Enhanced Fast Discovery of Item Sets Based On Utility and Support Measures

S.Kanimozhi, P.Ranjana

Abstract: Utility based data mining is a new research area interested in all types of utility factors in data mining processes and targeted at incorporating utility considerations in data mining tasks. The experimental evaluation on artificial datasets shows that utility based algorithm executes only for the low threshold value. The proposed project fix the high threshold value for Frequent Item set & scan the whole database with high threshold value. It takes both the utility and support measures into consideration [1]. This method makes the item sets highly and frequently utilized and it generates different kinds of item sets namely High utility and high frequent itemsets (HULF), High utility and low frequent itemsets (HULF), Low utility and high frequent itemsets (LUHF), Low utility and low frequent itemsets (LUHF). Customer Relationship Management (CRM) is incorporated into the system by generating a list of customers who are frequent buyers of these four different kinds of item sets.

Keywords: Data Mining, Utility and Frequency Based Item set Mining, Customer Relationship Management, High Utility High Frequent Itemsets, High Utility Low Frequent Itemsets, Low Utility High Frequent Itemsets, Low Utility Low Frequent Itemsets.

I. INTRODUCTION

Data mining and knowledge discovery from data bases has received much attention in recent years. Data mining, the extraction of hidden predictive information from large databases is a powerful new technology with great potential to [1] help companies focus on the most important information in their data warehouses. Knowledge Discovery in Databases is the non-trivial process of identifying valid, previously unknown and potentially useful patterns in data. These patterns are used to make predictions or classifications about new data, explain existing data, summarize the contents of a large database to support decision making and provide graphical data visualization to aid humans in discovering deeper patterns. The KDD process comprises of a few steps leading from raw data to some form of new knowledge. The volume of data contained in a database often exceeds the ability to analyze it efficiently, resulting in a gap between the collection of data and its understanding. In knowledge discovery, techniques are constantly being developed and improved for discovering various types of patterns in databases. While these techniques were shown to be useful in numerous applications, new problems have also emerged. One of the major problems is that, in practice, it is all too easy to discover a huge number of patterns in a database. Most of these patterns are actually useless or uninteresting to the user. But due to the huge number of patterns, it is difficult for the user to comprehend and to identify those patterns that are interesting to him/her.

II. REVIEW OF THE EXISTING SYSTEM

The traditional association rule mining is used to identify frequently occurring patterns of item sets. ARM model treats all the items in the database equally by only considering if an item is present in a transaction or not. However, frequency of occurrence may not express the semantics of applications, because the user's interest may be related to other factors, [2] such as cost, profit, or aesthetic value. In this section describes existing algorithms, techniques used for frequent item set mining. Some of the algorithms are shown below.

A. UMining algorithm

UMining algorithm for mining all high utility itemsets using pruning strategy is guaranteed to never prune a high utility itemset. The framework of the UMining algorithm is shown below. The functions called by the UMining algorithm are Scan, CalculateAndStore, Discover, Generate, and Prune. The Scan function finds the set of all items in the transaction database T. The CalculateAndStore function accesses transaction database T to calculate the actual utility value of each k-itemset in C_k. It is assumed that each itemset S in C_k has associated with it a u field, denoted u(S), for storing its utility value. The Discover function selects all high utility itemsets in candidate set C_k. The Generate function generates all possible candidate k-itemsets from the (k-1) itemsets in C_{k-1}. The Prune function calculates the utility upper bound of each itemset in C_k based on the utility values of the candidate itemsets in C_{k-1}, and then refines moves any itemset with a utility upper bound less than minutil from C_k. It is assumed that each itemset S has an associated b field, denoted b(S), for storing the utility upper bound of S. The UMining algorithm follows the basic framework of the Apriori algorithm, but there are significant differences in three sub functions (Prune, CalculateAndStore, and Generate functions).

Algorithm UMining (T, f, minutil, K)

Task

Discovery of High Utility Itemsets

Input

Transaction database T
Utility function f
Utility value threshold minutil
Maximum size of itemset K.

Output

A set of high utility itemsets H.

[1] I = Scan (T);
[2] C_0 = I;
Discover - S, μ ∈ s that e transactions. If too many τ  rm of high utility itemsets. For a given  U)  μ ≥ al for Util s,μT
DBsupport(s,μ)=
μ and support threshold s the extended support measure Item set extended support measure can be identified:

\[ \text{support}(s, \mu) = \frac{|T_{s,\mu}|}{|DB|} \]

Item set S is utility-frequent if for a given utility threshold μ and support threshold s the extended support measure support(S, μ) is greater or equal to s. The support measure is always greater or equal to the extended support measure. Proof is trivial because when computing extended support we count only those transaction containing given itemset S that also gives minimum utility on S, but when computing “ordinary” support we count all transactions containing S. The practical consequence of [5] this statement is that frequent itemset mining algorithms can be used to mine utility-frequent itemsets.

**Input:**
- Database DB
- Constraints minutil and minsup

**Output:**
- all utility frequent itemsets

**Problem identification**

The practical usefulness of the frequent item set mining is limited by the significance of the discovered item sets. There are two principal limitations. A huge number of frequent item sets that are not interesting to the user are often generated when the minimum support is low. For example, there may be thousands of combinations of products that occur in 1% of the transactions. If too many [3] uninteresting frequent item sets are found, the user is forced to do additional work to select the item sets that are indeed interesting. Support, as defined based on the frequency of item sets, is not an adequate measure of a typical user's interest. This is because any organization is typically profit-oriented rather than towards the sale count value.

**III. SOLUTION TO THE PROPOSED PROBLEM**

**A. Design of EFUFM**

The FUFM (Fast Utility-Frequent Mining) which finds all utility-frequent itemsets within the given utility and support constraints threshold. Utility-frequent itemsets are a special form of high utility itemsets. For a given utility threshold μ each itemset S is associated with a set of transactions defined as T_{s,μ} = {T|S ⊆ T \land u(S, T) ≥ μ \land T \in DB}. On the basis of this set of transactions an extended support measure can be identified:

\[ \text{support}(s, \mu) = \frac{|T_{s,\mu}|}{|DB|} \]

The FUFM algorithm generates high utility itemsets using Combination Generator. It is simpler and executes faster than UMining algorithm.

**Algorithm FUM**

**Task**
Discovery of High Utility Itemsets

**Input**
Database DB {Set of Transactions}
Transaction \( T \in DB \)
Utility value threshold minUtil

**Output**
High Utility Itemsets HU

[1] Compute the utility value \( \forall \) single itemset
[2] For each \( T \in DB \)
[3] begin
[4] if \( T \notin S \) \{ where \( S \subseteq DB \mid S = [0 .. T-1] \} \]
[5] begin
[6] CandidateSet = CombinationGenerator (T)
[7] For each C ∈ CandidateSet
[8] begin
[9] if \( C \notin H \cup (U(C, T) \geq \text{min Util}) \)
[10] HU.add (C);
[12] end
[13] end
[14] return (HU);

**B. FUM and FUFM algorithm**

FUM algorithm generates high utility itemsets using Combination Generator. It is simpler and executes faster than UMining algorithm.

**Algorithm FUM**

**Task**
Discovery of High Utility Itemsets

**Input**
Database DB {Set of Transactions}
Transaction \( T \in DB \)
Utility value threshold minUtil

**Output**
High Utility Itemsets HU

[1] Compute the utility value \( \forall \) single itemset
[2] For each \( T \in DB \)
[3] begin
[4] \( C_k = \text{CalculateAndStore} \ (C_k, T, f) \);
[5] \( H = \text{Discover} \ (C_k, \text{minutil}) \);
[6] while (\(|C_k| > 0 \) and \( k < = K \))
[7] \( k = k + 1 \);
[8] \( C_k = \text{Generate}(C_{k-1}, I) \);
[9] \( C_k = \text{Prune} \ (C_k, C_{k-1}, \text{minutil}) \);
[10] \( C_k = \text{CalculateAndStore} \ (C_k, T, f) \);
[11] \( H = H \cup \text{Discover} \ (C_k, \text{minutil}) \);
[12] return (H);

**FUFM algorithm**

FUFM (Fast Utility-Frequent Mining) which finds all utility-frequent itemsets within the given utility and support constraints threshold. Utility-frequent itemsets are a special form of high utility itemsets. For a given utility threshold μ each itemset S is associated with a set of transactions defined as T_{s,μ} = {T|S ⊆ T \land u(S, T) ≥ μ \land T \in DB}. On the basis of this set of transactions an extended support measure can be identified:

\[ \text{support}(s, \mu) = \frac{|T_{s,\mu}|}{|DB|} \]
Umining algorithm generates high utility item sets and FUFM algorithm generates high utility and high frequent item sets. By combining the Umining and FUFM algorithm generates different kinds of item sets namely High utility and high frequent item sets, High utility and low frequent item sets, Low utility and high frequent item sets and Low utility and low frequent item sets. Customer Relationship Management is incorporated into the system by generating a list of customers who are frequent buyers of these four different kinds of item sets.

IV. FRAMEWORK FOR THE GENERATION OF EFUFM BASED ON UTILITY AND SUPPORT MEASURES

A. Generation of HUHF, HULF, LUHF, LULF item sets

HUHF mining algorithm
This algorithm is designed to generate high utility and high frequent item sets. It follows the basic framework of FUM and FUFM algorithms.

Task
Discovery of High Utility and High Frequent Item sets

Input
Database DB
Constraints minUtil and minSup

Output
High Utility and High Frequent Itemsets (HUHF)

1. Compute high utility itemsets H using FUM algorithm.
2. For each itemset I in H begin
   3. Compute support s
   4. if s >= minSup HUHF.add (I)
5. end
6. return (HUHF)

LULF Mining Algorithm
This algorithm is designed to generate Low utility and low frequent item sets. There are two phases in this algorithm. In the first phase using exhaustive search low utility itemsets are determined. In the second phase, using set difference

1. Let L = 1
2. Find the set of candidates of length L with support >= minSup
3. Compute extended support where T(s, μ) = {T | S ⊆ T ∧ u(S, T) < μ ∧ T ∈ DB} for all candidates and output low utility high frequent itemsets
4. L += 1
5. Use the frequent itemset mining algorithm to obtain new set of frequent candidates of length L from the old set of frequent candidates.
6. Stop if the new set is empty otherwise go to [3]
7. Return (LULF)

Table 1: Performance of EFUFM vs Umining algorithm

<table>
<thead>
<tr>
<th>High utility threshold value</th>
<th>EFUFM</th>
<th>Low utility threshold value</th>
<th>Utility mining algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HUI</td>
<td>Execution time (milliseconds)</td>
<td>HUI</td>
</tr>
<tr>
<td>1.25%</td>
<td>146</td>
<td>348.27</td>
<td>0.25%</td>
</tr>
<tr>
<td>1.5%</td>
<td>133</td>
<td>362.68</td>
<td>0.5%</td>
</tr>
<tr>
<td>1.75%</td>
<td>118</td>
<td>423.11</td>
<td>0.75%</td>
</tr>
<tr>
<td>2%</td>
<td>107</td>
<td>462.72</td>
<td>1%</td>
</tr>
</tbody>
</table>
function low utility low frequent itemsets are generated from LU and LUHF. Thus we generate Low utility and low frequent itemsets.

Algorithm LULFM

Task
Discovery of Low Utility and Low Frequent Itemsets

Input
Database DB
Constraints minUtil and minSup

Output
Low Utility and Low Frequent Itemsets (LULF)

1. Compute the utility value \( \forall \) single item set
2. For each \( T \in DB \)
3. begin
4. if \( T \notin S \) \{where \( S \subseteq DB \mid S = [0 .. T-1] \}
5. begin
6. Candidate set = CombinationGenerator (T)
7. For each \( C \in \) CandidateSet
8. begin
9. if \(( C \neq H ) \wedge U(C,T) < \text{minutil})\)
10. LU.add (C);
11. end
12. end
13. end
14]LULF = LU \LUHF /* Set Difference */
15. Return (LULF)

B. Generation of list of customers who buy these different kinds of itemsets

This algorithm is designed to generate a list of customers who frequently buy high [2]utility and high frequent itemsets, low utility and low frequent itemsets, low utility and high frequent itemsets, low utility and low frequent itemsets. Then Customer Relationship Management is incorporated into the system by tracking the customers who are frequent buyers of the different kinds of itemsets.

V. RESULTS AND DISCUSSION

In this section compares the performance of EFUFM and Umining algorithm. EFUFM executes faster than Umining and FUFM algorithm. Fig 2. Shows the execution time of Umining algorithm for Low threshold Value.

IV. CONCLUSION

The UMining and FUM algorithms are for mining all high utility item sets with low threshold value. The proposed method gives a new perspective in analyzing the item sets and this method scan the database with high threshold value. It is faster than the existing algorithm. The item sets that are both high frequent and high utility can be obtained using this method. From the basic framework of these algorithms the different kinds of item sets namely high utility [7] high frequent, high utility low frequent, low utility high frequent and low utility low frequent item sets are generated. Then Customer Relationship Management is incorporated into the system by tracking the customers who are frequent buyers of the different kinds of item sets.

REFERENCES


AUTHOR’S PROFILE

S. Kanamozhi received her B.E degree from Periyar Maniammai College of Technology for women under Trichy Anna University. Currently pursuing M.Tech degree in Hindustan University. Her research area is Data mining.

P. Ranjana received her M.E degree from Anna University. Currently pursuing Ph.D degree in Hindustan University. She has 14 years of teaching experience. She is working in Hindustan University as Assistant Professor in Department of Computer Science and Engineering. Her research area are graph theory and data mining.