2016 3rd International Conference on Information Science and Control Engineering

A scalable recipe recommendation system for mobile application

ZhengXian Li, Jinlong Hu[*]*, Jiazhao Shen, Yong Xu*
School of Computer Science and Engineering
South China University of Technology
Guangzhou, China
jlhu@scut.edu.cn

Abstract—*There are more and more users to use mobile application to get the cooking recipe information, but to recommend right recipe for each user is challenging for the complex food and diet information. In this paper, a hybrid recommendation system is proposed for personalized recipe mobile application. We describe a scalable recommendation system to process the massive data from mobile applications based on the spark clusters, and design a hybrid recommendation algorithm, which combines content-based filtering and collaborative filtering to improve the recipe recommended effectiveness. Our experiments reveal that the recommendation system has scalable computational capability.*

Keywords-recipe recommendation system; mobile application; hybrid recommendation

I. INTRODUCTION

Recommendation System is known as an effective solution for information overload and there are many recommendation research achievements have been made from academia and industry [1-2]. By diming a binary relation between user-item, recommendation system can help user find what they are interested in, and build personalized recommendation to meet their needs. Recent years, the term kitchen economics has been risen up, and mobile apps for providing recipe information has gradually become the focus in our life.

However, the growing mobile Internet services and content will gradually exceed the acceptable range of people. Another factor resulting mobile information overload is, mobile devices have limitations in displaying, terminal handling, input or output capabilities. Ricci et al. [3, 4] proved that those which can be applied to traditional recommendation system was not that effectively to mobile recommendation system, since it has higher requirements in *User Mobility*, *Device Mobility* and *Wireless Access*.

In this paper, a scalable recommendation system based on Apache Spark [5] are designed and implemented for personalized recipe recommendation for mobile apps. In the recommendation system, we also propose a hybrid recommendation algorithm, which combines content-based filtering and collaborative filtering, to improve the recipe recommended effectiveness. In the end, we evaluate the performance of the recommendation system.

II. RELATED WORKS

Spark is an open source implementation of memory-based MapReduce framework [6, 7]. While inheriting Hadoop [8] MapReduce's thought of divide and conquers, Spark plays its memory-based computing advantages, and stored the intermediate output in memory instead of disk in Hadoop, thus reducing disk IO and gain a higher efficiency.

One of the widely used recommended algorithms is item-based collaborative filtering algorithm [9]. Using user's scoring record for item, the collaborative filtering (CF) system computes the similarity of users or goods, and the TopN items with highest score will be recommended to active user. However, it has been proved that the algorithm seems not to work with a high accuracy as expected since the traditional CF approach drops much significant information such as time, place, or companion. Adomavicius et al. [10] proposed a multidimensional collaborative filtering algorithm to filter unwanted data according to the contextual information. By adding other useful information such as time and place, it improved the output accuracy. The MAR-CF, proposed by Ahn et al.[11], and Critique-based Mobile recommender system by Ricci [12], taked mobility of mobile Internet into consideration to come up with their models. Jiao Yu-peng [13] used Mahalanobis distance based FSVM [14] to identify customer's purchase behavior, and predict it by the computing model.

In the recent years, it was proved that using a hybrid model for recommendation worked much better than a single one in many cases. It was because hybrid approach used additional data input which was limited only to another preferred model, or the data was firstly enhanced, and then actually used by other models [15]. M. A. Ghazanfar et al. [16] and A. Gunawardana et al. [17] proposed different hybrid recommendation system and behaved effectively.

In addition, cold-start and the sparse data problem could seriously affect the final recommendation effects. The researchers have used a variety of methods to deal with this problem, such as Co-Clustering smoothing based CF algorithm introduced by Wei et al. [18].

III. DESIGN AND IMPLEMENTATION OF THE SYSTEM

A. Overall Architecture

Generally, specific recommendation systems have different target users in different areas, and the hybrid recommender system in this paper is designed for recipes personalized recommendations, and provides users with recipes browsing service. Most traditional apps use client-server architecture, with client providing user interaction interface and server business processing and data maintenance. However, the variety of recipe always leads to information overload. Thus, the system in this paper adds a personalized recommender engine into it, so that for different user we can recommend what he or she is more interested in.

The framework of the hybrid recommendation system for recipe mobile apps is show as Figure 1. There are three layers, Data Layer, Server Layer and Client Layer in the framework.

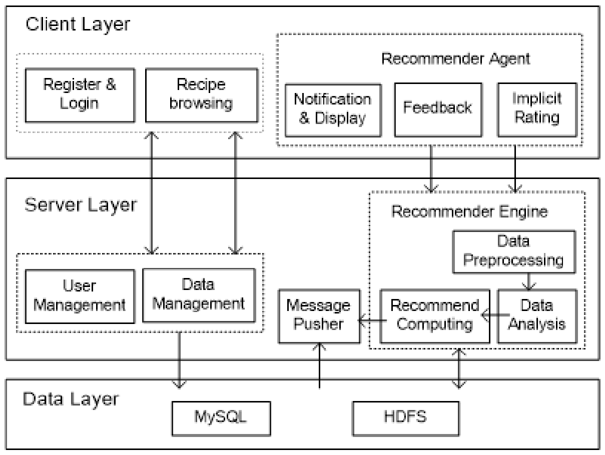

Figure 1. Framework of the Recommendation System

As showed in Figure 1, our system cover two part in the Server Layer and Client Layer, business processing subsystem and recommendation engine subsystem. The business processing subsystem work alike as a traditional C/S architecture system, which may include register and login, recipe browsing and other common operations in client, and providing user management, data management in server. The recommendation engine subsystem could also be divided into client and server part. The client agent collects and uploads user's ratings, feedback or other online behavior to the recommender engine, then the server engine will take series of analysis and computing, finally the engine pushes the recommendation result to client or user part.

B. RECOMMENDER ENGINE

The recommender engine includes engine part at server layer and agent part at client layer. There are notification and display module, feedback module and implicit rating module at agent part, and data preprocessing module, data analysis module and recommend computing module at server part.

When users active in applications, it will generate lots of behaviors such as rating, browsing, quit app, comments, etc., behind which tell users' potentially interest. Therefore, collecting the behavior log, then uploading to the server is one of the important steps to guarantee the engine capable of running normally.

Data analysis and preprocessing: Most of data uploaded are log files, which mostly are in the form of plain text, and cannot be directly involved into the operation. So we need data analysis and preprocessing, including but not limiting to fill in the missing options, duplicate the redundant data. Another important part of task during preprocessing is to map text data to numeric data for distributed platform data processing. All of processed data mentioned above will be stored in distributed file system. The whole data processing cycle is showed as Figure 2.

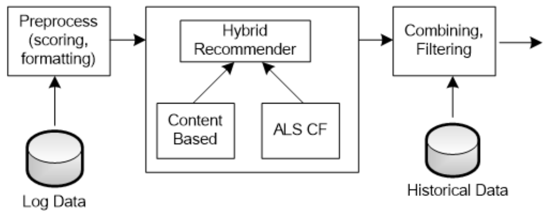

Figure 2. Data Processing Cycle

Recommendation computing: Recommendation computation is the core of the recommendation engine module, it acts on the dataset and conclude a recommended list. Different from most of recommendation on traditional e-commerce, the target product in this paper is recipe, which means if user make a specific dish, he or she might be supposed to make it again in a short future, whereas in e-commerce when a user buy some products, he or she will be a little bit possible to buy the same in a short time. Therefore, rating engineering should be acted on both rated and not-rated-yet recipe, that means, when using recommendation module to compute rating for a recipe, we should take the rated one into consideration, and then eliminate them from the final recommendation list.

C. Implementation

To implement the system, we choose Android mobile platform as our client, and J2EE for server. We deploy Apache Spark distributed computing platform as our recommendation engine.

For store and manage the various data, the business processing subsystem use MySQL to maintain recommendation result or other metadata. Meanwhile, a Hadoop Distributed File System (HDFS) is selected for the engine subsystem need to store unstructured data such as user profile, user behavior log data etc.

IV. RECOMMENDER ALGORITHM

In this paper we use a hybrid system with pipelined recommender approach to improve the recipe recommended effectiveness, as showed in Figure 3.

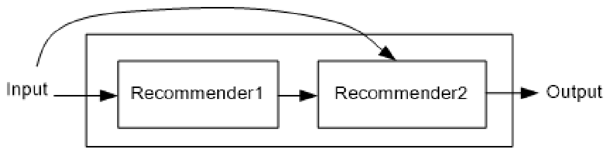

Figure 3. Pipelined Recommender Approach

In the hybrid recommender system, we use content-based algorithm [19] and model-based CF algorithm [20]. CF will use a group of user information (ratings for example) to generate the recommendation result, but often ignore the impact of product features to users, and CF cannot work well for a new user or a new product because of the cold start problem. CB approach relies on product features and text description, and it never has cold start problem as CF does, but listing a series of question to collect users' interest can always degrade the user experience.

To address the above problem, we combine the two algorithms, content-based algorithm and model-based CF algorithm. In this paper, we focus on the case, as showed in Table 1 and Table 2.

Table 1. user behavior to recipe (1 for like, null for dislike)

User	Recipe1	Recipe2	Recipe3	Recipe4	Recipe5
Target		1		1	
User1		1	1		
User2	1	1			1
User3					1

Table 2. recipe taste

Recipe	Taste
recipe1	Piquancy
recipe2	Light
recipe3	Light
recipe4	Light
recipe5	Piquancy

If we use CF to compute the similarity of 'like' for recipe, the target user seems to have similar interest with user 1 and user 2, cause they are all mark 'like' in recipe 2. However, according to the taste of dish (content-based), we can observe that user 1 has higher similarity with target user, because we can know from Table 2 that, user 2 like both piquancy and light food, while user 1 only like light taste as target dose. This scenario illustrates that using CB acts as supplementary to CF algorithm, allows users with same taste closer, making a higher accuracy of recommendation.

A. Content-Based Recommendation

Content-based recommendation systems analyze item descriptions to identify items that are of particular interest to the user. Although the details of various systems differ, content-based recommendation systems share in common a means for describing the items that may be recommended, a means for creating a profile of the user that describes the types of items the user likes, and a means of comparing items to the user profile to determine what to recommend. The profile is often created and updated automatically in response to feedback on the desirability of items that have been presented to the user [19].

B. Model-Based CF algorithm

Model-Based CF algorithm applies machine learning theory to the recommendation system to predict the rating. One of well-known model-based CF is a LSA [21] based one, the thought goes as following.

Assume all of the item can be divided into N classes, use a numeric value to represent user's preference to the classes, the greater of the value tell the stronger preference, generating a User-Class matrix. Using another value to explain the represent degree for the classes to items, and generate a Class-Item matrix. It's obviously that when a user has more preference on some class, the typical item of this class will get a high score. Take User-Item matrix [22] as notation R, according to formula (1), we can factor R as the form of matrix product.

$$R = U^T M \tag{1}$$

Where U represents User-Class matrix, M for Class-Item matrix. Using a 'user – item - rating' dataset to train the model to generate U and M. It's fixed loss function [22] showed as formula(2)

$$f(U,M) = \sum_{(i,j)\in I} (r_{ij} - u_i^T m_j)^2 + \lambda\left(\sum_i n_{u_i}\|u_i\|^2 + \sum_j n_{m_j}\|m_j\|^2\right) \tag{2}$$

I means rated user-item pair, r_{ij} is the element of matrix R, λ for regularization, u_i^T for the row of UT, m_j for the column of M, n_{u_i} represents the total number of item that user u_i has rated, n_{m_j} represents the total numbers of the user who have rated for m_j. To get the minimum of function $f(U,M)$, we use ALS algorithm. And then, we could predict that rating of those not-rated-yet user item pair through U and M. Generating the whole rating matrix, we then could make our recommendation list.

V. EXPERIMENT

In the former part we describe the architecture and the recommender algorithm, now in this section we will take a performance experiment, and anlysisthe system's concurrent ability.

The Spark cluster has been setup with five nodes, one for master and the other for workers, each of which has 8GB memory. We take 'ML-20m MovieLens' from GroupLens [23] as our dataset, the dataset includes 20,000,263 ratings from 138,493 users on 27,278 movies. In the test, select 40,000, 80,000, 120, 000 and 138,493 of the total users and their ratings as the source input, for every different data input, do the following operation: store in the HDFS, the back-end read and preprocessing the rating records, and then finish the model training , prediction and generating recommendation list. To get a better model, in data splitting step, we try to cover training, validation, and test set with all the different ratings(from score 1 to 5). Then, for every data input, given an increasing number of worker node, we observe variety state of the running time.

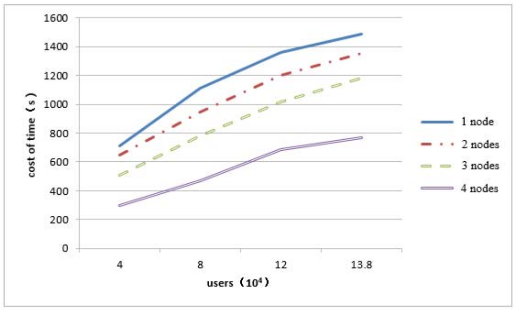

Fig 4. Performance test of the system

In Figure 4, we can find that for every single data input, increasing the numbers of worker nodes can help decrease the running time, the more of nodes, the less cost of time, that means the system has a nice parallelism. From another perspective, given a constant number of nodes, it shows that with data input increasing, the curve of the running time tends to flatten, which illustrates the system has nice computational capabilities.

I. CONCLUSION

In this paper, we proposed a hybrid recommendation system for personalized recipe mobile application. A hybrid recommendation algorithm, combining content-based and collaborative filtering, is designed to improve the recipe recommended effectiveness. Based on the Spark clusters, the recommendation system is scalable to process the massive data from mobile apps. Our experiments reveal that the recommendation system has scalable computational capability to process massive recipe information.

ACKNOWLEDGMENT

This work was supported in part by the Science and Technology Planning Project of Guangdong Province, China (No.2013B090500087, No.2014B010112006), and the Scientific Research Joint Funds of Ministry of Education of China and China Mobile (No.MCM20150512).

REFERENCES

[1] Adomavicius G, Tuzhilin A. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. on Knowledge and Data Engineering, 2005,17(6):734 749. [doi: 10.1109/TKDE.2005.99]

[2] Resnick P, Varian HR. Recommender systems. Communications of the ACM, 1997,40(3):56 58. [doi: 10.1145/245108.245121]

[3] Ricci F. Mobile recommender systems. Int'l Journal of Information Technology and Tourism, 2011,12(3):205-231.

[4] Averjanova O, Ricci F, Nguyen QN. Map-Based interaction with a conversational mobile recommender system. In: Proc. of the Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2008). Washington: IEEE Computer Society, 2008. 212-218. [doi: 10.1109/UBICOMM.2008.16]

[5] Apache Spark, http://spark.apache.org/

[6] Jeffrey Dean, and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. OSDI, page 137-150. USENIX Association, (2004).

[7] J. Dean, S. Ghemawat, MapReduce: Simpli ed Data Processing on Large Clusters,OSDI 2004.

[8] Apache Hadoop, http://hadoop.apache.org/

[9] Sarwar,Badrul; Karypis,George; Konstan,Joseph; Riedl,John (2001). Item-based collaborative filtering recommendation algorithms[J]. Proceedings of the 10th international conference on the World Wide Web (ACM): 285–295.

[10] Adomavicius,G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A., Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. ACM Transactions on Information Systems(TOIS), 23(1):103-145, 2005

[11] Ahn,H., Kim,K., and Han,I., Mobile advertisement recommender system using collaborative filtering: MAR-CF. In Proceedings of 2006 Conference of the Korea Society of Management Information Systems, pages 709-715.

[12] Ricci F, Nguyen QN (2007) Acquiring and revising preferences in a critique-based mobile recommender system. Intell Syst IEEE 22(3):22–29

[13] YP Jiao(2015). “Application of FSVM Based on Mahalanobis Distance for User Behavior Recognitio vior Recognitio” Science and Technology Vision.

[14] Inoue T.; Abe, S.. Fuzzy support vector machines for pattern classification. Neural Networks, 2001. Proceedings. IJCN”O1. International Joint Conference on, Volume: 2,2001, Page@): 1449 - 1454

[15] Jannach, M. Zanker and G. Friedrich “Recommender Systems: An Introduction”, Cambridge University Press, 2010

[16] M. A., Ghazanfar and A. Prugel-Bennett, “A scalable, accurate hybrid recommender system”. Proceedings of the 2010 Third international conference on Knowledge Discovery and Data Mining, pp. 94-98, 2010.

[17] Gunawardana and C Meek, “A unified approach to building hybrid recommender systems”. Proceedings of the third ACM conference on Recommender Systems, pp. 117-124, 2009.

[18] S Wei, J Xiao, N Ye(2013). “Collaborative Filtering Algotithm Based on Co-Clustering smoothing” Journal of Computer Research and Development

[19] Michael J. Pazzani and Daniel Billsus, Content-Based Recommendation Systems, 2007

[20] M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system-a case study. In ACM WebKDD 2000 Web Mining for E-commerce Workshop, 2000.

[21] Susan T. Dumais (2005). "Latent Semantic Analysis". Annual Review of Information Science and Technology 38: 188.

[22] LI Gai,LI Lei. Collaborative filtering algorithm based on matrix decomposition[J]. CEA, 2011, 47(30): 4-7.

[23] MovieLens, http://grouplens.org/datasets/movielens