A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets

Ying Liu, Wei-keng Liao, and Alok Choudhary

Electrical and Computer Engineering Department,
Northwestern University, Evanston, IL, USA 60208
{yingliu, wkliao, choudhar}@ece.northwestern.edu

Abstract. Traditional association rules mining cannot meet the demands arising from some real applications. By considering the different values of individual items as utilities, utility mining focuses on identifying the itemsets with high utilities. In this paper, we present a Two-Phase algorithm to efficiently prune down the number of candidates and precisely obtain the complete set of high utility itemsets. It performs very efficiently in terms of speed and memory cost both on synthetic and real databases, even on large databases that are difficult for existing algorithms to handle.

1 Introduction

Traditional Association rules mining (ARM) [1] model treat all the items in the database equally by only considering if an item is present in a transaction or not. Frequent itemsets identified by ARM may only contribute a small portion of the overall profit, whereas non-frequent itemsets may contribute a large portion of the profit. In reality, a retail business may be interested in identifying its most valuable customers (customers who contribute a major fraction of the profits to the company). These are the customers, who may buy full priced items, high margin items, or gourmet items, which may be absent from a large number of transactions because most customers do not buy these items. In a traditional frequency oriented ARM, these transactions representing highly profitable customers may be left out. Utility mining is likely to be useful in a wide range of practical applications.

Recently, a *utility mining* model was defined [2]. Utility is a measure of how "useful" an itemset is. The goal of utility mining is to identify high utility itemsets that drive a large portion of the total utility. Traditional ARM problem is a special case of utility mining, where the utility of each item is always 1 and the sales quantity is either 0 or 1.

There is no efficient strategy to find all the high utility itemsets due to the nonexistence of "downward closure property" (anti-monotone property) in the *utility mining* model. A heuristics [2] is used to predict whether an itemset should be added to the candidate set. We refer this algorithm as MEU (Mining using Expected Utility) for the rest of this paper. However, the prediction usually overestimates, especially at the beginning stages, where the number of candidates approaches the number of all the combinations of items. Such requirements can easily overwhelm the available

T.B. Ho, D. Cheung, and H. Liu (Eds.): PAKDD 2005, LNAI 3518, pp. 689 – 695, 2005.

Table 1. A transaction database

(a) Transaction table. Each row is a transaction. The columns represent the number of items in a particular transaction. TID is the transaction identification number

TID \ ITEM	A	B	C	D	E
T_1	0	0	18	0	1
T_2	0	6	0	1	1
T_3	2	0	1	0	1
T_4	1	0	0	1	1
T_5	0	0	4	0	2
T_6	1	1	0	0	0
T_7	0	10	0	1	1
T_8	3	0	25	3	1
T_9	1	1	0	0	0
T_{10}	0	6	2	0	2

(b) The utility table. The right column displays the profit of each item per unit in dollars

ITEM	PROFIT ($) (per unit)
A	3
B	10
C	1
D	6
E	5

(c) Transaction utility (TU) of the transaction database

TID	TU	TID	TU
T_1	23	T_6	13
T_2	71	T_7	111
T_3	12	T_8	57
T_4	14	T_9	13
T_5	14	T_{10}	72

memory space and computation power of most of the machines. In addition, MEU may miss some high utility itements when the variance of the itemset supports is large.

The challenge of utility mining is in restricting the size of the candidate set and simplifying the computation for calculating the utility. In order to tackle this challenge, we propose a Two-Phase algorithm to efficiently mine high utility itemsets. A performance study has been conducted on real and synthetic data, obtaining significant improvement in terms of speed and accuracy over the best existing algorithm [2]. Our algorithm easily handles very large databases that existing algorithms cannot handle.

The rest of this paper is organized as follows. Section 2 overviews the related work. In Section 3, we propose the Two-Phase algorithm. Section 4 presents our experimental results and we summarize our work in section 5.

2 Related Work

Researches that assign different weights to items have been proposed in [3, 4, 5, 6]. These weighted ARM models are special cases of utility mining.

A concept, *itemset share*, is proposed in [7]. It can be regarded as a utility because it reflects the impact of the sales quantities of items on the cost or profit of an itemset. Several heuristics have been proposed.

A utility mining algorithm is proposed in [8], where the concept of “useful” is defined as an itemset that supports a specific objective that people want to achieve. The definition of utility and the goal of his algorithm are different from those in our work.

3 Two-Phase Algorithm

We start with the definition of a set of terms that leads to the formal definition of utility mining problem. The same terms are given in [2].

- $I = \{i_1, i_2, \ldots, i_m\}$ is a set of items.
- $D = \{T_1, T_2, \ldots, T_n\}$ be a transaction database where each transaction $T_i \in D$ is a subset of I.
- $o(i_p, T_q)$, *local transaction utility value*, represents the quantity of item i_p in transaction T_q. For example, $o(A, T_8) = 3$, in Table 1(a).
- $s(i_p)$, *external utility,* is the value associated with item i_p in the Utility Table. This value reflects the importance of an item, which is independent of transactions. For example, in Table 1(b), the external utility of item A, s(A), is 3.
- $u(i_p, T_q)$, *utility*, the quantitative measure of utility for item i_p in transaction T_q, is defined as $o(i_p,T_q) \times s(i_p)$. For example, $u(A, T_8) = 3 \times 3 = 9$, in Table 1
- $u(X, T_q)$, *utility of an itemset X in transaction* T_q, is defined as $\sum_{i_p \in X} u(i_p,T_q)$, where $X = \{i_1, i_2, \ldots, i_k\}$ is a k-itemset, $X \subseteq T_q$ and $1 \leq k \leq m$.
- $u(X)$, *utility of an itemset* X, is defined as $\sum_{T_q \in D \wedge X \subseteq T_q} u(X,T_q)$. (3.1)

Utility mining is to find all the itemsets whose utility values are beyond a user specified threshold. An itemset X is a *high utility itemset* if $u(X) \geq \varepsilon$, where $X \subseteq I$ and ε is the minimum utility threshold, otherwise, it is a *low utility itemset*. For example, in Table 1, $u(\{A, D, E\}) = u(\{A, D, E\}, T_4) + u(\{A, D, E\}, T_8) = 14 + 32 = 46$. If $\varepsilon = 120$, {A, D, E} is a low utility itemset.

To address the drawbacks in MEU, we propose a novel Two-Phase algorithm. In Phase I, we define *transaction-weighted utilization* and propose a model — *transaction-weighted utilization mining*. This model maintains a *Transaction-weighted Downward Closure Property*. Thus, only the combinations of high transaction-weighted utilization itemsets are added into the candidate set at each level during the level-wise search. Phase I may overestimate some low utility itemsets, but it never underestimates any itemsets. In phase II, only one extra database scan is performed to filter the overestimated itemsets.

3.1 Phase I

Definition 1. (Transaction Utility) The *transaction utility of transaction* T_q, denoted as $tu(T_q)$, is the sum of the utilities of all the items in T_q: $tu(T_q) = \sum_{i_p \in T_q} u(i_p,T_q)$. Table 1 (c) gives the transaction utility for each transaction in Table 1.

Definition 2. (Transaction-weighted Utilization) The *transaction-weighted utilization of an itemset X*, denoted as $twu(X)$, is the sum of the transaction utilities of all the transactions containing X: $twu(X) = \sum_{X \subseteq T_q \in D} tu(T_q)$. (3.2)

For the example in Table 1, $twu(AD) = tu(T_4) + tu(T_8) = 14 + 57 = 71$.

Definition 3. (High Transaction-weighted Utilization Itemset) For a given itemset X, X is a *high transaction-weighted utilization itemset* if $twu(X) \geq \varepsilon'$, where ε' is the user specified threshold.

Theorem 1. (Transaction-weighted Downward Closure Property) Let I^k be a k-itemset and I^{k-1} be a (k-1)-itemset such that $I^{k-1} \subset I^k$. If I^k is a high transaction-weighted utilization itemset, I^{k-1} is a high transaction-weighted utilization itemset.

Proof: Let T_{I^k} be the collection of the transactions containing I^k and $T_{I^{k-1}}$ be the collection containing I^{k-1}. Since $I^{k-1} \subset I^k$, $T_{I^{k-1}}$ is a superset of T_{I^k}. According to Definition 2, $twu(I^{k-1}) = \sum_{I^{k-1} \subseteq T_q \in D} tu(T_q) \geq \sum_{I^k \subseteq T_p \in D} tu(T_p) = twu(I^k) \geq \varepsilon'$ ❑

The *Transaction-weighted Downward Closure Property* indicates that any superset of a low transaction-weighted utilization itemset is low in transaction-weighted utilization. That is, only the combinations of high transaction-weighted utilization (k-1)-itemsets could be added into the candidate set C_k at each level.

Theorem 2. Let *HTWU* be the collection of all high transaction-weighted utilization itemsets in a transaction database *D*, and *HU* be the collection of high utility itemsets in *D*. If ε'= ε, then $HU \subseteq HTWU$.

Proof: $\forall X \in HU$, if *X* is a high utility itemset, then

$$\varepsilon' = \varepsilon \leq u(X) = \sum_{X \subseteq T_q} u(X, T_q) = \sum_{X \subseteq T_q} \sum_{i_p \in X} u(i_p, T_q) \leq \sum_{X \subseteq T_q} \sum_{i_p \in T_q} u(i_p, T_q) = \sum_{X \subseteq T_q} tu(T_q) = twu(X)$$

Thus, *X* is a high transaction-weighted utilization itemset and $X \in HTWU$. ❑

According to Theorem 2, we can utilize the *Transaction-weighted Downward Closure Property* in our *transaction-weighted utilization mining* in Phase I by assuming ε' = ε and prune those overestimated itemsets in Phase II.

Figure 1 shows the search space of Phase I. The level-wise search stops at the third level, one level less than MEU. (For larger databases, the savings should be more evident.) *Transaction-weighted utilization mining* model outperforms MEU in several aspects:

1) **Less candidates** — When ε' is large, the search space can be significantly reduced at the second level and higher levels. As shown in Figure 1, four out of 10 itemsets are pruned because they all contain item A. However, in MEU, the prediction hardly prunes any itemset at the beginning stages.
2) **Accuracy** — Based on Theorem 2, if we let ε'=ε, the complete set of high utility itemsets is a subset of the high transaction-weighted utilization itemsets discovered by our *transaction-weighted utilization mining* model. However, MEU may miss some high utility itemsets when the variation of itemset supports is large.
3) **Arithmetic complexity** — One of the kernel operations in the Two-Phase algorithm is the calculation for each itemset's transaction-weighted utilization as in equation 3.2, which only incurs add operations rather than a number of multiplications in MEU. Thus, the overall computation is much less complex.

3.2 Phase II

In Phase II, one database scan is performed to filter the high utility itemsets from high transaction-weighted utilization itemsets identified in Phase I. The number of high

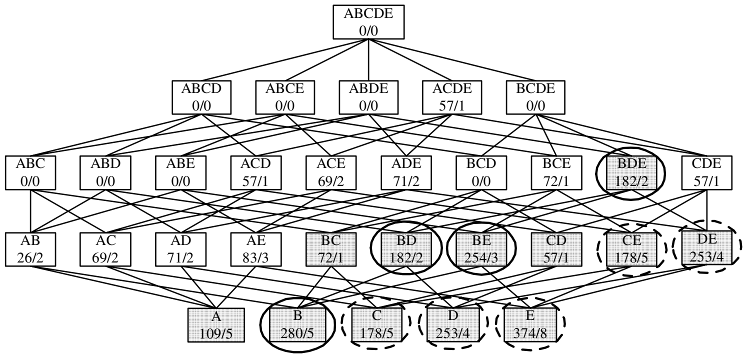

Fig. 1. Itemsets lattice related to the example in Table 1. ε' = 120. Itemsets in circles (solid and dashed) are the high transaction-weighted utilization itemsets in *transaction-weighted utilization mining model*. Gray-shaded boxes denote the search space. Itemsets in solid circles are high utility itemsets. Numbers in each box are transaction-weighted utilization / number of occurrence

transaction-weighted utilization itemsets is small when ε' is high. Hence, the time saved in Phase I may compensate for the cost incurred by the extra scan in Phase II.

In Figure 1, the high utility itemsets ({B}, {B, D}, {B, E} and {B, D, E}) are covered by the high transaction-weighted utilization itemsets. One database scan is performed in Phase II to prune 5 of the 9 itemsets since they are not high utility itemsets.

4 Experimental Evaluation and Performance Study

We run all our experiments on a 700-MHz Xeon 8-way shared memory parallel machine with a 4 Gbytes memory.

4.1 Synthetic Data from IBM Quest Data Generator

We use a synthetic database [9], T20.I6.D1000K. However, the IBM Quest data generator only generates the quantity of 0 or 1 for each item in a transaction. We randomly generate the quantity of each item in each transaction, ranging from 1 to 5. Utility tables are also synthetically created by assigning a utility value to each item randomly, ranging from 0.01 to 10.00. Observed from real world databases that most items are in the low profit range, we generate the utility values using a log normal distribution.

In Table 2, the number of candidate itemsets generated by Phase I at the first database scan decreases dramatically as the threshold goes up. However, the number of candidates generated by MEU is always 499500. We don't provide the exact numbers for MEU because it actually takes an inordinate amount of time (longer than 10 hours) to complete the second scan. Observed from Table 2, the *Transaction-weighted Downward Closure Property* in transaction-weighted utilization mining model can help prune candidates very effectively.

4.2 Real-World Market Data

We also evaluated the Two-Phase algorithm using a real world data from a major grocery chain store. There are 1,112,949 transactions and 46,086 items in the database. Each transaction consists of the products and the sales volume of each product purchased by a customer at a time point. The utility table describes the profit of each item.

We evaluate the scalability of our algorithm by varying the threshold. As shown in Table 3, it is fast and scales well. MEU doesn't work with this dataset because the number of 2-itemset candidates is so large (over 2 billion) that it overwhelms the memory available to us. Actually, very few machines can afford such a huge memory cost.

Table 2. The number of candidate itemsets generated by Phase I of Two-Phase algorithm vs. MEU

Threshold \ Databases		T20.I6.D1000K	
		Phase I	MEU
0.5%	*1st scan*	315615	499500
	2nd scan	18653	-
1%	*1st scan*	203841	499500
	2nd scan	183	-
1.5%	*1st scan*	135460	499500
	2nd scan	8	-
2%	*1st scan*	84666	499500
	2nd scan	1	-

Table 3. Scalability in execution time with varying minimum utility threshold of the real world database

Thresh-old	Running time (seconds)	# Candidates	# High transaction-weighted utilization (Phase I)	# High utility (Phase II)
1%	25.76	11936	9	2
0.75%	33.3	23229	26	3
0.5%	53.09	69425	80	5
0.25%	170.49	627506	457	17
0.1%	1074.94	7332326	3292	80

5 Conclusions

This paper proposed a Two-Phase algorithm that discovers high utility itemsets highly efficiently. The transaction-weighted utilization mining we proposed not only restricts the search space, but also covers all the high utility itemsets. Only one extra database scan is needed to filter the overestimated itemsets. Our algorithm requires fewer database scans, less memory space and less computational cost compared to the best existing utility mining algorithm. It can easily handle very large databases for which other existing algorithms are infeasible.

Acknowledgements

This work was supported in part by NSF grants CCF-0444405, CNS-0406341, CCR-0325207, DOE grant DE-FC02-01ER25485 and Intel Corp.

References

1. Agrawal and R. Srikant: Fast algorithms for mining association rules. 20th VLDB (1994)
2. Hong Yao, Howard J. Hamilton, and Cory J. Butz: A Foundational Approach to Mining Itemset Utilities from Databases. SDM (2004)
3. C.H. Cai, Ada W.C. Fu, C.H. Cheng, and W.W. Kwong: Mining Association Rules with Weighted Items. IDEAS (1998)

4. W. Wang, J. Yang, and P. Yu: Efficient Mining of Weighted Association Rules (WAR). 6th KDD (2000)
5. Feng Tao, Fionn Murtagh, and Mohsen Farid: Weighted Association Rule Mining using Weighted Support and Significance Framework. 9th KDD (2003)
6. S. Lu, H. Hu, and F. Li: Mining weighted association rules. Intelligent Data Analysis, 5(3) (2001), 211-225
7. B. Barber and H.J.Hamilton: Extracting share frequent itemsets with infrequent subsets. Data Mining and Knowledge Discovery, 7(2) (2003), 153-185
8. Raymond Chan, Qiang Yang, Yi-Dong Shen: Mining high utility Itemsets. ICDM (2003)
9. IBM data generator, http://www.almaden.ibm.com/software/quest/Resources/index.shtml