International Journal of Technology Research and Management
ISSN (Online): 2348-9006
Vol 5 Issue 3 March 2018

An Item Elimination Based Technique for Mining High Utility Items from a Data Set

Heena Ansari[1], Prof. Abhishek Raghuwanshi[2]
Department of Information Technology, Mahakal Institute Of Technology, Ujjain (MP)[1]
HOD IT Dept., Department of Information Technology, Mahakal Institute Of Technology, Ujjain (MP) [2]
heenakhanalex@gmail.com[1]

Abstract: *The data mining and their different applications are becomes more popular now in these days a number of large and small scale applications are developed with the help of data mining techniques i.e. predictors, regulators, weather forecasting systems and business intelligence. There are two kinds of model are available for namely supervised and unsupervised. The performance and accuracy of the supervised data mining techniques are higher as compared to unsupervised techniques therefore in sensitive applications the supervised techniques are used for prediction and classification. This paper presents a high utility item set mining technique. In this technique, the useless patterns are removed at the initial stage of mining. So it is helping in getting less time consumption.*

1. INTRODUCTION

In utility mining [3,4] we concentrate on utility value of itemset while in frequent item set mining we concentrate that how frequently items appears in transactional database.

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into needful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of the analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases [2].

Data: Data are any facts, numbers, or text that can be processed by a computer. Today, organizations are accumulating vast and growing amounts of data in different formats and different databases. This includes:

- Operational or transactional data such as, sales, cost, inventory, payroll, and accounting
- Non-operational data, such as industry sales, forecast data, and macro-economic data
- Meta data - data about the data itself, such as logical database design or data dictionary definitions

Information: The patterns, associations, or relationships among all this data can provide information. For example, analysis of retail point of sale transaction data can yield information on which products are selling and when [1].

Knowledge: Information can be converted into knowledge about historical patterns and 5future trends. For example, summary information on retail supermarket sales can be analyzed in light of promotional efforts to provide knowledge of consumer buying behavior. Thus, a manufacturer or retailer could determine which items are most susceptible to promotional efforts.

Data Warehouses: Dramatic advances in data capture, processing power, data transmission, and storage capabilities are enabling organizations to integrate their various databases into data warehouses. Data warehousing is defined as a process of centralized data management and retrieval. Data warehousing, like data mining, is a relatively new term although the concept itself has been around for years. Data warehousing represents an ideal vision of maintaining a central repository of all organizational data. Centralization of data is needed to maximize user access and analysis. Dramatic technological advances are making this vision a reality for many companies. And, equally dramatic advances in data analysis software are allowing users to access this

data freely. The data analysis software is what supports data mining [2].

Some methods were proposed for mining high utility item or itemsets from the databases, such as UMining [9], Two-Phase [7,8], IIDS [6] and IHUP [5]. UMining algorithm [9] proposed by Yao et al. used an estimation method to prune candidate itemset in memory. Also it is shown to have good performance but it cannot capture the complete set of high utility itemsets since some high utility patterns may be pruned during the process.

2. BASIC CONCEPTS

The basic definitions are as follows:

Definition 1: A frequent itemset is a set of items that appears at least in a pre-specified number of transactions. Formally, let I = {i1, i2, . . . , im} be a set of items and DB = {T1, T2, ..., Tn} a set of transactions where every transaction is also a set of items (i.e. itemset).

Definition 2. The utility of an item ip is a numerical value yp defined by the user. It is transaction independent and reflects importance (usually profit) of the item. External utilities are stored in an utility table.

Definition 3: The utility of an item set X in a transaction Ti is denoted by U(X,Ti) & it is calculated as follows. For example,U({*AC*}, *T1*) = U({*A*}, *T1*) + U({*C*}, *T1*) = 5 + 1 = 6.

Definition 4: The utility of an item set X in D is denoted by U(X) & it is calculated as follows For example, U({*AD*}) = U({*AD*}, *T1*) + U({*AD*}, *T3*) = 7 + 17 = 24.

Definition 5: An itemset is called a *high utility itemset* if its utility is no less than a user-specified *minimum utility threshold* which is denoted as *min_util*. Otherwise, it is called a *low utility itemset.*

Table 1:Transaction Data Set

TID	TRANSACTION	TU
T1	(A,1) (C,1) (D,1)	8
T2	(A,2) (C,6) (E,2) (G,5)	27
T3	(A,1) (B,2) (C,1) (D,6) (E,1) (F,5)	30
T4	(B,4) (C,3) (D,3) (E,1)	20
T5	(B,2) (C,2) (E,1) (G,2)	11

Table 2: Item & correspondent profit

ITEM	A	B	C	D	E	F	G
PROFIT	5	2	1	2	3	1	1

Definition 5. The transaction utility of a transaction *Td* is denoted as *TU*(*T*d) and defined as *u*(*Td, Td*). For example, *TU*(*T1*) = *u*({*ACD*}, *T1*) = 8.

3. LITERATURE SURVEY

Some methods were proposed for mining high utility item or itemsets from the databases, such as UMining [9], Two-Phase [7], IIDS [6] and IHUP [2]. UMining algorithm [9] proposed by Yao et al. used an estimation method to prune candidate itemset in memory. Also it is shown to have good performance but it cannot capture the complete set of high utility itemsets since some high utility patterns may be pruned during the process.

Although IHUP finds HTWUIs without generating any candidates for high transactional weighted utility item sets and achieves a better performance than IIDS and Two-Phase, it still produces too many high transactional weighted utility item sets in phase I. however IHUP and Two-Phase produce the same number of high transactional weighted utility item sets in phase first since they use transaction- weighted utilization mining model [7] to overestimate the utilities of the itemsets. It is seems that ,this model may overestimate too many low utility itemsets as HTWUIs and produce too many candidate itemsets in phase first . Such a large number of high transactional weighted utility item sets HTWUIs degrades the mining performance in phase first. In terms of execution time and memory consumption. Besides, the number of HTWUIs in phase first also affects the performance of the algorithms in phase second since the more HTWUIs are generated in phase first. The more execution time is required for identifying high utility itemsets in phase second.

As stated above, the number of HTWUIs generated in phase first forms a crucial problem to the performance of algorithms. In view of this, we propose four strategies to reduce the estimated utility values of the item and itemsets. By applying the proposed strategies, the number of candidates generated in phase first can be reduced effectively and the high utility itemsets can be identified more efficiently since the number of itemsets needed to be checked in phase second is highly reduced in phase first.

Both Charm and Closet [8,9] inherit the same data structures and computing framework of their big brothers

dEclat and FP-Growth respectively. They implement Algorithm 4, but they differ in the way the closed frequent itemsets are stored in order to exploit the Sub-sumption Lemma. Charm adopts a hash table, were the hash function is the sum of the transactions ids supporting an itemset. Closet uses a trie- like structure, indexed by a two-level hash. The first level is based on the last item of the itemset to be checked and the second on its support. FP-Close [12] is inspired to Closet, thus using the same divide et impera approach and same FP-tree data structure. What makes FP-Close different from other CFIM algorithms is the application of the projecting approach to the historical collection of closed frequent itemsets. Not only a small dataset is associated to each node of the tree, but also a pruned subset of the closed itemsets mined so far is forged and used for duplicate detection. Indeed, this technique is called progressive focusing and it was introduced by [10] for mining maximal frequent itemsets. Together with other optimizations, this truly provides dramatic speed-up, making FP-Close order of magnitudes faster than Charm and Closet, and also making it worth to be celebrated as the fastest algorithm at the FIMI workshop 2003 [11].

Min Utility	UMining	2 Phase	HTWUI	IHUP
30	0.56	0.54	0.49	0.47
40	0.50	0.50	0.44	0.44
50	0.49	0.45	0.42	0.41
60	0.48	0.44	0.40	0.40
70	0.42	0.40	0.39	0.37

Chi et al. [13] propose an algorithm called Moment for mining frequent closed itemsets over data streams. It uses a CET Tree (Closed Enumerate Tree) to maintain the main information of itemsets. Each node in CET Tree represents an itemset with different node type. Some nodes in CET Tree are not closed so that there are still some redundant nodes in CET Tree. Moment must maintain huge CET nodes for a frequent closed itemset. Chi et al. indicated that the ratio of CET nodes for a closed itemsets is about 20:1. If there are a large number of frequent closed itemsets, it will consume a lot of memory space. When a new transaction arrives, the node is inserted and updated according to its node type. The exploration of frequent itemsets and node type checking are time consuming. CFI-Stream is another algorithm for this problem [14]. Only the closed itemsets are maintained in a lexicographical ordered tree which is called DIU Tree (DIrect Update Tree). Each node consists of a closed itemset and its support count. When a new transaction X arrives, CFI-Stream will generate all the subsets of X, and check if each subset Y is closed or not after the transaction arrives. To check whether an itemset Y is closed or not, CFI-Stream may need to search all supersets of Y from DIU Tree. It takes a lot of time to generate all the subsets of a new transaction and search their supersets from DIU Tree.

4. PROPOSED METHODOLOGY

Step 1: Input:

- A Transaction data Base T
- Minimum utility value

Step 2: Scan the transaction data base and calculate the weighted transaction utility of each item. Only those item are included in the initial high utility item set mining list whose weighted transaction utility is more than the minimum utility.

Step 3: In this step, we eliminate all those items from the transaction data base T, whose utility is less than the minimum utility. Then transaction data base T will be transformed into a compressed data base T1. Now this T1 will be used in finding the high utility item sets of greater size.

Step 4: From candidate of size 1, we recursively create candidates of greater size as follows:

1. From candidate of size 1, we recursively create candidates of greater size as follows:
   - For each itemset I1 and I2 of level k-1
   - we compare items of itemset1 and itemset2. If they have all the same k-1 items and the last item of itemset1 is smaller thanthe last item of itemset2, we will combine them to generate a candidate
   - Calculate weighted transaction utility of itemset using the compressed data set T1
   - if the weighted transaction utility is high enough
   - add it to the set of HUI (High utility items sets)

- Continue this process until there are candidates to combine

Step 5: Return all high utility itemsets found

Step 6: End of process.

5. CONCLUSION

The data capturing technologies is also increasing. In utility mining we concentrate on utility value of itemset while in frequent item set mining we concentrate that how frequently items appears in transactional database. In this paper, we surveyed the list of existing high utility mining techniques. However we surveyed different concepts of Association rule mining and frequent itemset mining techniques which play significant role for basic of utility itemset mining but we restricted ourselves to the classic high utility mining problem. This paper has proposed a time efficient algorithm for mining high utility item sets from a transaction data set.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. of the 20th Int'l Conf. on Very Large Data Bases, pp. 487-499, 1994.
[2] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. Efficient tree structures for high utility pattern mining in incremental databases. In IEEE Transactions on Knowledge and Data Engineering, Vol. 21, Issue 12, pp. 1708- 1721, 2009.
[3] R. Chan, Q. Yang, and Y. Shen. Mining high utility itemsets. In Proc. of Third IEEE Int'l Conf. on Data Mining, pp. 19-26, Nov., 2003.
[4] A. Erwin, R. P. Gopalan, and N. R. Achuthan. Efficient mining of high utility itemsets from large datasets. In Proc. of PAKDD 2008, LNAI 5012, pp. 554-561.
[5] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.
[6] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. In Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, Jan., 2008.
[7] Y. Liu, W. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In Proc. of the Utility-Based Data Mining Workshop, 2005.
[8] Piatetsky-Shapiro, G. (1991), Discovery, analysis, and presentation of strong rules, in G. Piatetsky-Shapiro & W. J. Frawley, eds, „Knowledge Discovery in Databases", AAAI/MIT Press, Cambridge.
[9] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In DMKD '00: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pages 21–30, May 2000.
[10] M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In SDM '02: Proceedings of the second SIAM International Conference on Data Mining, April 2002.
[11] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In FIMI '03: Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, November 2003.
[12] G. Grahne and J. Zhu. Fast algorithms for frequent itemset mining using fptrees. IEEE Transactions on Knowledge and Data Engineering, 17(10):1347 1362, 2005.
[13] Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window. In: Proceedings of 2004 IEEE International Conference on Data Mining, Brighton, pp. 59–66 (2004)
[14] Jiang, N., Gruenwald, L.: CFI-Stream: Mining Closed Frequent Itemsets in Data Streams. In: Proceedings of 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, pp. 592–597 (2006)