Research on Mahout-based personalised recommendation application for libraries

Yanpeng Tu

NUIST Reading Academy, Nanjing University of Information Engineering, Nanjing, China

ag803835@student.reading.ac.uk

Abstract. Personalised recommendation systems are widely used in various fields because they can provide personalised recommendation services to users based on their characteristics or historical behaviour data. This paper addresses the problem of low utilization of library resources due to the lack of personalized service capability in libraries, and implements personalized recommendation service in libraries through the collaborative filtering recommendation module provided by Mahout framework, and the test results also verify the initial personalized recommendation effect.

Keywords: Collaborative filtering; Similarity; Recommender systems; Hadoop; Mahout.

1. Introduction

Today, with the rapid development of information technology and the Internet, people are no longer in the old days of information scarcity. The new technologies represented by " big data, artificial intelligence, cloud computing, Internet of things technology" have brought convenience to people's lives, while also generating a huge amount of data. However, due to the multiple contents and complex structure of these data, it is difficult for people to discover or find the information they care about and need in such a huge amount of data every day, because the value density of these information is too low for each individual, making the utilisation of information too low.

There are two broad solutions to this situation, one of which is the more traditional search engine and the other is a personalised recommendation system. Traditional search engines require users to have a clear need, such as what they want or what they want to know, in order to search by keywords and terms, and once users are unable to express their needs or do not have clear and effective search content, the search system plays a very small role. In this situation of big data, multiple scenarios and unspecified user information needs, personalised recommendation systems are needed. Personalised recommendation systems are used to model the user's behavioural preferences through different personalised recommendation algorithms, to predict the items or information that the user is likely to be interested in and recommend them to the user, i.e. by analysing the user's characteristics or historical behavioural data, discovering the user's potential needs and then actively recommending the information that the user may need. This personalised recommendation technology is now very mature, and basically the user experience is relatively good. Many APPs such as B-site and Jitterbug, which are popular among young people, Jingdong and Taobao on e-commerce platforms, and various video software such as Youku and AiQiyi have all implemented personalised recommendation systems. These personalised recommendations ensure that users can see the content they are interested in when using these software, thus improving their experience and satisfaction.

2. Introduction to the referral system

There are many different types of personalised recommendation systems, depending on the personalised recommendation algorithm. Here we introduce two main types, one based on collaborative filtering and the other on content-based recommendation algorithms. The most important collaborative filtering algorithms include user-based and item-based collaborative filtering algorithms.

User-based collaborative filtering algorithms make recommendations for users based on the interests of similar users. The algorithm assumes that if the behaviour of two users is similar, their interests are also similar, and the core idea is to recommend items to the target user that other users like with similar interests.

The item-based collaborative filtering algorithm starts with similar items and makes recommendations for the user. By calculating the similarity between two items, the similar items are recommended to the user. Instead of calculating the similarity between items based on the attributes of the item content, the algorithm calculates the similarity of the user by analysing the user's behavioural history. The algorithm considers item A and item B to be similar on the basis that users who like item A also like item B.

The flowchart of the user-based collaborative filtering recommendation algorithm is shown in the figure

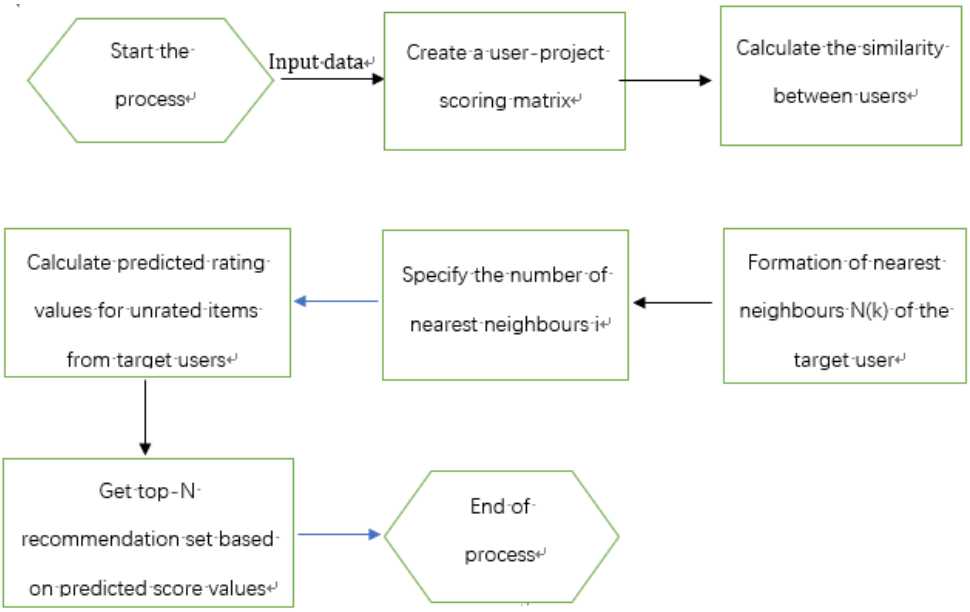

Fig. 1 User-based collaborative filtering process

Specifically, for the personalised recommendation process for books, the user-based collaborative filtering recommendation can be divided into the following steps.

(1) In the actual development, as the borrowing records often lack the user's rating data, the reader's borrowing time or the cumulative browsing time of electronic resources can be converted into the user's rating of the book or electronic resource through certain rules, so that the reader's preference for the book or resource can be reflected in a quantitative way, for example, the rating range can be set to 0-5， the more interested the reader is in a particular book or resource, the greater the score will be. Of course we must take into account differences in readers' reading habits and reading speed, so the conversion function is a way to introduce the average length of time spent reading or browsing to calculate readers' scores. The rating conversion function formula is.

$$R_{ui} = \begin{cases} 5 & t_{ui} \geq 2\overline{t_u} \\ \dfrac{t_{ui}}{2\overline{t_u}} \times 5 & t_{ui} < 2\overline{t_u} \end{cases}$$

where Rui is reader u's rating of book or resource i, tui is reader u's borrowing or browsing time for book or resource i, and $\overline{t_u}$ is reader u's average reading time.

(2) Calculating the similarity between users, user similarity calculation is one of the key steps of the recommendation algorithm. The purpose of the similarity calculation between users is to find the k users that are most similar to the target user, and there are many ways to calculate the similarity, including Cosine similarity, Jaccard similarity, Pearson correlation coefficient, etc. For the similarity calculation of the reader-book rating matrix the Pearson correlation coefficient can be used, and the Pearson similarity can be calculated using the following formula.

$$Sim_{pcc}(u,v) = \frac{\sum_{k\in I_u\cap I_v}(ru_k - \overline{r_u})(rv_k - \overline{r_v})}{\sqrt{\sum_{k\in I_u\cap I_v}(ru_k - \bar{r}_u)^2}\sqrt{\sum_{k\in I_u\cap I_v}(rv_k - \bar{r}_v)^2}}$$

By obtaining the Pearson correlation coefficient between different users through the above equation, we can obtain the similarity matrix W for different users, where w__ {u,v} is the similarity between two different u and v.

$$W = \begin{bmatrix} w\_\{11\} & w\_\{12\} & \cdots & w\_\{1n\} \\ w\_\{21\} & w\_\{22\} & \cdots & w\_\{2n\} \\ & \cdots & \cdots & \\ w\_\{m1\} & w\_\{m2\} & \cdots & w\_\{mn\} \end{bmatrix}$$

(3)From the similarity matrix W we can easily analyse the degree of similarity between different users, sort according to the similarity value, find the k nearest neighbours that are closest to the target user's interests, get the books or resources that these most similar nearest neighbours like and that the target user has not borrowed or browsed, and calculate the target user's predicted rating of these book and goods resources, the predicted rating can be calculated using the following formula

$$r_{ui} = \bar{r}_u + \frac{\sum_{v\in S(u,k)\cap U_i} w_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v\in S(u,k)\cap U_i} w_{uv}}$$

(4)The top N items will be recommended to the target user based on the ranking of the prediction scores.

3. Mahout Recommendation Framework

For the specific implementation of a personalised recommendation system for libraries, we have used the recommendation framework from Apache Mahout. Taste is an efficient implementation of the collaborative filtering algorithm provided by Apache Mahout, a Java-based implementation of an extensible, efficient recommendation engine. Taste is designed to meet the performance, flexibility and scalability requirements of a recommendation engine, and the Taste components are designed to ensure excellent performance in real-world scenarios.

The component structure of Taste is shown in the diagram below：

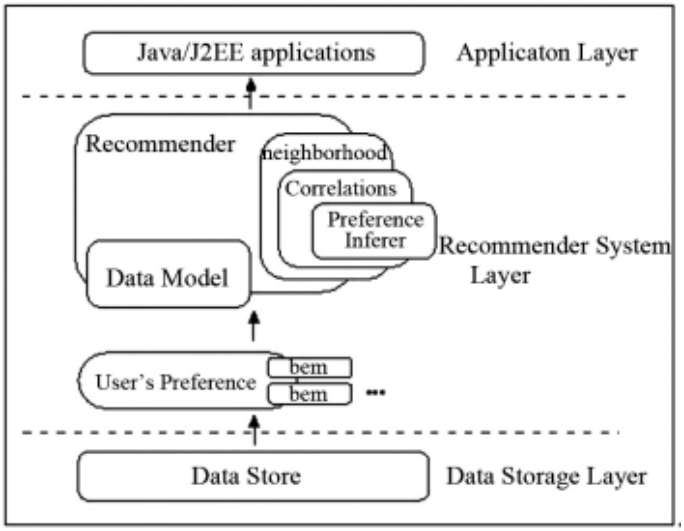

Fig. 2 Taste component structure

The Taste component consists of four main parts：

(1)DataModel: DataModel is an abstract interface for user preference information, its concrete implementation supports extracting user preference information from any kind of data source, Taste provides JDBCDataModel and FileDataModel by default, which support reading user preference information from database and file respectively. In this article, we use FileDataModel to read user ratings data.

(2)ItemSimilarity and UserSimilarity: This section is used to calculate ItemSimilarity and UserSimilarity, and the Taste component contains various methods for calculating similarity, including cosine similarity, Pearson's correlation coefficient and other similarities.

(3)UserNeighborhood: In the user-based recommendation method, recommendations are generated based on finding "neighbourhood users" with similar preferences to the current user, and this component is used to define the "neighbourhood users" adjacent to the target user ". Therefore, this component is only used in user-based recommendation algorithms.

(4)Recommender: Recommender is the abstract interface of the recommendation engine, it is the core component of Taste, which can be used to generate a recommendation list of items for a given user.

Core code implementation：

We initially implemented personalised recommendations for the library using the Taste component provided by Apache Mahout, the core code of which is shown below：

```
public static void main(String[] args) throws TasteException, IOException {
        String file = "datafile/rating.csv";
        DataModel dataModel = RecommendFactory.buildDataModel(file);
        RecommenderBuilder rb1 = BookEvaluator.userEuclidean(dataModel);
        RecommenderBuilder rb2 = BookEvaluator.itemEuclidean(dataModel);
        LongPrimitiveIterator iter = dataModel.getUserIDs();
        while (iter.hasNext()) {
            long uid = iter.nextLong();
            System.out.print("userEuclidean          =>");
            result(uid, rb1, dataModel);
            System.out.print("itemEuclidean          =>");
            result(uid, rb2, dataModel);
        }
    }
public static void result(long uid, RecommenderBuilder recommenderBuilder, DataModel dataModel) throws TasteException {
        List list = recommenderBuilder.buildRecommender(dataModel).recommend(uid, RECOMMENDER_NUM);
        RecommendFactory.showItems(uid, list, false);
}
```

4. Validation of results.

In order to implement personalised book recommendations, we built the corresponding development and testing environment, with the main hardware and software configurations shown in the table below.

Table 1. Software and hardware development environment

Serial number	Projects	Parameter or version number
1	CPU main frequency	2.6GHz
2	Memory	16G
3	Disk space	1T
4	Operating systems	Win10 64bit
5	Java	1.6.0
6	Maven	3.6.1
7	Mahout	0.8
8	Hadoop	1.1.2

The book borrowing records of 32567 students of different majors and grades from a university library from 2018-2021 were used as the dataset for this study. The dataset was divided into two parts,

with 80% used as the training dataset and the remaining 20% used as the testing dataset. These book borrowing records were used in this study to generate recommended books for the students.

Fig. 3 Library personalised recommendation results

The recommendation system was finally deployed on a university library ILS server, running on Solaris 10, Java and PHP5, and served as an OPAC embed. Scheduling is performed by shell scripts. Recommendation generation takes about 20 minutes. Recommendations are generated every half month or so, and each time you can see that the recommendations change according to the circulation history.

From the six-month trial run, it can be seen that the recommendation system has received a lot of readers' attention.

Table 2. Recommended statistical results

Statistical items	Numerical values
Log in readers	14245
View Recommended Readers	1238
View the percentage of recommended readers	8.7%
Number of recommended readers on loan	89
Percentage of recommended readers on loan	7.2%
Highest value of borrowing referrals (copies)	8

According to the statistical analysis of about half a year's operation, about 8.7% of all registered readers have viewed the recommendation information of the same recommendation, about 7.2% of readers who have viewed the recommendation information have borrowed the books recommended by the system, and the most borrowed books through the recommendation information has reached 8 books. From the readers' feedback, the current personalised recommendations have achieved initial results, but there is still a need to optimise and adjust them according to the different attributes and interests of the readers.

5. Summary

While the new generation of information technology, represented by the Big Data, has brought convenience to people's lives, it has also caused information overload. The Taste component of Apache Mahout provides several modules, including collaborative filtering recommendation algorithms, and we used the API interface provided by the component to realise personalised recommendation of library resources. The test results show that the recommendations are effective and reliable.

References

[1] OU Wei-hong, YANG Yong-qin. Research on the Book Recommendation System Based on Mahout under the Big Data Platform [J], (Guangzhou University of Science and Technology, Guangzhou 510550, Guangdong).

[2] MA Hongjian, ZHANG Yunfan, CHEN Jin, BU Hongchao Design and implementation of a book recommendation system for colleges and universities [J], Information Technology and Informatization. 2021, (12): 49-51.

[3] Wu Jieming, Li Wenxi, Research and design of a Mahout-based book recommendation engine [J], Industrial technology innovation. 2015, 2(03): 342-348.

[4] Li Hairong, Fang Zhongchun. Design and implementation of a recommendation system based on Mahout and collaborative filtering algorithms [J], Journal of Inner Mongolia University of Science and Technology. 2018, 37(03):260-263.

[5] Zhu Lijun. Design and implementation of a personalized book recommendation system for university libraries based on mahout [D], Nanchang University, 2018.

[6] Zhang Yue. Design and implementation of a Mahout-based book recommendation system [D], Shanxi University, 2017.

[7] Yan Liyang. Design and implementation of Mahout-based personalized book recommendation system [D], Northwest University for Nationalities, 2019.

[8] Chang Jiang. Research and implementation of an Apache Mahout-based recommendation algorithm [D]. University of Electronic Science and Technology, 2013.