Design and Implementation of the Cross-Harmonic Recommender System Based on Spark

Huang Jie[1(✉)], Liu ChangSheng[1,2], and Liu ChengLi[3]

[1] Department of Aviation Electronic Equipment Maintenance, Airforce Aviation Repair Institute of Technology, Changsha 410014, Hunan, China
huangjie918@126.com
[2] Hunan Key Laboratory of Intelligent Information Perception and Processing Technology, Zhuzhou 412007, Hunan, China
[3] School of Engineering, Computer and Aviation, University of León, 24071 León, Spain

Abstract. With the rapid development of information technology, information overload has become an important challenge of Internet. In order to alleviate the growing contradiction between users and massive data, the researchers proposed the concept of the cross-harmonic recommender system. By analyzing characteristic of datasets, recommendation algorithms and method for weight calculation, we introduced a fast and general engine for large-scale data processing and implemented the cross-harmonic recommender system based on Spark, aiming at improving accuracy, diversity and efficiency of the recommender system.

Keywords: Spark · Hybrid recommendation · Recommendation algorithm

1 Introduction

With the rapid development of information technology, the number of network users and data has increased sharply. It also brings the contradiction between the Internet users and the datas: How can users find useful information in the mass of data, on the other hand, how the Internet can meet the needs of personalized users.

The recommender system can alleviate the contradiction between some data and users [1]. However, the results of a single recommendation algorithm are greatly influenced by the sparsity of the preferred data. But it is still not effective in bridging the gap between them. The emerging hybrid recommendation system combines two or more recommendation algorithms to achieve better recommendation effect [2]. Therefore, it is urgent to design a high precision, high efficiency, diversity and extensibility cross-harmonic recommendation system.

Fund Source: Hunan education department scientific research project (No. 17C0009). Huang Jie, Associate professor/Master, main research fields: Information education in higher vocational, Computer applications (Cloud and Big data), etc. Liu Changsheng, professor/Doctor, main research fields: Computer Application, etc.

G. Gui and L. Yun (Eds.): ADHIP 2019, LNICST 302, pp. 461–474, 2019.
https://doi.org/10.1007/978-3-030-36405-2_47

2 The Technology of the Hybrid Recommendation System

2.1 The Concept of the Recommender System

There are two ways for users to get information, the active and passive ways. The typical active mode is the search engine classification directory; A typical model for a passive approach is the recommendation system [3].

Figure 1 shows that the basic principles of the recommender system, it mainly includes: data modules, algorithm modules and recommendation modules.

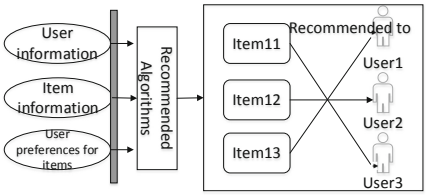

Fig. 1. Schematic diagram of the recommender system

The data modules are the basis of the recommender system. It can store users' basic information, such as items' preferences and other information; The algorithm modules use the recommendation algorithm to predict the preferences of unknown things; And the recommended modules recommend items for users with the recommended algorithm.

Hybrid recommendation algorithm is a new concept to improve the precision of various recommendation algorithms in order to meet various application requirements. The below Fig. 2 shows that the framework of the entire recommender system. Hybrid recommendation systems can be classified from the perspective of data source, algorithms and technologies:

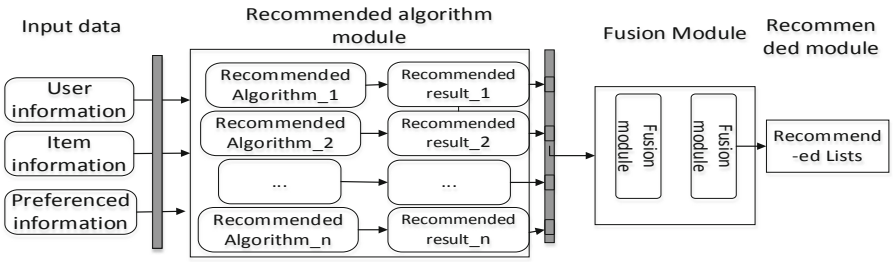

Fig. 2. Recommended system block diagram

(1) Mixed recommendation framework with multiple segments: it consists of three modules: offline modules, approximately online modules and online ones. Each module can meet the different needs of different users for response time, and the users select the recommendation result from the corresponding module as required.
(2) Hierarchical mixed recommendation system: The recommendation algorithm is divided by different precision, selecting high precision algorithms adapt to different application scenarios to select high precision algorithms. This recommender system uses a more sophisticated hybrid technology combining several indicators.
(3) Weighted mixed recommendation system: The recommendation algorithm is used to predict the score, the weighted value of each algorithm's prediction score is calculated by using the weight of the algorithm.
(4) Cross-harmonic recommender system: The system is to mix the results of various recommendation algorithms in proportion to generate mixed recommendation results, which have both diversity and better interpretability.

2.2 Recommended Algorithm

There are three recommended algorithms: content-based recommendation algorithm, collaborative filtering recommendation algorithm and model-based recommendation algorithm.

(1) Collaborative Filtering, CF: This algorithm was originally used to process message filtering in mail [4], and now it is widely used in e-commerce system.
Collaborative filtering algorithm is mainly based on the idea of object clustering, that is, object - like clustering. For example, if u and v have the same interest in goods, they think that u and v are similar users. The algorithm execution process is shown in Fig. 3 below.

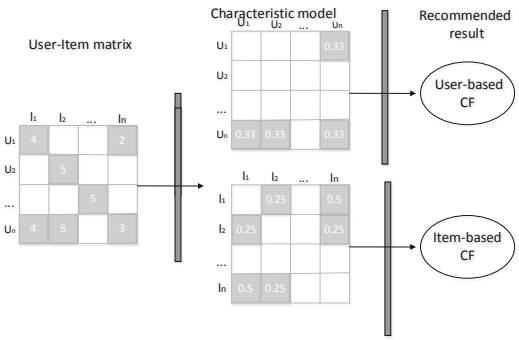

Fig. 3. Collaborative filtering algorithm process

(2) Content-based recommendation algorithm, CB, it is a commonly used algorithm in recommendation system. It mainly uses the similarity of the configuration file to recommend the items to the user [5].
(3) Model-based recommendation algorithm, MB, this algorithm trains the user implicit feature model and the object implicit feature model by using the scoring matrix, and predict unknown data with the two models.

2.3 Spark Technology

Spark is an ecosystem that provides solutions such as flow operation, iteration operation and graph calculation, as the Fig. 4 shown below.

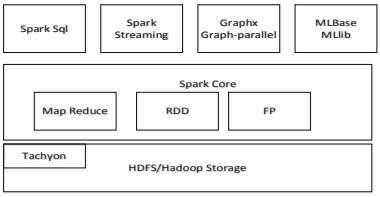

Fig. 4. Spark ecosystem diagram

Spark SQL: It conduct the big data query analysis [6]. By the SQL expression processing data query on Spark. To improve the query efficiency user-defined functions can also be called to combine query results with analysis results.

Sparking Streaming: It is a flow processing system, also can operate real-time data on mapping, reducing and joining. And the result are saved to the external file system [7].

GraphX: It is the graph operation API on Spark, it provides the concept and operation of elastic distributed attribute graph, and implement graph operation by the framework [8].

MLBase/Mllib: This component provides common machine learning algorithms and utilities, includes classification, regression, clustering and bottom optimization.

Tachyon: It is a distributed system with high efficiency, high fault tolerance and high reliability.

Spark: It is the core computing framework of the ecosystem, which can run either alone or on a cluster [9]. Cluster operation requires the help of the management system to distribute tasks to each node to run.

In short, the Spark system is based on RDD, it is a large data platform for parallel computing of memory computing with Spark framework as the core.

Different products in the system can be used to meet the computing needs of different systems.

3 Cross-Harmonic Recommender System Framework Based on Spark

There are four modules in the cross-harmonic recommender system based on Spark [10]. It includes the storage modules, Algorithm modules, Recommended modules [11] (Fig. 5).

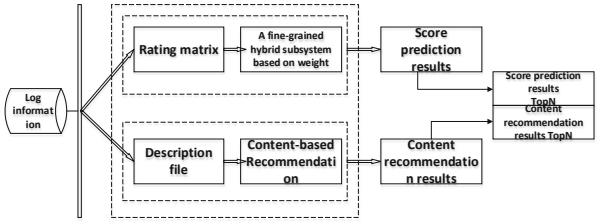

Fig. 5. The frame diagram of cross-harmonic and recommendation system based on Spark

(1) Storage modules: This module collects and cleanses the log data and stores it in distributed storage.
(2) A fine-grained hybrid subsystem module based on weight: The system module builds a user-item scoring matrix [12], it contains user-CF, item-CF and LFM. Using Fine-grained Weight Calculation Method to Calculate the weight vector of each recommendation algorithm [13], and make predictive recommendations for unknown score data.
(3) Algorithm modules: This module is to implement the distributed recommendation algorithm based on content. First, Creating feature properties files for users and items, then making predictions about user preferences. The formatting data file as the input of the recommendation algorithm module, Training feature model of recommendation algorithm is used for processing. Finally, the preference prediction based on the feature model is realized.
(4) Recommended modules: This module is the recommendation system to provide users with the recommendation list information. The results of each recommendation algorithm are combined with cross-harmonic technology to generate a personalized recommendation result for users.

4 Design of the Cross-Harmonic Recommendation System Module

4.1 Design of Data Storage Module

User preference data is the basis of recommendation system. The recommendation algorithm finds valuable information from the preference data to know about the characteristics of each use. Therefore, data storage module is the basic of Spark-based

cross-harmonic recommendation system. The main function of data storage module is to collect log data to extract data features and store data:

(1) Collecting datas: The data storage module mainly collects two types of log data. The first is the common user behavior log, which can be converted into a scoring matrix; The second type is the description file of user items, which contains the description information of user items and is often converted into the feature vector of user items.
(2) Extracting feature data: After the log file is collected, the data storage module cleans the outliers and null values of the missing values, and the data is converted to a specific data structure as the input to other modules.
(3) Data structure: After extracting the data features, the grading data format is converted to userFactor, itemFactor, Event, Rank and Time. The user and the object are identified by the user and object feature vectors respectively. Its data structure is like the below table (Table 1).

Table 1. Grading data structure

userFactor	itemFactor	Event	Rank	Time
<uf1, uf2, …, ufp>	<if1, if2, …, ifr>	4	4	978300760
<uf2, uf4, …, ufq>	<if3, if5, …, ifs>	6	1	978301260
<uf3, uf8, …, ufo>	<if6, if9, …, ift>	5	5	978302240

(4) Data storage: The data storage module stores the collected logs and formatted data in a distributed file system to ensure the reliability of data storage. Common distributed file systems include HDFS and Tachyon.

4.2 Fine-Grained Weight Hybrid Subsystem

The fine-grained weight hybrid subsystem is an important part of the cross-harmonic re-commendation system. The input data of this subsystem is user-item rating data. After the calculation of the weight hybrid subsystem, the mixed prediction score of user-item is output to complete the process of rating prediction.

4.3 Design of the Recommended Modules

Recommendation module is the module that the system interacts with users. The main function of this module is to generate personalized recommendation list for users. The recommendation module of the cross-harmonization recommendation system based on Spark adopts the recommendation method of cross-harmonization to generate personalized recommendation list for users. The cross-harmonic recommendation means that the system performs multiple recommendation algorithms, each of them compute a list of recommendation. Finally, the system selects a part of the result combination from each recommendation list and generates the recommendation list displayed by users.

5 Distributed Recommendation Algorithm

Recommendation algorithm is the basis of the system to predict user preference, and also the core content of the recommendation system, which directly affects the performance of the system. The cross-harmonic recommendation system based on Spark introduces content-based recommendation algorithm to improve the diversity of recommendation results and alleviate the problem of data sparseness.

5.1 Content Recommendation Algorithm Implementation Based on Spark

Content-based recommendation algorithm (CB) is one of the first recommendation algorithms to be applied in practice. This algorithm is good at information retrieval and other fields because of its simple idea and strong interpretability of recommendation results. The basic idea of CB algorithm is to recommend similar items to users according to their favorite items.

The recommendation algorithm flow based on content is shown in the Fig. 6, According to the flow chart, the algorithm consists of three steps:

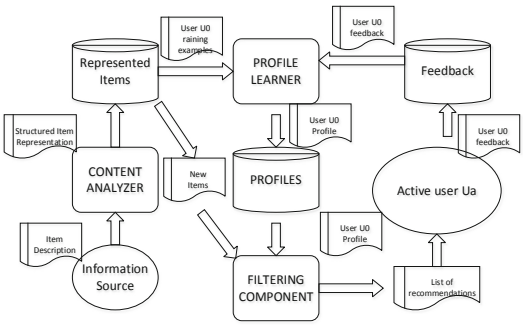

Fig. 6. Content recommendation algorithm processing based on Spark

(1) Character description: The content-based recommendation algorithm first extracts multiple keywords corresponding to users or items according to their description files, and these key words are used to form the feature vector of user items.
(2) Training preference model: This algorithm trains the user's preference model and generates the user preference configuration file according to the product feature vector that users like. Or training the feature model of the item according to the user feature vector of the item you like.
(3) Generating recommendation list: The content-based recommendation algorithm recommends the most relevant items to users by comparing the similarity between the user preference profile and the item feature vector; Or compare the user feature vector with the item feature profile and recommend the item to the relevant use. According to different recommendation objects, CB algorithm is divided into user-CB and item-CB.

5.2 User Recommendation Algorithm Implementation Based on Spark

User-CB algorithm trains the preferences configuration file according to the user feature vector. By comparing the correlation between other user feature vectors and the preferences profile of the item, this algorithm recommends the item to the most relevant multiple users. Therefore, the user-CB is implemented based on spark as the Fig. 7 shows, which contains three steps:

User_CB Algorithm Pseudo-code for distributed implementations based on Spark

```
01:Input :the description file of user and item
02:Output:the similarity between user and item profile
03:
04:user_factor=readMatrix.Split( '' )
05:for user in user_factor:
06:  for item in description file:
07:      if user like item:
08:              map{case(f1,f2,…,fn)=>item_profile}
09:      end if
10:   end for
11:end for
12:
13: calculate similarity between user factor and item
profile
14:
15:    for user in similarity:
16:        for item in similarity[u]:
17:            recommend item to other users
18:        end for
19:    end for
```

Fig. 7. User_CB Algorithm Pseudo-code

(1) Constructing the eigenvector: It is similar to the rating prediction recommendation algorithm to construct the user-item rating matrix, the recommendation algorithm based on user content first constructs the user's feature vector according to the description file, as the data base of system training. Content-based recommendation algorithms usually extract several keywords from a series of description files to constitute the user's eigenvector. So the format of the eigenvector is (userFactor_1, userFactor_2, …, userFactor_n), each userFactor means a characteristic keyword. User_CB this algorithm uses the textFile interface to read the user feature Vector file from the distributed file system and stores each user feature Vector in Vector format. By specifying that the parameter of the persist interface is MEMORY_AND_DISK, the user feature vector is stored in memory and disk to complete the construction of the feature vector;

(2) Training preference model: According to the user feature vector of the favorite item, the user_CB algorithm trains the preference configuration file of the item. First, use the filter operation to match users who like the item and get a collection of users. Then, the user's feature vector is mapped to the item preference vector by the map operation. Finally, the same characteristics are merged by the reduceByKey operation to obtain the preferences profile;

(3) Recommendation list: The user_CB algorithm compares the correlation between user feature vector and item preference configuration, which recommends items to relevant users. The algorithm uses the look-operation to find the matching features. The more matches there are, the more relevant they are.

5.3 Item Recommendation Algorithm Implementation Based on Spark

Item_CB algorithm is similar to the idea of user-CB. According to the product feature vector training user preference configuration file, compare the relationship between the user profile and the product feature vector, and recommend related items for the user. Therefore, the item-CB is implemented based on spark as the Fig. 8 shows, which contains three steps:

Item_CB Algorithm Pseudo-code for distributed implementations based on Spark

```
01:Input :the description file of user and item
02:Output:the similarity between user and item profile
03:
04:item_factor=readMatrix.Split( '' ).Map
05:for item in item_factor:
06:  for user in description file:
07:      if user like item:
08:              map{case(f1,f2,…,fn)=>user_profile}
09:      end if
10:  end for
11:end for
12:
13: calculate similarity between user factor and item profile
```

Fig. 8. Item_CB algorithm Pseudo-code

(1) Constructing the eigenvector;
(2) Training preference model;
(3) Recommendation list.

6 Analysis of Experimental Results

6.1 Content-Based Recommendation Algorithm Efficiency Test

As the cross-harmonic system implements the distributed recommendation algorithm based on content, which ensures the diversity of data types. Therefore, this experiment tests the efficiency of distributed content-based recommendation algorithm. The experimental process is as follows:

(1) Data Sets: The experiment USES movie-tags data set to compose test data of different scales, with data scale ranging from 100K to 1M. Each data set is divided into a training set and a test set.

(2) Recommendation Algorithm: The content - based recommendation algorithm is adopted in this experiment.
(3) Experiment parameter: N = 30, That is, the recommendation results are sorted according to the item correlation, and the first 30 items are selected to generate the recommendation list.
(4) Evaluation index: The execution time of distributed content-based recommendation algorithm

In this experiment, the article profile is constructed by using CB algorithm, the correlation between the user profile and the article profile is calculated, and the recommendation list is generated. The experimental results are shown in the Fig. 9.

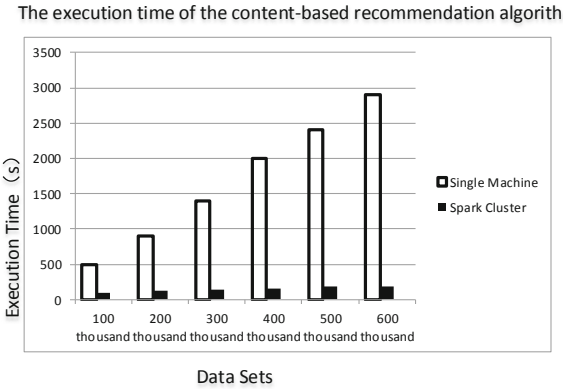

Fig. 9. Execution time of the content-based recommendation algorithm

(1) At the same data scale, the execution time of distributed content-based recommendation algorithm is better than that of single machine algorithm.
(2) The execution time of distributed content-based recommendation algorithm increases slightly with the increase of data scale, but it is always better than single-machine algorithm.
(3) The execution time of distributed content-based recommendation algorithm is less affected by data size. That is, compared with single-machine algorithm, distributed content-based recommendation algorithm is not sensitive to changes in data scale.

The introduction of CB algorithm cross-harmonic recommendation system based on Spark enables the system to handle more diverse data types, such as rating data with label information. And this algorithm has good execution efficiency.

6.2 Efficiency Test of the System

The cross-harmonic system based on Spark include fine-grained weight hybrid subsystems and distributed content-based recommendation algorithms, Therefore, the

execution time experiment process of this experiment test based on Spark cross - harmonic system is as follows:

(1) Data sets: The experiment uses movie-tags data set to compose test data of different scales, with data scale ranging from 100K to 1M. Each data set is divided into a training set and a test set;
(2) Recommendation algorithm: user collaborative filtering, Latent factor model, Content-based Recommendation.
(3) Experiment parameter: N = 30, That is, the recommendation results are sorted according to the item correlation, and the first 30 items are selected to generate the recommendation list.
(4) Evaluation indicator: system execution time.

In this experiment system, the user collaborative filtering cryptography model and the combination of user collaborative filtering and cryptic model and the recommendation algorithm based on content are implemented to test system execution time. The experimental results are shown in Fig. 10.

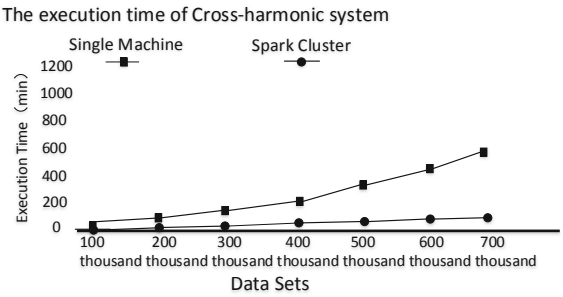

Fig. 10. The execution time of Cross-harmonic system

(1) At the same data scale, the execution time of distributed system is always better than that of single machine system. With the increase of data scale, the execution time of distributed system increases slightly, while the execution time of single machine system increases significantly. Therefore, Distributed systems are not sensitive to changes in data size.
(2) After implementing distributed content-based recommendation algorithm, the system efficiency is stable. The effectiveness of the algorithm is verified.
(3) The largest scale of single-machine system is 900,000 pieces of data, while distributed system can effectively process larger scale data and has stronger practical application value.

Scalability test of the system

In this section, by changing the number of nodes in the cluster, the scalability test scheme of hybrid recommendation system based on Spark. The steps are as follows:

(1) Data sets: MovieLens-100K, MovieLens-1M and BookCrossin. To meet the computing capacity of different Numbers of nodes, 100,000 scoring data were randomly selected from BookCrossing to form a BookCrossing-100K subset.
(2) Recommendation Algorithm: user-CF and item-CF.
(3) Evaluation index: the running time of the system is in minutes (min).

In this experiment, by changing the number of nodes in the cluster, the test system specifies the running time of the system on different scale clusters when the user-cf algorithm and item-cf algorithm are specified. The experimental results are shown in Figs. 11 and 12.

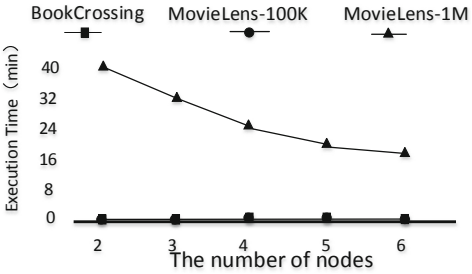

Fig. 11. The scalability of distributed user_CF algorithms

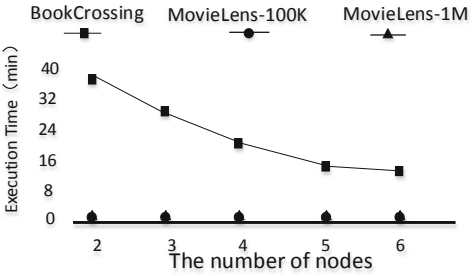

Fig. 12. The scalability of distributed item_CF algorithms

The experimental results show that:

(1) As the number of cluster nodes increases, the training time of both user-CF and item-CF algorithm in the system decreases.
(2) The larger the size of the data set, the faster the training time of the proposed algorithm decreases with the increase of the number of nodes. For example. When training movielens-im data, its training time decreases with the increase of the number of nodes.
(3) When the number of nodes increases to more than 6, the decrease speed of the training time of the proposed algorithm slows down. This is because the

communication cost between nodes will increase with the increase of the number of nodes, so the number of nodes in the cluster is not the more the better, it needs to take into account multiple factors such as the complexity of data size algorithm.

6.3 Spark SQL Performance Test

This article stores the recommendation results in memory as an RDD table, and users query the recommendation results interactively with Spark SQL. Therefore, this section tests the data query performance of Spark SQL and compares it with the performance of MySQL data query. The experimental process is:

(1) Data sets: Recommended list of size 100K, 1M and 10M;
(2) Evaluation indicators: Data query time in milliseconds (ms).

First, the movieLens-10M data set's recommended list of 72,000 users is copied to 0.12 million, 1.2 million and 12 million. Then, the three recommendation lists were randomly divided into six, in each experiment, 5 copies of the user recommendation list with a scale of 0.1, 1, 10 million were selected. Repeat the experiment 5 times and take the average. Spark SQL and MySQL Query the recommended list of 100 users separately to test the time to get the query results. Experimental results are shown in the Fig. 13 below:

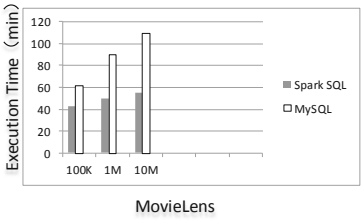

Fig. 13. Data query time of Spark SQL and MySQL

(1) Spark SQL Query the RDD table in memory, and MySQL uses SQL statements to get the results after writing the query results to the database. Therefore, the data query time of Spark SQL is shorter than that of MySQL.
(2) Because the Spark SQL data query engine assigns query tasks to multiple nodes to execute in parallel, Therefore, as the data size increases, MySQL query time increases significantly, while SparkSQL query time increases slowly.

In this section, a series of experiments are conducted to test the accuracy efficiency and scalability of the hybrid recommendation system based on Spark, and the performance bottleneck of the system and the cause of the bottleneck are further analyzed according to the experimental results. The final experimental results show that the hybrid recommendation system based on Spark has good performance in terms of precision efficiency and scalability.

7 Conclusion

Based on Spark, a distributed data processing platform, the hybrid recommendation system with high precision, high efficiency, diversity and extensibility is designed and implemented in this paper, which improves the accuracy and diversity of the recommendation results and reduces the training time of the model.

References

1. Shvachko, K., Kuang, H., Radia, S., et al.: The hadoop distributed file system. In: 2014 IEEE 26th Symposium on IEEE Mass Storage Systems and Technologies (MSST), pp. 1–20 (2014)
2. Zaharia, M., Das, T., Li, H., et al.: Discretized streams: an efficient and fault-tolerant model for stream processing on large cluster. HotCloud **12**, 34–56 (2015)
3. Meng, X., Bradley, J., Yavuz, B., et al.: Mllib: machine learning in apache spark. J. Mach. Learn. Res. **12**(34), 23–34 (2017)
4. Wang, Z., Sun, L., Zhu, W., et al.: Joint social and content recommendation for user-generated videos in online social network. IEEE Trans. Multimed. **15**(3), 698–709 (2017)
5. Koren, Y.: The bellkor solution to the netfix grand prize. Netflix Prize Doc. **34**, 32–45 (2016)
6. Zaharia, M., Chowdbury, M., Franklin, M.J., et al.: Spark: cluster computing with working sets. HotCloud **10**(11), 80–98 (2015)
7. Li, H., Ghodsi, A., Zaharia, M., et al.: Tachyon: reliable, memory speed storage for cluster computing frameworks. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 13–25. ACM (2016)
8. Engle, C., Lupher, A., Xin, R., et al.: Shark: first data analysis using coarse-grained distributed memory. In: Proceeding of the 2012 ACM SIGMOD International Conference on Management of Data (2012)
9. Armbrust, M., Xin, R.S., Lian, C., et al.: Spark SQL: relational data processing in spark. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1383–1394. ACM (2015)
10. Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Trans. Inf. Syst. **22**(1), 143–177 (2014)
11. Yang, R., Hu, W., Qu, Y.: Using semantic technology to improve recommender systems based on slope one. In: Li, J., Qi, G., Zhao, D., Nejdl, W., Zheng, H.T. (eds.) Semantic Web and Web Science, pp. 11–23. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-6880-6_2
12. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Major components of the gravity recommendation system. ACM SIGKDD Explor. Newsl. **9**(2), 80–83 (2014)
13. Nightingale, E.B., Chen, P.M., Flinn, J.: Speculative execution in a distributed file system. ACM (2010)