A New Approach for Approximately Mining Frequent Itemsets

Timur Valiullin, Joshua Zhexue Huang, Jianfei Yin, and Dingming Wu

Big Data Institute, College of Computer Science and Software Engineering
Shenzhen University
Shenzhen, China

Abstract. Mining frequent itemsets in transaction databases is an important task in many applications. This task becomes challenging when dealing with a very large transaction database because traditional algorithms are not scalable due to the memory limit. In this paper, we propose a new approach for approximately mining of frequent itemsets in a transaction database. First, we partition the set of transactions in the database into disjoint subsets and make the distribution of frequent itemsets in each subset similar to that of the entire database. Then, we randomly select a set of subsets and independently mine the frequent itemsets in each of them. After that, each frequent itemset discovered from these subsets is voted and the one appearing in the majority subsets is determined as a frequent itemset, called a popular frequent itemset. All popular frequent itemsets are compared with the frequent itemsets discovered directly from the entire database using the same frequency threshold. The recalls and precisions of the frequent itemsets from selected subsets are analyzed against the entire database. The experiment results demonstrate that the use of less than 10 percent of the transaction data in the database can achieve more than 87 percent accuracy. The new approach is very suitable for parallel implementation for large transaction database mining.

Keywords: Approximate Frequent Itemsets Mining, Random Sample, Partition

1. Introduction

Frequent itemsets mining is the first and most critical stage of finding association rules from a transaction database. Association rule mining is one of the main data mining tasks in many applications, such as basket analysis, product recommendation, cross-selling, inventory control, etc. Huge research efforts are devoted to solving frequent itemsets mining problem. Many of these works had considerable impact and led to a plenty of sophisticated and efficient algorithms for association rules mining, such as Apriori [1, 2], FP-Growth (Frequent Pattern Growth) [3–6], Eclat [7–9] and some others. However, the decade fast development of e-commerce, online and off-line shopping has resulted in fast growth of transaction data, which present a tremendous challenge to these existing algorithms, because these algorithms require a large memory to run efficiently on large transaction databases.

Parallel and distributed association rule mining algorithms were developed to handle large transaction databases. Parallel association rule mining algorithms use in-memory computing to efficiently mine association rules from a large transaction database. However, their scalability is limited by the size of memory of the parallel system. Distributed association rule mining algorithms [10, 11] were developed using MapReduce [12] and run on a Hadoop cluster platform. The algorithms have better scalability, but they are not efficient in mining large transaction datasets because of frequent I/O operations and communication overhead between nodes.

In this paper, we propose a new approach for mining frequent itemsets from a big transaction dataset. Similar to the distributed algorithms in MapReduce, we partition the dataset into disjoint subsets of the same size. However, we make the distribution of frequent itemsets in each subset similar to the distribution of frequent itemsets in the entire dataset. Then, we randomly select a set of subsets and run a frequent itemset mining algorithm independently to find the local frequent itemsets from each subset. After all frequent itemsets are discovered from the set of subsets, each frequent itemset is voted by all subsets and the one appearing in the majority subsets is determined as a frequent itemset, called a popular frequent itemset. All popular frequent itemsets are compared with the frequent itemsets discovered directly from the entire database using the same frequency threshold. The recalls and precisions of the popular frequent itemsets from the selected subsets are analyzed against the entire database to show how many true frequent itemsets in the entire transaction dataset can be discovered from the selected subsets.

We have conducted experiments to evaluate the new approach on two datasets. Empirically we have shown that the proposed method is not only capable of producing highly accurate frequent itemsets but also approximating the global frequency of frequent itemsets with very small error.

The remaining of this paper is organized as follows. Related works are discussed in Section 2. Section 3 describes the proposed approach. In Section 4, the details of the algorithm are described. Experiments evaluation is presented in Section 5. Finally, conclusions and future work are drawn in Section 6.

2 Related Work

Frequent itemsets mining is a well-studied problem in computer science. However, the enormous data growth made traditional methods inadequate. Therefore, parallel and distributed algorithms came in use.

Researchers in [13] introduced the parallel implementation of the FP-growth algorithm on GPU. In [10] and [11], the authors introduced two different approaches for mining frequent itemsets in a large database based on MapReduce. In [10], researchers presented two methods for frequent itemsets mining based on Eclat algorithm. The first one is a distributed version of Eclat that partitions the search space more evenly among different processing units, and the second one is a hybrid approach, where k-length frequent itemsets are mined by an Apriori variant, and then the found frequent itemsets are distributed to the mappers where frequent itemsets are mined

using Eclat. Authors of [11] presented a novel zone-wise approach for frequent itemsets mining based on sending computations to a multi-node cluster. All mentioned approaches have obtained a speed increase over the traditional algorithms and allowed to increase the size of the dataset used for mining. However, all introduced approaches require using the entire dataset to get the result. In [14], M. Riondato first introduced PARMA (Parallel Randomized Algorithm for Approximate association rule mining). Algorithm sends random subsets of the database to various machines in the cluster as an input. Then, each machine mines the received subset, and reducers combine the result. Research in [15] is the basis for the current work. Random sample partition (RSP) data model was presented, which showed that the block-level samples from an RSP data model can be efficiently used for data analysis.

3. A New Approach

In our approach, we split a big dataset into smaller disjoint subsets such that the distribution of frequent itemsets in each subset is similar to the distribution of frequent itemsets in the entire dataset. Mining smaller subsets allows using traditional frequent itemset mining algorithms without experiencing memory limit problems. By combining the results of random subsets, we are able to produce highly accurate approximate frequent itemsets.

3.1 Definitions

A transactional dataset $D=\{t_1,t_2,...,t_n\}$ is represented by a collection of n transactions, where each transaction t is a subset of the set of items $I=\{I_1,I_2,...,I_m\}$. An itemset A with k distinct items is referred as *k-itemset*. In this paper, we do not distinguish itemsets with different numbers of unique items. Given an itemset A, define $T_D(A)$ as the set of transactions in D which contain A. The number of transactions in $T_D(A)$ is defined as the support of A by D and denoted as $support(A)=|T_D(A)|$. The frequency of A, i.e., proportion of transactions containing A in D, is denoted as

$$freq_D(A) = \frac{|T_D(A)|}{n}.$$

Under the above definitions, the task for finding frequent itemsets from D with respect to a minimal frequency threshold θ is defined as follows.

Definition 1. *Given a minimum frequency threshold θ for $0<\theta\leq 1$, the frequent itemsets mining with respect to θ is finding all itemsets $\{A_i\}$ for $1\leq i\leq M$ with $freq(A_i)\geq\theta$, where M is the total number of frequent itemsets found in D. Formally, we define the whole set of frequent itemsets in D as*

$$FI(D,I,\theta)=\{(A_i,freq_D(A_i)): A_i\subset I, freq_D(A_i)\geq\theta\}.$$

Definition 2. *Let $FI(D,I,\theta)$ be the set of frequent itemsets in D with respect to θ and $M=|FI(D,I,\theta)|$ the number of frequent itemsets in FI. The accumulative distribution of frequent itemsets in FI is defined as*

$$P(f) = \frac{1}{M} \sum_{\forall A_i \in FI} I(freq_D(A_i)) \geq f$$

where I() is an indicator function and f is a frequency value for θ≤f≤1. The example of P(f) is shown in Fig. 1.

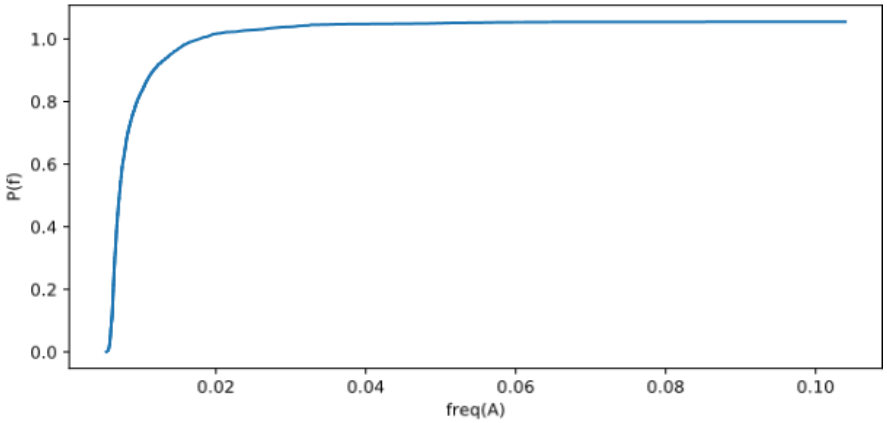


Fig. 1. Example of the accumulative frequent itemsets distribution

Let D be a big transactional dataset and $P=\{D_1,D_2,...,D_k\}$ a partition of D, where $\bigcup_{i=1}^{k} D_i = D$ and $D_i \cap D_j = \emptyset$ for $i \neq j$. D_i for $1 \leq i \leq k$ is named as a block of transactions of dataset D.

Definition 3. *Let $P_D(f)$ be the accumulative distribution of frequent itemsets $FI(D,I,\theta)$ and $P_{D_i}(f)$ the accumulative distribution of frequent itemsets $FI(D_i,I,\theta)$ for $1 \leq i \leq k$. P is a random sample partition of D if*

$$P_{D_i}(f) \rightarrow P_D(f) \text{ as } |D_i| \rightarrow |D|. \quad (1)$$

Definition 3 is a redefined definition of random sample partition in [15] with respect to frequent itemsets by replacing the condition of $E[\tilde{F}_k(t)]=F(t)$ with condition (1), where $\tilde{F}_k(t)$ denotes the sample distribution function of D_k and $E[\tilde{F}_k(t)]$ denotes its expectation.

Definition 4. *$FI_D(D,I,\theta)$ is called the set of global frequent itemsets and $FI_{D_i}(D_i,I,\theta)$ the set of local frequent itemsets. Accordingly, $P_D(f)$ is the accumulative distribution of global frequent itemsets and $P_{D_i}(f)$ is the accumulative distribution of the local frequent itemsets in D_i.*

3.2. Approximate Computing

When the transactional dataset D is big and cannot be held in memory, we cannot run a frequent itemset mining algorithm on D to find all frequent itemsets $FI_D(D,I,\theta)$. In this situation, we randomly select a set of l transaction blocks $\{D_1,D_2,...,D_l\}$ from the partition P and use the set of local frequent itemsets $FI_{D_i}(D_i,I,\theta)$ for $1 \le i \le l$ to estimate the set of global frequent itemsets $FI_D(D,I,\theta)$. This approach is called approximate frequent itemset mining.

Definition 5. *Let itemset A be a frequent itemset in $FI_{D_i}(D_i,I,\theta)$ for $1 \le i \le l$. A is called a popular frequent itemset if*

$$\sum_{i=1}^{l} I\left(A \in FI_{D_i}(D_i, I, \theta)\right) > \alpha \tag{2}$$

where I() is an indicator function and α is a given integer greater than l/2.

Definition 6. *The frequency of a popular frequent itemset A is defined as*

$$freq(A) = \frac{\sum_{i=1}^{l} freq_{D_i}(A) \times I(A \in FI_{D_i}(D_i, I, \theta))}{\sum_{i=1}^{l} I(A \in FI_{D_i}(D_i, I, \theta))}$$

The set of all popular frequent itemsets PFI from $FI_{D_i}(D_i,I,\theta)$ for $1 \le i \le l$ is the estimation of the set of global frequent itemsets $FI_D(D,I,\theta)$. Given PFI and assuming $FI_D(D,I,\theta)$ is known, an itemset A has the following status:

- true positive if $A \in PFI$ and $A \in FI_D(D,I,\theta)$
- false positive if $A \in PFI$ and $A \notin FI_D(D,I,\theta)$
- true negative if $A \notin PFI$ and $A \notin FI_D(D,I,\theta)$
- false negative if $A \notin PFI$ and $A \in FI_D(D,I,\theta)$

4. An Approximate Frequent Itemsets Finding Algorithm

In this section, we propose an approximate algorithm for finding the set of popular frequent itemsets from a set of l transaction blocks $\{D_1,D_2,...,D_l\}$ randomly selected from the partition of a big transactional dataset D, and using the popular frequent itemsets to estimate the set of frequent itemsets in D with respect to a given frequency threshold θ. The algorithm consists of three steps: converting the dataset D into a partition of k transaction blocks and randomly selecting l blocks from the partition; finding the local frequent itemsets for each of l selected transaction blocks; finding the popular frequent itemsets from the local frequent itemsets.

4.1 Generate Partition of Transaction Blocks

Given a transaction dataset *D*, the first step is to convert it to a partition of transaction blocks. *D* is preprocessed such that each record represents one purchase transaction and the transactions with one purchased item are removed. The pseudo code for creating the random partition is given in Algorithm 1.

Algorithm 1 RSP Blocks generation and selection

Input:
-*D*: **preprocessed data;**
-*l*: **number of subsets;**
-*m*: **subset size;**

1: **procedure** RSPBlocks(*D*,*l*,*m*)
2: $k = \frac{|D|}{m}$
3: **for** each D_i, 1 <= *i* <= *k* **do**
4: randomly assign *m* transactions from *D* to the *i*-th block without replacement
5: **end for**
6: randomly select *l* transaction blocks from the set of created *k* blocks, *l* <= *k*
7: **Output**: set of *l* transaction blocks of *D*
8: **end procedure**

4.2 Finding Local Frequent Itemsets

In this step, Apriori algorithm is called to find the local frequent itemsets from each of *l* transaction blocks $\{D_1,D_2,...,D_l\}$ with respect to a given minimum frequency threshold θ. Finally, *l* sets of local frequent itemsets are obtained. The pseudo code of obtaining local frequent itemsets is presented in Algorithm 2.

Algorithm 2 Local frequent itemsets mining

Input:
-$\{D_l\}$: **set of** *l* **transaction blocks of** *D*;
-θ: **minimum frequency threshold**

1: **procedure** LocalFIs($\{D_l\}, \theta$)
2: **for** each D_i, 1 <= *i* <= *l* **do**
3: $FI_i = Apriori(D_i, \theta)$
4: **end for**
5: **Output**: $\{FI_l\}$ - set of *l* sets of the local frequent itemsets
6: **end procedure**

4.3 Finding Popular Local Frequent Itemsets

The *l* sets of local frequent itemsets are united into one set of unique local frequent itemsets. For each frequent itemset in the united set, the number of its appearances in the *l* sets is checked with Eq. (2). If the condition is satisfied, the frequent itemset is considered as a popular frequent itemset. Otherwise, it is dropped. All local frequent itemsets in the united set are checked and the set of popular frequent itemsets is obtained. These popular frequent itemsets are used as the approximate set of the frequent itemsets in *D* with respect to the same minimum frequency threshold θ. The pseudo code is given in Algorithm 3.

Algorithm 3 Popular Frequent Itemset mining

Input:
-$\{FI_l\}$**: set of** *l* **local frequent itemsets;**

1: **procedure** PopularFIs($\{FI_l\}$)
2: $FI = dictionary(\cup_{i=1}^{l} FI_i)$ // for all frequent itemsets found, creating <key, value>pair, where itemset is a key and number of its repeats in all blocks is a value 3:
for each frequent itemset $\in FI$ **do**
4: **if** value$> \frac{k}{2}$ **then**
5: include frequent itemset to the set of popular frequent itemsets
6: **end if**
7: **end for**
8: **Output**: set of popular frequent itemsets
9: **end procedure**

5. Experiments

To demonstrate the performance of the approximate frequent itemsets algorithm, we conducted a series of experiments on two datasets. We run our algorithm several times with different numbers of transaction blocks and different block size, and compared the set of popular frequent itemsets with the exact set of frequent itemsets obtained from the entire dataset.

5.1 Datasets

We evaluated the proposed approach on 2 datasets downloaded from Kaggle.com and Open-Source Data Mining Library. Properties of datasets used in the experiments are described in Table 1.

Table 1. Properties of the datasets used in experiments

	Kaggle dataset	Online Retail dataset
Number of transactions	729148	541908
Number of items	791	2603
Average transaction length	8	4

5.2 Experiment Settings

In order to test the proposed algorithm, the set of popular frequent itemsets was compared against the set of frequent itemsets in the entire dataset to compute the accuracy, recall, and precision. We conducted 50 experiments for each set of parameters specified in Table 2 and averaged obtained results afterward. For both datasets, we used the same minimum frequency threshold for both local and global frequent itemsets. We chose threshold to be small enough to produce a big collection of the output frequent itemsets and set θ to be 0.005 for all experiments.

Testing was started with a comparison of the accumulative distribution of frequent itemsets for local and global frequent itemsets and proceeded with the evaluation of the different metrics of the popular frequent itemsets.

Table 2. Parameters used for experiments

Number of blocks	Block size
50	10000, 5000, 3500, 2000, 1000, 500
30	10000, 5000, 3500, 2000, 1000, 500
15	10000, 5000, 3500, 2000, 1000, 500
10	10000, 5000, 3500, 2000, 1000, 500
5	10000, 5000, 3500, 2000, 1000, 500

5.3 Evaluation Methods

For evaluation of the accuracy and sufficiency of obtained approximate frequent itemsets, we used the confusion matrix in our research. Using the confusion matrix allows analyzing the efficiency of the proposed approach more detailed by introducing three measures, namely recall, precision, and accuracy.

$$Recall = \frac{TP}{TP + FN}$$

shows the fraction of the global frequent itemsets that are contained in the popular FIs.

$$Precision = \frac{TP}{TP + FP}$$

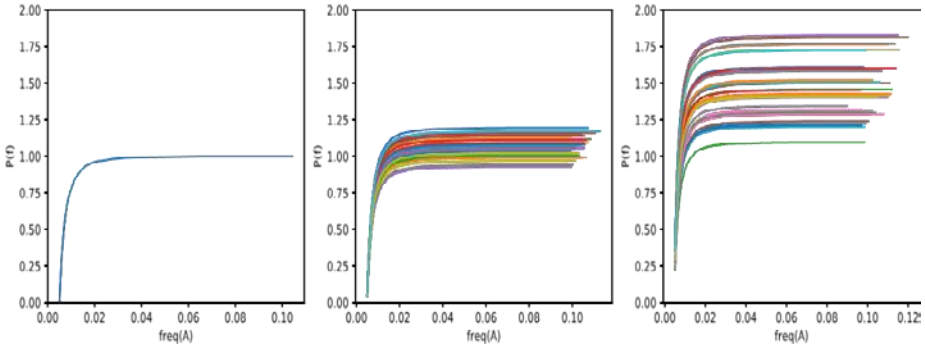
shows the fraction of the popular frequent itemsets that are contained in the set of the global frequent itemsets.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

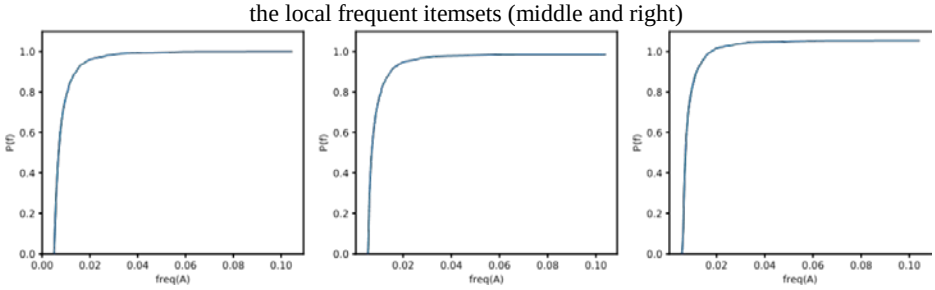
shows the proportion of accurate results among the total number of cases examined.

5.4 Experiment Results

We started with a comparison of accumulative distributions of the local and global FIs. Fig. 2(a) clearly shows that the accumulative distribution of global frequent itemsets is similar to the accumulative distributions of local frequent itemsets with a sufficiently large subset size (Fig. 2(a) left and middle graphs). Different colors show the differences of the accumulative distributions between the local frequent itemsets. However, decreasing the size of the subset leads to the growth of the number of local frequent itemsets (Fig. 2(a) right graph) and results in a significant difference between the accumulative distributions of the global and local frequent itemsets. Nevertheless, the accumulative distribution of the popular frequent itemsets shows almost identical accumulative distribution to the global frequent itemsets. The accumulative distributions of the global and popular frequent itemsets are represented in Fig. 2(b). It shows that the number of the popular FIs almost matches to the number of global FIs, and the overall frequency of the popular frequent itemsets is consistent with the frequency of the global frequent itemsets.



(a) Accumulative frequent itemsets distributions of the global frequent itemsets (left) and the local frequent itemsets (middle and right)



(b) Accumulative frequent itemsets distribution of the global frequent itemsets(left), accumulative frequent itemsets distributions of the popular frequent itemsets

Fig. 2. Accumulative frequent itemsets distributions. Number of subsets = 30, subset size (middle) = 10000, subset size (right) = 2000 (Kaggle dataset)

The proposed algorithm approximates the exact set of frequent itemsets in the entire dataset. The difference between approximate and global collections is false-

positive FIs. The number of false-positive frequent itemsets affects one of the accuracy measures, namely precision. Fig. 3 shows, how the precision value is affected by different experimental parameters. It is observed that using more transaction blocks decreases the number of false-positive itemsets, therefore increasing the precision. From the graph, we can see that the number of falsepositive frequent itemsets decreases as the growth of the block size.

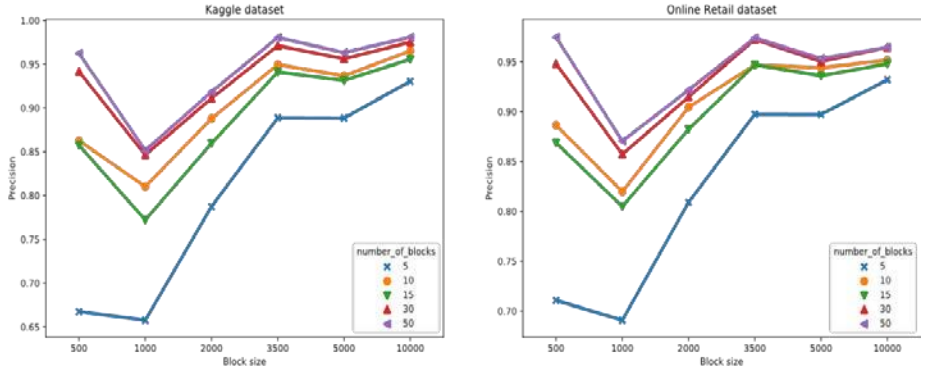


Fig. 3. Precision changes with different parameters

The overall change of the accuracy defined in terms of a confusion matrix is represented in Fig. 4. From the graphs, we can see that accuracy increase can be obtained by increasing both the number of subsets used and the subset size.

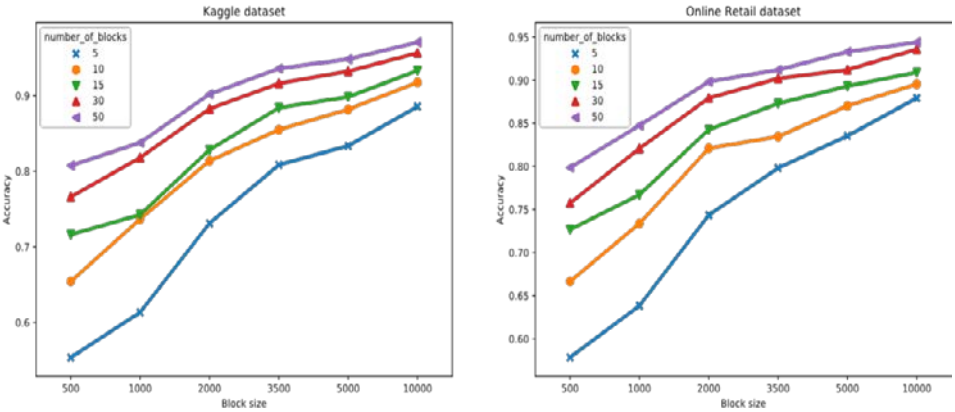


Fig. 4. Accuracy changes with different parameters

5.5. Result Analysis

To evaluate the efficiency of our approach, we conducted 50 independent experiments on both datasets with specified parameters in Table 2 to estimate the performance for each set of parameters. For each test run, the set of approximate frequent itemsets was compared to the exact set of the frequent itemsets obtained by mining the entire dataset. As a result, we received 50 different observations of elapsed time, recall, precision and accuracy for each set of parameters, and then averaged all values. Fig. 5 illustrates how the average accuracy changes with the change in the amount of data being mined. The graphs show that the proposed algorithm is capable of producing approximate frequent itemsets with above 87% of the accuracy, using only a little less than 10% of data.

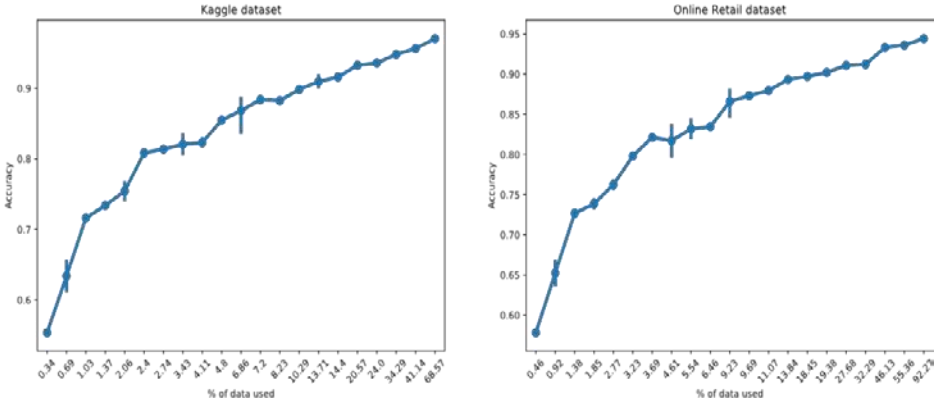


Fig. 5. Accuracy increases with the rise of data used to mine

We also conducted an evaluation of the estimated frequency error in the approximate frequent itemsets for all experimental parameters. In Fig. 6, we depict the distribution of the average absolute error in the frequency estimation, defined as:

$$\frac{\sum_{i=1}^{l} |freq_{D_i}(A) - freq(A)|}{|PFI| - |FI(D, I, \theta)|}$$

for all itemsets *A* that are contained in both the approximate and the global frequent itemsets. We can see that the flustration of the error decreases with the increase of the subset size. The error is reduced as the increase of the number of transaction blocks and the block size.

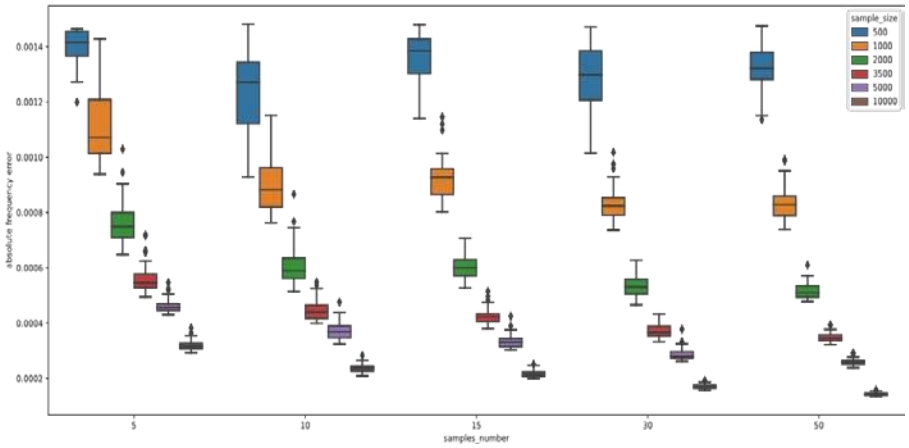


Fig. 6. Error in frequency estimations for different parameters (Kaggle dataset)

6. Conclusions and Future Work

In this paper, we have presented a new approach for mining approximate collections of frequent itemsets based on a random sample partition of the data. We have shown that using the RSP data model in big data can be very beneficial, especially in the frequent itemset mining task, since the size of transaction database grows much faster than the contained patterns change.

For the further work, we are going to implement the parallel version of the algorithm on a cluster and to conduct experiments on big datasets in terabyte scale. We will also conduct a theoretical analysis of the approach.

References

1. Agrawal, R., Imielinski, T., and Swami, A.: Mining association rules between sets of items in large databases. In Proceedings of SIGMOD, 1993.
2. Agrawal, R. and Srikant, R.: Fast algorithms for mining association rules in large data bases. In Proceedings of VLDB, 1994.
3. Han, J., Pei, J., and Yin, Y.: Mining frequent patterns without candidate generation. In Proceedings of the 19th ACM International Conference on Management of Data (SIGMOD), 2000.
4. Grahne, G. and Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In Proceedings of the CEUR Workshop Proceedings, 2003.
5. Racz, B. An fp-growth variation without rebuilding the fp-tree. In Proceedings of the CEUR Workshop Proceedings, 2003.
6. Grahne, G. and Zhu, J.: Reducing the main memory consumptions of fpmax* and fpclose. In Proceedings of the CEUR Workshop Proceedings, 2004.
7. Zaki, M.J., Parthasarathy, S., Ogihara, M., and Li, W.: New algorithms for fast discovery of association rules. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, 1997.
8. Zaki, M.J. and Gouda, K.: Fast vertical mining using diffsets. In Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining, 326–335, Washington, DC, USA, 2003.
9. Schmidt-Thieme, L.: Algorithmic features of eclat. In Proceedings of the Workshop Frequent Item Set Mining Implementations, 2004.

10. Moens, S., Aksehirli, E., and Goethals, B.: Frequent itemset mining for big data. In 2013 IEEE International Conference on Big Data, 2013.
11. Prajapati, D., Garg, S., and Chauhan, N.C.: Interesting association rule mining with consistent and inconsistent rule detection from big sales data in distributed environment. Future Computing and Informatics Journal **2** (1), 19–30 (2017).
12. Dean, J. and Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In Proceedings of the CACM, 107–113 (2004).
13. Jiang, H. and Meng, H.: A parallel fp-growth algorithm based on gpu. In 2017 IEEE 14th Int. Conf. E-bus. Eng, 97–102 (2017).
14. Riondato, M., DeBrabant, J.A., Fonseca, R., and Upfal, E.: Parma: a parallel randomized algorithm for approximate association rules mining in mapreduce. In Proceedings of the ACM International Conference on Information and Knowledge Management, 2012.
15. Salloum, S., He, Y., Huang, J.Z., Zhang, X., and Emara, T.Z.: A random sample partition data model for big data. In [Online]. Available: https://arxiv.org/abs/1712.04146, 2017.