Efficient Frequent Pattern Mining in Data Streams

Po-Jen Chuang and Yun-Sheng Tu
Department of Electrical and Computer Engineering, Tamkang University, Tamsui, New Taipei City, Taiwan 25137, R.O.C.
pjchuang@ee.tku.edu.tw

Abstract. In this paper, we present a new frequent pattern mining mechanism for data streams to enhance the performance of existing mechanisms, particularly to reduce the needed processing time and uplift the mining accuracy. Our new mechanism performs data mining by the Trie data structure (instead of the entry table in previous mechanisms) to access patterns at reduced processing steps and run time. To attain desirable mining accuracy, it adopts a Length Skip practice to skip less frequently used long patterns from the summary and preserve the space for more frequently used short patterns. Simulation results show that, when incorporated into existing frequent pattern mining algorithms, our new mechanism will effectively increase the mining accuracy at notably reduced run time.

1. Introduction
Frequent pattern mining [1-2] can be performed in such structures as Landmark [3-4], Sliding Window [5-6] and Time Fading [7]. In Landmark pattern mining, we can process data streams from a fixed previous time to the current time, preserving only important information (i.e., the summary) to typify the data – via the Batch-by-Batch or One-by-One approach. The Batch mode of mining has a key problem: It cannot process a batch before the buffer is full. That is, it may process data streams faster (in batches) but may not satisfy the real-time demand which the One-by-One mode is able to achieve (as it handles one data at a time). For instance, the Lossy Counting and Space Saving (LC-SS) algorithm [4] – a Landmark-based One-by-One approach – can handle bursty data and perform effective data mining to meet real-time demands with finite time and space. The operation of LC-SS, however, takes significant run time because it involves the Table data structure. In addition, with low minimum support thresholds (mst), it may fail to store the sorted frequent patterns or yield correct pattern recognition when handling huge volumes of complex data. To fix up the above loopholes in existing mining mechanisms, this paper presents a new frequent pattern mining mechanism to satisfy the real-time requirements and meanwhile reduce the required processing time. Unlike the LC-SS algorithm which manages data streams by the Table data structure, our mining mechanism uses the Trie data structure to handle data streams by less processing steps – i.e., at accelerated processing speed and reduced run time. It then adopts a Length Skip practice – which skips long frequent patterns to focus on short frequent patterns – to maintain satisfactory mining accuracy and reduce more run time. In practice, Length Skip uses sliding windows to hold the newest 10K data, sets a check point at every 10K and uses the data in the windows to examine (at each check point) if the mining accuracy drops below the set threshold. If it drops below, Length Skip will re-set an appropriate pattern length limit to manage the candidate patterns in the summary. By restricting pattern lengths, we can exclude longer patterns from entering summary so as to preserve more space for better fitting candidate patterns and as a result to lift up the mining accuracy.

To check the performance of the proposed mining mechanism, we have conducted simulations under various situations – using real datasets as the data streams to attain the run time and mining
accuracy. The obtained results show that, when our mechanism is incorporated into existing mining algorithms (such as LC-SS), it can significantly enhance the mining performance by turning out decreased processing time and increased mining accuracy.

2. The Proposed Mechanism

Enhanced LC-SS, either Skip LC-SS or R-skip [4], faces two major problems. (1) It takes considerable run time to scan the entry table. (2) It may not dig out truly frequent patterns when mst is low. In fact, with low mst, it may yield an extremely low mining accuracy. As observed, LC-SS takes much time to process data stream mining mainly because it accesses patterns by the entry table with a linear list – which requires much run time to complete the processing steps when the table size largely grows. To improve the situation, our new mechanism does not access patterns directly from the entry table. Instead, it involves a different data structure, the Trie structure, to avoid the problem in LC-SS, i.e., to reduce the mining steps and also processing time in LC-SS. Suppose there are $k$ entries in the table and the average length of the transaction is $L$, using the entry table (the linear list) to process update and replacement requires $O(kL)$ processing time, but using the Trie structure to finish the two steps takes only $O(2^L)$. It pinpoints that, in performing data mining, LC-SS involves the entire summary in the entry table, whereas our practice involves only some nodes in the Trie. Normally, the size of an entry table $k$ is 500K and the average length of a transaction $L$ is close to 10. We have $O(kL) >> O(2^L)$, an apparent reduction.

As observed from the result of running the FP-growth algorithm [8] over the Retail dataset [9-10] for R-Skip [4], the biggest pattern length in the summary is 17, whereas the length of truly frequent patterns is between 1 and 5. This indicates extremely long patterns are less likely the frequent patterns in need and, as a result, can be removed from the summary. The fact leads to the concept of length skip: i.e., to skip/remove longer patterns (less likely frequent patterns) from the summary so that the summary will have more space to cover shorter patterns which will more likely become frequent patterns. When the summary contains more potential frequent patterns, we can obtain the needed frequent patterns more handily to lift up the mining accuracy. In frequent pattern mining, the length skip concept is practically feasible given the fact that shorter patterns fit better in different applications [3]. However, our mining mechanism will perform the Length Skip practice only to lift up low F-scores (a harmonic mean of Recall and Precision [11]) at low mst – i.e., it will not involve the practice when the F-scores are good enough.

The Length Skip practice works as follows. Assuming a transaction is with length $L$, it will generate $2^L$ patterns. If the length of patterns to be stored in the summary is limited to $X$, $L > X$. That is, the summary will store only patterns with lengths $1~X$ and a total of $\sum_{i=1}^{X} \binom{L}{i}$ patterns will be generated in the replacement step – $\sum_{i=1}^{X} \binom{L}{i} < 2^L$. The patterns to be stored in the summary (the candidate patterns) hence decreases in numbers. With reduced candidate patterns, the case (the number of candidate patterns > the number of minimal entries) will less likely happen – i.e., compensation for each pattern to reach recall = 100% will grow at a lower speed and the mining accuracy will rise accordingly. In the Length Skip practice, Pattern Length Limits ($X$ mentioned above) are set by sliding windows. With sliding windows of a fixed size, we can practically store the most recent data and then use the stored data to predict/calculate a proper $X$ for performing the Length Skip practice.

In our Length Skip practice, each check point of sliding windows is with two assessing steps.

Step 1: Determine if the F-score drops below the set accuracy threshold.

Use the data in the current sliding window (briefed as current window data) to run the FP-growth algorithm and get truly frequent patterns for later accuracy verification. Then use the approximate frequent patterns (obtained by running R-skip on current window data) and truly frequent patterns to attain the mining F-score of current window data (briefed as the window F-score). If the attained window F-score exceeds the set accuracy threshold, do not modify $X$; otherwise, decrease $X$ by 1 and perform the next assessing step.

Step 2: If the obtained window F-score does not exceed the accuracy threshold, decrease $X$ by 1 each time until it goes over the threshold and take the newest $X$ to perform Length Skip in next mining.
The concept of our Length Skip can readily work with other data stream mining algorithms to enhance the mining accuracy. Only patterns with lengths $1\sim X$ are taken as candidate patterns and stored into the summary. We attain and examine the current window F-score at each check point to get a more appropriate $X$ and use the new $X$ to maintain desirable mining accuracy.

3. Performance Evaluation

The results of different pattern lengths (0~7) vs. different check points (10K, 20K…, 80K) at mst 0.1% collected in our simulation can be used to attain proper values of $X$ for the three datasets (Retail, T10I4D100K and BMS2 [9-10]). For instance, we find the proper value of $X = 4$ for dataset Retail at each check point and will hence put patterns with lengths $1\sim4$ in the summary. As for the other two datasets, we have different $X$ values at different check points.

For dataset BMS2, we attain $X = 6$ at check point 10K and include patterns with lengths $1\sim6$ in the summary. At check point 20K, we detect proper $X = 5$ and hence decrease $X$ by 1 (from 6 to 5), i.e., allowing only patterns with lengths $1\sim5$ in the summary. When seeing a smaller $X$ (<5) in subsequent check points, we will similarly update $X$ to adjust the pattern length limits ($1\sim X$) – in order to preserve the accuracy.

Dataset T10I4D100K reveals a different case: We have $X = 4$ at check points 10K~30K but $X = 5$ at 40K. As Length Skip only decreases $X$ when necessary, we will not update $X$ at 40K (i.e., will maintain the original value 4).

![Figure 1. Reducing run time by Length Skip.](image)

According to our simulation results mentioned above, we set $X = 4$ for datasets Retail and T10I4D100K and $X = 5$ for dataset BMS2. To check how Length Skip affects the mining accuracy, we then engage simulation runs to attain the F-scores with and without Length Skip for the three datasets. The results in Figure 2 display clearly low F-scores at smaller mst for all datasets without Length Skip. With Length Skip, the F-scores are stably and notably higher as Length Skip helps reduce the number of to-be-processed patterns to result in more space for incoming patterns.

According to the experimental evaluation, R-Skip takes more than 1000 seconds to process a
dataset, whereas R-Skip (Trie)+Length Skip takes less than 100 seconds. The significant reduction in run time is apparently the result of increased processing speed which, in turn, is the result of processing data mining in reduced steps. On the other hand, R-Skip is shown to yield nearly 0% accuracy at low mst. When working with Length Skip, it produces over 90% accuracy at the same mst. Besides R-Skip, we believe our proposed mechanism can also work with other mining algorithms to pursue significant performance gain in both run time and mining accuracy.

![F-score graph](image)

Figure 2. F-scores – with or without Length Skip – in different datasets.

4. Conclusion

This paper presents a new frequent pattern mining mechanism to fix the loopholes in existing mechanisms (such as LC-SS), especially to shorten the required run time and raise the mining accuracy. Different from LC-SS which accesses patterns from the entry table, we access patterns by the Trie structure which will handle data streams at reduced steps during the mining process – to increase the processing speed and shorten the run time. To enhance the mining accuracy and meanwhile further reduce the run time, we adopt a Length Skip practice to skip less significant long frequent patterns and save the summary space for more practical short patterns. Following the set pattern length limit, we can remove long patterns from the summary to save space for better fitting short patterns – i.e., to include more useful frequent patterns in the summary and, as a result, to bring up the mining accuracy. As the collected simulation results demonstrate, when put to work in the LC-SS algorithm, the Trie structure will notably reduce the processing time in original LC-SS while the Length Skip practice will efficiently uplift the mining accuracy especially at low mst. Given the fact that our mechanism explicitly reduces the run time and increases the mining accuracy in LC-SS, we believe it can be handily incorporated into other mining algorithms to advance the mining performance.
5. References
[1] Tsai, C.-W., et al. 2014. Data mining for internet of things: a survey. IEEE Communications Surveys & Tutorials, 16, 1 (2014), 77-97.
[2] Lee, G., et al. 2014. Sliding window based weighted maximal frequent pattern mining over data streams. Expert System Applications, 41, 2 (2014), 694-708.
[3] Ng, W. and Dash, M. 2010. A comparison between approximate counting and sampling methods for frequent pattern mining on data streams. Intelligent Data Analysis, 14, 6 (2010), 749-771.
[4] Yamamoto, Y., et al. 2014. Resource-oriented approximation for frequent itemset mining from bursty data streams. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (June 2014), 205-216.
[5] Lee, V. E., et al. 2014. Frequent pattern mining in data streams. Frequent Pattern Mining (2014), 199-224.
[6] Li, C.-W., et al. 2012. Mining frequent patterns from dynamic data streams with data load management. Journal of Systems Software, 85, 6 (2012), 1346-1362.
[7] Chen, L. and Mei, Q. 2014. Mining frequent items in data stream using time fading model,” Information Sciences, 257 (2014), 54-69.
[8] Han, J., et al. 2000. Mining frequent patterns without candidate generation. ACM SIGMOD Record, 29, 2 (2000), 1-12.
[9] Fournier-Viger, V. S., et al. 2015. SPMF: a Java open-source pattern mining library. Available: http://www.philippe-fournier-viger.com/spmf/ (2015).
[10] Goethals, B. 2015. Frequent Itemset Mining Dataset Repository. Available: http://fimi.ua.ac.be/ (2015).
[11] Precision and recall, Available: https://en.wikipedia.org/wiki/Precision_and_recall (2015).