IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. V (Mar-Apr. 2014), PP 34-40
www.iosrjournals.org

Utility Mining Algorithm for High Utility Item sets from Transactional Databases

Arati W. Borkar [1]
[1]*Department of CSE, Pune University, India*

Abstract: *The discovery of item sets with high utility like profits is referred by mining high utility item sets from a transactional database. Although in recent years a number of relevant algorithms have been proposed, for high utility item sets the problem of producing a large number of candidate item sets is incurred. The mining performance is degraded by such a large number of candidate item sets in terms of execution time and space requirement. When the database contains lots of long transactions or long high utility item sets the situation may become worse. Internet purchasing and transactions is increased in recent years, mining of high utility item sets especially from the big transactional databases is required task to process many day to day operations in quick time. There are many methods presented for mining the high utility item sets from large transactional datasets are subjected to some serious limitations such as performance of this methods needs to be investigated in low memory based systems for mining high utility itemsets from large transactional datasets and hence needs to address further as well. Another limitation is these proposed methods cannot overcome the screenings as well as overhead of null transactions; hence, performance degrades drastically. During this paper, we are presenting the new approach to overcome these limitations. We presented distributed programming model for mining business-oriented transactional datasets by using an improved Map Reduce framework on Hadoop, which overcomes not only the single processor and main memory-based computing, but also highly scalable in terms of increasing database size. We have used this approach with existing UP-Growth and UP-Growth+ with aim of improving their performances further. In experimental studies we will compare the performances of existing algorithms UP-Growth and UP-Growth+ against the improve UP-Growth and UP-Growth+ with Hadoop.*

Keywords: *Dataset Mining, Hadoop, Itemsets, MapReduce Framework, Transactional Dataset, UP-Growth, UP-Growth+.*

I. Introduction

Data Mining refers to extracting or mining knowledge from large amounts of data. In large databases finding of frequent patterns task is very important use full in many applications over the past few years. The primary goal is to discover hidden patterns, unexpected trends in the data. Data mining is concerned with analysis of large volumes of data to automatically discover interesting regularities or relationships which in turn leads to better understanding of the underlying processes. Data mining activities uses combination of techniques from database artificial intelligence, statistics, technologies machine learning . Utility Mining is one of the most challenging data mining tasks is the mining of high utility itemsets efficiently. Identification of the itemsets with high utilities is called as Utility Mining.

A Association rules mining (ARM) [1] is one of the most widely used techniques in data mining and knowledge discovery and has tremendous applications like business, science and other domains. Make the decisions about marketing activities such as, e.g., promotional pricing or product placements. A high utility itemset is defined as: A group of items in a transaction database is called itemset. This itemset in a transaction database consists of two aspects: First one is itemset in a single transaction is called internal utility and second one is itemset in different transaction database is called external utility. The transaction utility of an itemset is defined as the multiplication of external utility by the internal utility. By transaction utility, transaction weight utilizations (TWU) can be found. To call an itemset as high utility itemset only if its utility is not less than a user specified minimum support threshold utility value; otherwise itemset is treated as low utility itemset.

To generate these high utility itemsets mining recently in 2010, UP - Growth (Utility Pattern Growth) algorithm [2] was proposed by Vincent S. Tseng et al. for discovering high utility itemsets and a tree based data structure called UP - Tree (Utility Pattern tree) which efficiently maintains the information of transaction database related to the utility patterns. Four strategies (DGU, DGN, DLU, and DLN) used for efficient construction of UP - Tree and the processing in UP - Growth [11]. By applying these strategies, can not only efficiently decrease the estimated utilities of the potential high utility itemsets (PHUI) but also effectively reduce the number of candidates.

But this algorithm takes more execution time for phase II (identify local utility itemsets) and I/O cost.

Efficient discovery of frequent itemsets in large datasets is a crucial task of data mining. In recent years, several approaches have been proposed for generating high utility patterns; they arise the problems of producing a large number of candidate itemsets for high utility itemsets and probably degrade mining performance in terms of speed and space. Mining high utility itemsets from a transactional database refers to the discovery of itemsets with high utility like profits. Although a number of relevant approaches have been proposed in recent years, they incur the problem of producing a large number of candidate itemsets for high utility itemsets. Such a large number of candidate itemsets degrades the mining performance in terms of execution time and space requirement. The situation may become worse when the database contains lots of long transactions or long high utility itemsets.

Existing studies applied overestimated methods to facilitate the performance of utility mining. In these methods, potential high utility itemsets (PHUIs) are found first, and then an additional database scan is performed for identifying their utilities. However, existing methods often generate a huge set of PHUIs and their mining performance is degraded consequently. This situation may become worse when databases contain many long transactions or low thresholds are set. The huge number of PHUIs forms a challenging problem to the mining performance since the more PHUIs the algorithm generates, the higher processing time it consumes.

To provide the efficient solution to mine the large transactional datasets, recently improved methods presented in [1]. In [1], authors presented propose two novel algorithms as well as a compact data structure for efficiently discovering high utility itemsets from transactional databases. Experimental results show that UP-Growth and UP-Growth+ outperform other algorithms substantially in terms of execution time. But these algorithms further needs to be extend so that system with less memory will also able to handle large datasets efficiently. The algorithms presented in [1] are practically implemented with memory 3.5 GB, but if memory size is 2 GB or below, the performance will again degrade in case of time. In this project we are presenting new approach which is extending these algorithms to overcome the limitations using the MapReduce framework on Hadoop.

II. LITERATURE SURVEY

In this section we are presented the review of different methods presented for mining high utility itemsets from the transactional datasets.

- R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," [3] as they discussed a well-known algorithms for mining association rules is Apriori, which is the pioneer for efficiently mining association rules from large databases.

Cai et al. and Tao et al. first proposed the concept of weighted items and weighted association rules [5]. However, since the framework of weighted association rules does not have downward closure property, mining performance cannot be improved. To address this problem, Tao et al. proposed the concept of weighted downward closure property [12]. By using transaction weight, weighted support can not only reflect the importance of an itemset but also maintain the downward closure property during the mining process.

- Liu et al. proposed an algorithm named Two- Phase [8] which is mainly composed of two mining phases. In phase I, it employs an Apriori-based level-wise method to enumerate HTWUIs. Candidate itemsets with length k are generated from length k-1 HTWUIs, and their TWUs are computed by scanning the database once in each pass. After the above steps, the complete set of HTWUIs is collected in phase I. In phase II, HTWUIs that are high utility itemsets are identified with an additional database scan. Ahmed et al. [13] proposed a tree-based algorithm, named IHUP. A tree based structure called IHUP-Tree is used to maintain the information about itemsets and their utilities.

Each node of an IHUP-Tree consists of an item name, a TWU value and a support count. IHUP algorithm has three steps: 1) construction of IHUP-Tree, 2) generation of HTWUIs, and 3) identification of high utility item sets.

In step 1, items in transactions are rearranged in a fixed order such as lexicographic order, support descending order or TWU descending order. Then the rearranged transactions are inserted into an IHUP-Tree.

- In the framework of frequent itemset mining, the importance of items to users is not considered. Thus, the topic called weighted association rule mining was brought to attention [4], [26], [28], [31], [37], [38], [39].
- Cai et al. first proposed the concept of weighted items and weighted association rules [4].
- However, since the framework of weighted association rules does not have downward closure property, mining performance cannot be improved. To address this problem, Tao et al. proposed the concept of weighted downward closure property [28].
- There are also many studies [6], [26], [37] that have developed different weighting functions for weighted pattern mining.Survey on The MapReduce Framework for Handling Big Datasets [21]

Google's MapReduce [25] was first proposed in 2004 for massive parallel data analysis in shared-nothing clusters. Literature [26] evaluates the performance in Hadoop/HBase for Electroencephalogram (EEG) data and

saw promising performance regarding latency and throughput. Karim et al. [27] proposed a Hadoop/MapReduce framework for mining maximal contiguous frequent patterns (which was first introduced at literature in RDBMS/single processor-main memory based computing) from the large DNA sequence dataset and showed outstanding performance in terms of throughput and scalability [21].

Literature [28] proposes a MapReduce framework for mining-correlated, associated-correlated and independent patterns synchronously by using the improved parallel FP-growth on Hadoop from transactional databases for the first times ever. Although it shows better performance, however, it also did not consider the overhead of null transactions. Woo et al. [29], [30], proposed market basket analysis algorithm that runs on Hadoop based traditional Map Reduce framework with transactional dataset stored on HDFS. This work presents a Hadoop and HBase schema to process transaction data for market basket analysis technique. First it sorts and converts the transaction dataset to <key, value> pairs, and stores the data back to the HBase or HDFS. However, sorting and grouping of items then storing back it to the original nodes does not take trivial time. Hence, it is not capable to find the result in a faster way; besides this work also not so useful to analyze the complete customer's preference of purchase behavior or rules [21].

III. PROPOSED APPROACH FRAMEWORK AND DESIGN

In the literature we have studied the different methods proposed for high utility mining from large datasets. But all this methods frequently generate a huge set of PHUIs and their mining performance is degraded consequently. Further in case of long transactions in dataset or low thresholds are set, then this condition may become worst. The huge number of PHUIs forms a challenging problem to the mining performance since the more PHUIs the algorithm generates, the higher processing time it consumes. Thus to overcome this challenges the efficient algorithms presented recently in [1]. These methods in [1] outperform the state-of-the-art algorithms almost in all cases on both real and synthetic data set. However this approach in [1] is still needs to be improved in case of less memory based systems.

major contributions of this work are summarized as follows:

1. A novel algorithm, called ***UP-Growth* (*Utility Pattern Growth*)**, is proposed for discovering high utility itemsets. Correspondingly, a compact tree structure, called ***UP-Tree* (*Utility Pattern Tree*)**, is proposed to maintain the important information of the transaction database related to the utility patterns. High utility itemsets are then generated from the UP-Tree efficiently with only two scans of the database.
2. Four strategies are proposed for efficient construction of **UP-Tree** and the processing in **UP-Growth**. By these strategies, the estimated utilities of candidates can be well reduced by discarding the utilities of the items which are impossible to be high utility or not involved in the search space. The proposed strategies can not only efficiently decrease the estimated utilities of the potential high utility itemsets but also effectively reduce the number of candidates.
3. Both of synthetic and real datasets are used in experimental evaluations to compare the performance of **UP-Growth** with the state-of-the-art utility mining algorithms. The experimental results show that **UP-Growth** outperforms other algorithms substantially in terms of execution time, especially when the database contains lots of long transactions.

3.1 Proposed Work

When data sets go beyond a single storage capacity, it is necessary to distribute them to multiple independent computers. Trans-computer network storage file management system is called distributed file system. A typical Hadoop distributed file system contains thousands of servers, each server stores partial data of file system.

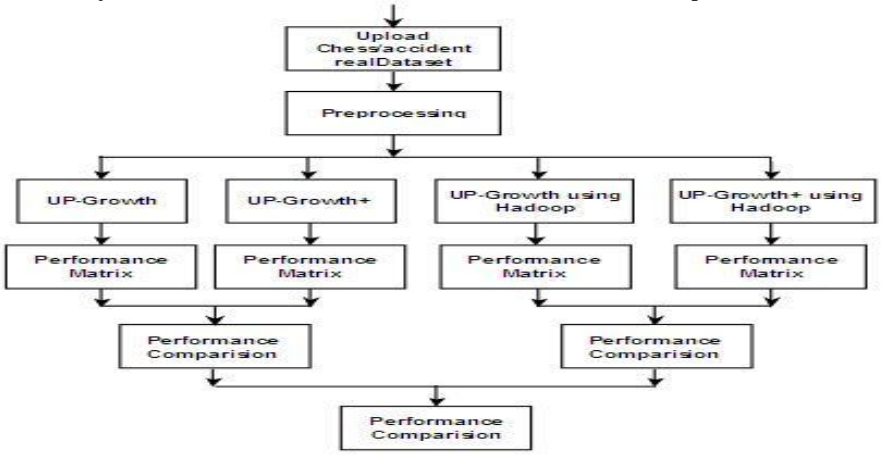

Figure 1: System architecture

3.2Aims and Objectives

In this project we have main aim is to present improved methods UP-Growth and UP-Growth+ with aim of improving its performance in terms of scalability and time:

- To present literature review different methods of frequent set mining over transactional datasets.
- To present the present new framework and methods.
- To present the practical simulation of proposed algorithms and evaluate its performances.
- To present the comparative analysis of existing and proposed algorithms in order to claim the efficiency.

3.3 Map Reduce Overview.

In distributed data storage, when parallel processing the data, we need to consider much, such as synchronization, concurrency, load balancing and other details of the underlying system. It makes the simple calculation become very complex. MapReduce programming model was proposed in 2004 by the Google, which is used in processing and generating large data sets implementation. This framework solves many problems, such as data distribution, job scheduling, fault tolerance, machine to machine communication, etc. MapReduce is applied in Google's Web search. Programmers need to write many programs for the specific purpose to deal with the massive data distributed and stored in the server cluster, such as crawled documents, web request logs, etc., in order to get the results of different data, such as inverted indices, web document, different views, worms collected the number of pages for each host a summary of a given date within the collection of the most common queries and so on.

We use Two algorithms, named *utility pattern growth* (UPGrowth) and UP-Growth+, and a compact tree structure, called *utility pattern tree* (UP-Tree), for high utility itemsets discovery and to maintain important information related to utility patterns within databases.scheduling, fault tolerance, machine to machine communication, etc. MapReduce is applied in Google's Web search. Programmers need to write many programs for the specific purpose to deal with the massive data distributed and stored in the server cluster, such as crawled documents, web request logs, etc., in order to get the results of different data, such as inverted indices, web document, different views, worms collected the number of pages for each host a summary of a given date within the collection of the most common queries and so on.

We use Two algorithms, named *utility pattern growth* (UPGrowth) and UP-Growth+, and a compact tree structure, called *utility pattern tree* (UP-Tree), for high utility itemsets discovery and to maintain important information related to utility patterns within databases.

UP-Growth algorithm

Input: UP-Tree , Header Table , minimum utility threshold , Item set = { $_1$, $_2$, . . , }.

Process:

1. For each entry in do
2. Trace links of each item. And calculate sum of node utility .
3. If ≥ t
4. Generate Potential High Utility Itemset (PHUI)

= ∪

5. Put Potential Utility of as approximated utility of
6. Construct Conditional Pattern Based .
7. Put local promising items into .
8. Apply Discarding Local Unpromising (DLU) to minimize path utilities of paths.
9. Apply DLU with _ _ h to insert path into .
10. If ≠ ∅ then call to UP-Growth.
11. End if
12. End for.

Output: All PHUI's in

UP-Growth+ algorithm

UP-Growth+ algorithm

In UP-Growth, minimum item utility table is used to reduce theoverestimated utilities. In UP-Growth+algorithm we replaceDiscarding Local Unpromising (DLU) with Discarding NodeUtility (DNU), DLN is replace with Decreasing local Node utilities for the nodes of local UP-Tree (DNN) and $Insert\_Recognized\_Path$ is replace by$Insert_Recognized_Pathmiu$ In the data mining process, when a path is retrieved, minimal node utility of each node in the path is also retrieved. Thus, we can simply replace minimum item utility with minimal node utility as follows,

Assume, p is the path in item $im - CPB$ and $UI\ im - CPB$ is the set of unpromising items in $im - CPB$. The path utility of p in $im - CPB$, i.e.,$(p, im - CPB)$, $im - CPB$, is recalculated as:

$pu\ p, im - CPB$
$= p.\ im\ .\ nu$
$- miu\ i \times p.\ count$
$\forall i \in UI\{im\} - CPB \wedge i \subseteq p$

Where $p.\ count$ is the support count of p in $im - CPB$.
Assume, a reorganized path $p = < N'i1\ , N'i2\ , \ldots\ , N'im >$ in $im - CPB$ inserted into $< N'i1\ , N'i2\ , \ldots\ , N'im >$ path in $im - tree$. Thus node utility of item node is recalculated as:

$Nik\ .\ nunew = Nik\ .\ nuold + pu\ p, im - CPB$
$- miu\ ij \times p.\ count$
m'
$j=k+1$

Where $N\ .\ nuold$ is the node utility of Nik in $im - tree$before adding p.

IV. Experimental Evaluation

In this section the Performance of the proposed algorithms evaluated[10]. The experiments were done on a 2.80 GHz Intel Pentium D Processor with 3.5 GB memory. The operating system is Microsoft Windows 7. The algorithms are implemented in Java language. In the experiments Both real (Table. 5) and synthetic datasets (Table. 6) are used.

Table 1 Parameter Settings of Synthetic Datasets

Parameter Descriptions	Default
|D|: Total number of transactions	100K
T: Average transaction length	10
|I|: Number of distinct items	1000
F: Average size of maximal potential frequent itemsets	6
Q: Maximum number of purchased items in Transactions	10

Table 2 : Characteristics of Real Datasets

Dataset	|D|	T	|I|	Type
Chain-store	1,112,949	7.2	46,086	Sparse
Chess	3,196	37.0	75	Dense

a) Evaluation on Real datasets In this part, on three real datasets we show the performance comparison is done: Chess and Chain-store. First, Chess and Chain-store we show the result in Fig. 2. In Fig. 2 (a), we the runtime of IHUPT&FPG is the worst, followed by UPT&FPG, UPT&UPG and UPT&UPG+ is the best. In Fig. 3 experimental results on real sparse datasets are shown. In Fig. 3 (a) and (b) performance on Chain-store dataset is shown. In Fig. 3 (a), the runtime of IHUPT&FPG is the worst, followed by UPT&FPG, UPT&UPG and UPT&UPG+ is the best. As more candidates are generated the performance of IHUPT&FPG is the worst. Among the three methods the execution timeof UPT&FPG is the worst since UP Growth+ and UP-Grow the efficiently prune the search space of local UP-Trees. We can observe that by comparing the performance of previous method to the performanceOf proposed methods substantially out performs.

In Fig. 4 Experimental results of phase II are shown. Runtime for phase II is very long for large databases such as Chain-store so we only show the result of chess. We can observe in Fig. 4, that the runtime for phase II is not only proportional to number of candidates in phase II but also increases fiercely. Therefore in phase II the performance is highly dependent on the runtime, since the overhead of scanning databases is huge.

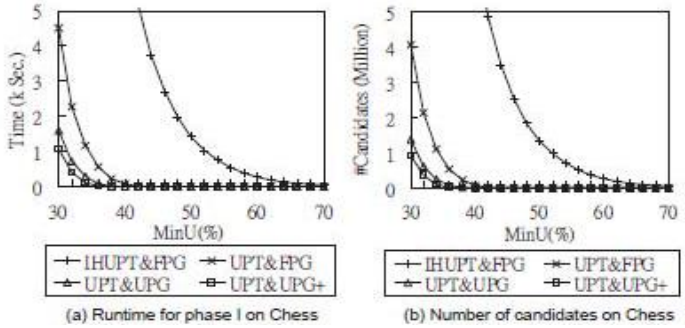

Figure 2: Performance Comparison on Dense Dataset

In this part, we show the performance comparison on three real data sets: dense data set Chess and sparse data sets Chain-store and Foodmart. First, we show the results on real dense data set Chess in Fig. 7.1. In Fig. 7.1a, we can observe that the performance of proposed methods substantially outperforms that of previous methods. The runtime of IHUPT&FPG is the worst, followed by UPT&FPG, UPT& UPG, and UPT&UPG+ is the best. The main reason is the performance of IHUPT&FPG and UPT&FPG is decided by the number of generated candidates. In Fig. 7.1, runtime of the methods is just proportional to their number of candidates, that is, the more candidates the method produces, the greater its execution time.

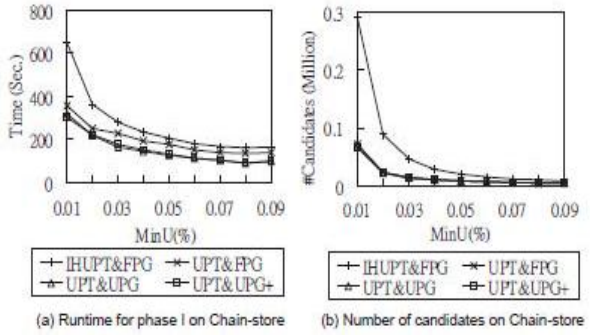

Figure 3 : Performance Comparison on Sparse Datasets

Experimental results on real sparse data sets are shown in Fig. 7.2. The performance on Chain-store data set is shown in Figs. 10a and 10b. In Fig. 7.2a, the runtime of IHUPT&FPG is the worst, followed by UPT&FPG, UPT& UPG, and UPT&UPG+ is the best. The performance of IHUPT&FPG is the worst since it generates the most candidates. Besides, although the number of candidates of UPT&FPG, UPT&UPG, and UPT&UPG+ are almost the same, the execution time of UPT&FPG is the worst among the three methods since UP-Growth+ and UP-Growth efficiently prune the search space of local UP-Trees.

4.1Expected Results

Practical work done is as shown in figure given below. Following figure shows the graphical representation of time verses algorithms. Performance is computed according to the time required for set of transactions.

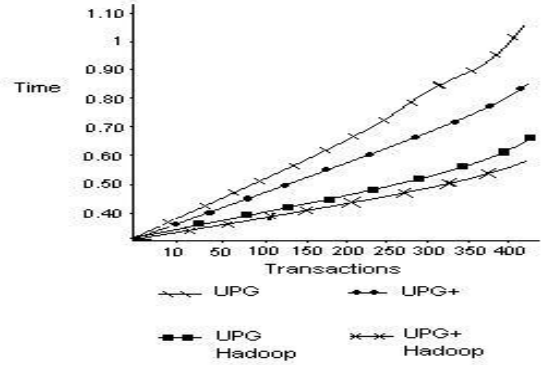

Figure 2: Performance comparison.

Experimental results show that UP-Growth and UP-Growth+ outperform other algorithms

substantially in terms of execution time. But these algorithms further needs to be extend so that system with less memory will also able to handle large datasets efficiently. The algorithms presented in [1] are practically implemented with memory 3.5 GB, but if memory size is 2 GB or below, the performance will again degrade in case of time. In this project we are presenting new approach which is extending these algorithms to overcome the limitations using the MapReduce framework on Hadoop.

V. Conclusion And Future Work

In this paper, we have proposed two efficient algorithms named UP-Growth and UP-Growth+ for mining high utility itemsets from transaction databases. For maintaining the information of high utility itemsets a data structure named UP-Tree was proposed. With only two database scans, from UP-Tree Potential high utility itemsets can be efficiently generated. To perform a thorough performance evaluation both real and synthetic datasets were used in the experiments. Results show that the strategies considerably improved performance by reducing both the search space and the number of candidates. we have proposed two efficient algorithms named UP-Growth and UP-Growth+ for mining high utility itemsets from transaction databases. For maintaining the information of high utility itemsets a data structure named UP-Tree was proposed. With only two database scans, from UP-Tree Potential high utility itemsets can be efficiently generated. To perform a thorough performance evaluation both real and synthetic datasets were used in the experiments. Results show that the strategies considerably improved performance by reducing both the search space and the number of candidate.

Acknowledgement

We the author gives special thanks to Dr. S. T. Singh sir to contributed this paper and Fellow for their valuable comments and sharing their knowledge.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. of the 20th Int'l Conf. on Very Large Data Bases, pp. 487-499, 1994.
[2] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. Efficient tree structures for high utility pattern mining in incremental databases. In IEEE Transactions on Knowledge and Data Engineering, Vol. 21, Issue 12, pp. 1708-1721, 2009.
[3] R. Chan, Q. Yang, and Y. Shen. Mining high utility itemsets. In Proc. of Third IEEE Int'l Conf. on Data Mining, pp. 19-26, Nov., 2003.
[4] A. Erwin, R. P. Gopalan, and N. R. Achuthan. Efficient mining of high utility itemsets from large datasets. In Proc. of PAKDD 2008, LNAI 5012, pp. 554-561.
[5] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.
[6] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. In Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, Jan., 2008.
[7] Y. Liu, W. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In Proc. of the Utility-Based Data Mining Workshop, 2005.
[8] B.-E. Shie, V. S. Tseng, and P. S. Yu. Online mining of temporal maximal utility itemsets from data streams. In Proc.of the 25th Annual ACM Symposium on Applied Computing, Switzerland, Mar., 2010.
[9] H. Yao, H. J. Hamilton, L. Geng, A unified framework for utility-based measures for mining itemsets. In Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining, pp. 28-37, USA, Aug., 2006.
[10] Sadak Murali & Kolla Morarjee" A Novel Mining Algorithm for High Utility Itemsets from Transactional Databases" Volume 13 Issue 11 Version 1.0 Year 2013 Online ISSN: 0975-4172 & Print ISSN: 0975-4350.
[11] Y. Liu, W. Liao, and A. Choudhary, "A Fast High Utility Itemsets Mining Algorithm," Proc. Utility-Based Data Mining Workshop, 2005.
[12] F. Tao, F. Murtagh, and M. Farid, "Weighted Association Rule Mining Using Weighted Support and Significance Framework," Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '03), pp. 661-666, 2003.
[13] J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," Proc. 21th Int'l Conf. Very Large Data Bases, pp. 420-431, Sept. 1995.