©2010 INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS (0975 - 8887)
VOLUME 1 – NO. 16

A Novel Utility and Frequency Based Itemset Mining Approach for Improving CRM in Retail Business

Shankar S
Sri Krishna College of Engg and Tech, Coimbatore,India.

Dr.T.Purusothaman
Government College of Tech, Coimbatore, India

Kannimuthu S
Sri Krishna College of Engg and Tech, Coimbatore,India.

Vishnu Priya K
Sri Krishna College of Engg and Tech, Coimbatore,India.

ABSTRACT

The paradigm shift from ‘data-centered pattern mining’ to ‘domain driven actionable knowledge discovery’ has increased the need for considering the business yield (utility) and demand or rate of recurrence of the items (frequency) while mining a retail business transaction database. Such a data mining process will help in mining different types of itemsets of varying business utility and demand. We here present a set of algorithms for mining all types of utility and frequency based itemsets from a retail business transaction database which would significantly aid in inventory control and sales promotion. This set of algorithms are also capable of identifying the active customers of each such type of itemset mined and rank them based on their total or lifetime business value which would be extremely helpful in improving Customer Relationship Management (CRM) processes like campaign management and customer segmentation.

Categories and Subject Descriptors

H.2.8 [Database Applications]: *Data Mining*

General Terms

Algorithms, Design, Experimentation, Performance, Human Factors

Keywords

Data Mining, Utility and Frequency Based Itemset Mining, Customer Relationship Management, Domain Driven Data Mining, High Utility High Frequency Itemsets, High Utility Low Frequency Itemsets, Low Utility High Frequency Itemsets, Low Utility Low Frequency Itemsets, Semantic Intelligence, Active Customer List Generation.

1 INTRODUCTION

Traditional data mining processes concentrate on extracting knowledge from large databases irrespective of its significance to the user or the business contexts. The large number of patterns mined during knowledge discovery makes it cumbersome for the user to understand and identify the patterns that are interesting to him. In the due course, the need for considering the economic utility of the itemsets in the data mining process has gained wide acceptance. The quantitative measures like the support measure used in the traditional Association Rules Mining (ARM) [1,2], which is used to identify frequently occurring patterns of itemsets, reflects only the statistical correlation of items. It does not reflect their semantic significance. Such measures reflecting the statistical correlation may not measure how useful an itemset is in accordance with a user’s preferences (i.e., profit). The profit of an itemset depends not only on the support (the ratio of total count of the corresponding itemset occurrence to the total number of transactions) of the itemset, but also on the prices of the items in that develop a utility based itemset mining approach [14], to enable the user to conveniently express his or her preferences centered around the economic usefulness of itemsets as utility values and then find itemsets with utility values higher than a minimum threshold utility value as set by the user.

The paradigm shift from ‘data-centered pattern mining’ to ‘domain driven actionable knowledge discovery’ has increased the need for considering the business yield (utility) and demand of the items (frequency) while mining a retail business transaction database. Such a data mining process will help in mining different types of itemsets of varying business utility and demand. Data mining has the potential to aid the companies in their quest to become more customers’ oriented [4]. It plays a critical role in the overall CRM process, which includes interaction with the datamart or warehouse in one direction, and interaction with campaign management software in the other direction [4]. Today the trend is to integrate the data mining and campaign management process in order to gain a competitive advantage [12]. Keeping this in mind, we here present a set of algorithms for mining all types of utility and frequency based itemsets from a retail business transaction database which would significantly aid in inventory control and sales promotion. These set of algorithms are also capable of identifying the active customers of each such type of itemset mined and rank them based on their total business value which would be extremely helpful in improving Customer Relationship Management (CRM) processes like campaign management and customer segmentation. Utility based data mining is a new research area entranced in all types of utility factors like profit, significance, subjective interestingness, aesthetic value etc., which add economic and business utility to existing data mining processes and techniques. A research area within utility based data mining known as high utility itemset mining is aimed at finding itemsets that interpose high utility. The proposed set of algorithms is built using a combination of our previously developed efficient Fast Utility Mining (FUM) algorithm [11] and the existing Fast Frequent Utility Mining (FUFM)[13] algorithm. The Fast Utility Mining FUM or (Umining_New) algorithm employed for finding all high utility itemsets which satisfies the given utility constraint threshold, is faster and simpler than the original UMining algorithm [14] which is based on utility upper bound property.

The remainder of the paper is organized as follows. The existing and the proposed algorithms for mining high utility itemsets are explained in section 2 along with the results and findings of the experimental evaluation. The utility and frequency based itemset mining algorithms are explained in the section 3. The method of identifying and ranking the customers of each type of utility frequent itemsets mined is also explained in the sub sections of section 3.

2 ALGORITHMS FOR MINING HIGH UTILITY ITEMSETS

A high Utility itemset is the one with utility value greater than the minimum threshold utility as specified by the user depending upon his context of usage. A well known model for mining such high-utility itemset was defined by Yao, Hamilton and Butz [15] which is a generalization of the share-mining model [3, 6]. A well known algorithm for mining such high utility itemset is the UMining algorithm [14]. However this algorithm bears great disadvantages especially when it is applied to large transaction databases with numerous distinct items. In a real time business scenario say a retail supermarket, the number of distinct items present in the supermarket will be considerably large. However any transaction performed by a customer will contain only a fraction of all the distinct items present in the supermarket. UMining algorithm takes into account all distinct items in the supermarket, regardless of whether they are bought by the customer or not. Hence it results in the calculation of the utility values for all combinations of all distinct items present in the supermarket using the information obtained from the transactions in a transactional database. In the case of very large transactional databases, this will lead to unacceptable execution time delays. It will also require expensive vast main memory storage to accommodate all possible combinations of all the distinct items every time the UMining algorithm executes. UMining algorithm does not provide an obvious provision for dealing with duplicate itemsets which could occur in any transactional database. In our experimental evaluations, we also found that there are some instances where UMining algorithm may fail to find some of the high utility itemsets from the available transactions in the transactional database. These drawbacks prevent UMining algorithm from providing maximum speed, accuracy, reliability and cost effectiveness in domain driven data mining applications.

In order to overcome the drawbacks encountered in UMining algorithm and to develop an algorithm best suitable for mining High Utility Itemsets (HUI) in domain driven data mining applications, we developed the Fast Utility Mining (FUM) algorithm[11]. FUM algorithm demonstrates an appreciable semantic intelligence by considering only the distinct itemsets involved or defined in a transaction and not the entire set of available itemsets. FUM algorithm efficiently handles the duplicate itemsets. It checks whether a transaction containing the combination of items purchased in it, repeats its occurrence in a later transaction. If a later transaction also contains same itemset purchased in any of the previous transactions, then that transaction is ignored from processing. From the experiments conducted on the partially artificial datasets, we clearly observed that FUM algorithm provides absolute accuracy and proves to be extremely efficient in finding every possible high utility itemset from the transactions in the database.

```
FUM ALGORITHM
Task:    Discovery of High Utility Itemsets
Input:   Database DB {Set of Transactions} Transaction
         $T \in DB$ Minimum Utility value threshold minUtil
Output:  High Utility Itemsets  H
[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 |  S = [0 .. T-1]}
[5]        begin
[6]      Candidateset  = CombinationGenerator(T)
[7]                For each C $\in$ CandidateSet
[8]                begin
[9]                if $(C \notin H) \wedge (U(C,T) \geq \min Util)$
[10]                 H.add (C);
[11]               end
[12]       end
[13]     end
[14]     return (H);
CombinationGenerator(T) - Generate all possible combinations of
itemset $\in$ T
```

Figure. 1. Pseudo code of the FUM algorithm

The miss rate of FUM algorithm is nil when compared to that of the UMining algorithm. In these ways, in a domain driven data mining application, FUM algorithm fits itself perfectly by mining all the High Utility Itemsets(HUI) with maximum speed, accuracy, reliability and cost effectiveness[11].

2.1 Discussion of the FUM Algorithm

Let us consider Table 1 and Table 2 as input to the proposed FUM algorithm. In step 1 of the algorithm (Pseudo code given in the Figure.1.); we compute the utility values of all single itemsets say A, B, C, D and E [13].

Table 1. Database with 5 Transactions and 5 distinct Items

TID	A	B	C	D	E
1	0	0	18	0	1
2	0	6	0	1	1
3	5	0	4	0	2
4	2	3	1	1	1
5	0	0	4	0	3

Table 2. External Utilities of Items Given in Table 1

Item	A	B	C	D	E
Profit	2	11	4	7	5

In the second step, we begin a loop for processing each and every transaction present in the DB one by one. In the fourth step, the algorithm generates the itemsets in the current transaction. For example in Table I, the first transaction is represented as CE according to our algorithm, since only those two items were purchased in that transaction. Our algorithm omits the remaining items A, B and D. This is in contrast to the existing Umining algorithm, where all the items are taken into account, regardless whether they are bought by the customer or not. Our algorithm introduces more semantics in this aspect by considering only the items and their combinations which are actually purchased by the customers. In a similar way, the remaining transactions are processed. In this way, the execution time required for the generation of high utility itemsets is considerably reduced, which is illustrated through various graphs in section 2.2.

The algorithm also checks (step 4), whether a transaction defined by an itemset purchased in it, repeats its occurrence in a later transaction. If a later transaction also contains the same itemset purchased in any of the previous transactions, then that transaction is ignored from processing. In this way, the duplicate itemsets are removed.

In Step number 6, the candidate itemsets are generated using the CombinationGenerator(T) function, which takes itemset, purchased in a particular transaction as input and generate the various possible combinations of the itemset.

The combination generation is based on the concept proposed by Kenneth H. Rosen, Discrete Mathematics and Its applications [8]. Firstly, the items for which the combination is to be generated is put in the form of an array. Then the getNext() method is called until there are no more combinations left. The getNext() method returns an array of integers, which tells the order in which to arrange the original array of letters. For e.g if an itemset $\in$ T is say, ABC the function generates {A, B, C, AB, AC, BC, and ABC}.

In the consecutive steps, the algorithm analyzes each candidate belonging to the candidate itemsets generated. In step number 9, the algorithm computes the utility value of each and every candidate, U(C, T) [13]. If the utility value of a candidate is found to be more than the minimum utility threshold, which is given as input by the user, (say a sales manager) then that particular candidate is added to the set of High Utility Itemsets {H} (in step 10 of the proposed algorithm). The condition $C \notin H$ in step 9 simply ensures no duplicate high utility itemsets are generated.

2.2 Experimental Evaluations of FUM Algorithm

Both the algorithms Umining [14] and the proposed FUM algorithm were implemented in the Java programming language with Microsoft SQL Server 2000 as the backend. All experiments were performed on a PC with Intel Pentium D series 2.80 GHz processor and 1GB of main memory. The tests were conducted on a dataset with 200 transactions. This dataset was extracted from the transactional database of a leading retail supermarket. We performed necessary pre-processing to obtain a dataset that suits our need in the desired form. The experiments were conducted by varying the minimum utility threshold from 0.25 % to 2% on 200 transactions with 20 distinct items (see Table 3). It can be observed that when the minimum utility threshold is 0.25%, the number of high utility itemsets mined using UMining algorithm and Fast Utility Mining (FUM) algorithm are same. But there is a huge difference in the execution time of the two algorithms.

FUM algorithm executes almost 18 times faster than the UMining algorithm. Similarly we observed that the execution time of FUM algorithm proved to be extremely less than that of UMining algorithm especially when large numbers of itemsets are identified as high utility itemsets. Even as the minimum utility threshold is varied from 0.25% to 2%, FUM algorithm continues to identify all the possible high utility itemsets from the given dataset, considerably faster than that identified using UMining algorithm.

Table 3. Performance Comparison of UMINING and FUM Algorithm

Minimum Utility Threshold	Umining		FUM	
	HUI	Execution Time (milli seconds)	HUI	Execution Time (milli seconds)
0.25%	409	3694.76	409	210.13
0.5%	274	2112.0	275	279.44
0.75%	188	1540.55	190	333.42
1%	163	1326.59	163	330.75
1.25%	147	661.37	147	350.27
1.5%	132	849.91	132	368.68
1.75%	117	927.09	117	427.22
2%	108	967.45	108	471.61

Table 4. Miss Rate Comparison of UMining and FUM Algorithms

Minimum Utility Threshold	# HUI (Umining)	# HUI (FUM)	Umining Miss rate
1%	163	163	0%
0.75%	188	190	1%
0.5%	274	275	0.3%
0.25%	409	409	0%
0.1%	479	479	0%

From the results of the above experiment, we also noted another interesting fact. FUM algorithm provides absolute accuracy and proves to be extremely efficient in finding every possible high utility itemset from the dataset. The miss rate of FUM algorithm is an absolute zero when compared to that of the UMining algorithm as shown in Table 4. Though this failure of UMining algorithm is of negligible significance, it may at times prove to be a costly failure especially if the high utility itemset missed to be found by the UMining algorithm means the most significant and most potential high utility itemset to the user in a specific business scenario.

Table 5. Performance Comparison of UMining and FUM Algorithms Based on Number of Distinct Items

Number of distinct items	Execution Time(milli seconds)	
	FUM	**Umining**
15	389.42	484.76
20	330.75	1326.59
30	362.84	4493.17
50	377.29	*
100	365.91	*

* Indicates that, we have to manually stop the system as it hanged while executing the Umining algorithm. Hence the execution time could not be measured for 50 and 100 items respectively.

The strength of FUM algorithm lies in the fact that it demonstrates semantic intelligence by considering only the distinct itemsets involved or defined in a transaction and not the entire set of available itemsets. To ensure our claim we conducted another experiment by continuously varying the number of distinct items that are totally available keeping the minimum utility threshold constant at 1% throughout the experiment.

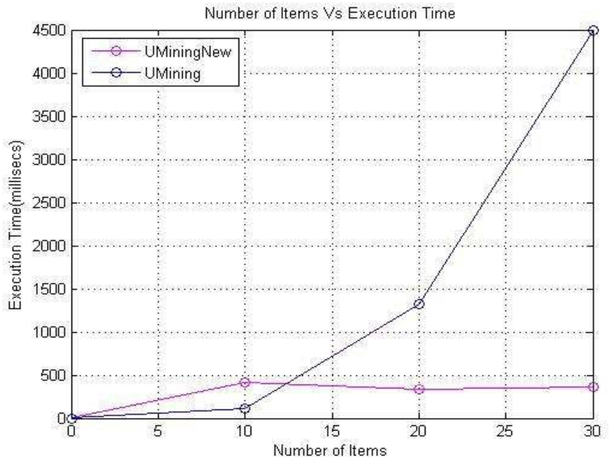

Figure. 2. Number of Items Vs Execution Time (FUM vs. Umining)

From the real time database that we obtained from the retail supermarket, we performed pre-processing in such a way to extract the required dataset with increasing number of distinct items that are taken into account. As shown in the Table 5 and graph in Figure.2, as the number of distinct items increased, the execution time of the UMining algorithm also increased drastically. When we experimented with 200 transactions and 50 distinct items, on a PC with Intel Pentium D series 2.80 GHz processor and 1GB of main memory, the system got hanged during the execution of UMining algorithm. Compared to UMining algorithm, the execution time of FUM algorithm was very less and exhibited a decreasing trend as the number of distinct items increased.

3 FRAMEWORK FOR GENERATION OF THE DIFFERENT TYPES OF ITEMSETS

Mining of Utility Frequent Itemsets is another interesting area that emerged with the rise of Utility Based Data Mining. In addition to subjectively defined utility, we also take into account the frequency of itemsets in utility frequent itemset mining [13]. Well known algorithms used in this type of mining are 2P-UF algorithm [7] and Fast Utility Frequent Mining (FUFM) [13] algorithm as we have observed in our literature review.

Fast Utility Frequent Mining - FUFM algorithm [13] treats utility-frequent itemsets as a special form of frequent itemsets which is in contrast with 2P-UF algorithm (based on the quasi support measure) since it treats them as a special form of high utility itemsets. It proves to be efficient because support measure has anti-monotone property and assures efficient mining approach by introducing a special form of support called as extended support. It also uses efficient frequent itemset mining methods to improve the speed of the algorithm.

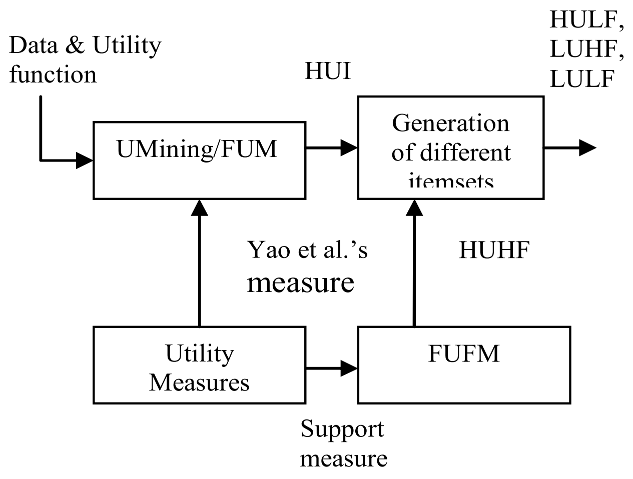

Figure. 3. Proposed System Architecture for Different Utility Frequent Itemset Mining

FUFM algorithm does not have disadvantages and inefficiencies of the 2P-UF algorithm such as huge space consumption and extremely slow candidate generation. FUFM algorithm finds High Utility High frequency (HUHF) items. From the discussions with our clients, we learnt and understood their knowledge requirements. Using HUHF itemsets generated by the FUFM algorithm and the High Utility Itemsets (HUI) generated using our FUM algorithm, we have generated three new itemsets namely High utility and low frequency itemsets (HULF), Low utility and high frequency itemsets (LUHF) and Low utility and low frequency itemsets (LULF). Based on our experiments, we have proposed system architecture for the generation of different kinds of utility frequent itemsets which is self explanatory as in Figure. 3.

3.1 Mining of the HUHF Itemsets

We made our efforts to incorporate support consideration in the FUM algorithm in order to mine the HUHF itemsets from a transaction database.

In the Fast Utility Mining with Frequency consideration algorithm (FUM-F algorithm), the High Utility Itemsets (HUI) are mined using the FUM algorithm initially. Then for each HUI mined, corresponding support is calculated and checked with the minimum support threshold (minSup). Those HUI with support greater than the minSup are added to the HUHF itemset list. FUFM and FUM-F algorithms were tested on the same datasets used for testing the FUM algorithm. The experiments were conducted by varying the minimum support threshold and minimum utility threshold for many sets of different number of transactions and distinct items.

As shown in the Table 6 and 7, though the number of HUHF Itemsets (HUHFI) found using FUM-F were slightly higher than that identified by FUFM, we can observe that FUFM executes very much faster than the FUM-F even when the number of transactions and distinct items increases.

FUM-F ALGORITHM

Task*:* Discovery of High Utility and High Frequency Itemsets
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
[3] begin
[4]         Compute support s
[5]         if s > = minSup
[6]                 HUHF.add (I)
[7] end
[8] return (HUHF)
```

Figure. 4. Pseudo code of the FUM-F algorithm

Table 6 Performance Comparison of FUM and FUFM Algorithms For 100 Transactions & 15 Distinct Items

Minimum Support Threshold (α)	Minimum Utility Threshold (μ)	FUFM		FUM-F	
		HUHFI	Execution Time (seconds)	HUHFI	Execution Time (seconds)
2	10	16	17.563	20	287.766
2	15	10	13.297	13	265.203
3	10	14	16.437	16	324.891
5	15	4	11.781	4	410.968

Thus FUM-F is reliable and accurate but not as fast as FUFM in finding the HUHF itemsets. If the business user lays importance on absolute accuracy and reliability, he can use FUM-F for mining all the HUHF itemsets. Instead if stress is on the execution time then FUFM algorithm would be the best fit to mine all the HUHF itemsets.

3.2 Generation of HUHFI Customer List

Once the HUHF itemsets are mined, the associated customers who buy HUHF items can also be identified and ranked based on their value. The customer who buys the maximum number of HUHF items will have the highest customer value and hence will be ranked at the top. The algorithm for the HUHFI customer list generation is given below which is self-explanatory.

HUHFI CUSTOMER LIST GENERATION ALGORITHM

Task*:* Generation of HUHFI customer list.
Input: Database DB; Constraints minUtil and minSup
Output: Ranked list of customers who buy HUHF items

```
[1]  Compute High utility and high frequent (HUHF) itemsets
     using FUFM algorithm
[2]  For each I ∈ HUHF itemset, scan the database DB to find
     the customers who buy that itemset
[3]  Increment the count value associated with the customer
     who is a buyer of I.
[4]  Stop if the HUHF is empty else Go to [2]
[5]  List the HUHF customers in descending order of the count
     value associated with each customer
[6]  return (list of HUHF customers)
```

Figure. 5. Pseudo code of the HUHFI customer list generation algorithm

Table 7.
Performance Comparison of FUM AND FUFM Algorithms for 200 Transactions & 100 Distinct Items

Minimum Support Threshold (α)	Minimum Utility Threshold (μ)	FUFM		FUM-F	
		HUHFI	Execution Time (sec)	HUHFI	Execution Time (sec)
2	10	52	15.75	60	169.484
3	10	44	15.891	52	139.156
5	15	15	8.422	16	279.718
5	20	7	4.125	9	198.953

To illustrate the mining of HUHF itemsets and the corresponding ranked customer list, we consider the following simple set of transactions as shown in Table 8. There are 10 transactions with 5 distinct items in the sample database. The external utilities of these items are given in the Table 9.

Table 8. Sample Database with 10 Transactions & 5 Distinct Items

TID	1	2	3	4	5	Cust ID
1	0	0	18	0	1	2
2	0	6	0	1	1	5
3	2	0	1	0	1	2
4	1	0	0	1	1	3
5	0	0	4	0	2	4
6	1	1	0	0	0	1
7	0	10	0	1	1	2
8	3	0	25	3	1	3
9	1	1	0	0	0	4
10	0	6	2	0	2	2

Table 9. External utilties of Items in the Sample Database

Item	1	2	3	4	5
Profit	3	10	1	6	5

The various possible itemsets and their respective utility and support values are calculated. Considering a minimum utility threshold (minUtil) of 36 and minimum support threshold (minSup) of 3, we applied the FUFM algorithm to this sample database. The list of HUHF itemsets mined using FUFM algorithm is shown in the Table 10. Then the HUHFI customer list generation algorithm was applied to this list of HUHF itemsets. The output is the ranked list of customers of HUHFI as shown in the Table 11 and 12.

Table 10. List of HUHF Itemsets Mined from the Sample Database using FUFM algorithm with their Utility & Support Values

Itemsets	Utility >= 36	Support >= 3
2	240	5
3	50	5
4	36	4
5	50	8
2, 5	240	3
3, 5	85	5
4, 5	56	4

Identical form of results were obtained when this approach was tested using the real time retail transaction database as explained in the section 2.2. The customer details can be then tracked using the customer ID.

Table 11. List of HUHFI Customer List Generated Using HUHFI Customer List Generation Algorithm

Itemset	Count	Cust 1	Cust 2	Cust 3	Cust 4	Cust 5
2	5	1	2	0	1	1
3	5	0	3	1	1	0
4	4	0	1	2	0	1
5	8	0	4	2	1	1
2, 5	3	0	2	0	0	1
3, 5	5	0	3	1	1	0
4, 5	4	0	1	2	0	1
Total	**34**	**1**	**16**	**8**	**4**	**5**

Table 12. Ranked List of HUHFI Customers

RANK	CUSTOMER ID
1	2
2	3
3	5
4	4
5	1

3.3 Mining of the HULF Itemsets

Once the HUHF itemsets are generated using either FUFM or FUM-F algorithm, all the other three types of itemsets can be mined easily. The algorithm for mining HULF itemsets follows the combined framework of FUM and FUFM algorithm. In all our experiments, we chose FUFM algorithm for mining HUHF itemsets. Here, the first phase is to generate High Utility Itemsets (HUI) using FUM algorithm. In the second phase High Utility High Frequent Itemsets (HUHFI) are generated using FUFM. Then using set difference function high utility low frequent itemsets are generated from HUI and HUHFI.

```
                    Algorithm HULFM
Task:  Discovery of High Utility and Low Frequency
          (HULF) Itemsets
Input:   Database DB; Constraints minUtil and minSup
Output: High Utility and Low Frequency Itemsets (HULF)

[1] Compute High utility itemsets HU using FUM algorithm.
[2] Compute High utility and high frequent itemsets HUHF using
FUFM algorithm.
[3] HULF = HU \ HUHF        /*set difference operation*/
[4] return (HULF)
```

Figure. 6. Pseudo code of the HULF itemsets mining algorithm

3.4 Mining of the LUHF Itemsets

This algorithm designed to generate Low Utility and High Frequency itemsets from transactional databases, follows the basic framework of FUFM algorithm with the extended support to mine the itemsets with low utility(reverse of high utility) but of high frequency.

```
                  ALGORITHM LUHFM
Task: Discovery of Low Utility and High Frequency (LUHF)
Itemsets
Input: Database DB; Constraints minUtil and minSup
Output: Low Utility and High Frequency Itemsets (LUHF)

[1] L = 1
[2] Find the set of candidates of length L with support >=minSup
[3] Compute extended support as explained in [13] where $TS,\mu = \{T|S \subseteq T \wedge u(S, T) < \mu \wedge T \in DB\}$ 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 (LUHF)
```

Figure. 7. Pseudo code of the LUHF itemsets mining algorithm

3.5 Mining of the LULF Itemsets

This algorithm designed to generate Low Utility and Low Frequency itemsets from transactional databases has two phases as shown in Figure. 8. In the first phase low utility itemsets are determined using exhaustive search. In the second phase the low utility and high frequency itemsets are removed from the low utility itemsets which results in the generation of Low Utility and Low Frequency (LULF) itemsets. It is built mainly using the framework of FUM algorithm with the concept of utility consideration reversed.

ALGORITHM LULFM

```
Task: Discovery of Low Utility and Low Frequency (LULF) Itemsets
Input: Database DB; Constraints minUtil and minSup,       LUHF
Output: Low Utility and Low Frequency Itemsets (LULF)
[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 |  S = [0 .. T-1]}
[5]   begin
[6]        Candidateset  =     CombinationGenerator (T)
[7]        For each C $\in$ CandidateSet
[8]        begin
[9]          if ( $C \notin H$ ) ^ U(C,T) < minutil )
[10]           LU.add (C);
[11]        end
[12]  end
[13]       end
[14]       LULF = LU \ LUHF     /*set minus operation*/
[15]       return (LULF)
CombinationGenerator(T) - Generate all possible combinations of itemset $\in$ T
```

Figure. 8. Pseudo code of the LULF itemsets mining algorithm

3.6 Generation of Customer List for Each Type of Itemset

The method of generating the ranked customer list for each type of itemsets is similar to the method explained in the section 3.2. The customer who buys the maximum number of items of a particular type of itemset will have the highest customer value with regard to that type of itemset and hence will be ranked at the top in the respective category.

HULFI CUSTOMER LIST GENERATION ALGORITHM

```
Task: Generation of HULFI customer list.
Input: Database DB; Constraints minUtil and minSup
Output: Ranked list of customers who buy HULF items
[1] Compute High utility and low frequent (HULF) itemsets using HULFM algorithm
[2]  For each I $\in$ HULF itemset, scan the database DB to find the customers who buy that itemset
[3]  Increment the count value associated with the customer who is a buyer of I.
[4]  Stop if the HULF is empty else go to [2]
[5]  List the HULF customers in descending order of the count value associated with each customer
[6]  return (list of HULF customers)
```

Figure. 9. Pseudo code of the HULFI customer list generation

Algorithm

LUHFI CUSTOMER LIST GENERATION ALGORITHM

```
Task: Generation of LUHFI customer list.
Input: Database DB; Constraints minUtil and minSup
Output: Ranked list of customers who buy LUHF items
[1] Compute Low utility and high frequent (LUHF) itemsets using LUHFM algorithm
[2] For each I $\in$ LUHF itemset, scan the database DB to find the customers who buy that itemset
[3] Increment the count value associated with the customer who is a buyer of I.
[4] Stop if the LUHF is empty else go to [2]
[5] List the LUHF customers in descending order of the count value associated with each customer
[6] return (list of LUHF customers)
```

Figure. 10. Pseudo code of the LUHFI customer list generation Algorithm

LULFI CUSTOMER LIST GENERATION ALGORITHM

```
Task: Generation of LULFI customer list.
Input: Database DB; Constraints minUtil and minSup
Output: Ranked list of customers who buy LULF items
[1]      Compute Low utility and low frequent (LULF) itemsets using LULFM algorithm
[2]      For each I $\in$ LULF itemset, scan the database DB to find the customers who buy that itemset
[3]      Increment the count value associated with the customer who is a buyer of I.
[4]      Stop if the LULF is empty else go to [2]
[5]      List the LULF customers in descending order of the count value associated with each customer
[6]      return (list of LULF customers)
```

Figure. 11. Pseudo code of the LULFI customer list generation Algorithm

3.7 Significance of the Proposed Approach

The generation of different types of itemsets based on their business utility and rate of recurrence in the transactions made by the customer can greatly aid in the inventory control and sales promotion. Identifying the corresponding customers will greatly benefit in customer segmentation, campaign management and Customer Relationship Management (CRM). Inventory control is a function of materials management, and the objective is to keep the total cost associated with the system to a minimum. Out of thousands of items held in an inventory of a typical organization, only a small percentage of them deserve management's closest attention and tightest control [9].

The High Utility and High Frequency Itemsets like say Basumati Rice can be stocked to a greater extent in order to meet the greater demand. Stringent safety measures can be followed in storing such items. This will ensure definite and increased profits.

The Low Utility and Low Frequency Items are also of importance to the sales and inventory management. From the view of inventory control, LULF items can be stocked less. When there is an excess availability of LULF items, they can be given as premium gifts for the active customers who purchase the HUHF items. Thus, the sales of HUHF items can also be increased. Moreover the organization will be able to provide premium service to the highly valuable active customers thereby strengthening the customer loyalty and such customer retention. There is a possibility that customers who buy HUHF itemsets may become responders for LULF itemsets if they feel that certain LULF items received as gifts are really worth useful to them. Such customers will start using the LULF items. Thus, its demand will increase simultaneously thereby increasing the business value. This also could benefit the business positively.

Certain High Utility and Low Frequency Itemsets can be of similar value and at times of greater value than the HUHF items. For example gems like diamonds, platinum etc may not be bought by the customers frequently. But such items have enormous business value. Identifying active customers of such HULF items using the proposed approach will be simple and efficient in segmenting the most valuable customers and designing a customized campaign management programme for such customers.

Advantage of LUHF itemset generation is that, it will increase the revenue of the business substantially. LUHF items are the most commonly purchased items. The maximization of the life time business value of the entire customer base is the prime objective of CRM. The key to attain this objective is to understand the behavior of the customer. Clear Customer Understanding requires properly focused customer segmentation and actions to maximize customer convention, retention, loyalty and profitability. As per the law of diminishing return in economics, unfocussed actions like unrestricted attempts to access or retain all the customers may impact the customer lifetime business value in a negative manner [10]. By using our approach several classes of customers can be segmented easily. Each class of customers has a unique importance. Within each class, the customers are once again ranked in the descending order of their current lifetime business value. Thus the proposed approach makes this entire process extremely simple. The changes in the customer behavior like the lift, shift and retention [10] can be studied effectively and appropriate campaign management programmes and other CRM programmes can be designed.

4 CONCLUSION

In this paper, we have explained about the novel Fast Utility Mining (FUM) algorithm. The experimental evaluations show that FUM algorithm is faster, more accurate, more reliable and best suited for domain driven data mining applications than the existing UMining algorithm. We discussed the method of incorporating the frequency consideration in FUM algorithm. A novel approach to mine different types of utility and frequency based itemsets using a framework of FUM and Fast Utility Frequent Mining (FUFM) algorithm has been explained with a simple illustration. Algorithms for identifying and ranking active customers of each type of itemsets and their significance in Customer Relationship Management (CRM) processes in retail business has also been discussed.

REFERENCES

[1] Agrawal R, Srikant R, ‘Fast algorithms for mining association rules’, *Proceedings of 20th International Conference on Very Large Databases*, Santiago, Chile, pp. 487–499, 1994.

[2] Agrawal R., Imielinski T., Swami A.: Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data*, Washington, D.C., may 1993, pp. 207–216.

[3] Carter C, Hamilton H J, Cercone N, ‘Share based measures for itemsets’, *Proceedings of First European Conference on the Principles of Data Mining and Knowledge Discovery*, pp. 14–24, 1997.

[4] Chris Rygielski, Jyun-Cheng Wang, David C. Yen, “Data Mining Techniques for Customer Relationship Management (CRM)”, *Technology in Society, Elsevier Science Ltd,* pp. 483-502, 2002.

[5] Han J, Pei J, Yin. Y. “Mining Frequent Patterns without Candidate Generation”. *Proc. of ACM-SIGMOD*, 2000.

[6] Hilderman R J Carter C L Hamilton H J Cercone N, ‘Mining market basket data using share measures and characterized itemsets’, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 159–170, 1998.

[7] Jieh-shan yeh, Yu-Chiang Li and Chin – Chen Chang, ‘Two-Phase Algorithms for a Novel Utility-Frequent mining Model’, *PAKDD Workshop*, pp. 433-444, 2007.

[8] Kenneth H. Rosen, “Discrete Mathematics and Its applications”, *Mc Graw Hill. 4th edition*, 298-300.

[9] Pradip Kumar Bala, “Mining Association Rules for Selective Inventory Control”, *Journal of the Academy of Business and Economics,* Feb 2008 available at http://findarticles.com/p/articles/mi_m0OGT/is_2_8/ai_n31215552

[10] Richard Boire, “Data Mining for Customer Loyalty”, *Direct Marketing*, March 2009.

[11] S.Shankar, Dr.T.Purusothaman, S.Jayanthi, Nishanth Babu, “A Fast Algorithm for Mining high Utility Itemsets”, Proceedings of the *IEEE International Advance Computing Conference (IACC 09)*, Patiala, India, 2009.

[12] Thearling K, “Increasing customer value by integrating data mining and campaign management software”, *Exchange Applications White Paper, Exchange Applications*, Inc.1998 available at http://www.crmforum.com/crm—forum—white—papers/icv/sld01.htm

[13] Vid Podpecan, Nada Lavra and Igor Kononenko, ‘A Fast Algorithm for Mining Utility-Frequent Itemsets’, *The Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases,* 2007.

[14] Yao H and Hamilton H J, ‘Mining itemset utilities from transaction databases’, *Data & Knowledge Engineering,* pp. 59: 603-626, 2006.

[15] Yao H, Hamilton H J, Butz C J, ‘A foundational approach to mining itemset utilities from databases’, *Proceedings of the Third SIAM International Conference on Data Mining*, Orlando, Florida, pp. 482–486, 2004.