An Improved Approach to Discover High Utility Item Set from Large Data Set

Shilpa Shrivastabva[1], Mr. Abhishek Tiwari[2]

[1]Research Scholar, Information Technology Department, MIT, Ujjain, MP India
[2]Reader, Information Technology Department, MIT, Ujjain, MP, India

Abstract— *The term data mining is often used to apply to the two separate processes of knowledge discovery and prediction. Knowledge discovery provides explicit information that has a readable form and can be understood by a user. Forecasting, or predictive modeling provides predictions of future events and may be transparent and readable in some approaches (e.g. rule based systems) and opaque in others such as neural networks. Moreover, some data mining systems such as neural networks are inherently geared towards prediction and pattern recognition, rather than knowledge discovery. Utility item set mining is addition to the frequent pattern mining. The goal of high utility item set mining is to find all item sets that give utility greater or equal to the user specified threshold. The deficiency of this approach is that it does not consider the statistical aspect of item sets. Utility-based measures should incorporate user-defined utility as well as raw statistical aspects of data. Consequently, it is meaningful to define a specialized form of high utility item sets, utility-frequent item sets which are a subset of high utility item sets as well as frequent item sets. In this paper we proposed an efficient approach to mine high utility items form transactional records.*

Keywords— ***Utility, Candidates, Transactions, Thresholds, Item set.***

I. INTRODUCTION

Mining high utility item sets is upgrades the standard frequent item set mining framework as it employs subjectively defined utility instead of statistics-based support measure. User-defined utility is based on information not available in the transaction dataset. It often reflects user preference and can be represented by an external utility table or utility function. Utility table (or function) defines utilities of all items in a given database (we can also treat them as weights). Besides subjective external utility we also need transaction dependent internal utilities (e.g. quantities of items in transactions). Utility function we use to compute utility of an item set takes into account both internal and external utility of all items in a item set. The most usual form that is also used in products of internal and external utilities of present items.

.

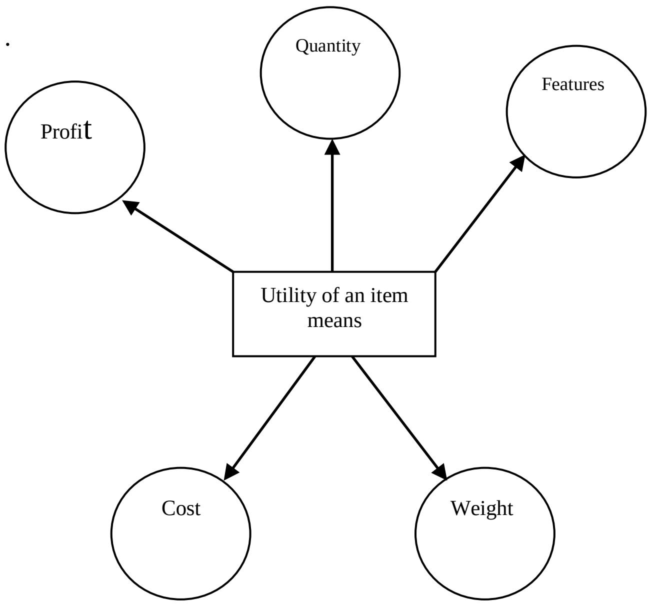


Fig.1.1: Utility means many things

II. ARCHITECTURE

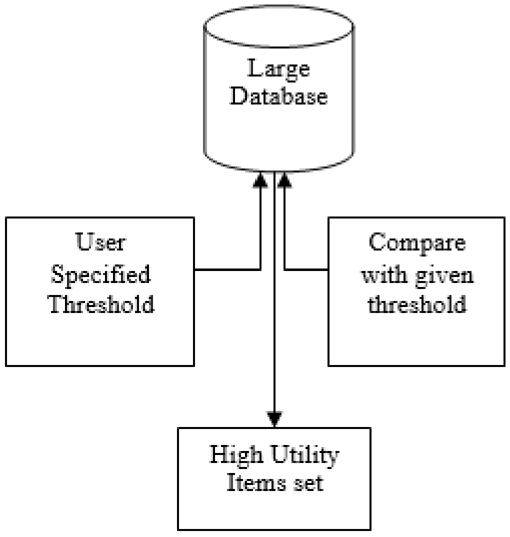


Fig.2: Architecture

User gives a minimum threshold value. We calculate total utility value entire database and compare with given threshold value. The item set which satisfy the given condition known as high utility.

III. BACKGROUND

Consider a simple example of transactional database

Table.1: Transactional data set

TID & ITEM	I1	I2	I3	I4	I5
T1	0	0	18	0	1
T2	0	6	0	1	1
T3	2	0	1	0	1
T4	1	0	0	1	1
T5	0	0	4	0	2
T6	1	1	0	0	0
T7	0	10	0	1	1
T8	3	0	25	3	1
T9	1	1	0	0	0
T10	0	6	2	0	2

The utility table, the right column displays the profit of each item per unit in dollars

Table.2: Profit table

ITEM	*PROFIT($)(Per Unit)*
I1	*3*
I2	*10*
I3	*1*
I4	*6*
I5	*5*

External Utility: - The external utility of an item i_p is a numerical value y_p defined by the user. It is transaction independent and reflects importance (usually profit) of the item. External utilities are stored in a utility table. For example, external utility of item I2 in Table 2 is 10.

Internal Utility:-The internal utility of an item i_p is a numerical value x_p which is transaction dependent. In most cases it is defined as the quantity of an item in transaction. For example, internal utility of item I5 in transaction T5 is 2 in table 1.

The utility of item: - The utility of item i_p in transaction T is the quantitative measure computed with utility function from above definition $u(i_p, T) = f(x_p, y_p)$, $i_p \in T$. For example: utility of item I5 in transaction T5 is $2 * 5 = 10$.

The utility of item set S in transaction T: - The utility of item set S in transaction T is defined as

$$u(S,T) = \sum_{i_p \in S} u(i_p, T),\ S \subseteq T$$

For example utility of itemset {I2, I5} in transaction T2 is
u({I2,I5} , T2) = u({I2} , T2) + u({I5} , T2) = 6 ∗ 10 + 1 ∗ 5 = 65.

The utility of item i_p in item set S: - The utility of item ip in item set S is defined as

$$u(i_p, S) = \sum_{T \in DB, S \subseteq T} u(i_p, T).$$

For example, utility of item E in item set {I2, I5} is u(I5, {I2,I5}) = u(I5, T2) + u(I5, T7) + u(I5, T10) = 20.

The utility of item set S in database DB: - The utility of item set S in database DB is defined as

$$u(S) = \sum_{T \in DB, S \subseteq T} u(S,T) = \sum_{T \in DB, S \subseteq T} \sum_{i_p \in S} f(x_p, y_p)$$

For example, utility of item set {I1,I5} in database from Table 1 is
u({I1,I5}) = u({I1,I5} , T3) + u({I1,I5} , T4) + u({I1,I5E} , T8) = 33.

The utility of transaction T:- The utility of transaction T is defined as

$$u(T) = \sum_{i_p \in T} u(i_p, T).$$

For example: utility of transaction T10 is
u(T10) = u({I2} , T10)+u({I3} , T10)+ u({I3} , T10) = 72.

The utility of database DB :- The utility of database DB is defined as

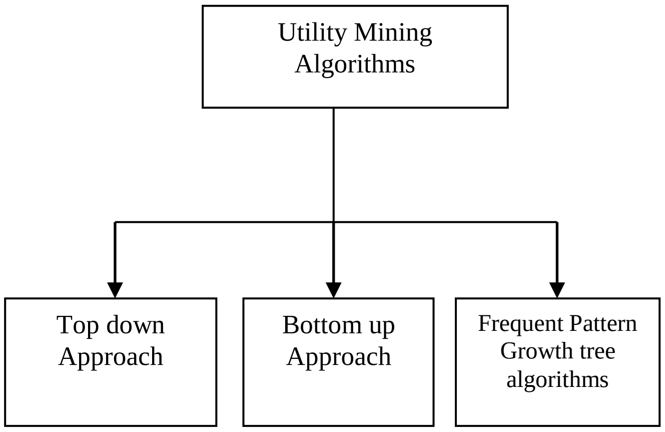
$$u(DB) = \sum_{T \in DB} u(T).$$

For example, utility of database DB from table 1 and table 2 is
u(DB) = u(T1) + . . . +u(T10) = 23 + . . . + 72 = 400.

The utility share of item set S in database:- The utility share of item set S in database DB is
For example, utility share of item set {I1,I4,I5} in database from Table 1 is U({I1,I4,I5}) = 46/400 = 0.115 = 11.5%.

IV. CATEGORIES OF UTILITY MINING ALGORITHMS

Utility mining algorithm is mainly classified into three categories. First categories include top down approach, second categories include bottom up approach and third approach based on Frequent Pattern Growth tree. Top down approach based algorithm are those algorithm which is based on up word words closer properties mean generate candidates set and then use pruning strategy to remove useless candidates. In seconds categories algorithms us utility pattern growth tree based concepts for generating use full pattern . These algorithms use tree like structure which contain main root node and other sub tree.


Utility Mining Algorithms
Top down Approach
Bottom up Approach
Frequent Pattern Growth tree algorithms

V. LITERATURE REVIEW

In (1994) Agarwal proposed the mining of association rules for finding the relationships between data items in large databases.
In 2003 Chan observes that the candidate set pruning strategy exploring the ant monotone property used in classical algorithm does not hold for utility mining.
In 2004 Yao defines the problem of utility mining formally. The work defines the terms transaction utility and external utility of an itemset. The mathematical model of utility mining was then defined based on the two properties of utility bound and support bound.
In 2006, 2007 Yao defines the utility mining problem as one of the cases of constraint mining. This work shows that the downward closure property used in the standard Classical algorithm and the convertible constraint property are not directly applicable to the utility mining problem.
In 2008 Li proposed two efficient one pass algorithms MHUI-BIT and MHUI-TID for mining high utility item sets from data streams within a transaction sensitive sliding window. Liu et al in proposes a Two-phase algorithm for finding high utility item sets.
In 2009 Shankar presents a novel algorithm Fast Utility Mining (FUM) which finds all high utility item sets within the given utility constraint threshold.
In 2010 Vincent S. Tseng, et. al. Proposed a data structure, named UP-Tree, and then describe a new algorithm, called UP-Growth, The framework of the UP-Growth.
In 2012 Mengchi Liu et al. proposed “Mining High Utility Item sets without Candidate Generation”].
In 2013 Arumugam P et al. proposed “Advance Mining Of High Utility Item sets In Transactional Data”.
In 2014 More Rani N. and Anbhule Reshma V “Mining High Utility Item sets From Transaction Database”[13,14].

VI. PROBLEM STATEMENT

In utility mining process very large number of candidates is generated. These useless candidates make execution process slow because we have to prune these items and consider only that item which satisfies the threshold value. So improving pruning strategies is a difficult task.
In previous algorithms the function used for calculating utility is also in efficient because some algorithm are based on expected utility mining model and some are based on transaction weighted utility model. Improving accuracy is also a challenge.

VII. PROPOSED ALGORITHM

We proposed an efficient method which combined reducing the cost of database scans by transaction merging and pruning search space by using utility and local utility.
Calculate Transaction-weighted utility value for one item by using *twu(X)* formula.
Generate high utility one item set by comparing Transaction-weighted utility value of each one item set with give utility threshold value ε .
Update the transaction weighted utilization table by subtracting the utility value of deleted one item set.
While (|C_k| > 0 and k =K) (more candidate)
Generate candidate set for next level and Transaction-weighted utility value for item set using *twu(X)* formula.
Generate high utility one item set by comparing Transaction-weighted utility value of each one item set with give utility threshold value ε.
Generate all high utility item set .

VIII. EXPERIMENTAL ANALYSIS

We evaluate the performance of proposed algorithm and compare it with iFUM and TP (Two Phase) algorithms. The experiments were performed on i3 processor (2.5GHz Intel Processor with 4M cache memory), 2GB main memory and 400 GB secondary memory , and running on Windows XP. The algorithms are implemented in using C# Dot Net Framework language version 4.0.1. Both synthetic datasets are used to evaluate the performance of the algorithms.

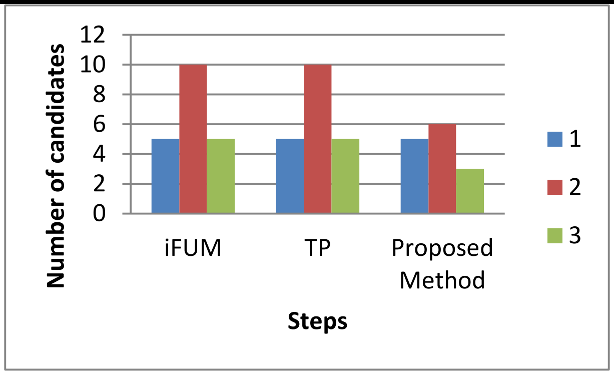


Fig.2: Comparison graph

IX. CONCLUSION

Mining Expected Utility Two Phase and several other algorithms have mine high utility item set very efficiently. But there is need to enhance this algorithm so that it can be applied to large sized dataset. The complexity factor for frequent pattern mining algorithm includes several factors like Execution time and I/O cost. The proposed method reduce candidates generations at different stages

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 item sets. In Proc. of Third IEEE Int'l Conf. on Data Mining, pp. 19-26, Nov., 2003.
[4] Y. L. Cheung, A. W. Fu, Mining frequent item sets without support threshold: with and without item constraints. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 6, pp. 1052-1069, 2004.
[5] K. Chuang, J. Huang, M. Chen, Mining Top-K Frequent Patterns in the Presence of the Memory Constraint, The VLDB Journal, Vol. 17, pp. 1321-1344, 2008.
[6] A. Erwin, R. P. Gopalan and N. R. Achuthan. Efficient Mining of High-utility Item sets from Large Datasets. In PAKDD 2008, LNAI 5012, pp. 554-561, 2008.
[7] A. W. Fu, R. W. Kwong and J. Tang, Mining N-Most Interesting Item sets, In Proc. of ISMIS'00, 2000.
[8] 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.
[9] J. Han, J. Wang, Y. Lu and P. Tzvetkov, "Mining Top-k Frequent Closed Patterns without Minimum Support," In Proc. of ICDM, 2002.
[10] Y. Hirate, E. Iwahashi and H. Yamana, TF2P-Growth: An Efficient Algorithm for Mining Frequent patterns without any Thresholds, In Proc. of ICDM 2004.
[11] H.-F. Li, H.-Y. Huang, Y.-C. Chen, Y.-J. Liu, S.-Y. Lee. Fast and Memory Efficient Mining of High Utility Item sets in Data Streams. In Proc. of the 8th IEEE Int'l Conf. on Data Mining, pp. 881-886, 2008.
[12] Y. Liu, W. Liao, and A. Choudhary. A fast high-utility item sets mining algorithm. In Proc. of the Utility-Based Data Mining Workshop, 2005.
[13] [13] Y.-C. Li, J.-S. Yeh and C.-C. Chang. Isolated Items Discarding Strategy for Discovering High-utility Item sets. In Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, 2008.
[14] S. Ngan, T. Lam, R. C. Wong and A. W. Fu, Mining N-most Interesting Item sets without Support Threshold by the COFI-Tree, Int. J. Business Intelligence & Data Mining, Vol. 1, No. 1, pp. 88-106, 2005.
[15] J. Pisharath, Y. Liu, B. Ozisikyilmaz, R. Narayanan, W. K. Liao, A. Choudhary and G. Memik, NU-Mine Bench version 2.0 dataset and technical report, http://cucis.ece.northwestern.edu/projects/DMS/MineBench.html.
[16] T. M. Quang, S. Oyanagi, and K. Yamazaki, Ex Miner: An Efficient Algorithm for Mining Top-K Frequent Patterns, ADMA 2006, LNAI 4093, pp. 436 – 447, 2006.