Turkish Online Journal of Qualitative Inquiry (TOJQI)

Volume 12, Issue 3, June 2021: 4251-4265

Research Article

ACCURATE RETRIEVAL OF HIGH UTILITY ITEM SETS FROM THE TRANSACTIONAL DATABASE TO SUPPORT SERVICE ORIENTED COMPUTING

[1*]Mr.M. Ganesan, [2]Dr. S.Shankar

ABSTRACT

In data mining, utility mining is a key field and high utility itemsets (HUIs) are revealed using this. From a multi-level dataset, HUIs are retrieved by this field. In a multi-level dataset, HUIs are retrieved using an existing algorithm called MUMA (multilevel utility mining algorithm). A tree structure is implemented in MUM algorithm and it is termed as MUMT (multilevel utility mining tree) for storing utility information of itemsets. For pruning the pattern, item profits are only considered in the existing work, which minimizes the accurate utility mining outcome and also ih has high computational overhead. For high utility mining, static data are considered in this work, so dynamic streaming data cannot be supported using this. Major objective of proposed research work is to develop a high utility mining extraction framework with reduced computation overhead. For the same, Dynamic Sliding Window Tree based Utility Mining Algorithm (DSWT-UMA) technique is introduced in this work. For high quality mining, considered the streaming data in this work. This work performs the construction of sliding tree by concerning time varying data for supporting high utility item mining form dynamic streamlining data. Based on profit and timing, considered the data pruning after constructing tree.This work prunes the item having old historic data and less profit. At last, based on this item counts, extracted the high utility items. In this work, JAVA is used for evaluating proposed associative classifier model. Better performance is exhibited by proposed technique than the existing work.

Keywords: Item count, pruning of data,sliding window tree algorithm,dynamic streaming data,Utility mining

[1*]Assistant Professor, Department of Information Technology Hindusthan college of Engineering and Technology Coimbatore India. Email-id: mganesan.it@hindusthan.net

[2]Professor & Head, Department of Computer Science & Engineering, Hindusthan college of Engineering and Technology Coimbatore India. Email-id: shankarhicet@hindusthan.net

I. INTRODUCTION

From large databases, potentially useful, previously unknown and non-trivial information are revealed using the process called data mining [1]. An important data mining technique called Association Rule Mining (ARM) is used for discovering rules/patterns in a large database [2].

Group of items having same occurrence are identified using ARM, for example, market basket analysis. There are two steps in the association rule mining: Frequent itemsets are generated at first [3].

Association rules are generated in the second stage. Identification of frequent itemsets is a major challenge in the association rule mining. In association rule mining, frequent itemset computation is a highly important step [4]. As, second sub-problem can be solved easily, generation of frequent itemsets are highly concentrated by researchers.

In transaction database, frequently occurring itemsets are termed as frequent itemsets [5]. Identification of all frequent itemsets is a major objective of Frequent Itemset Mining. Profit values of items with high and low selling frequencies are low and high [6]. For instance, profit of frequently selling items like pen, milk and bread are very low, in contrast, profit of infrequently selling items like diamond, platinum and gold.

So, in a database, requirement of computing most valuable itemsets/customers cannot be fulfilled by computing only traditional frequent patterns. In real world retail database, major in total profit is contributed by most valuable customers/itemsets [7]. This is a motivation behind the development of mining model for discovering customers/itemsets which has major contribution in profit [8]. The itemset having frequency support greater than the minimum user specified threshold is termed as frequent itemset.

Computation of itemsets having utility value greater than user specified minimum utility value is a problem of high utility itemset mining [9]. In a database, utility of itemset corresponds to profit or value associated with it [10]. For instance, with respect to profit, computer system is highly profitable when compared with telephone. Importance, profitability, interestingness defines utility [11]. It measured with respect to cost or some other user specified values. Following two aspects are involved in the item utility in transaction database:

(1) The distinct items importance are termed as external utility (e), i.e. unit profit and

(2) The items in transaction’s importance are termed as internal utility (i), i.e. quantity

Utility of Itemset (U) = external utility (e) * internal utility (i).

High utility mining are used in the applications like computation of important pattern in biomedical applications, website click stream analysis, online e-commerce management,retail stores [12]. On precise database, all the above mentioned high utility itemset mining techniques and their variations are designed to perform. A detailed information about transactions status will be there in precise database.

There is another class of real world transactional datasets termed as uncertain databases, where high utility Itemset mining techniques are proposed. In uncertain datasets, some information may not available or it may be lost. From uncertain databases, various techniques are used for extracting high utility itemsets.

For high quality mining, considered the streaming data in this work. This work performs the construction of sliding tree by concerning time varying data for supporting high utility item mining form dynamic streamlining data. Based on profit and timing, considered the data pruning after constructing tree.This work prunes the item having old historic data and less profit. At last, based on this item counts, extracted the high utility items.

II. RELATED WORKS

Mining of High Utility Itemset is a most familier concept and various algorithms are developed for mining HUIs like HUI Miner, D2HUP, UP-Growth, IIDS,IGUP, two phase [13]. In general, they are categorized into two classes: one-phased and two-phased algorithms. There are two phases in the two-phase algorithm characteristics.

Potential high utility itemsets candidate set are created in first phase. Precise utility of every candidate is computed in second phase, which are computed in first phase for identifying high utility itemsets. Examples of two-phase based algorithms include UP-Growth, IIDS, IHUP, two-phase and one phase based algorithms include HUI miner and D2HUP.

Liu et al in [13] computed high utility itemsets by implementing a two-phase algorithm. Itemsets with high utility values when compared to user specified threshold are computed using utility mining. In Phase I of this paper explains about transaction weighted utilization. Candidate is formed by adding combinations of high transaction weighted utilization item sets at every level in level-wise search. Overestimated itemsets are removed using another database scan in phase II.

Less memory space, less computational cost and few databases scans are required in two-phase. On real as well as synthetic databases with large amount of data, with respect to memory cost and speed, effective performance is shown by this method.

Shuning Xing etalIn [14]introduced a Fast Utility Tree (FU-Tree) for proposing UP-tree process. Number of scanning of original database are minimized by introducing the Link Queue in this technique and over estimated utility are minimized by adopting prefix utility. In construction trees time consumption, UP-Tree is outperformed by FU-Tress as shown in experimental and theoretical analysis. High utility itemsets mining efficiency is enhanced using this.

Ahmed et al. [15] mined high utility itemsets by implementing a tree-based algorithm, called IHUP. Transaction and high utility itemsets information are maintained using IHUP-Tree in this. A TWU value, support count and item name will be in every node of IHUP-Tree. Three steps are used in this algorithm for computing High Utility Itemsets. The IHUP-Tree is constructed in the first step, HTWUIs are generated in the second step and at last, high utility itemsets are identified using this HTWUIs.

Tseng et al. [16] used a data structure called as UP-Tree for implementing an algorithm termed as UP-Growth (Utility Pattern Growth) for maintaining transaction and high utility itemsets information. There are three parts in this algorithm framework: UP-Tree construction, potential high utility itemsets generation form UP-Tree using UP-Growth, high utility itemsets identification from potential high utility itemsets.

From global UP-Tree, PHUIs are generated effectively using two methods in UP-Growth. They are DLN (Decreasing local node utilities) and DLU (Discarding local unpromising items). In phase I, number of candidates are reduced effectively using DGN and DLU techniques. During global UP-Tree construction, they are applied. Especially with long transactions in database, better performance is shown by UP-Growth and it outperforms state-of-the-art algorithms.

M. Liu et In [17] implemented an efficient algorithm HUI Miner to mine high utility itemset and proposed a novel data structure termed as utility-list. In addition to utility information of itemsets, important pruning information toHUIMiner are provided by utility lists. Candidate high utility itemsets are not generated by HUI-Miner algorithm framework.

Without scanning the original database, high utility itemsets from utility-lists are mined using HUI-Miner after the construction of initial utility-lists from an original database. This database is scanned twice for considering initial utility lists. With respect to memory consumption and running time, over state of art algorithms, significant performance enhancement is achieved using this HUI-Miner.

Vincent S. Tseng et al [2]proposed a novel framework to mine top-k high utility itemset, where, HUIs count required for mining is represented k. Proposed two effective algorithms called TKO (mining Top-K utility itemsets in one phase) and TKU (mining Top-K Utility itemsets) to mine the itemsets without setting min_util.

III. HIGH UTILITY MINING BASED ON SLIDING WINDOW TREE

For high quality mining, considered the streaming data in this work. This work performs the construction of sliding tree by concerning time varying data for supporting high utility item mining form dynamic streamlining data. Based on profit and timing, considered the data pruning after constructing tree.This work prunes the item having old historic data and less profit. At last, based on this item counts, extracted the high utility items. Figure 1 illustrates the proposed research work’s overall processing flow.

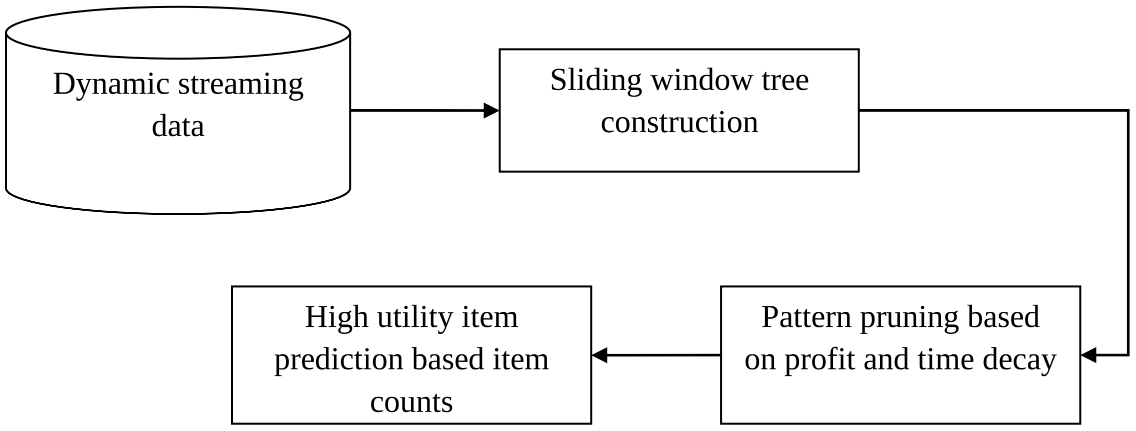
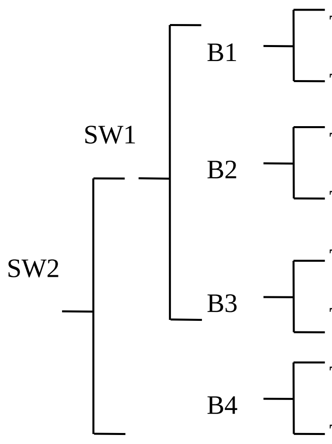

Dynamic streaming data
Sliding window tree construction
Pattern pruning based on profit and time decay
High utility item prediction based item counts

Figure 1. Processing flow of proposed research work

3.1. SYSTEM MODEL

Assume a finite set of items as $I = \{i1, i2,...,\}$; continuous sequence of transactions $\{T1, T2,...,Tn, . . .\}$ is used to form an uncertain data stream S. Thereare number of items in transaction $Tq = \{i1, i2,...,\}$ $(1 \leq l \leq L)$ in S and existential probability $p(ip, Tq)$, internal utility iu(ip, Tq) are associated with every item ip in Tq for indicating quantity value of ip inTq as well as likelihood of ip being present in Tq. In addition, eu(ip) represents item ip in I's external utility. Data cannot be stored as a infinite volume in an uncertain data stream and in dynamic adjustment, storage structure are needed for reflecting itemstes utility evolution.

So, it requires an sliding window with m most recent batches, which are represented as SW$i = \{Bi, Bi$+1,...,+m−1$\}$. In a time period, certain number of continuous transactions are there in a batch Bi; that is, $Bi = \{Tj, Tj$+1,...,+h−1$\}$. Over uncertain data stream, sliding window based HUIs mining is used for mining PHUIs of every new window, which is formed during the removal of oldest batch and insertion of newest batch in window.



TID	Transaction
T1	(a, 1, 0.81) (c, 1, 0.79) (d, 2, 0.93)
T2	(a, 2, 0.88) (c, 6, 0.92) (e, 2, 0.68) (f, 5, 0.83)
T3	(a, 1, 0.83) (b, 2, 0.78) (c, 3, 0.91) (d, 3, 0.86) (e, 1, 0.76)
T4	(b, 4, 0.072) (c, 3, 0.89) (d, 3, 0.86) (e, 2, 0.73)
T5	(b, 2, 0.93) (c, 2, 0.87) (e, 1, 0.68) (f, 2, 0.63)
T6	(a, 2, 0.93) (f, 5, 0.76)
T7	(a, 2, 081) (d, 1, 0.92) (f, 6, 0.78)
T8	(b, 4, 0.69) (c, 6, 0.83) (e, 3, 0.88) (f, 2, 0.63)

(a) Partial of an uncertain data stream

Item Name	a	b	c	d	e	f
External Utility	3	6	5	8	4	3

(b) Profit Table

Figure 2. Partial of an uncertain data stream with its profit table

For instance, a portion of an uncertain data stream with its profit table are illustrated in figure 2. Assumption is made that, two transactions are included in every batch and three batches are included in every sliding window: $B1 = \{T1, T2\}$, $B2 = \{T3, T4\}$, $B3 = \{T5, T6\}$, and $B4 = \{T7, T8\}$,SW1 = $\{B1, B2, B3\}$ represents first sliding window includes first three batches. After filling fourth batch $B4$ with transactions, SW2 = $\{B2, B3, B4\}$ forms the second sliding window.

3.2. SLIDING WINDOW TREE CONSTRUCTION

Arrival of newly generated transactions are initiated with the flow of data stream and sliding window SW moves forward with old transactions which are removed from SW. This section presents a technique to mine frequent patterns from SW's transactions dynamic set. Order of item arranging differs between FP-tree and SWP-tree representations. In descending frequency order sorted the items in FP-tree. If static data sets are mined using FP-tree, several desirable propertiesare produced using this ordering.

Twice the scanning of data stream is needed for tree construction, if frequent patterns in a data stream are mined using a FP-tree. It cost high in various stream mining applications. In addition, there will be a change in descending frequency order of items in tree, if new transactions are arrived and it is a highly a cost consuming process. Because of this the SWP-tree items are in some predefined total order $\propto$, for instance, data items lexicographic order.

Constructed an item-index table for indexing SWP tree nodes. Following fields are there in every entry of that table: def, tid, item-name and a pointer to point the item's earliest occurrence in SWP-tree. For transactions in SWP tree, decayed frequencies sum are recorded using item-index table's def field. At a specified moment, all the items that are included in the SWP-tree are recorded by item-index table, which is not the case in frequent item-header table used in FP-tree. Based on projection order, stored the entries of item-index table, which are utilized to organize SWP tree nodes.

In SWP tree, there is a triple 'item-name:tid:def' in every non-root node for representing stored item's name, most recent transactions identification which is being updated in tree including the item, decayed frequency of an item. The pseudo-code used to incrementally update SWP-tree is given in algorithm 1, upon the arrival of new transaction, it is executed. Item recorded in node are returned using function called node or item. String concatenation operator is represented as $\oplus$. Procedure of SWT construction is as follows,

```
Algorithm: SWT construction

Input: $T_c$represents transaction which are arrived newly

N represents sliding window size

Θrepresents user support threshold

϶representsmaximum support error

frepresentsdecay factor

ST representsSWP tree

IT representsitem-index table

Output: Updated SWP tree with added$T_c$

1. A pointer TP is created to points to ST's root

2. String variable created and initialized as an empty string

3. for l$\leftarrow$ 1 to $|T_c|$ do

4.      TP's children are searched for computing a node Node such that String $\oplus$ item (Node) matches l-prefix pattern of $T_c$'s projection

5. If Node is found then

6.      Update tid and def in Node for reflecting addition of $T_c$

7. Else

8.      New node is created Node such that String $\oplus$ Item (Node) matches l-prefix pattern of $T_c$'s projection

9.      Record correct tid and def in Node

10.     Link Node with other nodes in tree that represent same item (If any)

11. end

12. Append item (Node) to string

13. If item (Node) matches an entry e in IT then

14.     Update tid and def in e

15. else–
```

```
16.     A new entry e is created and make it point to Node
17.     Record correct tid and def in e
18 end
19. Make TP point to Node
20. end
```

During the arrival of new transaction T_c, based on projection order of T_c, SWP tree is added with the items in $T_c$as indicated in lines 5–12 of Algorithm 1. As mentioned in lines 13–18 of Algorithm 1, with either an updated entry or a new entry, modified the item-index table after the insertion of an item. Point should be noted is, after the incorporation of T_c with SWP tree, for efficiency, updated only the record-keeping information of nodes which are along the path that register T_c's projection.

In update, updated the information having association with newly arrived transaction. In the following manner, other nodes correct decayed frequencies are obtained. Assume a pattern P is registered by a node with a tid of T_j.tid and decayed frequency $freq_d(P, T_j)$. With decay factor f and current transaction T_i, correct decayed frequency for T_i, i.e., $freq_d(P, T_i)$, is computed as$freq_d(P, T_i) \times f^{(T_i.tid) - T_j.tid}$.

3.3. RULE PRUNING BASED ON TIME DECAY

Developed a time decay model for differentiating recent and historic transactions in a data stream while mining frequent patterns. This model decays the patterns occurrence count in transactions with time. A pattern's decayed frequency is defined as $freq_d(P, T)$, if current transaction in SW is T. The $freq_d(P, T)$ is initialized as 1, if T arrives and has P, else it is made as 0. The $freq_d(P, T)$ is multiplied by a decay factor f, $0 < f < 1$ during the arrival of every new transaction.

Then, $freq_d(P)$ becomes $\sum_{i=1}^{N} freq_d(P, T_i)$, where, SW transactions are represented as T_1, T_2, . . . , T_i, . . . , $T_N$and in this oldest transaction is given by T_i and newest transaction is given by T_N. If P is there in transaction T_i, then $r_i = 1$, else it will be 0. With the decayed frequency definition, following expression can be obtained in a straightforward manner as,

$$freq_d(P) = \sum_{i=1}^{N} r_i \times f^{N-1}$$

Any pattern P's decayed frequency is less than its original frequency called true frequency because $f< 1$,i.e., $freq_d(P) <freq(P)$. A frequent pattern with true frequency θN will be lost, if θis used as minimum support threshold. This is because decayed frequency must be less than θN under time decay model.

Between pattern's decayed frequency and true frequency, maximum allowed error called ϵN ($0 < \epsilon < \theta$) is introduced for addressing this issue. In can also be mentioned as, under the time decay model, new support threshold corresponds to $\theta - \epsilon$. Frequent pattern mining accuracy under time decay model is made equal to conventional frequent-pattern mining by selecting proper value of f.

In sliding window, with respect to transactions, measured the precision and recall values. At first, assurance of 100% recall in time decay model are computed. Assume a condition, where there only θN transactions with pattern P, and isis frequent because $freq(P) >= \theta N$. Even with minimum decayed frequency of P, while mining the sliding window, P must be selected for assuring 100% recall.

Performed the rule pruning as per the measured time decay. From the database, eliminate the patterns having old historic time. That means, from input pattern database, eliminated the patterns with lesser $freq_d$value for supporting effective as well as accurate prediction.

3.4. UTILITY MINING BASED ON ITEM COUNT

A novel technique called Sketch and count based Tilted time frame is used for predicting frequent items after pattern pruning. Within a specified window interval, frequency counts are estimated using hCount sketch as mentioned before. An array of $M \times H$ counters is maintained in hCount algorithm, where, parameters defined by allowed error and data characteristics are represented as M and H.

Mapping between item occurrence and H counters are done in this algorithm using H hash functions, which are incremented by one subsequently. Minimum value of all counter to where items are mapped is used for computing frequency of item's estimate.

In this work, algorithm needs to operated in K intervals instead of single window interval. There are tilted time-frame in these windows as follows, b most recent stream values are covered by first window w_0,that is transactions $T_{n-b+1}, \ldots, T_n$. The w_0 size is defined using the parameter b and it is termed as batch size. Next b transactions are covered by second window, which has the size b. Then, every subsequent window size a doubled value of previous window size. The i-th window size in general is expressed as $w_i = 2^{i-1} b$, $0 < i < K$.

The hCount sketch is extended to an array of $M \times H \times K$ elements for accounting all K window intervals. This is done by replacing every one of $M \times H$ counters $c_{m,h}$in original structure by an array $c_{m,h}[]$ of K counters, for $0 \leq m < M$, $0 \leq h < H$[3]. Counts for newly incoming stream values based on hCount algorithm are stored in this arrays first element c[0], in some cases, it may be stored in second element. Counts historical values are stored in subsequent elements, and this count refers to respective window interval. In addition, item frequencies history are tracked by them.

The straightforward technique for implementing the shifting operation is the utilization of respective counter and intermediate windows. New data which are depicted in gray are received always by a counter corresponding to first window, c[0]. Every b transactions are shifted sequentially using counter values. In this model, it is easy to answer item frequency questions. In a specific window interval wq, if a query for frequency of an item comes in, counters which stores information on time intervals overlapping with wq are identified and estimates from these counters are added.

Point to be noted is, estimation in introduced with errors, if counter time intervals and query interval wq are not aligned. Over intervals, frequencies are counted by us, which do not belongs to wq. This condition is true for query interval’s two ends. Simplicity is a major advantage of this algorithm.

These advantages are comes from the expense of memory for intermediate windows. The (2K − 1)S memory is needed by NaiveCount, where, windows count is represented as K and memory needed in hCount or some other technique is represented as S and in the worst case, K number of shift operations are performed.

IV. RESULTS AND DISCUSSION

In this work, JAVA is used for evaluating proposed associative classifier model. Comparison of performance is made between existing MUMA (multilevel utility mining algorithm) and proposed Dynamic Sliding Window Tree based Utility Mining Algorithm (DSWT-UMA) techniques. With respect to accuracy of prediction, these comparison are made. Memory consumption, execution time, Peak high utility itemset, High utility Item set rate are the performance metric used in this work for making comparison between existing and proposed techniques.

4.1. HIGH UTILITY ITEMSET MINING COMPARISON

The high-utility itemset mining problem corresponds to the computation of group of items called itemsets, which produced high profit in a database while selling together. A threshold termed as “minutil” (the minimum utility threshold) is a user specified one. All the high-utility itemsets are produced as an output of high-utility itemset mining algorithm . High-utility itemsets are the group of items, which guarantees “minutil” profit.

Table 1 list the values obtained for HUI metric in simulation and figure 3 illustrates the graphical comparison of the same.

Table 1.HUI comparison values

Dataset	Number of HUI retrieved				
	MU-ET	MU-LG	MU-UP	MUMA	DSWT-UMA
Retail	480	510	530	600	640
MSNBC	250	290	310	350	380

Figure 3 illustrates the comparison of above simulation values in a graphical representation.

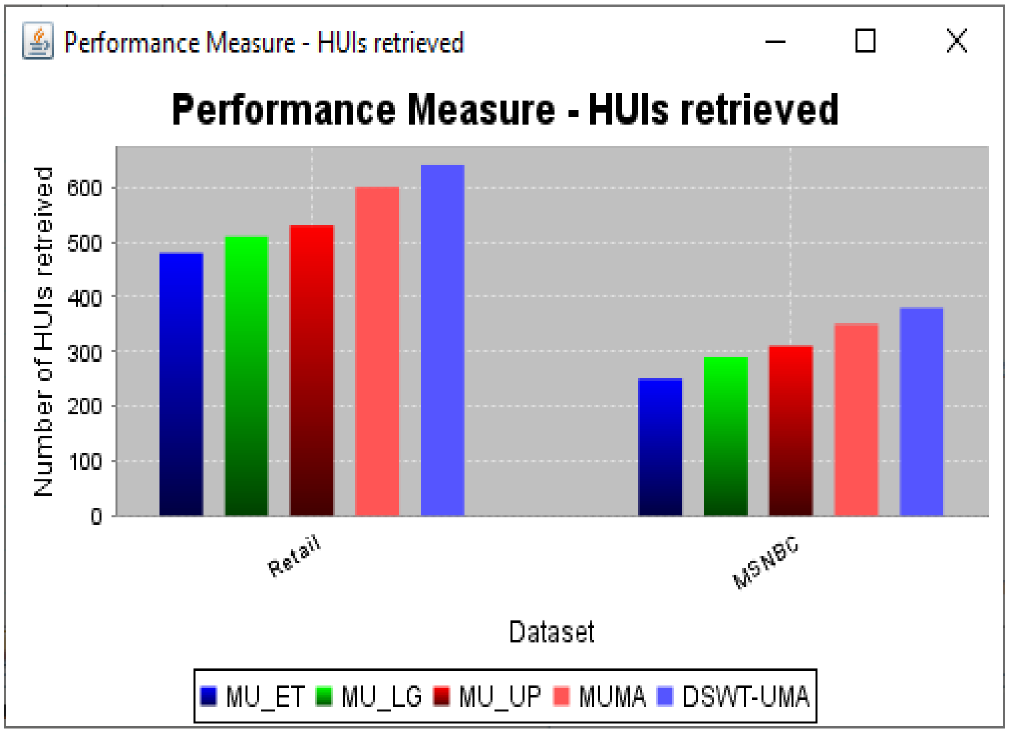


Figure 3.HUI comparison

The HUI performance metric comparison between existing MUMA and proposed DSWT-UMA techniques are illustrated graphically in figure 1. As shown in that graph, around 12% increase in HUI rate is achieved in proposed DSWT-UMA technique in comparison to existing MUMA technique.

4.2. PEAK HIGH UTILITY ITEMSET MINING

Peak high utility itemsets (PHUIs) for finding itemsets which produces a high utility in specified time duration than the usual. Various real-life applications requires this computation of such patterns. For instance, in a retail store, procurement management and planning can be enhanced by computing the time duration, where an itemset gives more profit than the usual time.

Moving average crossover concept used in time series analysis forms the base for peak high utility itemsets concept. In a database D, Peak High Utility Itemsets (PHUIM) mining problem corresponds to the computation of peak high utility itemsets having their peak windows, with user-defined parameters $1 \leq$ minLength $\leq$ WD, lMinutil> 0 and $\lambda > 1$. The PHU I s represents set of all PHUIs.

Table 3 list the values obtained for PHUI metric in simulation and figure 4 illustrates the graphical comparison of the same.

Table 2. PHUI comparison values

Dataset	Number of PHUI retrieved				
	MU-ET	MU-LG	MU-UP	MUMA	DSWT-UMA
Retail	600	630	670	780	810

MSNBC	300	320	350	460	490

Figure 4 illustrates the comparison of above simulation values in a graphical representation.

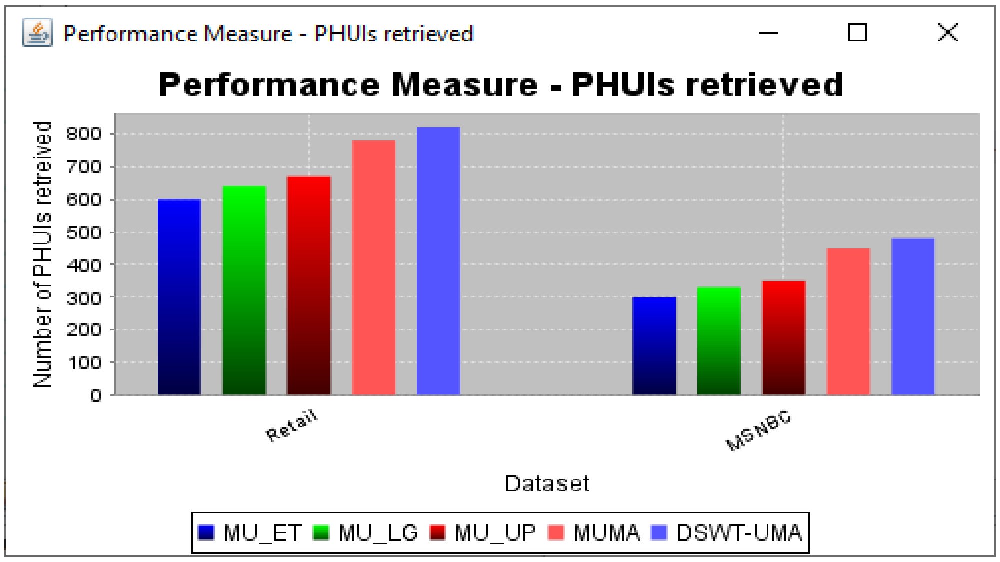


Figure 4.PHUI comparison

PHUI performance metric comparison between existing MUMA and proposed DSWT-UMA techniques are illustrated graphically in figure 4. As shown in that graph, around 75% increase in PHUI rate is achieved in proposed DSWT-UMA technique in comparison to existing MUMA technique.

4.3. EXECUTION TIME

Table 3 list the values obtained for run time metric in simulation and figure 5 illustrates the graphical comparison of the same.

Table 3. Execution time comparison values

Dataset	Execution Time in ms				
	MU-ET	MU-LG	MU-UP	MUMA	DSWT-UMA
Retail	278000	197000	156000	137000	124000
MSNBC	385000	299000	245000	165000	145000

Figure 5 illustrates the comparison of above simulation values in a graphical representation.

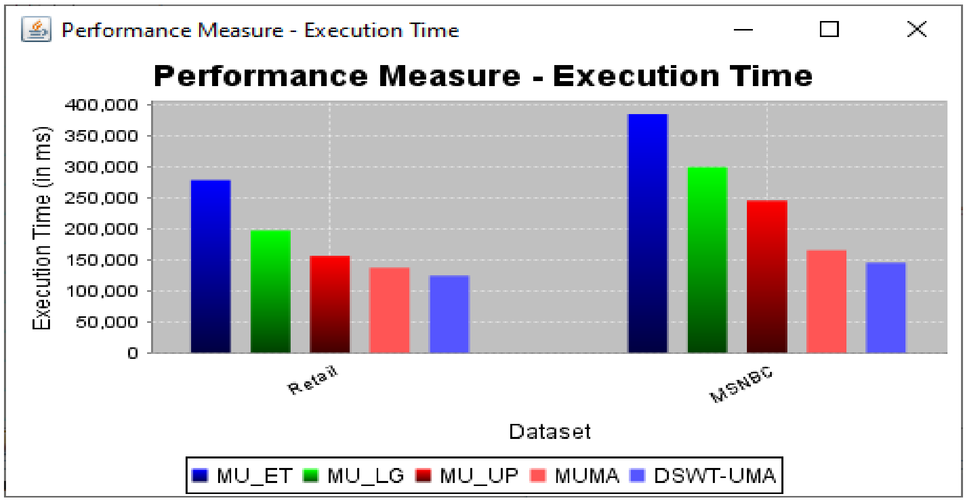


Figure 5. Run time comparison

Runtime performance metric comparison between existing MUMA and proposed DSWT-UMA techniques are illustrated graphically in figure 5. As shown in that graph, around 47% of reduction in run time is achieved in proposed DSWT-UMA technique in comparison to existing MUMA technique.

4.4. MEMORY CONSUMPTION COMPARISON

Table 4 list the values obtained for memory consumption metric in simulation and figure 6 illustrates the graphical comparison of the same.

Table 4. Memory consumption comparison values

Dataset	Memory Consumption in MB				
	MU-ET	MU-LG	MU-UP	MUMA	DSWT-UMA
Retail	1010	660	470	340	280
MSNBC	2480	1890	1250	480	450

Figure 6 illustrates the comparison of above simulation values in a graphical representation.

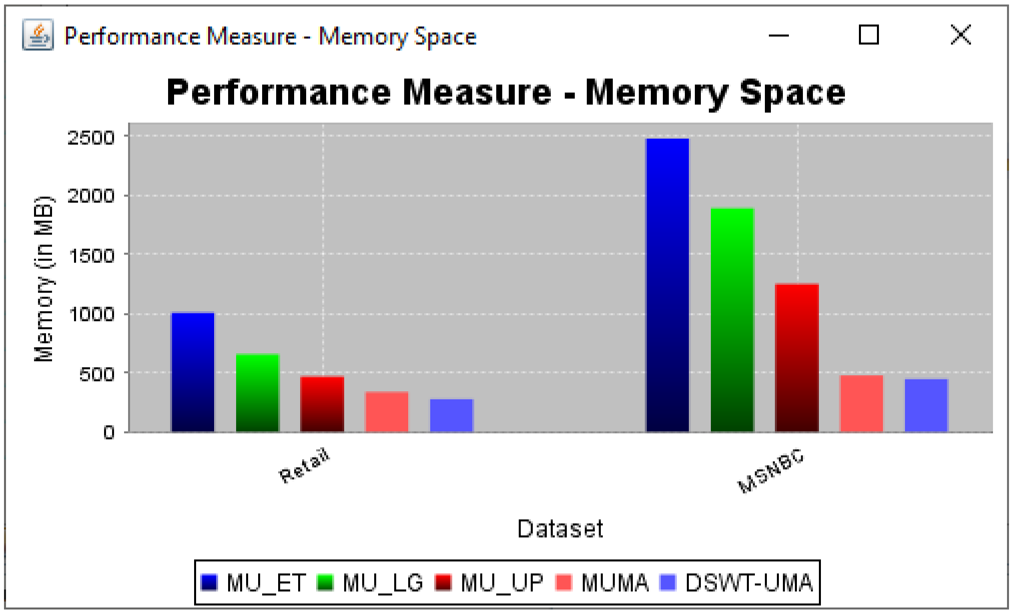


Figure 6. Memory consumption comparison

Memory consumption performance metric comparison between existing MUMA and proposed DSWT-UMA techniques are illustrated graphically in figure 6. As shown in that graph, around 13% of increase in F-Measure rate of proposed DSWT-UMA technique is achieved in comparison to existing MUMA technique.

V. CONCLUSION

For high quality mining, considered the streaming data in this work. This work performs the construction of sliding tree by concerning time varying data for supporting high utility item mining form dynamic streamlining data. Based on profit and timing, considered the data pruning after constructing tree.This work prunes the item having old historic data and less profit. At last, based on this item counts, extracted the high utility items. Java simulation environment is used for making overall performance evaluation of this work and it is shown that better performance is exhibited by proposed technique when compared with existing techniques. HUI rate shows 12% better performance in the proposed method DSWT-UMA. Likewise it shows 75% increased PHUI rate, 47% lesser run time and 13% increased f-measure rate in the proposed technique DSWT-UMA which is better than the existing MUMA technique.

REFERENCES

1. Tan, P. N., Steinbach, M., & Kumar, V. (2016). *Introduction to data mining*. Pearson Education India.
2. Li, L., Lu, R., Choo, K. K. R., Datta, A., & Shao, J. (2016). Privacy-preserving-outsourced association rule mining on vertically partitioned databases. *IEEE Transactions on Information Forensics and Security*, *11*(8), 1847-1861.
3. Yuan, X. (2017, March). An improved Apriori algorithm for mining association rules. In *AIP conference proceedings* (Vol. 1820, No. 1, p. 080005). AIP Publishing LLC.

4. Wang, T., Li, N., &Jha, S. (2018, May). Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)* (pp. 127-143). IEEE.
5. Wu, D., Luo, D., Jensen, C. S., & Huang, J. Z. (2019, April). Efficiently Mining Maximal Diverse Frequent Itemsets. In *International Conference on Database Systems for Advanced Applications* (pp. 191-207). Springer, Cham.
6. Valiullin, T., Huang, J. Z., Yin, J., & Wu, D. (2019, October). A New Approach for Approximately Mining Frequent Itemsets. In *Data Analytics and Management in Data Intensive Domains: XXI In-ternational Conference DAMDID/RCDL'2019 (October 15–18, 2019, Kazan, Russia): Conference Proceedings. Edited by Alexander Elizarov, Boris Novikov, Sergey Stupnikov.–Kazan: Kazan Federal University, 2019.–496 p.* (p. 63).
7. Ryang, H., & Yun, U. (2017). Indexed list-based high utility pattern mining with utility upper-bound reduction and pattern combination techniques. *Knowledge and Information Systems*, *51*(2), 627-659.
8. Saha, A., &Khairnar, V. D. (2018, April). Mining Utility Patterns. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 728-731). IEEE.
9. Gan, W., Lin, J. C. W., Fournier-Viger, P., Chao, H. C., Hong, T. P., & Fujita, H. (2018). A survey of incremental high-utility itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(2), e1242.
10. Nguyen, T. D., Vu, Q. B., & Nguyen, L. T. (2019, December). Efficient algorithms for mining maximal high-utility itemsets. In *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)* (pp. 428-433). IEEE.
11. Lin, J. C. W., Ren, S., & Fournier-Viger, P. (2018). MEMU: more efficient algorithm to mine high average-utility patterns with multiple minimum average-utility thresholds. *IEEE Access*, *6*, 7593-7609.
12. Wang, Z. (2016, May). Mechanism Study for E-commerce Affecting China's Future Marketing Mode. In *2016 International Conference on Economy, Management and Education Technology*. Atlantis Press.
13. Liu, Y., Liao, W. K., &Choudhary, A. (2005, August). A fast high utility itemsets mining algorithm. In *Proceedings of the 1st international workshop on Utility-based data mining* (pp. 90-99).
14. Xing, S., Liu, F., Wang, J., Pang, L., & Xu, Z. (2015, December). Utility Pattern Mining Algorithm Based on Improved Utility Pattern Tree. In *2015 8th International Symposium on Computational Intelligence and Design (ISCID)* (Vol. 2, pp. 258-261). IEEE.
15. Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, *21*(12), 1708-1721.
16. Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010, July). UP-Growth: an efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 253-262).
17. Liu, M., & Qu, J. (2012, October). Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (pp. 55-64).