz processor and 4G memory. The Hadoop version is 0.20.2. We run each algorithm five times, and we calculate the average performance.

6.1. Performance

In this section, we evaluate the performance (execution time) gains of DSingCF by the MapReduce-based distributed framework on Hadoop. Particularly, we run the two implementations of DSingCF, rating-oriented and ranking-oriented, on the ML-1M dataset using a varying number of computers (1, 2, 4, 8, 16, and 32).

Figs. 5 and 6 show the execution time and their corresponding relative speed-up. From the results we can see that the MapReduce-based distributed framework leads to significant speedup in DSingCF. Comparing to 1 computer, the averaged relative speed-up ratios go from 1.9 to 17.7 for the rating-oriented implementation

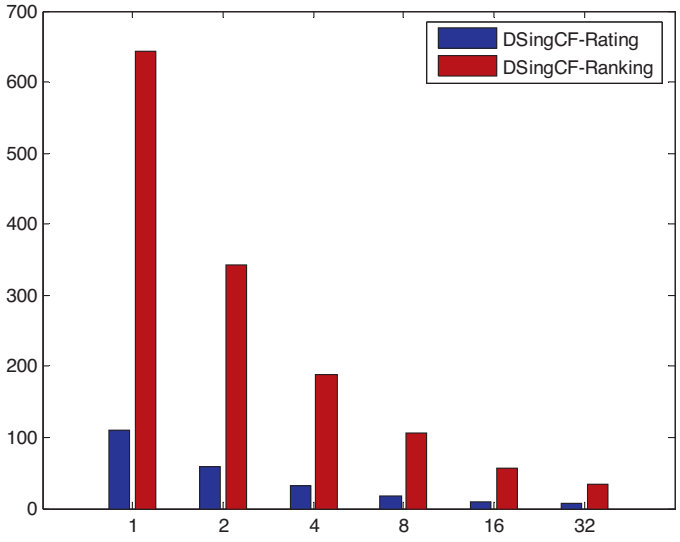

Fig. 5. Execution time (1000 s) of the DSingCF algorithms with respect to different number of computers.

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

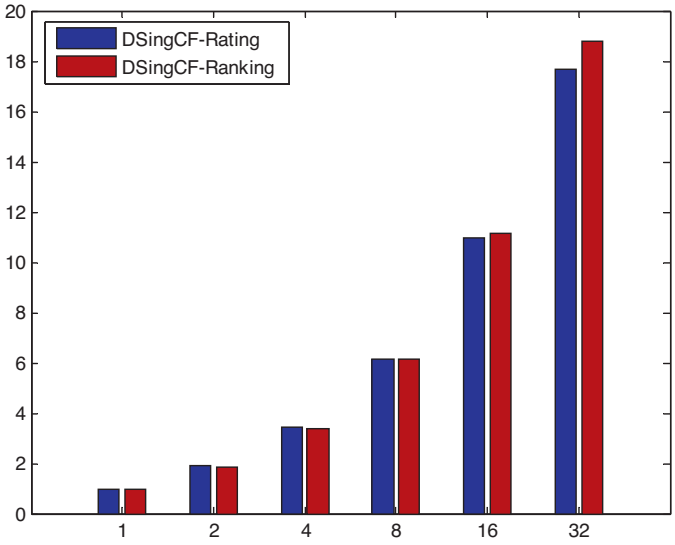

Fig. 6. The speed-up ratios of DSingCF algorithms with respect to different number of computers.

of DSingCF. These numbers are from 1.9 to 18.8 for the ranking-oriented implementation of DSingCF.

Figs. 7 and 8 demonstrate the scalability of the DSingCF algorithms running with 1/32, 1/16, 1/8, 1/4, 1/2, and the full numbers of users, respectively. From the figures we can see that the MapReduce framework did not change the time complexity of the original algorithms. Since the time complexity of rating-oriented SingCF is $O(m^2n)$ while the complexity of ranking-oriented SingCF is $O(m^2n^2)$, the averaged relative speed-up ratio of DSingCF with respect to 1/2 number of users trends to 4 in comparison with the total number of users.

6.2. Accuracy

6.2.1. Rating-oriented CF

For rating-oriented collaborative filtering (CF), we use two widely used evaluation criteria, Mean Absolute Error (*MAE*) and Root Mean Squared Error (*RMSE*), to evaluate our algorithm.

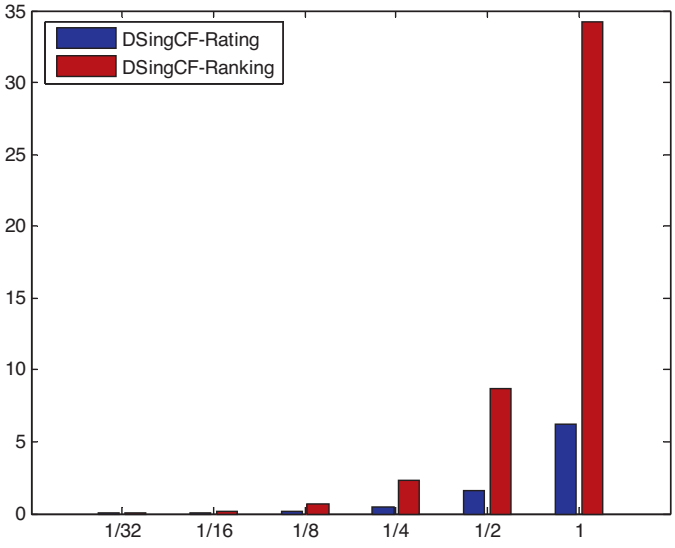

Fig. 7. Execution time (1000 s) of DSingCF algorithms with respect to different volume of data.

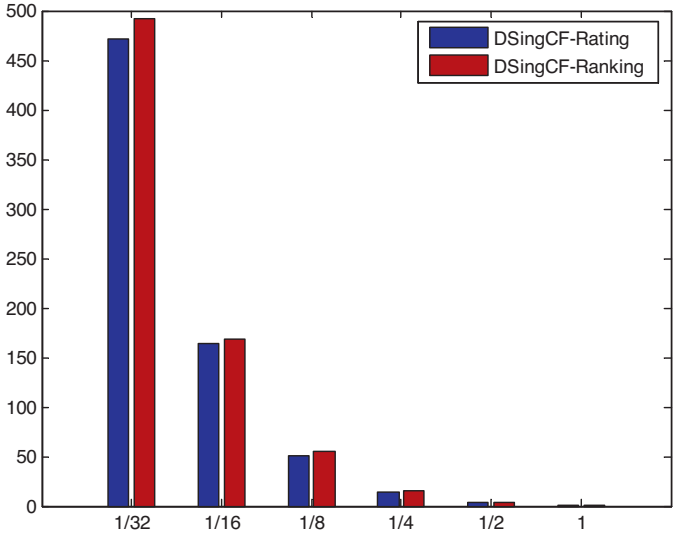

Fig. 8. The speed-up ratios of DSingCF algorithms with respect to different volume of data.

Table 1
Performance comparison on ML-100K.

Algorithm	*MAE*	*RMSE*
CF	0.9798	1.1839
SingCF	0.9199	0.9546

MAE is the most widely used evaluation metric in CF research. Let R_T be the test data, where each element $r_{u,i}^{(T)} \in R_T$ is the score assigned by user u to item i. *MAE* computes the average of the absolute difference between the predictions and true ratings:

$$MAE = \frac{\sum_{r_{u,i}^{(T)} \in R_T} |r_{u,i}^{(T)} - r_{u,i}^{(P)}|}{|R_T|}$$

RMSE is another popular metric, which is used in the Netflix Prize (http://www.netflixprize.com/) for movie recommendation performance:

$$RMSE = \sqrt{\frac{\sum_{r_{u,i}^{(T)} \in R_T} (r_{u,i}^{(T)} - r_{u,i}^{(P)})^2}{|R_T|}}$$

We choose a conventional user-based CF method as our comparison basis, which measures similarity between users using the Pearson correlation coefficient and predicts ratings using Eq. (2). Our implementation of SingCF is based on the conventional CF. A direct comparison of the two will provide valuable and irreplaceable insights.

Tables 1 and 2 show the performance comparisons under the *MAE* and *RMSE* measures. From the tables we can see that our proposed SingCF algorithm can achieve significant accuracy improvement, resulting from effective incorporation of singular ratings. For example, For the ML-100K dataset, the *MAE* and *RMSE* of the SingCF are 0.9199 and 0.9546, gaining 6.1% and 19.4% improvement, respectively. For the ML-1M dataset, the *MAE* and *RMSE* of the

Table 2
Performance comparison on ML-1M.

Algorithm	*MAE*	*RMSE*
CF	0.9467	1.0546
SingCF	0.9038	0.9339

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

SingCF are 0.9038 and 0.9339, gaining 4.5% and 11.4% improvement, respectively.

6.2.2. Ranking-oriented CF

For ranking-oriented CF, the widely used measurement is the Normalized Discounted Cumulative Gain (*NDCG*) (Järvelin and Kekäläinen, 2002) metric, which is popular in information retrieval for evaluating ranked document results.

In the context of collaborative filtering, item ratings assigned by users can naturally serve as graded relevance judgments. Specifically, the *NDCG* metric is evaluated over certain number n of the top items on the ranked item list.

Let U be the set of users and $r_{u,p}$ be the rating score assigned by user u to the item at the pth position of the ranked list from u. The average *NDCG* at the nth position with respect to all users U is defined as follows.

The value of *NDCG* ranges from 0 to 1. A higher value indicates better ranking effectiveness. *NDCG* is sensitive to the ratings of the highest ranked items. This is modeled by the discounting factor $\log(1+p)$ that increases with the position in the ranking. This is a highly desirable characteristic for evaluating ranking quality in recommender systems, because, same as Web search, most users examine the first few items from the recommended list only. The top-ranked items are far more important than other items (Liu and Yang, 2008).

We use EigenRank (Liu and Yang, 2008), a state-of-the-art ranking-based CF algorithm, as our main comparison basis. Our implementation of SingCF is based on EigenRank. A direct comparison of the two will provide valuable and irreplaceable insights. Besides, we also use CoFiRank, another state-of-the-art ranking-based collaborative filtering algorithm as our comparison basis. In particular, EigenRank measures similarity between users with Kendall tau rank correlation coefficient for neighborhood selection, predicts pairwise preferences of items with a preference function, and aggregates the predicted preferences into a total ranking using a greedy algorithm. CoFiRank uses Maximum Margin Matrix Factorization and employs structured output prediction to directly optimize ranking scores instead of ratings. In addition, we also include comparisons with UVS, a conventional user-based CF method, which measures similarity between users using the Pearson correlation coefficient, predicts ratings using Eq. (2), and then ranked the items for each user according to their predicted rating scores for the purpose of obtaining a ranking of items.

Figs. 9 and 10 show the performance comparisons under the NDCG metric. From the figures we can see that our SingCF outperforms all comparison bases. For ML-100K, SingCF achieved the best performances on NDCG@3–5, and the second best performances on NDCG@1–2, only slightly lower than CoFiRank but significantly higher than EigenRank and UVS. For ML-1M, SingCF achieved the best performances on NDCG@2–5, and the second best performances on NDCG@1, only slightly lower than CoFiRank, but significantly higher than EigenRank and UVS.

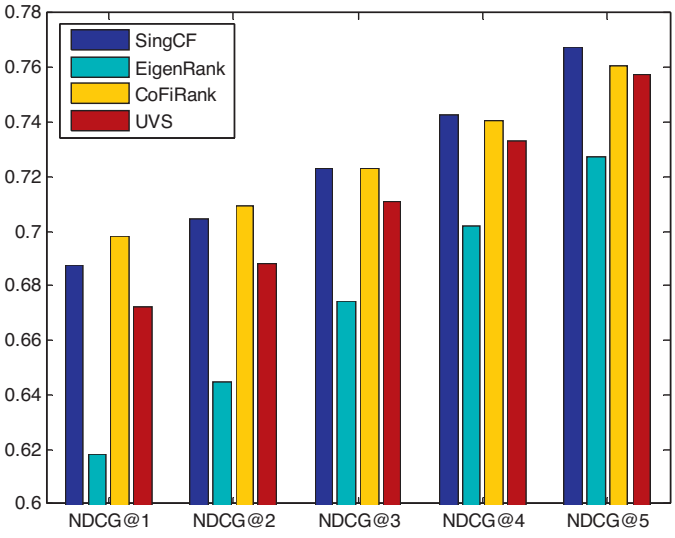

Fig. 9. Performance comparison on ML-100K.

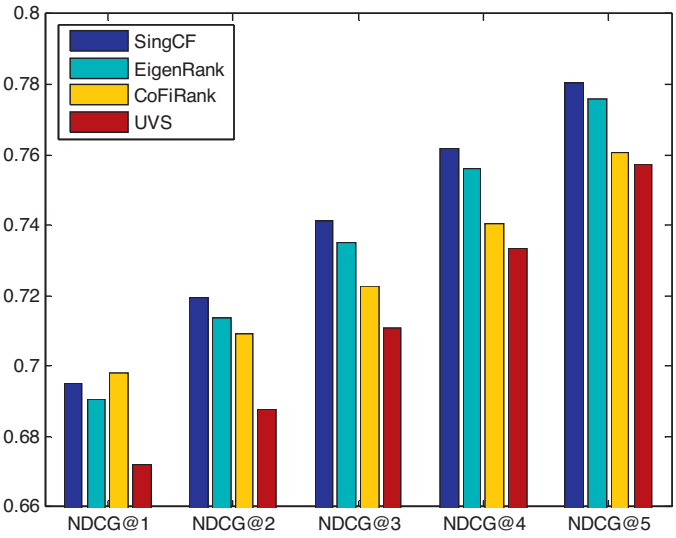

Fig. 10. Performance comparison on ML-1M.

7. Conclusion

In this paper, we proposed an extended collaborative filtering technique, SingCF, that incorporated singular ratings for making full use of the limited data resources and improving recommendation accuracy. In particular, we first estimated the unrated ratings and transform the singular ratings into dual ones. Then we performed the CF process to discover neighborhood users and made more accurate predictions for recommendations. We also showed the equivalence between the two paradigms of CF algorithms: rating-oriented and ranking-oriented. Based on the equivalence, we developed two versions of SingCF, for rating-oriented and ranking-oriented CF, respectively. Furthermore, we developed a MapReduce-based distributed computing framework on Hadoop, which significantly improve recommendation efficiency. Experiments validated the effectiveness of our algorithms.

There are several promising directions for future work. First, we plan to perform a systematic study on SingCF, investigating the various factors that may affect its performance. Second, it is interesting to study other possible models for estimating the unrated ratings for singular data. Finally yet importantly, the proposed algorithm is not limited to ranking-based CF, and we plan to adapt it to model-based CF and examine its effectiveness.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (61272240 and 71171122), the Humanity and Social Science Foundation of Ministry of Education of China (12YJC630211), the Specialized Research Foundation of Ministry of Education of China for Returned Overseas Scholars (20124501), the Natural Science Foundation of Shandong Province of China (2012BSB01550), the Specialized Research Foundation of Jinan for

Returned Overseas Scholars (20120201), and the Startup Fund for New Faculties of Shandong University of Finance and Economics (20110301).

Appendix.

Proof of Theorem 1

According to the definition of the user–(item–item) preference matrix $p_{m\times n(n-1)}$, $p^2_{u,(i,j)} = 1$.

Let I_C be the set of items where both u and v hold certain preference on each pair of items, formally, $I_C \in I_{u,v} \wedge \forall i,j \in I_C \rightarrow p_{u,(i,j)}p_{v,(i,j)} \neq 0$. Based on Corollary 1, the Pearson correlation coefficient (Eq. (1)) can be rewritten as follows:

$$s_{u,v} = \frac{\sum_{i,j\in I_C}(p_{u,(i,j)} - \bar{p}_u)(p_{v,(i,j)} - \bar{p}_v)}{\sqrt{\sum_{i,j\in I_C}(p_{u,(i,j)} - \bar{p}_u)^2}\sqrt{\sum_{i,j\in I_C}(p_{v,(i,j)} - \bar{p}_v)^2}}$$

$$= \frac{\sum_{i,j\in I_C} p_{u,(i,j)}p_{v,(i,j)}}{\sqrt{\sum_{i,j\in I_C} p^2_{u,(i,j)}}\sqrt{\sum_{i,j\in I_C} p^2_{v,(i,j)}}} = \frac{\sum_{i,j\in I_C} p_{u,(i,j)}p_{v,(i,j)}}{k(k-1)}$$

For a target pair of items (i, v) and two users u and v, $p_{u,(i,j)}p_{v,(i,j)}$ can be considered as an indicator function, such that it is equal to +1 if the preference on items i and j is concordant in users u and v while −1 if the preference is discordant. Formally,

$$p_{u,(i,j)}p_{v,(i,j)} = \begin{cases} +1 & \text{if } (r_{u,i} - r_{u,j})(r_{v,i} - r_{v,j}) > 0 \\ -1 & \text{if } (r_{u,i} - r_{u,j})(r_{v,i} - r_{v,j}) < 0 \end{cases}$$

Thus the sum of $p_{u,(i,j)}p_{v,(i,j)}$ for all possible item pairs is $2(N_c - N_d)$, and $s_{u,v}$ can be rewritten as follows:

$$s_{u,v} = \frac{2(N_c - N_d)}{k(k-1)}$$

which is equivalent to the Kendall tau correlation coefficient based on the common preference sets (Eq. (4)). □

Proof of Theorem 2

Based on Corollary 1, the rating-oriented prediction formula based on $p_{m\times n(n-1)}$ (Eq. (2)) can be rewritten as follows:

$$p^P_{u,(i,j)} = \bar{p}_u + \frac{\sum_{v\in N_u} s_{u,v}(r_{v,(i,j)} - \bar{r}_v)}{\sum_{v\in N_u} s_{u,v}} = \frac{\sum_{v\in N_u} s_{u,v}p_{v,(i,j)}}{\sum_{v\in N_u} s_{u,v}} \quad (7)$$

which is the same formula of the rating-oriented prediction formula based on the common preference sets (Eq. (6)).

Besides, for memory-based algorithms, ranking-oriented CF performs same as rating-oriented CF, where a set of most similar users are discovered as the neighborhood users, based on which the scores/relationships of the unrated items/preferences are predicted for recommendation.

Thus, a ranking-oriented CF is equivalent to a rating-oriented CF based on user–(item–item) preference matrix. □

References

Adomavicius, G., Tuzhilin, A., 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17 (6), 734–749.
Chakravarthy, S.R., Helen, D., Karatza, 2013. Two-server parallel system with pure space sharing and Markovian arrivals. Comput. Oper. Res. 40 (1), 510–519.
Chen, Y., Tsai, W.T., 2014. Service-Oriented Computing and Web Software Integration, 4th ed. Kendall Hunt Publishing.
Chen, L.-C., Kuo, P.-J., Liao, I.-E., Huang, J.-Y., 2013. Scaling out recommender system for digital libraries with MapReduce. In: Proceedings of the Eighth International Conference on Grid and Pervasive Computing (GPC), pp. 40–47.
Cohen, W.W., Schapire, R.E., Singer, Y., 1999. Learning to order things. J. Artif. Intell. Res. 10, 243–270.
Dean, J., Ghemawat, S., 2004. MapReduce: simplified data processing on large clusters. In: Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI), pp. 137–149.
Deshpande, M., Karypis, G., 2004. Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. 22 (1), 143–177.
Goldberg, D., Nichols, D., Oki, B.M., Terry, D., 1992. Using collaborative filtering to weave an information tapestry. Commun. ACM 35 (12), 61–70.
Han, J., Kamber, M., Pei, J., 2011. Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J., 1999. An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 230–237.
Herlocker, J., Konstan, J.A., Riedl, J., 2002. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. ACM Trans. Inf. Syst. 5, 287–310.
Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T., 2004. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22 (1), 5–53.
Hofmann, T., 2004. Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst. 22, 89–115.
Järvelin, K., Kekäläinen, J., 2002. Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. 20 (4), 422–446.
Jiang, J., Lu, J., Zhang, G., Long, G., 2011. Scaling-up item-based collaborative filtering recommendation algorithm based on Hadoop. In: Proceedings of the 7th IEEE World Congress on Services (SERVICES), pp. 490–497.
Kendall, M.G., 1938. A new measure of rank correlation. Biometrika 30, 81.
Lee, C.-R., Chang, Y.-F., 2013. Enhancing accuracy and performance of collaborative filtering algorithm by stochastic SVD and its MapReduce implementation. In: Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPS), pp. 1869–1878.
Linden, G., Smith, B., York, J., 2003. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. 7 (1), 76–80.
Liu, N.N., Yang, Q., 2008. EigenRank: a ranking-oriented approach to collaborative filtering. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 83–90.
Liu, N.N., Zhao, M., Yang, Q., 2009. Probabilistic latent preference analysis for collaborative filtering. In: Proceedings of the 18th ACM conference on Information and Knowledge Management (CIKM), pp. 759–766.
McNee, S.M., Riedl, J., Konstan, J.A., 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems (CHI), pp. 1097–1101.
Rendle, S., Freudenthaler, C., Gantner, Z., Lars, S.-T., 2009. BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI), pp. 452–461.
Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J., 1994. GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW), pp. 175–186.
Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2000. Analysis of recommendation algorithms for e-commerce. In: Proceedings of the 2nd ACM Conference on Electronic Commerce (EC), pp. 158–167.
Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J., 2001. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web (WWW), pp. 285–295.
Schelter, S., Boden, C., Schenck, M., Alexandrov, A., Markl, V., 2013. Distributed matrix factorization with MapReduce using a series of broadcast-joins. In: Proceedings of the Seventh ACM Conference on Recommender Systems (RecSys), pp. 281–284.
Shani, G., Heckerman, D., Brafman, R.I., 2005. An MDP-based recommender system. J. Mach. Learn. Res. 6, 1265–1295.
Si, L., Jin, R., 2003. Flexible mixture model for collaborative filtering. In: Proceedings of the 20th International Conference on Machine Learning (ICML), pp. 704–711.
Su, X., Khoshgoftaar, T.M., 2009. A survey of collaborative filtering techniques. Adv. Artif. Intell. 2009, Article No. 4.
Sun, J., Wang, S., Gao, B.J., Ma, J., 2012. Learning to rank for hybrid recommendation. In: Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM), pp. 2239–2242.
Wang, S., Sun, J., Gao, B.J., Ma, J., 2012. Adapting vector space model to ranking-based collaborative filtering. In: Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM), pp. 1487–1491.
Weimer, M., Karatzoglou, A., Le, Q.V., Smola, A.J., 2007. CoFiRank: maximum margin matrix factorization for collaborative ranking. In: Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS), pp. 1593–1600.

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045

Ruzhi Xu is currently a Full Professor at Qilu University of Technology and Shandong University of Finance and Economics. He received Ph.D. in Computer Science from Fudan University, China, in 2005. He was a visiting professor at Arizona State University in 2011. His research interests include cloud computing, software engineering, and optimization.

Shuaiqiang Wang is currently an Associate Professor at Qilu University of Technology and Shandong University of Finance and Economics. He received B.Sc. and Ph.D. in Computer Science from Information Retrieval Lab, Shandong University, China, in 2004 and 2009 respectively. He was an exchange Ph.D. student at Hong Kong Baptist University in 2009. He was a postdoctoral research associate at Texas State University in 2010. His research interests include data mining, machine learning, and information retrieval.

Xuwei Zheng is received B.Sc. in Mathematics from Shandong University of Finance and Economics. Currently he is a graduate student at the same school. His research interests include data mining and cloud computing.

Yinong Chen is currently a Senior Lecturer at Arizona State University. He received his B.Sc. and M.Phil. in Computer Science from Chongqing University in China, and Ph.D. in Nature Science from University of Karlsruhe in Germany. He was a postdoctoral research fellow at University of Karlsruhe from 1993 to 1994. He was a Lecturer and Senior Lecturer respectively at University of Witwatersrand from 1994 to 2000. His research interests include service oriented architecture and cloud computing.

Please cite this article in press as: Xu, R., et al., Distributed collaborative filtering with singular ratings for large scale recommendation. J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2014.04.045