A MapReduce-Based Association Rule Mining Using Hadoop Cluster—An Application of Disease Analysis

Namrata Bhattacharya, Sudip Mondal and Sunirmal Khatua

Abstract With the expansion and future potential in the field of healthcare industry, it is necessary to analyze a large amount of noisy data to obtain meaningful knowledge. Data mining techniques can be applied to remove inconsistent data and extract significant patterns. Association rule mining is a rule-based method which uncovers how items are associated with each other. Apriori algorithm is a broadly used algorithm for mining frequent itemsets for association rule mining. However, the performance of the apriori algorithm degrades with the large volume of data. So a parallel and distributed algorithm is required for efficient mining. In this paper, we provide an efficient implementation of the apriori algorithm in Hadoop MapReduce framework. We have considered medical data to produce rules which could be used to find the association between disease and their symptoms. These rules can be used for knowledge discovery to provide guidelines for the healthcare industry.

Keywords Association rule mining · Apriori algorithm · Hadoop MapReduce · Big data · Distributed computing

1 Introduction

Data mining [1], also known as Knowledge Discovery in Databases(KDD), is a process of extracting a meaningful pattern from massive transaction databases and other repositories. Popular data mining techniques are association rule mining [2], classification [3], sequential pattern analysis [4], data visualization [5], etc.

N. Bhattacharya (✉) · S. Mondal · S. Khatua
Department of Computer Science and Engineering, University of Calcutta, Kolkata, India
e-mail: namrata.bhattacharya03@gmail.com

S. Mondal
e-mail: sudip.wbsu@gmail.com

S. Khatua
e-mail: skhatuacomp@caluniv.ac.in

H. S. Saini et al. (eds.), *Innovations in Computer Science and Engineering*, Lecture Notes in Networks and Systems 74,
https://doi.org/10.1007/978-981-13-7082-3_61

Agrawal and Srikant proposed apriori algorithm in the year 1994. Apriori algorithm is designed to operate on transactional databases (for example, list of symptoms caused by a particular disease among patients) for generating association rules. The rapid advancement of information technology has resulted in the accumulation of a large amount of data for organizations. Thus, data growing beyond the few hundreds of terabytes become unfeasible to manage, store, and analyze on a single sequential machine. The main limitation of apriori algorithm is the repeated scan of dataset and wastage of computing resources to store a large number of temporary candidate sets. Moreover, single processor system with limited computing resources is insufficient, which makes the algorithm performance inefficient. So improvements are needed in order to reduce the time complexity. The solution can be achieved by Hadoop MapReduce model for parallel and distributed computing that readily and efficiently process massive datasets [6] on large clusters of commodity hardware in a fault tolerance manner.

The medical data collected from healthcare unit are used to identify the nature of the disease from the list of possible diagnoses by examining the symptoms. This method is known as the differential diagnosis. The rules generated by association rule mining are helpful in identifying the symptoms corresponding to a disease [7]. Although they may not list all the symptoms, they can be used to narrow down some representative symptoms which can be used to diagnose the disease and select the initial treatment.

The paper is structured as follows. Section 2 briefly describes the association rule mining, apriori algorithm, and Hadoop MapReduce programming model. Section 3 briefly discusses the proposed MapReduce framework and Sect. 4 analyzes result. Finally, Sect. 5 presents conclusion and future scope.

2 Background

2.1 Association Rule Mining

In data mining, association rules are useful for analyzing frequent itemset data and using the criteria of support and confidence to find the recurring relationships among data called associations. Association rules are implication expressions of the form $\{A \rightarrow B\}$ where the intersection of set A and set B is a null set.

Support provides an estimate of the probability of occurrence of an event, i.e., $P(X \cup Y)$. The *support* of an itemset {X, Y} is given by Eq. 1. If the *support* of an itemset is greater than or equal to the *minimum_support*, then that itemset is added to the set of frequent itemsets.

$$\textbf{Support(X, Y)} = \frac{No\ of\ transactions\ that\ contain\ X\ and\ Y}{No\ of\ transactions\ in\ the\ database} \tag{1}$$

The *confidence* of rule {X → Y} indicates the probability of both antecedent and consequent appearing in the same transaction. *Confidence* is given by Eq. 2. If the *confidence* of the inference made by a rule is greater than or equal to the *minimum_confidence*, then that rule is added to the set of association rules [8, 9].

$$\textbf{Confidence(X} \longrightarrow \textbf{Y)} = \frac{Support(X \cup Y)}{Support(X)} \tag{2}$$

Among many applications of association rule mining, one of the major applications is disease analysis, which includes the mapping of illness to their symptoms [2]. For example, a patient who suffers from diabetes is somehow likely to suffer hypertension as well. Association rule mining is used to discover implications such as $\{Diabetes\} \rightarrow \{hypertension\}$. More formally, association rules are of the form given in Eq. 3.

$$\{Diabetes\} \longrightarrow \{Hypertension\} \quad [Support = 3\%, Confidence = 75\%] \tag{3}$$

In Eq. 3, *support* of 3% means that diabetes and hypertension occurred together in 3% of all the transactions contained in the database, while *confidence* of 75% means that the patients who suffer from diabetes, 75% of the times, suffer from hypertension as well.

2.2 Apriori Algorithm

The apriori algorithm proposed by Agrawal and Srikant in 1994 is one of the influential data mining algorithms which is used for mining the frequent itemsets from a given dataset containing a huge number of transactions.

Let $T = \{T_1, T_2, T_3 \ldots T_m\}$ be a set of transactions and $T_i = \{I_1, I_2, I_3 \ldots I_n\}$ be the set of items in transaction T_i for $1 \leq i \leq m$. Let $I = \{I_1, I_2, I_3 \ldots I_k\}, 0 \leq k \leq n$ be a subset of items formed by the items in a transaction that are obtained from the transactional database. If an itemset consists of k items, then this itemset is termed as k-itemset [1]. Apriori algorithm uses a level-wise frequent pattern mining where k-itemsets are used to explore (k + 1)-itemsets for mining association rules.

Apriori algorithm is based on apriori principle which states that "if an itemset is frequent, then all of its subsets must also be frequent." [1].

$$\forall X, Y : (X \subseteq Y) \implies \text{Support}(X) \geq \text{Support}(Y)$$

This principle holds true because of the anti-monotone property of *support*.

2.3 Hadoop MapReduce Framework

Hadoop is a Java-based open source and powerful tool that uses MapReduce architecture for store, design, and analysis of very large datasets like Google file system. MapReduce programming model is applied for distributed storage and processing of a large dataset. The distributed architecture consists of computer clusters built from commodity hardware. The fundamental part of Apache Hadoop is Hadoop Distributed File System (HDFS) and MapReduce programming model [10].

Google developed a MapReduce implementation that extends to large clusters of machines comprising thousands of machines. It uses MapReduce for generating data for many areas like web search service, sorting, data mining, machine learning, and many other systems [11]. The evolution of huge data in health care has brought a lot of challenges in terms of data transfer, storage, computation, and analysis. So big data can be implemented in different dimensions of research as well as developments in biomedical and health informatics, bioinformatics, sensing, and imaging will help in future clinical research. Another important factor is study and analysis of health data will contribute to the success of big data in medicine [12].

3 Distributed Apriori Algorithm for Association Rule Mining

3.1 Problem Formulation

Apriori algorithm suffers some major challenges despite being clear and simple. The major limitation in mining frequent itemsets from large dataset is costly wasting of time to hold the huge number of candidate itemsets and frequent itemsets satisfying the *minimum_support* [1]. Long itemset will consist of a large number of combinatorics of smaller frequent itemset. For instance, a frequent itemset of length 100 will contain 100 frequent 1-itemset. Thus, the total number of frequent itemsets it contains is given in Eq. 4.

$$^{100}C_1 + {}^{100}C_2 + \cdots + {}^{100}C_{100} \approx 1.27 \times 10^{30} \tag{4}$$

Thus, if a computer takes 1 millisecond to compute each itemset, then it will take approximately 1.27×10^{30} millisecond which is huge. Also, if there are 10^4 frequent 1-itemsets, it may generate more than 10^7 candidate itemsets; thus, it will scan the database many times repeatedly [1]. In Sect. 3.2, we have discussed a Hadoop MapReduce-based solution for association rule mining.

3.2 *Proposed MapReduce Framework for Association Rule Mining*

This section will address the distributed association rule mining algorithm that has been used to generate the association rules. The paper performs an efficient data mining to group the frequently occurring symptoms for a specific disease as core symptoms and identify the significant association between them.

A MapReduce job splits the input transaction database into various blocks, and a mapper is invoked once for each transaction passed as arguments. The map task parses one transaction at a time and extracts each itemset included in the transaction it received as input. After processing, the mapper sends the itemset to the partitioner by emitting the itemset and frequency as <key, value> pair, where 'key 'is a candidate itemset and "value" is 1.

The partition task collects all the intermediate <key,value> pair emitted from the map task as its input and works like a hash function. Based on the key size of each key, i.e., the size of each itemset, the partitioner specifies that all the values for each itemset are grouped together and maps all the values of a single key go to the same reducer. The output pairs of all partitioner are shuffled and exchanged to make the list of values associated with the same key as <key,list(value)> pairs.

Reduce task collects each key passing all the values emitted against the same key as arguments, i.e., <key,list(value)> pairs emitted by partitioner task. Then, it sums up the values of respective keys. Candidate itemset whose sum of values is $supportcount \leq minimum\_support\_count$ is discarded. The result from all reducers is written to the output file. The whole process is shown in Fig. 1.

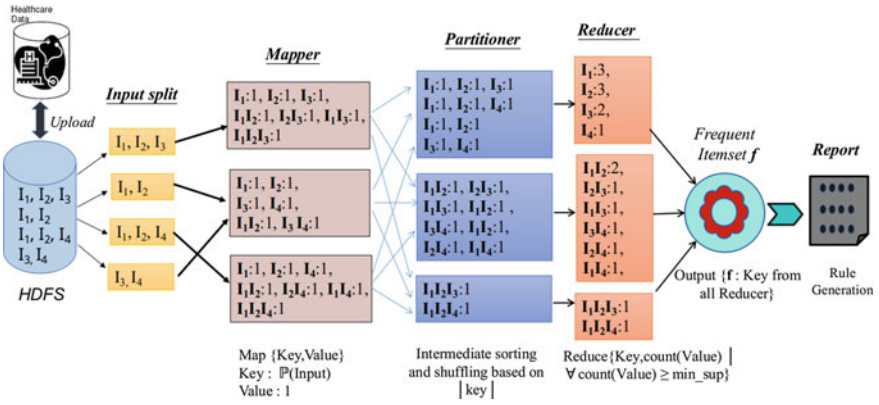

Fig. 1 Data flow in proposed MapReduce framework

4 Experiment and Results

In this section, we discuss the implementation specifications and experimental analysis performed.

4.1 Dataset and Experimental Setup

The dataset used for our experiment is a comma separated file that contains various types of disease and their symptoms. The file contains a collection of patient-wise disease-symptom transactions with disease representing a single transaction and each column in a row being the symptoms. We have generated five datasets with a varying number of transactions for the analysis. A total number of transaction in each dataset are 25,000, 50,000, 100,000, 150,000, and 200,000, and each transaction contains 10 items on average.

We have setup Hadoop single-node, 2-node, and 3-node cluster for our experiment, to compare the implementation of the distributed algorithm with centralized system. All experiments have been performed on machines that contain CentOS 7 64-bit, Eclipse Neon, and Hadoop version 2.7.1. Both the algorithms have been implemented in Java: JDK version is "1.8.0_131".

4.2 Performance Improvement of Association Rule Mining Algorithm Using Hadoop

We obtain Fig. 2 from running the algorithm, which is a graph that shows the comparison of the execution time of algorithm when implemented on non-MapReduce Java platform with the execution time of the algorithm when implemented on the Hadoop cluster and to test if there would be an improvement in the running time as the number of nodes increased.

When the algorithm is executed on the dataset which contains 200,000 transactions using the non-MapReduce code, it takes approximately 231 min. We are using the same algorithm for executing on Hadoop single-node cluster, it takes approximately 68 min, and on Hadoop multinode cluster with two and three nodes, it takes approximately 41 and 28 min, respectively. Thus, we can see in Fig. 2, 87.8% time is reduced by executing the algorithm in three-node cluster, 82.25% time in two-node cluster, and 70.5% time in single-node cluster with respect to the non-MapReduce algorithm. This is a significant improvement in the performance of the algorithm using distributed framework Hadoop. The same performance improvement using Hadoop can be witnessed for other datasets also. In our experiment, as the number of nodes increases, the execution time of the algorithm decreases drastically.

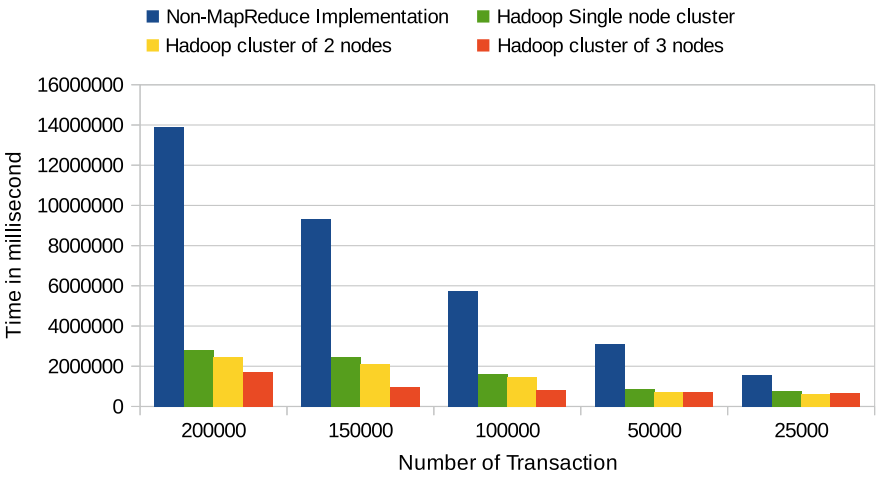

Fig. 2 Analysis of running time comparison based on number of transactions

We observe that the decrease in execution time is not always at the same rate. This depends upon the size of the data. For the dataset with 50,000 and 25,000 number of transaction, we can observe that the time required for executing the program in two-node cluster (11 and 10 min, respectively) is less than running the program in three-node cluster (12 and 11 min, respectively). This can be attributed to the fact that the input file is split into the block size of HDFS and the operations on the block are performed by the node it is assigned. If the number of nodes exceeds the number of blocks, it is a waste of resources. Thus, in some instances, increasing the number of nodes would not lead to the decrease in time. In almost every instance, a larger dataset in block size and executed on multiple nodes needs lower execution time rather than execute with a smaller dataset. It is because of Hadoop operates on larger HDFS block size to compute with its worker node.

We have obtained Fig. 3 by executing the apriori algorithm in the distributed platform, which is a graph for different itemsets that shows the *support* and *confidence* for patients. X-axis represents the diseases and Y-axis represents the value of support and confidence, respectively.

From Fig. 3, we can observe that risk of "influenza" is significant if the patient already carries "fever" since it gives association as 80% which is greater than the *minimum_confidence* value of 60%. But the converse is not always true, i.e., risk of "fever" is not significant if the patient already carries "influenza" since we can observe $\{influenza \rightarrow fever\}$ gives association as 44% which does not cross the *minimum_confidence* value.

The system can further be enhanced to find strong association rules. We can say the dataset {influenza,dry cough} contains strong association among its attributes as both the association rules, $\{influenza \rightarrow drycough\}$ and $\{drycough \rightarrow influenza\}$ range over the *minimum_confidence* value.

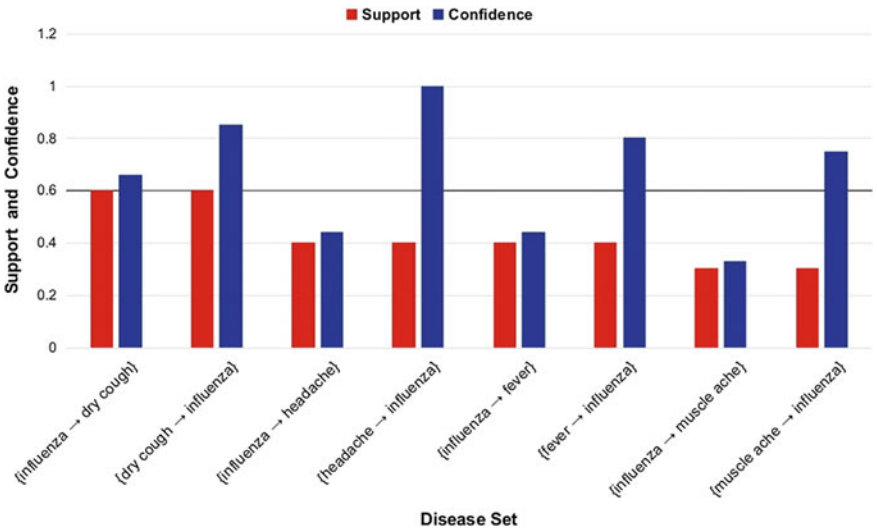

Fig. 3 Analysis of support and confidence for different itemsets

5 Conclusion and Future Scope

Efficient Apriori algorithm needs to be designed to mine large volume of data so that it can be implemented in parallel and distributed environment. The proposed model has applied data mining technique for production of association rule using Hadoop MapReduce in the distributed environment on a large clinical database. The result shows higher data processing time when the algorithm has been applied to the single centralized system over the distributed system. The proposed algorithm has developed on big data sets in the distributed system and it shows notable performance improvement. In future, we want to apply this algorithm to cloud computing system for better access and execute in real time.

References

1. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam
2. Agrawal R, Imieliski T, Swami A (1993, June) Mining association rules between sets of items in large databases. In: ACM sigmod record, vol. 22, no. 2, pp 207–216. ACM
3. Chen G, Liu H, Yu L, Wei Q, Zhang X (2006) A new approach to classification based on association rule mining. Decis Support Syst 42(2):674–689
4. Koper A, Nguyen HS (2011, December) Sequential pattern mining from stream data. In: International conference on advanced data mining and applications, pp 278–291. Springer, Berlin
5. Keim DA (2002) Information visualization and visual data mining. IEEE Trans Vis Comput Graph 1:1–8

6. Yang HC, Dasdan A, Hsiao RL, Parker DS (2007, June). Map-reduce-merge: simplified relational data processing on large clusters. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data, pp 1029–1040. ACM
7. Han J, Fu Y, Wang W, Koperski K, Zaiane O (1996, June) DMQL: a data mining query language for relational databases. In: Proceedings of 1996 SiGMOD, vol 96, pp 27–34
8. Agrawal R, Srikant R (1994, September) Fast algorithms for mining association rules. In: Proceedings of 20th international conference on very large data bases, VLDB, vol 1215, pp 487–499
9. Fan W, Bifet A (2013) Mining big data: current status, and forecast to the future. ACM sIGKDD Explor Newsl 14(2):1–5
10. Shvachko, K, Kuang H, Radia S, Chansler R (2010, May) The hadoop distributed file system. In: 2010 IEEE 26th symposium on mass storage systems and technologies (MSST), pp 1–10. IEEE
11. Han J, Fu Y (1999) Mining multiple-level association rules in large databases. IEEE Trans Knowl Data Eng 5:798–805
12. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113