D_008

A Framework for Top-k Frequent Closed Patterns Mining

Tran MINH QUANG (*) Shigeru OYANAGI (*) Katsuhiro YAMAZAKI (*)

Abstract: *Frequent pattern sets gained from conventional frequent patterns mining algorithms are commonly huge and contain redundant information that hinder user from analyzing interesting knowledge. Mining and managing such redundant patterns also make the mining algorithms inefficient. Mining only the frequent closed patterns avoids tracing and managing redundant patterns, yet still reserves the useful knowledge. Mining top-k frequent patterns allows users to control the number of patterns to be discovered for analyzing. This paper proposes a framework combining the two aforementioned advanced ideas to mine top-k frequent closed patterns effectively and efficiently.*

1. Introduction

Recent years the frequent patterns mining has widely contributed to many aspects of the data mining problems such as association rules mining, cluster analyzing, classification studying and so on. Therefore, it should be considered more deeply not only on the aspect of improving the mining performance as Apriori-based approaches [1] or more prominent, FP-growth-based approaches [2] do, but should also be focused on the usability and user-friendliness aspects of the mining algorithms.

Conventional methods commonly present redundant, huge sets of patterns that might become useless for users since they might not be able to analyze for interesting knowledge from those sets. To help user control the number of patterns that they want to handle, the idea of mining *top-k* frequent patterns has been introduced [3]. In this approach, instead of specifying an enigmatic minimum support threshold, users specify a number of k highest frequency patterns that they want to obtain. Moreover, to avoid the redundancy, the idea of mining frequent *closed* patterns [4] has been proposed in recent years. Following to this approach, no subset X' of a pattern X whose support is equal to that of X ($sup(X') = sup(X)$) will be presented since X' can be inferred from X. Combining these two ideas can make the algorithms more robust. It might contain all the difficulties of the *top-k* mining and those of the mining frequent *closed* patterns, however, this should be an interesting field to study. This research aims to propose a framework for mining *top-k* frequent *closed* patterns (*top-k closed* mining) effectively and efficiently.

2. Raising support threshold

In *top-k* mining the support threshold is not given in advance thus it is very important to raise this value from 0 (initially setting) effectively to a higher, suitable value. Obviously raising support for *top-k closed* mining contains its own special features. Several raising methods are proposed as following.

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

2.1. Separating the list of items

After the first scan on the dataset, a list of items sorted by the descending order of their support, called *F-list*, is constructed. Without any prior knowledge from the association of data in the dataset, for safety, we assume that all the items with the same support might always occur together. According to this assumption, we can divide *F-list* into segments each of which contains items whose supports are the same. After that we can identify the newly support threshold, called the *border_sup*. An example of this method is illustrated in Figure 1 bellow.

a:8	b:7	c:6	d:4	**e:3, f:3, i:3**	g:1, h:1

Figure 1. Segmentation the item list

Figure 1 shows that the *border_sup* for mining *top-6*, *top-5*, *top-4* frequent *closed* patterns is raised from *0* to *1*, to *3*, and to *4*, respectively (corresponding to the support of 6^{th} and 5^{th}, 4^{th} segments).

2.2. Decomposing segments

We can imagine that in the real datasets, items whose supports are the same rarely always occur together. Therefore we should try to recognize which items should not come together and decompose the segments that contain them. To do so, we first propose some definitions and then analyze their characteristics.

Definition 1 (Closed node) A *closed node* is a node whose count is greater than the sum of those of its children.

Definition 2 (Closed item) A *closed item* is an item that has at least one closed node.

Lemma 1 A closed item cannot be in the same segment with its right-sibling.

Rationale: Let a be a closed item and b be its right-sibling in the same segment. According to the definition of the segment, we can see that b occurs at every occurrence of a. Therefore, in the FP-tree every node labeled a will has a child node labeled b whose support is the same of that of node a. This means a is not a closed item, which conflict to the assumption.

Owing to lemma 1, we can infer that a closed item should be the last item in a segment. Therefore, if an item is recognized to be closed, then it should be at the position at which the corresponding segment is divided. The problem is how to recognize closed items. If it is recognized by looking for their corresponding closed nodes, we will turn back to the original problem of identifying closed nodes (i.e. how to identify closed nodes without traversing the FP -tree?). Lemma 2 bellow will help.

Lemma 2 An item that once occurred as the last position in a *sorted* transaction (before building the tree all transactions have to be sorted) is a closed item.

An example of the decomposing multiple segments is illustrated in the Figure 2 and described as follows. After the first scan on the dataset, a list of items, *F-list*, is built (the upper list). Following segmentation procedure, *F-list* is divided into *2* segments and the *border_sup* for mining *top-2*, for example, frequent *closed* patterns is set to *1* (the support of the 2^{nd} segment, also the last segment). At the second scan, transactions

are sorted (as showed in the 3rd column of the table) and we recognize that *f* is a closed item since it appears at the end of the *4th* transaction. Therefore the first segment (*efi: 3*) is divided into *2* segments (at the position of *f*) as shown in the lower *F-list*. At the same time, the *border_sup* for mining *top-2* frequent *closed* patterns is raised from *1* to *3* (the support of the *2nd* segment, the new one).

A transactional dataset

Tid	Transaction	Sorted transaction
1	e, i, f	e, f, i
2	e, g, i	e, i, g
3	h, i, f	i, f, h
4	f, e	e, f

Item list (F-list)

e:3, f:3, i:3	g:1, h:1

⇩

e:3, f:3 i:3	g:1, h:1

Figure 2. Decomposing multiple segments contained in the *F-list*

In summary, after the first scan on the dataset, *F-list* is built and divided into segments. At the second scan for building the FP-tree, transactions will be sorted thus the closed items are recognized and are used to decompose the multiple segments. When the multiple segments are decomposed, more segments with high support value are created increasing the opportunity to raise the *border_ sup* faster. We refer the readers to [5] for the rationale of the lemma *2*.

2.3. Exploring the closed nodes – RSCI method

According to two support-raising methods above, each segment represents for only one closed pattern. In fact, items in the beginning of the *F-list*, whose support is much greater than the current *border-sup*, may represent for many closed patterns that satisfy the *border-sup*. We proposed a method that can explore the closed nodes to figure out the support of the potential closed patterns which might be grown from those closed nodes. This method is called the "Raising Support based on the Closed Items" (RSCI) method and described as following.

Procedure of RSCI:

- Start from the top of the header table and refer to the list of the *closed items* (already recognized). If the support of the considering *closed item* is greater than the current *border_sup* then traverse the tree via the structure of the item's node-link
- Extract all the conditional bases of the *closed item* and mark the *closed conditional bases*. A *closed conditional base* is a conditional base extracted from the root to a *closed node* (at least one closed conditional base will be extracted since a *closed item* has at least one *closed node*)
- Create a list of *closed conditional items* (the items in the *closed conditional bases*). This list is called the F_{cci}*-list*
- Update the counts of items in the F_{cci}*-list* by adding the count of corresponding items extracted from the *non-closed conditional bases,* and the sort them by the descending order of their supports
- Segment the F_{cci}*-list*
- Decompose F_{cci}*-list* base on the *local closed* items
- From the current status of the F_{cci}*-list* we can raise the current *border_sup* based on the judgment that each segment of F_{cci}*-list* represents for one closed pattern of which support is the same of that of the segment

3. Mining strategies and closure-checking

In the conventional FP-growth algorithm, the bottom-up direction is used to mine frequent patterns, while the *top-k* mining employs the top-down direction since the algorithm can stop anytime when *k* highest frequency patterns have been already presented. In *top-k closed* mining, the mining strategy not only affects to the mining performance but also influences to the way of closure-checking. What strategy (or strategies) should be chosen to make the mining efficient and the closure-checking effective is an interesting question waiting for solutions.

Obviously, top-down mining strategy is appropriate for *top-k* mining, yet it is not suitable for mining frequent *closed* patterns. The reason is that, if the top-down strategy is employed, the short patterns will be mined first and they may be subsumed by the longer patterns extracted later. This means, the algorithm wastes the computation time for mining short but un-closed patterns. In contrast, the bottom-up can void this disadvantage since the long patterns will be mine first, and then some checking procedures will be performed to avoid wasting computation time for mining non-closed patterns. Our problem is the combination between *top-k* mining and frequent *closed* patterns mining, the mining strategies should be the synthesizing of both the aforementioned strategies.

- **Top down**: is employed in identifying the conditional pattern bases. The global FP-tree will be traversed in the *top-down* direction to identify the conditional bases and build the 1st level conditional FP-tree[1].
- **Bottom up**: is used to mine the frequent patterns from the conditional FP-trees, thus save the computation time for extracting the shorter, unclosed patterns by using the closure-checking method.

4. Conclusions

This research has proposed a framework for mining *top-k* frequent *closed* patterns in which instead of tracing patterns, some methods for tracing and raising support threshold were proposed. After the support threshold is raised to a high value, which is close to the optimal support, the algorithm can mine the *top-k* frequent *closed* patterns easily. Moreover, a hybrid mining strategies (top-down and bottom-up) were proposed to improve the performance of the mining and the closure-checking tasks. The details of the implementation and evaluations for this framework are deferred to the future work.

References:

[1] Agrawal, R, and Srikant, R. *Fast algorithm for mining association rules*. VLDB, 1994, pp. 478-499.

[2] Han, J., Pei, J., and Yin, Y. *Mining frequent patterns without candidate generation*. SIGMOD, 2000, pp. 1-12.

[3] Quang, T.M., Oyanagi, S., and Yamazaki, K., *ExMiner: An efficient algorithm for mining top-K frequent patterns*. In LNAI 4093, Springer-verlag Berlin Heidelberg , 2006, pp. 436 – 447.

[4] Pei, J., Han, J., and Mao, R. *CLOSET: An efficient algorithm for mining frequent closed itemsets*. In proc. of DMKD, 2000, pp. 11-20.

[5] Quang, T.M. *Usability and user-friendliness oriented frequent Patterns mining*. Master thesis, Ritsumeikan University, Japan, 2006.

[1] We refer the reader to the FP-growth [2] method for more details