Computer Science and Information Systems 18(3):641–656 https://doi.org/10.2298/CSIS200124015V

A New Approximate Method For Mining Frequent Itemsets From Big Data ⋆

Timur Valiullin, Joshua Zhexue Huang⋆⋆, Chenghao Wei, Jianfei Yin, Dingming Wu, and Iuliia Egorova

Big Data Institute, College of Computer Science and Software Engineering
Shenzhen University
518000 Shenzhen, China
{timur, zx.huang, chenghao.wei, yjf, dingming}@szu.edu.cn, jnegorova@gmail.com

Abstract. Mining frequent itemsets in transaction databases is an important task in many applications. It becomes more challenging when dealing with a large transaction database because traditional algorithms are not scalable due to the limited main memory. In this paper, we propose a new approach for the approximately mining of frequent itemsets in a big transaction database. Our approach is suitable for mining big transaction databases since it uses the frequent itemsets from a subset of the entire database to approximate the result of the whole data, and can be implemented in a distributed environment. Our algorithm is able to efficiently produce high-accurate results, however it misses some true frequent itemsets. To address this problem and reduce the number of false negative frequent itemsets we introduce an additional parameter to the algorithm to discover most of the frequent itemsets contained in the entire data set. In this article, we show an empirical evaluation of the results of the proposed approach.

Keywords: Approximation Method, Frequent Itemsets Mining, Random Sample Partition, Big Transactional Database.

1. Introduction

Frequent itemsets mining is the first and most critical step of finding association rules from a transaction database. Association rules mining is one of the main data mining tasks in many applications, such as basket analysis [3], product recommendation [20], cross-selling [10], etc. Huge research efforts have been devoted to solving frequent itemsets mining problem. Many of these studies 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) [8,6,13,7], and Eclat [22,21,18]. However, decade of a fast development of e-commerce, online and offline shopping has resulted in the fast growth of transaction data, which presents a tremendous challenge to these existing algorithms, because these algorithms require large memory to run efficiently on large transaction databases.

Parallel and distributed association rules mining algorithms were developed to handle large transaction databases. Parallel association rules mining algorithms use in-memory

⋆ This is an extended version of the DAMDID 2019 conference paper "A New Approach for Approximately Mining Frequent Itemsets."

⋆⋆ Corresponding author

computing to efficiently mine association rules from a large transaction database. However, their scalability is limited by the size of the memory of the parallel systems. Distributed association rules mining algorithms were developed using MapReduce [5], and run on a Hadoop cluster platform. These algorithms have better scalability, but they are not efficient for mining of a large transaction data sets because of frequent I/O operations and communication overhead between nodes.

In this paper, we propose a new approach to solve the problem of mining frequent itemsets from a big transaction data set. Similar to the distributed algorithms in MapReduce, we partition the data set into same size disjoint subsets. However, *we make the distribution of frequent itemsets in each subset similar to the distribution of frequent itemsets in the entire data set by random assignment of the transactions from the entire data set to the subsets without replacement.* As such, we can randomly select a few subsets and run a frequent itemsets mining algorithm on each subset independently to discover the local frequent itemsets from it. After all frequent itemsets are discovered from the subsets, each frequent itemset is voted by all subsets and the one appearing in the majority of subsets is determined as the frequent itemset, called a *popular frequent itemset*.

In this approach, we define the frequency (relative support) threshold for finding all frequent itemsets in the entire transaction database as the global frequency threshold, and the frequency threshold for mining all frequent itemsets in a subset of the transaction database as the local frequency threshold. Based on these two thresholds, we introduce an algorithm for finding the set of approximate frequent itemsets that estimates the set of frequent itemsets in the entire data set. Initially, the algorithm makes the local frequency threshold equal to the global frequency threshold for subsets to mine local frequent itemsets. This setting results in a high-accurate approximate set of frequent itemsets, but the result also contains insignificant amount of both false positive and false negative frequent itemsets. To address this problem, we introduce an additional parameter to make the local frequency threshold smaller than the global frequency threshold for subsets to produce the set of local frequent itemsets. Using a reduced local frequency threshold helps to drastically reduce the number of false negative frequent itemsets and produce high-accurate approximate frequent itemsets that cover most of the frequent itemsets contained in the entire data set with respect to the global frequency threshold.

We conducted experiments to evaluate the approximate solutions on two real world data sets. To evaluate the performance of our method, all popular frequent itemsets discovered from the selected subsets using the local frequency threshold are compared with the frequent itemsets found directly from the entire database using the global frequency threshold. The recalls and precisions of the popular frequent itemsets obtained from the selected subsets are used as evaluation measures. The empirical results have shown that the proposed method is capable of producing high-accurate approximate frequent itemsets and discovering most of the frequent itemsets contained in the entire database that can be found with the global frequency threshold. The comprehensive analysis also shows that reducing the local frequency threshold in the selected subsets enables obtaining all true frequent itemsets.

The remaining part 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 presented. Experiment results are shown 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.

Authors of [17] implemented a new algorithm called Partition to mine approximate frequent itemsets which achieved both CPU and I/O improvements over Apriori by partitioning the database into a number of non-overlapping partitions so that the partitions are small enough to be handled in the main memory to generate local candidates. On the next step, the local results are merged and the global frequency of the local frequent itemsets is checked in the entire data set. In [19], H. Toivonen introduced new sampling based algorithms to make association rules mining more efficient in terms of computational cost. The proposed approach uses a sample of a data set with a lowered frequency threshold to generate a bigger collection of frequent itemset candidates, and then verifies the candidates with the entire database. Researchers in [9] introduced the parallel implementation of the FP-growth algorithm on GPU. In [11] and [12], the authors introduced two different approaches for mining frequent itemsets in a large database based on MapReduce. In [11], 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 the k-length frequent itemsets are mined by an Apriori variant, and then the found frequent itemsets are distributed to the mappers in which frequent itemsets are mined using Eclat. Authors of [12] 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 data set used for mining. However, all introduced approaches require using the entire data set to get the result, which faces the memory bottleneck when dealing with big data and working in a distributed environment.

Researchers in [4], [15] introduced sampling techniques which theoretically proved the existence of the tight bounds of the sample size that guarantees the approximation with respect to the parameters specified by the user. The sample size for mining is not dependent on the size of the database and the number of items. However, in case of a real big data the proposed method might not be applicable since the big sample data, used to achieve the approximation with respect to the parameters specified by the user, may not fit in the memory of a single machine. The proposed approaches are not suitable for the distributed environment. In [14], M. Riondato introduced PARMA algorithm (Parallel Randomized Algorithm for Approximate association rule mining). The 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. Our work follows the idea described in [16]. The 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 data set into disjoint subsets such that the distribution of frequent itemsets in each subset is similar to the distribution of frequent itemsets in the

entire data set. Mining smaller subsets allows using traditional frequent itemset mining algorithms without experiencing memory limit problems. In this research, we have empirically shown that by combining the results of randomly selected subsets, we are able to produce a highly accurate approximate set of frequent itemsets.

3.1. Definitions

A transactional data set $D = \{t_1, t_2, ..., t_n\}$ 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 to *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. the proportion of transactions containing A in D, is denoted as $freq_D(A) = \frac{|T_D(A)|}{n}$, called a *relative support* or *frequency* of A.

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

Definition 1. Given a minimum global 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

$$FI(D, I, \theta) = \{(A_i, freq_D(A_i)) : A_i \subset I, freq_D(A_i) \geq \theta\} \tag{1}$$

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

$$P(f) = \frac{1}{M} \sum_{\forall A_i \in FI} \mathcal{I}(freq_D(A_i) \leq f) \tag{2}$$

where $\mathcal{I}()$ is an indicator function and f is a frequency value for $\theta \leq f \leq 1$. An example of $P(f)$ is shown in Figure 1.

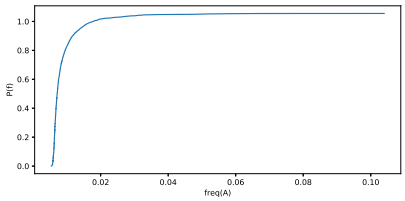


Fig. 1. Example of the accumulative frequent itemsets distribution, where $\theta = 0.005$

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

of transactions of data set 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

$$P_{D_i}(f) \rightarrow P_D(f) \quad as \quad D_i \rightarrow D \tag{3}$$

where $D_i \rightarrow D$ implies that D_i approaches D as the size of D_i increases. D_i for $1 \leq i \leq k$ is called an RSP data block.

Definition 3 is a redefined definition of random sample partition in [16] with respect to frequent itemsets by replacing the condition of $E[\tilde{F}_k(t)] = F(t)$ with condition (3) where $\tilde{F}_k(t)$ denotes the sample distribution function of t in D_k and $E[\tilde{F}_k(t)]$ denotes its expectation.

3.2. Approximate Frequent Itemsets Mining

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

Definition 4. Let itemset A be a frequent itemset in $FI_{D_i}(D_i, I, \theta)$ for $1 \leq i \leq l$. A is called a *popular frequent itemset* if

$$\sum_{i=1}^{l} \mathcal{I}(A \in FI_{D_i}(D_i, I, \theta - \epsilon)) > a \tag{4}$$

where $\mathcal{I}()$ is an indicator function, ϵ for $0 \leq \epsilon < \theta$ is a parameter to reduce the local frequency threshold from the global frequency threshold value θ, and a is a given integer greater or equal to $l/2$, so Equation 4 is a simple majority voting.

The set of all popular frequent itemsets PFI from $FI_{D_i}(D_i, I, \theta)$ for $1 \leq i \leq l$ 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 one of the following statuses:

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

We intentionally omit the true negative frequent itemsets from the above definition because we are not interested in mining infrequent itemsets (itemsets with frequency $< \theta$).

4. An Approximate Frequent Itemsets Finding Algorithm

In this section, we propose a new algorithm for finding the set of approximate frequent itemsets from a set of l RSP data blocks $\{D_1, D_2, ..., D_l\}$ randomly selected from the partition of a big transaction data set D, and using the local frequent itemsets to estimate the set of frequent itemsets in D with respect to a global frequency threshold θ. The algorithm contains four steps: RSP data blocks generation; RSP data blocks selection; local frequent itemsets mining; finding the approximate set of frequent itemsets from the sets of local frequent itemsets by voting.

The pseudo code is presented in Algorithm 1. The inputs are: a transaction database D, the number of transactions in each RSP data block m, the number of RSP data blocks selected for finding frequent itemsets l, two parameters θ and ϵ, and the parameter of the popular frequent itemset voting condition α. The output of this algorithm is the set of the popular frequent itemsets PFI.

The first step in lines 1-6 is to convert D to a partition of k RSP transaction blocks. Each record in D represents one purchase transaction and the transactions with one purchased item are removed. Given the size of the RSP data block m, the number of RSP data blocks in the partition k is the number of transactions in D divided by m. To generate k RSP data blocks, m transactions are randomly selected from D without replacement and assigned to each RSP data block. In the second step in lines 7-9, l RSP data blocks are randomly selected from the k RSP data blocks of the partition without replacement. In the third step in lines 10-15, Apriori algorithm is called with parameter value of $(\theta - \epsilon)$ as the local frequency threshold to find the local frequent itemsets in each of l RSP transaction blocks $\{D_1, D_2, ..., D_l\}$. In the fourth step in lines 16-23, all local frequent itemsets are voted by all sets of local frequent itemsets. The result of a frequent itemsets mining algorithm is a set of string objects, therefore in this article, when comparing the local results and counting the number of appearances of a frequent itemset in all RSP data blocks, we mean an exact match of string objects. Thus, if a local frequent itemset occurs in RSP data blocks more than the given number as shown in Equation 4, the frequent itemset is added to the set of popular frequent itemset; otherwise, it is discarded.

In this article, we provide a comprehensive empirical analysis of the influence of parameter ϵ on the approximate result. We separately consider two cases. The first one is when we set $\epsilon = 0$ which indicates that the local frequency threshold is equal to the global frequency threshold θ. The second one is $0 < \epsilon < \theta$ which indicates that the local frequency threshold is smaller than the global frequency threshold θ. In Section 5, we will show that even the small size of the data block with sufficient number of selected RSP data blocks enables obtaining high quality estimation of the set of global frequent itemsets, however the set of popular frequent itemsets will contain insignificant amount of false positive frequent itemsets, moreover setting $\epsilon = 0$ does not allow discovering all true frequent itemsets from the selected RSP data blocks. To address this problem, we reduce the local frequency threshold for mining local frequent itemsets. Decreasing of the local frequency threshold by increasing the value of ϵ will result in increasing of the number of the local frequent itemsets for each RSP data block. Thus, more true positive frequent itemsets can be discovered from the selected RSP data blocks, i.e. the number of false negative frequent itemsets will be reduced, however it will also increase the content of false positive frequent itemsets in the approxiamte solution. Empirical analysis for choosing ϵ and experiment results for both cases are shown in Section 5.

Algorithm 1 Algorithm for approximate discovery of frequent itemsets from a big transaction database using random sample partition

Input:
-D: transaction database;
-m: number of transactions in each RSP data block;
-l: number of RSP data blocks selected for finding frequent itemsets;
-θ: the global minimum frequency threshold;
-ϵ: local minimum frequency threshold deduction parameter;
-α: popular frequent itemset voting condition parameter.

procedure RSP_GENERATION(D, m)
  $k = \frac{|D|}{m}$ ▷ $|D|$ is the number of transactions in D and k is the number of RSP blocks to be generated.
  for each D_i, $1 <= i <= k$ **do**
    randomly assign m transactions from D to the i-th RSP data block without replacement
  end for
end procedure
procedure RSP_BLOCK_SELECTION($\{D_k\}, l$)
  $\{D_l\}$ = random.sample($\{D_k\}, l$) ▷ randomly select l RSP data blocks from the set $\{D_k\}$ and put them in set $\{D_l\}$.
end procedure
procedure LOCAL_FIS($\{D_l\}, l, \theta, \epsilon$)
  for each D_j, $1 <= j <= l$ **do**
    $FI_j = Apriori(D_j, \theta - \epsilon)$
    $\{FI_l\}$.append(FI_j) ▷ $\{FI_l\}$ is the set of l sets of local frequent itemsets.
  end for
end procedure
procedure POPULAR_FIS($\{FI_l\}$)
  $\{FI\} = dictionary(\cup_{i=1}^{l} FI_i)$ ▷ for all frequent itemsets found, create $<key, value>$ pair, where itemset is a key and the number of repeats in all RSP data blocks is a value.
  for each $frequent\_itemset \in \{FI\}$ **do**
    if value $>\alpha$ **then**
      PFI.append($frequent\_itemset$) ▷ append a popular frequent itemset to the set of popular frequent itemsets.
    end if
  end for
end procedure
Output: set of the popular frequent itemsets PFI

5. Experiments

We conducted 30 experiments on two real world transaction data sets. To evaluate the quality of the approximate results, we compared popular frequent itemsets with the frequent itemsets found directly from the entire database with respect to the global frequency threshold θ. The comparison has shown that our approach produced good approximate results and is efficient in mining approximate frequent itemsets from a big transaction database. In this section, we present the real world data sets, experiment settings, evaluation methods and the experiment results.

5.1. Data Sets and Experiment Settings

The two data sets used in these experiments were downloaded from Kaggle.com and Open-Source Data Mining Library, respectively. The characteristics of the data sets are listed in Table 1.

Table 1. The two data sets used in experiments

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

Thirty experiments were conducted on each data set with different settings on the number of RSP data blocks and the block sizes. The setting values of these experiments are given in Table 2. The global frequency threshold was set as $\theta = 0.005$ so a big set of frequent itemsets was discovered from the entire data set.

Table 2. Settings on number of RSP data blocks and block sizes

Number of RSP data blocks	Block sizes
50	10000, 5000, 3500, 2000, 1000
30	10000, 5000, 3500, 2000, 1000
15	10000, 5000, 3500, 2000, 1000
10	10000, 5000, 3500, 2000, 1000
5	10000, 5000, 3500, 2000, 1000

5.2. Evaluation Measures

In these experiments, we used the following three measures to evaluate the approximate results of the proposed approach. In these evaluations, the set of popular frequent itemsets PFI was compared with the set of global frequent itemsets. The popular frequent itemsets in PFI were divided into two classes: true positive frequent itemsets TP and false positive frequent itemsets FP. There is another class of false negative frequent itemsets FN, which can only be found in the global frequent itemsets. Based on the three sets of frequent itemsets, we define the evaluation measures as follows:

$$Recall = \frac{|TP|}{|TP \cup FN|} \tag{5}$$

where $|TP|$ is the number of frequent itemsets in TP and $|TP \cup FN|$ is the number of the global frequent itemsets because $TP \cup FN = FI_D(D, I, \theta)$. This measure shows how good the algorithm is in discovering true frequent itemsets. Since this measure is very important in frequent itemsets mining, we use the parameter ϵ to reduce the local

frequency threshold while mining local frequent itemsets for increasing the recall value. As ϵ approaches to θ, the recall value will get close to 1.

However increasing ϵ to θ will affect precision value defined as:

$$Precision = \frac{|TP|}{|TP \cup FP|} \tag{6}$$

where $TP \cup FP = PFI$. Precision measures the fraction of the true frequent itemsets in the set of popular frequent itemsets. This measure shows how good the algorithm is in avoiding discovering of the false positive frequent itemsets. Using $(\theta - \epsilon)$, $\epsilon > 0$ as the local frequency threshold to mine local frequent itemsets tends to discover more local frequent itemsets, therefore potentially increasing the number of false positive frequent itemsets in the approximate result, which negatively affects the precision measure.

To consider the global frequent itemsets, the accuracy measure is defined as:

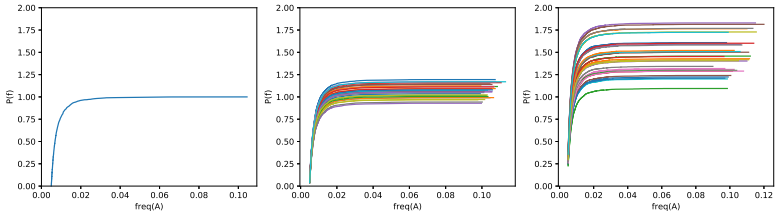
$$Accuracy = \frac{|TP|}{|TP \cup FN \cup FP|} \tag{7}$$

This measure evaluates the approximate result in terms of both precision and recall. Since $TP \cup FN \cup FP = PFI \cup FI_D(D, I, \theta)$, this measure evaluates the quality of the approximate solution in terms of its ability to discover true frequent itemsets, avoiding discovering infrequent itemsets (itemsets with the global frequency $< \theta$).

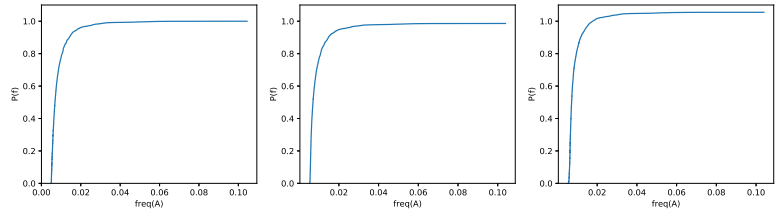
5.3. Experiment Results With $\epsilon = 0$

In this series of experiments, we set $\theta = 0.005$ and $\epsilon = 0$, i.e. the local frequency threshold is equal to the global frequency threshold. With these settings, we ran Algorithm 1 on the entire data set on different numbers of data blocks and block sizes as specified in Table 2. Figure 2(a) shows the accumulative distributions of frequent itemsets in the whole Kaggle data set and RSP data blocks of different sizes. Subplot on the left shows the distribution in the entire data set. The plots in the middle are the distributions of the frequent itemsets in the RSP data blocks with 10000 transactions. We can see that the distributions of the frequent itemsets in the RSP data blocks are similar to each other and also similar to the distribution in the entire data set. The plots on the right show the distributions of the frequent itemsets in the RSP data blocks with 2000 transactions. Because the block size is much smaller than the blocks for distributions in the middle, a big difference displays among the blocks although the shapes of the distributions are similar to the distribution of the entire data set. We can say that the RSP data blocks in the middle are better samples of the entire data set than the RSP data blocks on the right.

Figure 2(b) shows the accumulative distributions of frequent itemsets in the entire data set and the accumulative distributions of popular frequent itemsets in a number of RSP data blocks of different sizes. Figure 2(b) shows that the three distributions are very similar to each other. The distribution in the middle subplot is very close to the distribution of frequent itemsets in the entire data set. The distribution on the right is also close to the one in the middle. This indicates that the popular frequent itemsets improve the results of frequent itemsets from individual data blocks even with a small block size, and are better estimates of the frequent itemsets in the entire data set with respect to the same frequency



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



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

Fig. 2. Accumulative frequent itemsets distributions. Number of RSP data blocks = 30, block size (middle) = 10000, block size (right) = 2000. (Kaggle data set)

threshold θ. Generally speaking, these figures show that the RSP data blocks are good random samples for estimating the frequent itemsets contained the entire data set.

The quality of the approximate results from randomly selected RSP data blocks is evaluated with the measures of Accuracy, Precision and Recall defined in the previous section. Figure 3 shows the changes of accuracy against the data block size in different numbers of selected RSP data blocks. The left figure shows the results of Kaggle data set and the right figure is the results of Online Retail data set. We can see that the accuracy increases as the block size increases. Also, the more the RSP data blocks are selected for voting the popular frequent itemsets, the higher the accuracy of the approximate result is.

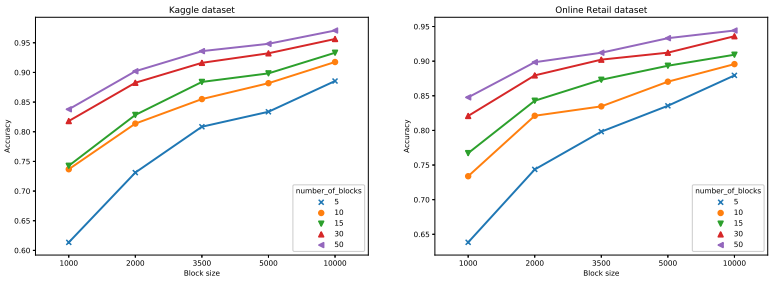


Fig. 3. Accuracy changes with different numbers of RSP data blocks and block size

Figure 4 shows the change of precision against the data block size in different numbers of RSP data blocks. The trends are same as the trends of accuracy because the two measures are essentially same in evaluating the findings of the true positive frequent itemsets in the entire data set from a set of selected RSP data blocks. In all cases, we can approach above 90% of precision with a small portion of the entire data. Based on these figures, we can assume that the bigger number of RSP data blocks is more important than the block size because of the popular frequent itemsets voting. More investigations are needed to find the reasons behind this phenomenon.

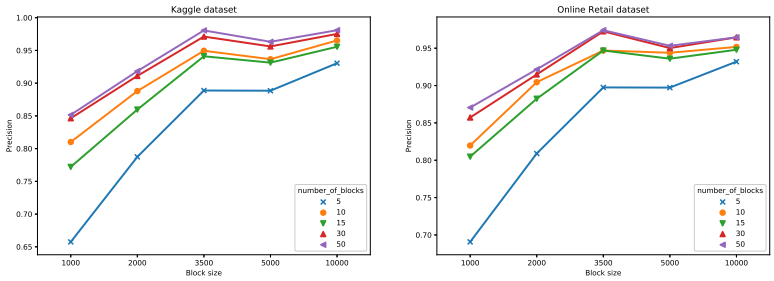


Fig. 4. Precision changes with different numbers of RSP data blocks and block size

In frequent itemsets mining with this approximate approach, Recall is an important measure. Figure 5 shows the change of recalls against the block sizes and different numbers of RSP data blocks. We can see that the trends of recall are different from the trends of accuracy and precision. The general trend is still that the recall increases as the block size and the number of RSP data blocks increase. However, the number of RSP data blocks has a bigger impact on recall. Generally speaking, recall is over 90% even with a few RSP data blocks of a small size, but it rarely arrived 100% in case when the local frequency threshold is equal to the global frequency threshold.

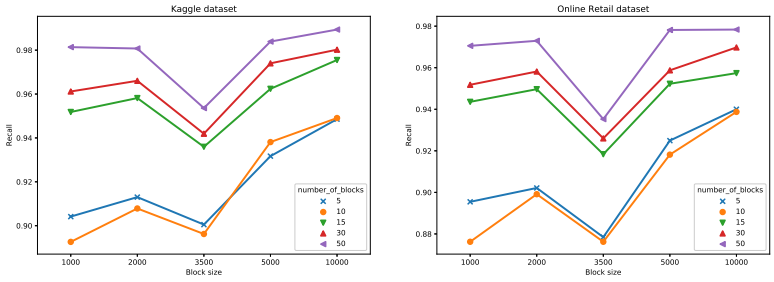


Fig. 5. Recall changes with different numbers of RSP data blocks and block size

5.4. Experiment Results With $0 < \epsilon < \theta$

To increase recall of popular frequent itemsets from the selected RSP data blocks, we set the local frequency threshold to $(\theta - \epsilon)$ and make $\epsilon > 0$. In this series of experiments, we investigate the change of recalls as ϵ increases from 0 to θ. Since the local frequency threshold is reduced, more frequent itemsets will be discovered for the same RSP data blocks. The number of the local frequent itemsets increases as ϵ increases, so the recall of the popular frequent itemsets will increase as well. However, as the number of the local frequent itemsets increases, the number of the false positive frequent itemsets also increases which decreases the accuracy and precision. Therefore, ϵ should be adjusted so that the popular frequent itemsets should not contain too many false positive frequent itemsets. In these experiments, we empirically investigated the value of ϵ which can make a better trade-off between recall, accuracy and precision.

In these experiments, we used the same parameter settings from Table 2. For each setting, we ran Algorithm 1 with a reduced local frequency threshold $(\theta-\epsilon)$, $\epsilon > 0$ to find the popular frequent itemsets from the selected RSP data blocks. Then, we compared the popular frequent itemsets with the frequent itemsets found from the entire data set with the global frequency threshold θ to compute the recall. For each set of parameters from Table 2 we tested several ϵ values. We computed the recall distributions of the popular frequent itemsets discovered with different values of ϵ for corresponding numbers of RSP data blocks and block sizes. Figure 6 shows the recall distributions for all parameter settings from Table 2 for Online Retail Data Set. The rows index is the numbers of RSP data blocks in the descending order as (50, 30, 15, 10, 5), and the columns indicate the sizes of RSP data blocks in transactions as (10000, 5000, 3500, 2000, 1000). Different ϵ values were used in each setting as shown in each display block. From the first row in Figure 6, we can see that for the settings of larger block sizes and more RSP data blocks, a small increase of ϵ, e.g., from 0.0005 to 0.001, results in a big increase of recall which can reach to 1. As the number of RSP data blocks is reduced, the larger ϵ is required to make the recall approach to 1, as shown in the left column of Figure 6. Therefore, as the block size and the number of RSP data blocks are reduced, a big increase of ϵ is required to increase the recall to 1. For example, in the right and bottom display block of Figure 6, ϵ value is 0.002 to make the recall approach 1. Since the global frequency threshold $\theta = 0.005$, the local frequency threshold is reduced with 40% from the global frequency threshold in order to obtain a 100% recall.

We calculated precision with empirically found ϵ value that guarantees the absence of the false negative frequent itemsets in the approximate solution, i.e. recall = 1 for most settings for both data sets. The precisions of the approximate results with different settings are shown in Figure 7. We can see the trends that precision increases sharply as the block size and the number of RSP data blocks increase. In the case of few small RSP data blocks, although the recall is high, the precision is low because a large number of false positive frequent itemsets exist due to a significant reduction of the local frequency threshold. As the block size and the number of RSP data blocks increase, ϵ decreases. Small ϵ allows to maintain low number of false positive frequent itemsets in the approximate solution, which makes the precision stay high. We can see that the precisions in both data sets are over 80% in the settings of large block size and number of RSP data blocks.

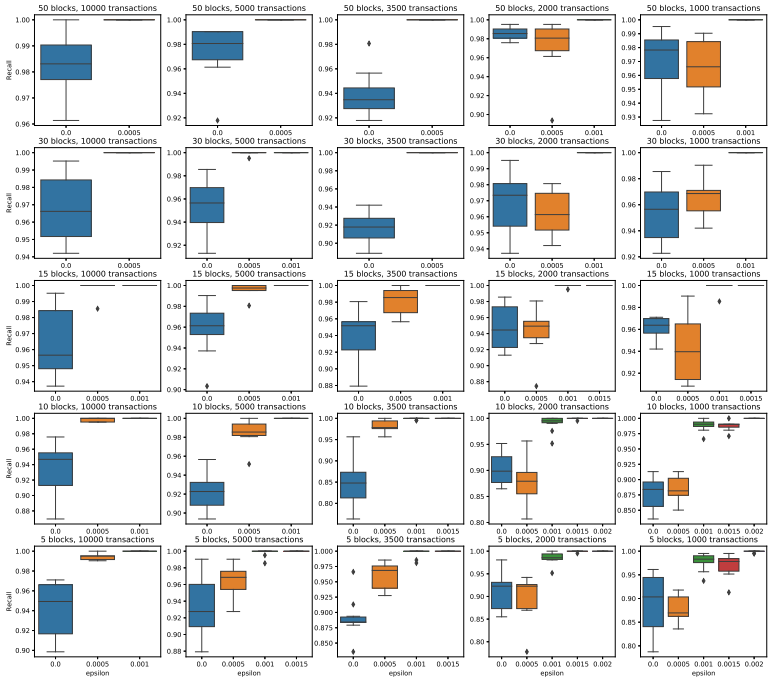


Fig. 6. Distributions of recalls on the different settings of block sizes, numbers of RSP data blocks and ϵ values

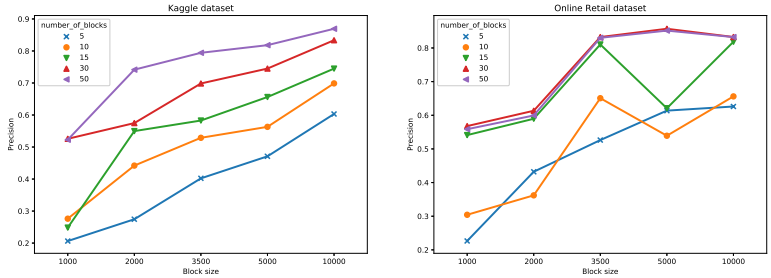


Fig. 7. Precision of the approximate solution with reduced frequency threshold changes with different numbers of RSP data blocks and block sizes

6. Conclusions and future work

In this paper, we have presented a new approach for mining approximate frequent itemsets based on a random sample partition of a big transaction database. We have shown that using the RSP data model for mining frequent itemsets can be very beneficial when the amount of data is very large and traditional single machine or distributed and parallel approaches are no longer able to process it. In this work, we introduced an algorithm for approximate frequent itemsets mining and experimentally showed that the algorithm is able to produce high-accurate frequent itemsets with random sample data blocks, and capable of discovering of all frequent itemsets from the entire data set which is achieved by parameter ϵ. The proposed approach is highly suitable for a distributed architecture and can be effectively run on a computing cluster.

For the further work, we will carry out theoretically statistical analysis on the quality of the outputs of the proposed algorithm and investigate an automatic way of estimating the parameter ϵ. Besides, we are going to implement a parallel version of the algorithm on a cluster and conduct experiments on big data sets in terabyte scale.

Acknowledgments. This research was supported by the National Natural Science Foundation of China under Grant 61972261.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of SIGMOD (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of VLDB (1994)
3. Annie, L., Kumar, A.: Market basket analysis for a supermarket based on frequent itemset mining. IJCSI International Journal of Computer Science Issues 9(3), 257–263 (2012)
4. Chakaravarthy, T., Pandit, V., Sabharwal, Y.: Analysis of sampling techniques for association rule mining. In: ICDT '09 Proceedings of the 12th International Conference on Database Theory. pp. 276–283 (2009)
5. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the CACM. pp. 107–113 (2004)
6. Grahne, G., Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In: Proceedings of the CEUR Workshop Proceedings. Melbourne, FL (2003)
7. Grahne, G., Zhu, J.: Reducing the main memory consumptions of fpmax* and fpclose. In: Proceedings of the CEUR Workshop Proceedings. Brighton, UK (2004)
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 19th ACM International Conference on Management of Data (SIGMOD). Dallas, TX, USA (2000)
9. Jiang, H., Meng, H.: A parallel fp-growth algorithm based on gpu. In: 2017 IEEE 14th Int. Conf. E-bus. Eng. pp. 97–102 (2017)
10. Kazienko, P., Pilarczyk, M.: Data mining for inventory item selection with cross-selling considerations. New Generation Computing 26, 227–244 (2008)
11. Moens, S., Aksehirli, E., Goethals, B.: Frequent itemset mining for big data. In: 2013 IEEE International Conference on Big Data (2013)
12. Prajapati, D., Garg, S., Chauhan, N.: 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)

13. Racz, B.: An fp-growth variation without rebuilding the fp-tree. In: Proceedings of the CEUR Workshop Proceedings. Brighton, UK (2003)
14. Riondato, M., DeBrabant, J., Fonseca, R., 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. Riondato, M., Upfal, E.: Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. In: ECML PKDD: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 25–41 (2012)
16. Salloum, S., Huang, J., He, Y.: Random sample partition: A distributed data model for big data analysis. IEEE Transactions on Industrial Informatics 15(11), 5846 – 5854 (2019)
17. Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. In: VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases. p. 432–444 (1995)
18. Schmidt-Thieme, L.: Algorithmic features of eclat. In: Proceedings of the Workshop Frequent Item Set Mining Implementations. Brighton, UK (2004)
19. Toivonen, H.: Sampling large databases for association rules. In: VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases. p. 134–145 (1996)
20. Wong, R., , Fu, A., Wang, K.: Mining evolving association rules for e-business recommendation. Data Mining and Knowledge Discovery 11, 81–112 (2005)
21. Zaki, M., Gouda, K.: Fast vertical mining using diffsets. In: Proceedings of the 9th ACM International Conference on Knowledge Discovery and DataMining. pp. 326–335. Washington, DC, USA (2003)
22. Zaki, M., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining. Newport Beach, CA, USA (1997)

Timur Valiullin received the master's degree from the Institute of Computational Mathematics and Information Technologies of Kazan Federal University, Kazan, Russia, in 2018. Currently He is a Ph.D. candidate in Computer Science at Shenzhen University, Shenzhen, China.

Joshua Zhexue Huang received the Ph.D. degree in Computer Science from The Royal Institute of Technology, Stockholm, Sweden, in 1993. He is currently a Distinguished Professor of College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, China. He is also the Director of Big Data Institute in Shenzhen University, and the Deputy Director of National Engineering Laboratory for Big Data System Computing Technology, Shenzhen, China.

Chenghao Wei obtained his B.Eng. degree in electronic engineering from the Wuhan University of Science and Technology, Wuhan, China, 2009. He received his M.Eng. degree with distinction from the Department of Electrical Engineering and Electronics in the University of Liverpool, U.K., 2010. Thereafter, he continued his study as a Ph.D. candidate in the same department for oil-immersed power transformer fault diagnosis. Since then, he joined the Big Data Institute of Shenzhen University as a research assistant. His current research focuses on machine learning, big data application, pattern recognition, data analysis.

Jianfei Yin received the Ph.D. degree in Computer Science from the South China University of Technology, Guangzhou, China, in 2005. He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China.

Dingming Wu is Associate Professor of College of Computer Science & Software Engineering at Shenzhen University, China. She received her Bachelor degree in Computer Science at Huazhong University of Science and Technology, Wuhan, China in 2005, and a Master degree in Computer Science at Peking University, Beijing, China in 2008. She received a Ph.D. degree in Computer Science at Aalborg University, Denmark, in 2011. Her research concerns data management, query processing, information retrieval, and data mining.

Iuliia Egorova received the master's degree from the Institute of Computational Mathematics and Information Technologies of Kazan Federal University, Kazan, Russia, in 2020.

Received: January 24, 2020; Accepted: July 15, 2020.