A Recommender System for Ordering Platform Based on An Improved Collaborative Filtering Algorithm

Chengchao Yu[a], Qingshi Tang[b], Zheng Liu[a], Bin Dong[b], Zhihua Wei*[c]

[a]*Department of Computer Science and Technology, Tongji University, Shanghai201804, China*
yuchengchao@tongji.edu.cn, liuzheng@tongji.edu.cn

[b]*Shanghai Zhuoji Information Technology ltd., Shanghai200001, China*
tangqs@dajialai.cn, dongbin@dajialai.cn

[c]*Key Laboratory of Embedded, System and Service Computing, Tongji University, Shanghai201804, China*
zhihua_wei@tongji.edu.cn

***Abstract*—With the development of ordering platform, an increasing number of people are paying their attention to design a suitable recommender system. Most of the traditional recommender systems are based on the abundant rating information of users. However, Only historical order data can be provided to the recommender system in ordering platform as training data. This paper proposes an improved Collaborative Filtering algorithm based on historical order data of restaurants. The recommender system includes two parts: 1) rule generation module, we define a new method for measuring the similarity between dishes. Furthermore, we incorporate an incremental learning method in this module. 2) recommendation module, we design user interest vector and propose a noise filtering method. Experimental results demonstrate that the proposed algorithm can effectively improve the performance of recommendation in terms of the accuracy and coverage ratio. Moreover, our recommender system has been successfully put into service.**

***Keywords*—recommender system, collaborative filtering algorithm, incremental learning**

I. Introduction

Recommender Systems attempt to recommend items to fit a user's interests from an overwhelming set of choices. Such systems are widely used in a variety of applications such as e-commerce [1] (Netflix, Amazon, Alibaba, eBay, etc.), information retrieval [2] (iGoogle, MyYahoo, Baidu, etc.),mobile applications [3]. With the rapid development of self-service ordering system, recommender systems are especially important in markets. When a customer uses a terminal application to order dishes, the customer is often unable to quickly find a satisfactory dish from the menu. Hence, a good recommender system for food dishes allows users not only to obtain a satisfactory menu, but also to enhance the turnover of the restaurant.

At present, the main recommender system algorithms include: content-based algorithm [4], collaborative filtering algorithm (CF) [5], association rules algorithm [6], Singular value decomposition (SVD) algorithm [7] and hybrid recommendation algorithm [8]. These recommendation algorithms focus on the binary relationship between the user and the item, and most of them can be transformed into a scoring problem. A recommended list is generated based on ranking the items according to the scores of items. In general, it is difficult to collect the necessary information about foods and customers in the recommendation system for our ordering system. Therefore, the recommended algorithm based on collaborative filtering is the suitable approach because it makes use of past user activities.

The CF algorithms recommend items to users based on other users' hobbies and past activities. It can be divided into two categories: the CF algorithms based on users [9] and the CF based on items [10]. The CF algorithm based on the users is to find the users that have similar interests with the target user and then recommend the items that the target user did not contact but might be interested to the target user. However, as the number of users grows, it will become increasingly difficult to calculate the user's interest matrix, hence it isn't suitable for the recommender system for our ordering system due to the requirements of the time performance. The type and quantity of food for each brand is basically in a steady state, which makes the algorithm of CF based on items be meaningful to realize. It recommends the items similar to those they liked before to the target users, calculating the similarity between items by analyzing the user's behavior records. The traditional CF algorithms based on items take the item-user matrix R as input and each element $R_{i,j}$ in the matrix represents the jth user's score for the ith item [11]. However, for our ordering system, we only have historical orders, which lacks scoring information, therefore we propose a new input matrix $R^{food-order}$ and a new similarity measure principle to implement the CF algorithm based on items.

In this paper, we designed and implemented a recommender system for food dishes based on the improved CF algorithm, which includes modules such as rule generation, incremental

*Corresponding authors.

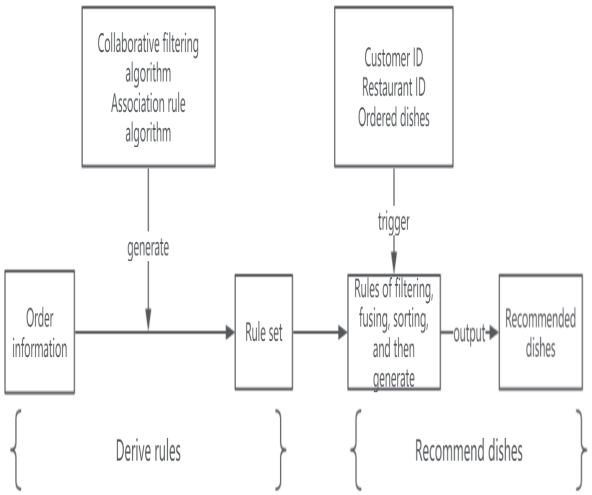

Fig. 1. The diagram for our recommender system

learning, recommendation, food's hotness degrading. The recommender system has been successfully served on $Zhuoji$'s ordering platform and has got great feedback.

II. Methodology

Previous research has focused on constructing the user profile [12] based on data with rich information. Due to the flaws in the data of the ordering system, we have made improvements based on traditional recommendation algorithms. On considering the efficiency of the ordering system, the recommender system is divided into two modules: rule generation module and recommendation module, which is illustrated in Fig. 1.

A. Rule Generation Module

The purpose of this module is to train in advance and get the recommended rules, hence the job of the recommended module is to match rules or complete simple arithmetic of rules, which can ensure the effectiveness and feasibility of the recommender system. In fact, the structures of rules are different when using different recommended algorithms. The rule structure defined by the FP-Growth algorithm [13] is frequent item sets and you can get the matching scores between the food ordered and frequent items to recommend appropriate dishes. However, we define the similarity matrix R^{n*n} between items as a rule structure for the CF algorithm, n is the total number of food.

Given the historical orders, we present it as the user's vector $U_k(x_1, x_2, \ldots, x_n)$ and then the food's vector will be presented as $F_z(u_1, u_2, \ldots, u_m)$, m is the total number of historical orders.

$$x_i = \begin{cases} 1 & when\ ith\ food\ was\ ordered\ by\ user\ k \\ 0 & otherwise \end{cases} \quad (1)$$

$$u_j = \begin{cases} 1 & when\ jth\ user\ ordered\ the\ zth\ food \\ 0 & otherwise \end{cases} \quad (2)$$

When we take U_k and F_z as input to the rule generation module, we can easily define its similarity function between food A and food B.

$$COS(A,B) = \frac{\sum_{i=1}^{n}(A_i * B_i)}{\sqrt{\sum_{i=1}^{n}(A_i^2) * \sum_{i=1}^{n}(B_i^2)}} \quad (3)$$

When $A, B \subseteq F_z$, we find that the result of $A * B$ stands for the frequency that food A and food B were ordered jointly and $|A|^2$ can be considered as the total times that food A were ordered, therefore we define the similarity function $SIM(A,B)$ as follows: When we take U_k and F_z as input to the rules generation module, we can easily define its similarity function between food A and food B.

$$SIM(A,B) = \frac{C(A,B)}{\sqrt{(T(A) * T(B))}} \quad (4)$$

The C stands for the function that calculating the co-occurrence number of food A and food B, and T is the function to calculate the total times of the food that were ordered. By using the $SIM(A,B)$, we finally get the recommendation rules R^{n*n} and store it into our database.

Incremental learning [14] refers to a learning system that continuously learns new knowledge from a new sample and preserves most of what has been learned before. It's important for the recommendation algorithm of ordering system due to the foods may change for each brand, which can improve the flexibility of the recommender system. Thanks to the $SIM(A,B)$, we can design an effective incremental learning strategy, which makes it possible to apply the recommendation algorithm to the ordering system.

$$SIMINL(A,B) = \frac{O(C(A,B)) + N(C(A,B))}{\sqrt{((O+N)T(A) * (O+N)T(B)}} * \sigma \quad (5)$$

The function O aims to get the old model's parameters that has been learned before and the function N is to learn the new knowledge from new orders. σ is the factor of food's hotness degrading [15]. Therefore the value of $SIMINL(A,B)$ will be treated as the new rules to update the similarity matrix R^{n*n}.

B. Recommendation Module

When the service of recommender system is turned on, all the rules will be transferred to the RAM of the server and then it will provide the recommended service to the ordering system. In the recommended module, the most critical task is to describe the user's interest vector $I(v_1, v_2, \ldots, v_n)$.

$$v_i = \begin{cases} 1 & when\ ith\ food\ has\ been\ ordered\ by\ user \\ Like(i) & otherwise \end{cases} \quad (6)$$

The function $Like(i)$ attempts to score the degree of user preference for the ith food.

Given a vector of the unfinished order U, we need to convert it into the user's interest vector I. For our recommendation module, the interest degree is measured by the strategy that finding the dishes that are most similar to those that have been ordered. Hence, we define the function $Like(i)$ as follows:

$$Like(i) = \sum_{j=1}^{w} SIMR(i,j), \qquad when\ j \subseteq H \quad (7)$$

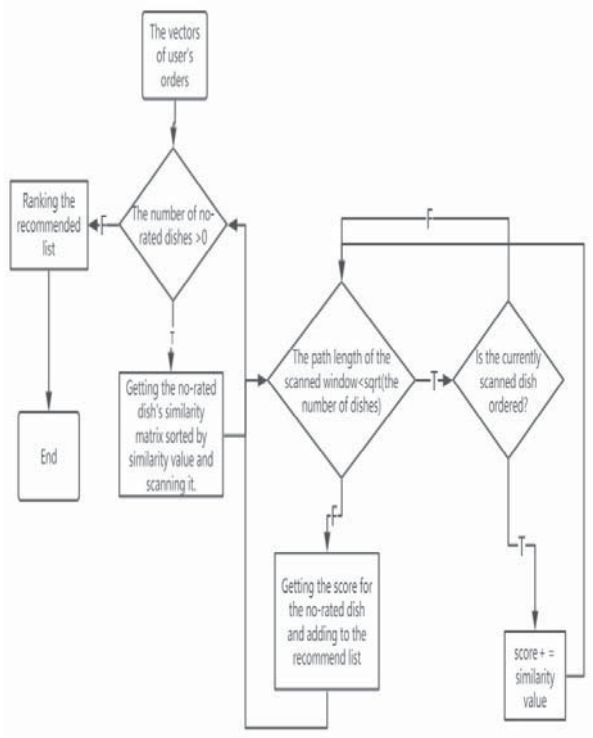

Fig. 2. The flow diagram for recommendation module

The function $SIMR(i, j)$ is a descending sort of rule set based on the value of $SIM(A, B)$. The H is a collection of all dishes that have been ordered. The w is a scanned window, which has two advantages: reducing the calculation time of the function $Like(i)$ and filtering some noise in the rule's database.

$$w = \sqrt{n} \tag{8}$$

When we get the user's interest vector $I = (v_1, v_2, \ldots, v_n)$, we will filter the dishes that have been ordered and then sort it based on the value of $Like(i)$. Finally the recommender system will send the top-k food id to the client.The flow chart of the recommendation module is shown in Fig. 2.

III. Experiments

A. Experimental Dataset

To facilitate research on the task of our recommender system, we take 12641 orders adopted by one restaurant as data, which contains 122 dishes. In this paper, the FP-growth algorithm and the CF based on items algorithm are experimented respectively. We take 10,000 orders as training set and 2641 orders as test set.

B. Experimental Setting

Training: The parameters of the two algorithms are set as follows:

1) The FP-growth algorithm: In our FP-growth model, the value of the Minsupp was set to 0.01 and the value of the Minconf was set to 0.8, which are important parameters in the FP-growth algorithm.

TABLE I
Results of test

Models	Coverage Ratio	Accuracy
FP-growth	83.2%	**74.3%**
CF based on items	**99.0%**	71.4%

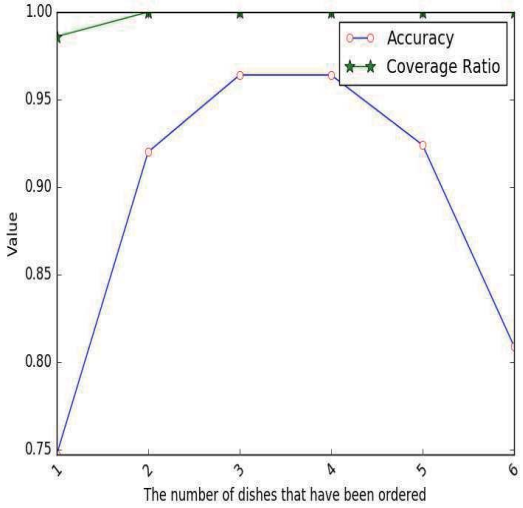

Fig. 3. The results of different dishes that have been ordered

2) The CF based on items algorithm: In the CF model, we set the length of sliding window to $\sqrt{n}$, n is the total number of dishes served by a restaurant. In addition, we set the incremental interval to 24h.

Evaluations: We evaluated the performance of the proposed models on two metrics:

1) Coverage Ratio, i.e., the ratio of dishes that have been ordered in the rule base. In other words, it can be expressed as the ratio that the quantities of orders that can get recommended results R and total orders T,

$$Coverage\ Ratio = \frac{R}{T} \tag{9}$$

2) Accuracy, i.e., the ratio of correctly recommended dishes.

C. Experiment Results and Analysis

We randomly selected 2141 orders from the test set, and manually removed the error cases. Through the FP-growth algorithm, we obtained 2641 recommended rules. Table 1 shows the coverage ratio and accuracy of the FP-growth model and the CF model.

The FP-growth model achieves satisfactory results in terms of accuracy, thanks to its powerful ability of mining the frequent item sets, which can perform well in recommendation. However, it can not meet the requirement of actual application in terms of coverage ratio, which can lead to the situation where it is often not possible to recommend dishes.

TABLE II
THE RESULTS AFTER REMOVING POPULAR DISHES

NumOD	RN=0	RN=1	RN=2
1	74.8%	70.6%	65.4%
2	92.0%	87.7%	79.7%
3	96.4%	90.5%	78.0%
4	96.4%	85.3%	68.7%
5	92.4%	72.6%	54.5%
6	80.9%	52.3%	40.8%

The CF model achieves a lower accuracy than the FP-growth model, but it still performs well in terms of accuracy. In addition, we can draw the conclusion that the advantage of the CF algorithm is its high coverage ratio, it can meet the requirements of the actual applications. The CF algorithm based on items recommends dishes by comparing the similar dishes, hence the dishes can be recommended when the dishes already ordered have appeared in the training data set.

In Fig. 3, it shows the performance of the CF model with the increase in the number of dishes that have been ordered. The graph of accuracy first increases, and then decreases. With the number of ordered dishes increasing, the CF model can get more information on the dishes, hence the performance of the recommendation shows an upward trend. However, the number of dishes in each order of a restaurant is usually kept on a constant M. Therefore, the noise of the dish information becomes larger when the number of dishes that have been ordered exceeds M, which will have a bad effect on the recommended results.

The popular dishes of each restaurant will undoubtedly be the preferred first choice in many cases for the recommender system, but in fact, the managers of restaurants often want to recommend some suitable dishes besides the popular dishes. Therefore, we also did some experiments on testing the performance of the recommender system after removing the popular dishes. The experimental results show that our recommender system still performs well, which is illustrated in Table 2. The $NumOD$ is the number of dishes that have been ordered and the RN is the number of popular dishes that be removed.

The managers of some restaurants are likely to change menus, so incremental learning is essential for recommender systems that using historical orders as a training set. The incremental learning results based on Eq. 5 are shown in Fig. 4. The curve of accuracy shows a trend of partial steady and an overall increase with time, which testified the effectiveness of incremental learning for our recommender system.

As shown in Fig. 5, it has been successfully served on $Zhuoji$'s ordering platform. The dishes in the black-bordered box are the top 4 recommendations and the dishes that have been ordered are enclosed in the blue-bordered box. The recommendations have gotten great feedback from customers.

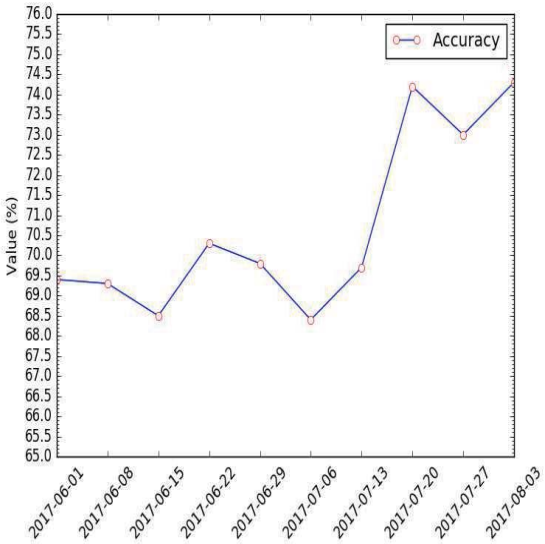

Fig. 4. The results of incremental learning

Fig. 5. The recommender system for $Zhuoji$'s ordering platform

IV. CONCLUSION

In this paper, we proposed an improved collaborative filtering algorithm for the recommender system. It not only made full use of historical orders, but also took greater advantage of real-time orders to improve the accuracy of recommendation. Experimental results demonstrated that, compared with the FP-Growth algorithm, the proposed algorithm performed almost the best in each metric, especially in the metric of coverage ratio, it almost met the demand of the practical applications. Although slightly lower in accuracy than the FP-growth algorithm, the proposed algorithm still performed well in practical applications. Future work will focus on dealing with the instability of the the accuracy ratio with the increasing number of dishes that have been ordered.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61573259.

REFERENCES

[1] H. Xu, R. Zhang, C. Lin, and W. Gan, "Construction of e-commerce recommendation system based on semantic annotation of ontology and user preference," *Telkomnika Indonesian Journal of Electrical Engineering*, vol. 12, no. 3, 2013.

[2] Y. Gupta, A. Saini, and A. K. Saxena, "A new fuzzy logic based ranking function for efficient information retrieval system," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1223–1234, 2015.

[3] L. O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, and J. J. Samper-Zapater, "Recommetz: A context-aware knowledge-based mobile recommender system for movie showtimes," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1202–1222, 2015.

[4] L. Iaquinta, M. D. Gemmis, P. Lops, G. Semeraro, M. Filannino, and P. Molino, "Introducing serendipity in a content-based recommender system," in *International Conference on Hybrid Intelligent Systems*, 2008, pp. 168–173.

[5] A. Hegde and S. K. Shetty, "Collaborative filtering recommender system," *Esrsa Publications*, vol. V4, no. 3, 2015.

[6] C. Kim and J. Kim, "A recommendation algorithm using multi-level association rules," in *Ieee/wic International Conference on Web Intelligence*, 2003, p. 524.

[7] R. Gupta, A. Jain, S. Rana, and S. Singh, "Contextual information based recommender system using singular value decomposition," in *International Conference on Advances in Computing, Communications and Informatics*, 2013, pp. 2084–2089.

[8] M. Salehi and I. N. Kmalabadi, "A hybrid attribute–based recommender system for e–learning material recommendation," *Ieri Procedia*, vol. 2, no. 4, pp. 565–570, 2012.

[9] G. H. Cheng, "A collaborative filtering recommendation algorithm based on user clustering in e-commerce personalized systems," *Advanced Materials Research*, vol. 267, pp. 789–793, 2011.

[10] B. M. Kim and Q. Li, "Probabilistic model estimation for collaborative filtering based on items attributes," in *Ieee/wic/acm International Conference on Web Intelligence*, 2004, pp. 185–191.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *International Conference on World Wide Web*, 2001, pp. 285–295.

[12] D. Bouneffouf, "Towards user profile modelling in recommender system," *Computer Science*, 2013.

[13] D. Zhang, D. Zhang, D. Zhang, M. Zhang, and E. Y. Chang, "Pfp: parallel fp-growth for query recommendation," in *ACM Conference on Recommender Systems*, 2008, pp. 107–114.

[14] T. Yu, O. J. Mengshoel, A. Jude, E. Feller, J. Forgeat, and N. Radia, "Incremental learning for matrix factorization in recommender systems," in *IEEE International Conference on Big Data*, 2017, pp. 1056–1063.

[15] M. K. Su, E. P. Lim, and F. Zhu, *A Survey of Recommender Systems in Twitter*. Springer Berlin Heidelberg, 2012.