Top-k-FCI: Mining Top-K Frequent Closed Itemsets in Data Streams

Jun LI, Sen GONG

Henan University, Kaifeng 475001, China

Abstract
With the generation and analysis of stream data, such as network monitoring in real time, log records, click streams, a great deal of attention has been concerned on data streams mining in the field of data mining. In the process of the data streams mining, it is more reasonable to ask users to set a bound on the result size. Therefore, in this paper, an real-time single-pass algorithm, called Top-k-FCI (top-K frequent closed itemsets of data streams), is proposed for mining top-K closed itemsets from data streams efficiently. A novel algorithm, called can (T)(candidate itemset of the T), is developed for mining the essential candidate of closed itemsets generated so far. Experimental results show that the proposed Top-k-FCI algorithm is an efficient method for mining top-K frequent itemsets from data streams.

Keywords: Data Mining; Data Streams; Frequent Closed Itemset; Top-K

1. Introduction and Related Work
In recent years, many applications involve the generation and analysis of a new kind of data, called stream data, where data flow in and out of an observation platform (or window) dynamically. Different from the traditional data sets, such data streams are fast changing with the time, massive and potentially infinite. Due to these characteristics, therefore researchers are concerned on the analysis when data stream changes; especially they are interest in the relative high-level trends and deviations and so on. In short, the data mining in data stream is a rapidly evolving field, so the mining in data stream in a dynamic environment will allow people to quickly get accurate and useful information.

The traditional data mining is on the relative static database. Since the introduction of association rules which Agrawal has presented, the study of frequent itemsets has been greatly researched. In 1993 Agrawal has proposed the algorithm Apriori [1], which is carrying out by two stages. The first is mining all frequent itemsets; the second is to extract association rules from the obtained frequent itemsets. The most important part of mining association rules is how to mine frequent itemsets. Then closed frequent item sets which are the compact form frequent itemsets have also been greatly studied. Pasquier [2] proposed the concept of closed frequent itemsets, which are also called closed frequent patterns. Closed frequent itemsets is such an itemset, the collection of them can deduced all the frequent itemsets and the number has greatly reduced, further the storage of them are also has been greatly improved in space and time requirements.

† Corresponding author.
Email addresses: lj@henu.edu.cn (Jun LI).

1553-9105/ Copyright © 2011 Binary Information Press
December, 2011
After the association rules and the closed frequent itemset have been proposed, a lot of researchers studied frequent itemset. For example, Pasquier designed the algorithm A-Close [2] which is a frequent closed itemsets mining algorithm; Pei proposed a recursive algorithm CLOSET [3] and constructed the conditional FP-Tree; Zaki put forward the algorithm CHARM [4] which has the best performance of existing algorithms. In addition, the DCI_Closed [5], BIDE [6] and so many classical algorithms on mining closed frequent itemset.

Many researchers present some improved algorithm for data stream mining [7-13]. In literature [7], the authors discussed the outlier detection in mixed data stream and presented the statistical information definition based on the sliding window model which can estimate the distribution of data stream approximately. In data stream if we mine frequent itemset the number of frequent itemset is very large, so in data stream mining FCI can save plenty of efficiency about time and space. Manknu proposed an algorithm Lossy counting [8] on the current transaction of the entire data stream which can mine approximate frequent itemset. Giannella, Jiawei Han and so on proposed an algorithm FP-stream [9] which is to mine approximate frequent itemsets at any time intervals. In literature [10], Tu presented an algorithm F-Stream for computing frequency counts based on sliding window. F-Stream can detect $\epsilon$-approximate frequent items of a data stream using $O(\epsilon^{-1})$ memory space and the processing time for each data item is $O(\epsilon^{-1})$. Huafu Li gave the Dsm-MFI [11] algorithm, which is used to maximal frequent itemsets in recently time by SFI-forest data structure to store a summary of the current data model. Relative to the limited memory space, the mining of closed frequent itemsets is better met the needs of users and also satisfied the reality.

The algorithm Moment [12] used the data structure CET to store all the information of all itemsets in the data stream. According to the changes in sliding window Moment updates the four different itemsets in real time of the data structure CET, but the shortage of it is that the items’ historical information is not saved in the structure. The algorithm A-moment [13] which is an improved algorithm of the Moment, but it still used the CET structure and it take an approximate technique to improve the efficiency of data mining algorithms. Literature [14] is the first time to give the mining algorithms of Top-k in which Tops-Tree data structure is used to mine itemsets. Literature [15] used the way of closed itemset lattice to mine Top-k frequent closed itemsets. Due to the user in the data stream can not be accurately set the minimum support threshold, the algorithms has been proposed. The experiment results also show that the algorithm TKC-DS is better than other algorithms at present.

2. Problem Definition

In this section, the research problem of mining top-k closed frequent items over data stream with a transaction sensitive sliding window is defined.

**Definition 1.** A transaction data stream $TDS = \{T_1, T_2, T_3, \ldots, T_N\}$ is an unbounded sequence of transactions, where $N$ is TID of latest incoming transaction $T_N$.

**Definition 2.** In a transaction data stream, a transaction-sensitive sliding window $SW = [T_{N-w+1}, T_{N-w+2}, \ldots, T_N]$ is a window that slides forward for every incoming transaction, where $N$ is TID of latest incoming transaction and $w$ is the size of $SW$. 

**Definition 3.** The support of an itemset \( X \), denoted as \( \text{sup}(X) \), is the number of transactions of \( SW \) containing \( X \) as a subset.

**Definition 4.** An itemset \( X \) is called a frequent closed itemset (FCI), if at first the itemset is frequent, and if there exist no itemset \( X_0 \) such that (1) \( X_0 \) is a proper superset of \( X \), and (2) every transaction containing \( X \) also contains \( X_0 \).

**Definition 5.** A closed itemset \( X \) is called a top-\( K \) closed itemset if there exist no more than \( (K-1) \) closed itemsets whose support is higher than that of \( X \).

The closed itemset lattice which used to storage the itemset in current window. The following Tab. 1 describes the five transactions in the sliding window; the structure of lattice will be introduced in the following passage. In this section, the proposed data structure closed itemset lattice (CIL) is defined and the constructing process of the CIL is introduced as follows.

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>cd</td>
</tr>
<tr>
<td>T2</td>
<td>ab</td>
</tr>
<tr>
<td>T3</td>
<td>abc</td>
</tr>
<tr>
<td>T4</td>
<td>acd</td>
</tr>
<tr>
<td>T5</td>
<td>bc</td>
</tr>
</tbody>
</table>

Because using the method of bit vector to denote itemset not only can decrease the time and memory requirement in the process of program, but also improve the efficiency of the algorithm, CIL is constructed by itemset bit vector in this paper. The dimension of the bit vector is \( m \), which is the number of items in database. Every subset of items in database can be uniquely determined by a bit vector. Such as itemsets \( I = \{a, b, c, d\} \), and \( I \) has a subset \( \{a, c\} \). This subset \( I \) can be uniquely determined by bit vector \( (1010) \).

Closed itemset lattice(CIL) is an extended lattice structure and defined as follows. CIL is composed of three substructures: a list of bit-vectors of items (BV-list), a set of closed itemset tree (CI-tree), a list of supports of closed itemsets (S-list), and a hash table of closed itemsets (HTC).

In CI-tree(i), besides the root node, the other nodes are consisting of root node and its right sibling (siblings) which do bitwise and operation. So we can obtain the child node of the current root node. In the BV-list the single itemset and each node in CI-tree(i) we will store it into an array, which is stored in the form of matrix. According to the numerical calculation of the matrix each row we can obtain the support count of itemset. When any of the support count of node is zero in the CI-tree(i), the algorithm deletes it from the closed itemsets tree, and don’t carry out its siblings’ operations. Because the node’s support in the current window is zero, it can not be the frequent itemsets, which is according the prior theory and its corollary, namely support a superset of an itemset with the support is zero is certainly not frequent itemset.

According the above, the final closed itemset lattice of the table is in Fig 1.

### 3. Top-k-FCI Algorithm

#### 3.1. The Component of the Top-k-FCI

In this section, the components of the Top-K-FCI algorithm will be introduced as follows:
1) **The basic damped sliding window.** The data in traditional relational database is static and stability, but the data in data stream is potentially infinite and it changes dynamically. Therefore the data processing in data stream is continuous, so the algorithm used the mechanism of sliding window.

![Diagram of CI-tree and BV-list](image)

This paper is based on the sliding window (SW), and divided sliding window into b basic windows BW. Also the algorithm gives them a damped factor \( \varepsilon \) for each window, the value of \( \varepsilon \) is \((0, 1]\), if the value is 1 then the basic damped window is changed to the general basic sliding window. The basic damped window \((BW - \varepsilon)\) can be described as follows. A window on the current window when the phase of extracting the patterns, the summary of the basic structure in a basic damped sliding window to other windows, namely the following description of the information in closed itemset lattice must be multiplied by the factor \( \varepsilon \), the support counts of current patterns to the previous basic window is updated after the calculation of factor.

2) **Support improved method.** To set the weights for each item in the transaction database, this paper called it as the weighted support count. Given the itemset \( I = (i_1, i_2, \cdots, i_n) \), the corresponding weight of the each itemset is set to \((w_1, w_2, \cdots, w_n)\). The weight of them are set by the user which expressed the importance of itemsets \( i_j, 0 \leq w_j \leq 1 \) \((j = 1, 2, \cdots, n)\).

After setting the weights of each itemset, record the support count about closed itemsets after the arrival in the basic damped sliding window. Then all the closed itemset in closed lattice structure are stored in a hash table with the key as support. Given the \( i \)-th window has \( n \)-closed itemsets, from 1 to \( n \) closed itemsets the support recorded as \( sup_1, sup_2, \cdots, sup_n \), and then we can obtain the weighted count:

\[
sup_i = sup(X) \times \sum_{j \in X} w_j, \quad (1)
\]

so the minimum support count in \( i \)-th window is:

\[
sup(i) = \left\lfloor \frac{\sum_{j=1}^{n} sup(j)}{n} \right\rfloor. \quad (2)
\]

And then the next coming window’s \( m\text{insup} \) is:

\[
m\text{insup}(i + 1) = \left\lfloor \frac{\sum_{j=1}^{n} sup(j)}{n} \right\rfloor. \quad (3)
\]

3) **Generate the candidate itemset.** Since the characteristics of data stream, we can’t use the brute force method or the way of priori method to obtain the candidate itemsets. And the space and time efficiency is bad, and memory requirements on the consumption are also too large. This paper proposes a new method of candidate itemsets generation \( Can-(T) \), which greatly reduces the space and memory requirements, and the
method of generating candidates has more accurate results, and also can avoid the problem of data bumps. The Can-(T) algorithm is showed as follows.

\textbf{Can-(T) Algorithm:}
\begin{itemize}
  \item \textit{Input:} A transaction which would be delete when the phase of window sliding;
  \item \textit{Output:} a set of candidate closed itemset;
  \item \textit{Method}:
  \begin{itemize}
    \item \textbf{Begin:} There are already a CIL with \( SW_i \), the set \( \text{subset}(T_d) \) is generated by the delete transaction \( T_d \);
    \item for each node \( n_j \subseteq T_d \)
    \item \textbf{if} \( (\text{sup}(n_j) < \text{minsup}(i+1)) \)
    \item delete \( n_j \) from \( T_d \);
    \item /* the minsup is the support computed by the program in the (1)*/
    \item The set \( \text{Cad-FCI} = \text{subset}(T_d) \);
    \item *the not all subset transaction \( T_d \) subset(\( T_d \)) is the set of the subset beside the itemset that has already been in the HTC */
    \item \( \text{Cad-FCI} = \text{Cad-FCI} \cap(T_d) \);
    \item endif
    \item endfor
  \end{itemize}
  \item \textbf{end}
\end{itemize}

3.2. The Framework of the Top-k-FCI

Given the content attributes and the relational structure of an instantiation with a set of entities, an RMN specifies the cliques and potentials at a template level, i.e., it specifies a conditional distribution over all of the labels of all of the entities.

To mine the top-k closed frequent itemset in data stream, the paper used the follow three steps to describe the algorithm Top-k-FCI: (1) delete an old transaction; (2) add a new transaction \( T_n \); (3) the output of the top-k closed frequent itemset.

1) The Del-old-tra algorithm. When the sliding window removes an old transaction in \( T_d \), the first step is make the entire bit vector in the BV-list right shift one bit. The most left one is moved out, according to the basic size of the window that the length \( l \) of the transaction, then all the vectors in the bit vector list, the most right bit is set to 0 on. As indicated in Fig. 1, the length of sliding window is 4. When the delete transaction \( T_1(c,d) \), all the bit vector should be left shift one bit. That is a: 0111 into 1110, b: 0111 into 1110, c: 1011 into 0110, d: 1000 into 0000. Also the CI-num(a) is 2, CCI-boundary (b) is 0; because itemset a and b are not in the transaction \( T_1(c,d) \). Therefore they are not be processed. CI-num(c) is 1, CCI-boundary(c) is 1, due to CI-num(c) \( \leq \) CCI-boundary (c), so the algorithm has to update all children in the transaction and minus their support count. In the figure above there is no such node, therefore the items c and d can not be processed.

\textbf{Del-old-tra Algorithm:}
\begin{itemize}
  \item \textit{Input:} A generated CIL of the current window;
  \item \textit{Output:} a new CIL after the delete of the transaction \( T_d \);
  \item \textit{Method}:
  \begin{itemize}
    \item \textbf{Begin:}
    \item All the bit-vector of the BV-list are left shifted by one bit, and fill them with the value 0;
    \item /*the leftmost bit was dropped and the rightmost bit is set to 0;*/
    \item For each item \( X_i \) of \( T_d \) do
    \item \textbf{if}(CI-num(\( X_i \))\( \leq \) CCI-boundary (\( X_i \)) then
    \item for each \( n_j \subseteq T_d \)
    \item \textbf{sup}(n_j) = \text{sup}(n_j)+1;
    \item endif
    \item endfor
  \end{itemize}
  \item \textbf{end}
\end{itemize}
if(Sup(n_j) = 0)
    delete the node n_j from CI-tree(X_i);
    delete the n_j from HTC;
elseif(Sup(n_j) >= minsup(SW_{i+1}))
    update the key of the node n_j in HTC;
endif
endfor
endif
else
    generate all the candidate FCI of the transaction T_d by the algorithm Gen-FCI;
    for each FCI Y in the set of Cad-FCI or the subset of the transaction T_d
        /*generate by the above algorithm Can-(T)*/
        if Y with the prefix of X_i
            sup(Y) = sup(Y) + 1;
            if(sup(Y) = 0 or sup(Y) < minsup(SW_{i+1}))
                delete Y from the CI-tree(X_i);
                delete Y from the HTC;
            elseif(sup(Y) >= minsup(SW_{i+1}))
                update the key of Y in HTC;
            endif
            endif
        endif
    endfor
end

2) The add-new-tra algorithm. With the arrival of a data stream, add a transaction on the Top-k-FCI basic damped BW- sliding window. The closed frequent itemset in the HTC should be updated with the changing of the lattice CIL along adding a new transaction. The previous itemsets which are non-frequent itemsets may become frequent closed itemsets; and the frequent itemsets FCIs in the HTC, the support of them will increase too. The key values will change and then modify the closed itemsets tree. When adding a new transaction to the Top-k-FCI, the algorithm is as follows:

Add-new-tra Algorithm:
Input: A generated CIL of the current window;
Output: a new CIL after add the new transaction T_n;
Method
Begin:
    Cad(add) = subset(T_n) \cup Bd-(T_n)
    foreach item X_i in T_n
        foreach itemset Y in Cad-(add)
            if Y with the prefix of X_i
                if(HTC-check(Y) = false && sup(X_i) >= minsup(i))
                    insert Y into CI-tree(X_i);
                    /*because Y is a FCI*/
                    insert Y to the table HTC;
                    update the Y’s Bit-vector;
                    CI-num(X_i) = CI-num(X_i) + 1;
                    update the subset of the new FCI Y;
                    /*update the subset’s bit-vector, the key of the HTC*/
                else
                    /* the new itemset Y has already in the HTC */
                    sup(Y) = sup(Y) + 1;
                    update the bit-vector of Y;
                    update the key of Y in HTC;
                    update the subset of the new FCI Y;
                    /*update the subset’s bit-vector, the key of the HTC*/
                endif
            endif
        endfor
    endfor
end
3) The output(HTC) algorithm. The output of top-k frequent closed itemsets is the last step of Top-k-FCI algorithm, according to a given value of K. HTC recorded all the closed itemsets FCI, and the key value is the support count which stored in ascends order. Therefore the algorithm only has to check the number of FCI in the output set.

**output(HTC) Algorithm:**
Input: A HTC in the current window
Output: a set of top-k frequent closed items

Method
Begin:
set number =0;
for each key \( k_i \) of HTC do
  for each FCI \( X_i \) with the key \( k_i \) of HTC
    Output from HTC;
    number=number+1;
  endfor
  if(number\( \geq \)K) then
    break;
  endif
endfor
end

4. Experiment

The synthetic data set, T20I4D100K, generated by IBM synthetic data generator (Agrawal, Imielinski, & Swami, 1993; Agrawal & Srikant, 1994) is used to evaluate the performance of the proposed algorithm, where \( T \) is the average item per transaction, \( I \) is the average length of maximal patterns, and Test system using a Windows XP, experimental environment is Microsoft Visual C++6.0.

With the increase number of the windows in the data set, Top-k-FCI algorithm maintains the structure of CIL will consume the longer processing time. Updating the CIL structure will also need great memory space when delete and insert a transaction.

The following figure shows the accuracy of mining results with the proposed algorithm Top-k-FCI. We give the results of Top-k-FCI algorithm and the TKC-DS algorithm’s performance. From the experiments of Fig. 2, we can find that the proposed Top-k-FCI algorithm is efficient method for mining top-K frequent closed itemsets.

![Figure 2: the number of the FCI in the data stream](image-url)
5. Conclusions

In this paper, we study the problem of mining top-K frequent closed itemsets from continuous data streams. An efficient one pass, real-time mining algorithm, called Top-k-FCI (top-K frequent closed itemsets of data streams), is proposed. An effective algorithm about candidate itemset is developed for maintaining the essential information so far. Experiments show that the proposed algorithm is an efficient approach for finding top-K frequent closed itemsets from a basic damped data stream sliding window.

References

[9] Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, Philip S. Yu. FP-stream. Mining frequent patterns in data streams at multiple time granularities.2002.