

Survey on Approaches for Sequential Pattern Mining and High Utility Sequential Pattern Mining

Uma Dave¹ Jayna Shah²

¹Student ²Assistant Professor

^{1,2}Department of Computer Engineering

^{1,2}Sardar Vallabhbhai Patel, Institute Of Technology, Vasad, Gujarat, India

Abstract— Sequential pattern mining plays an important role in many applications, such as bioinformatics and consumer behaviour analysis. However, the classic frequency-based framework often leads to many patterns being identified, most of which are not informative enough for business decision-making. So a recent effort has been to incorporate utility into the sequential pattern selection framework, so that high utility (frequent or infrequent) sequential patterns are mined which address typical business concerns such as dollar value associated with each pattern. So this paper presents detailed different approaches adopted for Sequential pattern mining algorithms as well as high utility sequential pattern mining techniques.

General Terms: Sequential Pattern Mining ,Utility Mining , High utility Sequential pattern Mining

Key words: GSP, SPADE, SPAM, FREESPAN, PREFIXSPAN, su, suub , asu , hus

I. INTRODUCTION

Association rule mining is a popular data mining task which is essential to a wide range of practical applications viz. Supermarket promotions, biomedical applications etc. However, time regularity of items in the databases cannot be found by using traditional association rule mining approaches. So the concept of Sequential Pattern Mining has emerged (Agrawal and shrikant, 1995) for discovery of regularity knowledge which has proven to be very essential for order based critical business problems such as behaviour analysis, gene analysis, web log mining and DNA sequences.

Basically Sequential pattern Mining is the process of finding complete set of frequent subsequence that is the subsequence whose occurrence frequency in the sequence database is greater than or equal to minimum support. So it considers not only of the frequency relationships of items in the pattern but also the order relationship of items according to timestamp of items. Consider a sequential pattern like “< (Computer), (Printer) > “. The pattern states that most customers will buy a printer in a period of time after they have bought a computer.

However the pattern found by Sequential pattern Mining techniques do not reflect any other factors such as cost, price or profit. Consider another pattern “< (Diamond), (Necklace) > “in a sequence database. Assume it is not high frequency pattern in the database. So it is not identified by the process of Sequential pattern Mining but the pattern may contribute a large portion to the overall of the shop due to its high profit. Accordingly some product with high profit but low frequency may not be discovered by traditional approaches.

So to address the above problem a new concept called “Utility” has been introduced in Sequential Pattern

Mining which considers not only quantity of items in a sequence but also the individual profit of items in a quantitative sequence database. However, when new sequences are inserted in database, the whole procure of mining high utility sequential has to be duplicated. So Incremental Mining has been developed for mining High Utility Sequential Pattern from incremental database.

This survey focuses different sequential pattern mining and high utility sequential pattern mining algorithm.

II. SEQUENTIAL PATTERN MINING

The sequential pattern mining problem was first addressed by Agrawal and Srikant and was defined as follows: “Given a database of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, sequential pattern mining is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern.” For example, having bought a notebook, a customer comes back to buy a PDA and a WLAN card next time.

Sequential pattern mining has been actively studied during latest years. As a result of recent research, there is a diversity of algorithms for sequential pattern mining. These algorithms have basic steps to form reasonable sequential patterns, and the steps consist of the following five phases:

A. Sort Phase:

In this phase, Transaction database is sorted by user id and then sorted by transaction time. Through this phase, the transaction database becomes sequence database ordered by time.

B. L-Itemsets (Large Item Sets) Phase:

In this phase, a large item is a certain item which is presented frequently. The acceptable standard of selection of large items is minimum support. Support of items indicates the number of peoples who contain the items. So, minimum support is a threshold value for picking out large items. We can determine minimum support as 20 percent or 25 percent of the whole transactions, and then choose the items whose frequencies are above the minimum support.

C. Transformation Phase:

In this phase, the sequence database which is passed through the first phase transforms into reduced sequence database. The reduced database consists of selected sequences which contain large items. According to transformation of sequence database, we can easily focus on significant sequence.

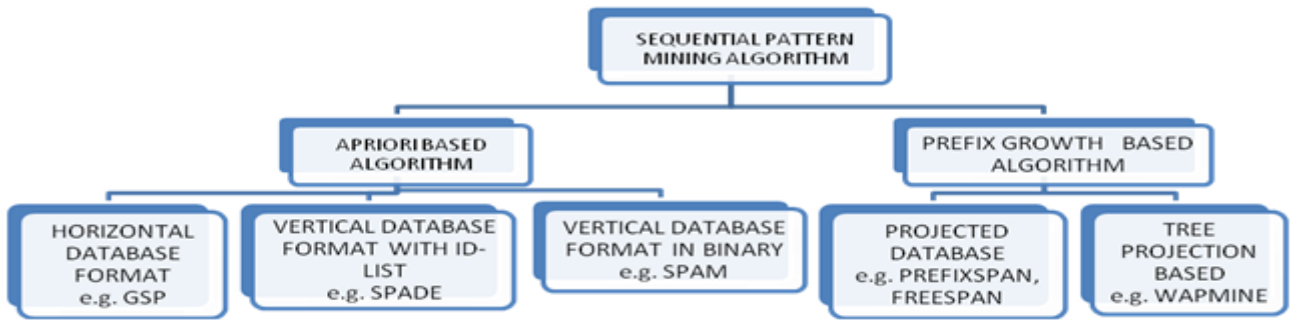


Fig. 1: Classification of Sequential Pattern Mining Algorithm

D. Sequence phase:

In this phase, we can generate sequential pattern within the transformed sequential database which is a result of third phase.

E. Maximal Phase:

This phase selects the maximal sequential pattern among the candidate sequential patterns. The maximal sequential pattern is the final pattern of sequential pattern mining.

III. SEQUENTIAL PATTERN MINING ALGORITHM

Sequential pattern mining algorithm are mainly classified into two types:

- (1) Apriori based algorithm
- (2) Prefix growth based algorithm.

The detailed classification of this algorithm is shown in fig1. The description of this algorithm is given as below.

A. Apriori based algorithm:

The Apriori was proposed by Agrawal and Srikant in 1994, based on Apriori property and use the Apriori-generate join procedure to generate candidate sequences. The apriori property states that "All nonempty subsets of a frequent itemset must also be frequent." It is also described as antimonotonic (or downward-closed), in that if a sequence cannot pass the minimum support test, its entire super sequences will also fail the test.

Key features of Apriori-based algorithm are:

1) Breadth-First Search:

Apriori-based algorithms are described as breath-first (level-wise) search algorithms because they construct all the k-sequences, in kth iteration of the algorithm, as they traverse the search space.

2) Generate-And-Test:

This feature is used by the very early algorithms in sequential pattern mining. Algorithms that depend on this feature only display an inefficient pruning method and generate an explosive number of candidate sequences and then test each one by one for satisfying some user specified constraints, consuming a lot of memory in the early stages of mining.

3) Multiple Scans Of The Database:

This feature entails scanning the original database to ascertain whether a long list of generated candidate sequences is frequent or not. It is a very undesirable

characteristic of most apriori-based algorithms and requires a lot of processing time and I/O cost.

a) GSP (Generalized Sequential Pattern Mining Algorithm):

It is a horizontal data format based sequential pattern mining algorithm. The first scan of database finds all of the frequent items which form the set of single item frequent sequences. Each subsequent pass starts with a seed set of sequential patterns, which is the set of sequential patterns found in the previous pass. This seed set is used to generate new potential patterns, called candidate sequences. Each candidate sequence contains one more item than a seed sequential pattern, where each element in the pattern may contain one or multiple items. The number of items in a sequence is called the length of the sequence. So, all the candidate sequences in a pass will have the same length. The scan of the database in one pass finds the support for each candidate sequence. All of the candidates whose support in the database is no less than min support form the set of the newly found sequential patterns. This set then becomes the seed set for the next pass. The algorithm terminates when no new sequential pattern is found in a pass, or no candidate sequence can be generated.

b) SPADE (Sequential Pattern Discovery using Equivalent Classes):

Besides the horizontal formatting method (GSP), the sequence database can be transformed into a vertical format consisting of items' id-lists. The id-list of an item a is a list of (sequence-id, timestamp) pairs indicating the occurring timestamps of the item in that sequence. Searching in the lattice formed by id-list intersections, completes the mining in three passes of database scanning. Nevertheless, additional computation time is required to transform a database of horizontal layout to vertical format, which also requires additional storage space several times larger than that of the original sequence database.

c) SPAM (Sequential Pattern Mining using A Bitmap Representation):

It integrates the ideas of GSP, SPADE, and FreeSpan. The entire algorithm with its data structures fits in main memory, and is claimed to be the first strategy for mining sequential patterns to traverse the lexicographical sequence tree in depth-first fashion. SPAM traverses the sequence tree in

depth-first search manner and checks the support of each sequence-extended or itemset-extended child against `min_sup` recursively for efficient support-counting. SPAM uses a vertical bitmap data structure representation of the database which is similar to the id list in SPADE. SPAM is similar to SPADE, but it uses bitwise operations rather than regular and temporal joins. When SPAM was compared to SPADE, it was found to outperform SPADE by a factor of 2.5, while SPADE is 5 to 20 times more space-efficient than SPAM, making the choice between the two a matter of a space-time trade-off.

B. Prefix Growth Based Algorithm:

The prefix growth-method emerged in the early 2000s, as a solution to the problem of generate-and-test. The key idea is to avoid the candidate generation step altogether, and to focus the search on a restricted portion of the initial database.

The search space partitioning feature plays an important role in pattern-growth. Almost every pattern-growth algorithm starts by building a representation of the database to be mined, then proposes a way to partition the search space, and generates as few candidate sequences as possible by growing on the already mined frequent sequences, and applying the apriori property as the search space is being traversed recursively looking for frequent sequences.

The early algorithms started by using projected databases, for example, FreeSpan [Han et al. 2000], Prefix Span [Pei et al. 2001].

Key features of prefix growth-based algorithm are:

1) Search Space Partitioning:

It allows partitioning of the generated search space of large candidate sequences for efficient memory management. There are different ways to partition the search space. Once the search space is partitioned, smaller partitions can be mined in parallel. Advanced techniques for search space partitioning include projected databases and conditional search, referred to as split-and-project techniques.

2) Tree Projection:

Tree projection usually accompanies pattern-growth algorithms. Here, algorithms implement a physical tree data structure representation of the search space, which is then traversed breadth-first or depth-first in search of frequent sequences, and pruning is based on the apriori property.

3) Depth-First Traversal:

That depth-first search of the search space makes a big difference in performance, and also helps in the early pruning of candidate sequences as well as mining of closed sequences [Wang and Han 2004]. The main reason for this performance is the fact that depth-first traversal utilizes far less memory, more directed search space, and thus less candidate sequence generation than breadth-first or post-order which are used by some early algorithms.

4) Candidate Sequence Pruning:

Pattern-growth algorithms try to utilize a data structure that allows them to prune candidate sequences early in the mining process. This result in early display of smaller search space and maintain a more directed and narrower search procedure.

a) FREESPAN (Frequent Pattern-Projected Sequential Pattern Mining):

FreeSpan was developed to substantially reduce the expensive candidate generation and testing of Apriori, while maintaining its basic heuristic. In general, FreeSpan uses frequent items to recursively project the sequence database into projected databases while growing subsequence fragments in each projected database.

Each projection partitions the database and confines further testing to progressively smaller and more manageable units. The trade-off is a considerable amount of sequence duplication as the same sequence could appear in more than one projected database. However, the size of each projected database usually (but not necessarily) decreases rapidly with recursion.

b) PREFIXSPAN(Prefix-Projected Sequential Pattern Mining):

This algorithm is presented by Jian Pei, Jiawei Han and Helen Pinto representing the pattern-growth methodology, which finds the frequent patterns after scanning the sequence database once. The database is then projected, according to the frequent items, into several smaller databases. Finally, the complete set of sequential patterns is found by recursively growing subsequence fragments in each projected database.

Although the PrefixSpan algorithm successfully discovered patterns employing the divide-and-conquer strategy, the cost of memory space might be high due to the creation and processing of huge number of projected sub-databases.

c) WAP-MINE:

It is a pattern growth and tree structure-mining technique with its WAP-tree structure. Here the sequence database is scanned only twice to build the WAP-tree from frequent sequences along with their support; a header table is maintained to point at the first occurrence for each item in a frequent itemset, which is later tracked in a threaded way to mine the tree for frequent sequences, building on the suffix.

The WAP-mine algorithm is reported to have better scalability than GSP and to outperform it by a margin. Although it scans the database only twice and can avoid the problem of generating explosive candidates as in apriori-based methods, WAP-mine suffers from a memory consumption problem, as it recursively reconstructs numerous intermediate WAP-trees during mining, and in particular, as the number of mined frequent patterns increases.

IV. UTILITY MINING

The traditional association rule mining and sequential pattern mining based approach normally includes only statistical measures like frequency of occurring pattern and doesn't allow users to quantify their preferences according to the usefulness of itemsets using utility values. In reality, however, some high profit products in transaction data may occur with lower frequencies.

For example jewels and diamonds are high utility products but may not occur with high frequency in a database in comparison with food or drinks. Hence high-

profit but low frequency product combinations may not be found by using traditional techniques. So the new concept called Utility Mining has been emerged in data mining which considers the individual profits and quantities of items in transaction. The itemsets with utilities being larger than or equal to a predefined threshold are then found as high utility itemsets.

Utility can be defined as “measure of how useful”(i.e. “profitable”) an item set is”. The meaning of an item set utility is interestingness, importance or profitability of an item to users. The utility can be measured in terms of cost, quantity, profit, popularity etc. For example, a computer system may be more profitable than a telephone in terms of profit. Consider the another example, If a sales analyst involved in some retail research needs to find out which item sets in the stores earn the maximum sales revenue for the stores he or she will define the utility of any item set as the monetary profit that the store earns by selling each unit of that item set.

| Algorithm | Advantages | Disadvantages |
|--|--|---|
| Apriori Based Algorithm | | |
| GSP (Generalized Sequential Pattern Mining Algorithm) | Simple and easy to learn | Computationally expensive. Requires lots of processing time. Consumes lots of memory |
| SPADE (Sequential Pattern Discovery using Equivalent Classes) | It completes the mining in three passes of database scanning. | Additional computation time is required to transform a database of horizontal layout to vertical format, which requires additional storage space. |
| SPAM (Sequential Pattern Mining using A Bitmap Representation) | It uses a vertical bitmap data layout allowing for efficient counting. | The entire algorithm with its data structures fits in main memory. |
| PREFIX GROWTH ALGORITHM | | |
| FREESPAN (Frequent pattern-projected Sequential Pattern Mining) | It mines the complete set of patterns and is efficient and runs faster than APRIORI algorithm. | It has to keep the whole sequence in the original database without length reduction. Since the growth of a subsequence is explored at any split point in a candidate sequence, it is costly. |
| PREFIXSPAN (Prefix-projected Sequential Pattern Mining) | It finds the frequent items after scanning the sequence | The cost of memory space might be high due to the creation and processing of |

| | | |
|---------|---|---|
| | database once. | huge number of projected sub-databases. |
| WAPMINE | Here the sequence database is scanned only twice to build the WAP-tree. | It suffers from a memory consumption problem, as it recursively reconstructs numerous intermediate WAP-trees during mining. |

Table 1: Comparison of Sequential Pattern Mining Algorithm

V. HIGH UTILITY SEQUENTIAL PATTERN MINING (HUS)

Algorithms for mining frequent sequences often result in many patterns being mined; most of them may not make sense to business, and those with frequencies lower than the given minimum support are filtered. This limits the actionability of discovered frequent patterns. The integration of utility into sequential pattern mining aims to solve the above problem and has only taken place very recently. Basically utility are of 2 types :Internal and External Utility. Internal utility represents the quantity of item in that pattern and external utility represent the profit associated with item. Consider the quantitative sequence database consisting of 4 sequences as shown in table 2 and profit associated with each item as shown in table 3. Terms related to utility are described as follows.

| SEQUENCE ID | SEQUENCES |
|-------------|---------------------------------|
| 1 | <a(1), {b(1), c(15), c(3)}> |
| 2 | <a(3), {a(2), c(8)}, e(2)> |
| 3 | <{a(3),b(2)}, c(2), e(2), f(3)> |
| 4 | <a(2), c(4),e(2)> |

Table 2: Quantitative Sequence Database

| Item | a | c | e | f |
|--------|---|---|---|---|
| Profit | 3 | 1 | 5 | 2 |

Table 3: Profit of Each Item

A. Utility:

The utility of an item is multiplication of external and internal utility so the utility of item a in sequence 1 is 3(3*1).

B. Sequence Utility:

It represents the total profit of that sequence. For example the calculate the sequence utility of the sequence 4 is calculated as,

$$Su(4) = (2 * 3) + (4*1) + (2*5)$$

$$= 6 + 4 + 10$$

$$= 20$$

C. Utility Of Subsequence In A Sequence:

It specifies the MAXIMUM PROFIT of that subsequence in sequence. The utility of subsequence S in a sequence is

the maximum utility values of all combinations in the sequence. For example the utility of the subsequence <ac> is calculated as,

$$U_y(ac) = \max [18(3 * 1 + 1 * 15), 17(3 * 3 + 8 * 1), 11(3 * 3 + 1 * 2), 10(2 * 3 + 1 * 4)] = 18$$

D. Limitation Of Utility Mining:

In Frequent sequential pattern mining, downward closure property serves as the foundation of pattern mining algorithms. However, this property doesn't hold in high utility pattern mining problem.

| | | | | | | |
|--------|---|---|---|---|---|---|
| Item | a | B | C | d | e | f |
| Profit | 2 | 5 | 4 | 3 | 1 | 1 |

Table 3: Quality Table

| Sid | Sequences |
|-----|--|
| 1 | <(e,5) [(c,2) (f,1)](b,2)> |
| 2 | <[(a,2)(e,6)][(a,1)(b,1)(c,2)][(a,2)(d,3)(e,3)]> |
| 3 | <(c,1)[(a,6)(d,3)(e,2)]> |
| 4 | <[(b,2)(e,2)][(a,7)(d,3)][(a,4)(b,1)(e,2)]> |
| 5 | <[(b,2)(e,3)][(a,6)(e,3)][(a,2)(b,1)]> |

Table 5: Quantitative Sequence Database

Consider For example consider table 4 and 5,

$$U_{\max}(\langle ea \rangle) = 10 + 16 + 15 = 41 \text{ but}$$

$$U_{\max}(\langle e \rangle) = 5 + 6 + 2 + 2 + 3 = 18,$$

This is lower than its super-pattern.

E. Sequence Utility Upper Bound (Suub):

This property of utility satisfies Downward Closure Property. This property can be used as a measure for pruning sequences at early stages that are not expected to qualify as high utility sequence.

For example sub (a) is calculated as,

$$su_{ub}(a) = su(1) + su(2) + su(3) = 31 + 33 + 47 + 20 = 131$$

F. Actual Sequence Utility Of Subsequence (Asu):

The Actual Sequence Utility of subsequence is the summation of the maximum utility values of subsequence in all sequences.

For example, asu of <ac> is calculated as,

$$asu_{\langle ac \rangle} = u_{1,\langle ac \rangle} + u_{2,\langle ac \rangle} + u_{3,\langle ac \rangle} + u_{4,\langle ac \rangle} = 18 + 17 + 11 + 10 = 56$$

There are various algorithms for finding high utility sequential pattern. Some of them are described as under

VI. USPAN ALGORITHM

USpan is composed of a lexicographic q-sequence tree, two concatenation mechanisms, and two pruning strategies. USpan consequently uses a depth-first search strategy to traverse the LQS-Tree to search for high utility patterns.

A. Lexicographic Q-Sequence Tree:

It is a tree structure satisfying the following rules:

- Each node in T is a sequence along with the utility of the sequence, while the root is empty.

- Any node's child is either an I-Concatenated or S-Concatenated sequence node of the node itself.
- All the children of any node in T are listed in an incremental and alphabetical order.

B. Concatenation Mechanisms:

Suppose we have a k-sequence t, we call the operation of appending a new item to the end of t to form (k+1)-sequence concatenation. If the size of t does not change, we call the operation I-Concatenation. Otherwise, if the size increases by one, we call it S-Concatenation. For example, <ea>'s I-Concatenate and S-Concatenate with b result in <e(ab)> and <eab> respectively.

For generating the children's utility based on the utility of its parent, utility matrix of a q-sequence is introduced. Each element in the matrix is a tuple; the first value shows the utility of the q-item, and the second is the utility of the remaining items in the q-sequence; known as remaining utility. Every sequence is stored in the memory in the form of a utility matrix.

C. Pruning Strategies:

Basically there are 2 pruning strategies

1) Width Pruning:

To avoid selecting the unpromising items, a strategy called width pruning strategy is developed. This is based on the sequence-weighted downward closure property (SDCP), which is similar to the transaction-weighted downward closure property (TDCP).

Suppose we have a k-sequence t, a new item i concatenates to t and results in a (k+1)-sequence t'. If <SWU(t')> \geq ξ , we say item i is a promising item to t. Otherwise i is called an unpromising item.

2) Depth Pruning Strategy:

The width pruning strategy avoids constructing unpromising patterns into the LP-Tree while a depth pruning strategy stops USpan from going deeper by identifying the leaf nodes in the tree.

So in this way USPAN mines high utility itemsets.

D. The Projection-Based Algorithm With The Pruning Strategy:

In this algorithm a quantitative sequence database (QSD) consists of sequences which include a subset of items with quantities and a minimum sequence utility threshold. For Finding the HUS from QSD we have to calculate sequential utility of each sequence and SUUB and ASU of each item. The items whose SUUB satisfies threshold forms promising only those items can make HUS patterns and the remaining items are removed from the QSD. Then recalculate the Su of modified sequences of QSD. Generate r+1 subsequence composed of the r prefix where r represents the number of items in the current set to be processed. After that calculate SUUB and ASU of that generated subsequence and check against threshold. The pattern is considered as HUS pattern if its ASU is larger than or equal to threshold.

VII. CONCLUSION

This survey paper gives detailed classification of sequential pattern mining algorithm along with their merits and demerits. As the pattern found by sequential pattern mining algorithm do not show any business value and impact, the

state of the art modification in the field of sequential pattern mining community has successfully emerged as High Utility Sequential Pattern Mining approach.

REFERENCES

- [1] Yin.j, Zheng.z, Cao.l , 2012. USpan:An Efficient Algorithm for Mining High Utility Sequential Patterns, In the 18th ACM SIGKDD international conference on knowledge discovery & data mining, 660-668.
- [2] Lan,GJ, Hong.TP, Tseng.V.S, Wang.s.l, 2012. An Improved Approach for Sequential Utility Pattern Mining. IEEE International Conference on Granular Computing, 226-230.
- [3] Lan,GJ, Hong.TP, Tseng.V.S, Wang.s.l, 2014. Applying the maximum utility measure in high utility sequential pattern mining , Expert Systems with Applications 41 Elsevier,5071-5081.
- [4] Lina.c.w, Hongb.t, Hsiang.w.l, 2011 An effective tree structure for mining high utility itemsets Expert Systems with Applications 38,7419-7424.
- [5] Mortazavi.p.j , Asi.b, Wang.t ,Chen.T, 2004. Mining Sequential Patterns by pattern –growth:The PrefixSpan approach. IEEE Transaction on knowledge & data engineering, 1424-1440.
- [6] Chand.c, Thakkar.a, Ganatra.a ,2012 Sequential Pattern Mining: Survey and Current Research Challenges” , International Journal of Soft Computing and Engineering (IJSCE) ISSN:., Volume-2, Issue-1, 2231-2307
- [7] Mabroukeh.n.r and Ezeife.c.i, 2010 A Taxonomy of Sequential Pattern Mining Algorithms, ACM Computing Surveys, Vol. 43, No. 1, Article 3, 3:1 to 3:43
- [8] Bhattacharya1.s, Dubey2.s, 2012, High Utility Itemset Mining, International Journal of Emerging Technology and Advanced Engineering (ISSN, Volume 2, Issue 8) 2250-2459.
- [9] Mooney.c and Roddick.j, Sequential Pattern Mining – Approaches and Algorithms, ACM Journal Name, Vol. V, No. N, M 20YY,1–46.
- [10]Fournier-Viger1,Wu2.c, Zida1.s,Tseng2.v,2014 FHM: Faster High-Utility Itemset Mining using Estimated Utility Co-occurrence Pruning , Proc. 21st International Symposium on methodologies for Intelligent Systems(ISMIS) Springer, LNAI, 83-92.
- [11]Lina.c, Hongb.t, Lua aDe.w, 2011,An effective tree structure for mining high utility itemsets Expert Systems with Applications 38 ,7419-7424.
- [12]Hong.TP, Lan,GJ, Wang.s.l,2009, Mining High Average-Utility Itemsets ,IEEE International Conference on Systems, Man, and Cybernetics San Antonio, TX, USA 2526-2530
- [13]Parmar.d.k., Rathod y.a.,Patel, m.m, 2013 , Survey on High utility oriented Sequential Pattern Mining , IEEE International conference on computational intelligence. 1-7.
- [14]Erwin1.a, Gopalan1.r, Achuthan2.n.r, 2007, CTU-Mine: An Efficient High Utility Itemset Mining Algorithm Using the Pattern Growth Approach, Seventh International Conference on Computer and Information Technology-IEEE.71-76.
- [15]S.Shankar', Dr.T.Purusothaman2, S.Jayanthi3, Nishanth Babu', 2009,A Fast Algorithm for Mining High Utility Itemsets, IEEE International Advance Computing Conference (IACC) 1459-1464.
- [16]Lin.c, Hong.t, Lang.g, 2010, Incrementally Mining High Utility Itemsets in Dynamic Databases , IEEE International Conference on Granular Computing 303-307.