One Real-time Personalized Recommendation Systems Based On Slope One Algorithm

Zilei SUN
Computer and Information Management Center
Tsinghua University
Beijing, China

Nianlong LUO
Computer and Information Management Center
Tsinghua University
Beijing, China

Wei KUANG
Computer and Information Management Center
Tsinghua University
Beijing, China

Abstract—With the development of recommendation systems, reasonable designed recommendation algorithms and application on actual systems have been important fields of personalized recommendation. System characteristics and users' specific needs become two key considerations of these algorithms. Since many actual systems are user-intensive and their items are significantly classified, we design a real-time personalized recommendation systems based on Slope One algorithm. It shows a good computing complexity in our analysis, and deserves to be widely promoted.

Keywords: collaborative filtering; personalized recommendation; real-time recommendation;

I. INTRODUCTION

Since the concept of personalized recommendation was proposed, recommendation systems and related algorithms have greatly developed in the past twenty years [1][2]. As search engines, personalized recommendation system has been an important tool for Internet users in obtaining useful information from massive data. The development of recommendation system mainly depends on two factors, the accuracy of the recommendation algorithm and the characteristics of users and items in the system. Generally, recommendation algorithm includes the following categories, collaborative filtering method [3], content-based method [4][5], and hybrid method [6], and the recommendation system can also be classified into many types, such as user-intensive, item-intensive, tag-based or classification-based. Only when well combined with recommendation algorithm, recommendation system can be possible to obtain best recommend results. This paper will propose a real-time personalized recommendation system based on a common scene in recommendation system.

II. PROBLEM DESCRIPTION

Considering the specialty of items and users, we can classify recommendation systems into the following two categories. One is general B2C sites, like Amazon and Taobao, which have complex kinds of users and items, whose features are difficult to identify and possibly change dramatically in a short time. The other is some specific websites, like Mtime(Movie website), Douban(book website), which have users with stable hobbies and items with limited types and well classified.

This paper will concentrate on the latter scenario: a knowledge resources management and service system. The main users of this system are teachers and students, which have professional background, and the items are mostly documents and videos, which have specific markup information, including affiliated course, keywords and terminology information. The system is committed to providing users with personalized recommendations according to the behaviors of users and characteristics of items.

According to actual demand, the design of our system will focus on following three goals:

(1)Specialty: Considering the professional background of the users and items, the system should have clear guidance in its recommendation. The recommendation simply depending on the ratings to any items by one user is impracticable.

(2)Real-time: The needs of users are changing all the time. Since traditional recommendation systems often provide the recommended results based on the users' rating history records, which can't satisfy users' current demand. For example, the users of B2C websites often say that they would get full-screen of recommendations with the same type which they bought several days ago. In fact, they don't need this information any more. Although not easy, we need to catch users' latest demand, and provide the appropriate recommendations.

(3)Personalization: Most of the movie websites or book websites, like Mtime and Douban, will recommend the related items to the users according to the item they visit currently. One limitation of these is that this recommendation is not user-based. In other words, different users would get the same recommendation while they were visiting the same item. Obviously, this approach does not meet the development trend of the recommendation system and needs to make changes.

For any ordinary recommendation system, the latter two goals are important indicators to measure its recommended quality. All recommendation system can be divided into four types by these two dimensions as Table 1:
TABLE 3 THE COMPARISON BETWEEN ITEM-BASED AND SLOPE ONE

<table>
<thead>
<tr>
<th>Relationship between items</th>
<th>Item-Based Algorithm $[7]$</th>
<th>Slope One $[8]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{ij} = \frac{\sum_{\alpha \in C_{ij}} \left( r_{\alpha} - \bar{r}<em>j \right) \left( r</em>{\alpha} - \bar{r}<em>i \right)}{\sqrt{\sum</em>{\alpha \in C_{ij}} \left( r_{\alpha} - \bar{r}<em>j \right)^2 \sqrt{\sum</em>{\alpha \in C_{ij}} \left( r_{\alpha} - \bar{r}_i \right)^2}}$</td>
<td>$dev_{ij} = \sum_{\alpha \in C_{ij}} \frac{r_{\alpha} - r_{ij}}{\bar{r}_i}$</td>
<td>Linear operation, computing the number and rating difference of the items that rated by same user.</td>
</tr>
<tr>
<td>Vector operations, computing the similarity of ratings between two items that rated by same user.</td>
<td>$c_{ij} = card(T_i(\chi) \cap T_j(\chi))$</td>
<td></td>
</tr>
</tbody>
</table>

Selecting neighbor items
According to the TOP-N principle $[9]$, selecting the items that have the highest similarity with the target item as neighbor items.

Setting threshold $k$ and selecting the items that same rated by more than $k$ users as neighbor items.

Prediction
$\hat{r}_{ui} = \frac{\sum_{j \in N_k(u)} \left( w_{ui} \cdot r_{uj} \right)}{\sum_{j \in N_k(u)} w_{ui}}$

$R_{ij} = \{ j \mid j \in R_u, j \neq i, \text{card}(T_i(\chi) \cap T_j(\chi)) \geq k \}$

$\hat{r}_{ui} = \frac{\sum_{j \in N_k(u)} c_{ij} \cdot (dev_{ij} + r_{ij})}{\sum_{j \in N_k(u)} c_{ij}}$

$\star R_{ui}$ is the rating of item $I_i$ given by user $U_u$, while $\bar{r}_j$ is the average rating of $I_j$, $T_i(\chi)$ is the set of the users who have rated item $I_i$, and $R_u(\chi)$ is the set of the items rated by $U_u$. Card(S) stands for the number of elements in the set, $c_{ij}$ stands for the number of users that rate both $I_i$ and $I_j$. $dev_{ij}$ is the rating difference between $I_i$ and $I_j$.

TABLE 1 CLASSIFICATION OF RECOMMENDATION SYSTEM

<table>
<thead>
<tr>
<th></th>
<th>Non-Real-Time</th>
<th>Real-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Personalized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Similar to the website</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Make no specified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>requirements for users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and the time.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personalized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Based on users’ rating</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. The latest visit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>behaviors have not</td>
<td></td>
<td></td>
</tr>
<tr>
<td>been taken into account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Combining the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>latest visit behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and the history</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rating records.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Provide different</td>
<td></td>
<td></td>
</tr>
<tr>
<td>results for each user.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In fact, for complex algorithm and vast amounts of data, practical recommended systems can hardly reach the two objectives at the same time. Therefore, we will put forward a new recommendation system, which can be applicable to established scenarios and fulfills the three goals above. It should be a good combination of user-personalization and real-time property, and provide users with the best recommend results.

III. RELATED ALGORITHM

The commonly accepted classification of collaborative filtering based on prediction model: user-based or item-based. Different algorithms have different application scenarios. In practical, they may mainly face the following problems:

TABLE 2 APPLICATION SCENE ANALYSIS OF RECOMMENDATION ALGORITHM

<table>
<thead>
<tr>
<th></th>
<th>User-Based Algorithm</th>
<th>Item-Based Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational</td>
<td>$O(m^2n)$</td>
<td>$O(mn^2)$</td>
</tr>
<tr>
<td>complexity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* $m$ is the number of users and $n$ is the number of items.

In the application scenarios of this paper, the actual user number is close to the item number and may be far outweigh it in the future. Meanwhile in this case, to get better accuracy, item-based algorithm is more suitable. With most computing tasks finished in the background, the overall efficiency of the system will be good and meet the requirement of real-time. In addition, we will do some efficiency optimization based on the obvious classification of items, so it is very appropriate to adopt item-based algorithm here.
The process of item-based algorithm can be expressed as three steps: calculating the relationship between items, screening the neighbor items of the target item, predicting the rating to target item by the ratings of the neighbor items that user rated before.

The comparison of two famous item-based algorithms is shown in Table 3.

Although the two algorithms have the same computational complexity, Slope One algorithm is more efficient in the step of similarity measurement between items, which is critical to the overall computational complexity. Moreover, the accuracy of Slope One is close to Item-Based algorithm, so it will be applied in our system.

IV. REAL-TIME RECOMMENDATION SYSTEM MODEL

A whole recommendation system consists of three modules: behavior record module, model analysis module and recommendation module.

Behavior record module is responsible for recording user behaviors that may reflect their interest, including the items that users bought, the items that users visited, the ratings that users gave, and the time that users spend on an item. In a recommendation system based on collaborative filtering, the most important information is mainly the users’ rating records. Unfortunately, not every system can get enough ratings, and 0-1 rating scheme can be used here, 1 and 0 mean the user visits an item or not. In this case, the final prediction stands for the possibility whether the user would visit the item, not the specific rating.

Model analysis module is the core of the algorithm and its main task is dealing with the data which are collected in the last module. Various modeling and statistical methods will be involved here, and the user’s interests will be constructed in some form.

Recommendation module connects the user and the system, gives out the final prediction based on the result of model analysis module. The final recommendation mechanism will be established in this module, such as TOP-N, threshold-based and so on.

As discussed in last section, the recommendation system can be divided into non-real-time system and real-time system. The former fully comply with the above process, as shown in the following figure:

Figure 1 Collaborative Filtering Algorithm Flowchart

To ensure the real-time feature of recommendation, the design of recommendation system in this paper would be a little more complex. Considering of the computational complexity, some parts that contain a large amount of computation would be moved to the background. The specific flowchart is presented in figure 2.

Figure 2 Real-time Recommendation System Model

Not as figure 1, our system breaks the model analysis module, and two parts of work are finished in background. Firstly, preprocess of user ratings, which involves a lot of computing and database access, should be finished and updated while the system is free. Secondly, the generation of candidate item list, which is close to target item in ratings or content. This design not only improves the pertinence of the recommendation, but also avoids lots of useless predictions.

In the actual operation, when one user visits an item and need some recommendation, the system will obtain the candidate item list of the target item and predict the rating of everyone in it. Higher predicted items would be recommended to the user.

V. SYSTEM DESIGN

This section will introduce the implementation details of our real-time recommendation system by module and discuss the feasibility of this system on processes and algorithm complexity.

A. Rating Pre-Processing Module

The core of this module is the calculation of the relations between item pairs in Slope One algorithm, and relevant factors can be described in following three aspects:

1. the adopted algorithm is item-based,
2. the recommendations users expected may have clear professional-oriented,
3. all items have a clear mark of course id.
In order to best match the needs of the system, we calculate the relations by course and user, and only the item relations in the same course would be considered.

The corresponding tables in the database are shown as follows:

<table>
<thead>
<tr>
<th>Userid</th>
<th>Itemid</th>
<th>Rating</th>
<th>CourseId</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>237</td>
<td>5</td>
<td>52</td>
<td>22-09-2010</td>
</tr>
<tr>
<td>2</td>
<td>237</td>
<td>7</td>
<td>52</td>
<td>12-20-2011</td>
</tr>
<tr>
<td>3</td>
<td>239</td>
<td>5</td>
<td>17</td>
<td>02-01-2011</td>
</tr>
</tbody>
</table>

The process of the algorithm can be expressed as:

For each tempCourseId in courseList {
    Select userid, itemid, rating from dataset order by userid, itemid where courseId= tempCourseId;
    For every userid {
        tempList = userid’s rating list;
        For each tempItemId1 in tempList {
            For each tempItemId2 in tempList {
                Dev(tempItemId1, tempItemId2) = rating(userid, tempItemId1) - rating(userid, tempItemId2);
                CommonUsers(tempItemId1, tempItemId2)++;  //Stored by orthogonal list
            }
        }
        //Stored by orthogonal list
        Write back to Item-correlation-matrix
    }
}

The average-case computational complexity of the algorithm is $O(k_2k_3m)$, where $k_1$ is the number of courses and $k_2$ is the average item number every user has visited. Moreover, I/O operation of database has a computational complexity of $O(n^2/k_1)$.

B. Item Training Module

Same as the last module, item training module will be executed in the background. We adopt LDA algorithm to classify the documents. Then we can compute the neighbor items by similarity of items’ topic distribution. The algorithm will run beforehand to generate models. It has a computational complexity of $O(knw)$, where $k$ is iteration count, and $w$ is the average number of words in every item.

Both two backstage modules have acceptable computational complexity.

C. Real-time Recommended Module

Real-time prediction mainly includes the following steps:

1. When one user pays a visit to a target item, the system would get its neighbor items from the database, including the neighbors based on ratings which was calculated in rating pre-processing module, and the neighbors based on content which was generated in item training module.

2. The system would read the rating list of current user from Rating Table, and all items from Item Relation Table that related to the neighbor items mentioned above. To speed up the database I/O, clustered index for column ‘Itemid1’ would be create in Item Relation Table and the complexity of database access would be $O(\log_2 m + k_4 \log_2 n)$, where $k_4$ is the length of candidate item list.

3. Real-time prediction uses the formula of Slope One algorithm:

$$r_{i,j} = \frac{\sum_{c \in R(i,j)} \frac{c}{\sum_{\bar{c} \in R(i,j)} c} + r_{i,j}}{\sum_{c \in R(i,j)} c}$$

The average-case computational complexity of each prediction is $O(k_2 + k_4)$, where $k_2$ is the average item number every user has visited, and $k_4$ is the average number of related items of each item.

Then the whole real-time prediction has an average-case computational complexity of $O(k_2(k_2 + k_4))$ and a database access complexity of $O(\log_2 m + k_4 \log_2 n)$, which are acceptable for a real-time recommendation system.

VI. CONCLUSION

With the development of recommendation systems and recommendation algorithm, the accuracy of the algorithm is no longer the only research hotspots. Algorithms and application on actual systems have been important fields of personalized recommendation. Especially, System characteristics and users’ specific needs become two key considerations of these algorithms. In this paper, combining the actual knowledge resources management and service system, we design a real-time personalized recommendation system based on Slope One algorithm. In actual, many real systems have more users than items and its items have a professional classification, and our design can be well applied in these systems. It shows a good computing complexity in our analysis, and deserves to be widely promoted.

ACKNOWLEDGMENT

The work was supported by the National High Technology Research and Development Program of China (863 Program) (No. 2008AA01Z131).

REFERENCES


[8] Lemire D, MacLachlan A, Slope One Predictors for online rating-based collaborative filtering. In: SIAM Data Mining (SDM’05), Newport Beach, California, April 21-23, 2005
