Journal of Engineering and Applied Sciences 12 (4): 857-863, 2017
ISSN: 1816-949X

Frequent Pattern Mining in Data Streams

[1]Shirin Mirabedini, [2]Mahdi Ahmadi Panah and [3]Maryam Darbanian
[1]Department of Computer Engineering,
[2]Department of Management, [3]Department of Theology,
Payame Noor University, P.O. Box 19395-3697, Tehran, I.R. Iran

Abstract: The problem of frequent pattern mining in continuous streams has been widely studied in the literature because of its numerous applications to a variety of data stream mining problems such as clustering and classification. In addition, frequent pattern mining also has numerous applications in diverse domains such as retail market data analysis, network monitoring, web usage mining and stock market prediction. The algorithmic aspects of frequent pattern mining in streams have been explored very widely. This study provides an overview of these methods.

Key words: Association rule, data streams, frequent pattern mining, data mining, association rule mining

INTRODUCTION

Now a day's many organization and researcher taking interest in stream data mining. Currently there are many sources produces stream of data continuously which are unbounded, rapid and highly dynamic in nature. Example include sensor networks, wireless networks, radio frequency identification, customer click streams, telephone records, multimedia data, scientific data, sets of retail chain transactions, etc. These sources are called data streams.

Due to stream data are continuous, rapid, time varying and unpredictable and unbounded and require quick repose. Therefore, traditional data mining algorithms which are designed for static data are not suitable for mining stream data because it cannot fulfill the requirement of stream data mining.

Data stream mining is important research topic in research community and the number of researches also growing in this field. Many algorithm and technology developed and evolved for handling complexity and volume of data, still there is need of wide view in different methods. Since, it is under research area there are wide chances of exploration.

Preliminaries: A data stream D = $\{B_1, B_2, ..., B_N\}$ is an infinite sequence of batches where each batch B_i contains a set of transactions, i.e., Bi = $\{T_1,T_2, ...,T_K\}$ where k>0. Each transaction T = $(T_{ID}, I_1, I_2, ..., I_n)$ is a set of items such that T⊆D while n is called the size of transaction and T_{ID} is unique identifier of the transaction. An itemset is a non-empty set of items. An itemset with size k is called an m itemset.

There are many types of window model which use to process data stream. A window, W, can be either time-based or count-based and either a landmark window or a sliding window. W is time-based if W consists of a sequence of fixed-length time units where a variable number of transactions may arrive within each time unit.

W is count-based if W is composed of a sequence of batches where each batch consists of an equal number of transactions. W is a landmark window if W = $\{T_1, T_2, ..., T_T\}$; W is a sliding window if W = $\{T_{T-w+1}, ..., T_T\}$ where each T_i is a time unit or a batch, T_1 and T_T are the oldest and the current time unit or batch.

An association rule is an implication of the form A⇒B, where A⊂W, B⊂W and A∩B = Ø. It helps to discover combination of goods. The occurrence frequency or frequency of an itemset (x) I is the number of transactions that contain the itemset in a batch B and denoted as freq (x). Occurrence frequency is also called as absolute support. Support of X denoted by supp (X) is freq (X)/N where N is total number of transactions received in W in data stream.

It is also called as relative support. Support (A⇒B) = P (A∪B). Confidence of a rule X⇒Y denoted by conf (X) is supp(X∪Y)/supp (X) where c is the percentage of transactions received in W, containing A that also contain B. Confidence (A⇒B) = P (B|A).

Corresponding Author: Shirin Mirabedini, Department of Computer Engineering, Payame Noor University, P.O. Box 19395-3697, Tehran, I.R. Iran

The lift value indicates that how many more times itemset occurred than expected. It can interpret the importance of a rule. It is measure of a rule but it cannot be define as minimum lift to minimum support or minimum confidence.

Lift(X⇒Y) = confidence/expected confidence = Supp(X U Y)/Supp (X)* Supp (Y) An itemset X is called as Frequent Itemset (FI) if supp (X)> = minSupp where minSupp is user defined minimum threshold support. An itemset X is closed in W if there exists no proper super-itemset Y such that Y has the same support count as X in W. An itemset X is a Frequent Closed Itemset (FCI) in W if X is both closed and frequent. An itemset X is a Frequent Maximal Itemset (FMI) in W if X and Y are frequent and there exists no super-itemset Y such that X⊂Y.

To mine FIs/FMIs/FCIs over a window in data stream, it is necessary to keep infrequent itemsets, because it may become frequent later.

Example: In the example, there are 5 transactions. Each transaction contains number of items. For the simplicity we use A, B, C, D, E, F letters to denote items. In Table 1, A, B, C, D, E, F occurred 3, 4, 4, 3, 4, 2 times, respectively. Here minimum support = 4 is set. Therefore, only 3 item B, C and E satisfy minimum support threshold. Therefore, this items are frequent items because their occurrence values are equal to the threshold value. Items A, D and F are called as infrequent item sets. So, they are omitted. Thus, this is called as frequent pattern mining.

Let's take nonempty subsets of l = {B, C, E} are {B, C}, {B, E}, {C, E}, {B}, {C} and {E}. If the minimum confidence threshold is, say, 80%, then only 2 rules are output and they are (1) {B^E}⇒{C}, Confidence = 100% (2) {C^E}⇒{B} = 100%. Relative Support = 3/5 = 0.6 that means it occurs in 60% of all transactions. Lift (X⇒Y) {B, E}⇒{C} has a lift of 0.6/(0.6×0.8) = 1.25.

MATERIALS AND METHODS

General issues in data stream mining: There are some crucial issues that need to be taken into account when developing association rule for stream data.

Data processing mechanism: According to the research of Zhu and Shasha (2002), there are three data stream processing models, Landmark, Damped and Sliding windows as shown in Fig. 1 (Jiang and Gruenwald, 2006).

Table 1: Letters to denote items

ID	Item set
1	A, B, C, E
2	A, D, E
3	A, B, C, D, E, F
4	B, C, E, F
5	B, C, D

The Landmark model mines all frequent itemsets over the entire history of stream data from a specific time point called landmark to the present. In this model, we treat each time point after the starting point equally important. This model is not suitable for mining where most recent information and real time data are very important such as stock market.

The Damped model mines frequent itemsets over stream data. In stream data, each transaction has weight and this weight decreases with time. So in this model new and old transaction has different weights. Due to above characteristic of damped model, it is known as Time Fading model. The Sliding window model mines frequent itemset over stream data by temporary storing part of the data and processed. In this model, size of sliding window decided by need of application and system resources.

Besides above mention windows, Han *et al.* (2007) proposed tilted time window model. In this model, we are interested in frequent itemsets over a set of windows (Giannella *et al.*, 2003). Each window corresponds to different time granularity for example we are interested in every 10 minutes for the hour before that. Each transaction in this window has weight.

Memory management: This is major issue in mining stream data. This includes choosing of efficient and compact data structure algorithm which can efficiently stored, updated and retrieved data. In traditional algorithm, we do multiple scan over available data. This is not possible in data stream because there is not enough memory space to store all the transaction and their counts. In simple terms, memory size is bounded and hugh amount of data are arrives continuously. If we store the information in disks, the additional I/O operation will increase the processing time.

Data preprocessing: Data preprocessing is crucial aspect in the process of data mining. If data input to algorithm is not in proper format then it cannot process efficiently. So, preprocessing is needed and in which existing data transform into new data which is in proper format and suitable for processing. Different data mining tools available in the market have different formats for input which makes the user forced to transform the existing input dataset into the new format.

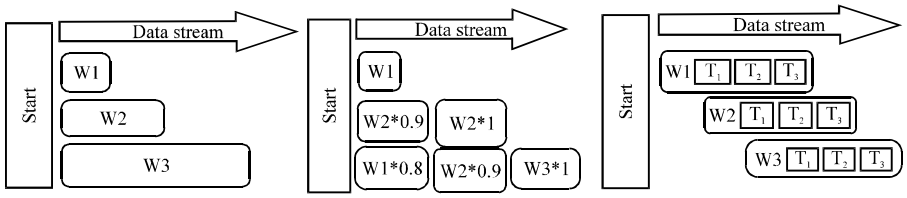

Fig. 1: Stream processing mechanism: a) Landmark window; b) Damped window and c) Siliding window

One pass algorithm: There are many algorithms for mining stream data. Based on result, they are categorizing as exact algorithms or approximate algorithms. In exact algorithms, the result consist of all the itemsets which satisfy support values greater than or equal to threshold support. To produced accurate result in stream data, additional cost is needed. In approximate algorithms, the result is approximate result with or without an error guarantee.

Concept drift: Since, stream data are rapid and time varying, we cannot assume that total number of class are fixed because itemsets which are frequent can change as well with arrival of new data. So, there is need of frequent updating of model, because old data are inconsistent with the new data. This problem is known as concept drift. If we neglect non frequent itemsets from consideration which can be frequent itemset later, we cannot get this information. Therefore, technique is needed to handle concept drifting.

Producing and maintain association rules: Mining Association rule involves a lot of memory and CPU costs. There is also one problem; processing time is limited to only one online scan. So, there is need of real time maintaining and updating association rule. However, stream data, if we update association rules too frequently, the cost of computation will increase drastically.

Resource aware: Resources such as memory space, CPU and sometimes energy are very precious in data stream mining. One cannot ignore the resources availability, for example when main memory is totally used up in processing algorithm, data will be lost and it lead to inaccuracy of results. In general, if we don't consider this problem, it will degrade the performance of the mining algorithm.

RESULTS AND DISCUSSION

Data stream mining algorithms: Mining frequent patterns in static (Agrawal *et al*., 1993; Han *et al*., 2000; Hu *et al*., 2008; Tanbeer *et al*., 2008; Tsay *et al*., 2009) and incremental (Zhang *et al*., 2007; Lee and Yen, 2008) databases has been well-addressed over the past decade. The first frequent pattern mining algorithm was Apriori which was proposed in 1993 to find association rules (Hsu *et al*., 2004; Shen *et al*., 1999; Wang, 2008; Wang *et al*., 2006; Chen and Wei, 2002; Lee *et al*., 2007; Tsay and Chien, 2004) among patterns. This technique finds the frequent patterns of length k from the set of already generated candidate patterns of length k _ 1. The main performance limitations of Apriori-like approaches result from the requirement for multiple database scans and a large number of candidate patterns, many of which prove to be infrequent after scanning the database. To overcome these problems, Han *et al*. (2000) proposed the Frequent Pattern tree (FP-tree) and the FP-growth algorithm; this algorithm reduces the number of database scans by two and eliminates the requirement for candidate generation. Introduction of this highly compact FP-tree structure led to a new avenue of research with regard to mining frequent patterns with a prefix-tree structure. However, the static nature of an FP-tree and the requirement for two database scans limit the applicability of this algorithm to frequent pattern mining over a data stream.

General issues and research issues associated with frequent pattern mining over a data stream were reviewed by Gaber *et al*. (2007), Jiang and Gruenwald (2006), respectively. However, because the scope of this work includes mining a data stream using a sliding window mechanism, we provide a thorough literature review focusing primarily on studies related to window-based approaches.

Most studies about finding frequent patterns in a data stream are based on the landmark window model (Lee and Wang, 2007; Yu *et al*., 2004; Zhi-Jun *et al*., 2006) or the sliding window model (Chang and Lee, 2005; Chi *et al*., 2006; Leung and Khan, 2006a, b; Lin *et al*., 2005; Mozafari *et al*., 2008; Li and Lee, 2009). The first attempt to mine frequent patterns over the entire history of streaming data was proposed by Manku and Motwani (2002). They developed two single-pass algorithms,

sticky-sampling and lossy counting both of which are based on the anti-monotone1 property; these algorithms provide approximate results with an error bound. Zhi-Jun *et al*. (2006) used a lattice structure, referred to as a frequent enumerate tree which is divided into several equivalent classes of stored patterns with the same transaction-ids in a single class. Frequent patterns are divided into equivalent classes and only those frequent patterns that represent the two borders of each class are maintained; other frequent patterns are pruned. DSM-FI (Lee and Wang, 2007; Li *et al*., 2005) is another algorithm that was developed to mine frequent patterns using a landmark window. Every transaction is converted into k (the total number of items in the transaction) small transactions and inserted into an extended prefix-tree-based summary data structure called the item-suffix frequent itemset forest. Manku and Motwani (2002) developed an FP-treebased algorithm, called FP-stream to mine frequent patterns at multiple time granularities using a novel titled-time window technique. When a new batch of transactions arrives, the algorithm processes the stream data using an FP-growth technique. The major limitation of this batch-by-batch processing approach is that when a stream flows, the FP-stream needs to build an FP-tree to capture the stream contents of each new batch.

Mining recent frequent patterns using the sliding window technique has also been studied in the literature. Lin *et al*. (2005) presented a method to mine a data stream for frequent patterns using a time-sensitive sliding window, i.e., the window size is defined by a fixed period of time. In this approach, the incoming stream within a window time period is divided into several batches and frequent patterns are mined in each batch individually. Using a discounting mechanism, the method discards the old patterns by using an approximation table that provides the approximate counts of the expired data items. Chang and Lee (2005) proposed estWin that finds recent frequent patterns adaptively over an online transactional data stream using the sliding window model. This algorithm requires the minimum support threshold and another parameter termed the significant support to adaptively maintain the approximate frequent patterns window after window.

Most of the methods discussed above find approximate frequent patterns (Giannella *et al*., 2003; Manku and Motwani, 2002; Silvestri and Orlando, 2007; Yu *et al*., 2006) with an error bound or additional pruning threshold. Very few techniques (Chang and Lee, 2003; Leung and Khan, 2006a, b; Li and Lee, 2009; Ye *et al*., 2005; Mozafari *et al*., 2008) find the exact set of recent frequent patterns from a data stream.

Li and Lee (2009), the researchers proposed an Apriori-based algorithm, called MFI-TransSW which finds complete set of recent frequent patterns by using bit-sequences to keep track of the occurrence of all items in the transactions of the current fixed-sized sliding window. To remove old data and to reflect the inclusion of new data it performs a bit-wise left-shift operation for all bit-sequences. This approach is based on transaction-sensitive sliding window where the bit-sequence update operation is performed at the arrival of every single transaction. The MFI-TransSW applies the level-wise candidate-generation-andtest methodology to find the complete set of recent frequent patterns from the current window. Therefore, it suffers from the Apriori (Agrawal *et al*., 1993) limitation of huge candidate pattern generation, especially when mining stream data that contain large number of and/or long frequent patterns and/or with lower support thresholds. Furthermore, the transaction-by-transaction update mechanism may limit its performance when stream flows at high speed. Again, since the approach maintains the bit-sequence information in full for all items in the window, it fails to achieve memory efficiency when the window contains large number of transactions and distinct items which is very common in data stream environment. Even though MFITransSW discovers recent frequent patterns from a data stream, it differs significantly from the proposed technique in both mining approach and data processing strategy.

The other algorithm DSTree (Leung and Khan, 2006) which discovers exact frequent patterns from a data stream. DSTree uses a sliding window mechanism in which the window is divided into a fixed number of equalsized, non-overlapping batches of transactions. A canonical ordered prefix-tree structure is used to store the current window information. Each node in the tree maintains a list to explicitly store its frequency count in each batch. To avoid tree traversal during extraction of the old batch information from the tree, DSTree keeps track of the last visited batch at each node by using an additional pointer to the last updated batch number. To reflect the sliding of window it shifts the contents of frequency lists in the nodes. Upon a mining request, after capturing the information for a full window, an FP-growth-based mining technique is used to mine the complete set of frequent patterns from the tree (Fig. 2).

The DSTree, however, has several limitations. First, because it stores items in canonical order, it does not guarantee a highly compact tree structure which is essential when handling stream data to avoid massive

storage overhead, to reduce the search space and to accelerate the FP-growth-based mining operation. Second, DSTree maintains the batch information (i.e., frequency count) in each node with the help of a list of frequency counts; this list at each node further increases the tree size. Third, another storage overhead issue with DSTree is the need to maintain an extra batch pointer at each node to indicate the last-visited batch for this node. Fourth, as described by Leung and Khan (2006a, b), during the tree update phase DSTree does not visit all nodes in the tree (i.e., it avoids traversing the whole tree) and does not perform shifting the frequency counts list at each node in practice. Therefore, it does not perform the frequency list update operation for the nodes which are not visited during the new incoming batch. Such tree update operation may leave some invalid nodes in the DSTree structure for the current window. DSTree may carry over some expired information that produces some 'garbage' nodes. Such 'garbage' nodes in the current DSTree will obviously increase the tree size. Fifth, due to the enormous number of 'garbage' nodes, the total number of nodes in the tree may become unmanageable if the knowledge in the stream changes over time. Sixth, to ensure exact and consistent mining, before executing the mining algorithm, the tree needs to be 'cleaned' to get rid of the 'garbage' nodes (i.e., deleting all the nodes that are no longer valid for the current window and then adjusting the tree structure to update the frequency count list) which may be a costlier task than traversing the tree once to refresh its content after each batch. Therefore, the size and mining efficiency of a DSTree are highly dependent on the distribution of data in transactions because the frequency-independent item ordering of the tree structure may not provide as much prefix sharing as possible among all the paths in the tree structure. As a result, DSTree is likely to have a much extended mining time compared to tree structures arranged according to frequency-descending item ordering. Furthermore, DSTree was constructed based on the assumption of no main memory limitation which is unrealistic when handling very large amounts of data, such as data streams.

In contrast, CPS-tree algorithm maintains exactly the same information about the stream data as DSTree does by storing it in an FP-tree-like highly compact tree structure and, thereby, ensures a more storage-efficient data structure. The highly compact tree structure offers an efficient FP-growth-based mining platform. Further, storage efficiency is achieved by maintaining the list of frequency counts only at the last node of each path representing a transaction, instead of keeping this information at each node. Moreover, the CPS-tree does not need the extra batch pointer at each node to keep track of the last update. The CPS-tree constantly updates itself by extracting the expired transactions after each slide of the window, guaranteeing that the tree will not contain 'garbage' nodes and constantly ensuring a ready-to-mine tree status.

Like FP-tree, each node in a CPS-tree explicitly maintains parent, children and node traversal pointers and a support counter to record the total frequency of the node in the path. In addition, each tail-node maintains a pane-counter. The structures of an ordinary node and a tail-node are shown in Fig. 3. An example to illustrate the construction of a CPS-tree from stream data. Figure 4

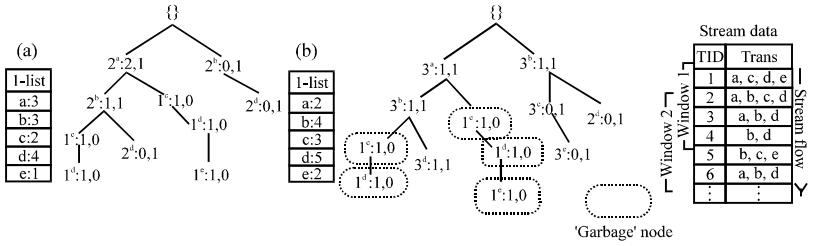

Fig. 2: DSTree for the stream data: a) DSTree at window 1 and b) DSTree at window 2

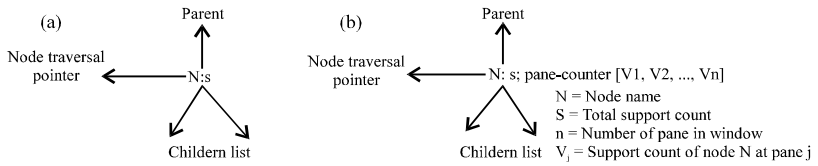

Fig. 3: Nodes of a CPS-tree: basic structure: a) An ordinary node and b) A tail-node

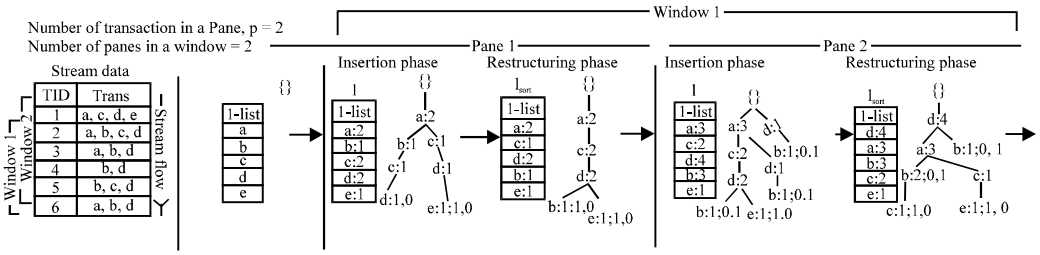

Fig. 4: Construction of the CPS-tree: a) Data stream; b) Initial empty Cps-tree; c) CPs tree after inserting Pane 1; e) CPs tree after inserting Pane 1; d) CPs tree after restructuring Pane 1 and f) CPs tree after restructuring Pane 1 and 2, i.e., at Window 1

shows a data stream (Fig. 4a) with corresponding transaction IDs and a step-by-step construction procedure for a CPS-tree (Fig. 4b-f).

CONCLUSION

In this study, we provide a survey of research on mining data streams. We focus on frequent pattern mining and have tried to cover both early and recent literature related to mining association rule over window sliding model or landmark window model. In particular, we have discussed in detail a number of pattern growth based algorithms on mining over data streams. Moreover, we have addressed the merits and the limitations and presented an analysis of the algorithms which can provide insights for end-users in applying or developing an appropriate algorithm for different streaming environments and various applications.

We believe that as many new streaming applications and sensor network applications are becoming more mature and popular, streaming data and sensor data are also becoming richer.

More high-speed data streams are generated in different application domains such as millions of transactions generated from retail chains, millions of calls from telecommunication companies, millions of ATM and credit card operations processed by large banks and millions of hits logged by popular web sites. Mining techniques will then be very significant in order to conduct advanced analysis such as determining trends and finding interesting patterns, on streaming data. It is our intention to present this survey to simulate interests in utilizing and developing the previous studies into emerging applications.

REFERENCES

Agrawal, R., T. Imielinski and A. Swami, 1993. Mining association rules between sets of items in large databases. Acm. SIGMOD. Rec., 22: 207-216.

Chang, J. and W. Lee, 2003. Finding recent frequent itemsets adaptively over online data streams. Proceedings of the International Conference on Knowledge Discovery and Data Mining, August 24-27, 2003, Washington, DC., USA., pp: 487-492.

Chang, J.H. and W.S. Lee, 2005. estWin: Online data stream mining of recent frequent itemsets by sliding window method. Inform. Sci., 31: 76-90.

Chen, G. and Q. Wei, 2002. Fuzzy association rules and the extended mining algorithms. Inf. Sci., 147: 201-228.

Chi, Y., H. Wang, P.S. Yu and R.R. Muntz, 2006. Catch the moment: Maintaining closed frequent itemsets over a data stream sliding window. Knowledge Inform. Syst., 10: 265-294.

Gaber, M.M., A. Zaslavsky and S. Krishnaswamy, 2005. Mining data streams: A review. ACM. Sigmod Rec., 34: 18-26.

Giannella, C., J. Han, J. Pei, X. Yan and P.S. Yu, 2003. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In: Next Generation Data Mining, Kargupta, H., A. Joshi, K. Sivakumar and Y. Yesha (Eds.). CRC Press, USA., pp: 191-212.

Han, J., H. Cheng, D. Xin and X. Yan, 2007. Frequent pattern mining: Current status and future directions. Data Ming Knowl. Discovery, 15: 55-86.

Han, J., J. Pei and Y. Yin, 2000. Mining frequent patterns without candidate generations. ACM SIGMOD Record, 29: 1-12.

Hsu, P.Y., Y.L. Chen and C.C. Ling, 2004. Algorithms for mining association rules in bag databases. Inf. Sci., 166: 31-47.

Hu, T., S.Y. Sung, H. Xiong ans Q. Fu, 2008. Discovery of maximum length frequent itemsets. Inf. Sci., 178: 69-87.

Jiang, N. and L. Gruenwald, 2006. Research issues in data stream association rule mining. ACM. SIGMOD. Rec., 35: 14-19.

Lee, A.J. and C.S. Wang, 2007. An efficient algorithm for mining frequent inter-transaction patterns. Inf. Sci., 177: 3453-3476.

Lee, A.J., R.W. Hong, W.M. Ko, W.K. Tsao and H.H. Lin, 2007. Mining spatial association rules in image databases. Inf. Sci., 177: 1593-1608.

Lee, Y.S. and S.J. Yen, 2008. Incremental and interactive mining of web traversal patterns. Inform. Sci., 178: 287-306.

Leung, C.K.S. and Q.I. Khan, 2006a. Efficient mining of constrained frequent patterns from streams. Proceedings of the 10th International Symposium on Database Engineering and Applications (IDEAS'06), December 11-14, 2006, IEEE, Canada, ISBN:0-7695-2577-6, pp: 61-68.

Leung, C.K.S. and Q.I. Khan, 2006b. DSTree: A tree structure for the mining of frequent sets from data streams. Proceedings of the 6th International Conference on Data Mining (ICDM 06), December 18-22, 2006, IEEE, Canada, ISBN:0-7695-2701-7, pp: 928-932.

Li, H.F. and S.Y. Lee, 2009. Mining frequent itemsets over data streams using efficient window sliding techniques. Expert Syst. Applic., 36: 1466-1477.

Li, J., D. Maier, K. Tufte, V. Papadimos and P.A. Tucker, 2005. No pane, no gain: Efficient evaluation of sliding-window aggregates over data streams. ACM. SIGMOD. Rec., 34: 39-44.

Lin, C.H., D.Y. Chiu, Y.G. Wu and A.L.P. Chen, 2005. Mining frequent itemsets from data streams with a time-sensitive sliding window. Proc. SIAM Int. Conf. Data Min., 119: 68-79.

Manku, G.S. and R. Motwani, 2002. Approximate frequency counts over data streams. Proceedings of the 28th International Conference on Very Large Databases, August 20-23, 2002, New York, USA., pp: 346-357.

Mozafari, B., H. Thakkar and C. Zaniolo, 2008. Verifying and mining frequent patterns from large windows over data streams. Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE 2008), April 7-12, 2008, IEEE, Los Angeles, California, ISBN:978-1-4244-1836-7, pp: 179-188.

Shen, L., H. Shen and L. Cheng, 1999. New algorithms for efficient mining of association rules. Inf. Sci., 118: 251-268.

Silvestri, C. and S. Orlando, 2007. Approximate mining of frequent patterns on streams. Intell. Data Anal., 11: 49-73.

Tanbeer, S.K., C.F. Ahmed, B.S. Jeong and Y.K. Lee, 2008. Efficient single-pass frequent pattern mining using a prefix-tree. Inform. Sci., 179: 559-583.

Tsay, Y.J. and Y.W.C. Chien, 2004. An efficient cluster and decomposition algorithm for mining association rules. Inf. Sci., 160: 161-171.

Tsay, Y.J., T.J. Hsu and J.R. Yu, 2009. FIUT: A new method for mining frequent itemsets. Inf. Sci., 179:1724-1737.

Wang, C.Y., S.S. Tseng and T.P. Hong, 2006. Flexible online association rule mining based on multidimensional pattern relations. Inf. Sci., 176: 1752-1780.

Wang, F.H., 2008. On discovery of soft associations with most fuzzy quantifier for item promotion applications. Inf. Sci., 178: 1848-1876.

Ye, F.Y., J.D. Wang and B.L. Shao, 2005. New algorithm for mining frequent itemsets in sparse database. Proceedings of the 2005 International Conference on Machine Learning and Cybernetics, Vol. 3, August 18-21, 2005, IEEE, Nanjing, China, ISBN:0-7803-9091-1, pp: 1554-1558.

Yu, J.X., Z. Chong, H. Lu and A. Zhou, 2004. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. Proceedings of the 30th nternational Conference on Very Large Data Bases,Aug. 31-Sept. 3, Toronto, Canada, pp: 204-215.

Yu, J.X., Z. Chong, H. Lu, Z. Zhang and A. Zhou, 2006. A false negative approach to mining frequent itemsets from high speed transactional data streams. Inf. Sci., 176: 1986-2015.

Zhang, S., J. Zhang and C. Zhang, 2007. EDUA: An efficient algorithm for dynamic database mining. Inf. Sci., 177: 2756-2767.

Zhi-Jun, X., C. Hong and C. Li, 2006. An efficient algorithm for frequent itemset mining on data streams. Proceedings of the Industrial Conference on Data Mining (ICDM) 2006, July 14-15, 2006, Springer, Leipzig, Germany, pp: 474-491.

Zhu, Y. and D. Shasha, 2002. StatStream: Statistical monitoring of thousands of data streams in real time. Proceeding of the 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China, pp: 358-369.