An Enhanced Apriori Algorithm Using Hybrid Data Layout Based on Hadoop for Big Data Processing

Yassir ROCHD† and Imad HAFIDI†,

IPOSI Laboratory, National School of Applied Science, Hassan I University, Khouribga, Morocco

Summary
Frequent itemset mining is one of the data mining methodes implemeted to find frequent patterns, utilized in prediction, association rule mining, classification, etc. Apriori algorithm is an iterative method , that is used to discover frequent itemsets from transactional dataset. It scans entire dataset in every iteration to come up with the large frequent itemsets of various cardinality, which sounds efficient for small data but not useful for big data. To resolve the problem of treatment dataset in every iteration, we present an algorithm called Hybrid Frequent Itemset Mining on Hadoop (HFIMH) which uses the vertical layout of dataset to solve the problem of treatment the dataset in every iteration. Vertical dataset conveys information to discover support of every itemsets, and the idea of set intersection is utilized to compute it. We compare the execution of HFIMH with another Hadoop based implementation of Apriori algorithm for different datasets. Experimental results demonstrate that our approach is better.

Key words:
Data mining, Frequent itemset mining, Apriori, Big data, Hadoop.

1. Introduction

Recently the enormous advances in science and technology have affected the data size. Morever to its extensive , these enormous data are also unstructured anf semi unstructered in nature, involving the difficulties in capture, storage,sharing, search and analysis.that's why it's named "Big Data" [1, 2].

Data mining is the way of analyzing data from huge amount of stored information in organisation for decision making.it stands as a search ground to discover concealed information from the database.There different roles requiered in data mining such as Classification, association mining rule, clustering, sequential pattern discovery, etc [3]. By association mining rule we mean a rule-based learning technique, finding significant relationships between data object in dataset. So, just frequent itemsets are taken to build the association rules to decrease the number of potential itemsets.In order to extract frequent itemsets from data, different algorithms

along with differentameliorations were suggested, for example Apriori [4], FP-Growth [5] and Eclat [6].

Apriori algorithm is one of the mostly used methods to discover item set from transactional database. The use of the original Apriori algorithm became ineffective to treat, when the sharp expanded in data size. Big data need a high set of resources for storage and treatment Traditional single-machine computation is not sufficient to process with big data. Thus , multi-machine handling is reasonable to store and deal huge data in distributed mode. There are numorous distributed computing frameworks available, e.g., MPI, Hadoop, etc. Hadoop [7] is an open-source framework for big data processing in distributed environment. Hadoop uses the MapReduce programming model to write applications and execute them across cluster of machines in distributed manner.

Many limitations of Apriori algorithm in distributed environment are as follows:

1. The Apriori algorithm scans the entire dataset in every iteration to calculate the support of each itemset.
2. For big transactional data, a huge number of candidate sets are generated in every iteration.
3. All candidate itemsets needs the storage in main memory for further handling to produce frequent itemsets.

To resolve the mentioned limitations, we suggest an effective frequent pattern mining algorithm named Hybrid Frequent Itemset Mining on hadoop (HFIMH), which exploits the concept of vertical dataset. We compare our algorithm with other Hadoop-based Apriori implementation and noticed that our algorithm outperforms in respect of execution time.

The organization of the paper is as follows. Section 2 describes the preliminaries of HFIMH, i.e., Apriori algorithm and Hadoop framework. Section 3 discusses about the related work in the area of frequent pattern mining. In Sect. 4, we propose the HFIMH algorithm followed by experiments and result analysis in Sect. 5. Finally, we conclude the paper in Sect. 6.

Manuscript received June 5, 2018
Manuscript revised June 20, 2018

2. Preliminaries

2.1 Apriori and association rules

Association rule mining [8] is a process which is meant to discover frequent patterns, correlations, associations, or causal structures from data set, transactional databases, and other forms of data repositories. Given a set of transactions, association rule mining aims to discover the rules which enable us to predict the occurrence of a specific item based on the occurrences of the other items in the transaction.

Association rules indicates itemsets that occur frequently in transactions [9]. The mining or formation of association rules consists of two steps:

- Frequent Itemset Mining: finding the itemsets that occurs frequently in the database from on support.
- Association Rule Generation: generating large rules from the frequent itemsets based on confidence.

habitually, an Association Rule Mining Algorithm results in the production of a large number of association rules and it is difficult for users to validate all the rules manually. Thus, only interesting rules or non-repeating rules are to be generated.

The Apriori Algorithm, which utilizes a bottom up approach for the generation of candidate itemsets, is the more widely accepted algorithm for Association Rule Mining. This algorithm works based on the property called as Apriori Property which means that “all the subsets of a frequent itemset must also be frequent” [9]. If we know that a particular itemset I is infrequent, then it is not required to count its supersets [4].

2.2 Hadoop and Mapreduce

Encouraged by benefits of parallel execution in the distributed environment, the Apache Foundation came up with open source platform, Hadoop, for faster and easier analysis and storage of different varieties of data [7]. HDFS and MapReduce programming model are two integral parts of it. Google File System gave birth to HDFS (Hadoop Distributed File system) [10], which mainly deal with storage issues. Contrary to the RDBMs, it follows WORM (write-once read-many) model in order to split large chunk of data to smaller data blocks then join them to the free node available [11]. Stored Input data blocks are kept in more than one node in order to achieve high performance and fault tolerance.

MapReduce which is inspired by Google's MapReduce [12] is known to be a linearly adaptable programming model. It contains two main functions a map ( ) function and a reduce ( ) one, both of which work in a synchronous manner in order to operate on one set of key value pairs, and that, to produce the other set of key value pairs [11].These functions are equally valid for any size of data irrespective of the degree of the cluster. MapReduce uses the feature known as data locality to collocate the data with the compute node, so that data access is fast. It follows shared nothing architecture which eliminates the burden from the programmer of thinking about failure. The architecture itself detects failed map or reduce task and assigns it to a healthy node [13, 14].

2.3 Related Work and Problem Statement

Many research works have been carried out in the field of frequent pattern mining and association rule mining. A wide range of knowledge extraction techniques has been discussed that are currently being explored for big data. Many improved versions of Apriori algorithms have been proposed for past decades. Most of the variants of Apriori algorithm were developed to work for small size of data in a singlemachine system. With the introduction of big data for last few years, single-machine system seems to be incapable to handle big data. A lot of research has been made for frequent pattern mining in multi-machine environment, i.e., distributed computing environment. Hadoop is one of the prominent distributed computing frameworks,which is adopted by many researchers for frequent pattern mining in big data.

A parallel version of Apriori algorithm is provided by Li et al. [15], which iteratively produce the frequent itemsets by applying the basic MapReduce functions. Yu et al. [16] have suggested another MapReduce-based Apriori algorithm, which applies possible candidate set generation on each transaction and extract all frequent itemsets in a individual iteration. Thus, it is costly to provide all possible candidate set in memory. The famous ECLAT algorithm is applied in distributed environment [17] to cope with big data. Two new methods named Dist-Eclat and BigFIM are proposed. Dist-Eclat is distributed version of ECLAT which is fairly divide search space on computing nodes and BigFIM is optimization of Dist-Eclat to manage the mining algorithm on large data. There are other distributive implementations of Apriori algorithm which are presented in [18, 19, 20].All the above algorithms are developed in MapReduce over Hadoop framework either in single stage or multi-stage.

In such improving system, the transactional data set is either represented in a horizontal format [18, 19, 23] or vertical format [20, 21, 22, 24]. Most of the previous MapReduce Apriori implementation use horizontal disposition of the dataset. In all such algorithms, to compute the support of any item set, all transactions are scanned one after other in every iteration which is a time-consuming task. The paper [21, 22] implemented the MapReduce-based Apriori algorithm in the vertical format.

Here, generation of candidate item set require extra iteration. The algorithm also produces some unimportant and useless candidate item sets. Counts of these item sets are gathered at the reducer which improves the whole communication cost concerned.

To solve the above limits, we suggest effective frequent pattern mining algorithm, which operates the notion of vertical and horizontal dataset. The support for various cardinality of itemsets is computed by utilizing the vertical dataset shared with all the executors of cluster. Support count is computed by using Intersection algorithm [25]. Our contributions in this paper are listed as follows:

1. A distributed algorithm is proposed for frequent pattern mining, which is implemented in Hadoop framework.
2. Vertical layout of the dataset is used in every iteration to resolve the problem of scanning the complete dataset.
3. The horizontal dataset is distributed and processed on each machine to reduce the number of candidate sets.
4. Generation of the minimum number of subsets of the candidates in order to accelerate the calculation of their supports from the vertical base by the concept of intersections.

3. The proposed algorithm

3.1 Description

The suggested algorithm concentrates on the problem of scanning the whole dataset in each iteration, which results the high I/O cost and disk space. In addition, each node gets to the whole dataset during mining process, which requires a huge capacity for storage of memory. According to our knowledge, none of the distributed implementations of Apriori on hadoop have examined the strategy for sharing for all the nodes, the database of the vertical data revised in each k phases. In order to quicken the calculation of the supports, we used the concept of intersections.

The vertical dataset is made by the list of items/itemset followed by its transaction IDs. The vertical dataset furnishes the benefit that there is no need to scan the entire dataset in every iteration, and it contains enough information to calculate support of the possible candidate sets. In each record of vertical data, TIDs are sorted in ascending order, which makes support computation easier. Support of k-candidate itemset can be calculated by intersection of the TIDs of itemsets, such as we stop the intersection when the minimum support condition is satisfied. If the minimum support is 10, why continue to 15 when we can validate it at 10. Yet, original horizontal data are distributed and processed on cluster nodes to generate the candidates according to each transaction. The complete algorithm is divided into two phases, which are represented in next subsections.

The phase 1 :The first phase of the algorithm generates frequent items of 1-cardinality in a vertical layout. Since we are dealing with big data, dataset may have a great number of transactions. We save huge transactional data in Hadoop distributed file system (HDFS) of Hadoop framework, and multiple partitions of data are distributed across cluster nodes. The vertical dataset includes only the frequent items and consists a list of items along with corresponding transactions which contains those items, i.e., $\{x\ (Ti\ )|x \in Ti\ \}$, where x is an itemset and Ti is the set of Transaction IDs. The transactional data are charged into HDFS, which permits better use of cluster memory and ameliorates fault tolerance. Then, map function is applied on every item to produce a (key, value) pair, where key is the item and value is the list of transaction IDs. The phase 1 produces only vertical singleton items. The diagram illustrating the transformation from horizontal to vertical Layout is shown in Figure 1.

After the vertical data generation, pruning step is applied on itemsets to filter out the non-frequent items. Therefrom, at the end of phase 1 only the frequent items are part of vertical dataset and all the non-frequent items are eliminated from the original horizontal input data, which reduces the data size. This revised dataset is further processed in next phase. The proposed algorithm is shown in Figure 2.

Algorithm for generating frequent items of 1-cardinality in a vertical layout

Input: Horizontal data-set
Output: Vertical data-set
1. Each Mapper operates on its local database
2. Every Mapper generates local tid-list for all items.
3. Every Mapper exchanges tid-list with all other
4. To get total number of transaction associated with every item
5. Each Reducer regroups the tid-no related with an item from all Mapper output.
6. To form global tid-list for each item.

Fig. 2 The proposed algorithm for phase 1

The phase 2: The frequent itemset of k cardinality, where $k \geq 2$ are generated by the second phase of the HFIMH. This phase is an iterative procedure, which generates k-frequent item sets in the k-th pass of iteration. The vertical dataset, which is generated in the last pass, resolves the problem of scanning the entire dataset to compute the support of itemsets.

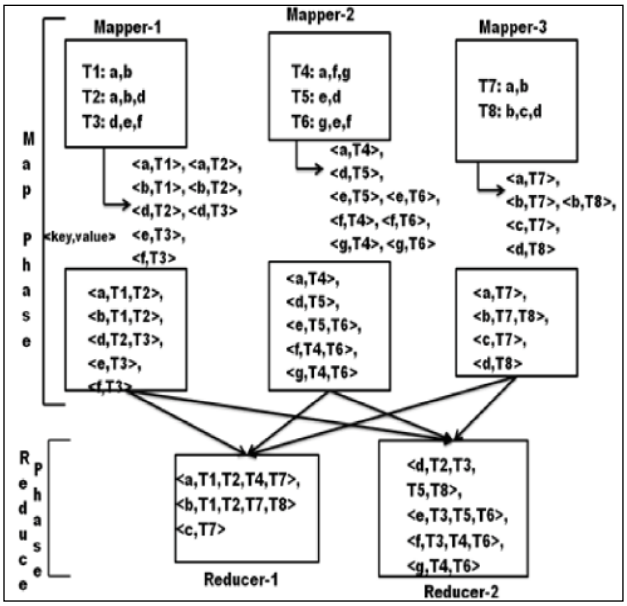

Fig. 1 Diagram illustrating Horizontal to Vertical Data Layout Conversion in MapReduce.

The vertical data are in charge of keeping enough information to compute the support for each itemset and generate all potential candidates. All the executors of cluster used shares the vertical data, after the first phase the vertical base contains the frequent 1-itemset. This data have smaller sizes than the actual horizontal data.

Thus, to reduce the cost of I/O and required disk space, the vertical dataset should necessarily be scanned. The revised horizontal data from phase 1 are distributed among all the executors in order to make the algorithm workable in parallel fashion. However, the fact that HFIMH generates the candidate set from each transaction instead of generating all possible candidate itemsets causes the reduction of the number of candidate sets in each iteration. The revised horizontal data are distributed over cluster nodes, items are separated from each transaction and an item list is prepared in each transaction. We store the vertical data in distributed cache in hadoop to be shared by all the executors. This shared vertical data is scanned and used to count the support for each candidate itemsets. After each iteration k, this dataset is updated by adding the new k -frequent itemsets in the k-th pass with their lists of TIDs to get ready for next pass.

All potential candidates of k-cardinality are produced in k-th pass from item list from every transaction. For each candidate itemset Ck= I1I2…Ik instead of generating all these subsets in order to obtain its support , we reduce the search by joining: Only the TDIs of two subsets Sk-1and Sk such as Sk-1 = I1I2…Ik-1 and Sk =Ik knowing that Ck=Sk-1Sk, the TIDs of the both subsets Sk-1 and Sk are retrieved from the shared vertical data, and all the common transactions IDs of the subsets are stored in a list common.

A set of elements is frequent, if the length of the common list is not less than min-sup. All frequent Item sets are accumulated in the form (key, value) where the key is a set elements and the value is the list of transaction IDs to add them to the shared vertical database to prepare for following passage (k + 1). The proposed algorithm is shown in Figure 3.

Algorithm for generating frequent k-itemset

Input: horizontal dataset with only frequent items
min_sup=Minimum Support Threshold
K≥ 2: Pass
Output: frequent_itemset: list of frequent itemsets

1. foreach transaction t in Revised_horizontal_data
2. map(trans_ID, t)
3. foreach item I in t
4. item_list= { $I_1, I_2, \ldots I_j$}
5. end foreach
6. end map
7. end foreach
8. foreach item_list in t
9. combinations= CombinationGenerator(item_list,K)
10. foreach combination C in combinations
11. $S_{k-1} = \{I_1 I_2 \ldots I_{k-1}\}, S_k = \{ I_k \}$
12. subset ={S_{k-1}, S_k}
13. common=null
14. foreach itemset IS in subset
15. Search the itemset IS in Vertical_data
16. get the corresponding list_of_transIDs
17. If(common==null) then
18. common=list_of_transIDs
19. end if
20. common=common intersect list_of_transIDs
21. end foreach
22. if(length(common)≥min_sup) then
23. frequent_itemset= frequent_itemset List(C, common)
24. end if
25. end foreach
26. end foreach
27. Update the Vertical_data with new frequent_itemset for the next pass

Fig. 3 The proposed algorithm for phase 2

4. Complexity analysis

Let, D be the dataset containing total k items in n transactions and m is the number of items in largest transaction. All the transactions are scanned once in first phase and each element is represented in a vertical layout, which takes O (n × m) time in worst case. All the items go through the pruning stage that will scan all k items, which will need maximum O(k) time. Thus, the time complexity of phase 1 is measured by O(n × m + k).

Phase 2 of HFIMH is an iterative procedure; hence, we find the time complexity for i th iteration/pass. After Phase 1, the original dataset is revised so that only the frequent elements that remains. Suppose that revised horizontal data from phase 2 consists total K items, N transactions and M be the length of largest transaction, such that $K \leq k$, $N \leq n$ and $M \leq m$.

All the transactions from revised horizontal data are scanned, and an item list is created, which requires O(N) time. then, all potential candidates are generated from the items of every item list, which requires O(MCi) time equal to $O(M \min\{i, M-i\})$ since after each pass, the value of i approaches of M. We access each candidate and create a list of only two subsets. The time complexity to create subsets is equal to O(2).We suppose that vertical data contains v itemset and c maximum count of transaction IDs for an itemset, where $c \leq M$. All subsets needs to scan all the itemsets from vertical data and find the common transaction IDs, which requires maximum $O(v \times c)$ time. All the candidates are pruned by comparing the length of common to minsup, which will take O(1) time. Therefore, the upper bound for time required for phase 2 will be $O(N \times M \min\{i, M-i\} \times 2 \times v \times c)$. Total complexity of HFIMH is the sum of complexity of phase 1 and phase 2.

5. Experimental and Result Analysis

To execute the HFIMH in distributed environment, a Hadoop cluster of variable length is utilized, where every node has Intel® Core ™ i5- 3230M CPU@2.60GHz processing units and 6.00GB RAM with Ubuntu 12.04 and Hadoop 2.2.0, HDFS was utilized for storage of input dataset and output frequent itemsets. T10I4D100K and Pumsb act as experimental data. They can be downloaded at http://fimi.ua.ac.be/dataJ, the first is small and the second is large.

In the experiment, comparing the run time of our algorithm with the MR-Apriori algorithm [18]. Here, running time signifies the total execution time between input and output, and both the algorithms are implemented on Hadoop platform.In the above Figure 4, the vertical axis indicates run time in seconds and the horizontal axis indicates the different size of datasets. It can be seen that when the dataset is small (T10I4D100K), runtime of both the algorithm is near about the same. But, when dealing with large dataset (Pumbs), runtime of HFIMH algorithm will become shorter as compared to MR-Apriori.

In our method, we have utilized vertical database layout and set theory of Intersection, which simplified the steps, as frequent 2-itemsets to k-itemsets are produced in a single step from 1-itemsets. The principal advantage of the above system is that it dercreases the overall time spent in finding support of the candidate item set. The intersection algorithm permits us to calculate the support by simply counting the common transactions in every element of candidate sets. The time elapsed in the scanning of the database is reduced significantly. Also, number of candidates are reduced in HFIMH, which ameliorates the running time. This demonstrates that HFIMH gives better performance and can be utilized efficiently for the treatment of large datasets.

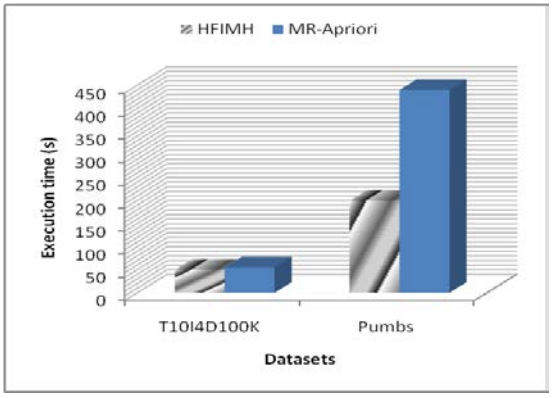

Fig. 4 Comparison of the runtime

The following experiment evaluates the scalability of HFIMH, which is also measured by the running time .The dataset T10I4D100K is utilized here. The experiment is realized on condition that the number of cluster computer nodes ranges from 2 to 8 while the support degree remains to be 0.5%.

In Figure 5, x-axis shows the number of computer nodes of Haddop cluster and y-axis indicates the running time of HFIMH algorithm. Figure 5 shows the running time with various numbers of computer nodes.With more computer nodes, HFIMH needs less execution time, and the curve of HFIMH has a nearly linear decline. HFIMH demonstrates a characteristic of near-linear scalability.

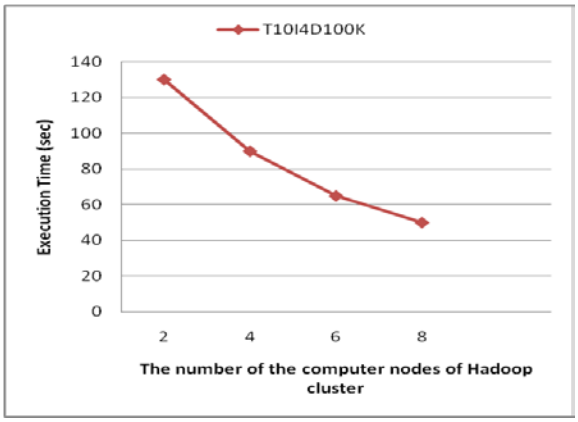

Fig. 5 The running time with different computer nodes

6.Conclusion

As one of the traditional mining algorithms, Apriori helps us to find frequent patterns from transactional dataset. As proposed in this paper, the hybrid frequent itemset mining algorithm (HFIMH) use both vertical and horizontal layout of data to solve the mentioned defiances in the Apriori Algorithm. In fact, HFIMH is a double-phase procedure that functions effectively in distributed environment.
However, for more balanced workload, further research is needed to identify a way of loading the input data to the nodes. More than that, to permit even better execution times, a hybrid implementation can be achieved between in-memory and Hadoop implementations.

References

[1] I.A.T. Hashem, I. Yaqoob, NB. Anuar, S. Mokhtar, A. Gani, and S.U. Khan, "The rise of big data on cloud computing," Information Systems, Vol.47, pp.98–115, 2015.

[2] C.L. Philip, and C.Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on big data," Information Sciences, Vol.275. pp.314–347, 2014.

[3] J. Han, M. Kamber, and J. Pei, "Data mining: concepts and techniques (3th ed)," A volume in The Morgan Kaufmann Series in Data Management Systems. New York, Elsevier, 2011.

[4] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," In Proceedings of the 20th Very Large Data Bases Conference .Santiago, Chile, pp.487–499, 1994.

[5] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation, " In Proceedings of the international conference on Management of data. Dallas, Texas, USA, Vol.29, No.2, pp.1-12, 2000.

[6] M.J. Zaki, S. Parthasarathy, and M. Ogihara, "Parallel algorithms for discovery of association rules," Data Mining and Knowledge Discovery, Vol.1, No.4, pp.343-373, 1997.

[7] Apache Hadoop. http://hadoop.apache.org.

[8] V. Umarani, and M. Punithavalli, "A Study on Effective Mining of Association Rules From Huge Databases," International Journal of Computer Science and Research, Vol.1, No.1, pp. 30-34, 2010.

[9] R. Agrawal, and J.C. Shafer, "Parallel mining of association rules," in IEEE Transactions on Knowledge and Data Engineering, Vol.8, No.6, pp.962-969, 1996.

[10] S. Ghemawat, H. Gobioff, H, and S.T. Leung, ,"The Google file system," In Proceedings of the nineteenth Symposium on Operating Systems Principles, Vol.37, No.5, pp. 29–43, 2003.

[11] Yahoo developer network. Hadoop tutorial. https://developer.yahoo.com/hadoop/tutorial/

[12] J. Dean, and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the 50th anniversary issue , Vol.51, No.1, pp.107–113, 2008.

[13] R. Saritha, and M.U. Rani, "Mining frequent item sets using MapReduce paradigm," International Journal of Engineering Sciences Research , Vol.4, No.1, pp.1293-1297, 2013.

[14] M.J. Zaki, "Parallel and distributed association mining: A survey," In IEEE concurrency , Vol.7, No.4, pp.14– 25, 1999.

[15] N. Li, L. Zeng, Q. He, and Z. Shi, "Parallel Implementation of Apriori Algorithm Based on MapReduce," In Proceedings of the 2012 13th ACIS International Conference Software Engineering, Artificial Intelligence Networking and Parallel/Distributed Computing. Kyoto, JAPAN, pp. 236–241, 2012.

[16] Yu. Kun-Ming, L. Ming-Gong, H. Yuan-Shao, and C. Shi-Xuan, "An efficient Frequent Patterns Mining Algorithm based on MapReduce Framework," In International Conference Software Intelligence Technologies and Applications. Hsinchu, TAIWAN, pp.1-5, 2014.

[17] S. Moens, E. Aksehirli, and B. Goethals, " Frequent Itemset Mining for Big Data," In IEEE International Conference Big Data. Silicon Valley, CA, USA, pp.111–118, 2013.

[18] X. Lin, "MR-Apriori: Association Rules Algorithm Based on MapReduce", In: 5th IEEE International Conference on Software Engineering and Service Science, Beijing, China, pp.141-144, 2014.

[19] X.Y. Yang, Z. Liu, and Y. Fu, "MapReduce as a programming model for association rules algorithm on Hadoop," In IEEE 3rd International Conference on Information Sciences and Interaction Sciences. Chengdu, China, pp.99-102, 2010.

[20] R. Dharavath, V. Kuman, C. Kuman, and A. Kuman, "An Apriori-Based Vertical Fragmentation Technique for Heterogeneous Distributed Database Transactions," In Proceeedings of the international conference on Advanced Computing, Networking, and Informatics, Kolkata, India, pp.687–695, 2014.

[21] S. Dhanya, M. Vysaakan, and A. Mahesh, "An enhancement of the MapReduce Apriori algorithm using vertical data layout and set theory concept of intersection," Intelligent Systems Technologies and Application, Vol.385, pp.225–233, 2016.

[22] A, Imran, and P. Ranjan, "Improved Apriori Algorithm Using Power Set on Hadoop", In Proceedings of the First International Conference on Computational Intelligence and Informatics. Hyderabad, India, pp.245-254, 2017.

[23] W. Mao, and W. Guo, "An improved association rules mining algorithm based on power set and Hadoop," International Conference on Information Science and Cloud Computing Companion. Guangzhou, China, pp.236–241, 2013.

[24] M.Ibrahim, M.H.Maghny , and M.A.Abdelaziz, " Fast Vertical Mining Using Boolean Algebra," International Journal of Advanced Computer Science and Applications, Vol.6, No.1, pp.89-96, 2015.

[25] K. Geetha, and S.K. Mohiddin, "An Efficient Data Mining Technique for Generating Frequent Itemsets," International Journal of Advanced Research in Computer Science and Software engineering, Vol.3, No.4, pp.571-575, 2013.

Yassir Rochd Mathematics and Computer Science teacher in High School .PhD student at the Laboratory of Process Engineering and Optimization of Industrial Systems (IPOSI) of the doctoral studies center of Hassan I University Settat, Morocco. His research interests are big data , data mining and text mining.

Imad Hafidi PhD is a professor (since 2009) at the national school of applied sciences Khouribga. He gets accreditation to supervise research in 2013. Before coming to Khouribga, he completed many programs: PhD degree (2005) in Applied Mathematics at the National institute of applied sciences Lyon (INSA), and the MS degree (2008) in Software Engineering at the School of Mines Saint-Etienne. His research focuses Text Mining, Data Integration and Big Data.