Survey on Frequent Pattern Mining over Data Streams

B. Subbulakshmi
Assistant Professor

Dr. C. Deisy
Associate Professor

A. Periya Nayaki
PG Student

Department of Computer Science and Engineering,
Thiagarajar College of Engineering,
Madurai, India.

Abstract

Frequent Pattern Mining plays an important role in the field of data mining. It discovers an interesting associated pattern from databases. The concept of frequent pattern mining is extended to dynamic database mining and the data streams today. Data Stream is continuous, unbounded sequence of data elements that arrives at high speed from a specified source. Some of the real time examples of data streams are Web Click Streams, Sensor Networks, Stock Market, Retail Chain Transactions and the like. Unlike static database mining, there are lots of challenges and various data processing models are used for data stream mining. So, the study of existing algorithms is needed to design the efficient algorithms and data structure in the concept of frequent pattern mining over data streams is very important for the researchers. Hence, in this paper we reviewed the concept of data streams and overview of various algorithms for the extraction of frequent patterns based on data processing models defined for data stream mining.

Keywords

Concept Drift, Landmark Window, Sliding window, Damped Window

1. Introduction

A data stream is a continuous, unbounded, high speed, ordered sequence of items that arrived in a timely order generating from a specified source day by day. In data stream, the data is generated at high speed; the mining process on stream data is a very tedious task compared to static data mining. In static data mining, the size of the data is known prior to the user. But, in dynamic data mining (Data Streams) the size of data is not known to the user. So, there are lots of challenges and models are described to process the data streams. Hence, Data Stream Mining is an emerging and very challenging area in the data mining community today.

As the lot of algorithms has been introduced for mining stream data, there is a need to perform the association rule mining (Frequent Pattern Mining) over data streams. The main aim of this association rule mining is to identify the items that occur frequently in the database. There are two measures consider for extracting the association rules. They are support and confidence which reflects the usefulness of association rules where support (S) is the percentage of transactions that the item (A) occur in the dataset and confidence (C) is used to generate the rules over the data (e.g., A transaction which contains A and also contains B).

In Data Streams, Frequent Pattern Mining is used to generate the reports on web log data, estimating the frequency on internet packet streams, stock tickers etc. The concept becomes change in dynamic data mining over a period of time as incoming data arrived.

A window consists of the sequence of transactions from the transaction data stream. A window is either time based or count based. A window is a time based if it consists of the transactions with fixed time units. A window is count based if it consists of collection of batches where each batch contains set of transactions with fixed count.

Data Streams can be classified into two categories namely Online Streams and Offline Streams. In Offline Data Streams, the regular bulk arrival of data has been collected and stored in any backup devices or data warehouse and processed in offline. For example, consider

the web log data for generating reports after collecting the web clicks from the log file and process at any time in offline. In contrast to offline data streams, the real time updated data have been processed one by one in online data streams. Our mining process over online data streams is too fast as possible as the data arrival rate in online data streams. For example, predicting the frequency of internet packet streams is considered as the real time data because the packets are generated continuously over the internet and process the one packet at a time. The other online data stream applications are sensor monitoring system, network traffic analysis, intrusion detection system etc.

1.1 Challenges in Data Streams

There are some fundamental challenges [1] for extracting frequent patterns from data streams. They are as follows:

1. In Data Streams, due to continuous, unbounded, high speed, the data have been generated in all organizations (both offline and online). Hence, there is no enough time to rescan the entire data for multiple times as do in static data mining.
2. The data stream mining algorithm needs to handle the concept drifting problem.

Concept Drift: Since, the data streams are time-varying in nature; the set of frequent patterns change as time varies. In association rule mining, when the new incoming data is added from the data stream for processing (data become changed over a time) some of the frequent pattern becomes infrequent and vice-versa. This is known as concept drift. The following figure shows the example of concept drift.

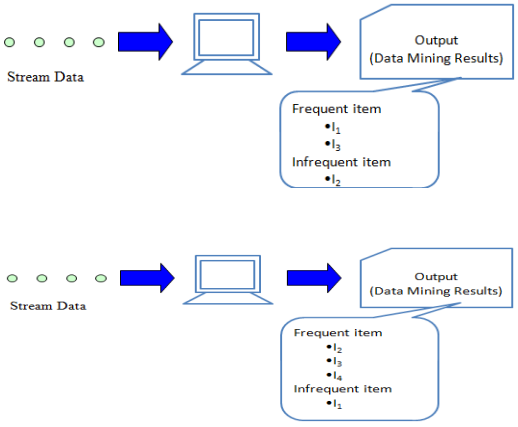

Figure1. Example of Concept Drift

In figure 1, Concept Drift is defined with an example. I_1 is frequent in initial mining process becomes infrequent and infrequent item I_2 becomes frequent after adding some transactions for processing.

3. The data generating in online data streams are very high; our mining algorithm is also very fast as possible as the incoming data rate.
4. The analysis results of data stream mining keep changing as well. Hence, it's an incremental process (i.e., the highly updated data is to be maintained at all time).
5. Due to high data arrival rate and limited system resources, the mining algorithm supports these resource adaptations.

1.2 Types of Algorithm

Based on the results obtained from mining, there exist two algorithms namely exact algorithm and approximate algorithm in association rule mining. The exact algorithm is that the complete set of frequent patterns is extracted without an error bound. All items which are greater than the minimum threshold value are collected. Moreover to know the exact results, it needs an additional cost of processing. Due to unbounded, high speed characteristics of data in the data stream, some of the algorithm maintains the short itemsets (closed or maximal). The approximate algorithm maintains the approximate set of frequent patterns with some error bound. It can consider two approaches for approximate mining: They are either false-positive or false-negative. The false positive approach contains some of the infrequent patterns in addition to set of frequent patterns in the result. The false negative approach includes infrequent patterns but some frequent patterns are also missing in the resultant part.

1.3 Data Processing Models

There are three types of data stream processing models [1] namely, Landmark Model, Damped Model or Time Fading Model, Sliding Window Model.

Landmark Model processes the entire history of stream data over the some specific point in the past and in the present. In this

model, summary data is to be maintained in the data structure.

Sliding Window Model maintains and processes the part of the stream data in the current window. The result from sliding window model reflects the recent frequent itemsets. The old transactions are deleted when the new transactions arrived into the current window for processing due to unbounded, high speed characteristic of data in nature. The size of the window depends on the application and the system resources.

Damped Window Model processes the stream data based on the weight assigned to each transaction. Here, the older transactions are assigned by less weight towards the itemset frequencies and higher weight for recent data. Damped Window Model is also one of the types of sliding window model. In this model, the decay rate is used to reduce the effect of old transactions from the window. This model brings the recent frequent itemsets in the mining result.

Based upon the application and user needs, the model has been chosen for mining process.

2. Analysis-Frequent Pattern Mining Over Data Streams

2.1 Lossy Counting Algorithm

Manku and Motwani [5] proposed a paper "Approximate Frequency Counts over Data Streams" in 2002. In this paper, the author proposed a Lossy Counting algorithm for frequency counts over the singleton items. Initially, the stream of transactions is to be filled in main memory as many numbers as possible. Then, the stream data is divided into a sequence of buckets and bucket id is assigned for each bucket. The current bucket is denoted as b_c and α be the number of buckets processed in the current batch. The inputs for this algorithm are support and error rate. By using these values, the approximate frequent patterns are collected over the entire history of the data stream.

There are three modules has been implemented in this paper. They are Buffer, Trie and setGen. In Buffer module, the input stream of transactions is filled with memory repeatedly for processing. Transactions in each bucket are represented by the item - id. Here, a bitmap is used to represent the transaction boundaries. A bit per item id indicates that the last member of the transaction. In Trie module, the compact prefix tree D is to be constructed with pre order traversal in lexicographic order is maintained. Each node is labelled as <item-id, Freq, Δ, level> where item-id is the element id in the transaction, Freq is the frequency count of that item in the current batch, Δ is the maximal allowable error, level of the node from the root. The root node is always labelled as 0. The level of any other node is one more than that of its parent. The prefix tree D is updated as follows:

- Updation: If (set, Freq, Δ) $\in$ D, then update prefix tree D by increase the frequency count of item in that particular node. The deletion happens if the entry (Freq+Δ) $\leq$ b_c.
- Creation: If set $\geq$ α and also present in the current batch, but does not present in D. Then create a new entry as (set, Freq, $b_c - \alpha$).

In setGen module, the subsets of itemsets are generated from the item ids in singleton items. This module is activated if either the set is in Trie or the item which exceeds α in the current batch. If any subset does not satisfy a threshold value to make entry into the tree data structure after updating and creation, then the superset is also deleted from the prefix tree.

There are no false negatives and all frequent itemsets are outputted using this algorithm. But, setting up of error bound value should be a tedious work. Because, if this value is small, then the lot of approximate sub frequent itemsets is generated. Hence, it takes more memory space and more CPU processing power.

2.2 DSM-FI Algorithm

Li and Lee et. al [6] proposed a paper "An Efficient Algorithm for mining frequent itemsets over the entire history of data streams" in 2004. In this paper, the author proposed the prefix tree based; in-memory data structure called ISFI (Item Suffix Frequent Itemsets) forest based on DSM-FI algorithm. It generates an approximate amount of frequent itemsets over the entire history of data streams.

An ISFI-Forest consists of two components such as HT (Header Table) and SFI-Trees (Sub Frequent Itemsets). The batch of transactions is processed together. For each item in the transaction, the corresponding SFI-tree is generated. Each unique item X, the sub

frequent itemsets prefixed by Y, the DSM-FI inserts an entry into the HT(Y) or if X is already in HT(Y), it just increments the Freq (X). ISFI tree consists of four fields: item-id, Freq, batch-id, link where item-id represents the identifier of that item in a transaction, Freq represents the frequency count of that item, batch-id represents the id of the processing batch and link represents the link points to the first head node of X in the SFI-tree.

Let us consider an incoming batch of transaction, $y_1, y_2, y_3, \ldots, y_{k-1}, y_k$. The item suffix tree is generated for $(y_1, y_2, y_3, \ldots, y_{k-1}, y_k)$, $(y_2, y_3, \ldots, y_{k-1}, y_k)$, $(y_3, \ldots, y_{k-1}, y_k)$, (y_{k-1}, y_k) , (y_k) . DSM-FI periodically prunes the itemsets which are less than threshold value S. This pruning happens reconnection of the nodes in the SFI-tree.

Figure 2 shows an example of constructing a sample ISFI- forest for transaction containing an item {FGH}. It consists of HT's and SFI-trees for each unique item F, G, H.

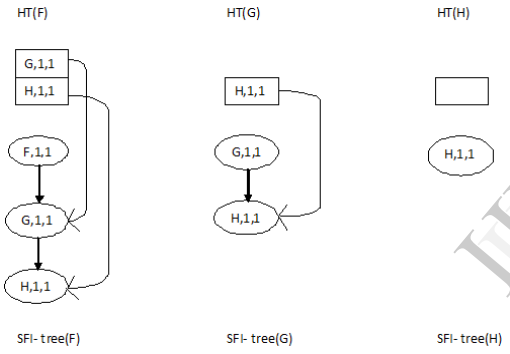

Figure2. Sample ISFI-Forest Construction

A SFI-tree is a more compact data structure comparable to prefix tree. For example, (y_1, y_2, y_3) is an FCI (Frequent Closed Itemset), it represented by two paths in a prefix tree (y_1, y_2, y_3) and (y_1, y_3). But in the SFI - tree it is represented by a single path (y_1, y_2, y_3). Moreover, higher computational cost is to be needed for maintaining the compactness of the SFI-tree. Since, more tree traversals are required to collect the frequency information of the itemsets.

2.3 FP-Stream Algorithm

Chris Gianella and Han et. al [4] proposed a paper "Mining Frequent Patterns in Data Streams at Multiple Time Granularities" in 2004. In this paper, the authors have implemented the algorithm called FP-Stream to extract the complete set of frequent patterns with an approximate error bound. This algorithm used the tilted time window model to extract the frequent itemsets. This model uses the finer granularity to mine the recent data and coarser granularity to mine the long term or historical data. To guarantee the completeness of frequent patterns, the infrequent patterns are not deleted after processing the batch of transactions. Because, those infrequent patterns will become frequent again in future. In addition to that in realistic, due to the limited size of memory, we can't store the entire streaming data. Hence, FP-Stream divides the data into three categories: Frequent, Sub-Frequent and Infrequent. The user has to specify the σ (minimum support) and the error support €. The itemsets which are greater than σ are considered as the frequent patterns. The itemsets which is less than σ but greater than € are considered as the sub-frequent set because they become frequent later. The infrequent patterns which are less than € are pruned. Because, this will not affect that much of calculating support. This algorithm maintains the paired sections: Pattern Tree, Tilted time window table of each frequent item in the node. To reduce the records from the window (i.e., pruning the itemset X), Freq(X) be the computed frequency over the time Ti and Ni be the number of transactions that occur in Ti where $1 \le i \le \tau$.

Finds a point n before that point, choose a transaction T_k between T_1 and T_n and sum of total computed frequency over T_1 and T_k is always less than the relaxed minimum support € for some k transactions, and prune those frequency records by considering unpromising $(freq_1(X).....freq_k(X))$.

For eg:

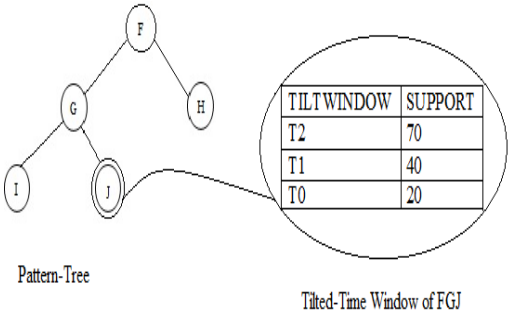

Figure3. FP-Stream structure

Figure 3 shows an example for structure of FP-Stream. The tilted time window model is mostly given the importance on recent data than the old

data as the sliding window does. But it does not delete the historical data completely. As time varies, the FP-Stream structure becomes very large. Moreover, updating and scanning over this large structure degrades the mining throughput.

2.4 Compact Pattern Stream Tree Algorithm

Tanbeer and Chowdry et al [11] proposed a paper "Sliding Window based frequent pattern mining over data streams" in 2009. In this paper, the authors proposed the compact prefix tree based structure called CPS-tree to store the exact and complete recent frequent patterns. There are two phases in this algorithm. They are Insertion Phase and Restructuring Phase. In Insertion Phase, the items are inserted into each node in the CPS-tree by a predefined item order. The item has to be maintained as order in the list called I-List. In Restructuring Phase, the I-List was sorted in frequency-descending order and the CPS - tree was also restructured based on the I-List. These two phases are repeatedly executed several number of times while processing the pane of transactions. Thus, the tree construction was started with the insertion phase and ends with restructuring phase. The restructuring phase follows the Branch Sorting method and the path adjusting method.

In the CPS - tree, there are two types of nodes which has to be maintained: Ordinary node and tail node. The former one maintains only support count in addition to the item in the node I (σ) where I is the item in that node and σ is the support count for that item in the current window. The latter one maintains I (σ; {p_1, p_2,...., p_n}) indicates that item present in last node , total support count in the current window and item present in which pane (pane counter) in the current sliding window, p_1 represents the oldest pane, p_n represents the latest pane. For example, {a, b, c} are the items present in the particular transaction in the second pane among 4 panes in the current sliding window. Here c is the tail node and it is represented as {0, 1, 0, 0}.

In contrast to DSTree (Each node represents the pane information along with their support count. It takes more memory space), here the pane information only maintains in the tail node. So, it saves a lot of memory space. In this tree construction phase, there is a small overhead of tree restructuring cost.

2.5 Weighted Sliding Window Algorithm

Pauray S.M.Tsai [10] proposed a paper "Mining Frequent Itemsets in data streams using the weighted sliding window model" in 2009. In this paper, the author proposed WSW algorithm to discover all frequent patterns over the data streams. Here, the user wants to specify the number of windows, size of the window, minimum support count and weight for each window. The size of the window is specified by time, not by transactions and the user has to be given different weights to different windows based on the importance of the data in the particular section. For example, the data in the current point are more important than the older ones. Hence, the highest weight has to be assigned for recent data.

For example,

Assume that the current time point for mining process is T_1 and the number of windows = 3. The weight α_j where $\sum_{j=1}^{3}$ assigned for each window. $\alpha_1 = 0.5$, $\alpha_2 = 0.4$, $\alpha_3 = 0.2$ and minimum support is 0.2. The support count of item F in W_{1a}, W_{1b}, W_{1c} are 10, 20,30 respectively. The support count of item G in W_{1a}, W_{1b}, W_{1c} are 50, 60, 70. The number of transactions in each window is 300, 200, 200 and minimum support for each window is 60, 40, 40.

Weighted Support count (F):

$$= (10*0.5) + (20*0.4) + (30*0.2)$$

$$= 19$$

Weighted Support count (G):

$$= (50*0.5) + (60*0.4) + (70*0.2)$$

$$= 63$$

The minimum weighted support threshold is calculated by summation of weight of each window and minimum support. In this example, the minimum weighted support count = (60*0.5) + (40*0.4) + (40*0.2) = 54.

- The total support count (10+20+30) for item 'F' is 60. But, the weighted support count for this item is 19 which is less than the actual weighted support (54) value. Hence, it is not considered as the frequent pattern.
- The support count (50+60+70) for item 'G' is 180. But, the weighted support count for this item is 63 which are greater than the actual weighted support (54) value. Hence, it is considered as the frequent pattern in the current sliding window.

From this example, we infer that the frequent patterns depend on the weight assigned for each window. Even though, the support count of an item is high, the weighted support count is low. Hence, the setting of the weight for each window of the user is reasonable and significant.

2.6 Weighted Support Frequnt Itemsets Algorithm

Younghee Kim and Wonyoung et. al [12] proposed a paper "Mining Frequent Itemsets with Normalized Weight in Continuous Data Streams" in 2010. In this paper, the author proposed the algorithm called Weighted Support Frequent Itemsets (WSFI) to maintain the important frequent itemsets based on the weight assigned to each item. Moreover, the frequent items are maintained in a tree structure called a WSFP - tree (Weighted Support Frequent Pattern-tree) that stores the compressed form of crucial information about the frequent itemsets. Weighted support is assigned to each item. The weighted support is normalized within the range of $WSmin(Y) \leq W(Y) \leq WSmax(Y)$. The weighted support of an itemset Y can be calculated by $Support(Y) * Weight(Y)$. The normalized minimum weighted support was calculated by $WSmin(Y) = Support(Y) * Wmin(Y)$ and the normalized maximum weighted support was calculated by $WSmax(Y) = Support(Y) * Wmax(Y)$. The important patterns are discovered using this weighted support count. The frequent itemsets are no less than this normalized weighted support. During processing, the itemsets are divided into three categories: Frequent items, Latent items and infrequent items. There are two factors considered: α and € where the itemsets which are greater than α are considered as the frequent itemsets and the itemsets which are less than they € are considered as the infrequent itemsets and its pruned. Because it does not affect that much of calculated support, the itemsets which are all maintain in-between α and € are considered as the latent items. These latent items will become either frequent or infrequent in the future.

α (Minimum weighted support threshold) is calculated by.,

$$\alpha = ((MAX(WSmin(Y)) + MIN(WSmax(Y)))/2$$

€ (Minimum weighted support error threshold) is calculated by. ,

$$€ = ((MIN(WSmin(Y)) + MIN(WSmax(Y)))/2$$

If $WS(Y) \geq \alpha(Frequent)$, WS(Y) < € (Infrequent),$€ \leq WS(Y) \leq \alpha(Latent)$ where € is the minimum weight support error threshold is within the range of [0,α]. Here, the setting up of weight range is very much important in this process.

2.7 Variable Size Sliding Window Algorithm

Mahmood Deypir et. al [8] proposed a paper "Towards a Variable size Sliding Window Model for frequent itemset mining over data streams" in 2012. In this paper, the author proposed a new algorithm VSW to extract the recent frequent itemsets over the data streams using dynamic window size. The optimal window size was determined automatically based on the amount of changes occurred in the set of frequent patterns. For this purpose, the user wants to give the minimum change threshold. This value determined how much the user interested in recent changes in the frequent patterns. In this algorithm, there are two phases: Window Initialization Phase and Window Sliding Phase. In former one, initial window size is specified by the user. The transactions are filled in that window and frequent itemsets are mined and stored in the prefix tree. The frequent patterns were mined using the ECLAT algorithm [2]. After completing the process of initial window, some set of transactions is inserted into a sliding window (Pane: Batch of transaction). After extracting the frequent itemsets from the pane, the prefix tree is updated. At that time, the Window Sliding Phase was activated to reduce the window size by delete old transactions from the window. The checkpoint is specified at the end of Initial window to diagnose the changes occurred in the set of frequent patterns. Due to pane insertion, some amount of frequent patterns becomes infrequent and vice versa (Concept change or Concept Drift). After detecting that concept change, if this value is greater than the user defined minimum change threshold value, then the transactions that lie before the checkpoint and the essence stored in the prefix tree was also deleted and checkpoint has updated to the point where the concept change was detected. Due to efficiency issues, the concept change was detected after every pane insertion. The conceptchange was calculated by using the formula,

$$FChange = \frac{F^+ + F^-}{F + F^+}$$

Where, F^+ are newly emerged frequent patterns

F^- are newly disappeared frequent patterns

For example,

F_T={A,B,C,D,AB,AC,AD,AE,BC,BD,BE,CD,ABC,ABD}

$F_{T'}$={B,C,D,E,BC,BD,BE,CD,CE,DE,BCD,BCE,BDE,CDE}

FChange = (7+7) / (14+7) = 0.67

From this example, we infer that the window shrinks if this Fchange value is greater than the user defined change threshold value. Otherwise, the window expands and the mining process continues.

Table 1. Comparative Analysis of Algorithms

Name of the Algorithm	Window Model	Algorithm Type	Updation Rate	Merits	Limitations
Lossy Counting	Landmark	Approximate	Batch Wise	No False Negatives in results.	Setting up of relaxed minimum support threshold leads to dilemma.
DSM-FI	Landmark	Approximate	Batch Wise	Compact tree structure has been designed to store the frequent patterns.	It needs more tree traversals for the frequency count.
CPS	Sliding Window	Exact	Batch Wise	It maintains the frequency count lists at the last node which can reduces the size of the prefix tree.	Additional computational cost needed for restructuring the tree after every pane insertion.
FP-Stream	Tilted time window	Approximate	Batch Wise	It extracts Complete set of frequent patterns using time sensitive data streams.	FP-Stream tree becomes very large over a time.
WSW	Sliding Window	Exact	Batch Wise	A single pass algorithm was developed to discover the frequent itemsets.	Weights of each window affected the mining results. So, user should specify the reasonable weight for each window.
WSFP	Sliding Window	Exact	Transaction Wise	It collects the important recent frequent patterns with limited memory space.	Initial setting of normalized minimum and maximum weight is given as random.
VSW	Sliding Window	Exact	Batch Wise	Obselete Transactions are deleted with respect to Fchange value.	The prefix tree becomes very large where there is no changes occurred in the frequent patterns in processing.

Conclusion

In this paper, we have reviewed a lot of old and recent algorithms for frequent patterns mining over data streams based on various window models. In addition to that, we have discussed the types of algorithms to give ideas for researchers to develop the exact or approximate algorithms. Moreover, the comparative table also shows that an overview of various stream mining algorithms with merits and limitations. In future, we will develop a new algorithms and techniques to overcome the shortcomings of existing algorithms.

References

[1] Agarwal R, Srikant R, "Fast algorithms for mining association rules in large databases", In: Bocca J, Jarke M, Zaniolo C (eds) Proceedings of the 20th international conference on very large databases, Santiago de Chile, Chile, September 1994, pp. 487-499.

[2] Bart Goethals, "Frequent Set Mining", Data Mining and Knowledge Discovery Handbook, pp. 377-397.

[3] Chang-Hung Lee, Cheng-Ru Lin, Ming-Syan Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining", In: Proceeding of the 2001 ACM CIKM international conference on information and knowledge management, Atlanda, Georgia, USA, November 2001, pp. 263-270.

[4] Chris Gianella, Jiawei Han, Jian Pei, Xifeng Yan, Philip S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities", In: Kargupta H, Joshi A, Sivakumar D, Yesha Y (eds) Data Mining: next generation challenges and future directions, MIT/AAAI Press, pp. 191-212.

[5] Gurmeet Singh Manku and Rajeev Motwani, "Approximate Frequency Counts over Data Streams", In: Proceedings of the 28th international conference on very large databases, Hong Kong, August 2002, pp. 346-357.

[6] Hua-Fu Li, Suh-Yin Lee, Man-Kwan Shan, "An efficient algorithm for mining frequent itemsets over the entire history of data streams", In: Proceedings of the first international workshop on knowledge discovery in data streams, in conjunction with the 15th European conference on machine learning ECML and the 8th European conference on the principals and practice of knowledge discovery in databases PKDD, Pisa, Italy, 2004.

[7] James Cheng, Yiping Ke, Wilfred Ng, "A survey on algorithms for mining frequent itemsets over data streams", *Springer*, Knowledge Information Systems, May 2007,

[8] Mohmood Deypir, Mohammad Hadi Sadreddini, Sattar Hashemi, "Towards a variable size sliding window model for frequent itemset mining over data streams", Computers & Industrial Engineering, Vol. 63, February 2012, pp. 161-172.

[9] Nan Jiang and Le Gruenwald, "Research Issues in Data Stream Association Rule Mining", ACM SIGMOD Record, Vol. 35, No.1, March 2006, pp. 14-19.

[10] Pauray S.M. Tsai, "Mining frequent itemsets in data streams using the weighted sliding window model", *Elsevier*, Expert Systems with Applications, Vol. 36, March 2009, pp. 11617-11625.

[11] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, Young-Koo Lee, "Sliding Window-based frequent pattern mining over data streams", *Elsevier*, Information Sciences, Vol. 179, July 2009, pp. 3843-3865.

[12] Younghee Kim, Wonyoung Kim, Ungmo Kim, "Mining Frequent Itemsets with Normalized Weight in Continuous Data Streams", Journal of Information Processing Systems, Vol. 6, No.1, March 2010, pp. 79-89.