INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

EFFICIENT ALGORITHMS FOR MINING HIGH UTILITY ITEMSETS FROM TRANSACTIONAL DATABASES

NITA B. PARMAR, ASST. PROF. S. K. JAISWAL

G.H. Raisoni College of Engineering, Nagpur (M.S.), India.

Accepted Date: 15/03/2016; Published Date: 01/05/2016

Abstract: Existing algorithms for utility mining are column enumeration based, adopt an Apriori-like candidate set generation-and-test approach**,** and thus are inadequate on datasets with high dimensions or long patterns. To solve the problem, this paper proposes a hybrid model and a row enumeration based algorithm, i.e., inter-transaction, to discover high utility itemsets from two directions: existing algorithms such as UMining [1] can be used to seek short high utility itemsets from the bottom, while inter-transaction seeks long high utility itemsets from the top. Experiments on synthetic data show that our method achieves high performance, especially in large high dimensional datasets.

Keywords: Long High Utility Itemset; Hybrid Mode; Utility; Intersection Transaction;

Corresponding Author: MISS. NITA B. PARMAR

Access Online On:

www.ijpret.com

How to Cite This Article:

Nita B. Parmar, IJPRET, 2016; Volume 4 (9): 195-204

PAPER-QR CODE

INTRODUCTION

The main task of traditional association rule mining (ARM) is to identify frequent item sets. It treats all the items equally by assuming that the utility of each item is always 1 (item is present) or 0 (item is absent). For example, in a transact milk which occupy 10% of the total transaction number, contributing 1% of the total profit If the support threshold is 8%, according to traditional algorithms for frequent item set mining, milk will be reported as a frequent item set and birthday cake will be ignored. But in fact, the market professional must be more interested in birthday cake because it contributes a larger portion to total profit than milk. According to Expectancy Theory [2], we have the well-known equation "motivation = probability * utility", which says that motivation is determined by the utility of making a decision and the probability of success. Table 1 is an example of a simplified transaction database where the total utility value is 162. The number in each transaction in table 1 is the sales profit of each item.

Table1: A transaction database

	A	B	C	D	E
T1	0	0	5	0	1
T2	2	3	0	0	0
T3	3	5	15	7	4
T4	0	0	4	7	2
T5	4	5	8	0	0
T6	9	4	0	0	2
T7	6	0	8	3	6
T8	0	0	0	6	3
T9	3	5	0	8	3
T10	3	5	6	1	8

If the support threshold is 3 and the utility threshold is 50, {A,B} is a frequent but not a High utility itemset. On the other hand, {A,B,C} is both a frequent and high utility itemset, {A,B,C,D} is neither a frequent nor a high utility itemset and {A,B,C,D,E} is a high utility but non-frequent

itemset. We can draw a conclusion: downward closure property doesn't apply to utility mining. monotone, succinct, nor convertible [3][4]. Because of this property, most algorithms for frequent pattern mining [5][6][7][8][9][10] can't be used to find high utility itemsets.

Although lots of researches have been Conducted to improve the usefulness of traditional ARM [12][13][14][15], they are all utility-related, not utility-based. To the best of our knowledge, only UMining [1] and Two-phase can be used for utility mining, but both of them are Apriori-like algorithms and are inadequate on datasets with long patterns or high dimensions. To solve the problem, we propose a hybrid top-down/bottom-up search model and a partitioning-based algorithm, inter-transaction, to discover all high utility itemsets from two directions. Under the hybrid model, existing algorithm such as UMining [1] can be used to search the short high utility itemsets by starting from the bottom, while the inter-transaction searches long high utility itemsets by starting from the top, they complement each other.

Inter-transaction is based on the characteristic that there are few common items between or among long transactions, which means that the intersection of multiple long transactions is usually very short. In a high dimensional data environment, the characteristic is especially obvious. This paper emphasizes the introduction of inter-transaction.

2 Inter-Transaction Algorithm

Any high utility itemset must be in a closed itemset (pattern). This means if we can firstly identify all closed itemsets, then mine each closed itemset separately to find all high utility subsets that the closed itemset contains, all high utility itemsets can be identified. Like CARPENTER and TD-CLose [9], inter-transaction is based on row enumeration. Since each closed itemset can be expressed as an intersection transaction [8], mining all intersection transaction has the same power as mining all closed itemsets. In order to avoid the costly process of pattern matching and the complicated data structure such as X-conditional transposed table (which are used in CARPENTER), inter-transaction enumerates every intersection transaction and then computes the current utility values of all subsets that the intersection transaction contains.

2.1. Partition Method

If N is the number of transactions (rows), there will be 2^N combinations of transactions at the worst situation. As the number of transactions grows, the explosive growth of the combination of rows causes the performance of row-enumeration methods decrease dramatically. In a real database, the number of transactions can easily reach to several millions, and enumerating all

the 2^N intersection transactions is not feasible. To solve the problem, the inter-transaction adopts a partition method to divide a database into multiple partitions, with each partition containing a fitting amount of transactions. In the first scan of a database, inter-transaction finds all local long high utility itemsets from every partition, and then these local long high utility itemsets are merged to generate a set of potential long high utility itemsets. In the second scan of the database, the actual utility and support for these itemsets are computed and global high utility itemsets are identified.

2.2. Task Decomposing

If the partition number is n, the size of partitions will be N/n, and the total number of potential intersection transactions becomes $n2^{N/n.}$ When N is too large, enumerating $n2^{N/n}$ intersection transactions is still not feasible. The partition method is insufficient in reducing the search space. On the contrary, short itemsets are relatively dense; Based on the different features, it's reasonable for us to decompose the mining task into two sub problems (discovering long high utility itemsets and short high utility itemsets), so that we can choose proper algorithms to solve the sub problems separately.

2.3 Algorithms

Based on the above discussion, inter-transaction can be described as follows:

Input: A database D, minutil, minlen

Output: All long high utility itemsets.

1) P=partition-database(D) //divide D into

multiple partitions

2) n=number of partitions

3) **for** i=1 to n **do begin**

4) read-in-partition (Pi∈P)

5) H^i=gen-LHU-itemsets(Pi)

6) **end**

7) **for** (i=minlen; $H^j{}_i \neq \varphi$, j=1,2,...n ; i++) **do begin**

8) $C_i^G \approx U\ j\ 1,2,...n\ H_i^J$

9) **end**

10) **for** i=1 to n **do begin**

11) read_in_partition (Pi∈P)

12) **for all** candidates c∈C^G compute utility

c.utility in terms of equation (3), along with

support c.count

13) **end**

14) H^G={ c∈C^G |c.utility≧minutil}

15) Answer= H^G

Notations used in inter-transaction are shown in table 2. The algorithm is very similar to the partition algorithm, but there are two important differences between them. One is that in the partition algorithm, the size of partitions is chosen in terms of the main memory size, such that at least those itemsets and other information that is used for generating candidates can fit in the main memory, whereas inter-transaction seeks balance between keeping a higher local utility threshold and reducing the amount of transactions in a partition. The other difference is that the inter-transaction discovers only local long high utility itemsets via enumerating intersection transactions.

Table 2: Notations used in algorithm

Notation	Meaning
H_k^p	Set of local high utility k-itemsets in partition p
H^p	Set of local high utility itemsets. $H^p = \bigcup_{k>minlen} H_k^p$
C_k^G	Set of global candidate k-itemsets
C^G	Set of global candidate itemsets. $C^G = \bigcup_{k>minlen} C_k^G$
H_k^G	Set of global high utility k-itemsets
H^G	Set of global high utility itemsets. $H^G = \bigcup_{k>minlen} H_k^G$

2.4 Data Layout Alternatives For Inter-Transaction

Conceptually, a database is a two-dimensional matrix and can usually be implemented in four different ways (HIV, HIL, VTV, and VTL) [22][23]. If we express each transaction in horizontal item-vector format (HIV), an intersection transaction can be obtained from the intersection of bit-vectors. For example, if the number of items is 8k, we have to use 1k bytes (8k bits) to express each transaction. In order to perform the intersection of two transactions, 8k bit operations are needed and this is intolerable. Besides the HIV format, we also store each transaction in HIL format. Although this method will waste lots of memory, the cost is affordable because the partition method can save lots of memory. HIV format is used to perform the intersection of bit-vectors, while HIL format is used to store redundant information, which can guide us to choose only necessary bits in a bit-vector to perform bitwise logical (And) operation. Let S1 be the length of transaction T1 in HIL format, S2 be the length of transaction T2, if S1>S2, we choose T2 (the shorter transaction in HIL format) as the benchmark to determine on what bits the bitwise logical operation should be performed: if the k-th bit in T2 is 1, bitwise And operation should be performed on the bit, all other bits should be set 0 in the result of intersection operation. For example, there are 3 transactions, the corresponding data can be seen in table 3:

Table 3: The process of computing T(1,2,3)

	HIV format	HIL format
T_1	1011,1100,1100,0110,001	1,3,4,5,6,9,10,14,15,19
T_2	***1***000,0000,0000,0000,0***11***	1,18,19
T(1,2)	***1***000,0000,0000,0000,00***1***	1,19
T_3	1011,1100,1101,0110,001	1,3,4,5,6,9,10,12,14,15,19
T(1,2,3)	1000,0000,0000,0000,001	1,19

If we want to get T(1,2,3), we can first get T(1,2) by intersecting transactions T1 with T2, then we get T(1,2,3) by intersecting T(1,2) with T3. In order to get T(1,2), we choose the shorter transaction T2 as the benchmark and decide bitwise And operations should be performed only on the first bit, the eighteenth bit and the nineteenth bit (written in bold Italic in table 3). As an intermediate result, we get T(1,2)={1,19}(in HIL format). In the subsequent process of intersecting T(1,2) with T3, we choose T(1,2) as the benchmark and decide that bitwise And operations should be performed only on the first bit and the last bit. We get the final result T(1,2,3)={1,19}. As shown in table 3, only five bit operations are needed for the whole process.

In this way, the amount of bit operations linearly depends only on the length of the short transaction in HIL format. The experiment shows this method outperforms VIPER and DIFFSET. \

Experimental Results

All the experiments were performed on a 2GHz Legend server with 4GB of memory, running windows 2003. The program was coded in Delphi 7. Seven datasets were used in our experiments; all were generated by IBM quest data generator [24]. Six of them are 40.I30.D8000K with 0.5k, 1k, 2k, 4k, 8k and 16K items respectively, the seventh is T20.I6.D8000K with 4k items, where T# stands for the average length of transactions, I# for the average length of maximal potentially large itemsets and D# for the number of transactions. Because the generator only generates the quantity of 0 or 1 for each item in a transaction, we use Delphi function "RandG" to generate random numbers with Gaussian distribution, which mimic the quantity sold of an item in each transaction. The unit profit of each item is defined as item ID%100, where % is a modulus operator.

Figure 1 presents the scalability of inter-transaction by increasing the number of transactions from 0.25M to 8M. Experimental results show that our algorithm scales linearly with the number of transactions. We also modified our program to find long frequent itemsets, and a similar trend is obtained just as shown in figure 1

Figure 2 shows the performance when varying the number of items. Different from other algorithms, the performance of inter-transaction improves as the number of items increases. The reason is that the number of items has a direct relationship with the sparseness of a dataset. The more items there are, the sparser the dataset, and the shorter the intersection of two transactions. That means inter-transaction can enumerate all long intersection transactions easily in a sparse dataset. From figure 2 we can observe that Inter-transaction is suitable for those datasets with more than 1k items.

In figure 3, minlen is the minimum length of itemsets that the inter-transaction can discover within a reasonable time. It actually decides the task assigned to inter-transaction. Figure 3 shows that minlen decreases as the number of items increases, which means inter-transaction can complete more mining tasks in a sparse database. To test the total performance of the hybrid model, we choose Two-phase to mine short high utility itemsets and then compare the performance of the hybrid method (here hybrid method means inter-transaction + Two-phase) and Two-phase.Experiments were performed on T20I6D8000K and T40I30D8000K, minlen is set 3 for T20I6D8000K and 5 for T40I30D8000K, the number of items is 4k. Corresponding performance curves are illustrated in figures 4 and 5.

From the two figures we can observe the hybrid model is not suited for dataset with only short patterns, this is because inter-transaction can't take obvious effect on these datasets. Two-

phase has to extend short itemsets step by step to obtain long itemsets, while inter-transaction obtains long itemsets directly by intersecting relevant transactions.

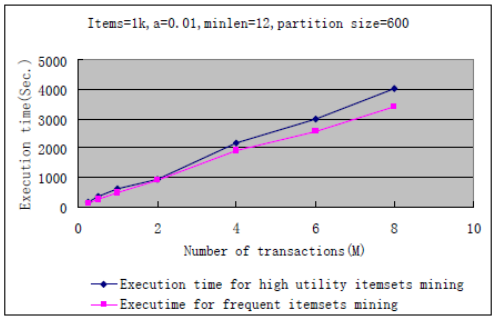

Fig.1. Scalability with the number of transactions

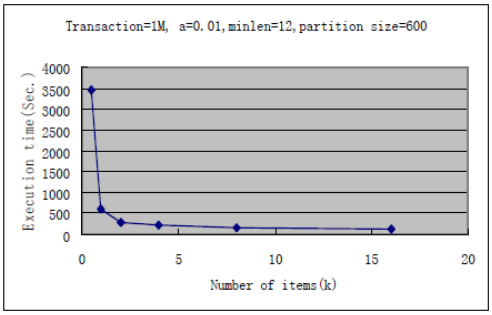

Fig.2. Scalability with the number of items

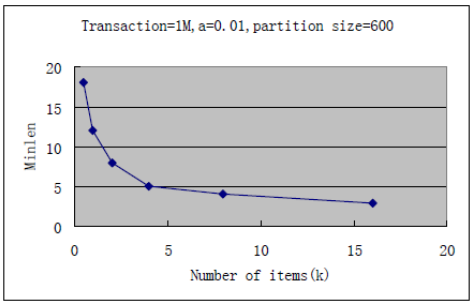

Fig.3. The effect of the number of items on minlen

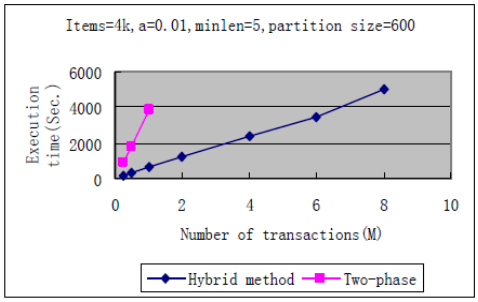

Fig.4. The execution time for T40I30D8000

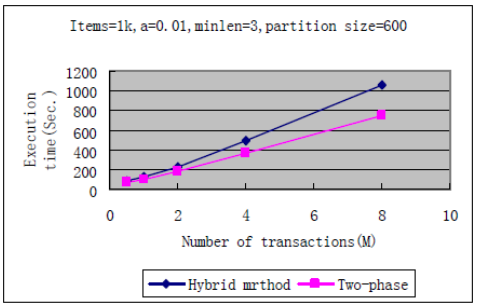

Fig. 5. The execution time for T20I6D8000

CONCLUSIONS

The paper proposes a hybrid model to discover high utility itemsets from two directions. The intention of the hybrid model is to decompose a complex problem into two easy sub- problems, then use proper methods to solve them separately. Inter-transaction integrates the advantages of the partition algorithm and row enumeration algorithms. In the meantime, its performance is affected by the characteristics of the database, including data skew and the number of items. Parameters such as minimum length threshold minlen and the size of partitions also affect its performance.

REFERENCES

1. Yao H. and Hamilton, H.J., Mining itemset utilities from transaction databases, Data & Knowledge Engineering, 59 , 2006, pp. 603 –626.
2. V. H. Vroom, Work and Motivation, John Wiley, 1964.
3. Shen Y. D., Zhang Z. and Yang Q, Objective-oriented utility-based association mining, Proceedings of the 2002 IEEE International Conference on Data Mining, 2002, pp. 426-433.

4. Yao H. and Hamilton H.J., A Unified Framework for Utility Based Measures for Mining Itemsets, Proceedings of the 2006 International Workshop on Utility-Based Data Mining, Philadelphia, PA, 2006, pp. 28-37.
5. Ramesh C. Agarwal, Charu C. Aggarwal and V.V.V. Prasad, A tree projection algorithm for generation of frequent itemsets, Journal of Parallel and Distributed Computing, 61(3), 2000, pp. 350-371.
6. Feng P., Gao. C.and et al.. CARPENTER: Finding closed patterns in long biological database. Proc. of SIGKDD, Washington, 2003, pp. 413-419.
7. Hongyan Liu, Jiawei Han and et al. Mining frequent Patterns from Very High Dimensional Data: A Top-down Row Enumeration Approach. Proceedings of the Sixth SIAM International Conference on Data Mining, Bethesda, Maryland, 2006, pp. 20-22.
8. K. Wang, S. Zhou, J. Han, and Profit mining: from patterns to action, Proceedings of International Conference on Extending Database Technology, 2002, pp. 70-87.
9. Lin TY, Yao YY, Louie E. Mining Value Added Association rules, Proceedings of PAKDD, 2002, pp. 328-333.
10. Aumann Y., Lindell Y., A Statistical Theory for Quantitative Association Rules, Journal of Intelligent Information Systems, 20, 2003, pp. 255–283.
11. Geoffrey I. Webb, Discovering associations with numeric variables, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001, pp. 383-388.
12. C.H. Cai, Ada W.C. Fu, C.H. Cheng and W.W. Kong, Mining Association Rules with Weighted Items, Proceedings of the International Database Engineering and Applications Symposium, 1998, pp. 68-77.