A Fast Clustering-based Recommender System for Big Data

Hong-Quan Do*[1], T.H.-An Nguyen**, Quoc-Anh Nguyen**, Trung-Hieu Nguyen**, Viet-Vu Vu*, Cuong Le*

*VNU Information Technology Institute, Vietnam National University, Hanoi, Vietnam

**FPT University, Hanoi, Vietnam

[1]Corresponding Author: quandh@vnu.edu.vn

Abstract—**For years, recommender systems (RS) have emerged as a powerful tool to enable users to find appropriate information according to their needs. Different recommendation methods have been proposed and can be categorized as collaborative filter, content-based, and Hybrid/Ensemble approach. However, the exponential growth of digital information in the recent decades often referred to Big Data, poses new challenges for the current RS. Following this spirit, our work proposes a novel fast clustering-based Recommendation method (denoted as FCR) designed on top of Apache Spark. Comprehensive experiments on a real-world dataset have verified the advantages of our proposed method. It is effective in alleviating the problem of data sparsity and item cold-start. The training and inference time is quick while the slight increase of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) is acceptable.**

Keywords—**Recommender System, Big Data, Clustering-based Recommendation, Item cold-start, Data sparsity, Apache Spark**

I. INTRODUCTION

The remarkable effectiveness of recommendation systems (RS) has been demonstrated in the problem of information overload as early as the 1990s. Nowadays, they are a significant integral part of most modern systems, for example, in many E-commerce platforms like Amazon, E-Bay, Lazada, or Vietnamese companies such as Tiki, Mobile World Investment Corporation, to mention but a few. Users will receive personalized suggestions from RS based on their previous choices and interests.

Over the years, different recommendation methods have been proposed and can be broadly categorized as Collaborative filtering (CF), Content-based (CBF), and Hybrid/Ensemble approach. To begin with, CF computes the similarity between users or items based on the rating information of customers, thence inferring recommendations. Content-based recommender systems (CBF), on the other hand, make recommendations according to the features of items - such as movie title, actors, or its genre - that the user preferred in the past. Moreover, these two mentioned approaches can naturally be integrated into a hybrid/ensemble recommender system [1].

However, it is worth noting that while there were increasing efforts to investigate new methods for RS, not many works have adapted RS in a big data environment. The phenomenon of the Big Data explosion was realized as massive growth in data. It poses many challenges in storing, integrating, and managing these large volumes of data, and last but not least, extracting relevant information needed in such large volumes of data. Not surprisingly, although several recommendation methods have proved good performance for small-scale datasets, applying them in Big data environment is not straightforward (Figure 1). For example, collaborative filtering (CF) could be one of the most popular and widely-used recommendations since it has received success in different application areas [2], [3], [4]. Most of the CF based Recommender Systems work only on the user-item interaction/rating matrix. Thus it is useful when side information is limited or hard to collect. However, in real scenarios, when many users could supply very few ratings, the user-item interaction matrix is very sparse. Besides, the large number of users/items could also make the matrix huge to store, and the recommendation task costs much time.

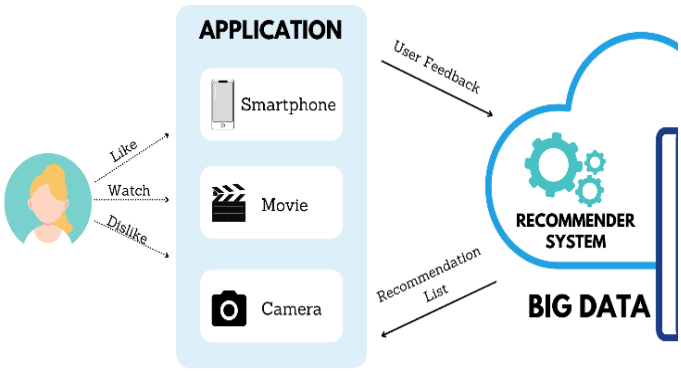


Figure 1. Recommender System in Big Data Environment

These reasons thereby motivate us to propose a novel straightforward but fast clustering-based recommendation method. It can address the problem of data sparsity and item cold-start and can be adapted to handle large-scale data set. The main idea behind the technique is a divide-and-conquer strategy in which the number of users can be grouped into different clusters based on their preferences. The prediction can also be made directly according to their preferences and the importance of the user within their cluster. Experimental results point out that the method is effective in alleviating the problem of new item cold start. The training and inference time is quick while the slight increase of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) is acceptable. We denoted our method as FCR – **F**ast **C**lustering **R**ecommender.

The contributions of this work can be summarized as follows.

- Since many users are a significant issue in large-scale datasets, our proposed method first adopts a clustering algorithm to divide a set of users into *k* clusters, assuming that the users in the same cluster will have similar tastes. This work establishes a new way to represent user features according to the user's preference of movie genre.
- To advance the use of clusters and boost the inference time, we define the importance of the user within their cluster, then the rating prediction can be made directly according to their preferences and significance in the cluster.
- We adapt our proposed recommender system to run in the context of big data. The data is stored in Hadoop Distributed File System (HDFS), and operations are performed on Apache Spark.
- Experiments are conducted on the real-world dataset MovieLens to verify our proposed method regarding MAE, RMSE, the training and inference time.

The remaining parts of this report will be structured as follows. The related works will be discussed in Chapter 2 while Chapter 3 summarizes some of the essential research backgrounds. Next, we depict our proposed method - FCR in Chapter 4. The implementation and evaluation of our work will be described in Chapter 5. Finally, the report draws a conclusion and future work in Chapter 6.

II. RELATED WORK

The utilize of clustering in recommendation process has received several researchers' attention. For example, [5] proposed a collaborative filtering algorithm based on clustering user preferences, in which users are divided into optimistic user groups, pessimistic user groups, or neutral user groups. To highlight the importance of user preference, the authors proposed a new similarity measure and defined a neighbor set of users to be used in their user-user CF. Additionally, [6] proposed a reliable neighbors-based collaborative filtering in which they adopted K-means on the rating matrix but in two different tasks. Firstly, the items are clustered in order to estimate the missing ratings. The obtained estimated rating matrix is then used to cluster users. Then, the final rating of user *u* for item *i* is computed according to the similarity between *u* and those users in the same set S with *u* – The set S is defined as the intersection between the cluster of user *u* and the reliable user set related to item *i*.

To address the problem of data sparsity and ratings diversity, [7] proposed a rating-based algorithm, named RBRA. RBRA represents a user by constructing a vector of mean, variance, and range of the user's rating to find their neighbors more reliably. In contrast, BiFu [8] uses a bi-clustering method to fuse both user and item information in order to infer predictions.

Particularly in the era of big data, we have seen an exponential increase in the volumes of data in recent decades. Rapid progression in technology and increasing use of online interactions has led to data explosion. This greatly inspired researchers to improve the performance of recommender systems or adapt them in the context of big data for handling large-scale datasets. In 2013, [9] used matrix factorization and random forest on big data recommendation systems that focuses on dividing the big data problem into smaller subproblems to speed up the process. In 2015, [10] explained how to use a recommendation system with big data. The paper proposed using CF, CBF, and hybrid filtering on Hadoop framework. Their result showed that with properly applied algorithms, the program's running time only increases slightly while the amount of the data is grown significantly. In 2018, [11] evaluated the effect of Big Data on recommendation quality by experimenting with the performance of a regression model on the example of Internet Search.

III. PRELIMINARIES

This section presents essential research backgrounds necessary for understanding the remaining parts of the paper, including the mathematical notations used, K-means clustering, collaborative filtering, and the big data frameworks.

A. Mathematical Notations

In this work, we use consistent mathematical notations in the rest of this paper, as summarized in Table 1.

TABLE 1. SYMBOL DENOTATION

U	The entire set of users. n = $\|U\|$
I	The entire set of items. m = $\|I\|$
R	The user-item interaction matrix consists of n users and m items
$r_{u,i}$	A rating of user *u* to the item *i*. If $r_{u,i}$ is "observed", $r_{u,i} \in \{1,2,3,4,5\}$, otherwise $r_{u,i}$ is set by zero implying an "unobserved" rating
Z	The set of non-missing entries, $Z = \{(u,i): r_{u,i}\ is\ observed\}$
$p_{u,i}$	The predicted rating of the user *u* for item *i*
$\bar{r_u}$ $\bar{r_i}$	The average of a user *u*'s rating and an item *i*'s rating respectively
r_u	The vector of all ratings provided by user *u*
r_i	The vector of all ratings to the item *i*
U_i	the set of users who rated item *i*
I_u	the set of items rated by user *u*
C_u	The user set who are in the same cluster with user *u*
$sim(u,v)$	The similarity score between user *u* and user *v*
$sim(i,j)$	The similarity score between item *i* and item *j*
k	Number of desired clusters
k'	Number of neighbours used for ranking
$S[u]$	The set of k'-nearest neighbours for the user *u*
X	The user feature/preference, in which X_u is the user preference of user *u*

B. K-Means Clustering

The main objective of clustering is to partition a data set into K clusters such that data in the same cluster is much more similar than those in a different cluster. Clustering is an unsupervised learning technique and is a critical task in machine learning and data mining for discovering structure and the relationship inside the data set. There are different kinds of clustering algorithms, such as partition-based, density-based, graph-based, to name a few.

K-Means [12] is a partition-based algorithm considered simple but highly effective because of its compatibility with high-dimensional data. The algorithm preselects K initial clustering centers (named as centroids), then iteratively assigns each data point to the nearest centroid. After that, it computes the mean of all the points for each cluster and sets the new centroid. This clustering process runs iteratively until the centroids converge. The steps of K-means are summarized in Algorithm 1.

Algorithm 1. K-Means clustering
Input
X: The dataset of points $\{x_1, x_2, \dots x_n\}$; k: Number of desired clusters; MaxIters: limit of iterations
Output
- Centre $\{c_1, c_2, \dots c_k\}$ implicitly dividing X into k clusters $\{C_1, C_2, \dots C_k\}$
- Cluster label of each point x_i
Begin
1. Randomly initialize k centroids $\{c_1, c_2, \dots c_k\}$
2. Declare iter ← 0
3. Repeat until the centroids converge OR (iter == MaxIters)
  3.1. Assign each point x_i to its closest centroid
  3.2. Set the new centroid c_i as the centre of mass of all points in C_i
  3.3. iter++
End

C. Collaborative Filtering

The work concerns the user – user collaborative filtering to suggest items for a user based on their similarity with other users (denoted as USER-CF in Algorithm 2). Given the user-item rating matrix $R^{n\,x\,m}$, to infer the rating $p_{u,i}$ the algorithm first calculates the similarities between the target user *u* and all other users. Next, it selects the top k' similar users and takes the weighted average of ratings from these k' users with similarities as weights.

The adjusted cosine similarity [13] can be used to compute how close two users are, as given in Equation 1.

$$Sim\,(u,v) = \frac{\sum_{j\in I_u\cap I_v}(r_{u,j}-\overline{r_u})(r_{v,j}-\overline{r_v})}{\sqrt{\sum_{j\in I_u\cap I_v}(r_{u,j}-\overline{r_u})^2}\sqrt{\sum_{j\in I_u\cap I_v}(r_{v,j}-\overline{r_v})^2}} \quad (1)$$

, where $\overline{r_u}$ and $\overline{r_v}$ is the average of the user u's ratings and v's rating respectively. $Sim\,(u,v)$ ranges in [0,1], and the larger $Sim\,(u,v)$ presents two user are more closely similar.

To make a prediction, let S[u] be the set of k'-nearest neighbours for the user u, the estimate rating of user *u* for item *i* is computed in Equation 2 as a weighted average of the k'-nearest neighbours.

$$p_{u,i} = \overline{r_{u_{new}}} + \frac{\sum_{v\in S[u]} sim(u,v)*(r_{v,i}-\overline{r_v})}{\sum_{v\in S[u]}|sim(u,v)|} \quad (2)$$

Complexity Analysis: This CF approach takes time complexity of $O(n^2.m.k')$, and the space complexity of $O(n.k')$ for the recommendation training. For the prediction step, it is a trade-off between time complexity and space complexity here. If all pairs of user similarity do not be stored or a completely new user appears, it takes $O(n^2.m.k')$ for one rating prediction, otherwise it takes $O(k')$.

Algorithm 2. USER-USER COLLABORATIVE FILTERING (USER-CF)
Begin
1. **for each** user u ∈ U\U_i **do**
  1.1. neighbours ← ∅, $p_{u,i}$ ← 0 , denominator ← 0
  1.2. **for** v in U **do**
    neighbours[v] ← $Sim(u,v)$ (see Eq.1)
  1.3. $S[u]$ ← sort(neighbours)[: k']
  1.4. **for** v in $S[u]$ **do**
    1.4.1. $p_{u,i}$ += $Sim(u,v) * r_{v,i}$
    1.4.2. denominator += $Sim(u,v)$
  1.5. **end for**
  1.6. $p_{u,i}$ = $p_{u,i}$ / denominator (see Eq.2)
2. **end for**
3. **return** p
End

D. Big Data Frameworks

Big data has great potential but raises new challenges in storing, integrating, and managing large volumes of data. Both academics and industry have given big data much attention. Google introduced the MapReduce approach for processing large-scale data in early 2008 [14]. Yahoo launched an open-source implementation of MapReduce, named Apache Hadoop two years later [15]. Hadoop Distributed File System (HDFS) is primary data storage system in Hadoop that parallelizes files in a Hadoop-native format across clusters.

In contrast, Apache Spark is also an open-source, but relatively newer project [16]. Spark processes and retains data in RAM using a concept known as Resilient Distributed Dataset (RDD). RDD is a distributed collection of data partitioned across nodes. Due to the in-memory technology of Spark RDDs, the performance of Spark has been reported to be optimal over Hadoop [16].

IV. DATA AND PROPOSED RECOMMENDATION METHOD

A. Dataset Descriptions

One of the most widely used benchmark data sets for studying and researching recommender systems is MovieLens 100K [17]. It can be observed that the data sparsity and item cold-start problem occur in this dataset. First, a high level of sparsity indicates that most values in the rating matrix are unknown [1]. The sparity level of MovieLens 100K is 93.695%; it could result in impediments to practical recommendations. Second, the distribution of movie ratings depicted in Figure 2 reveals the fact that the bulk of movies are relatively unpopular with only few ratings provided. This potential issue of item cold-start will be clarified in detail during our experiment in Chapter 5. Some statistics of the data set is shortly shown in Table 2.

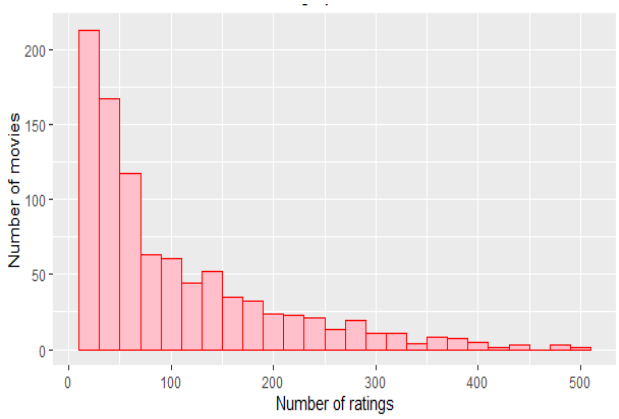


Figure 2. The distribution of ratings per movie in MovieLens 100K [1]

Two dataset files are employed in our experiments, including "*u.data*" containing the ratings and "*u.item*" showing the genre information about movies. An item can belong to one or many among the 19 genres. Apart from that, no demographic information is used.

TABLE 2. DESCRIPTIONS OF THE DATASET USED.

	#Users	#Items	#Ratings	#Genres	Sparsity
Movie Lens 100k	943	1,682	100,000	19	93.695%

B. Training step

Given a training set as the user-item interaction matrix R, our proposed method starts with a clustering process. The set of users will be divided into different clusters. To do so, we establish a user feature/preference for each user. Assume that the category of an item is known – In case of MovieLens dataset the genre of movie is among 19 different categories, the user preference according to their preference of movie genre is introduced in Definition 1.

Definition 1. Let H the set of available categories, and R_{u,C_p} the set of ratings that a user u provides to those items of category H_p, the user feature of user u is X_u given by:

$$X_u = \begin{cases} \frac{1}{|R_{u,C_p}|}\sum_{u,i \in R_{u,C_p}} r_{ui}, & |R_{u,C_p}| > 0 \\ \frac{1}{|C \setminus C_p|}\sum_{v \in U, v \# u} H_v, & |R_{u,C_p}| = 0 \end{cases} \quad (3)$$

The user feature X is used as the input for Algorithm 1 to divide U into k clusters $C = \{C_1, C_2, \dots C_k\}$. To evaluation the clustering quality such that the optimal value of k could be selected, we use the Elbow Method [18].

It is worth noting that storing data in big data platform is a little bit more sophisticated. First, we dump the dataset into Hadoop as Hadoop Distributed File System (HDFS). Then, for the clustering task, we load appropriate data into different partitions. The K-means is configured to run on these partitions in parallel (Figure 3).

The main tasks in the master node are as follows.

- Horizontally split the data set and distribute them to RDDs
- Randomly initialize centroids $\{c_1, c_2, \dots c_k\}$ and broadcast them to all nodes.
- Receive the information about clusters and centroids

The tasks in each client node are as follows:

- Receive initial centroids $\{c_1, c_2, \dots c_k\}$ and sub-dataset from the master node
- Calculate distances between all instances locally.
- Assign data to an appropriate cluster
- Send partial information of cluster and centroid back to the master node.

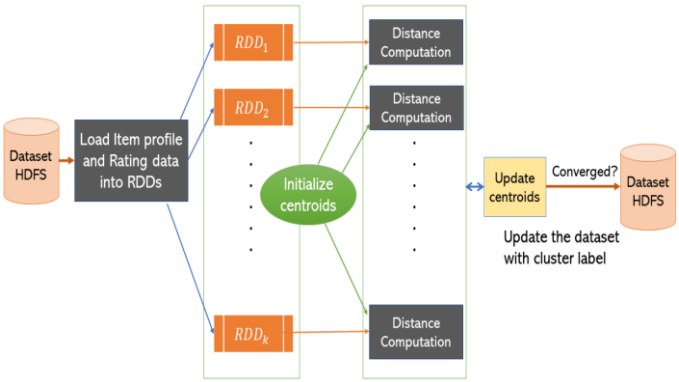


Figure 3. The configure to run K-means clustering in parallel in many RDDs

Once the set of clusters is given, for each user we establish their importance within the clusters as shown in Definition 2.

Definition 2. The importance of a user within their cluster $\tau_{C_u}^u$ is defined as:

$$\tau_{C_u}^u = \frac{1}{2}\left(\frac{n_u - min_{v \in C_u}(n_v)}{max_{v \in C_u}(n_u) - min_{v \in C_u}(n_v)} + 1\right) \quad (4)$$

, where n_u is the number of ratings that the user u provides in the training set.

τ will be in the range of [0.5, 1]. The Eq.4 means that the greater number of ratings the user u supplies, the more important he/she is.

C. Prediction step

The method is proposed to provide recommendations directly based on the given user preference X_u introduced in Definition 1 and the factor $\tau_{C_u}^u$ introduced in Definition 2. Let $p_{u,i}$ be the predicted rating of the user u for item i, it is computed as follows.

$$p_{u,i} = \tau_{C_u}^u * X_u^i + \left(1 - \tau_{C_u}^u\right) * \frac{\sum_{v \in C_u} r_{v,i}}{|C_u|} \quad (5)$$

The linear combination of rating prediction indicates that the more important a user is within their cluster, the more contributions in total prediction are based on the user's preference themself. The average rating plays a role of a baseline.

D. Complexity Analysis

The training step only takes $O(n)$ for establishing the user preference, but the time complexity mostly depends on the selection of the clustering algorithm. In the naive solution of K-means and running it in the sequential mode, if the number of clusters k and the dimension d are fixed, the problem can be exactly solved in time $O(n^{dk+1})$. However, in the parallel mode, let t is the number of iterations, k is the number of RDDs, the total time complexity is $O(n.k.t)$. For the prediction step, since the proposal method provides recommendations directly based on the user preference, one prediction takes $O(1)$, noting that the average rating within a cluster can be pre-computed in time $O(max_{i\in U}(|C_i|)$.

V. IMPLEMENTATION AND EVALUATION

A. Evaluation metrics

Two well-known metrics for evaluating the performance accuracy are Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) [19]. They are defined as follows:

$$MAE = \frac{1}{|T|}\sum_{u,i\in T}|(r_{ui} - p_{ui}| \quad (6)$$

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{u,i\in T}(r_{ui} - p_{ui})^2} \quad (7)$$

where r_{ui} and p_{ui} refer to the ground-truth rating and estimated rating of the user u for item i in the test set T respectively while $|T|$ is the number of ratings in the test set.

Both metrics are ranges from 0 to ∞, and a lower error value indicates a better recommendation accuracy. However, the different is that in MAE the errors scale linearly while in RMSE the errors are squared meaning that RMSE gives relatively high weight to larger errors.

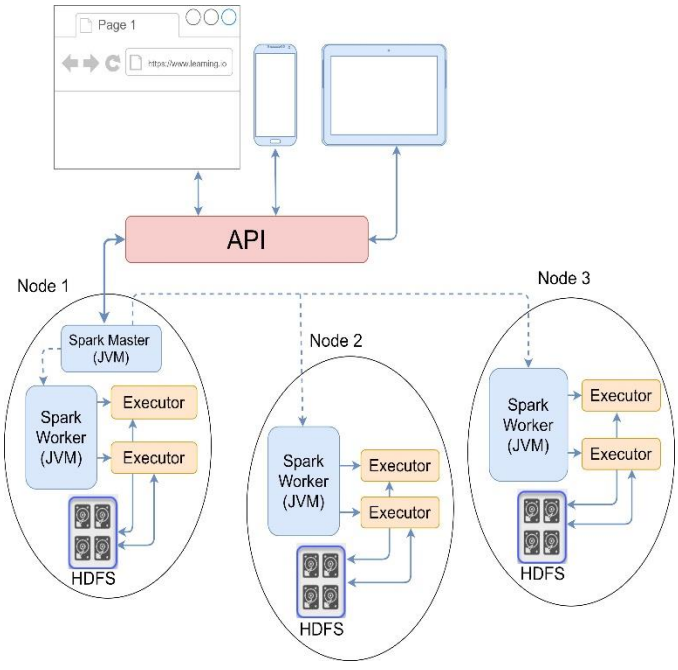


Figure 4. Overview of Our System Architecture

B. Experiment setup

Our system is built with the data layer and the application layer that are connected via API drivers. The data set is written to HDFS. Then for each task, appropriate data is load in a distributed way into RDDs and transformed to be analysed using Spark. We develop the data layer containing 3 nodes: one for Spark Master and two for Spark Workers. Recommendation method is implemented and run in the Spark cluster mode. In the software configures, we use Spark 3.1.2, Hadoop 2.7.7, and Python 3. The system was built in virtual machines with the hardware configures illustrated in Table 3, and Figure 4 gives a general review of our system architecture.

TABLE 3. THE HARDWARE DESCRIPTION

	Memory (Gb)	Disk (Gb)	OS
Master	8	60	Ubuntu 18.0.4
Worker-1	4	30	Ubuntu 18.0.4
Worker-2	4	30	Ubuntu 18.0.4

Performance comparison

We implement 2 recommendation algorithms: User-based collaborative filtering (USER-CF) - which is a collaborative filtering method based on user adjusted cosine similarity illustrated in Algorithm 2, and our proposed method (FCR). We applied five-fold cross validation strategy for experiments. By that, the dataset is split randomly into a training set and a test set in a ratio of 8:2 respectively. The number of neighbours being set in USER-CF is 30.

Besides normal cases, we validate the effectiveness of our proposed method in addressing the item cold-start problem. This problem appears when new items arrive with no or very few interactions. In the whole MovieLens 100K dataset, there are 141 movies received only 1 rating. Among them, we select items appeared in the test set, but not in the training set to perform other experimental comparisons. All results obtained in the normal case and new item cold-start case are depicted in Table 4.

TABLE 4. THE PERFORMANCE COMPARISION OF IMPLEMENTED RECOMMENDATION ALGORITHMS ON MOVIELENS 100K

	MAE	RMSE	Training time (ms)	Inference Time/ rating (ms)
USER-CF	**0.78**	**0.99**	0.16	2.123
FCR	0.80	1.02	**0.053**	**0.016**
USER-CF (Item cold-start)	0.89	1.17	-	0.277
FCR (Item cold-start)	**0.88**	**1.12**	-	**0.002**

Furthermore, in terms of only MAE and RMSE, we compared some of those public methods in the literature that solved the close issues and experimented on the same dataset as summarized in Table 5.

TABLE 5. THE PERFORMANCE COMPARISION WITH PUBLIC RESEARCH METHODS ON MOVIELENS 100K

	MAE	RMSE
RBRA [7]	0.8479	1.0387
BiFu [8] [6]*	0.8359	1.0196
RNCF [6]**	0.5199	0.6703
FCR	0.80	1.02

* *BiFu has not been tested in MovieLens 100K in its orginial paper [8]. This result of BiFu is reported from [6].*

** *The performance of RNCF is varied according to a vast number of parameters. The presented results are the best one obtained according to the author.*

Obviously, from the obtained results, it can be seen that our method addresses the problem of item cold-start effectively - 4.46% less error than the traditional USER-CF method. Besides, the training time is 3 times faster and the average inference time is almost 133 times faster than USER-CF as a consequence of the fact that the rating is predicted in a constant time.

Nevertheless, as a trade-off for a fast execution time, when we evaluate our recommendation method in two prediction accuracy metrics: MAE and RMSE on the whole testing set, their values increased slightly (3% larger) compared to those of USER-CF and quite far worse than the RNCF algorithm [6]. However, the method is proven better than RBRA [7] and BiFu [8]. The results of RBRA, BiFu, and RNCF are presented according to the original papers. Since we have not done any re-implementation of these algorithms, we have not evaluated them in their ability to alleviate the item cold-start issue and their time complexity.

VI. CONCLUSIONS

The work has proposed a fast clustering-based recommendation method addressing data sparsity and the item cold-start problem. The method was designed on top of Apache Spark. The experimental results showed that the training and inference time of the method is speedy while the slight increase MAE and RMSE is acceptable. The limitation of the work is the experimental dataset is somewhat small. In our future work, experiments on larger datasets will be conducted. Besides, we expect a design of new clustering-based methods by exploiting the properties of matrix factorization approaches.

REFERENCES

[1] H. Q. Do, T. H. Le and B. Yoon, "Dynamic Weighted Hybrid Recommender Systems," *International Conference on Advanced Communication Technology,* no. 22, pp. 644-650, 2020.

[2] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme and G. Stumme, "Tag recommendations in folksonomies," in *European Conference on Principles of Data Mining and Knowledge Discovery*, 2007.

[3] Y. Koren, R. Bell and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE,* no. 0018-9162/09, pp. 42-48, 2009.

[4] H. Hwangbo, Y. S. Kim and K. J. Cha, "Recommendation system development for fashion retail e-commerce," *Electronic Commerce Research and Applications,* vol. 28, pp. 94-101, 2018.

[5] J. Zhang, Y. Lin, M. Lin and J. Liu, "An effective collaborative filtering algorithm based on user preference clustering," *Applied Intelligence,* vol. 45, 2016.

[6] L. Zhang, X. Sun, J. Cheng and Z. Li, "Reliable Neighbors-Based Collaborative Filtering for Recommendation Systems," in *Neural Computing for Advanced Applications,* 2020, pp. 60-71.

[7] F. Xie, M. Xu and Z. Chen, "RBRA: a simple and efficient rating-based recommender," in *International Conference on Advanced Information Networking and Applications Workshops*, 2012.

[8] D. Zhang, C. Hsu, M. Chen, Q. Chen, N. Xiong and J. Lloret, "Cold-Start Recommendation Using Bi-Clustering and Fusion for Large-Scale Social Recommender Systems," *IEEE Trans. Emerg. Topics Comput.,* vol. 2, no. 2, pp. 239-250, 2014.

[9] B. A. Hammou, A. A. Lahcenab and S. Mouline, "An effective distributed predictive model with Matrix factorization and random forest for Big Data recommendation systems," *Expert Systems with Applications,* vol. 137, pp. 253-265, 2019.

[10] J. P. Verma, B. Patel and A. Patel, "Big Data Analysis: Recommendation System with Hadoop Framework," *International Conference on Computational Intelligence & Communication Technology,* pp. 92-96, 2015.

[11] S. Maximilian, G. Sapi and S. Lorincz, "The effect of big data on recommendation quality: The example of internet search," *Econstor,* no. 284, 2018.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Berkeley Symposium on Mathematical Statistics and Probability,* 1967.

[13] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based Collaborative Filtering Recommendation Algorithms," in *Proceedings of ACM World Wide Web Conference,* 2001.

[14] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," vol. 51, no. 1, p. 107–113, 2008.

[15] A. S. Foundation, "Hadoop," 2010. [Online]. Available: https://hadoop.apache.org.

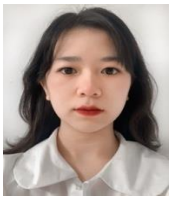
[16] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker and I. Stoica, "Apache Spark: a unified engine for big data processing," *Commun. ACM,* vol. 59, no. 11, p. 56–65, 2016.

[17] F. M. Harper and J. A. Konstan., "The MovieLens Datasets: History and Context," *ACM Trans. Interact. Intell. Syst.,* vol. 5, no. 4, 2016.

[18] D. Ketchen and C. Shook, "The application of cluster analysis in strategic management research: an analysis and critique.," *Strateg. Manag. J.,* vol. 17, no. 6, p. 441–458, 1996.

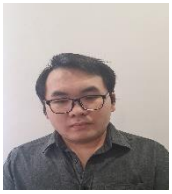
[19] A. Gunawardana and G. Shani, "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks," *Journal of Machine Learning Research* , vol. 10, pp. 2935-2962, 2009.

Hong-Quan Do received a double M.S. degree in Information and Communication Technology from University of Science and Technology of Hanoi, Vietnam and The University of Rennes 1, France in 2015. He is a researcher at Information Technology Institute, Vietnam National University, Hanoi. His research concentrates primarily on Clustering, Semi-supervised clustering, and Image processing. At the present, he has been involved in many projects related to E-government, and E-Commerce Recommendation applications.



T.H.-An Nguyen is studying at the University of Greenwich. She was in the top 3 of the school during the Summer 2021. Her research interests include Recommender Systems and Regression.

Quoc-Anh Nguyen is a student in the major of Information Technology at the University of Greenwich Vietnam. He was honoured as the best student in the 4th semester. He has experiences in developing and implementing real-time, API, Microservices systems. Currently, he has been involved in different website projects including financial, healthcare, and HR management systems.



Trung-Hieu Nguyen is a 6th semester student at the University of Greenwich (Vietnam). He was an honored student in the 5th semester for being the best at Business Intelligent subject. His current study goal is to focus on working with Machine Learning with Python. His future target is working in the field of Clustering and Regression.

Assoc. Prof. **Viet-Vu Vu** received the B.S. degree in Computer Science from Ha Noi University of Education in 2000, a M.S. degree in Computer Science from Ha Noi University of Technology in 2004, and a Doctor Degree in Computer Science from Paris 6 University in 2011. He is a researcher at Information Technology Institute, Vietnam National University, Hanoi. His research interests include clustering, active learning, semi-supervised clustering, and E-government applications.

Dr. **Cuong Le** is a lecturer in School of Applied Mathematics and Informatics (SAMI), Hanoi University of Sciences and Technology (HUST) from 1998 to 2016. Since 2016 until now he in Information Technology Institute (ITI), Vietnam National University, Hanoi (VNU, Hanoi) since 2016. He got his PhD at HUST. His research interests lie in the area of information security, mathematics computation and quaternion and Clifford analysis as well as PDE.