IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 4, Ver. VI (Jul.-Aug. 2016), PP 68-74
www.iosrjournals.org

Efficient Algorithm for Mining High Utility Itemsets from Large Datasets Using Vertical Approach

S. Renu Deepti(Assistant prof), B. Srivani(Assistant prof)
CSE,VNRVJIET,Hyderabad, India
CSE,VNRVJIET, Hyderabad, India

Abstract: *High Utility Itemset Mining is a challenging task as the Downward Closure Property present in frequent itemset mining does not hold here. In recent times many algorithms have been proposed for mining high utility itemsets ,but most of them follow a two- phase horizontal approach in which candidate itemsets are generated first and then the actual high utility itemsets are mined by performing another database scan. This approach generates a large number of candidate itemsets which are not actual high utility itemsets thus causing memory and time overhead to process them. To overcome this problem we propose a single phase algorithm which uses vertical database approach. Exhaustive search can mine all the high utility itemsets but it is expensive and time consuming. Two strategies based on u-list structure and item pair co- existence map are used in this algorithm for efficiently pruning the search space to avoid exhaustive search. Experimental analysis over various databases show that the proposed algorithm outperforms the two-phase algorithms UP-Growth, UP-Growth+ and IHUP in terms of running times and memory consumption.*

I. Introduction

Recent advances in database facilities led to the increased use of databases by many organizations leading to storage of large data. Extraction of knowledge and information from this data is a developing area of research .

Frequent itemset mining is identifying set if items whose count in the transaction database is greater than a predefined minimum value. Frequent itemset mining is identifying set of items whose count in the transaction database is greater than a predefined minimum value. Frequentitemset mining follows *downward closureproperty*. According to this property if an itemset is infrequent then all the supersets of that itemset are also infrequent so it is not required to check the supersets of the infrequent itemsets thus preventing checking all the itemsets. But frequent itemset mining doesn't take into account the profit/utility of each item and the importance of each item in a transaction. So the high utility itemset mining is used to discover itemsets with high utility. But the downward closure property which is used as for pruning infrequent itemsets does not hold in high utility itemset mining. So mining high utility itemsets is a complex task.

1.1 Data Mining

Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses.

Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, previously unknown and potentially useful patterns in data. These patterns are used to make predictions or classifications about new data, explain existing data, summarize the contents of a large database to support decision making and provide graphical data visualization to aid humans in discovering deeper patterns. Data mining is the process of revealing nontrivial, previously unknown and potentially useful information from large databases. Discovering useful patterns hidden in a database plays an essential role in several data mining tasks, such as frequent pattern mining, weighted frequent pattern mining, and high utility pattern mining. Among them, frequent pattern mining is a fundamental research topic that has been applied to different kinds of databases, such as transactional databases, bioinformatics, Web click-stream analysis, and mobile environments. In view of this, utility miningemerges as an important topic in data mining field. Mining high utility itemsets from databases refers to finding the itemsets with high profits.

1.2 Utility Mining

In frequent itemsets mining, relative importance of each item is not considered. To address this problem, weighted association rule mining was proposed. In this framework, weights of items, such as unit profits of items in transaction databases, are considered. Mining high utility itemsets from databases refers to finding the itemsets with high profits. Here, the meaning of itemset utility is interestingness, importance, or profitability of an item to users.

Utility of items in a transaction database consists of two aspects:

- The importance of distinct items, which is called ***external utility***, and
- The importance of items in transactions, which is called ***internal utility***.

Utility of an itemset is defined as the product of its external utility and its internal utility. An itemset is called a ***high utility itemset*** if its utility is no less than a user-specified minimum utility threshold; otherwise, it is called a low-utility itemsets. Mining high utility itemsets from databases is an important task and has a wide variety of applications such as business promotion in chain hypermarkets, cross-marketing in retail stores. However, mining high utility itemsets from databases is not an easy task since ***downwardclosure property*** in frequent itemset mining does not hold. In otherwords, pruning search space for high utility itemset mining is difficult because a superset of a low-utility itemset may be a high utility itemset. A naive method to address this problem is to enumerate all itemsets from databases by the principle of exhaustion. Obviously, this method suffers from the problems of a large search space, especially when databases contain lots of long transactions or a low minimum utility threshold is set. Hence, how to effectively prune the search space and efficiently capture all high utility itemsets with no miss is a crucial challenge in utility mining.

Most of the high utility mining algorithms 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. Existing methods often generate a huge number of potential high utility itemsets and their mining performance is degraded consequently. The 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.

1.3 Problem Statement:

Given a transaction database D and a user-specified minimum utility threshold min_util, the problem of mining high utility itemsets from D is to find the complete set of the itemsets whose utilities are larger than or equal to min_util.

1.4 Proposed method

The proposed algorithms for mining high utility itemsets from large static datasets is given which uses a vertical approach for mining process. It is a single phase approach as opposed to the all the state-of-art algorithms which are two-phase. In chapter 4 efficient algorithm for high utility itemsets from dynamic databases is proposed which uses an incremental strategy for mining new set of high utility itemsets without scanning the original database again.

II. Related Work

To address the problem of high utility itemsets mining, an algorithm named ***Two Phase*** algorithm was proposed composing 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.

Although two-phase algorithm reduces search space by using TWDC property, it still generates too many candidates to obtain HTWUIs and requires multiple database scans. To overcome this problem, an isolated items discarding strategy to reduce the number of candidates was proposed. By pruning isolated items during level-wise search, the number of candidate itemsets for HTWUIs in phase I can be reduced. However, this algorithm still scans database for several times and uses a candidate generation-and-test scheme to find high utility itemsets.

To efficiently generate HTWUIs in phase I and avoid scanning database too many times, a tree-based algorithm, named ***IHUP*** wasproposed. 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 itemsets.

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 step 2, HTWUIs are generated from the IHUP-Tree by applying FP-Growth. Thus, HTWUIs in phase I can be

found without generating any candidate for HTWUIs. In step 3, high utility itemsets and their utilities are identified from the set of HTWUIs by scanning the original database once.

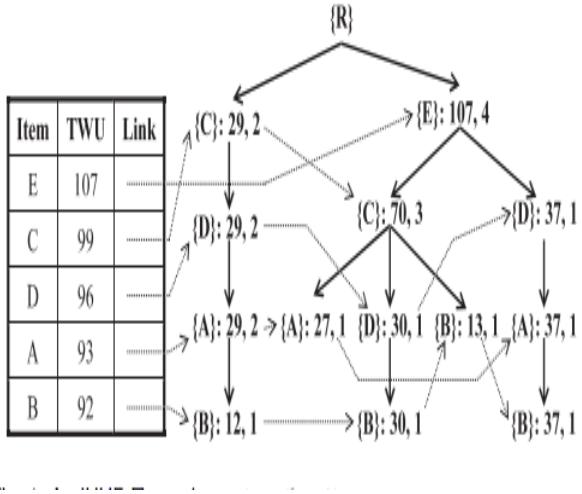

Fig.1. An IHUP-Tree when min_util = 40

Although IHUP achieves a better performance than IIDS and Two-Phase, it still produces too many HTWUIs in phase I. Note that IHUP and Two-Phase produce the same number of HTWUIs in phase I since they both use TWU framework to overestimate itemsets' utilities. However, this framework may produce too many HTWUIs in phase I since the overestimated utility calculated by TWU is too large. Such a large number of HTWUIs will degrade the mining performance in phase I substantially in terms of execution time and memory consumption. Moreover, the number of HTWUIs in phase I also affects the performance of phase II since the more HTWUIs the algorithm generates in phase I, the more execution time for identifying high utility itemsets it requires in phase II.

As stated above, the number of generated HTWUIs is a critical issue for the performance of algorithms. By applying the given strategies, the number of generated candidates can be highly reduced in phase I and high utility itemsets can be identified more efficiently in phase II.

2.1 UP-Tree

Mining of high utility itemsets using UP-Tree algorithm consists of three steps:

1. Scanning the database twice to construct the global UP-Tree using first two strategies (given below)
2. Generating potentially high utility itemsets recursively from global UP-Tree and local UP-Trees by UP-Growth using third and fourth strategies
3. Identifying actual high utility itemsets from the set of potential high utility itemsets

2.1.1 Data structure

Here a compact tree structure called UP-Tree is used to maintain the information of transactions and high utility itemsets which facilitates the mining performance and avoids scanning of the original database repeatedly.

Each node *N* of UP-Tree contains *N.name, N.count, N.nu, N.parent, N.hlink* and a set of child nodes where *N.name* is the node's item name, *N.count* is node's support count, *N.nu* is node's node utility or overestimated utility of the node. *N.parent* is the parent node of the node and *N.hlink* is a node link which points to a node whose item name is the same as *N.name*. There exists a table called *header table* for traversing the UP-Tree. By following the links in header table and the nodes in UP-Tree, the nodes with same names can be traversed efficiently. Each entry of a header table records an item name, an overestimated utility and a link which points to the last occurrence of the node which has the same item as the entry in the UP-Tree.

2.1.2 Strategy 1: Discarding Global Unpromising Items during Constructing a Global UP-Tree (DGU)

The global UP-Tree is constructed in two scans of the original database. TU of each transaction is computed in the first scan. Also TWU of each single item is calculated. By (Transaction-weighted downward closure.) property, which says for any itemset X, if X is not a HTWUI, any superset of X is a low utility itemset, an item and its supersets are unpromising to be high utility itemsets if its TWU is less than the minimum utility threshold. Such item is called *unpromising item.*

When a transaction is retrieved from the database during the second scan, the unpromising items should be removed from the transaction and their utilities should be subtracted from the transaction's TU. During the second scan transactions are also inserted into a UP-Tree. This is the first strategy.

Strategy 1. DGU: Discarding global unpromising items and their actual utilities from transactions and transaction utilities of the database.

Thus, when utilities of itemsets are being estimated, utilities of unpromising items can be regarded as irrelevant and can be discarded.

After pruning unpromising items, new TU's are called reorganized transaction utilities or RTU. Since the utilities of unpromising items are excluded, RTU must not be larger than TWU. By reorganizing the transactions, not only less information is needed to be recorded in UP-Tree, but also smaller overestimated utilities for itemsets are generated. Strategy DGU uses RTU to overestimate the utilities of itemsets instead of TWU. DGU is quite effective especially when transactions contain lots of unpromising items, such as those in sparse data sets. Moreover, DGU can be easily integrated into TWU-based algorithms. Also DGU can be performed repeatedly till all reorganized transactions no global unpromising items. Though it may need several database scans, by performing DGU for several times number of potential high utility itemsets can be reduced.

2.1.3 Strategy 2 : Decreasing Global Node Utilities during Constructing a Global UP-Tree

It is known that the tree-based framework for high utility itemset mining applies the divide-and-conquer technique in mining processes, thus the search space can be divided into smaller subspaces.

The second strategy for decreasing overestimated utilities is to remove the utilities of descendant nodes from their node utilities in global UP-Tree. This process is performed during the construction of the global UP-Tree.

Strategy 2. DGN: Decreasing global node utilities for the nodes of global UP-Tree by actual utilities of descendant nodes during the construction of global UP-Tree.

The utilities of the nodes that are closer to the root of a global UP-Tree are further reduced by applying strategy DGN. DGN is suitable for the databases containing lots of long transactions. That is, the more items a transaction contains, the more utilities can be discarded by DGN.

2.1.4 Constructing a Global UP-Tree by Applying DGU and DGN

As discussed above, constructing a global UP-Tree needs two scans of the original database. In the first scan, each transaction's TU and each item's TWU are calculated. By doing this we can get promising as well as unpromising itemsets. DGU is then applied after getting all the promising items. The transactions are reorganized by pruning the unpromising items and sorting the remaining promising items in a fixed order. Each transaction after the above reorganization is called a *reorganized transaction*.

Then a function Insert_Reorganized_Transaction is called to apply DGN during constructing a global UP-Tree. After a transaction has been reorganized, it is inserted into the global UP-Tree. After all the reorganized transactions are inserted, a global UP-Tree is formed. Node utilities of the nodes in UP-Tree are less than those in other tree based algorithms since the node utilities are effectively decreased by the two strategies DGU and DGN.

2.1.5 Mining Method: UP-Growth

A basic method of generating potentially high utility itemsets, is by mining global UP-Tree by FP-Growth after the construction of global UP-Tree. As too many candidates will be generated, UP-Growth has proposed two more strategies. By applying these strategies, overestimated utilities of itemsets can be decreased and thus the number of potential high utility itemsets can be further reduced.

2.1.5.1 Strategy 3: Discarding Local Unpromising Items during Constructing a Local UP-Tree (DLU)

For the two strategies, a minimum item utility table is maintained to keep minimum item utilities for all global promising items in the database. Minimum item utilities can be collected during the first scan of original database. Minimum item utilities are utilized to reduce utilities of local unpromising items in conditional pattern bases instead of exact utilities. An estimatedvalue for each local unpromising item is subtracted from the path utility of an extracted path.

Strategy 3. DLU: Discarding local unpromising items and their estimated utilities from the paths and path utilities of conditional pattern bases. DLU can be recognized as local version of DGU. It provides a simple but useful schema to reduce overestimated utilities locally without an extra scan of original database.

2.1.5.2 Strategy 4: Decreasing Local Node Utilities during Constructing a Local UP-Tree (DLN)

The same as DLU, DLN can be recognized as local version of DGN. By the two strategies, overestimated utilities for itemsets can be locally reduced in a certain degree without losing any actual high utility itemset.

The process of mining potential high utility itemsets by UP-Growth is done as follows: First, the node links in UP-Tree corresponding to the item, which is the bottom entry in header table, are traced. Found nodes are traced to root of the UP-Tree to get paths related to the item. All retrieved paths, their path utilities and support counts are collected into the item's conditional pattern base.

A conditional UP-Tree can be constructed by two scans of a conditional pattern base. In the first scan, local promising and unpromising items are found by summing the path utility for each item in the conditional pattern base. Then, DLU is applied to reduce overestimated utilities during the second scan of the conditional pattern base.

Then the path is reorganized by the descending order of path utility of the items in the conditional pattern base.

DLN is applied during inserting reorganized paths into a conditional UP-Tree. Then the potential high utility itemsets are found by recursively calling the procedure of UP-Growth

III. Mining High Utility Itemsets From Large Static Datasets

To overcome the problems faced by existing two-phase algorithms, we propose a single phase algorithm which discovers all high utility itemsets using two pruning strategies based on u-lists structure and item pair co-existence map. These pruning strategies are used to efficiently prune the itemsets in the search space which is otherwise exponentially high due to all the possible enumerations of items in the database.

In the first step, Transaction Weighted Utility of each item and Transaction Utility of each transaction is calculated. The transactions are then reorganized by removing the items with utility less than min_util and by arranging remaining items in the ascending order their Transaction Weighted Utility.

Tid	Item	Util.	Item	Util.	item	Util.	item	Util.	item	Util.	TU
T1	c	2	b	2	d	5					9
T2	e	4	c	3	b	2	a	4	d	5	18
T3	c	2	a	4	d	5					11
T4	e	4	c	2							6
T5	e	8	b	4	a	5	d	5			22
T6	c	1	b	8	a	3					12
T7	d	5									5

Fig. 2.Reorganized transactions

The U-List for each item is then constructed. First the U-Lists for all the 1-itemsets with Transaction weighted utility greater than *min_util* are constructed. Then U-Lists for 2-itemsets of the form {pq} are constructed from U-Lists of 1-itemsets {p} and {q}.

3.1 Strategy 1

U-List for k+1 itemsets are formed only if sum of its iu's and ru's in the U-List of its corresponding k-itemsets is greater than or equal to min_util i.e., if sum of iu's and ru's in the U-List of an itemset A is lesser than min_util, then any extension A' of the itemset A cannot be a high utility itemset.

Proof:

For all transactions T such that A' ⊆T

Given A' is an extension of A. Let A'-A denote the items present in A' but not in A.

As A'⊆T this implies A-A' ⊆ T/A

So u(A',T) = u(A,T) +u((A'-A),T)

= u(A,T) +$\Sigma_{i\epsilon(A'-A)}$u(i,T)

≤ u(A,T) +$\Sigma_{i\epsilon(T/A)}$u(i,T)

= u(A,T) +ru(A,T)

Let id(T) represent the id of transaction T, tids(A) and tids(A') represent the tid set in A's U-List and A' 's U-List respectively .

As A⊆A' this implies tids(A) ⊆ tids(A')

So, u(A') = $\Sigma_{id(T)\epsilon\ tids(A')}$u(A',T)

$\Sigma\leq_{id(T)\epsilon\ tids(A')}$ u(A,T) +ru(A,T)

$\Sigma\leq_{id(T)\epsilon\ tids(A)}$ u(A,T) +ru(A,T)

Utility of an itemset A' which is an extension of itemset A is less or equal to sum of ru's and iu's in the U-List of A.

Therefore if $\Sigma_{id(T)\epsilon\ tids(A)}$ u(A,T) +ru(A,T) < min_util then u(A') is less than min_util Hence Proved.

For example consider the U-List of the itemset {ec} in fig 5. If we consider min_util as 30, {ec} should be pruned from being extended because the sum of ru's and iu's is less than min_util.

3.2 Strategy 2

U-List for k+1 itemset $P(i_1i_2\ldots i_k,i_{k+1})$ is formed from U-Lists of two k itemsets $P1(i_1i_2\ldots i_{k-1}i_k)$ and $P2(i_1i_2\ldots i_{k-1}i_{k+1})$ only if $TWU(i_k, i_{k+1})$ is greater than or equal to min_util.

Proof:

It is clear that $P(i_1i_2\ldots i_{k+1})$ is super set of $\{ i_k\ i_{k+1}\}$

If $TWU(i_k,i_{k+1})$ <min_util then $TWU(P(i_1i_2\ldots i_{k+1}))$ is also less than min_util according to Property 1, if $TWU(P(i_1i_2\ldots i_{k+1}))$<min_util then P is not a high utility itemset.

Therefore itemset P can be pruned if $TWU(i_ki_{k+1})$<min_util.

Hence Proved

For example consider U-Lists of two itemsets {abc} and {abe}. The U-Lists of {abc} and {abe} are joined to form U-List of {abed} only if TWU(c,e) is greater than or equal to min_util.From the reorganized database in fig 6 the TWU(c,e) can be calculated as follows

$TWU(c,e)= \Sigma TU(<T1,T2,T3,T4,T6> \Lambda <T2,T4,T5>)$

$=TU(T2)+TU(T4)$

$=24$

As TWU(c,e)<min_util the itemset {abce} formed by joining {abc} and {abe} will not be a high utility itemset and hence can be pruned before performing the join.

IV. Proposed Algorithm

Algorithm: U-Vertical Algorithm

Input : B:an itemset (initially empty),
Ext(B): a set of 1-extensions of B,
the min_util threshold,
the item pair co-existence map

Output: all high utility itemsets with *B as prefix*

for each itemset $B_X \in Ext(B)$
if $sum(UL(B_X.iu's)) \geq min\_util$
print B_X
end if
if $sum(UL(B_X).iu's) + sum(UL(B_X).ru's) \geq min\_util$ then //Strategy 1
$Ext(B_X) \leftarrow NULL$
for each itemset $B_Y \in Ext(B)$ such that $y \in t(x)$
/*t(x) Is the set of items with TWU less than
TWU(x)*/
if $TWU(x, y) \geq min\_util$ //Strategy 2
$B_{XY} \leftarrow B_X \cup B_Y$
$UL(B_{XY}) \leftarrow Join(B_X, B_Y)$
$Ext(B_X) \leftarrow Ext(B_X) \cup B_{XY}$
end if
end for
end if
U-Vertical(B_X, $Ext(B_X)$, min_util)
end for

V. Experimental Evaluation And Results

The algorithm presented had been experimented with real time databases Retail-Store and Accidents database.The running time and memory requirement values for various min_util values of retail store database is shown in fig 9.

Min_Util (x 1000)	U-Vertical		UP-Growth	
	Running Time(s)	Memory (MB)	Running Time (s)	Memory (MB)
500	1.06	33.006	23.6	37.71
300	9.09	187.9	288.65	99.99
200	31.7	58.2	962.5	256
350	5.89	91.29	145.9	83.7

Fig3 Running times and memory requirement for retail-stores database

The running time and memory requirement values for various min_util values of accidents database is shown in fig 10.

Min_Util (x 1000)	U-Vertical		UP-Growth	
	Running Time(s)	Memory (MB)	Running Time (s)	Memory (MB)
50	0.41	11.3	3.6	33.9
35	1.38	22.546	50.289	199.171
25	3.38	281.47	244.27	305.765
20	7.144	170.79	629.78	543.26

Fig4 .Running times and memory requirements for accidents database

From the above values of running time and memory requirement it can be observed that the algorithm U-Vertical outperforms UP-Growth in terms of time and memory complexity

VI. Conclusion

In this paper we presented the algorithm for mining high utility itemsets which outperforms UP-Growth, UP-Growth+ and other two-phase algorithms. The algorithm proposed in the paper is designed for static databases. It can be further extended to design an efficient algorithm for mining high utility itemsets from dynamic databases.

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]. 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..
[4]. Vincent S. Tseng1 , Cheng-Wei Wu1 , Bai-En Shie1 , and Philip S. Yu2. UP-Growth: An Efficient Algorithm for High Utility Itemset Mining, In IEEE Transactions on Knowledge and Data Engineering , Vol. 25, No. 8, August 2013.
[5]. A. Erwin, R.P. Gopalan, and N.R. Achuthan, "Efficient Mining of High Utility Itemsets from Large Data Sets," Proc. 12th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD), pp. 554-561, 2008.
[6]. M.Y. Eltabakh, M. Ouzzani, M.A. Khalil, W.G. Aref, and A.K. Elmagarmid, "Incremental Mining for Frequent Patterns in Evolving Time Series Databases," Technical Report CSD TR#08-02, Purdue Univ., 2008.
[7]. Y. Liu, W.-K. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In Proc. Utility- Based Data Mining Workshop, pages 90–99, 2005.
[8]. V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu. Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Transactions on Knowledge and Data Engineering, 2012, doi: 10.1109/TKDE.2012.59.
[9]. R. Chan, Q. Yang, and Y. Shen, "Mining High Utility Itemsets," Proc. IEEE Third Int'l Conf. Data Mining, pp. 19-26, Nov. 2003.
[10]. B. Barber and H. J. Hamilton. Extracting share frequent itemsets with infrequent subsets. Data Mining and Knowledge Discovery, 7(2):153–185, 2003.
[11]. C. W.Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng. Efficient mining of a concise and lossless representation of high utility itemsets. In Proc. IEEE Int'l Conf. Data Mining, pages 824 –833, 2011.
[12]. C. W.Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng. Efficient mining of a concise and lossless representation of high utility itemsets. In Proc. IEEE Int'l Conf. Data Mining, pages 824 –833, 2011.
[13]. Fournier-Viger, P., Wu, C.-W., Gomariz, A., Tseng, V. S.: VMSP: E_cient Vertical Mining of Maximal Sequential Patterns. In: Proc. 27th Canadian Conference onArtificial Intelligence, Springer, LNAI, pp. 83-94 (2014).
[14]. Frequent Itemset Mining Dataset Repository. http:// fimi.ua.ac.be/, 2012