Knowledge-Based Systems 96 (2016) 96–103

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

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

ARTICLE INFO

Article history:
Received 10 April 2015
Revised 28 December 2015
Accepted 29 December 2015
Available online 6 January 2016

Keywords:
Recommender systems
Collaborative filtering
MAE/RMSE
Evaluation

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 the 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.

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 [5–7], Academia [8–11], 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 Fig. 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:

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

* Corresponding author at: School of Computer Science, China University of Geosciences, Wuhan 430074, China. Tel.: +8618627010303.
E-mail address: jeff.f.zhang@gmail.com (F. Zhang).

http://dx.doi.org/10.1016/j.knosys.2015.12.025
0950-7051/

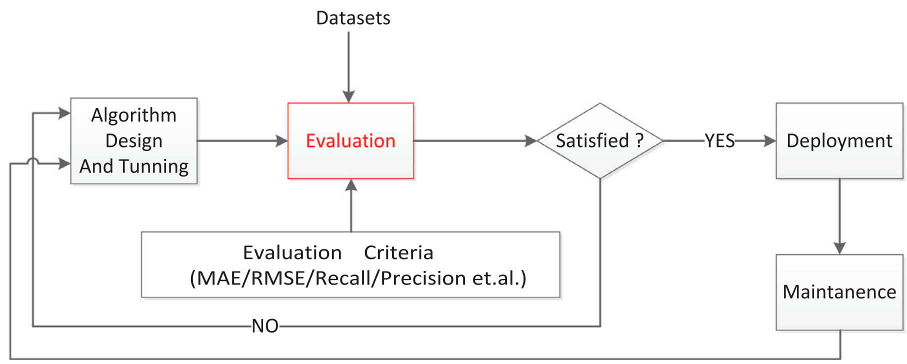

Fig. 1. Flowchart of the Design of a Recommender System.

and the RMSE of the N pairs is evaluated as:

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

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.

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 Fig. 1. As we have pointed out, *evaluation* is an inalienable part of the design process. Since 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:

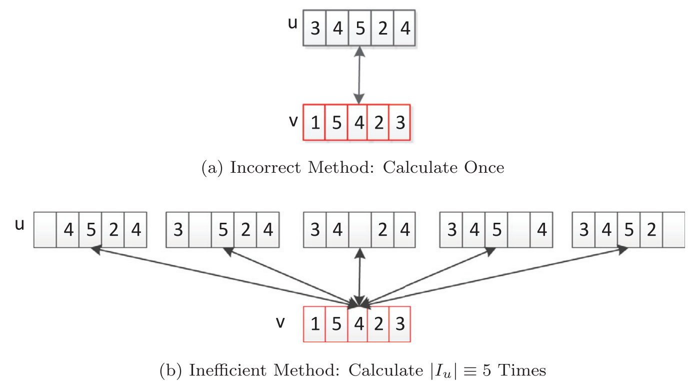

Fig. 2. Methods to Calculate Similarity Between users u and v.

(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, however, they are not equal indeed! The method is illustrated in Fig. 2a.
A correct method to evaluate MAE/RMSE should include the calculation 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 each predicted 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 Fig. 2b.

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] 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.