International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:4, 2007

Generating Frequent Patterns through Intersection between Transactions

M.jamali, F.taghiyareh

***Abstract*—** the problem of frequent itemset mining is considered in this paper. One new technique proposed to generate frequent patterns in large databases without time-consuming candidate generation. This technique is based on focusing on transaction instead of concentrating on itemset. This algorithm based on take intersection between one transaction and others transaction and the maximum shared items between transactions computed instead of creating itemset and computing their frequency. With applying real life transactions and some consumption is taken from real life data, the significant efficiency acquire from databases in generation association rules mining.

***Keywords*—**association rules, data mining, frequent patterns, shared itemset

I. Introduction

The amount of data stored in databases continues to grow fast. Intuitively, this large amount of stored data contains valuable hidden knowledge, which could be used to improve the decision-making process of an organization especially in decision support systems. The knowledge obtained by these data doesn't use all data and there is great gap between knowledge and data and this gap between our knowledge and data increased rapidly. Data mining applications and association rules try to decrease this gap. The discovery of relationships between items in one database can be very useful to increase the power of decision makers in one organization.

The problem of finding association rules first was introduced by Agrawal and his cooperators in [1].The frequent itemset and association rule mining problem have received a great deal of attention and many algorithms have been proposed to solve this problem[10,11,12,13,14,15].

Discovering association rules in these algorithms usually done in two phase. In the first phase frequent itemset generated and then interesting rules extract from frequent itemset. The task of discovering all frequent itemset is quite challenging especially in large databases. The database could be massive, containing million of transactions. One of the main problems that mining frequent itemset is suffered is scalability of them into large databases. In large databases generating all candidate itemset and combining them to generate frequent itemset is too time consuming. Some algorithms such as FP-growth [7] have been proposed to solve this problem without generating candidate itemset but that complexity is high and require much memory to create those data structures.

In this paper a new technique was proposed to discovery frequent itemset without consuming long time for generating all candidate itemset.

This paper is organized as follows. In first section the some definition for mining frequent itemset and problem of finding association rules in large databases, is described. In second section our technique to discovery frequent itemset is proposed. In section III we discuss about finding association rules based on frequent itemset that discovered in previous section. In next section some experimental result is described and in last section conclusion of this paper will proposed.

M. jamali. Graduate Student in department of electrical and computer factuality of engineering, university of Tehran, Iran(e-mail: mjamali@ ece.ut.ac.ir).

F. Taghi yareh. Professor assistant in department of electrical and computer engineering-factuality of engineering, university of Tehran, Iran(e-mail: ftaghiyar@ ut.ac.ir).

II. Problem definition

Let I be a set of items. A set $X= \{i_1,....,i_k\} \subseteq I$ is called itemset or a k-itemset if contain k items. A transaction over I is a couple T = (tid , I) where tid is the transaction identifier and I is an itemset.

An association rule is an expression of the form X=> Y , where X and Yare itemset, and $X \cap Y = \{\}$. Such a rule expresses the association that if a transaction contains all items in X, then that transaction also contains all items in Y . X is called the body or antecedent, and Y is called the head or consequent of the rule.

The support of an association rule X => Y in database D is the support of $X \cup Y$ in D, and similarly, the frequency of the rule is the frequency of $X \cup Y$. An association rule is called frequent if its frequency exceeds a given minimal frequency threshold σ. The measure that demonstrates the possibility of presence Y in the transaction, when X occurs in it, is *confidence*. The confidence of the rule X=> Y is the fraction of transaction containing X which also contains Y. The rule is called confident if its confidence exceeds a given minimal confidence threshold denoted by γ.

In association rule mining we want to find all rules with the frequent and confidence above minimum threshold for frequent and confidence.

One of the most important problems with scalability of known algorithms for mining frequent itemset is the high number of candidates has to be generated. For generating long frequent pattern, all of its subset must be generated and frequency of them must be computed and then its frequent subset are been

combined to achieve one frequent pattern. This process takes a long time. For example to generate one frequent pattern of length 100, such as $\{a_1,a_2,\ldots,a_{100}\}$, the number of candidates that has to be generated will be at least 10^{30}. This number is obtained as below:

$$\sum_{i=1}^{100} \binom{100}{i} = 2^{100}-1 \approx 10^{30}$$

And it will require hundreds of database scans. The complexity of the computation increase exponentially. The scalability of patterns one of the main factors that influence the development of several new algorithms for association rules mining. An efficient method is frequent pattern growth that proposed to solve this problem [7]. This algorithm mines frequent itemset without the time consuming candidate generation process is essential for other known algorithms such as apriori[6] and its optimization, aprioriTid and aprioriHybrid[9]. Another algorithm that proposed and tried to solve this problem is DIC [1, 2]. In this algorithm itemset are dynamically added and deleted as transactions are read. This algorithm relies on the fact that for an itemset to be frequent, all of its subsets must also be frequent, so we only examine those itemset whose subsets are all frequent.

III. Frequent Itemset Mining

If there is any association between itemset X and Y really, we hope there is union X and Y at least one time in database transactions. It's means that if we have X=>Y, it is reasonable we had (X,Y) lonely at least one time in database transactions. In other words if itemset (A,B,C) frequent we must have this itemset at least in one transaction without any other item. That transaction in database layout must be in this schema:

	A	B	C	.....
Tid	1	1	1	000000000

Analyzing some real life transactions has showed us almost all of association rules have this property.

In some applications, the transaction database must have to be mined frequently to capture customer behavior. In such applications, the efficiency of data mining could be more important factor than the complete accuracy of results. In addition, in some applications the problem domain may be vaguely defined. Missing some marginal cases that have confidence and support levels at the borderline may have little effect on the quality of solution to the original problem. *Sampling* algorithm proposed by Toivonen[6] is attractive approach to find association rules in tiny time with lost a little accuracy. Another technique that used to discovery association rules in efficient time is *incremental updating* [8] that avoid mining the whole updated database again.

Definition 1: Let D be a transaction database over a set of items *I*, and σ a minimal support threshold. The collection of frequent itemset in D with respect to σ is denoted by

$F(D, \sigma) := \{X \subseteq I \mid support(X,D) >= \sigma \}$

The collection of frequent itemset F(D, σ) can be represented by the collection of maximal frequent itemset, or the collection of minimal infrequent itemset, with respect to set inclusion. For this purpose, Mannila and Toivonen introduced the notion of the Border of a downward closed collection of itemset [3].

Definition 2: (Border) Let F be a downward closed collection of subsets of *I*. The Border Bd(F) consists of those itemset $X \subseteq I$ such that all subsets of X are in F, and no superset of X is in F:

$Bd(F)=\{X \subseteq I \mid \forall Y \subset X: Y \in F \wedge \forall Z \supset X: Z \notin F\}$

Those itemset in Bd(F) that are in F are called the positive border $Bd^+(F)$:

$Bd^+(F)=\{X \subseteq I \mid \forall\ Y \subseteq X : Y \in F\ \wedge \forall Z \supset X: Z \notin F\}$

Several efficient algorithms have been proposed to find only the positive border of all frequent itemset. From a theoretical point of view, the border gives some interesting insights into the frequent itemset mining problem, and still poses several interesting open problems [3, 4, and 5].

The aim of this paper generate frequent itemset that are in $Bd^+(F)$ without creation small itemset and combining them to generate super itemset that frequent.

In our technique intersection of every transaction and other transaction is been computed. Result of intersection is been converted to decimal numbers. Intersecting between any transaction and other transaction is done by bitwise the string of transactions.

If we want to intersect these two transactions, the string of two transactions is and-ed in bitwise manner. We use notation T[i], which represent occurrence items in ith transaction, composed of some 0 and 1 bit.

For example in figure 1 we have two transactions. In first transaction T[1] is equal to 0111010 and in second transaction T[2] is 1101110.The result of intersecting between two transactions is: 0101010. This result represent that items 2, 4, 6 are shared with two transactions. More detail of this technique is illustrated in algorithm 1.

T_{id}	I_1	I_2	I_3	I_4	I_5	I_6	I_7
1	0	1	1	1	0	1	0
2	1	1	0	1	1	1	0

Figure1

Algorithm 1

---

frequent-set={}
i=0
For i:=1 to DBsize do
Shared-set={}
For j:=1 to DBsize do
S=T[i]&T[j]
If there isn't S in shared-set Add (S,1) to shared-set else add (S,++frequency) to shared-set.
End-for
If there is any *C* in share-set that *C*.frequency>= σ then
frequent-set[i]=*C;*
i++;
End-if
End-for

After the first phase the frequent itemset are in frequent-set. We must extract their all subset of them and create association rules between its items. If those association rules are confident, they add to association rules.

IV. Association rules mining

In this stage the frequent itemset have been computed and are in frequent-set[i]. For mining association rules, first, all frequent itemset are generated using Algorithm 1. Then, every frequent itemset I are in frequent-set is divided into a candidate head Y and a body X = I \Y. This process starts with Y = {}, resulting in the rule I) {}, which always holds with 100% confidence. After that, the algorithm iteratively generates candidate heads C_{k+1} of size k + 1, starting with k = 0.To compute the confidence of a candidate head Y, the support of I is retrieved from frequent-set and support X is computed by bit wise logical anding between extended X and transaction in database . Any itemset X extended by adding 0 for those itemset that not occur in X. for example if itemset X haven't item 1 and 9, the first and 9^{th} bit will be zero. All heads that result in confident rules are inserted into H_k. In the end, all confident rules are inserted into R. Detail of this algorithm described in Algorithm2 in below:

Algorithm 2

Input: D, σ, γ
Output: R(D,σ,γ)
1: Compute F(D, σ)
2: R := {}
3: for all I ∈ F do
4: R := R ⋃ I => {}
5: C1 := {{i} | i ∈ I};
6: k := 1;
7: while Ck ≠ {} do
8: // Extract all heads of confident association rules
9: H_k := {X ∈ C_k | confidence(I \ X=>X,D) >= γ }
10: // Generate new candidate heads
11: I = X ⋃ {Y [k]}
12: if ∀J ⊂ I, |J| = k : J ∈ H_k then
13: C_{k+1} := C_{k+1} ⋃ I
14: end if
15: k++
16: end while
17: // Cumulate all association rules
18: R := R⋃ {I \ X) X | X ∈ H1⋃ · · · ⋃ Hk}
19: end for

V. Experimental result

Our implementation of this algorithm on synthetic database showed us that by using this method the association rules mining is dine in low time and the efficiency of algorithm in compromise with other algorithms is better. But in this algorithm some of our association rules may be lost, although number of losted association rules, isn't high and there isn't important side effect on association rules mining.

VI. Summery and Conclusion

In this algorithm instead of focus on itemset and scanning transactions to discover frequent pattern, we concentrate on transactions independently and try to find those itemset that shared between one transaction and others.

References

[1]S. Brin, R. Motwani, J.D. Ullman, S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data", *SIGMOD Record*, Volume 6, Number 2: New York, June 1997, pp. 255 - 264.
[2]Su, Yibin, *Dynamic Itemset Counting and Implication Rules for Market Basket Data: Project Final Report, CS831*, April 2000.
[3] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. Data ining and Knowledge Discovery, 1(3):241– 258, November 1997.
[4] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 209–216. ACM Press, 1997.
[5] H. Mannila. Global and local methods in data mining: basic techniques and open problems. In P. Widmayer, F.T. Ruiz, R. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, Proceedings of the 29th International Colloquium on Automata, Languages and Programming, volume 2380 of Lecture Notes in Computer Science, pages 57–68. Springer, 2002.
[6]H. Toivonen. Sampling large databases for association rules. In T.M. Vijayaraman, A.P. Buchmann, C. Mohan, and N.L. Sarda, editors, Proceedings 22nd International Conference on Very Large Data Bases, pages 134–145. Morgan Kaufmann, 1996.
[7] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining and Knowledge Discovery, 2003.
[8]Adamo,J.,Data mining for association rules and sequential patterns,Springer,Berlin,2001
[9] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 307–328. MIT Press, 1996.
[10] T .Fukuda,Y. Morimoto, S.Moridhita, T. Tokuyama. Mining Optimized Association Rules for Numeric Attributes. ACM PODS Conference Proceedings, pages 182-191, 1996
[11] T. Fukuda, Y. Morimoto, S.Moridhita, T. Tokuyama. Data Mining using Two-dimensional Optimized Association Rules for Numeric Attributes:Schema, Algorithmd , Visualization. ACM SIGMOD Conference Proceedings, pages 13-23, 1996
[12] R. Rastogi, K.Shim. Mining optimized Association rules for categorical and numeric attributes. ICDE Conference Proceedings,pages 503-512,1998
[13] R. Rastogi, K.Shim. Mining optimized Support Rules for numeric attributes. ICDE Conference Proceedings,pages 126-135,1999
[14] R.Srikant, R. Agrwal. Mining Generalized Association Rules. VLDB Conference Proceedings, pages 407-419,1995
[15] R.Srikant, R. Agrwal. Mining Quantitative Association rules in large databases. ACM SIGMOD Conference Proceedings, pages 1-12,1996