Int. J. Mach. Learn. & Cyber. (2015) 6:875–882
DOI 10.1007/s13042-015-0345-6

ORIGINAL ARTICLE

Iterative sampling based frequent itemset mining for big data

Xian Wu[1] · Wei Fan[2] · Jing Peng[3] · Kun Zhang[4] · Yong Yu[1]

Received: 27 October 2014 / Accepted: 4 March 2015 / Published online: 20 March 2015
© Springer-Verlag Berlin Heidelberg 2015

Abstract Frequent pattern mining attracts extensive research interests over the past two decades: including mining frequent item sets from transactions, extracting frequent sequences from bio-arrays and detecting common subgraph from molecular structures. In the era of big data, the explosive data volume brings new challenges to frequent pattern mining: (1) Space complexity: both input data, intermediate results and the outputted patterns could be too large to fit into memory which prevents many algorithms from executing; (2) Time complexity: many existing approaches rely on exhaustive search or complicated data structures to mine frequent patterns which prove to be inapplicable for big data. To deal with these two challenges. we propose ISbFIM, an Iterative Sampling based Frequent Itemset Mining method. Rather than process the entire data set at once, ISbFIM samples computationally-manageable subsets and extracts frequent itemsets from these subsets. By repeating this process for a sufficient number of times, we can guarantee both theoretically and empirically that the frequent itemsets can be enumerated without running into a combinatorial explosion. ISbFIM can be easily parallelized and applied to mine item sets, sequences or structures. We implement a Map-Reduce version of ISbFIM to demonstrate its scalability on big data.

Keywords Frequent itemset mining · Big data · Iterative sampling · Parallelization · Map-reduce

✉ Xian Wu
wuxian@apex.sjtu.edu.cn

Wei Fan
wei.fan@gmail.com

Jing Peng
pengj@mail.montclaire.edu

Kun Zhang
kzhang@xula.edu

Yong Yu
yyu@apex.sjtu.edu.cn

1 Shanghai Jiao Tong University, Shanghai, China

2 Baidu Research Big Data Lab, Sunnnyvale, CA, USA

3 Department of Computer Science, Montclair State University, Montclair, USA

4 Department of Computer Science, Xavier University of Lousiana, New Orleans, USA

1 Introduction

Frequent pattern mining has been heavily studied in the past two decades. The main interest for pattern mining includes item sets, subgraphs, and sequences etc. For each type of patterns, tremendous progresses have been made, like Apriori [2] and FP-Growth [13] for frequent item set mining, [3] for frequent sequences mining, and [17] for common subgraph extraction.

In the era of big data, the data volume grows the at an explosive speed. According to a recent report from comScore,[1] 19.6 billion explicit core searches were conducted in January 2014. The massive web data brings new challenges to frequent pattern mining: (1) The input data is so large that multiple scanning over the entire set is computational unaffordable. Therefore, the exhaustive search based algorithms like Apriori are prohibited from practical use; (2) The intermediate results and the final outputted

[1] http://www.comscore.com/Insights/Press_Releases/2014/2/comScore_Releases_January_2014_US_Search_Engine_Rankings.

patterns could be too large to fit into memory. As a result, the tree structure employed by FP-Growth can exhaust all the memory and halt the frequent pattern extraction process. A straight-forward solution to both problems is to increase the threshold of frequency support which could reduce the scale of patterns. However, in terms of feature extraction, low support patterns prove to be discriminative in certain tasks [12]. Furthermore, in case of big data, even a relatively higher threshold can still return a large number of patterns.

To deal with big data, many approaches have been proposed to parallelize the A priori [1, 10, 19, 24, 32] and FP-Growth [18] algorithms. They usually adopt the following two manners: (1) Distributing the computing workload while restoring the data in a shared memory environment; (2) Apply the Map-reduce framework to partition both the computing workload and the data. The former type requires a large shared memory to restore intermediate generated and final outputted itemsets which is in-practical for big data, e.g. web search log; While the latter one could still assign unaffordable workload to some nodes. On one hand, frequent item set mining is NP-hard [30]; On the other hand, if the support is set too low, the program simply may not finish since the virtual memory can be exhausted [12].

To scale frequent itemset mining to big data, we propose an Iterative Sampling based Frequent Itemset Mining method (ISbFIM). In order to deal with large input, ISbFIM samples computationally-manageable subsets and extracts frequent patterns from these subsets with a higher threshold. Since the input volume is reduced and the support threshold is increased, both the time and space complexity of frequent pattern mining on each subset is computational affordable. Importantly, the time complexity for processing each subset is the same which avoids over-assigning workload to a single node. By iterating this sample-and-extract process many times, we can guarantee that most frequent patterns in terms of the entire data set have been enumerated. We derived a termination bound to guide how to set the size of each sample and determine the rounds of iterations. This proposed ISbFIM can not only deal with large scale data but also is suitable for parallel implementation. We have validated the theoretical termination bound and demonstrated the effectiveness of the proposed approach on multiple data sets.

2 Methodology

In this section, we will describe the proposed ISbFIM framework. Since ISbFIM mainly targets to mine frequent item sets, we refer patterns to itemsets in this paper.

Let N denote the number of input transactions and α denote the threshold of frequency support. If we extract the frequent item sets from these N transactions, the scale of the patterns returned is given by

$$O(N^{(1-\alpha)N})$$

where $N > 3$ and α is the frequency support threshold in percentage [29]. In case of big data, e.g. web search log, the full set of extracted patterns are too large to fit into memory. To scale down the pattern volume, we can either reduce the data size N or decrease the threshold α. Here we take relatively smaller samples from the entire data set and extract patterns with a higher threshold β within each sample. As a result, the number of retrieved patterns in each iteration will be much smaller. Thus they can be processed using reasonable computing resources. More importantly, the patterns of low frequency that are below global threshold α can be included as well, as the higher frequency in a sample data set could be much smaller when considering over the entire corpus. The proposed algorithm ISbFIM is summarized in Algorithm 1.

Algorithm 1: ISbFIM - Iterative Sampling based Frequent Pattern Mining

```
Input : 1: The Set of Transactions $D$
        2: Global Support Threshold $\alpha$
        3: The Set of Sampled Transactions $D'$
        4: Local Support Threshold $\beta$
        5: Number of Iterations $T$
Output: 1: Set of Result Patterns $X$
1 for $i \leftarrow 1$ to $T$ do
2     Let $D' = \{N'$ transactions randomly selected from $D$ without replacement$\}$;
3     Perform frequent item set extraction on $D'$ with the threshold $\beta$,
      $FP' = FIM(D', \beta)$;
4     Evaluate patterns in $FP'$, save the selected ones to $X$;
5     $X = X \cup EvaluateLocal(FP', D')$;
6 Evaluate patterns in $X$ on whole data set $D$, $X = EvaluateGlobal(X, D)$
```

In the main loop, the sample data set D' is randomly generated from D, (thus it permits parallel implementation as discussed in Sect. 3.2.2), and the frequent pattern extraction is performed with a higher threshold β. Then selected patterns are filtered according to some specific function *Evaluate* on D'. The functionality of *Evaluate* is defined by the corresponding application scenario. In case of discriminant feature selection [12], the *Evaluate* method can be defined as a fisher score filter. After a sufficient number of iterations, the loop terminates and all selected patterns are further filtered by evaluating on the whole data set D.

Through the above iterative sampling method, at each iteration, the scale of the patterns returned by the frequent pattern extraction algorithm is reduced. Let the size of a sample data set D' be $N' = \gamma N$, where $0 < \gamma \leq 1$. The scale of the problem at each iteration becomes

$$O((\gamma N)^{(1-\beta)\gamma N}).$$

Thus, the reduction in computation at each iteration is as follows:

$$\begin{aligned} RD &= N^{(1-\alpha)N} - (\gamma N)^{(1-\beta)\gamma N} \\ &\geq N^{(1-\alpha)N} - N^{(1-\beta)\gamma N} \\ &\geq N^{(1-\alpha)N} - N^{(1-\alpha)\gamma N} \\ &= N^{(1-\alpha)N} - N^{(1-\alpha)\gamma N} \\ &= N^{(1-\alpha)N} - (N^{(1-\alpha)N})^{\gamma}, \end{aligned} \tag{1}$$

where the first inequality follows from $\gamma N \leq N$, and the second inequality follows from $\alpha \leq \beta$. Let

$$C = N^{(1-\alpha)N}.$$

Then, the reduction in computation is at least

$$RD \geq C - C^{\gamma}. \tag{2}$$

As γ decreases (i.e., the sample set decreases), so does C^{γ}. Thus, the reduction in computation can be substantial. On the other hand, if a sample set is too small, it may bring in noisy patterns in term of the entire data set. Therefore, in practise, γ is chosen so that frequent pattern mining can be accomplished using available computational resources.

2.1 Required iterations

The local frequency β is usually set higher than the global frequency α. For example, suppose $N = 10{,}000$ and $\alpha = 1\,\%$. If we set $\gamma = 0.01$, thus $N' = \gamma N = 100$, and $\beta = \alpha$, the local frequency threshold is only 1. As a result, all possible combinations will be returned as candidate patterns. For those that appear only once or twice, there will not be sufficient examples to predict their frequency on the entire data set. By setting β higher, e.g., $\beta = 10\ \%$, we will have more data as evidence to evaluate the pattern's occurrence. The local frequency β should be chosen so that the following condition

$$\alpha N \geq \beta N'$$

is met.

After choosing the sample size N' and β according to available computational resources, the next step is to determine the required number of iterations T. This problem can be formulated as follows. Let FIM (D', β) denote the set of patterns extracted from one sample set D'. Also let η denote the coverage threshold. If $\eta = 0.9$, the problem is to determine the number of iterations T so that 90% of frequent patterns ($\geq \alpha$) in D will be selected as local frequent ones ($\geq \beta$) by ISbFIM. By choosing a proper η, solving the following inequality obtains the desired number of T.

$$\frac{|\ \mathrm{FIM}\ (\mathrm{D}, \alpha) \cap \bigcup_{\mathrm{i}=1}^{\mathrm{T}}\ \mathrm{FIM}\ (\mathrm{D}', \beta)|}{|\ \mathrm{FIM}\ (D, \alpha)|} \geq \eta \tag{3}$$

In the following, we discuss how to solve the above inequality. In the main loop of Algorithm 1, a data set D' is randomly sampled without replacement from D and frequent pattern extraction is performed on D'. For any pattern x_i with a frequency α in data set D, the probability that x_i will not be selected as a local frequent pattern in one sample set D' can be calculated as follows:

$$\begin{aligned} P(x_i \notin \ \mathrm{FIM}\ (D', \beta)) &= P(\beta_x < \beta) \\ &= \sum_{i=0}^{\beta N'-1} P(\beta_x N' = i) \\ &= \sum_{i=0}^{\beta N'-1} \frac{\binom{\alpha N}{i} \cdot \binom{N-\alpha N}{N'-i}}{\binom{N}{N'}} \end{aligned} \tag{4}$$

where β_i denotes the support in percentage of the pattern x_i in D'.

Let $p = P(x_i \notin \ \mathrm{FIM}\ (D', \beta))$. If we iterate $\lceil \frac{1}{1-p} \rceil$ times, the probability that this pattern x_i is selected as a frequent pattern is at least

$$1 - p^{\frac{1}{1-p}} \simeq 1 - 1/e = 0.632.$$

Thus, if there are L unique patterns in D with frequency exceeding α, i.e., FIM $(D, \alpha) = \{p_1, p_2, \ldots, p_L\}$, then after $\lceil \frac{1}{1-p} \rceil$ iterations, the average number of distinct frequent patterns selected will be (at least) about $0.632L$.

In order for ISbFIM to mine at least 90 % of L patterns, we first solve the following to obtain t

$$1 - p^{t\frac{1}{1-p}} = 0.9.$$

This implies that $p^{t\frac{1}{1-p}} = 0.1$. Taking logarithm, we obtain $t \ln(p^{\frac{1}{1-p}}) = \ln(0.1)$. Since

$$p^{\frac{1}{1-p}} \simeq e^{-1},$$

we have $t \simeq -\ln(0.1) = 2.3$. Therefore, after $T = \lceil \frac{2.3}{1-p} \rceil$ iterations, on average we will obtain at least 90 % of L frequent patterns in D.

Notice that since N, N', α, and β are chosen a priori, one can calculate the required number of iterations T. For example, if $N = 2{,}000$, $\alpha = 0.1$, $N' = 50$, $\beta = 0.2$, and $\eta = 0.9$, we have $p = 0.977$ according to Eq. (4). Thus, the required number of iterations is $T = \lceil \frac{2.3}{0.023} \rceil = 100$.

2.2 Evaluate method

In Algorithm 1, we employ two methods *EvaluateLocal* and *EvaluateGlobal* to filter non-informative patterns both in the local and global mining. The functionality of *EvaluateLocal* and *EvaluateGlobal* is defined by the actual application scenario.

2.2.1 Unlabeled data sets

For unlabelled data sets, the task is to select the global frequent patterns. The *EvaluateLocal* method will perform nothing but send out all local frequent patterns; On the other hand, the *EvaluateGlobal* method will aggregate and de-dup all the local frequent patterns and calculate their frequency in terms of the entire data sets. For those that fail to reach the global supporting threshold, *EvaluateGlobal* will abandon them and keep the remaining ones as the final results.

2.2.2 *Labeled data sets*

In case of discriminant frequent pattern mining, the first step decides if a frequent pattern is capable of providing discriminative separation locally on the sample set D'. The second step determines if the locally discriminant patterns are also discriminant globally on the entire data set D. In the following, we use *positive* and *negative* to denote the binary label on each transaction. For any frequent pattern x in the sample set D', N_0^x and N_1^x represent the number of positive and negative documents containing x, respectively. The following criterion is employed to locally measure x.

$$IG(x) = H(D') - H(D'|x), \tag{5}$$

where

$$H(D) = -\sum_{i\in\{0,1\}} \frac{N_i}{N'} \log \frac{N_i}{N'}$$
$$H(D|x) = -\frac{N^x}{N'} \sum_{i\in\{0,1\}} \frac{N_i^x}{N^x} \log \frac{N_i^x}{N^x} - \frac{N^{\bar{x}}}{N'} \sum_{i\in\{0,1\}} \frac{N_i^{\bar{x}}}{N^{\bar{x}}} \log \frac{N_i^{\bar{x}}}{N^{\bar{x}}}.$$

Those patterns satisfying $IG(x) > Threshold$ will be regarded as "locally discriminant". However, as the sample set D' is generated randomly, these locally discriminant patterns may not necessarily be discriminant when measured on the entire data set. Therefore, for all locally selected patterns, their ability will be re-evaluated on the entire data set D.

2.3 Discussion

The proposed ISbFIM algorithm replaces the one pass algorithm that extracts all frequent patterns at once, with an iterative sampling method. We have shown that with sufficient iterations, most desired number of frequent patterns in the entire data set can be extracted by the proposed ISbFIM algorithm. Besides, many patterns that are below the threshold α can be enumerated by ISbFIM as well (Sect. 3.2.1). The lowest possible frequency can be:

$$\frac{\beta N'}{N},$$

which is much smaller than α.

Table 1 Seven data sets used in experiments

Name	Number of Instance	Source
Blitzer07	2000	[7]
Pang02	2000	[23]
Pang04	10,660	[22]
Jindal08	100,000	[16]
Mushroom	8124	http://fimi.ua.ac.be/data/
Retail	88,162	http://fimi.ua.ac.be/data/
IBM	100,000	http://fimi.ua.ac.be/data/

There is one assumption in ISbFIM: in order to extract a globally frequent pattern, it needs to pass the *Evaluate* method in at least one local sample set.

Since each iteration process is independent, the proposed algorithm can be easily implemented in parallel using the straight forward "divide and process" and Map-Reduce model. Therefore, this algorithm can be accelerated easily.

3 Experiment

We evaluate the proposed ISbFIM algorithm from two perspectives: (1) We examine how precise the termination criterion is. That is, we examine how well our formal analysis matches with experimental results; (2) We implement ISbFIM according to the Map-Reduce model and evaluate its performance in a parallel environment.

3.1 Data sets

We use seven data sets for evaluation which are listed in Table 1. The first four consists of customer reviews which are labeled either positive or negative according to their sentiment orientation; The rest three are selected from frequent itemset mining dataset repository[2] which are unlabeled data sets.

3.2 Experimental results

3.2.1 *Termination criterion*

In this experiment, we examine the accuracy of the required number of iterations T for ISbFIM to terminate. As described in Sect. 2.1, if the sampling process in ISbFIM is performed over T times, more than η percents of frequent patterns will be extracted. Take the data set *Pang02* for example, if we set the sample size $N' = 100$, the global

[2] http://fimi.ua.ac.be/data/.

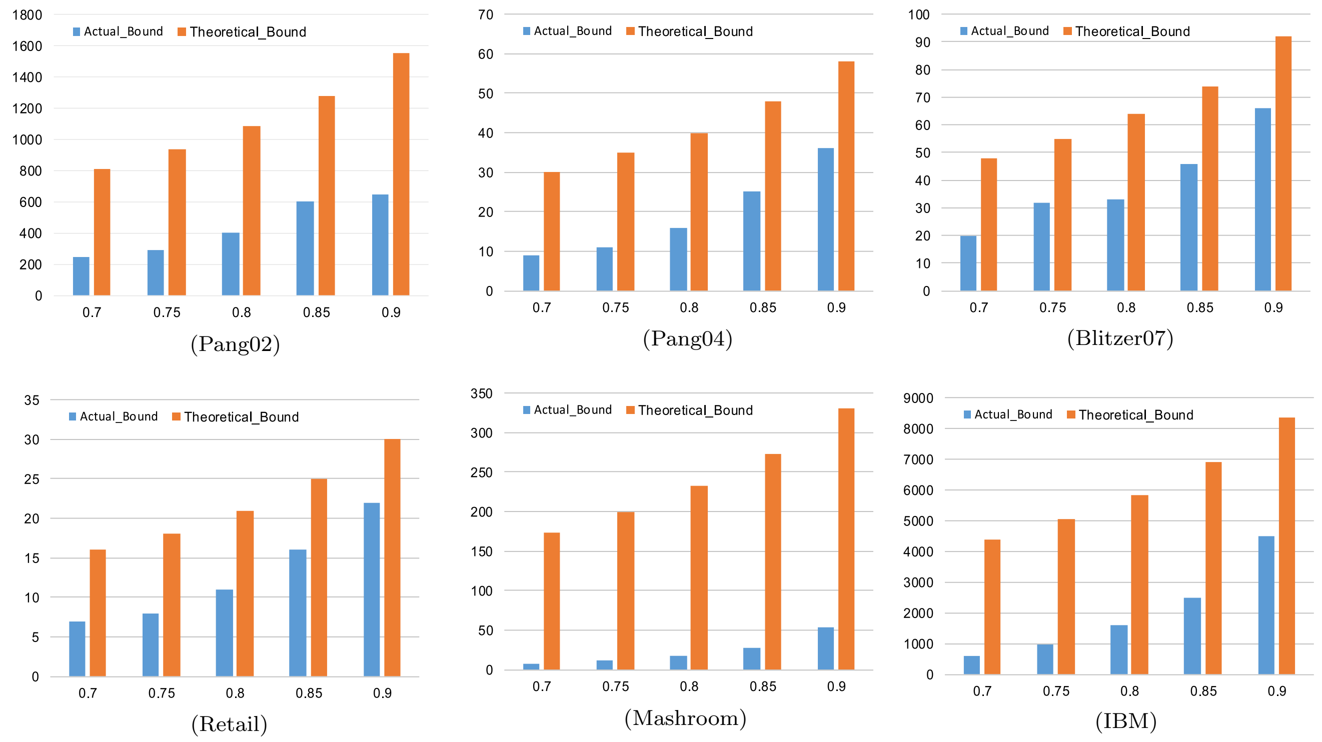

Fig. 1 Comparison of analytical and actual termination bounds on six data sets

frequency $\alpha = 0.1$, the local frequency threshold $\beta = 0.2$ and the percent $\eta = 0.7$, we can estimate the required iterations $T = 810$ via Eq. (3). In this manner, instead of directly extracting patterns from the entire data set with the threshold 10 %, we iteratively sample 100 documents and extract patterns with a higher threshold 20 % for 810 times. We can guarantee that more 70 % frequent patterns can be extracted. As a result, the scale of frequent pattern extraction is reduced significantly by using smaller sample size but higher support.

We calculate the theocratical bound T and compare them the actual bound. First, we set the sample size $N' = \frac{N}{20}$ and local frequency $\beta = 2\alpha$. Next, we let η equal tox 0.7, 0.75, 0.8, 0.85 and 0.9, respectively and calculate the corresponding T. We run the proposed ISbFIM algorithm on six data sets *Blitzer*, *Pang02*, *Pang04*, *Retail*, *Mushroom* and *IBM* and record the actual times of iterations when η of frequent patterns are extracted. Please note that these six data sets diff in the density of frequent item sets. Therefore we employ different global support thresholds to extract frequent item sets. For example, *Pang02* is a dense set thus we set the threshold to be 10% while *IBM* is a sparse data set thus we set the threshold to be 0.1%.

Figure 1 summarizes the analytical and actual iteration times. In all settings of η, the theoretical estimate is bigger than the actual measurement. Therefore, if the sampling process is performed for T times, more than η percent of frequent patterns can be extracted. Thus, the analytical estimate is proven to be correct in practise.

3.2.2 *Parallelization*

It is easy to implement ISbFIM in parallel. We implement ISbFIM according to the Map-Reduce model [11]. Figure 2 illustrates the detailed flow. The component *Sample Sets Generator* samples T (Calculated Theoretical Bound) data sets from the whole corpus and send them to mappers. Each mapper receives a single sample set at a time and extract local frequent patterns from it. Then the mapper will enumerate all the extracted patterns and send out the ones that pass the *Evaluate* filter. The reducer merges any duplicate patterns.

We implemented the parallel ISbFIM with Hadoop (Version 0.23)[3] and ran it on the data set *Jindal08* with the following setting: $N' = 2000, \alpha = 0.001, \beta = 0.004$ and $\eta = 0.9$. The number of iterations is 2489. We virtualize 16 nodes on an IBM x3950 sever and different nodes are configured with a dual core 2.9G CPU and 2G memory and is connected via gigabyte ethernet. The experiment is performed on 1, 4, 8, 16 nodes, respectively and the running time is displayed in Fig. 3. From this figure, we can

[3] http://hadoop.apache.org/.

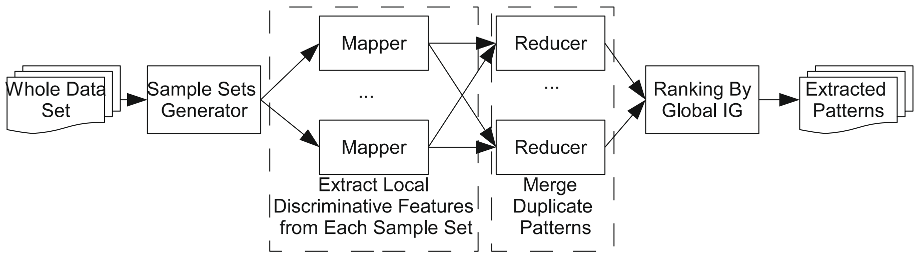

Fig. 2 The parallel implementation of ISbFIM

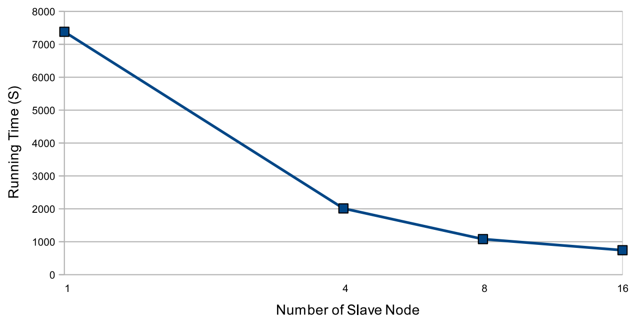

Fig. 3 Running time of ISbFIM in a parallel environment

see that the running time is approximately inversely proportional to the size of the cluster, demonstrating that ISbFIM is suitable for parallel acceleration.

4 Related works

The proposed work spans across several important research areas in frequent pattern mining: (1) distributed frequent pattern mining, (2) discriminant frequent pattern mining and (3) frequent pattern sampling.

4.1 Distributed frequent pattern mining

Anastasiu et al. [5] summarized existing works on mining frequent patterns from big data; among which, Park et al [1], Agrawal and Shafer [24, Cheung et al. [10] and Zaki et al. [32], etc. proposed parallelized version of Apriori algorithm. These approaches either requires shared memory computing environment or multiple scanning over the entire data set. Therefore they are not suitable for big data, e.g. web search log.

Li et al. [18] proposed PFP which parallelizes FP-Growth [13] with the Map-Reduce framework. In the "Map" stage, they divide the entire data set into small subsets and perform FP-Growth on each subset; In the "Reduce" stage, they distribute the built FP-trees according to the identities of items. However, in case of big data, PFP could allocate excessive data on one reducer which will halt the whole process. Therefore, [18] suggests use PFP to only select top K frequent patterns in case of big data. As to the proposed ISbFIM, each node takes equivalent and manageable work load and the failure on one node will not halt the whole process. Other parallelized version FP-growth like [31] require shared memory computing environment thus not practical for map-reduce framework. Besides, the shared memory could be exhausted in case of big data.

Hill et al. [14] developed a frequent subgraph mining algorithm over Map-Reduce which can handle the cases when the graph database cannot be loaded into main memory; Aridhi et al. [6] further provided a density based partition algorithm to enhance the default partition of general Map-Reduce framework. In this manner, the computing workload can be evenly distributed to each node which prevents certain nodes from running into exhausting situation. The difference between Aridhi et al. [6], Hill et al. [14] and our approach is the targeted problem. [6] targets the frequent subgraph mining while the proposed approach mainly targets the frequent item sets extraction. [20] enabled querying subgraph over a large database with Map-Reduce framework which focused on searching subgraphs instead of mining frequent subgraphs.

4.2 Discriminant frequent pattern mining

Discriminant frequent pattern mining targets to detect the frequent patterns that are discriminative given the labeling on the documents. Previous methods such as [8] that mine discriminant patterns typically employ a two-step batch process: (1) extracts frequent patterns whose support or frequency is above some reasonably large threshold, and (2) performs feature selection. However, when the frequency threshold is set too high, such approach cannot find those highly discriminant patterns whose frequency is lower than the given threshold. When the threshold is too low, candidate enumeration cannot finish before virtual memory is exhausted.

DDPMine [9] performs a branch-and-bound search for mining discriminant patterns without generating the complete pattern set. Since DDPMine needs to enumerate the whole data set to mine a single pattern, it's not suitable for big data due to the large volume of input data and output patterns.

The MbT approach (Model based search Tree), proposed by [12], is closest to the proposed method in this paper. MbT employs a "divide and conquer" process to combine frequent pattern mining and discriminant pattern recognition in a single framework. As a result, the complexity can be reduced significantly. However, MbT cannot not be easily paralleled which prohibits its application on big data.

4.3 Frequent pattern summary

To reduce the number of acquired frequent patterns, approaches such as Yan et al. [27], Thoma et al. [25], Wang and Parthasarathy [26] and Jin et al. [15] aim to find a succinct set of patterns that can work as well as the whole set in classification or clustering tasks. In the case that a pattern is subgraph [28] and [4] use structure similarity or random walk to find a set of patterns with high discriminant scores.

The above approaches and ISbFIM differ in targeted outputs: these approaches aim to detect representative frequent patterns, while ISbFIM targets to extract and scan frequent patterns. Besides, those approaches are tailored for specific tasks (clustering, classification, structure mining etc), therefore it is hard to apply them to general tasks.

Minato et al. [21] proposed a fast item set enumeration algorithm to find the minimum support to satisfy the LAMP condition. They target to find a maximal support to satisfy a given threshold condition. Their approach is customized for LAMP problem thus it is not easy to extend it to general frequent item set mining problem.

5 Conclusion

In this paper, we introduce ISbFIM to tackle two big data challenges in frequent pattern mining: (1) Efficiency: the large volume of input data prevents exhaustive search or complicated data structures from practise use; (2) Scalability: the massive intermediate and outputted patterns cause out-of-memory problems for many state-of-art approaches. In order to deal with a very large pool of pattern candidates mined from big data, we developed an iterative sampling procedure and derived a termination bound. This algorithm not only reduces the problem scale for each task but also is natural for parallel implementation. We have validated the theoretical termination bound and demonstrated the effectiveness of the proposed approach on multiple data sets.

References

1. Agrawal R, Shafer JC (1996) Parallel mining of association rules. IEEE Trans Knowl Data Eng 8:962–969
2. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pp 487–499
3. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of ICDE '95, pp 3–14
4. Al Hasan M, Zaki MJ (2009) Output space sampling for graph patterns. Proc VLDB Endow 2:730–741
5. Anastasiu DC, Iverson J, Smith S, Karypis G (2014) Big data frequent pattern mining. In: Aggarwal CC, Han J (ed) Pattern Frequent. Publishing, Mining, Springer International, pp 225–259
6. Aridhi S, d'Orazio L, Maddouri M, Nguifo EM (2015) Density-based data partitioning strategy to approximate large-scale subgraph mining. Inf Syst 48:213–223
7. Blitzer J, Dredze M, Pereira F (2007) Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of ACL '07. Prague, Czech Republic, Association for Computational Linguistics, pp 440–447
8. Cheng H, Yan X, Han J, wei Hsu C (2007) Discriminative frequent pattern analysis for effective classification. In: International Conference on Data Engineering, pp 716–725
9. Cheng H, Yan X, Han J, Yu PS (2008) Direct discriminative pattern mining for effective classification. In: Proceedings of ICDM '08. IEEE Computer Society, Washington, DC, USA, pp 169–178
10. Cheung DW, Han J, Ng VT, Fu AW, Fu Y (1996) A fast distributed algorithm for mining association rules. In: Proceedings of the fourth international conference on on Parallel and distributed information systems. IEEE Computer Society, Washington, DC, USA, pp 31–43
11. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51:107–113
12. Fan W, Zhang K, Cheng H, Gao J, Yan X, Han J, Yu P, Verscheure O (2008) Direct mining of discriminative and essential frequent patterns via model-based search tree. In: Proceeding of KDD '08. ACM, New York, NY, USA, pp 230–238
13. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '00, pp 1–12, doi:10.1145/342009.335372
14. Hill S, Srichandan B, Sunderraman R (2012) An iterative mapreduce approach to frequent subgraph mining in biological datasets. In: Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine, ACM, New York, NY, USA, BCB '12, pp 661–666, doi:10.1145/2382936.2383055
15. Jin R, Abu-Ata M, Xiang Y, Ruan N (2008) Effective and efficient itemset pattern summarization: regression-based approaches. In: Proceeding of KDD '08. ACM, New York, NY, USA, pp 399–407
16. Jindal N, Liu B (2008) Opinion spam and analysis. In: Proceedings of WSDM '08. ACM, New York, NY, USA, pp 219–230
17. Kuramochi M, Karypis G (2001) Frequent subgraph discovery. In: Proceedings of the 2001 IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, ICDM '01, pp 313–320
18. Li H, Wang Y, Zhang D, Zhang M, Chang EY (2008) Pfp: Parallel fp-growth for query recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems, ACM, New York, NY, USA, RecSys '08, pp 107–114, DOI 10.1145/1454008.1454027
19. Lin MY, Lee PY, Hsueh SC (2012) Apriori-based frequent itemset mining algorithms on mapreduce. In: Proceedings of the 6th International Conference on Ubiquitous Information

Management and Communication, ACM, New York, NY, USA, ICUIMC '12, pp 76:1–76:8

20. Luo Y, Guan J, Zhou S (2011) Towards efficient subgraph search in cloud computing environments. In: Proceedings of the 16th International Conference on Database Systems for Advanced Applications, Springer-Verlag, Berlin, Heidelberg, DASFAA'11, pp 2–13, http://dl.acm.org/citation.cfm?id=1996686.1996690
21. Minato S, Uno T, Tsuda K, Terada A, Sese J (2014) A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration. Machine Learning and Knowledge Discovery in Databases—European Conference, ECML PKDD 2014, Nancy, France, September 15–19, 2014. Proceedings, Part II, pp 422–436
22. Pang B, Lee L (2004) A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of ACL '04, Association for Computational Linguistics, Stroudsburg, PA, USA
23. Pang B, Lee L, Vaithyanathan S (2002) Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing -, vol 10. Association for Computational Linguistics, Stroudsburg, PA, USA, pp 79–86
24. Park JS, Chen MS, Yu PS (1995) Efficient parallel data mining for association rules. In: Proceedings of CIKM '95. ACM, New York, NY, USA, pp 31–36
25. Thoma M, Cheng H, Gretton A, Han J, peter Kriegel H, Smola A, Song L, Yu PS, Yan X, Borgwardt K (2009) Near-optimal supervised feature selection among frequent subgraphs. In. In SIAM Int'l Conf. on Data Mining
26. Wang C, Parthasarathy S (2006) Summarizing itemset patterns using probabilistic models. In: Proceedings of KDD '06. ACM, New York, NY, USA, pp 730–735
27. Yan X, Cheng H, Han J, Xin D (2005) Summarizing itemset patterns: a profile-based approach. In: Proceedings of KDD '05. ACM, New York, NY, USA, pp 314–323
28. Yan X, Cheng H, Han J, Yu PS (2008) Mining significant graph patterns by leap search. In: Proceedings of SIGMOD '08. ACM, New York, NY, USA, pp 433–444
29. Yang G (2004) The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, KDD '04, pp 344–353
30. Yang G (2006) Computational aspects of mining maximal frequent patterns. Theor Comput Sci 362(1–3):63–85
31. Zaïane OR, El-Hajj M, Lu P (2001) Fast parallel association rule mining without candidacy generation. In: Proceedings of the 2001 IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, ICDM '01, pp 665–668
32. Zaki M, Parthasarathy S, Ogihara M, Li W (1997) Parallel Algorithms for Discovery of Association Rules. Data Mining and Knowledge Discovery pp 343–373, doi:10.1023/A:1009773317876