International Journal of Advances in Computer Science & Its Applications– IJCSIA
Volume 5 : Issue 2 [ISSN : 2250-3765]
Publication Date: 30 October, 2015

Mining Infrequent Weighted Itemsets from Very Large Data based on MapReduce Model

T Ramakrishnudu and R B V Subramanyam

***Abstract*— Mining frequent and infrequent itemsets from a given dataset is the most significant area of data mining. When we mine both frequent and infrequent itemsets simultaneously, infrequent itemsets become very important because there are many useful negative association rules in them. Infrequent weighted Itemset mining is the process of mining infrequent items from weighted dataset. Many of the weighted Itemset mining algorithms scan the dataset many times. When the dataset size is very large, both memory usage and computational cost of mining algorithm is very expensive. In addition, single machine memory and other resources are insufficient to handle very large weighted datasets. Parallel and distributed computing is the alternative solution for these types of problems. In this paper we proposed infrequent weighted Itemset method on Hadoop-MapReduce framework, which can handle huge datasets. Experiments are performed on 8 node cluster with a synthetic dataset. The experimental results show that the proposed method is very efficient in handling very large datasets.**

***Keywords*— data mining, association rule, frequent Itemset, Infrequent Itemset, Weighted Itemset, Hadoop, MapReduce.**

I. Introduction

Knowledge Discovery in Database (KDD) [13] is the process of extracting interesting, previously unknown and potentially useful patterns from the large repositories. Frequent Itemset Mining (FIM) or Association Rule Mining (ARM) is a data mining task [13]. Frequent Itemset is actionable if its support count is greater than or equal to a user-specified threshold, called a *minimum support (ms)*, whereas Infrequent Itemset support count is below the *minimum support (ms)*. Association Rule Mining discovers associations among items in a transactional database [1].

Frequent Itemset Mining has been extensively studied in the literature since Agrawal et al. first introduced it in [1], [2]. Abundant effort has been devoted and methods proposed for efficiently discovering association rules [1],[2],[3],[4]. Association rules offer a useful and effective way to identify and represent certain dependencies between items in a database.

In recent years, there has been an increasing demand for infrequent Itemset mining. For instance, in [7, 8, 9, 12] algorithms for discovering infrequent itemsets have been proposed. However, many of the frequent and infrequent Itemset mining algorithms ignore the interest of an item in the given dataset. Thus, the actual importance of an Itemset is not possible to identify using the above approaches. By giving different importance to different items in the same transaction, few new models proposed in [11-17]. However, traditional frequent and infrequent weighted Itemset mining algorithms still suffering from the scalability, especially if the data size is very large.

When the dataset size is very large, both memory usage and computational cost of mining algorithm is very expensive. In addition, single machine memory and other resources are insufficient to handle very large weighted datasets.

Additionally, because of exponential growth of the data, the organizations have to deal with continually growing amount of data. As these data grow past hundreds of gigabytes towards terabytes or more, it becomes nearly unimaginable to mine them on a single machine. The solution for the above problem is the distributed computing.

Parallel and distributed computing is the alternative solution for these types of problems. Distributed data mining algorithms attempts to decompose the mining problem into sub- problems and solves the sub-problems using homogeneous machines such that each node works independently and concurrently. Although the distributed data mining improve the performance, but raises quite a few issues like dividing the input data, load balancing, communication cost between the working nodes and identifying the failure of nodes. To overcome the above problems Jeffery Dean et al [18, 19] introduced a new programming paradigm called MapReduce. MapReduce, as a simplified distributed programming model developed by Google [18, 19], is more appropriate for large data processing applications. It has been widely used in the tasks of data mining, machine learning and search engines etc.

In the MapReduce programming [18, 19], a distributed file system initially divide the input file and the data represented as <key, value> pairs. All computations are carried out by two functions called *Map* and *Reduce*. Both the functions *Map* and *Reduce* take <*key*, *value*> pair as an input and produce the same pair as an output. The *Map* function accepts an input pair and return intermediate <*key*, *value*> pair as an output. The *Reduce* function accepts an *intermediate key* and the set of *values* associated with that *key* as an input. It combines these values to form a possible small set of values. The *reduce* function write output to a distributed file in the Distributed File System (DFS).

Some of the researchers made an effort towards Frequent Itemset Mining (FIM) and the association rule mining (ARM) using MapReduce programming model [20-25] on transactional data. And few methods [26-27] deal with different kind of data. All the existing methods focus on

T Ramakrishnudu, R B V Subramanyam
Department of CSE, National Institute of Technology, Warangal
India

frequent Itemsets mining. In this paper, we focused on mining infrequent weighted Itemset from large weighted data using MapReduce framework.

The rest of this paper is organized as follows. Section 2 introduces the basic concepts and preliminary definitions. In Section 3, the existing methods are reviewed. The proposed method is presented in Section 4. Experimental results are given in Section 5. The concluding remarks are finally made in Section 6.

II. Concepts and Definitions

A. Basic Concepts

Let $I= \{i_1, i_2, i_3...i_n\}$ be a finite set of items and $W= \{w_1, w2, ..., w_n\}$ be a set of weights, these weights are non-negative numbers. The weighted item is defined as pair $<X, w(X)>$, where $X \in I$ is an item and $w(X)$ is the weight associated with X.

Definition 1: A weighted transaction T_w is a set of weighted items $<X, w(X)>$

Definition 2: A weighted transactional dataset D_w is a set of weighted transactions T_w.

$$D_w=\{T_{w1}, T_{w2}, \ldots, T_{wn}\}. \quad (1)$$

Definition 3 Item weight [12]: Item weight is a value associated with an item representing its significance. It is denoted as *w(item)*.

Definition 4 Weighting function [17]: Let T_w, be a weighted transaction, I_w (T_w) be a Set of weighted items in T_w, $T_w \in D_w$, the weighting function $W_f(I_w)$= aggregation of its item weights in that transaction T_w.

Definition 5 IWI-Support: Let I_w be a weighted Itemset, D_w be a weighted transactional dataset, I_w (T_w) be a set of items in T_w, $T_w \in D_w$, W_f be a weighting function, and the *IWI-Support* [17] is as shown below:

$$IWI - Support(Iw, Dw) = \sum_{Tw \in Dw | Iw \subseteq IS(Tw)} Wf(Iw, Tw) \quad (2)$$

Definition 6: A Weighted Frequent Itemset is one, its *IWI-Support* is greater than or equal to the user specified *weighted minimum support (wms)* value, then the item is called frequent weighted Itemset otherwise infrequent weighted Itemset.

B. Hadoop Framework

Hadoop framework is allows the distributed processing of large datasets across cluster of machines using simple programming models [28]. Hadoop is the parallel programming platform built on Hadoop Distributed File Systems (HDFS) for MapReduce computation. The HDFS is the distributed file system designed to run on commodity hardware. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large datasets. HDFS was originally built as infrastructure for the Apache web search engine project. HDFS is a part of Apache Hadoop main project [28].

C. MapReduce

MapReduce is a programming model and an associated implementation for processing and generating large datasets. Users specify a *map* and *reduce* functions, they takes *<key, value>* pair as an input and generates intermediate *<key, value>* pairs and merges all intermediate values associated with the same intermediate key respectively. Programs written in this paradigm are automatically parallelized and executed on a large cluster of commodity machines [18][191].

III. Related Work

Several methods have been proposed for mining frequent Itemsets based on MapReduce framework. But no method have been proposed for mining weighted frequent and weighted infrequent Itemsets based on MapReduce framework. Few existing methods which are used to mine frequent Itemsets using MapReduce framework are listed below.

In [20] the authors are proposed a one pass method. The algorithm needs only one scan (MapReduce job) to find all frequent k-itemsets. Initially, splitting will take place and after that each mapper will apply Apriori on that split and it will generate all length Itemsets. It produces output as Itemsets as <key, 1>. The reduce will take output of mapper and sum all values for particular keys, then prune infrequent Itemsets and finally generate all frequent Itemsets.

In *k-phase* algorithms [21-23] *k* scans (MapReduce jobs) are needed to find *k-frequent (k length)* items. These methods uses two different map functions: one for the first phase and one for rest of the phases. In the first phase the mapper function outputs *<item, 1>* pair's for each item contained in the transaction. The reducer collects all the support counts of an item and outputs the *<item, count>* pairs as a frequent *1-Itemset* to the L_1, when the count is greater than the minimum support count. Next the *k-Itemsets* are passed as an input to the mapper function and the mapper outputs *<item, 1>*, then the reducer collects all the support counts of an item and outputs the *<item, count>* pairs as a frequent *k-Itemset* to the L_k.

Othman Yahya et al [25], proposed a two-phase algorithm on Hadoop-MapReduce, which is more efficient than the previous one-phase and k-phase methods. It takes only two MapReduce phases to find all frequent *k-Itemsets*. In phase1, each input split is assigned a map task (executed by map worker) that calls a map function to process this split. The mapper function uses Apriori with the partial minimum support count; which is equal to the number of transactions in the split multiply by the minimum support threshold.

The mappers output is a list of intermediate *<key, value>* pairs grouped by the key via combiner, and stored in the map

worker where the key is an element of partial frequent *k-Itemsets* and the value is its partial count. When all map tasks are finished, the reduce task is started. The mappers output are shuffled to the reduce worker that calls a reduce function. The output of reduce function is a list (*Lp*) of <*key, value*> pairs, where the key is an element of partial frequent *k-Itemsets* and the value equal one, stored in HDFS.

In phase two, one extra input is added to the data flow of the previous phase, which is a file that contains all partial frequent *k-Itemsets*. The map function of this phase counts occurrence of each element of partial frequent *k-Itemset* in the split and outputs a list of <*key, value*> pairs, where the key is an element of partial frequent *k-Itemset* and the value is the total occurrence of this key in the split. The reduce function outputs a list (*Lg*) of <*key,value*> pairs, where the key is an element of global frequent k-itemsets and the value is its occurrence in the whole dataset. The main drawback of this method is the large number of partial frequent itemsets may overload the map functions of the phase-II.

Zahara Farzanaryar et al [26] proposed a method based on insignificant Itemset property, and it deals with social network data. It improves the method proposed in [25].

In [27] authors proposed a scalable and distributable binomial method, it deals with different type of data. It converts the input data into binomial format to take benefit of MapReduce method structures, and then mine association rules from that data. In this a layered approach is used to mine frequent Itemsets from the binomial data.

IV. Problem Description and Proposed Method

Most of the algorithms proposed for mining frequent Itemsets using Hadoop MapReduce, but no algorithm was proposed for mining infrequent Itemsets using Hadoop MapReduce. Infrequent itemsets become very important because there are many useful negative association rules in them. As mentioned in section I, many of the existing methods are suffering from scalability and complexity. Hence, in this paper we designed a method to mine weighted infrequent Itemsets from large data using Hadoop MapReduce framework.

Problem Statement: Given a weighted transactional database D_w and user-defined *weighted minimum support (wms)* value, the problem is to find infrequent weighted Itemsets using Hadoop MapReduce framework.

The proposed method based on MapReduce model to find infrequent weighted Itemsets is given below.

Method:
Step1: Scan the input dataset and Divide the input dataset into number of chunks and assign one chunk to each node.

Step2: The *Map* functions at each node:

a. Scans each transaction of the input data subset and generate local candidate Itemsets (all possible subsets of the transaction).
b. Calculate IWI-Support for all local candidate Itemsets. And generate and output intermediate <*Key, Value*>, defined as <*Itemset, IWI-Support*>.

Step3: The *Reducer* functions accept <*Itemset, IWI-Support*> as input:

a. Calculate IWI-support of the global candidate Itemset.
b. if IWI-Support(I) < User defined minimum threshold value, then assign to output list *L*.
else discard I.
c. Output the <*I, IWI-Support*>.

The detailed flow diagram of the method is shown in Figure.1. The pseudo code of the mapper and reducer are shown in the Figure.2 and Figure.3.

The mapper function accepts one input called input split, and the reducer function accepts two different inputs called intermediate values and weighted minimum support given by the user.

Map()
Input: Split-S_i;

Output: <key1, value1>; key: infrequent weighted k-Itemset of the split Si; value: local IWI-support.

Begin
1. *For each weighted transaction T_w in S_i.*
2. *For each weighted Itemset I_w in t_w. /* I_w is all possible subsets of t_w.*/*
3. *wsupp=Cal_IWI-support(I_w) /* Calculate IWI-Support of I_w in the transaction (T_w) */*
4. *output(I_w,wsupp)*
5. *End for*
6. *End for*

End;

Algorithm for Map

Reduce()
Input: <key1, value1>; key1: local weighted candidate Itemset of the split Si; value: local IWI-Support, weighted minimum support (wms).

Output: <key2, value2>; key2 is infrequent weighted Itemsets; value2 is global IWI-Support.

Begin
1. *Foreach I_w in key1 do*
2. *val2= Cal_global IWI-support(I_w) /* Calculate IWI-Support of I_w in the entire dataset */*
3. *If(val2< wms)then*
4. *output(I_w , val2)*
5. *End if*
6. *End for*

End

Algorithm for Reduce

v. Experimental Results

In this section we evaluate the performance of the proposed method running on cluster of nodes. To evaluate the performance of our method we formed few clusters with different size. All the experiments were conducted in a Hadoop 2.2.0 cluster where each node contains 2.20 GHz processors with 4 GB RAM, and a 500 GB hard disk and 2 a gigabyte Ethernet link.

We used a synthetic dataset in our experiments. It is a weighted transactional dataset. It consists 200 distinct items and the average size of the transaction is 100. The weights of items are assigned randomly from 0 to 100. Example weighted transactional dataset is shown in Table 1.

TABLE I. WEIGHTED TRANSACTIONA DATASET

Transaction ID	Weighted Items
Tid1	(i1,45) (i2,60) (i3,5) (i4,20)
Tid2	(i1,30) (i2,10) (i3,1) (i4,100) (i5,90)
Tid3	(i3,40) (i4,70)
Tid4	(i1,5) (i2,40) (i3,80)
Tid5	(i2,20) (i3,0)

We test our approach to find weighted infrequent Itemsets. A set of experiments conducted to show the behaviour of our approach at different weighted minimum support and dataset size in one cluster and different cluster size for fixed weighted minimum support. For better values each case is executed three times and the average values are taken.

"Fig.1" shows performance of the algorithm; the execution time of the algorithm is observed for different dataset size with a fixed weighted minimum support on 8 nodes cluster. The results show that the algorithm takes less time even for larger datasets.

"Fig. 2" depicts the performance of the algorithm; in this the execution time of the approach is observed for different weighted minimum support values for two different dataset sizes of 1GB and 10GB. The results show that there is no much difference in execution time when the weighted minimum support is changed.

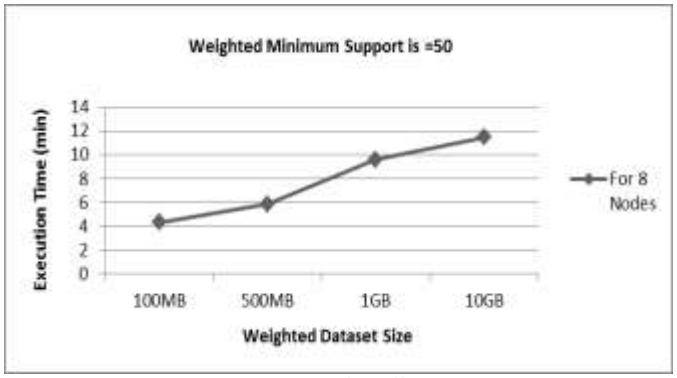

Figure 1 Execution time for different dataset size

Then we fix the weighted minimum support at 50 and analysed the behaviour of the proposed approach at different cluster size for two different data sizes of 1GB and 10GB. "Fig. 3" shows results of these experiments. The results show that there is a much difference in the execution time if the cluster size is less, but there is no much difference in case of number of nodes are increased in the cluster. Also we observed that the impact of MapReduce framework is very less when the cluster size small.

vi. Conclusion and Future Work

Finding infrequent weighted Itemset is one of the important frequent Itemset mining problems. The task of finding weighted infrequent items from very large data needs a lot of computational and memory power.

In this paper we have proposed a method to mine weighted infrequent Itemsets from very large data based on MapReduce model. The results show that the proposed approach in this paper is very efficient in finding weighted infrequent items from very large datasets. Also the experimental results show that the proposed method is more efficient as the data size is increased.

Our future research works include design of better weighting functions, which are more suitable for parallel or distributed data mining environment. Performance of the proposed method is depending on how the weights are calculated during map and reduce phases.

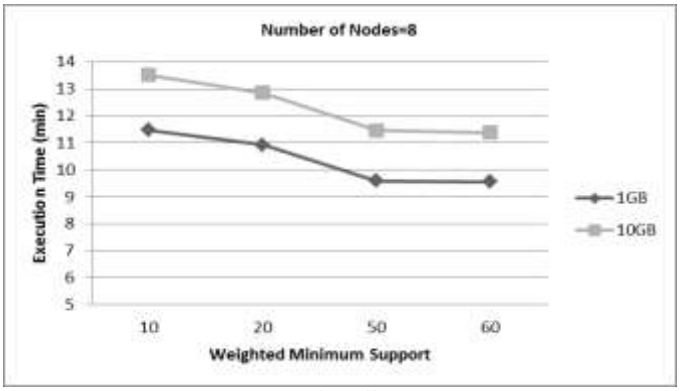

Figure 2 Execution time for different minimum support

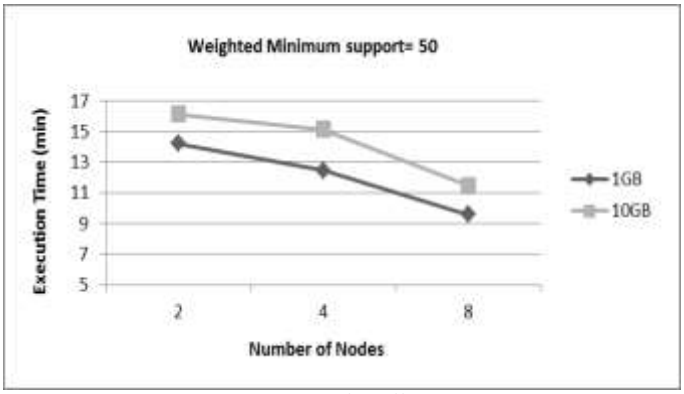

Figure 3 Execution time for different cluster size

References

[1] R. Agrawal, T. Imielinski, A. Swami, "Mining association rules between sets of items in large databases", In Proceedings of ACM SIGOM International Conference on Management of Data, New York, May 1993, pp. 207–216.

[2] R.Agrawal and R.Srikant, "Fast algorithms for mining association rules", In Proceedings of 20th International Conference on VLDB, Chile, May 1994, pp. 207–216.

[3] J.Han and Y.Fu, "Mining multiple-level association rules in large databases", IEEE Trans. on Knowledge and Data Engineering, Vol. 11, No 5, September 1999, pp. 798-805.

[4] XXXX

[5] Jiawei Han and Micheline Kamber, "Data mining: concepts and techniques", Morgan Kaufman, 2001.

[6] X. Wu, C. Zhang and S. Zhang,"Efficient mining of both positive and negative association rules", ACM Trans. on Information Systems, vol.22 (3), 2004, pp 381–405.

[7] Junfeng Ding, Stephen S.T. Yau, "TCOM, an innovative data structure for mining association rules among infrequent items", Computers and Mathematics with Applications, Vol. 57, No. 2, January 2009, pp. 290-301.

[8] Ling Zhou, Stephen Yau, "Efficient association rule mining among both frequent and infrequent items", Computers and Mathematics with Applications, Vol. 54, No.6, September 2007, pp. 737–749.

[9] Yuefeng Li, Abdulmohsen A, and Ning Z, "Mining positive and negative patterns for relevance feature discovery", Proceedings in the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, July 2010, pp.753-762.

[10] Antonella G, Luigi M, Domenico S and Edoardo S,"Solving inverse frequent Itemset mining with infrequency constraints via large-scale linear programs", ACM Transactions on Knowledge Discovery from Data, Vol.7 No.4, November 2013, article 18.

[11] Wei W and Jiong Y, "Efficient mining of weighted association rules", in the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, USA, August 2000, pp.270-274.

[12] Feng T, Fionn M and Mohsen F, "Weighted association rule mining using weighted support and significance framework", in the 9th ACM SIGKDD International Coference on Knowledge Discovery and Data Mining, USA, August 2003,pp.661-666.

[13] Yuanyuan Z, He J, Runian G and Xiangjun D, "Mining weighted negative association rules based on correlation from infrequent items", in International Conference on Advanced Computer Control, Singapore, January 2009, pp.270-273.

[14] Li Z, Zhang F, Lin X and Li K, "A fuzz weighted algorithm for mining infrequent association rules", in the 2nd International Conference on Information Management and Engineering, China, April 2010, pp.94-97.

[15] He J and Xiumei L, "Mining weighted negative association rules from infrequent itemsets based on multiple supports", in international Conference on Industrial Control and Electronics Engineering, Cina, August 2012, pp.89-92.

[16] Guo-Cheng L, Tzung-Pei H, Hong Yu L, Shyue-Liang W and Chun-Wei T, "Enhancing the efficiency in mining weighted frequent itemsets", in the IEEE International Conference on Systems, Man and Cybernetics Society, UK, October 2013, pp.1104-1108.

[17] Luca Cagliero and Paolo Garza "Infrequent weighted itemset mining using frequent pattern growth " IEEE Transactions on Knowledge and Data Engineering, Vol. 26, No. 4, April 2014, pp. 903-915.

[18] Jeffery Dean and Sanjay Ghemawat "MapReduce: simplified data processing on large clusters", 6th Symposium on Operating Systems Design and Implementation, October 2004, pp.107-113.

[19] Jeffery Dean and Sanjay Ghemawat "MapReduce: simplified data processing on large clusters", Communications of the ACM, Vol. 51, No.1, 2008, pp. 107-113.

[20] Lingiuan L and Min Z, "The strategy of mining association rule

[21] based on cloud computing", in International Conference on Business Computing Global Informatization, China, July 2011, pp.475-478.

[22] Ning Li, Li Z, Qing H and Zhongzhi S, "Parallel implementation of apriori algorithm based on MapReduce", 13th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Japan, August 2012, pp. 236-241.

[23] Ming-Yen Li, Pei-Yu L and Sue-Chen H, "Apriori-based frequent Itemset mining algorithms on MapReduce", The 6th International Conference on Ubiquitous Information Management and Communication, Malaysia, February 2012, pp.257-264.

[24] Xin Yue Y, Zhen L and YanFu, "Mapreduce as a programming model for association rules algorithm on hadoop", in 3rd International Conference on Information Sciences and Interaction Sciences, China, June 2010, pp. 99-102.

[25] Othman Y, Osman H and Ehab E, "An efficient implementation of apriori algorithm based on hadoop-MapReduce model", International Journal of Reviews in Computing, Vol. 12, December 2012, pp.57-67.

[26] Su-Qi W, Yu-Bin Y, Guang-Peng C, Yang G and Yao Z, "MapReduce-based closed frequent Itemset mining with efficient redundancy filtering", 12th International Conference on Data Mining Workshops, Belgium, December 2012, pp. 449-453.

[27] Zahara Farzanaryar, Nick Cercone, "Efficient mining of frequent itemsets in social network data based on MapReduce framework", in proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Canada, August 2013, pp.1183-1188.

[28] Mohammadhossein B and Madhi Niamanesh, "ScaniBino: An effective MapReduce-based association rule mining method", in proceedings of the the sixteenth International Conference on Electronic commerce, USA, August 2014, pp.1-8.

[29] Apache Hadoop Project, http://hadoop:apache.org/. accessed at 201408251930.

T Ramakrishnudu currently working as an Assistant Professor in the department of Computer Science and Engineering in National Institute of Technology Warangal, India. His research interests include Data Mining, Distributed Data Mining, Web Technologies and Big Data Analytics. He is a member in IEEE, ACM and Computer Society of India (CSI).

R B V Subramanyam currently working as Associate Professor in National Institute of Technology Warangal, India. He has published many journal and conference papers in the areas of Data Mining. Some of his research interests include Data Mining, Distributed Data Mining, Fuzzy Data Mining, Distributed Data Mining and Big Data Analytics. He is one of the reviewers for IEEE Transactions on Fuzzy Systems and also for Journal of Information and Knowledge Management. He is member in IEEE and The Institution of Engineers (India).