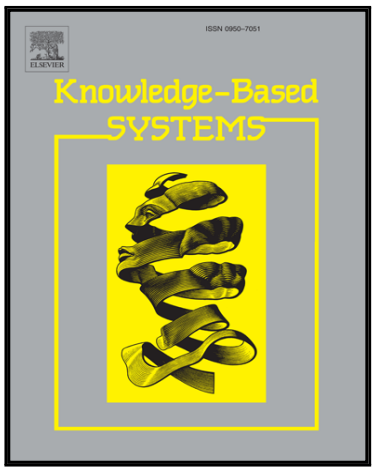


Fast Algorithms to Evaluate Collaborative Filtering Recommender Systems

Feng Zhang, Ti Gong, Victor E. Lee, Gansen Zhao, Chunming Rong, Guangzhi Qu

PII: S0950-7051(15)00507-9
DOI: 10.1016/j.knosys.2015.12.025
Reference: KNOSYS 3371

To appear in: *Knowledge-Based Systems*

Received date: 10 April 2015
Revised date: 28 December 2015
Accepted date: 29 December 2015

Please cite this article as: Feng Zhang, Ti Gong, Victor E. Lee, Gansen Zhao, Chunming Rong, Guangzhi Qu, Fast Algorithms to Evaluate Collaborative Filtering Recommender Systems, *Knowledge-Based Systems* (2016), doi: 10.1016/j.knosys.2015.12.025

Fast Algorithms to Evaluate Collaborative Filtering Recommender Systems

Feng Zhang[a,b,*], Ti Gong[a], Victor E. Lee[c], Gansen Zhao[d], Chunming Rong[e], Guangzhi Qu[f]

[a] *School of Computer Science, China University of Geosciences, Wuhan 430074, China*
[b] *Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China*
[c] *GraphSQL Inc., Mountain View, CA 94043 , USA*
[d] *School of Computer Science, South China Normal University, Guangzhou 510631, China*
[e] *Faculty of Science and Technology, University of Stavanger, Stavanger 4036, Norway*
[f] *Department of Engineering and Computer Science, Oakland University, Rochester, MI 48309, USA*

Abstract

Before deploying a recommender system, its performance must be measured and understood. So evaluation is an integral part of the process to design and implement recommender systems. In collaborative filtering, there are many metrics for evaluating recommender systems. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are among the most important and representative ones. To calculate MAE/RMSE, predicted ratings are compared with their corresponding true ratings. To predict item ratings, similarities between active users and their candidate neighbors need to be calculated. The complexity for the traditional and naive similarity calculation corresponding to user u and user v is quadratic in the number of items rated by u and v. In this paper, we explore the mathematical regularities underlying similarity formulas, introduce a novel data structure, and design linear time algorithms to calculate the similarities. Such complexity improvement shortens the evaluation time and will finally contribute to increasing the efficiency of design and development of recommender systems. Experimental results confirm the claim.

Keywords: Recommender Systems, Collaborative Filtering, MAE/RMSE,

*Corresponding author
Email address: jeff.f.zhang@gmail.com (Feng Zhang)

1. Introduction

Recommender systems are popular facilities widely deployed to address the challenge of overwhelming information. They are used to seek interesting information to targeted users from a large volume of data. Typical domains where we can see their real-world applications include E-commerce [1, 2], E-government [3, 4], Social Network [5, 6, 7], Academia [8, 9, 10, 11], Entertainment [12, 13], Telecom [14, 15], and so on. Readers can see the comprehensive progress in this topic in a recently published survey paper [16].

There are three basic methods to generate recommendations: collaborative filtering (CF), content-based filtering, and a hybrid approach. CF recommendation aims to produce a list of interesting items to active users based on the preferences of their like-minded neighborhood. Content-based filtering approaches utilize a series of discrete features of items, e.g., the genres, directors, and actors of movies, to generate recommendations. These two approaches are often combined to make hybrid recommender systems [17, 18].

A CF recommender system generally works in three steps. First, it calculates the similarities (cosine similarity, adjusted cosine similarity, Pearson correlation similarity, et al.) between the active user and other users. Second, it selects the active user's nearest neighbors based on the similarities obtained from the first step. Third, it recommends a list of top-k items by aggregating the nearest neighbors' preferences.

CF is generally believed to be one of the most successful techniques applied in recommender systems. A flowchart to illustrate the process of designing a recommender system is shown in Figure 1. It should be emphasized that *evaluation* is an important and inalienable part of designing a good CF algorithm. There are many metrics to evaluate the performance of recommender systems. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are among the most popular ones.

Given the N actual/predicted rating pairs $(r_{u,i}, p_{u,i})$, where u refers to a user and i to an item, the MAE of the N pairs is evaluated as:

$$MAE = \frac{|\sum_{i=1}^{N}(p_{u,i} - r_{u,i})|}{N} \quad (1)$$

and the RMSE of the N pairs is evaluated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(p_{u,i} - r_{u,i})^2}{N}} \quad (2)$$

Note that lower MAE and RMSE values indicate more accurate predictions, signifying better performance of recommender systems.

To evaluate MAE/RMSE, four steps need to be conducted:

(1) Divide the dataset into a test set (acting as active users, similarly hereinafter) and a training set.

(2) Compute and select top-k nearest neighbors in the training set based on a similarity metric, for a rated item of the current test user in the test set.

(3) Aggregate the ratings of the top-k nearest neighbors to calculate a prediction for the item.

(4) Repeat Step (2) and Step (3) until the prediction for every rated item of every user in the test set is calculated, and then compute MAE and/or RMSE for the test users.

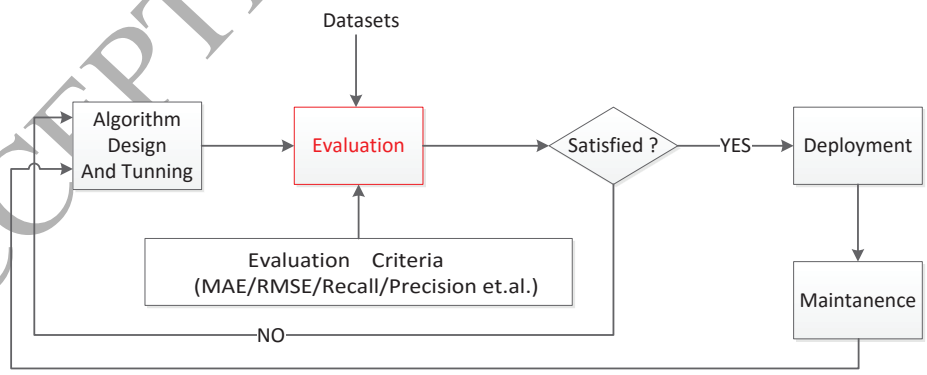


Figure 1: Flowchart of the Design of a Recommender System

Essentially, the major effort of our work is to shorten the time spent in the similarity calculation of Step (2).

1.1. Motivation

A flowchart to illustrate the design process of recommender systems is illustrated in Figure 1. As we have pointed out, *evaluation* is an inalienable part of the design process. As the scale of datasets is large, *evaluation* generally costs much time, which is mechanically spent on the running of recommender systems and is with a low technology content. Shortening the *evaluation* time can save the designers from waiting for the evaluation done, let them determine the performance more quickly, and focus on other work with a high technology content.

CF research papers have published few technical details about their evaluation processes. We have reviewed the major open source recommender systems which use neighborhood-based CF, such as Mahout[1], LensKit[2], Crab[3], Python-recsys[4], and easyrec[5], and we find that their MAE/RMSE evaluation procedures are either flawed or they ignore efficiency.

In particular, given a user u in the test set and a user v in the training set, two major problems in evaluating MAE/RMSE (computing the similarities specifically) are described as follows:

(1) *Incorrect method*: The similarity between u and v is calculated only once, which is then used to help generate the $|I_u|$ predictions[6]. This means in the above Step (2), in case of each rated item, the similarities between u and v are considered equal and are calculated only once, but they are not equal indeed! The method is illustrated in Figure 2a.

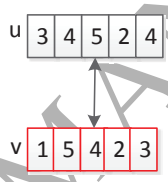
A correct method to evaluate MAE/RMSE should include the calculation

[1] http://mahout.apache.org/
[2] http://lenskit.org/
[3] http://muricoca.github.io/crab/
[4] http://ocelma.net/software/python-recsys/build/html/
[5] http://easyrec.org/home
[6] $|I_u|$ denotes the size of the set of items rated by user u. Please refer to Table 1

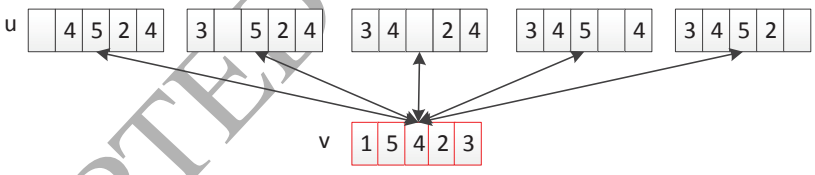
of predictions for every item rated by user u. For each prediction, there should be a unique similarity between u and v. In the end, the difference between the predicated rating and the true rating can be used to evaluate the MAE/RMSE.

The reason why the similarity should be calculated individually is that for every predicted item rating, the corresponding true rating in u should be treated as unknown.

(2) *Correct but inefficient method*: The similarities between u and v are considered separately, varying for each rated item in u. For each item rating prediction, the similarity is calculated once, so there are $|I_u|$ similarity calculations. Each time the calculation is with complexity of $O(|I_u| + |I_v|)$, so the whole complexity for the $|I_u|$ calculations is $O(|I_u| \times (|I_u| + |I_v|))$. The method is illustrated in Figure 2b.



(a) Incorrect Method: Calculate Once



(b) Inefficient Method: Calculate $|I_u| \equiv 5$ Times

Figure 2: Methods to Calculate Similarity Between u and v

The objective of this work is to design efficient algorithms for computing similarity in the process of evaluating recommender systems, whose complexity is of $O(|I_u|+|I_v|)$. In other words, the $|I_u|$ times of similarity calculations should be done in one iteration of I_u and I_v respectively.

1.2. Contributions

Most work to date has focused on response time, accuracy and scalability of recommender systems. There has been little attention paid to speeding up the evaluation performances of recommender systems. However, evaluation is a crucial step for designing and tuning recommender systems.

The contributions of this work are as follows.

Firstly, the need for more efficient algorithms to evaluate recommender systems is identified, which could facilitate their design and implementation.

Secondly, the underlying mathematical regularities behind the similarity formula are investigated to help design more efficient algorithms to evaluate MAE/RMSE.

This work can shorten the time in designing and implementing recommender system algorithms.

1.3. Structure

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces the basic knowledge on CF recommendation. Section 4 explores the mathematical regularities underlying similarity formulas and explains why the proposed methods are more efficient. Then the algorithms are articulated in Section 5. The experimental methods and results are shown in Section 6. Finally, Section 7 concludes the paper and future work is suggested.

2. Related Work

Recommender systems are becoming more and more popular with the onset of the World Wide Web and big data. Designing recommender systems requires overcoming challenges of accuracy, scalability, cold start, and privacy [19, 20, 21, 22].

Herlocker et al. [23] comprehensively review the key decisions in evaluating CF recommender systems. They demonstrate the importance of recommender system evaluation and they claim that accuracy is a major metric.

An accuracy metric measures how well a recommender system predicts ratings for specific items. Many accuracy metrics have been used for evaluation purpose in the design and application of recommender systems. The recommendation accuracy metrics can be classified into three categories: predictive accuracy metrics, classification accuracy metrics, and rank accuracy metrics.

Predictive accuracy metrics measure the quantity of similarity between predicted ratings and true ratings. Mean Absolute Error (MAE) and the related metrics, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Normalized Mean Absolute Error (NMAE) have been used to evaluate recommender systems in many cases [23, 24, 25].

Precision, recall, F1 metric, ROC (Receiver Operating Characteristic), nDCG (Normalized Discounted Cumulative Gain), and other related metrics belong to the classification accuracy category. These metrics come from information retrieval research, and they measure the fraction of prediction, recommendations, and searching results that are good or not [24, 26].

Rank accuracy metrics are unlike predictive accuracy metrics and classification accuracy metrics which directly measure the quality of the predicted items. Instead, they focus on the ordering quality of the recommended items. The Pearson product-moment correlation coefficient, half-life utility metric, and the NDPM (Normalized Distance-based Performance Measure) metric are some commonly used rank accuracy metrics.

To judge whether a recommender system is good or not, the most meaningful criteria may be whether the users can benefit from it and if the users are willing to use the services again due to the recommendations. The criteria are called *user acceptance of recommender systems*. For example, Armentano et al. [27] design a so called Technology Acceptance Model (TAM) to specifically deal with the acceptance of recommender systems. TAM includes three measures: perceived usefulness, perceived ease of use, and attribute towards using [28]. Armentano et al. empirically show the usage of TAM by the experiments of comparing a basic kNN algorithm and a context aware kNN algorithm.

In summary, recommender system evaluation is an important issue and

selecting proper evaluation metrics is critical. Indeed, many metrics have been proposed for evaluating the performance of recommendation systems. Gunawardana and Shani emphasize on the importance of evaluating recommender systems, focusing on how to decide proper metrics applied to evaluation since different metrics may favor different algorithms [25].

The point that *different metrics may favor different algorithms* is very important. Another point is that different metrics may favor different data. To apply TAM, extra data have to be available. Armentano et al. [27] obtain such data through collecting answers from users for the eleven questions to determine the *perceived usefulness*, *perceived ease of use* and *attribute towards using* of a recommender system. In fact, in the designing and implementing process of recommender systems, collecting such kind of data is not easy and even is impractical. Item rating data, whether in their implicit or explicit forms, are more popular. When real-valued ratings are available, prediction accuracy metrics, such as MAE and RMSE, may be among the most useful measures.

In practical studies, there are few efforts spent on designing efficient algorithms to evaluate recommendation algorithms. We argue that this is an important step to speed up algorithm development and implementation. We can see that open source projects have used inadequate methods to calculate MAE/RMSE. In this paper, we examine carefully the underlying features in the similarity equations and design fast evaluation algorithms.

3. Basics on Collaborative Filtering Recommendation

Recommender system techniques can be classified into three categories: content-based filtering, collaborative filtering, and hybrid filtering. Content-based filtering produces recommendations for active users according to the similarities between items. Collaborative filtering, however, provides recommendations based on the preferences of other like-minded users. Hybrid filtering combines these two methods to produce recommendations [29]. All such recommender systems are based on user-item rating matrices.

Table 1: Symbol Denotation

Symbols	**Meaning**
I_{uv}	the set of items rated by both user u and user v
I_u	the set of items rated by user u
I_v	the set of items rated by user v
$r_{u,i}$	user u's rating score for item i
$r_{v,i}$	user v's rating score for item i
$\bar{r}_u$	the average of ratings by user u
$\bar{r}_v$	the average of ratings by user v
$\bar{r}_i$	the average ratings for item i
$p_{u,i}$	user u's predicted rating for item i

The underlying rationale behind CF is the assumption that users like those items that their similar users like. Therefore, to generate recommendations for an active user, the first step is to find his/her similar users, which are collectively called the nearest neighbors. Three popular and widely-used user similarity metrics for CF are cosine-based, adjusted cosine-based, and Pearson correlation-based [30]. Using the notation for items, users, and ratings described Table 1, we present formulas for each of these three.

Cosine-based similarity is formalized as Equation (3):

$$sim(u,v)_{cos} = \frac{\sum_{i \in I_{uv}} (r_{u,i} r_{v,i})}{\sqrt{\sum_{i \in I_u} (r_{u,i})^2} \sqrt{\sum_{i \in I_v} (r_{v,i})^2}} \tag{3}$$

Adjusted cosine-based similarity is formalized as Equation (4):

$$sim(u,v)_{acos} = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i)}{\sqrt{\sum_{i \in I_u} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{i \in I_v} (r_{v,i} - \bar{r}_i)^2}} \tag{4}$$

Pearson correlation-based similarity can be formalized as Equation (5):

$$sim(u,v)_{pearson} = \frac{\sum_{i \in I_{uv}}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I_{uv}}(r_{v,i} - \bar{r}_v)^2}} \quad (5)$$

After determining the nearest neighbors of user u, denoted as NBS_u, a typical method to predict user u's rating score for its unrated item i can be formalized as Equation (6) [31].

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in NBS_u} sim(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in NBS_u} sim(u,v)} \quad (6)$$

4. Methodologies

This section first gives a formal definition for the recommender system problem presented above. It then presents theorems which prove that MAE/RMSE can be evaluated with time complexity $O(|I_u| \times |I_v|)$, for each of the three user similar metrics presented in the previous section.

Definition 1 (Recommendation Problem). There are a set of users $U = \{u_1, u_2, ..., u_n\}$ and a set of items $I = \{i_1, i_2, ..., i_m\}$. Users' preferences for items are represented using a $n \times m$ ratings matrix $M = (R_1, R_2, ..., R_n)^T$. Each user u rates a set of items $I_u = \{i_{j_1}, i_{j_2}, ..., i_{j_k}\}$, and $I_u \subseteq I$. We say each $R_u = (r_{u,j_1}, r_{u,j_2}, ..., r_{u,j_k})$ is a rating vector corresponding to user u. The rating value of user u on item i is denoted by $r_{u,i}$. $r_{u,i}$ can be any real number, but more often it is an integer, e.g., in the range $[1, 5]$. Typically, users rate only a very small percentage of items, i.e., $r_{u,i}$ is unknown for most pairs (u, i), which is cause of the data sparsity problem. The task of recommendation is as follows: given a user $u \in U$ (u is called the active user) and an item $i \in I$ (i is called the target item) for which $r_{u,i}$ is unknown, predict a rating for u on item i. The predicted rating is denoted by $\hat{r}_{u,i}$. The items with the highest predicted ratings are recommended to the active user u.

To evaluate MAE/RMSE, for each non-null $r_{u,i}$, we calculate $\hat{r}_{u,i}$ and compare them. Specifically, considering user u, for each rating item in I_u, one corresponding set of nearest neighbors needs to be determined. So there are $|I_u|$ sets of nearest neighbors for user u. For the purpose of computing MAE/RMSE contributed by user u, if user v is a nearest neighbor of user u, there are at most $|I_u|$ similarities between u and v to be considered: the case without r_{u,j_1}, the case without r_{u,j_2}, ..., and so on. Therefore, to compute all the similarities between u and v, a naive approach requires $|I_u| \times (|I_u| + |I_v|)$ similarity computations. Figure 2b illustrates an end to end example for this correct but inefficient calculation.

The major contribution of our work is shown in the following theorems, which indicate that the similarity between u and v can be calculated in $O(|I_u| + |I_v|)$ time complexity, in stead of $|I_u| \times (|I_u| + |I_v|)$.

Theorem 1 (To determine the MAE/RMSE components contributed by test user u corresponding to training user v based on Pearson correlation similarities, the time complexity is $O(|I_u| + |I_v|)$).

Proof Sketch:

To obtain the MAE/RMSE components contributed by u, the naive but correct Pearson correlation similarity calculation runs Equation (7) $|I_u|$ times, costing complexity of $O(|I_u| \times (|I_u| + |I_v|)$.

$$sim(u,v)^{!k}_{pearson} = \frac{\sum_{i \in I_{uv}, i \neq k}(r_{u,i} - \bar{r}_{u!k})(r_{v,i} - \bar{r}_{v!k})}{\sqrt{\sum_{i \in I_{uv}, i \neq k}(r_{u,i} - \bar{r}_{u!k})^2}\sqrt{\sum_{i \in I_{uv}, i \neq k}(r_{v,i} - \bar{r}_{v!k})^2}} \tag{7}$$

where $sim(u,v)^{!k}_{pearson}$ denotes Pearson correlation similarity between user u and user v for an item k. Other denotations of symbols in Equation (7), similarly, in Equations (8), (9), (10) as well, have been shown in Table 1 except $\bar{r}_{u!k}$ and $\bar{r}_{v!k}$. $\bar{r}_{u!k}$ denotes the average of ratings (excluding the rating for item k) by user u, and $\bar{r}_{v!k}$ denotes the average of ratings (excluding the rating for item k) by user v. Note that the difference between $sim(u,v)^{!k}_{pearson}$ and $sim(u,v)_{pearson}$ in Equation (5) is that the former is the similarity between user u and user v by treating user u's rating for item k as unknown, but the latter includes u's

rating for item k as known.

To obtain complexity of $O(|I_u| + |I_v|)$, we decompose Equation (7) and find the regularities that can lead to more efficient calculations. The deductions in Equations (8), (9) and (10) show the efforts. In the deductions, to facilitate the understanding of the proof, suppose $\mathscr{N}$, $\mathscr{D}_1$ and $\mathscr{D}_2$ denote the numerator $\sum_{i \in I_{uv}, i \neq k}(r_{u,i} - \bar{r}_{u!k})(r_{v,i} - \bar{r}_{v!k})$, denominator part $\sum_{i \in I_{uv}, i \neq k}(r_{u,i} - \bar{r}_{u!k})^2$ and denominator part $\sum_{i \in I_{uv}, i \neq k}(r_{v,i} - \bar{r}_{v!k})^2$ in Equation (7) respectively, and set $S1 = \sum_{i \in I_{uv}} r_{u,i} r_{v,i}$, $S2 = \sum_{i \in I_v} r_{v,i}$, $S3 = \sum_{i \in I_{uv}} r_{u,i}$, $S4 = \sum_{i \in I_u} r_{u,i}$, $S5 = \sum_{i \in I_{uv}} r_{v,i}$, $S6 = \sum_{i \in I_{uv}} (r_{u,i})^2$, $S7 = \sum_{i \in I_{uv}} (r_{v,i})^2$.

Then, the deductions show that the calculations of $\mathscr{N}$, $\mathscr{D}_1$ and $\mathscr{D}_2$, can be decomposed into $r_{u,k}$, $r_{v,k}$, $|I_u|$, $|I_v|$, $|I_{uv}|$, and a set of summations which are INDEPENDENT of k. They are just simple addition, subtraction, multiplication, and division, of these summations and components, and the time cost is $O(1)$. There are at most $|I_u|$ items, so the time spent in this part is $O(|I_u|)$. Note that all of the summations, $S1$, $S2$, $S3$, $S4$, $S5$, $S6$, $S7$, and other components, $|I_u|$, $|I_v|$, $|I_{uv}|$, can be precomputed with time complexity at most $O(|I_u| + |I_v|)$. In summary, the time dominating the similarity computation rests in the precomputing of the summations and other components, so the whole time complexity can be represented by $O(|I_u| + |I_v|)$.

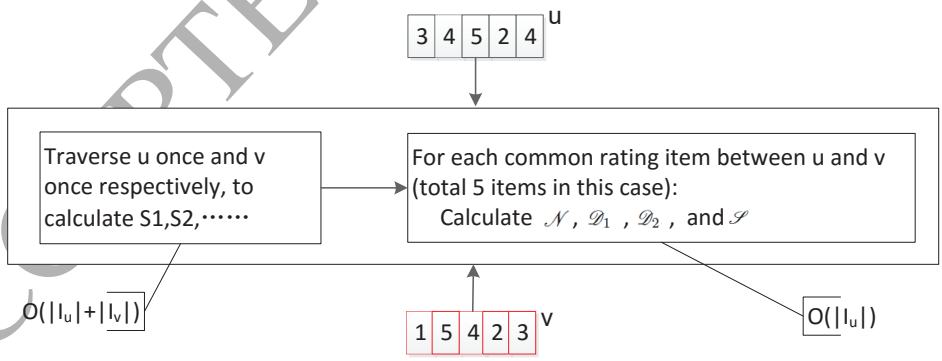


Figure 3: A Faster Method to Calculate Similarity Between u and v

In summary, there are two major steps to conduct the similarity calculation.

The first step costs complexity of $O(|I_u| + |I_v|)$, and the second costs $O(|I_u|)$. The process is shown in Figure 3 with the sample inputs of user u and user v.

$$
\begin{aligned}
\mathscr{N} &= \sum_{i \in I_{uv}, i \neq k} (r_{u,i} - \bar{r}_{u!k})(r_{v,i} - \bar{r}_{v!k}) \\
&= \sum_{i \in I_{uv}, i \neq k} r_{u,i} r_{v,i} - \bar{r}_{v!k} \sum_{i \in I_{uv}, i \neq k} r_{u,i} \\
&\quad - \bar{r}_{u!k} \sum_{i \in I_{uv}, i \neq k} r_{v,i} + (|I_{uv}| - 1) \bar{r}_{u!k} \bar{r}_{v!k} \\
&= \sum_{i \in I_{uv}, i \neq k} r_{u,i} r_{v,i} \\
&\quad - \frac{\sum_{i \in I_v, i \neq k} r_{v,i}}{|I_v| - 1} \sum_{i \in I_{uv}, i \neq k} r_{u,i} \\
&\quad - \frac{\sum_{i \in I_u, i \neq k} r_{u,i}}{|I_u| - 1} \sum_{i \in I_{uv}, i \neq k} r_{v,i} \\
&\quad + (|I_{uv}| - 1) \frac{\sum_{i \in I_u, i \neq k} r_{u,i}}{|I_u| - 1} \frac{\sum_{i \in I_v, i \neq k} r_{v,i}}{|I_v| - 1} \\
&= \sum_{i \in I_{uv}} r_{u,i} r_{v,i} - r_{u,k} r_{v,k} \\
&\quad - \frac{\sum_{i \in I_v} r_{v,i} - r_{v,k}}{|I_v| - 1} (\sum_{i \in I_{uv}} r_{u,i} - r_{u,k}) \\
&\quad - \frac{\sum_{i \in I_u} r_{u,i} - r_{u,k}}{|I_u| - 1} (\sum_{i \in I_{uv}} r_{v,i} - r_{v,k}) \\
&\quad + (|I_{uv}| - 1) \frac{\sum_{i \in I_u} r_{u,i} - r_{u,k}}{|I_u| - 1} \frac{\sum_{i \in I_v} r_{v,i} - r_{v,k}}{|I_v| - 1} \\
&= S1 - r_{u,k} r_{v,k} \\
&\quad - \frac{S2 - r_{v,k}}{|I_v| - 1} (S3 - r_{u,k}) \\
&\quad - \frac{S4 - r_{u,k}}{|I_u| - 1} (S5 - r_{v,k}) \\
&\quad + (|I_{uv}| - 1) \frac{S4 - r_{u,k}}{|I_u| - 1} \frac{S2 - r_{v,k}}{|I_v| - 1}
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
\mathscr{D}_1 &= \sum_{i\in I_{uv}, i\neq k} (r_{u,i} - \bar{r}_{u!k})^2 \\
&= \sum_{i\in I_{uv}, i\neq k} ((r_{u,i})^2 - 2r_{u,i}\bar{r}_{u!k} + (\bar{r}_{u!k})^2) \\
&= \sum_{i\in I_{uv}} (r_{u,i})^2 - (r_{u,k})^2 \\
&\quad - 2(\sum_{i\in I_{uv}} r_{u,i} - r_{u,k}) \frac{\sum_{i\in I_u} r_{u,i} - r_{u,k}}{|I_u| - 1} \\
&\quad + (|I_{uv}| - 1)(\frac{\sum_{i\in I_u} r_{u,i} - r_{u,k}}{|I_u| - 1})^2 \\
&= S6 - (r_{u,k})^2 \\
&\quad - 2(S3 - r_{u,k}) \frac{S4 - r_{u,k}}{|I_u| - 1} \\
&\quad + (|I_{uv}| - 1)(\frac{S4 - r_{u,k}}{|I_u| - 1})^2
\end{aligned}
\tag{9}
$$

250

$$
\begin{aligned}
\mathscr{D}_2 &= \sum_{i\in I_{uv}, i\neq k} (r_{v,i} - \bar{r}_{v!k})^2 \\
&= \sum_{i\in I_{uv}, i\neq k} ((r_{v,i})^2 - 2r_{v,i}\bar{r}_{v!k} + (\bar{r}_{v!k})^2) \\
&= \sum_{i\in I_{uv}} (r_{v,i})^2 - (r_{v,k})^2 \\
&\quad - 2(\sum_{i\in I_{uv}} r_{v,i} - r_{v,k}) \frac{\sum_{i\in I_v} r_{v,i} - r_{v,k}}{|I_v| - 1} \\
&\quad + (|I_{uv}| - 1)(\frac{\sum_{i\in I_v} r_{v,i} - r_{v,k}}{|I_v| - 1})^2 \\
&= S7 - (r_{v,k})^2 \\
&\quad - 2(S5 - r_{v,k}) \frac{S2 - r_{v,k}}{|I_v| - 1} \\
&\quad + (|I_{uv}| - 1)(\frac{S2 - r_{v,k}}{|I_v| - 1})^2
\end{aligned}
\tag{10}
$$

Similarly, we have Theorem 2.

Theorem 2 (To determine the MAE/RMSE components contributed by test user u corresponding to training user v based on adjusted cosine similarities, the time complexity is $O(|I_u| + |I_v|)$).

Proof Sketch:

Likewise, if the naive approach is utilized, Equation (11) needs to be calculated $|I_u|$ time and the whole complexity is $O(|I_u| \times (|I_u| + |I_v|))$.

$$sim(u,v)^{!k}_{acos} = \frac{\sum_{i \in I_{uv}, i \neq k}(r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i)}{\sqrt{\sum_{i \in I_u, i \neq k}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{i \in I_{uv}, i \neq k}(r_v - \bar{r}_i)^2}} \tag{11}$$

where all the symbol denotations have been shown in Table 1 except $sim(u,v)^{!k}_{acos}$, which denotes the adjusted cosine similarity between user u and user v by treating user u's rating for item k as unknown.

Note we have set $S1$, $S2$, $S3$, $S4$, and $S5$ in Theorem 1, furthermore, we set $S8 = \sum_{i \in I_{uv}} \bar{r}_i$, $S9 = \sum_{i \in I_{uv}} (\bar{r}_i)^2$, $S10 = \sum_{i \in I_u} (r_{u,i})^2$, $S11 = \sum_{i \in I_u} (\bar{r}_i)^2$, $S12 = \sum_{i \in I_v} (r_{v,i})^2$, $S13 = \sum_{i \in I_v} (r_{v,i})^2$. Following the proof process of Theorem 1, the deductions below shown in Equations (12), (13) and (14), also prove that Theorem 2 is true.

$$
\begin{aligned}
\mathscr{N} &= \sum_{i \in I_{uv}, i \neq k} (r_{u,i} - \bar{r}_i)(r_{v,i} - \bar{r}_i) \\
&= \sum_{i \in I_{uv}, i \neq k} r_{u,i} r_{v,i} - \sum_{i \in I_{uv}, i \neq k} \bar{r}_i (\sum_{i \in I_{uv}, i \neq k} r_{u,i} + \sum_{i \in I_{uv}, i \neq k} r_{v,i}) + \sum_{i \in I_{uv}, i \neq k} (\bar{r}_i)^2 \\
&= \sum_{i \in I_{uv}} r_{u,i} r_{v,i} - r_{u,k} r_{v,k} \\
&\quad - (\sum_{i \in I_{uv}} \bar{r}_i - \bar{r}_k)(\sum_{i \in I_{uv}} r_{u,i} - r_{u,k} + \sum_{i \in I_{uv}} r_{v,i} - r_{v,k}) \\
&\quad + \sum_{i \in I_{uv}} (\bar{r}_i)^2 - (\bar{r}_k)^2 \\
&= S1 - r_{u,k} r_{v,k} \\
&\quad - (S8 - \bar{r}_k)(S3 - r_{u,k} + S5 - r_{v,k}) \\
&\quad + S9 - (\bar{r}_k)^2
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
\mathscr{D}_1 &= \sum_{i \in I_u, i \neq k} (r_{u,i} - \bar{r}_i)^2 \\
&= \sum_{i \in I_u, i \neq k} ((r_{u,i})^2 - 2 r_{u,i} \bar{r}_i + (\bar{r}_i)^2) \\
&= \sum_{i \in I_u} (r_{u,i})^2 - (r_{u,k})^2 \\
&\quad - 2(\sum_{i \in I_u} r_{u,i} - r_{u,k}) \bar{r}_i \\
&\quad + \sum_{i \in I_u} (\bar{r}_i)^2 - (\bar{r}_k)^2 \\
&= S10 - (r_{u,k})^2 \\
&\quad - 2(S4 - r_{u,k}) \bar{r}_i \\
&\quad + S11 - (\bar{r}_k)^2
\end{aligned}
\tag{13}
$$

$$
\begin{aligned}
\mathscr{D}_2 &= \sum_{i \in I_v, i \neq k} (r_{v,i} - \bar{r}_i)^2 \\
&= \sum_{i \in I_v, i \neq k} ((r_{v,i})^2 - 2 r_{v,i} \bar{r}_i + (\bar{r}_i)^2) \\
&= \sum_{i \in I_v} (r_{v,i})^2 - (r_{v,k})^2 \\
&\quad - 2(\sum_{i \in I_v} r_{v,i} - r_{v,k}) \bar{r}_i \\
&\quad + \sum_{i \in I_v} (\bar{r}_i)^2 - (\bar{r}_k)^2 \\
&= S12 (r_{v,i})^2 - (r_{v,k})^2 \\
&\quad - 2(S2 - r_{v,k}) \bar{r}_i \\
&\quad + S13 - (\bar{r}_k)^2
\end{aligned}
\tag{14}
$$

Theorem 3 (To determine the MAE/RMSE components contributed by test user u corresponding to training user v based on cosine similarities, the time complexity is $O(|I_u| + |I_v|)$).

Proof Sketch:

The traditional and naive cosine similarity is shown in Equation (15). And the whole proof procedure is similar with those of Theorem 1 and Theorem 2.

$$
sim(u,v)^{!k}_{cos} = \frac{\sum_{i \in I_{uv}, i \neq k} (r_{u,i} r_{v,i})}{\sqrt{\sum_{i \in I_u, i \neq k} (r_{u,i})^2} \sqrt{\sum_{i \in I_v, i \neq k} (r_{v,i})^2}} \tag{15}
$$

5. Algorithms

To calculate MAE/RMSE, it is necessary to maintain a data structure NNVector (Figure 4), which stores the lists of the nearest neighbors, one for each rated item in the test set of users.

The NNVector in its essence is an array where each element is a list of the nearest neighbors. It is obvious that the size of the NNVector corresponding to user u is determined by the number of his/her rated items, $|I_u|$.

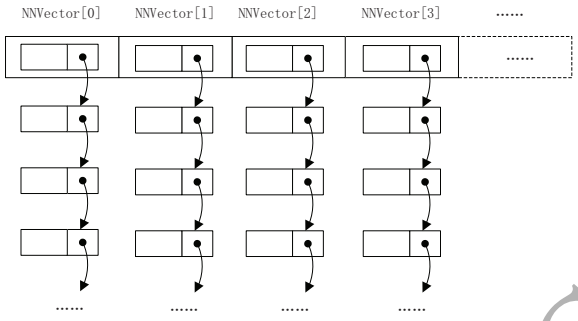


Figure 4: User's Neighborhood List NNVector

Algorithm 1: Naive algorithm to judge whether v is in u's neighborhood corresponding to each rated item in u using the Pearson correlation similarities

input : sets of rated items of users u and v, I_u and I_v; rating vectors of users u and v, r_u and r_v; maximum number of nearest neighbors, N.

output: u's neighborhood list NNVector

```
1 Calculate common rated items between u and v, and store them
  into set citems;
2 foreach ci ∈ citems do
3     Calculate the similarity 𝒮 based on Equation (7);
4     if NNVector[ci].size < N or 𝒮 > the least distance in the NN
      list NNVector[ci] then
5         Insert v to list NNVector[ci] as the neighborhood of u
          corresponding to item ci;
6     end
7 end
```

Algorithm 2: Fast algorithm to judge if v is in u's neighborhood corresponding to each rated item in u using the Pearson correlation similarities

input : sets of rated items of users u and v, I_u and I_v; rating vectors of users u and v, r_u and r_v; maximum number of nearest neighbors, N.

output: u's neighborhood list NNVector.

Calculate $S1 = \sum_{i \in I_{uv}} r_{u,i} r_{v,i}$, $S2 = \sum_{i \in I_v} r_{v,i}$, $S3 = \sum_{i \in I_{uv}} r_{u,i}$, $S4 = \sum_{i \in I_u} r_{u,i}$, $S5 = \sum_{i \in I_{uv}} r_{v,i}$, $S6 = \sum_{i \in I_{uv}} (r_{u,i})^2$, $S7 = \sum_{i \in I_{uv}} (r_{v,i})^2$;

Calculate common rated items between u and v, and store them into set *citems*;

foreach $ci \in citems$ **do**

  Calculate $\mathscr{N}$ based on Equation (8);

  Calculate $\mathscr{D}_1$ based on Equation (9);

  Calculate $\mathscr{D}_2$ based on Equation (10);

  Calculate $\mathscr{S} = \frac{\mathscr{N}}{\mathscr{D}_1 \times \mathscr{D}_2}$;

  if *NNVector[ci].size* $< N$ *or* $\mathscr{S} >$ *the least distance in the NN list NNVector[ci]* **then**

    Insert v to list NNVector[ci] as the neighborhood of u corresponding to item *ci*;

  end

end

A traditional and naive method to judge if user v is user u's neighbor is shown in Algorithm 1, in which Step 3 has to be calculated $|I_u| + |I_v|$ times, such that the complexity concerning the neighborhood calculation is $O(|I_u| \times (|I_u| + |I_v|))$.

However, in the well-designed Algorithm 2, the complexity of both Step 2 and Step 3 is $O(|I_u| + |I_v|)$, which is the dominating time cost of the algorithm.

6. Experiments

The contributions of this study pertain to the reduction of computation time for the evaluation of the MAE and RMSE metrics. From the findings of mathematical regularities of the formulas, their complexity is reduced from $O(|I_u| \times (|I_u| + |I_v|))$ to $O(|I_u| + |I_v|)$. Experiments corroborate this. Note that this study doesn't cover the precision, scalability and cold start problems in recommender systems. Therefore there is no difference in the computational results, except for the speed they are computed.

We conducted experiments on three benchmark datasets: two MovieLens datasets, the MovieLens 100K and the MovieLens 1M[7], and one Jester dataset[8]. MovieLens 100K is a movie rating dataset consisting of 100,000 ratings (1-5) from 943 users on 1,682 movies, and each user has rated at least 20 movies. MovieLens 1M contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users. We choose the version of Jester with 1,810,455 ratings issued by 24,983 different users for 100 different jokes, and each user has rated 36 or more jokes.

The algorithms were implemented in C/C++. They were run on an Ubuntu 12.04 virtual machine with a host of Windows 8 64-bit operating system running on a laptop (Intel Core i7-2640M CPU 2.80GHz with 8GB RAM).

Ten-fold cross evaluation method was used to implement the experiments. Pearson correlation and adjusted cosine were chosen as similarity metrics to conduct experiments. The number of nearest neighbors was fixed at 20.

[7] http://www.grouplens.org/node/73

[8] http://eigentaste.berkeley.edu/dataset/

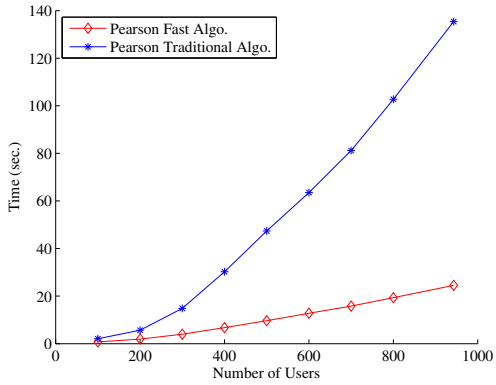


Figure 5: Time vs. Number of Users (MovieLens 100K)

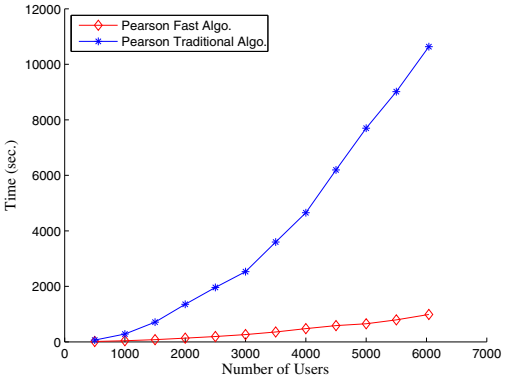


Figure 6: Time vs. Number of Users (MovieLens 1M)

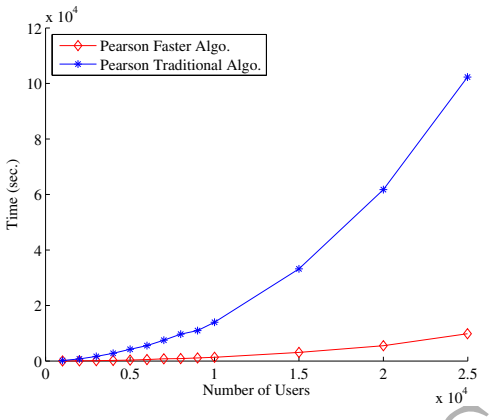


Figure 7: Time vs. Number of Users (Jester)

Figure 5 reports the experimental results on MovieLens 100K, Figure 6 on MovieLens 1M and Figure 7 on Jester. The three figures show that the elapsed time spent in computing MAE/RMSE grows as the number of users increased. According to these figures, not only does our new method run faster than the traditional method does, but also the gap between them grows as the number of users grows.

Figure 5 shows that for the MovieLens 100M dataset, when the number of users grows from 100 to 943, the maximum number of users in the dataset, the ratio of elapsed time spent on the traditional algorithm to that spent on the fast algorithm increases from around 3:1 to 6:1. For the MovieLens 1M dataset, as the number of users grows from 500 to 6040, the ratio climbs from around 5:1 to 10:1. And for the Jester dataset, similar trends can be seen but the acceleration performance is more significant. It suggests that greater volumes of data lead to more significant improvement in time complexity, which conforms to the complexity analysis in Section 5.

Note that both Figure 5, Figure 6 and Figure 7 only illustrate results based on Pearson correlation similarities. Tables 2, 3 and 4 show in detail, for the metrics of Pearson correlation similarities and adjusted cosine correlation similarities, how the elapsed time changes with respect to the number of users. The t-

Table 2: Time vs. # of Users – MovieLens 100K

# of Users	Elapsed Time			
	t-pearson	f-pearson	t-acosine	f-acosine
100	2.06	0.74	2.55	0.77
200	5.65	1.95	6.90	2.02
300	14.83	3.96	17.30	4.21
400	30.23	6.73	35.76	7.12
500	47.35	9.67	55.69	10.29
600	63.52	12.81	74.67	13.35
700	81.16	15.79	95.86	16.65
800	102.71	19.32	118.17	20.23
943	135.37	24.52	165.87	27.44

pearson column lists the results generated by traditional Pearson correlation similarities; The f-pearson column lists the results generated by fast Pearson correlation similarities; The t-acosine column lists the results generated by traditional adjusted cosine correlation similarities, and the f-acosine column lists the results generated by fast adjusted cosine correlation similarities.

Table 2 is for the MovieLens 100K dataset, Table 3 is for the MovieLens 1M dataset, and Table 4 is for the Jester dataset. These three tables show similar trends; the improved algorithms outperform the traditional ones, with the Pearson algorithms having a slightly faster computation time than their corresponding adjusted cosine counterparts.

This study doesn't work on the precision, scalability and cold start problems in recommender systems. Therefore all our experiments are about the CPU time. The results agree with the analysis and deduction of the equation regularities. From all the experiments, it is reasonable to conclude that the proposed methods reduce the time spent in evaluation of recommender systems and thus contribute to more efficient design and implementation of recommender systems.

Table 3: Time vs. # of Users – MovieLens 1M

# of Users	Elapsed Time			
	t-pearson	f-pearson	t-acosine	f-acosine
500	62.69	11.79	75.36	11.84
1000	283.25	39.89	326.52	41.85
1500	710.28	80.90	864.48	85.87
2000	1354.8	133.28	1561.70	151.19
2500	1960.52	192.07	2351.96	216.78
3000	2531.29	262.44	3165.22	296.52
3500	3598.12	358.05	4394.13	392.92
4000	4651.64	478.19	6090.39	523.33
4500	6194.16	584.43	7634.36	670.81
5000	7696.86	652.55	8930.91	847.44
5500	9014.55	791.96	10727.60	1010.13
6040	10633.30	987.61	12992.30	1166.62

Table 4: Time vs. # of Users – Jester

# of Users	Elapsed Time			
	t-pearson	f-pearson	t-acosine	f-acosine
1000	164.24	14.09	161.87	13.57
2000	753.91	59.45	745.85	52.00
3000	1640.96	128.22	1590.47	149.71
4000	2789.75	216.90	3247.65	289.68
5000	4241.98	339.97	4450.88	387.23
6000	5530.78	511.69	6634.41	569.72
7000	7490.72	813.13	7932.84	755.42
8000	9686.30	871.01	11279.50	1004.21
9000	10957.90	1117.71	12757.20	1273.27
10000	13981.51	1358.00	15594.90	1552.65
15000	33212.89	3104.16	40123.12	3519.61
20000	61762.19	5512.76	70234.67	6280.87
24983	102311.44	9831.81	112891.23	9920.65

7. Conclusions

To address the challenges of the big data era, the efficiency of developing recommender system algorithms must be improved. Algorithm evaluation is an important part in the development process. In this paper, we explored the mathematical regularities in the MAE/RMSE computations and developed efficient algorithms to evaluate them. This work is an integral part of the process for the design and implementation of recommender system algorithms.

However, in the big data era, as the size of data becomes ever large, algorithms need to be not only efficient but also parallelized. Our future work will combine both efficiency and parallelism to develop improved evaluation algorithms for CF.

8. Acknowledgements

The study is partially supported by the Natural Science Foundation of Hubei Province under Grant No. 2015CFB450, the enterprise-funded latitudinal research project under Grant No. 2014196221, and the Guangzhou Research Infrastructure Development Fund under Grant No. 2012224-12.

References

[1] J. B. Schafer, J. Konstan, J. Riedl, Recommender systems in e-commerce, in: Proceedings of the 1st ACM conference on Electronic commerce, 2000, pp. 158–166.

[2] J. Wang, Y. Zhang, Opportunity model for e-commerce eecommendation: Right product, right time, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2013, pp. 303–312.

[3] J. Lu, Q. Shambour, Y. Xu, Q. Lin, G. Zhang, Bizseeker: A hybrid semantic recommendation system for personalized government-to-business e-services, Internet Research 20 (3) (2010) 342–365.

[4] M. Al-Hassan, H. Lu, J. Lu, A semantic enhanced hybrid recommendation approach: A case study of e-government tourism service recommendation system, Decision Support Systems 72 (4) (2015) 97–109.

[5] H. Kautz, B. Selman, M. Shah, Referral web: Combining social networks and collaborative filtering, Communications of the ACM 40 (4) (1997) 63–65.

[6] X. Liu, K. Aberer, Soco: A social network aided context-aware recommender system, in: Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 781–802.

[7] P. Kazienko, K. Musial, T. Kajdanowicz, Multidimensional social network in the social recommender system, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 41 (4) (2011) 746–759.

[8] Q. He, J. Pei, D. Kifer, P. Mitra, L. Giles, Context-aware citation recommendation, in: Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 421–430.

[9] D. W. McDonald, Evaluating expertise recommendations, in: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, 2001, pp. 214–223.

[10] K. D. Bollacker, S. Lawrence, C. L. Giles, Discovering relevant scientific literature on the web, IEEE Intelligent Systems and their Applications 15 (2) (2002) 42–47.

[11] K. Sugiyama, M.-Y. Kan, Scholarly paper recommendation via users recent research interests, in: Proceedings of the 10th Annual Joint Conference on Digital Libraries, 2010, pp. 29–38.

[12] S. K. Lee, Y. H. Cho, S. H. Kim, Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations, Information Sciences 180 (11) (2010) 2142–2155.

[13] A. B. Barragáns-Martnez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, E. Costa-Montenegro, A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition, Information Sciences 180 (22) (2010) 4290–4311.

[14] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, J. Lu, A hybrid fuzzy-based personalized recommender system for telecom products/services, Information Sciences 235 (20) (2013) 117–129.

[15] F. Zaman, G. Hogan, S. van der Meer, J. Keeney, S. Robitzsch, G.-M. Muntean, A recommender system architecture for predictive telecom network management, IEEE Communications Magazine 53 (1) (2015) 286–293.

[16] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: A survey, Decision Support Systems 74 (6) (2015) 12–32.

[17] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowledge-based Systems 46 (2013) 109–132.

[18] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender Systems: An Introduction, Cambridge University Press, 2010.

[19] M. Zhang, J. Tang, X. Zhang, X. Xue, Addressing cold start in recommender systems: a semi-supervised co-training algorithm, in: Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval (SIGIR'14), 2014, pp. 73–82.

[20] F. Cacheda, V. Carneiro, D. Fernndez, V. Formoso, Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems, ACM Transactions on the Web (TWEB) 5 (1) (2011) 2:1–2:33.

[21] H.-F. Yu, C.-J. Hsieh, S. Si, I. S. Dhillon, Parallel matrix factorization for recommender systems, Knowledge and Information Systems 41 (3) (2014) 793–819.

[22] F. Zhang, V. E. Lee, R. Jin, k-corating: Filling up data to obtain privacy and utility, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14), 2014, pp. 320–327.

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

[24] F. H. del Olmo, E. Gaudioso, Evaluation of recommender systems: A new approach, Expert Systems with Applications 35 (3) (2008) 790–804.

[25] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, Journal of Machine Learning Research 10 (12) (2009) 2935–2962.

[26] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, T.-Y. Liu, A theoretical analysis of ndcg type ranking measures, in: Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013), 2013, pp. 25–54.

[27] M. G. Armentano, R. Abalde, S. Schiaffino, A. Amandi, User acceptance of recommender systems: Influence of the preference elicitation algorithm, in: Proceedings of the 2014 9th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP '14), 2014, pp. 72–76.

[28] F. D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quarterly 13 (3) (1989) 319–340.

[29] S. Ansari, R. Kohavi, L. Mason, Z. Zheng, Integrating e-commerce and data mining: Architecture and challenges, in: Proceedings of the 2001 IEEE International Conference on Data Mining, 2007, pp. 27–34.

[30] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International World Wide Web Conference, 2001, pp. 285–295.

[31] X. Su, T. M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence 2009 (4) (2009) 1–19.