Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 9050872, 8 pages
https://doi.org/10.1155/2022/9050872

Research Article

Collaborative Filtering-Based Music Recommendation in Spark Architecture

Yizhen Niu

Xinyang Vocational and Technical College, Xinyang, Henan 464000, China

Correspondence should be addressed to Yizhen Niu; nyz@xyvtc.edu.cn

Received 30 March 2022; Accepted 18 April 2022; Published 12 May 2022

Academic Editor: Zaoli Yang

The use of recommendation algorithms to recommend music MOOC resources is a method that is gradually gaining ground in people's lives along with the development of the Internet. The often used ALS collaborative filtering algorithm has an irreplaceable role in personalised recommender systems via the Spark MLlib platform. In the study, it is investigated how Spark can be used to implement efficient music system recommendations. The collaborative filtering algorithm based on the ALS model in the Spark architecture is currently the most widely used technique in recommendation algorithms, allowing for the analysis and optimisation of computational techniques. The project-based collaborative filtering algorithm used in the article enables the recommendation of music by avoiding personal information about the user. More accurate user recommendations are achieved by predicting the user's preferences and focusing on the top ranked and highly preferred music recommendations. The method improves the performance of the recommendation algorithm, which is optimised by Spark shuffle on top of resource optimisation, and its performance improved by 54.8% after optimisation compared to when there is no optimisation.

1. Introduction

With the development of the music market, the number of digital music is growing exponentially and online music service platforms are emerging rapidly, allowing users to listen to music more conveniently through the Internet [1]. It has become increasingly difficult for users to choose their favourite songs from the vast amount of songs available, wasting a lot of time [2]. In order to provide information quickly and accurately, recommendation systems are one of the commonly used methods. Based on the interest characteristics of different users, the recommendation system mines the vast amount of information for resources that users may be interested in or need and makes recommendations.

The music recommender system (MRS) has become a popular field, which analyses users' listening habits and song characteristics to recommend songs that meet their needs and preferences [3]. Through music recommendation, not only can we actively provide suitable music for users to choose but also improve the quality of online music services. Wang Bingxiang explored users' music preferences based on tags and user features to achieve personalised song recommendations [4]. Bogdanov et al. proposed a content-based and collaborative filtering weighted fusion music recommendation algorithm to address the shortcomings of traditional music recommendation algorithms, which can recommend music of interest to users more quickly than traditional recommendation algorithms [5]. Based on the improved Apriori algorithm, Lijuan Yu achieved personalised music recommendation by deep learning of user information [6]. Considering the difficulty of obtaining users' personal information, a study proposed a music recommendation method based on music metadata and collaborative filtering, which can recommend various music items without using users' personal information.

Since the 20th century, with the gradual development of the Internet, people's lives have gradually moved from the era of information scarcity to the era of high-speed information [7]. In this era, both information consumers and information producers have encountered great challenges. As information consumers, it is a very difficult task to find

information of interest to them from a large amount of information. The most crucial thing is that it is not easy for information producers to get their information noticed by a large number of users [8]. Recommender systems are an important tool for solving this paradox [9]. The task of a recommendation system is to connect users and information, on the one hand, helping them to discover information that is valuable to them, and on the other hand, enabling information to be presented to users who are interested in it, thus achieving a win-win situation for both information consumers and information generators. Recommendation algorithms can recommend information or objects (e.g., movies, TV shows, music, books, news, images, and web pages) to users that are likely to be preferred. Among recommendation algorithms, collaborative filtering is one of the most used algorithms today [10]. The original application of "collaborative filtering" was proposed in 1992 by Goldberg, Nicols, Oki, and Terry [11], applied to the problem of information overload at the Paloma Alto Research Centre of Xerox, the original recommendation system tapestry only dealt with internal company mail, and the amount of data is small.

The model-based ALS (alternating least squares) collaborative filtering algorithm provides a good method for dimensionality reduction, and Spark, the latest generation of big data processing engine, is very suitable for handling complex iterative calculations of ALS to gradually replace the Hadoop big data computing platform [12, 13]. This study optimizes the music MOOC resources and shuffle based on the underlying operation mechanism of Spark and the characteristics of the ALS algorithm, and by reasonably allocating the music MOOC resources and merging the files generated by the shuffle process, it can reduce the memory storage pressure and reduce the disk I/O, which reduces the recommendation time, and the result is reduced recommendation time and improved recommendation efficiency.

2. Introduction to the Algorithm

2.1. ALS Recommendation Algorithm. The key problem with the implicit semantic model [14] based recommendation algorithm used here is how to decompose the high-dimensional matrix obtained from the data, essentially the matrix of relationships between numerous users and products reduced in dimensionality. In this study, the recommendation algorithm based on the ALS (alternating least squares) model is optimised. This matrix is often sparse because not every user has rated every item [15]. The core of ALS is the decomposition of the user-product rating matrix, which yields two low-rank matrices. During the decomposition of the matrices, the missing ratings are added, and based on these additional ratings, the user can be recommended a product. The missing ratings are added during the matrix decomposition process [16].

For matrix $R_{m\times n}$, ALS aims to find two low-rank matrices $X(m\times k)$ and matrix $Y(k\times n)$ to approximate $R_{m\times n}$, i.e.,

$$R_{m\times n} \approx X_{m\times k}^{T} Y_{k\times n}, \tag{1}$$

where $R_{m\times n}$ represents the user's tasting matrix for the good, $X(m\times k)$ represents the user's preference matrix for the implied features, and $Y(k\times n)$ represents the matrix of implied features contained in the good, where $k \ll \min(m,n)$; in order to make the product of matrices X and Y approximate R as closely as possible, the minimization squared error loss function is used.

$$\sum_{i=1}^{n}\left(r_{ui} - x_u^{\mathrm{T}} y_i\right)^2$$
$$L(X,Y) = \sum_{i=1}^{n}\left(r_{ui} - x_u^{\mathrm{T}} y_i\right)^2, \tag{2}$$

where r_{ui} denotes the rating of item i by the u^{th} user, x_u denotes the vector of implicit features of user u's preferences, y_i denotes the vector of implicit features of item i, and $x_u^{\mathrm{T}} y_i$ is an approximation of user u's rating of the item, with the following regularization term added to prevent overfitting.

$$L(X,Y) = \sum_{i=1}^{n}\left(r_{ui} - x_u^{\mathrm{T}} y_i\right)^2 + \lambda\left(\left|x_u\right|^2 + \left|y_i\right|^2\right), \tag{3}$$

where x_u and y_i are coupled together and are not well solved, so ALS is invoked by first fixing Y and taking the partial derivative of the loss function $L(X,Y)$ with respect to x_u, i.e.,

$$\frac{\partial L(X,Y)}{\partial x_u} = -2Y_{k\times n} r_u + 2Y_{k\times n} Y_{k\times n}^{\mathrm{T}} x_u + 2\lambda x_u. \tag{4}$$

So, there are

$$\frac{\partial L(X,Y)}{\partial x_u} = 0,$$
$$x_u = \left(Y_{k\times n} Y_{k\times n}^{\mathrm{T}} + \lambda E\right)^{-1} Y_{k\times n} r_u. \tag{5}$$

Fixing X in the same way, the symmetry gives

$$y_i = \left(X_{m\times k} X_{m\times k}^{\mathrm{T}} + \lambda E\right)^{-1} X_{m\times k} r_i. \tag{6}$$

The above two steps are repeated, and the root mean square error (RMSE) is introduced as a conditional parameter for the termination of the iteration.

$$\widetilde{R} = X^{\mathrm{T}} Y,$$
$$\mathrm{RMSE} = \sqrt{\frac{\sum (R - \widetilde{R})^2}{N}}. \tag{7}$$

When the value of the root mean square error (RMSE) varies very little and is less than a preset value, the result is considered to have converged and the iteration is stopped or the number of iterations can be preset and the iteration is stopped when the preset number of iterations is reached [17].

2.2. Project-Based Collaborative Filtering. Collaborative filtering [7] (CF) is a very important algorithm in recommender systems for content that cannot be described simply and adequately in terms of metadata. Collaborative filtering

techniques work by building a rating database (user-item matrix) that contains the user's preferences for items. Recommendations are then made by matching users with relevant interests and preferences by calculating similarities between users based on their historical behavioural data. The recommendation method A recommendation is made by creating a neighbourhood for the user and using the neighbourhood to get items that user *i* has not rated but have been rated by other users in the group. A prediction is usually a value *Rij*, which represents user *i*'s prediction score for item *j*. A recommendation is usually a list of the top *N* items that the user likes best, as shown in Figure 1.

Collaborative filtering techniques can be divided into two categories [18, 19]: user-based CF and item-based CF. User-based collaborative filtering calculates the similarity between users by comparing their ratings of the same item and then calculates the predicted rating of an item by the active user as a weighted average of the ratings of that item by users similar to the active user, where the weight is the similarity of those users to the target item. Collaborative project-based filtering uses similarity between projects rather than similarity between users to calculate predictions. By retrieving all items rated by active users from the user-item matrix, an item similarity model is built to determine how similar the retrieved items are to the target items, and then, the *k* most similar items are selected and their corresponding similarity is determined [20]. Predictions are made by a weighted average of active users' ratings of similar items *k*. Typically, two popular similarity measures are used to calculate the similarity between users and items, namely, relevance-based similarity (Pearson's correlation coefficient) and cosine-based similarity.

2.3. Spark Shuffle Analysis and Optimization. Due to the multiple iterations of the ALS algorithm, the performance of the Spark job is mainly consumed in the shuffle phase, which also contains a large number of disk I/O, serialization, network transmission, and other operations, so the performance of the ALS recommendation algorithm should be improved by optimizing the shuffle process. The MapReduce process requires the same type of data from each node to be brought together at a node for computation [21], and the process of bringing these data distributed in different nodes together according to certain rules is called shuffle.

After running ShuffleMapTask in parallel, each ShuffleMapTask creates 4 BK caches and the corresponding SBF disk files. After ShuffleMapTask [22] finishes executing, ShuffleMapTask does not recreate new output files when running in parallel, but reuses the output files created by the previous ShuffleMapTask, the output file created by ShuffleMapTask, and write the data to ShuffleMapTask' output file [23]. When the ResultTask pulls data, it only pulls a small amount of data, and each output file may contain multiple output files given to itself by ShuffleMapTask, so that the number of output files is reduced by a factor of two. In a real application, suppose there are 100 nodes, one executor per node, and each executor is allocated 2 CPU cores with 1000 ShuffleMapTask and ResultTask, respectively, which will generate 1 million output files and put a lot of pressure on the disk for I/O [24]. After optimizing ConsolidateFiles, each executor executes 10 ShuffleMapTasks, so the number of output files per node is 2000, and the total number of output files for 100 nodes is reduced to 200,000, reducing the disk I/O pressure by a factor of 5.

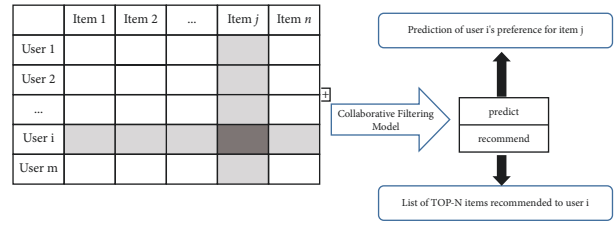

Figure 1: Collaborative filtering algorithm.

3. Methodology of This Paper

The music recommendation method in this study consists of four stages, namely, extraction of music data, normalisation of listening frequency data, similarity measure between music based on the data, and user preference prediction and music recommendation. The music recommendation algorithm uses an item-based collaborative filtering algorithm.

3.1. Music MOOC Resource Extraction. The music MOOC resources used in this article are mainly information contained in audio files, which are used to identify and present audio content, in order to extract music metadata to measure the similarity between music. Music metadata includes mainly identifying information about the music (track number, song title, release date, and genre) and audio analysis information (tempo and loudness).

3.2. Regularisation of the Frequency of Listening. Users listen to each of their favourite songs at different frequencies, so it is necessary to normalise the frequency data of users listening to music by transforming the raw data in a certain proportion so that it falls into a small specific interval [25]. In this study, the normalised listening frequency data are mapped to [1, 10] using a minimum-maximum normalisation method. The normalised preference $P_{u,i}$ of user u for song i is calculated as follows.

$$P_{u,i} = \frac{L_{u,i} - \min(L_u)}{\max(L_u) - \min(L_u)} (S_{\max} - S_{\min}) + S_{\min}, \quad (8)$$

where $L_{u,i}$ is the frequency of listening to song i by user u, $\min(L_u)$ is the minimum frequency of listening by user u, $\max(L_u)$ is the maximum frequency of listening by user u, $S_{\max}$ is the maximum value of the data mapping range, and $S_{\min}$ is the minimum value of the data mapping range.

3.3. Using Data to Measure Similarity between Music. First, the top M songs were selected in the order of the most frequently listened songs, and the top N users who had listened to more than 25 songs were also selected; second, an

$M \times N$ song-user matrix was created, and the matrix was populated with the minimum-maximum normalised user music preference data; third, the target user was identified, and the music that the user had listened to was distinguished from the music that the user had not listened to, and the preference values of the top N users for each song were vectorized, and the cosine similarity was calculated using equation (8). Finally, the extracted song metadata are also vectorized and the cosine similarity is calculated, and the two similarity values are added together as the final similarity value [21].

3.4. Preference Prediction and Music Recommendation. Using the predicted preference values, the top N songs can be recommended to the user from the final generated list of song recommendations. The predicted preference values are calculated as follows.

$$P_{u,i} = \frac{\sum_{\text{all similar items},N}\left(s_{i,N} * R_{u,N}\right)}{\sum_{\text{all similar items},N}\left(\left|s_{i,N}\right|\right)}, \tag{9}$$

where $P_{u,i}$ is the predicted preference value for song i that target user u has not listened to, $s_{i,N}$ is the similarity to song N among the top n songs judged to be most similar to music i among the songs that user u has listened to, and $R_{u,N}$ is the preference value of the songs N that user u has listened to. To obtain the predicted preference value $P_{u,i}$, the top n songs with the highest similarity are calculated. The final number of songs recommended in order of high predicted preference is changed from 5 to 30, and a list of song recommendations is generated.

4. Experimental Analysis

To verify the feasibility of the music recommendation system, the public music dataset, Million Song Dataset [26] was used for the experimental data. This dataset contains the features, metadata, music IDs, user IDs, and listening frequencies of millions of songs and is provided by The Echo Nest [27]. The music metadata are mainly taken from the song summary data and genre attribute data in the Million Song Dataset. The summary data consist of basic information such as song title, producer and release date, and audio analysis information; the genre attribute data consist of identifiers and music genres. In the experiments, only music data for which both summary data and genre attribute data were present were extracted, and the extracted music listening frequency data were normalised using minimum-maximum normalisation to map the data to [1, 10].

To make the recommendations, the top 100 songs were selected in order of the most frequently listened songs and 10,272 data from the top 500 users who had listened to more than 25 songs were filtered. In the experiment, 70% of the data was used as the training dataset and 30% of the data was used as the test dataset. The training set was used to calculate the similarity and expected preference between music and to generate a list of recommended music for the target users. The generated expected preferences and recommended music lists were compared with the test set to verify the performance of the recommendation method. It was found that the performance of the recommendation system was optimal when the number of similar songs was specified as 10 when predicting music preferences. In the experiments, the number of similar songs was fixed at 10 and the top N of the music with the highest predicted preference was changed from 5 to 30 (change step of 5) to compare the performance of the recommendation method proposed in this paper with that of the traditional collaborative item filtering recommendation method.

Table 1: Precision, recall, and $F1$ value for top N variation of the number of recommended songs.

Top N	Traditional method			The proposed method		
	Precision	Recall	$F1$	Precision	Recall	$F1$
5	0.09	0.06	0.07	0.12	0.09	0.10
10	0.09	0.13	0.11	0.11	0.16	0.13
15	0.09	0.20	0.13	0.10	0.22	0.14
20	0.09	0.27	0.14	0.10	0.28	0.15
25	0.09	0.33	0.14	0.10	0.34	0.15
30	0.09	0.40	0.15	0.10	0.41	0.15

The experimental results are measured using precision, recall and $F1$. Precision indicates the proportion of items recommended by the recommendation system that are actually preferred by the user, while recall indicates the proportion of items preferred by the user that are actually recommended. The precision and recall can be measured by the following equation:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{11}$$

TP + FP indicates the total number of music items recommended by the system, FN indicates the number of items preferred by the user but not recommended by the system, TP + FN indicates the total number of music items listened to by the user, and TP + FP indicates the total number of music listened to by the user.

The formula for calculating the $F1$ value is as follows:

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{12}$$

The results of the experiments are given in Table 1.

A visual comparison between the traditional method and the method proposed in this study in three areas is shown in Figure 2.

Table 1 provides the precision, recall, and $F1$ values according to the variation of top N in the number of recommended songs. When the top N is close to 30, the difference in performance between the traditional recommendation method and the proposed recommendation method is small, but when top N is 5–15, the proposed method showed excellent performance.

The most popular 50 music tracks were selected from the dataset, and 10 of these 50 tracks were randomly selected and

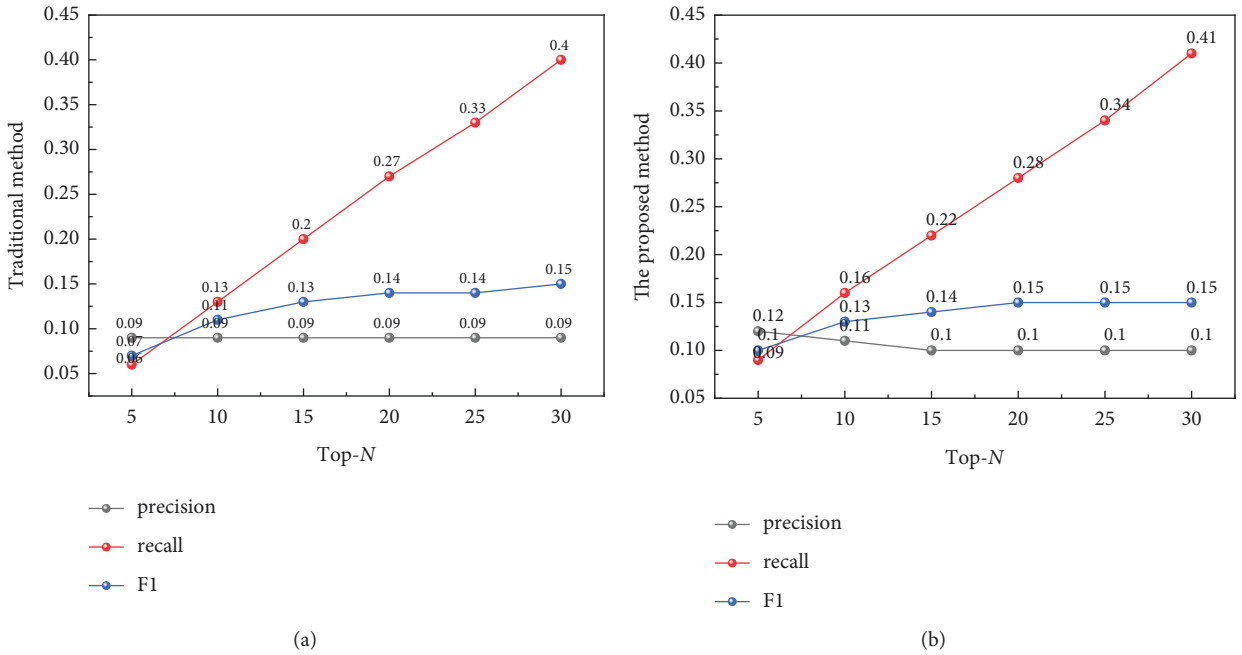

Figure 2: Comparison between the (a) traditional method and the (b) proposed method.

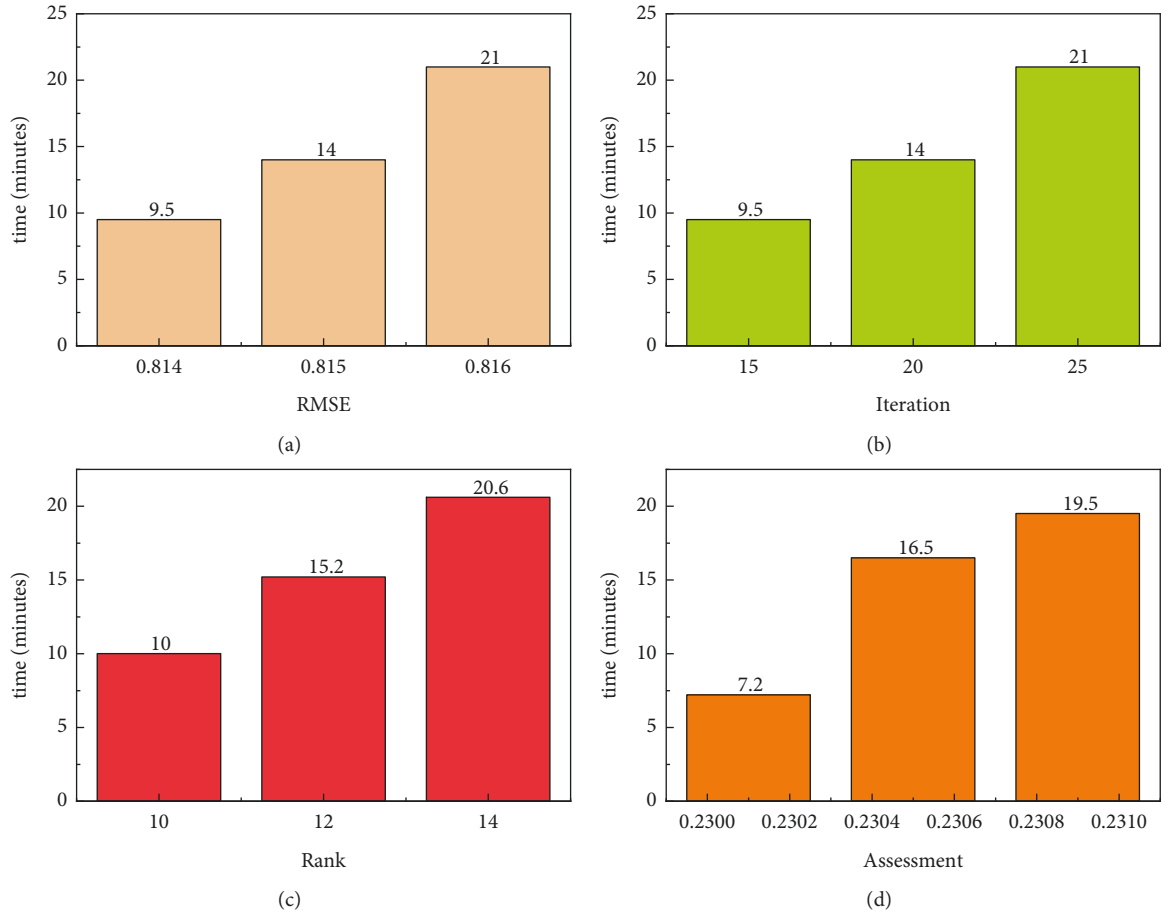

Figure 3: Comparison of four metrics on runtime. (a) RMSE. (b) Iteration. (c) Rank. (d) Assessment.

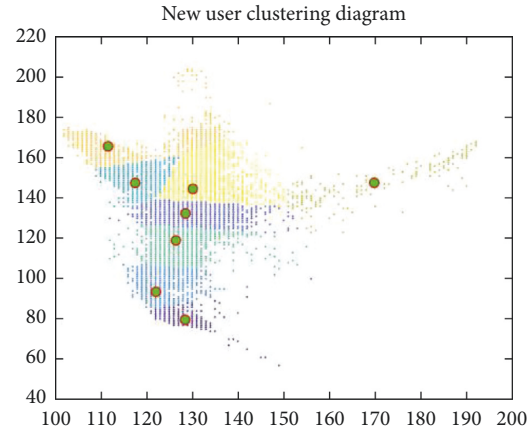

Figure 4: Schematic diagram of K-means clustering results for new users.

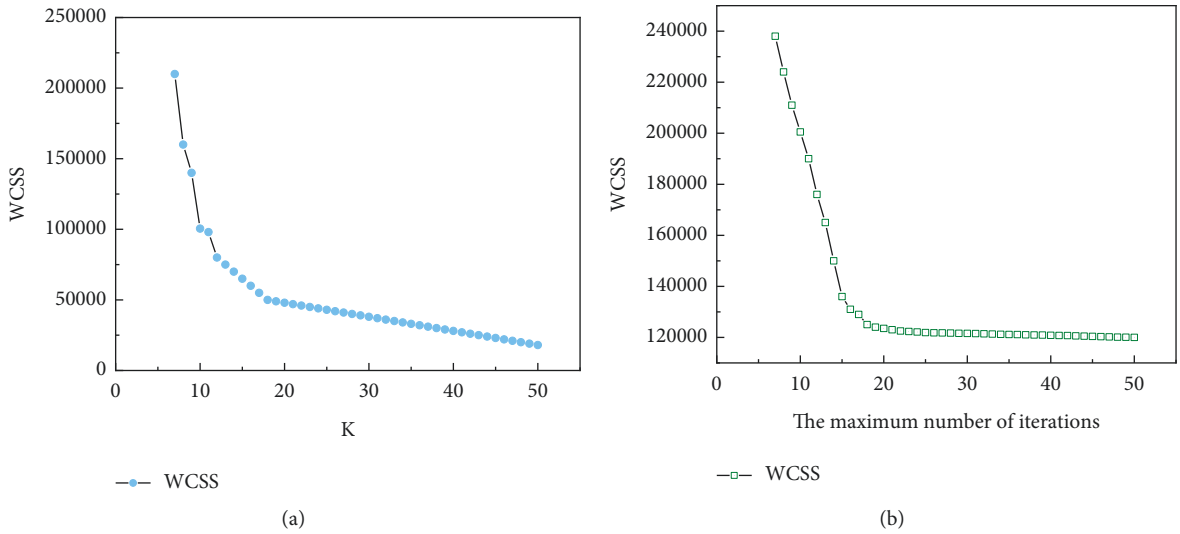

Figure 5: Effect of K value (a) and maximum number of iterations (b) on WCSS.

scored by new users, and a feature vector of users was generated based on their scores. Three sets of experiments were conducted to observe the relationship between the four metrics (RMSE, root mean square error; iteration, number of iterations; rank, matrix dimension; assessment, recommendation baseline) and the running time, namely, no optimisation, resource optimisation, and shuffle optimisation.

Figure 3(a) shows the change in RMSE versus runtime for the three sets of experiments. Figure 3(b) shows the optimal number of iterations versus runtime for the three sets of experiments. Figure 3(c) shows the optimal matrix dimension versus runtime for the three sets of experiments, and in Figure 3(d), assessment is the percentage above the recommended benchmark evaluation line. Assessment remains the same, the recommendation time is reduced from 21 minutes to 14 minutes by improving the resource allocation, improving efficiency by 33.3%, and the total efficiency is reduced from 14 minutes to 9.5 minutes by optimizing the file merge on the map side of shuffle, improving efficiency by 54.8%.

The ALS algorithm used in the previous study cannot solve the problem of cold start of users. The K-means algorithm is a common algorithm in the clustering algorithm, which needs to solve two problems, one is the selection of the number of clusters K and the other is the initial clustering centre. A diagram of the clustering results is shown in Figure 4.

In this study, the clustering algorithm mainly deals with the cold start problem, only responds to the recommendation problem of new users, and then mixes with ALS after the algorithm model is trained to achieve the clustered ALS recommendation algorithm. WCSS becomes the intracluster variance sum, which is also the objective function of K-means clustering. The article selects the optimal parameters by fixing the maximum number of iterations and the value of K. The plotted WCSS trend is shown in Figure 5.

Table 2: Music recommendation algorithm evaluation results.

Recommended quantity, N	5	10	20	50
Accuracy	3.41%	2.97%	2.51%	1.75%
Recall	9.83%	7.35%	7.07%	6.56%

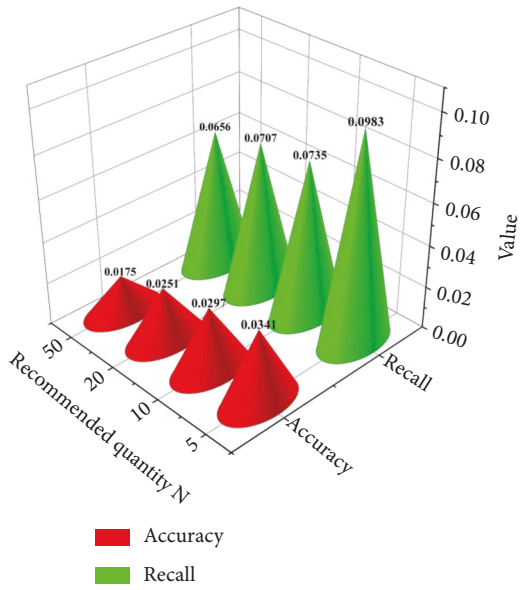

Figure 6: Plot of accuracy vs. recall for ten experiments with different numbers of recalls.

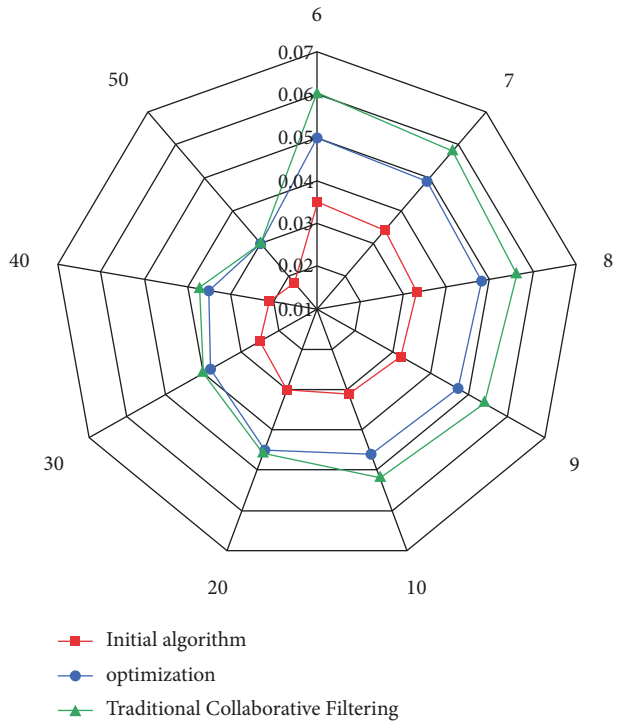

Figure 7: Comparison of the performance of the optimised initial algorithm.

After clustering all new users, experiments were next conducted on the classified population. The experiments were conducted 10 times, the training and test sets were randomly assigned, and the average of the accuracy and recall calculated for each experiment was finally used as the final evaluation result. The specific evaluation results are given in Table 2.

A comparison of accuracy and recall for different numbers of recalls is shown in Figure 6.

As can be seen from the data in the table, the accuracy and recall rates have improved dramatically compared to the initial algorithm, and the ability to mine music preferences has improved exponentially; a more detailed accuracy comparison is shown in Figure 7.

As shown in Figure 7, the performance of the optimised music resource recommendation algorithm shows a significant improvement in accuracy compared to the initial algorithm.

5. Conclusion

In order to reduce the difficulty for users to select their favourite music, this study proposes a method to recommend music based on listening frequency data and music data based on Spark architecture. The proposed method has better recommendation performance compared to traditional item-based collaborative filtering recommendations. The proposed shuffle process of merging the file outputs on the map side is an important but easily overlooked problem in real production. In order to be applied in practical music recommendation, it is necessary to be able to quickly calculate the similarity between music and generate a recommendation list. Properly sampled data, then, are expected to contribute to improving the performance of practical music recommendations. Since no optimal weight values were found between the type attributes of the music data used in the article, it is difficult to make further improvements in the efficiency of the recommendations. The next judgement is made regarding the weight relationships of the individual attributes, and then, the recommendations made will be more in line with the needs of the user.

Data Availability

The datasets used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] J. Lv, B. Wu, C. Liu, and X. Gut, "A parallel framework for face classification and search from massive videos based on spark," in *Proceedings of the 2018 IEEE Fourth International Conference on Multimedia Big Data*, pp. 1–7, BigMM), Xi'an, China, September 2018.

[2] M. A. Rahman, J. Hossen, and C. Venkataseshaiah, "A self-tuning approach using machine learning to improve performance of spark in big data processing," in *Proceedings of the 2018 7th International Conference on Computer and Communication Engineering (ICCCE)*, pp. 274–279, Kuala Lumpur, Malaysia, September 2018.

[3] A. Sheshasaayee and J. V. N. Lakshmi, "An insight into tree based machine learning techniques for big data analytics using Apache Spark," in *Proceedings of the 2017 International Conference on Intelligent Computing*, pp. 1740–1743, Instrumentation and Control Technologies (ICICICT), Kerala, India, July 2017.

[4] S. Srivastava, A. Nigam, and R. Kumari, "Towards efficient and scalable big data analytics: mapreduce vs. RDD's," in *Proceedings of the 2017 International Conference on Information Technology (ICIT)*, pp. 272–275, Bhubaneswar, India, December 2017.

[5] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambó, and E. P. Gómez, "Semantic audio content-based music recommendation and visualization based on user preference examples," *Information Processing & Management*, vol. 49, no. 1, pp. 13–33, 2013.

[6] W. Ma, F. Ay, C. Lee, and G. Gulsoy, "Fine-scale chromatin interaction maps reveal the cis-regulatory landscape of human lincRNA genes," *Nature Methods*, vol. 12, no. 1, pp. 71–78, 2014.

[7] S. Im, H. Kim, J. Maeng, J. Yu, and Y. Cha, "On the mean square displacement of a random walk on a graph," *European Journal of Combinatorics*, vol. 51, pp. 227–235, 2016.

[8] Li Bo, "Study on massive e-govenment data cloud storage scheme based on Hadoop," in *Proceedings of the IEEE International Conference on Software Engineering & Service Science*, IEEE, Beijing, China, May 2013.

[9] M. Zaharia, M. Chowdhury, T. Das et al., "Resilient distributed datasets: a faulttolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systens Design and Implementation*, USENIX Association, San Jose, CA, USA, April 2012.

[10] M. R. Ronchi, S. Kim, and Y. Yue, "A rotation invariant latent factor model for moveme discovery from static poses," *IEEE*, vol. 16, pp. 11–17, 2017.

[11] Y. L. Shen and R. M. Jin, "Leaning personal + social latent factor model for social recommendation," in *Proceedings of the 18th ACM SIGKDD Intemational Conference on Knowledge Discovery and Data Mining*, pp. 1303–1311, Beijing, China, August 2012.

[12] N. Polatidis and C. K. Georgiadis, "A dynamic multi-level collaborative filteringmethod for improved recommendations," *Computer Standards and Interfaces*, vol. 51, 2017.

[13] M. Duan, K. Li, X. Liao, and K. Li, "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2337–2351, 2018.

[14] E. Elsebakhi, F. Lee, E. Schendel et al., "Large-scale machine learning based on functional networks for biomedical big data with high performance computing platforms," *Journal of Computational Science*, vol. 11, pp. 69–81, 2015.

[15] W. Lin, Z. Wu, L. Lin, A. Wen, and J. Li, "An ensemble random forest algorithm for insurance big data analysis," *IEEE Access*, vol. 5, no. 5, pp. 16568–16575, 2017.

[16] Á B. Hernández, M. S. Perez, S. Gupta, and V. Muntés-Mulero, "Using machine learning to optimize parallelism in big data applications," *Future Generation Computer Systems*, vol. 86, no. 86, pp. 1076–1092, 2018.

[17] S. Ramírez-Gallego, S. García, J. M. Benítez, and F. Herrera, "A distributed evolutionary multivariate discretizer for big data processing on Apache spark," *Swarm and Evolutionary Computation*, vol. 38, pp. 240–250, 2018.

[18] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3-4, pp. 145–164, 2016.

[19] B. Zhao, H. Zhou, G. Li, and Y. Huang, "Large-scale topic model training on distributed data-parallel platform," *Big Data Min Anal*, vol. 1, no. 1, pp. 57–74, 2018.

[20] Y. Yan, M., L. Lu, and C. Guo, "Big-data-driven Based Intelligent Prognostics Scheme in Industry 4.0 Environment," in *Proceedings of the 2017 Prognostics and System Health Management Conference*, pp. 1–5, PHMHarbin), Harbin, China, July 2017.

[21] K. Zhang, Y. Tanimura, H. Nakada, and H. Ogawa, "Understanding and improving disk-based intermediate data caching in Spark," in *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*, pp. 2508–2517, Boston, MA, USA, December 2017.

[22] S. Caı´no-Lores, J. Carretero, B. Nicolae, O. Yildiz, and T. Peterka, "Spark-Diy: A framework for interoperable spark operations with high performance block-based data models," in *Proceedings of the 2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT)*, pp. 1–10, Zurich, Switzerland, December 2018.

[23] G. Ditzler, S. Hariri, and A. Akoglu, "High performance machine learning (HPML) framework to support DDDAS decision support systems: design overview," in *Proceedings of the 2017 IEEE 2nd International Workshops on Foundations and Applications of Self∗ Systems (FAS∗W)*, pp. 360–362, Tucson, AZ, USA, September 2017.

[24] A. Gupta, H. K. Thakur, R. Shrivastava, P. Kumar, and S. Nag, "A big data analysis framework using Apache spark and deep learning," in *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 9–16, New Orleans, LA, USA, November 2017.

[25] A. T. Hadgu, A. Nigam, and E. Diaz-Aviles, "Large-scale learning with AdaGrad on spark," in *Proceedings of the 2015 IEEE International Conference on Big Data (Big Data)*, pp. 2828–2830, Santa Clara, CA, USA, November 2015.

[26] A. Koliopoulos, P. Yiapanis, F. Tekiner, G. Nenadic, and J. Keane, "A parallel distributed weka framework for big data mining using spark," in *Proceedings of the 2015 IEEE International Congress on Big Data*, pp. 9–16, New York, NY, USA, July 2015.

[27] S. N. Lighari and D. M. A. Hussain, "Testing of algorithms for anomaly detection in big data using Apache spark," in *Proceedings of the 2017 9th international conference on computational intelligence and communication networks (CICN)*, pp. 97–100, Girne, Northern Cyprus, September 2017.