Journal of Southeast University (English Edition) Vol. 24, No. 3, pp. 335 – 338 Sept. 2008 ISSN 1003—7985

New algorithm of mining frequent closed itemsets

Zhang Liang Ren Yonggong Fu Yu

(School of Computer and Information Technology, Liaoning Normal University, Dalian 116029, China)

Abstract: A new algorithm based on an FC-tree (frequent closed pattern tree) and a max-FCIA (maximal frequent closed itemsets algorithm) is presented, which is used to mine the frequent closed itemsets for solving memory and time consuming problems. This algorithm maps the transaction database by using a Hash table, gets the support of all frequent itemsets through operating the Hash table and forms a lexicographic subset tree including the frequent itemsets. Efficient pruning methods are used to get the FC-tree including all the minimum frequent closed itemsets through processing the lexicographic subset tree. Finally, frequent closed itemsets are generated from minimum frequent closed itemsets. The experimental results show that the mapping transaction database is introduced in the algorithm to reduce time consumption and to improve the efficiency of the program. Furthermore, the effective pruning strategy restrains the number of candidates, which saves space. The results show that the algorithm is effective.

Key words: frequent itemsets; frequent closed itemsets; minimum frequent closed itemsets; maximal frequent closed itemsets; frequent closed pattern tree

Mining frequent itemsets is an essential problem in many data mining fields. Most mining algorithms for frequent itemsets are based on Apriori. But these algorithms have a lot of deficiencies. Therefore, to address the problem that Apriori produces huge frequent itemsets, Pasquier et al. proposed the concept of a frequent closed pattern[1], which has an exact support count of frequent itemsets and provides a smaller size of frequent itemsets than that of the frequent closed pattern. At a present, many algorithms such as A-Close[1], MAFIA[2], CHARM[3], Closet +[4] and FPclose[5] have been proposed.

This paper proposes a new efficient algorithm to mine the frequent closed itemsets. A unique frequent closed pattern tree (FC-tree) is developed to organize the frequent itemsets. The algorithm has the pretreatment of a transaction database and stores frequent itemsets in a Hash table, which reduces the searching space and time. By using the breadth-first search strategy, we can decrease the operation of pruning during the process of building the tree and restrain the number of candidates, which saves space and improves the efficiency of the algorithm. Comparison experiments show that the algorithm is much more efficient.

Received 2008-04-15.

Biographies: Zhang Liang (1982—), female, graduate; Ren Yonggong (corresponding author), male, doctor, professor, renyonggong@gmail.com.

Foundation items: The National Natural Science Foundation of China (No. 60603047), the Natural Science Foundation of Liaoning Province, Liaoning Higher Education Research Foundation (No. 2008341).

Citation: Zhang Liang, Ren Yonggong, Fu Yu. New algorithm of mining frequent closed itemsets[J]. Journal of Southeast University (English Edition), 2008, 24(3): 335 – 338.

1 Basic Definitions

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items, and database $D = \{T_1, T_2, \ldots, T_n\}$ be a set of transactions. Itemset $X \subseteq I$, the support count of an itemset X is the number of transactions containing the itemset in a database D, denoted by $\sup\_count(X)$. The support of an itemset X is the ratio of the number of transactions containing the itemset to the number of all transactions in a database D, denoted by $\sup(X)$. If $\sup(X) \geqslant \text{minsup}$, called the minimum support, we denote X as a frequent itemset. In the frequent itemset, the number of itemsets is denoted as the dimension or the length.

Definition 1 Let itemset $X \subseteq I$, $T \subseteq D$, we define the mapping relationship:

$t: 2^I \rightarrow 2^T$, $t(X) = \{t \in T \mid \text{transaction support itemset } X\}$;

$i: 2^T \rightarrow 2^I$, $i(T) = \{c \in I \mid \text{any transactions contained in } T \text{ support itemset } c\}$.

Definition 2 Let itemset $X \subseteq I$, if $i(t(X)) = X$ and X is a frequent itemset, then X is denoted maximal frequent closed itemsets, i. e., frequent closed itemsets[6].

Definition 3 Let frequent itemset $X \subseteq I$, for any proper subset Y of X, being $\sup(X) < \sup(Y)$, then X is denoted minimum frequent closed itemsets[7].

2 Generation of Frequent Closed Itemsets

2.1 Frequent closed pattern tree (FC-tree)

Definition 4 (frequent closed pattern tree) FC-tree[8–10] is a multilayer tree which is based on the support of the frequent itemsets. The node of the tree is denoted as $h(t)$, and both h and t are frequent itemsets; i. e., every node of the tree is a frequent itemset. Concurrently, the node also saves the frequency that users reach the node by the same prefixal path, i. e., the support of the node. The structure of the FC-tree is as follows:

```
TYPE Nodeptr = ^Nodetype
Nodetype = RECORD
CurrentItem_gat[ITEM_NUM]: Integer;
SpareItem[ITEM_NUM]: Integer;
ChildNum: Integer;
ChildNode[ITEM_NUM], FatherNode: Nodeptr;
CurrentNode_child: Integer;
FrequentNum: Integer
```

Both CurrentItem_gat[ITEM_NUM] and SpareItem [ITEM_NUM] are used by saving the current items and candidate items. ChildNum denotes the number of the children. Both ChildNode[ITEM_NUM] and FatherNode are pointers which point out the node's children and node's father. Cur-

rentNode_child denotes that the child is the current node in the father's node. FrequentNum denotes the support of the itemset. ITEM_NUM is the number of the items of the frequent transaction database D.

2.2 Using FC-tree to mine frequent closed itemsets

Because the maximal frequent closed itemsets (frequent closed itemsets) are the subsets of the frequent itemsets, we can change the problem of finding all the frequent itemsets to searching all maximal frequent closed itemsets. Besides, frequent closed itemsets contain the support of all frequent itemsets. The process of searching frequent closed itemsets is divided into two steps:

Step 1 Finding out the minimum frequent closed itemsets;

Step 2 Confirming the maximal frequent closed itemsets through the minimum frequent closed itemsets (frequent closed itemsets).

However, before step 1, the database needs to be scanned. The information of the data, frequencies and transactions is stored in array. Whereafter, we use a bit vector to obtain the support of the extended subsets, and make use of the properties of minimum frequent closed itemsets as the pruning strategy. Finally, we create the frequent closed pattern tree first by the breadth, use the tree to save the frequent closed itemsets, and then determine the maximal frequent closed itemsets.

Property 1 Let itemset $X = \{x_1, x_2, \ldots, x_k\}$ be the minimum frequent closed k-itemset, then any $(k-1)$-sub-itemset is the minimum frequent closed $(k-1)$-itemsets.

Deduction 1 The FC-tree contains all the minimum frequent itemsets. If any child node is an infrequent itemset, then the father node is also an infrequent itemset.

Proof (reduction to absurdity) If father node Freque_father[i] is a frequent itemset, then based on property 1, any subsets of Freque_child must be a frequent itemset. Besides, Freque_child[k] is contained by Freque_child, then Freque_child[k] must be a frequent itemset too. However, it conflicts with the given that Freque_child[k] is not a frequent itemset. Then, father node Freque_father[i] is infrequent.

Property 2 Let itemset $X = \{x_1, x_2, \ldots, x_k\}$ be the minimum frequent k-itemset, then the support of any $(k-1)$-sub-itemset closed must be greater than the support of k-itemset.

Based on property 2, we can have the algorithm of mining minimum frequent closed itemsets.

Algorithm 1 Mining minimum frequent closed itemsets

Input: The quantity and address of the frequent n-itemsets;

Output: The address of the frequent $(n+1)$-itemsets.

$k = 0$

For $i = 1$ to Frequent Num //Frequent Num is the quantity of frequent n-itemsets in the FC-tree

For $j = 1$ to Children Num //Children Num is the quantity of Freque_father[i]'s children

Freque_child[k] saves Freque_father[i]'s child node

k++

EndFor

EndFor

For $i = 1$ to k //k is the quantity of frequent $(n+1)$-itemsets

If the frequent of Freque_child[i] = frequent_father; //frequent_father is the father node's frequent

Delete_node(Freque_child[i]);

Else For j = the position of the father node to the position of the last frequent n-itemset

If all of the members in Freque_child[i] itemset < the minimum members in Freque_father[j] itemset

Break;

Else If (the max member in Freque_child[i] itemset > the max member in Freque_father[j] itemset) && (Freque_father[j] itemset. Freque_child[i] itemset) && (Freque_child[i]'s frequent = Freque_father[j]'s frequent)

Delete_node(Freque_child[i]);

Break;

EndIf

EndFor

EndIf

EndFor

Property 3 Let X be the maximal frequent closed itemset, then there is the minimum frequent closed itemset Y, $t(X) = t(Y)$ is available.

Property 4 Let X be the maximal frequent closed itemset, then $i(t(X))$ is the maximal frequent closed itemset.

Deduction 2 Let Y be the minimum frequent closed itemset in transaction database D, then corresponding Y's maximal frequent closed itemset is $i(t(Y))$.

Deduction 3 Let minFCI be a set of minimum frequent closed itemsets in transaction database D, then maxFCI = $\{i(t(Y)) \mid Y \in \text{minFCI}\}$.

Property 5 Given itemset $Y \in \text{minFCI}$, $t(Y) = \{T_{Y_1}, T_{Y_2}, \ldots, T_{Y_m}\}$, then $i(t(Y)) = \cap T_{Y_j}$.

Based on deduction 3 and property 5, we have the algorithm of mining maximal frequent closed itemsets.

Algorithm 2 max-FCIA (maximal frequent closed itemsets algorithm)

Input: The root node of the tree; transaction database D';

Output: All maximal frequent closed itemsets in D'.

Find all of the minimum frequent closed itemsets Y in the tree through the root node, and then put Y in minFCI

$Y \in$ minFCI

For all $Y \in$ minFCI do

MY = T; //MY is used to save maximal frequent closed itemsets, originally the whole set is T

//T is the set of frequent 1-itemsets of the tree.

For all transactions $t \in D'$ do

For all $Y \in$ minFCI do

If $Y \subseteq t$ then

MY = MY $\cap$ t;

maxFCI = $\varnothing$;

For all $Y \in$ minFCI do

MaxFCI = maxFCI $\cup$ MY

3 Example of Application

Example Given $I = \{1, 2, 3, 4, 5, 6, 7\}$, the transaction database D's content is shown in Tab. 1.

Tab. 1 Transaction database D

Tid	Itemset
T_1	1, 2, 3, 4, 6
T_2	2, 4, 5, 7
T_3	1, 2, 3, 4, 5, 6
T_4	1, 2, 3, 4, 5
T_5	1, 2, 5, 7
T_6	2, 3, 4, 5

1) The pretreatment of the transaction database: Given the minimum support threshold minsup = 3/6 = 50%, delete the items which are infrequent items. Therefore, we can obtain the frequent transaction database D' as shown in Tab. 2.

Tab. 2 Transaction database D'

Tid	Itemset
T_1	1, 2, 3, 4
T_2	2, 4, 5
T_3	1, 2, 3, 4, 5
T_4	1, 2, 3, 4, 5
T_5	1, 2, 5
T_6	2, 3, 4, 5

2) Generate the minimum frequent closed itemsets: Using 1-frequent itemsets and Hash, we obtain the support of the child node. According to property 2 of minimum frequent closed itemsets, the algorithm prunes and builds the tree. From the root node, we compare the father node with the child node to their supports. For instance, $\sup(\{1\}) = \sup(\{1, 2\})$, delete the frequent itemset $\{1, 2\}$ from C_2, then we obtain the minimum frequent closed 2-itemsets $L_{c2} = \{\{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}, \{4, 5\}\}$. Also, delete the frequent itemsets $\{1, 3, 4\}$, $\{2, 3, 4\}$ from C_3, we obtain $L_{c3} = \{\{1, 3, 5\}, \{1, 4, 5\}, \{2, 3, 5\}, \{2, 4, 5\}, \{3, 4, 5\}\}$. Therefore, the minimum frequent closed itemsets in D' are $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}, \{4, 5\}, \{1, 3, 5\}, \{1, 4, 5\}, \{2, 3, 5\}, \{2, 4, 5\}, \{3, 4, 5\}\}$.

We can obtain the minimum frequent closed itemsets which is based on the FC-tree (see Fig. 1).

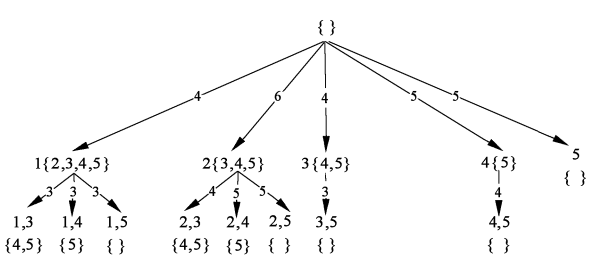

Fig. 1 minimum frequent closed itemsets based on FC-tree

3) Generate the maximal frequent closed itemsets: Given $M_{\{1\}} = M_{\{2\}} = M_{\{3\}} = M_{\{4\}} = M_{\{5\}} = M_{\{1,3\}} = M_{\{1,4\}} = M_{\{1,5\}} = M_{\{2,3\}} = M_{\{2,4\}} = M_{\{2,5\}} = M_{\{3,5\}} = M_{\{4,5\}} = M_{\{1,3,5\}} = M_{\{1,4,5\}} = M_{\{2,3,5\}} = M_{\{2,4,5\}} = M_{\{3,4,5\}} = \{1, 2, 3, 4, 5\}$. $T_1 = \{1, 2, 3, 4\}$ because T_1 supports $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1, 3\}$, $\{1, 4\}$, $\{2, 3\}$ and $\{2, 4\}$ in minimum frequent closed itemsets, we modify $M_{\{1\}}$, $M_{\{2\}}$, $M_{\{3\}}$, $M_{\{4\}}$, $M_{\{1,3\}}$, $M_{\{1,4\}}$, $M_{\{2,3\}}$ and $M_{\{2,4\}}$, the principle of modification is $M_Y = M_Y \cap \{1, 2, 3, 4\}$. For example, $M_{\{1\}} = \{1, 2, 3, 4, 5\} \cap \{1, 2, 3, 4\} = \{1, 2, 3, 4\}$, $M_{\{5\}}$, $M_{\{1,5\}}$, $M_{\{2,5\}}$, $M_{\{3,5\}}$, $M_{\{4,5\}}$, $M_{\{1,3,5\}}$, $M_{\{1,4,5\}}$, $M_{\{2,3,5\}}$, $M_{\{2,4,5\}}$ and $M_{\{3,4,5\}}$ are not changed. Deal with T_2, T_3, T_4, T_5, T_6 similarly. Finally, we obtain the maximal frequent closed itemsets: $\max\text{FCI} = M_{\{1\}} \cup M_{\{2\}} \cup M_{\{3\}} \cup M_{\{4\}} \cup M_{\{5\}} \cup M_{\{1,3\}} \cup M_{\{1,4\}} \cup M_{\{1,5\}} \cup M_{\{2,3\}} \cup M_{\{2,4\}} \cup M_{\{2,5\}} \cup M_{\{3,5\}} \cup M_{\{4,5\}} \cup M_{\{1,3,5\}} \cup M_{\{1,4,5\}} \cup M_{\{2,3,5\}} \cup M_{\{2,4,5\}} \cup M_{\{3,4,5\}} = \{\{2\}, \{1, 2\}, \{2, 4\}, \{2, 5\}, \{1, 2, 5\}, \{2, 3, 4\}, \{2, 4, 5\}, \{1, 2, 3, 4\}, \{2, 3, 4, 5\}\}$.

4 Experimental Evaluation

To prove the superiority of the algorithm max-FCIA, the experiments are performed on a PC with 512 MB of main memory and a Microsoft Windows XP server. We execute max-FCIA and A-Close. In the experiment, the T10I4D100K dataset is used. The database's characteristics are shown in Tab 3. The experimental results on databases are shown in Fig. 2, which demonstrate that the max-FCIA is efficient.

Tab. 3 Database characteristics

Database	Average length	Records
T10I4D100K	10	1 000

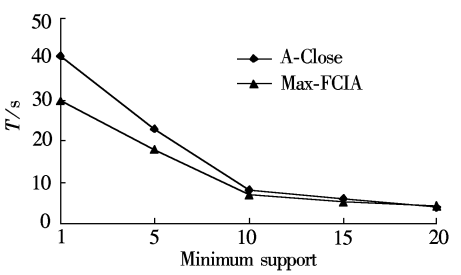

Fig. 2 Time comparison on T10I4D100K

5 Conclusion

We present an FC-tree for storing frequent closed itemsets, propose max-FCIA algorithm for mining frequent closed itemsets, and improve on the most important techniques. Applying the techniques for mining the maximal frequent itemsets will be our principal work in future.

References

[1] Pasquier N, Bastide Y, Taouil R, et al. Discovering frequent closed itemsets for association rules[C]//*Proc of the 7th Intl Conf on Database Theory*. Springer-Verlag, 1999: 398 − 416.

[2] Burdick D, Calimlim M, Gehrke J. MAFIA: a maximal frequent itemset algorithm[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2005, **17**(11): 1490 − 1504.

[3] Zaki M J, Hsiao C J. CHARM: an efficient algorithm for closed itemset mining[C]//*Proc of the 2nd SIAM Intl Conf on Data Mining*. Arlington: SIAM, 2002: 12 − 28.

[4] Wang Jianyong, Han Jiawei, Pei Jian. Closet + : searching for the best strategies for mining frequent closed itemsets[C]//*Proc of ACM SIGKDD'03*. Washington, DC, 2003: 236 − 245.

[5] Grahne G, Zhu J. Efficiently using prefix-trees in mining frequent itemsets [C]//*Proc of the IEEE ICDM Workshop on*

Frequent Itemset Mining Implementation (*FIMI*'03). Melbourne, Florida, USA, 2003.
[6] Yahia S B, Hamrouni T, Nguifo E M. Frequent closed itemset based algorithms: a thorough structural and analytical survey [J]. *SIGKDD Explorations*, 2006, **8**(1): 93 - 104.
[7] Zhu Yuquan, Song Yuqing. Research on an algorithm for mining frequent closed itemsets[J]. *Journal of Computer Research and Development*, 2007, **44**(7): 1177 - 1183. (in Chinese)
[8] Liu Junqiang, Sun Xiaoying, Zhuang Yueting, et al. Mining frequent closed patterns by adaptive pruning[J]. *Journal of Software*, 2004, **15**(1): 94 - 102. (in Chinese)
[9] Lü Cheng, Hao Ying, Zhang Hantao. Algorithm of mining frequent patterns based on the vertical bitmap[J]. *Journal of Shandong University*, 2007, **42**(5): 24 - 29. (in Chinese)
[10] Zhu Yuquan, Yang Hebiao, Sun Lei. *Data mining techniques* [M]. Nanjing: Southeast University Press, 2006: 27 - 77. (in Chinese)

一种新的频繁闭项目集挖掘算法

张 亮 任永功 付 玉

(辽宁师范大学计算机与信息技术学院,大连 116029)

摘要:为了解决频繁闭项目集挖掘中时间和存储开销大的问题,提出了一种基于 FC-tree(频繁闭模式树)的频繁闭项目集挖掘算法 max-FCIA(最大频繁闭项目集挖掘算法). 该算法利用哈希表映射事务数据库,通过对哈希表进行操作从而得到所有频繁项目集的支持度,进而生成包含所有频繁项目的有序树. 经过剪枝处理的有序树就是包含所有最小频繁闭项目集的 FC-tree,最后用最小频繁闭项目集生成频繁闭项目集. 实验结果表明,该算法通过映射事务数据库,减少了扫描数据库所浪费的时间,提高程序执行效率. 另外,运用有效的剪枝策略,避免了不必要候选项目集的生成,节省了存储空间,实验证明该算法是有效的.

关键词:频繁项目集;频繁闭项目集;最小频繁闭项目集;最大频繁闭项目集;频繁闭模式树
中图分类号:TP311.13