7P-7

Mining the *k*-Most Interesting Frequent Patterns

Tran MINH QUANG (*) Shigeru OYANAGI (*) Katsuhiro YAMAZAKI (*)
Ritsumeikan University – Graduate School of Science and Engineering

Abstract— **The idea of mining top-k frequent patterns releases users from the difficulty of guessing for a suitable minimum support threshold when mining for a desired number of frequent patterns. This idea is based upon an algorithm for mining frequent patterns without a minimum support threshold, but with a k number of strongest patterns. However, pursuing this idea, algorithms have to identify the minimum support threshold automatically by themselves, which causes slowing down their performance. In this paper, we propose an explorative mining algorithm, called ExMiner, to mine k-most interesting frequent patterns from a large scale dataset efficiently and effectively. The experiments on very large synthetic datasets reveal that our method is superior to the existing ones.**

Keywords- data mining; frequent patterns mining; k-most interesting frequent patterns; top-k mining; explorative mining.

I. INTRODUCTION

Frequent pattern mining is a fundamental problem in data mining and knowledge discovery. The discovered frequent patterns are used as the input for analyzing association rules, mining sequential patterns, recognizing clusters, and so on. However, discovering frequent patterns in large scale datasets is an extremely time consuming task. Various efficient algorithms have been proposed and published on this problem in the last decade. These algorithms can be classified into two categories: 1) the "candidate-generation-and-test" approach in which the Apriori algorithm [1] is the representative and the forerunner one; 2) the "patterns-growth" approach in which the prominent one is the FP-growth [2] algorithm. These algorithms, especially the FP-growth extended ones, improve the mining performance significantly.

Other fundamental problems in frequent pattern mining are the usability and user-friendliness. Conventional frequent pattern mining algorithms require users to provide a minimum support threshold, which is very difficult to identify without knowledge of the dataset in advance. A large minimum support threshold results in a small set of frequent patterns which users may not discover any useful information. On the other hand, if the support threshold is small, users may not be able to screen useful information from a huge set of frequent patterns.

Regarding to the usability and user-friendliness, top-k mining approach permits users to mine the k-most interesting frequent patterns without providing a support threshold. In the real world, users need to know the first k-most interesting frequent patterns, and then examine them to extract useful information. If they fail to discover useful information they can continue to mine the next k-most interesting frequent patterns and so on.

Several top-k mining approaches were introduced in [3], [4], [5]. These approaches tried to solve the difficulties of finding the "inner" minimum support threshold from a given number, top-k. The performance of mining top-k frequent patterns is improved significantly, especially in Top-k FP-growth [5]. However, there still remain some drawbacks in the manners of effectiveness and efficiency. These problems should be studied more deeply.

This research aims to propose a new algorithm to mine top-k frequent patterns, called "ExMiner". In this algorithm an explorative mining is performed first, before an actual mining is taken. Because of the explorative mining, an optimal inner support threshold is recognized and is provided as a support threshold for the actual mining phase. This fact substantially improves the effectiveness and efficiency of the algorithm. This research also proposes the idea of "build once – mine anytime" by extending the idea of the FP-growth algorithm. This idea provides a capacity of increasing the performance for the real life applications.

II. EXMINER ALGORITHM

ExMiner algorithm proceeds from the observation of mineral mining activities in the real life in which before taking an actual mining some explorative mining activities should be performed. The term ExMiner stands for the term "explorative miner". ExMiner algorithm extends the FP-growth to mine top-k frequent patterns effectively and efficiently with following 2 points: a) setting the internal threshold border_sup; b) taking an explorative mining to recognize an effective final internal support threshold which is used in the actual mining phase for discovering top-k frequent patterns. Following is the pseudo code of the ExMiner algorithm.

```
Input: Dataset $D$, number of patterns $k$
Output: top-$k$ frequent patterns
Method:
1. Scan $D$ to count support of all 1-itemsets
2. According to $k$, set border_sup and generate F-list
3. Construct an FP-tree according to F-list
4. Use VirtualGrowth(multiset<int>* supQueue, FP-tree) to
   explore the FP-tree and set the final internal support
   threshold $\theta$ to the smallest element, minq, of supQueue.
5. Mine the FP-tree with support threshold $\theta$. This mining
   phase outputs top-$k$ frequent patterns.
```

Figure 1. Pseudo code of the ExMiner algorithm

The pseudo code of the *VirtualGrowth* routine, in step 4, is described in figure 2.

An example of *VirtualGrowth* routine for mining top-7 frequent patterns from a given FP-tree is illustrated in figure 3. The queue, *Q,* is updated while traveling the tree following the node-links of items *f* and *c*. At the time item *a* is reached *minq* is *3*. Since the *support* of *a* is *3,* equals to *minq,* the routine stops. The current *minq=3* is the final internal support threshold to mine top-7 patterns.

* Graduate School of Science and Engineering,
Ritsumeikan University, Biwako-Kusatsu Campus Noji
Higashi 1 chome, 1-1 Kusatsu, 525-8577 Shiga-ken, JAPAN

```
Input: FP-tree, T; a queue of k empty items, Q
Output: k items in Q are fulfilled by the supports of potential
patterns and sorted by the descending order of their values.
Method:
1. Put supports of k elements in the header table, H, of T into
   Q. Set minq to the smallest value in Q
2. Start traveling T from the top of H.
3. Let a is a considering item. If sup(a) > minq then go to step
   4, else stop.
4. Recognize the supports of potential patterns. Let p is such a
   support value. If minq < p then update minq by p. The Q is
   then resorted automatically. Reach the next item in H and go
   to step 3.
```

Figure 2. Pseudo code of VirtualGrowth routine

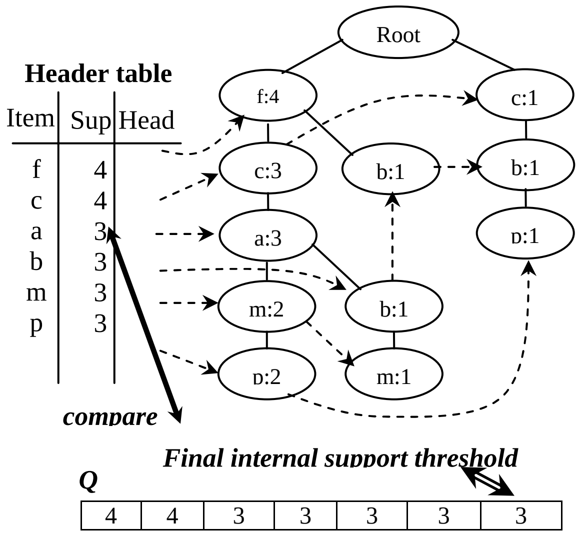

Figure 3. Example of a VirtualGrowth routine

III. Build Once Mine Anytime Approach

Using ExMiner algorithm to mine top-k frequent patterns, a new FP-tree has to be rebuilt from the scratch whenever a given top-k is changed. This algorithm contains three major tasks requiring three corresponding amount of times say, t_1, t_2, t_3. Those are the time of scanning the dataset to find frequent items, t_1; the time of building the tree, t_2; and the time of mining top-k frequent patterns, t_3. If a larger FP-tree is built first that it can be reused to mine top-k frequent patterns with any different value of top-k, we can save the times t_1 and t_2. If these two computation times are reduced, the performance is increased significantly. The idea of "build once mine anytime" allows us to do that. This idea is described as following.

A "large" FP-tree is built and its information is saved into the hard disk. When users want to mine top-k frequent patterns, the original "large" FP-tree is rebuilt based upon the information read from the hard disk. After handling the original FP-tree, the ExMiner algorithm can be applied to mine for any top-k frequent patterns. To mine top-k frequent patterns correctly, the original "large" FP-tree must contain at least *k* items. For example, an FP-tree with *10000* items can be used to mine any top-k frequent patterns where top-k not greater than *10000*. The original "large" FP-tree can be built in the computer-free time.

(1) FP-growth algorithm runs on the exact value of minimum support threshold, corresponding to a given top-k value

IV. Experimental Evaluation

In this section, the efficiency of the ExMiner algorithm and the "build once mine anytime" approach (BOMA-ExMiner for short) is compared to that of the Top-k FP-growth algorithm [5], the optimal Aprior algorithm, and the optimal FP-growth. The experiment was taken on the dataset D: T10I4D1000kN1000k created by using the IBM quest synthetic data generation code [6]. This test was performed on a 3.2GHz Pentium 4 PC with 1 GB RAM, Window XP, and MS. Visual C++ 6.0.

As in figure 4, the ExMiner algorithm outperforms the Top-k FP-growth. Moreover, the interesting thing is that the BOMA-ExMiner approach is even better than the optimal FP-growth (1).

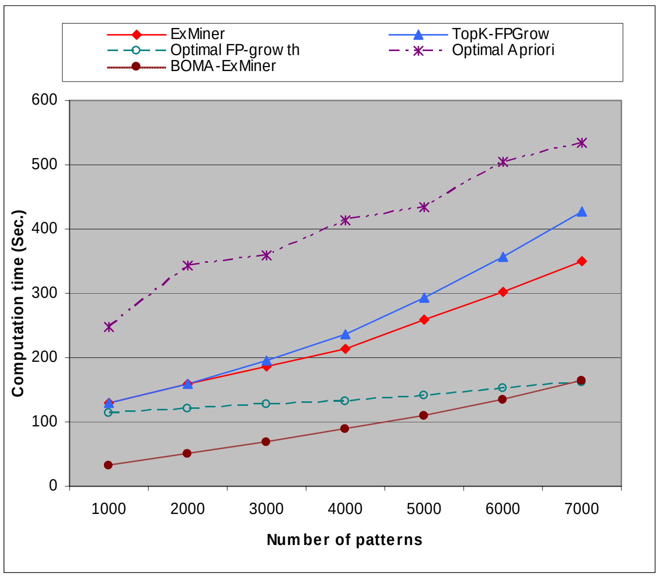

Figure 4. Running time of ExMiner, BOMA-Exminer

V. Conclusions and Future Work

This research proposed a new algorithm for mining top-k frequent patterns efficiently and effectively: the ExMiner algorithm. The combination of ExMiner algorithm with the idea of "build once mine anytime" increases the performance significantly. This approach is notably valuable in real life applications where data structure (the large FP-tree) can be prepared at the computer's free time. To do so, the response time in the mining phase is improved significantly. Applying these approaches to mine top-k frequent patterns sequentially is deferred to the future work.

References

[1] Agrawal, R, and Srikant, R. *Fast algorithm for mining association rules*. In proc. of VLDB '94. pp. 487-499, Santiago, Chille, Sept. 1994.

[2] Han, J., Pei, J., and Yin, Y. *Mining frequent patterns without candidate generation*. In proc. of ACM SIGMOD Conference on Management of Data, pp. 1-12, 2000.

[3] Fu, A.W., Kwong, R.W., Tang, J. *Mining N most interesting itemsets*. In proc. of ISMIS'00, 2000.

[4] Ly, S., Hong, S., Paul, P., and Rodney, T. *Finding the N largest itemsets*. In Proc. Int. Conf. on Data Mining, Rio de Janeiro, Brazil, pp. 211-222., 1998.

[5] Hirate, Y., Iwahashi, E., and Yamana, H. *TF^2P-growth: An efficient algorithm for mining frequent patterns without any thresholds*. In proc. of ICDM., 2004.

[6] IBM Quest Data Mining Project. Quest synthetic data generation http://almaden.ibm.com/software/quest/Resources/index.shtml