PROCEEDINGS OF SCIENCE

An Improved Frequent Pattern-growth Algorithm Based on Decomposition of the Transaction Database

Laisheng Xiang[1]
School of Management Science and Engineering, Shandong Normal University,Jinan, 250014,China
*E-mail:*459132653@qq.com

Fei Wei[2]
School of Management Science and Engineering, Shandong Normal University,Jinan, 250014,China
*E-mail:*weifei1027@163.com

Xiyu Liu[3]
School of Management Science and Engineering, Shandong Normal University,Jinan, 250014,China
*E-mail:*xyliu@sdnu.edu.cn

FP-Growth algorithm is an algorithm of mining frequent patterns based on the data structure of FP-Tree(Frequent Pattern-Tree), which runs slowly and takes up a lot of ram during the mining process of large transaction database and even sometimes cannot construct the corresponding FP-tree. On the basis of the original algorithm, a new improved approach is put forward, which uses less ram and can meet the requirement of large transaction database. Firstly, calculate the support of all items in the transaction database, exclude the item which cannot satisfy the minimum support and get frequent items, then construct their sub-databases in turn according to ascending order of the support of these frequent items; finally, merge frequent item sets of each sub-database in order to gain frequent patterns of the entire transaction database. In this paper, some instances and experiments are carried out. Experimental results demonstrate that the improved FP-Growth algorithm is more efficient than traditional FP-Growth algorithm and suitable for the mining of large-scale transaction databases.

CENet2015
12-13 September 2015
Shanghai, China

[1]Corresponding Author
[2]Speaker
[3]This research was supported by the Natural Science Foundation of China (No.: 61170038, 61472231), the Humanities and Social Sciences Project of the Ministry of Education of China (No.:12YJA630152), the Outstanding Young Scientist Award Foundation of Shandong Province (No.:BS2013DX037) and A Project of Higher Educational Science and Technology Program of Shandong Province (No. :J15LN28).

PoS(CENet2015)023

1.Introduction

The association rule is one of the most widespread topics in the area of data mining[1, 2]. Since the concept of the association rules was put forward, many researchers have carried out a large number of studies on association rule mining problems, in which frequent item set mining algorithm is an important research direction. Among many algorithms, FP-Growth algorithm is the most famous, whose core is that all information of transactions are compressed to a FP-Tree in order to get the database mappings for frequent item set mining[3]. Afterwards, its conditional FP-Tree is constructed for each frequent item to mine frequent patterns.

Whereas, construction of a FP-Tree requires a lot of rams. When a transaction database is large to a certain extent, the running speed of the algorithm could be greatly reduced or the FP-Tree based on the ram cannot be constructed [4], which leads to the failure of mining. To solve the problems existing in the traditional algorithm, an improved method is proposed, which obtains the sub-database of each frequent item through decomposition of the transaction database. And then, mine each sub-database respectively and acquire frequent patterns of the entire transaction database. Experiments prove that this improved algorithm takes less ram and is at least two times faster than the traditional algorithm during a large transaction database mining.

2. Traditional FP-Growth Algorithm

The basic idea of traditional FP-Growth algorithm is to compress the transaction by a tree structure while preserving the relationship between properties in transactions. It does not produce the candidate sets and uses the method of increasing frequent sets for data mining. The traditional FP-Growth algorithm is mainly the construction process of FP-Tree [5], whose main steps are shown as follows:

1. Construct FP-Tree according to the original transaction database.

(1) Firstly, scan the transaction database, export a collection of frequent items (1 item set) and corresponding support count and then find items to meet the minimum support count. Frequent items of the collection are arranged in descending order of the support count with the result collection denoted by *P*.

(2) Construct the original FP-Tree, create the root node of the tree, and tag it with “null”.

(3) Scan the transaction database twice. The terms of each transaction are processed according to the order (i.e. support count in descending order) in collection *P* and a branch is created for each transaction. In general, when a branch for a transaction is considered to be added, the count of each node along the common prefix increases by 1 and then new node and link is created for the item after the prefix.

(4) To facilitate the traversal, a frequent item header table is created so that each node directs its position in the tree by a point.

2. Mine frequent patterns on the FP-Tree recursively.

With regard to each frequent item (initial suffix mode) whose length is 1, construct its conditional model base and FP-Tree, and then mine the FP-tree recursively. The growth model is achieved by connecting the suffix model and frequent patterns which are produced by the conditional FP-Tree.

In the area of frequent pattern mining algorithm, FP-Growth algorithm opens a new way to effectively mine frequent patterns; however, its efficiency of time and space is not high enough. When a transaction database is so big, FP-Tree based on the ram sometimes is not successfully constructed. Simultaneously, the conditional FP-Tree is generated recursively in the course of mining frequent patterns. Generations and releases of conditional FP-Tree will spend a great quantity of time and space [6].

3. Improved FP-Growth Algorithm

Aiming at the shortcomings of traditional FP-Growth algorithm [7], this paper raises an improved FP-Growth algorithm based on decomposition of the transaction database. Its main core is to compose the sub-database for every frequent item whose length is 1 by scanning the transaction database, mining frequent patterns for each sub-database in view of its FP-Tree and then achieving corresponding frequent item sets. The sum of frequent item sets of each sub-database is all frequent patterns of the entire transaction database.

3.1 FP-Growth Algorithm Based on Decomposition of the Transaction Database

Assume that the transaction database be S and minimum support be *min_sup*. The implementing process of the improved FP-Growth algorithm is shown as follows:

1. Scan the transaction database S and get the collection M of all items whose length is 1 and their corresponding support is U; then delete the item whose support is below *min_sup*. At the same time, sort other items in descending order and gain the collection P. Set $P = \{p_n, p_{n-1}, ..., p_2, p_1\}$, and so $U(p_n) >= U(p_{n-1}) >= \ldots >= U(p_1)$.

2. Secondly, scan the transaction database S, remove the item whose support is below *min_sup* from each transaction in transaction database S and arrange other items according to the order in collection P; therefore, the new transaction database S' is produced.

3. Decompose the transaction database S' on the basis of the decreasing order of support count, and build up the sub-database for each item whose length is 1 in turn. Later, construct the corresponding FP-Tree so as to mine frequent patterns for each sub-database.

For each item p_m (m=1, 2, …, n-1, n) in collection P:

(1) Scan transaction database S'. Extract all transactions which contain the item p_m from it and compose the sub-database S_m for item p_m by combining them. Meanwhile, delete all item p_m to update the database S'.

(2) Scan the sub-database S_m and construct its FP-Tree thus to get frequent patterns by mining it. The achieved frequent item sets which contain the item p_m are recorded as Q_m.

When all frequent item sets Q_m (m=1, 2, …, n-1, n) of each item in collection P have been mined, merge all of them in order to obtain frequent patterns of the original transaction database S.

As we can see, the improved FP-Growth algorithm has the following two advantages when compared with the traditional FP-Growth algorithm.

1. Omit the construction of FP-Tree for the entire transaction database; at meanwhile, build the FP-Tree for each sub-database and mine it for frequent item sets respectively, which has reduced the complexity of space. In addition, the improved FP-Growth algorithm avoids the possibility of failure of constructing FP-Tree based on the ram in the traditional FP-Growth algorithm.

2. This improved FP-Growth algorithm decomposes the transaction database and separately mines each sub-database, which have reduced the mining difficulty and improved the efficiency greatly.

3.2 Algorithm Example

An example is given to clearly describe the execution of the improved FP-Growth algorithm based on decomposing the transaction database. In this case, database S includes 10 transactions, as shown in Table 1. Set *min_sup* is 2% and so support count is 2.

TID	Items	TID	Items
T100	B C D	T600	A B D E
T200	A C	T700	A B C
T300	A B F	T800	A B G
T400	A C	T900	B C E
T500	A B C	T1000	B D

Table 1: Database S

First of all, scan the transaction database *S* and obtain a collection *M* in which the length of item is 1. Remove the item *F* and *G* from database *S* whose support count is less than 2 and put other items in descending order so as to get the collection *P*={*B*:8, *A*:7, *C*:6, *D*:3, *E*:2}. Then, in terms of the order in collection P, rearrange database *S* to get the database *S* ′, which is shown in Table 2.

TID	Items	TID	Items
T100	B C D	T600	B A D E
T200	A C	T700	B A C
T300	B A	T800	B A
T400	A C	T900	B C E
T500	B A C	T1000	B D

Table 2: Database *S'*

According to the increasing order of support count of items in collection P, construct their corresponding sub-databases in turn. Here, we use item *E* as an example. Table 2 shows that the sub-database for item *E* is made up of transaction *T600* and *T900*; at the same time, delete all item *E* from database *S'*. With this analogy, the sub-databases of all items are shown as follows.

Sub-database of item *E*: *B A D E*; *B C E*

Sub-database of item *D*: *B C D*; *B A D*; *B D*

Sub-database of item *C*: *B C*; *A C*; *A C*; *B A C*; *B A C*; *B C*

Sub-database of item *A*: *A*; *B A*; *A*; *B A*; *B A*; *B A*; *B A*

Sub-database of item *B*: *B*; *B*; *B*; *B*; *B*; *B*; *B*; *B*

Finally, separately mine each sub-database for frequent item sets. Frequent patterns for the entire transaction database *S* are shown in Table 3.

Table 3: Frequent Patterns of Database *S*

Frequent item set	Support	Frequent item set	Support
E	20%	B C	40%
B E	20%	B A C	20%
D	30%	A	70%
B D	30%	B A	50%
C	60%	B	80%
A C	40%		

4. Experiments and Analysis

At the moment of mining the large transaction database, the improved FP-Growth algorithm runs faster and the efficiency is greatly increased when compared to the traditional FP-Growth algorithm. In order to confirm the effectiveness of this algorithm, the following experiments are carried out on the basis of improved points.

Experimental environment is shown as follows: the operating system is Windows XP, the development environment is the VC6.0 and the database has 20, 0000 transactions. Under this environment, the traditional FP-Growth algorithm and improved FP-Growth algorithm are all achieved. Test the database Respectively when the support is 5%, 15%, 25%, 35%, 45%, 55%. Results of the mining time for test are shown in Table 4.

Support (%)	Mining Time (s)	
	Traditional FP-Growth Algorithm	Improved FP-Growth Algorithm
5%	——	4555
15%	4480	2301
25%	3855	1095
35%	2634	810
45%	1885	664
55%	1267	501

Table 4: Mining Time of Two Algorithms under Different Supports

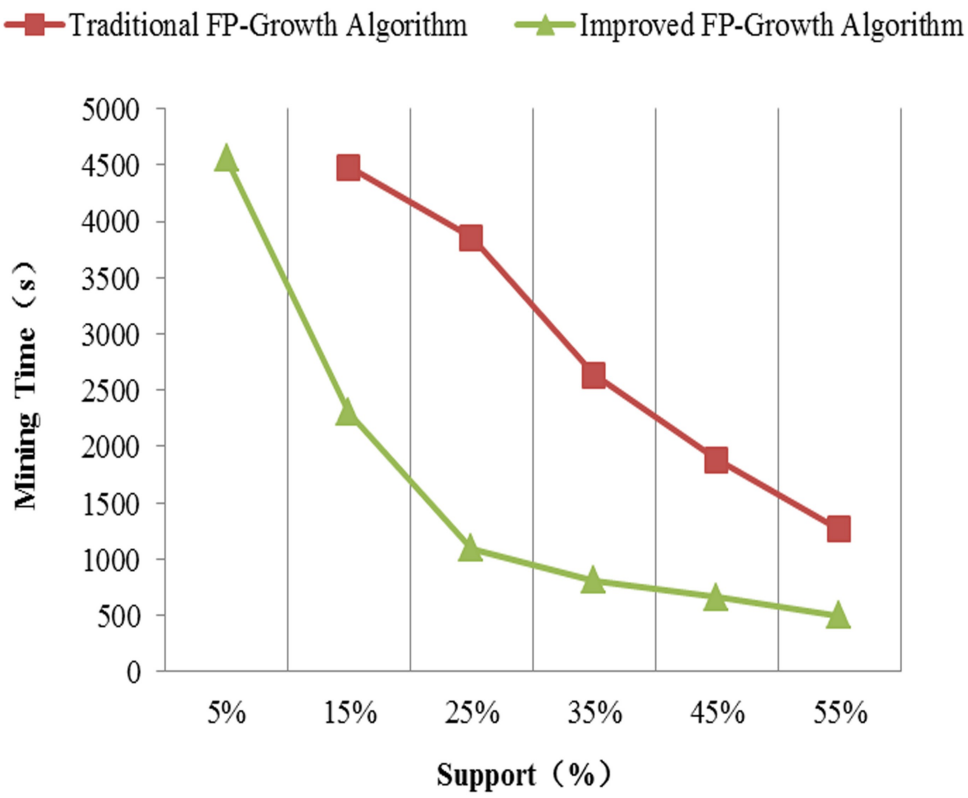

Figure 1: Comparison chart of mining time of two algorithms under different support

As shown in Table 4 and Fig. 1, when the support is 5%, the traditional FP-Growth algorithm is unable to mine the transaction database, but improved FP-growth algorithm can. When the support is respectively of 15%, 25%, 35%, 45%, 55%, the improved FP-Growth algorithm will use less time than traditional FP-Growth algorithm. In addition, by comparing the mining results (frequent patterns of the entire transaction database) of these two algorithms, the improved FP-Growth algorithm does not exert an impact on the mining results of the traditional FP-Growth algorithm, which has proved the effectiveness of this improved FP-Growth algorithm.

Upon analysis of the mining time of these two algorithms, we can see, when mining small transaction databases, the traditional FP-Growth algorithm runs faster and more efficiently. In contrast, it can operate slowly or doesn't build FP-Tree on the basis of ram when confronted with large transaction databases, which has resulted in the failure of mining; nevertheless, the improved FP-Growth algorithm mines each sub-database respectively by means of decomposing the database and obtains frequent patterns of this entire database by merging all their frequent item sets. It is faster and has been greatly improved in face of large transaction databases; however, this method is relatively complex and thus not applicable to mine small transaction databases.

5. Conclusion

As the efficiency of the traditional FP algorithm is low when mining large databases, the new association rule algorithm based on decomposition of the database is proposed, which runs by decomposing the entire transaction database. Then, it mines frequent item sets as to each sub-database respectively because the mining can be separated from the original transaction database. When the support is smaller or the database is larger, the improved FP-Growth algorithm can alleviate the huge demand of ram in the traditional FP-Growth algorithm and occupies less ram, which has greatly improved the efficiency.

References

[1] R . Agrawal, R. Srikant. *Fast algorithms for mining association rules*. In: Proc of 1994 Int'l Conf on Very Large Data Bases. San-tiago, Chili: VLDB Endowment.487-499(1994)

[2] J. S. Park, M. S. Chen, P. S. Yu. *An effective Hash-based algorithm for mining association rules.* In: Proc of 1995 ACM-SIGMOD Int'l Conf on Management of Data. San Jose. CA: ACM Press.175-186(1995)

[3] J. Han, J. Pei, Y. Yin. *Mining frequent patterns without candidate generation*. In: Proc of 2000 ACM-SIGMOD Int'l Conf on Management of Data. Dallas. TX: ACM Press.1-12(2000)

[4] X. P. Liu. *Research and application of association rule mining algorithm based on FP-Growth algorithm*. Hunan University, Hunan (2006)

[5] J. W. Han, Kamber. M, *Data mining concept and techniques*. Second edition. China machine press.157-159(2001)

[6] M. Fan, C. Li. *Mining frequent patterns in an FP-tree without FP-Tree generation*. Journal of computer research and development .40(8): 1216-1222. (2003)(In Chinese)

[7] C. Liu, X. H. Chen. *Research and improvement of FP-Tree algorithm in association rules.* Network security and technology (2012) (In Chinese)