© March 2016 | IJIRT | Volume 2 Issue 10 | ISSN: 2349-6002

A survey on techniques for mining frequent patterns over data stream

Niyati M. Mevada, Ms. Jayna B. Shah
Dept. of Computer Engineering ,
Sardar Vallabhbhai Patel Institute of technology, Vasad-388306, Gujarat , India

***Abstract*—Mining frequent itemsets in a data stream proves to be a difficult problem, as itemset arrives in rapid succession and storing parts of the stream is typically impossible. Finding frequent pattern from data streams have been of importance in many applications such as stock market prediction, sensor data analysis, e-business, network traffic analysis and telecommunication data analysis. In this paper, few recent and popular methods for extracting patterns from stream data have been studied. Also a comparative study of different methods with reference to the conditions in which they work, their benefits and drawbacks of these methods are presented in this work.**

***Index Terms*—frequent itemsets , data stream**

I. INTRODUCTION

A data stream is a continuous, huge, fast changing, rapid, infinite sequence of data elements. We can say data stream is an ordered sequence of elements that arrives in timely order. It is assumed that the stream can only be scanned once and hence if an item is passed, it cannot be revisited, unless it is stored in main memory. Different from data in traditional static datasets, data streams are continuous, unbounded, usually come with high speed and have a data distribution that often changes with time [1]. It is often refer to as streaming data. In this , it uses multiple segments for handling different size of windows over data streams. Storing these segments in a data structure, the usage of memory can be optimized. Many applications generate large amount of data streams in real time, such as sensor data generated from sensor networks, online transaction flows in retail chains, Web record and click-streams in Web applications, call records in telecommunications, performance measurement in network monitoring and traffic management.

Data streams can be further classified into offline data streams and online data streams. Offline data streams are characterized by regular bulk arrivals [4], such as a bulk addition of new transactions as in a data warehouse system. Online data streams are characterized by real-time updated data that come one by one in time, such as an a continuously generated transaction as in a network monitoring system. Bulk data processing is not possible for one streaming data. Due to the characteristics of data streams, there are some inherent challenges for mining streaming data [8]. First, each data element of stream should be examined at most once. Second, the memory usage in the process of mining data streams should be bounded even though new data elements are continuously generated from the streams. Third, each element due to the characteristics of data streams, there are some inherent challenges for mining streaming data. Fourth, the analytical outputs of the stream should be instantly available when the user requested. Finally, the errors of outputs should be constricted as small as possible. The continuous characteristic of streaming data makes it essential to use the algorithms which require only one scan over the stream for knowledge discovery. The huge nature of stream makes it impossible to store all the data into main memory or even in secondary storage. This motivates the design of a summary data structure with small footprints that can support both one-time and continuous queries [5]. In other words, one-pass data stream mining algorithms have to sacrifice the correctness in the analytical results by allowing some counting errors. Consequently, previous multiple-pass data mining algorithms studied for static datasets are not feasible for mining data streams.

Frequent itemset mining is a KDD technique which is the basic of many other techniques, such as association rule mining, sequence pattern mining, classification, and clustering. It is impossible to maintain all the elements of data streams [3]. This rapid generation of continuous streams of information has challenged our storage, computation and communication capabilities in computing systems. Data Stream mining refers to informational structure extraction as models and patterns from continuous data streams [5]. Data Streams have different challenges in many aspects, such as computational, storage, querying and mining. Data stream mining differs from traditional data mining since its input of mining is data streams, while the latter focuses on mining (static) databases. Compared to traditional databases, mining in data streams has more constraints and requirements. The mining task should proceed normally and offer acceptable quality of This result, one good stream mining algorithm to possess efficient performance and high throughput [7]. Slight approximate errors occurred in the mining result is usually acceptable by the user[2] [4].

II. LITERATURE SURVEY

Various methods to find frequent pattern identification from data streams are described and discussed here.

A. Bounded Frequent Itemsets Stream [BFI-Stream]

Bounded Frequent Itemsets stream (abbreviated as BFI-stream) algorithm, which uses a prefix-tree based structure, called BFI-tree, to maintain all accurate frequent itemsets from sliding windows over data streams. By monitoring the boundary between frequent itemsets and infrequent itemsets, it restricts the update process on a small part of the tree. Mining all frequent itemsets with accurate frequencies is just to traverse the tree. It is time efficient even when the user-specified minimum support threshold is small .It is proposed to mine all accurate frequent itemsets from sliding windows over data streams. It uses a prefix-tree based structure, called BFI-tree, to maintain all frequent itemsets in the sliding window. It is time efficient in both update and mining processes [8]. The BFI tree, consist of three important phases on the tree, such as constructing the BFI-tree in the window initialization phase, updating the BFI-tree in the sliding phase, and mining the tree at any time after the initialization phase.

In the initialization phase, we use the first transactions to build the BFI-tree. To build a BFI-tree, we first create a root node. Second, we create child nodes. Then we call the building procedure recursively on each child node . After the initialization phase, the sliding phase comes. While the new transaction comes into the sliding window, the oldest transaction is deleted. Because of the high-speed arrival, the BFI-tree should be updated quickly, and only a small part of the tree is affected [8] . All frequent itemsets in the sliding window are maintained in the BFI-tree, so the mining process is very simple

B. Compact Sliding Window [CSW]

An efficient method to discover the set of latest frequent patterns from dynamic data stream using sliding window model and CSW (compact sliding window) tree [9]. In addition, not only support conditions but also improved resulting maximal frequent patterns more quickly. Extensive experiments report that this method has outstanding performance in terms of runtime, memory usage as compare to previous algorithms.

The system architecture is shown below in figure 1 .The flow also shows the four main stages in compact sliding window.

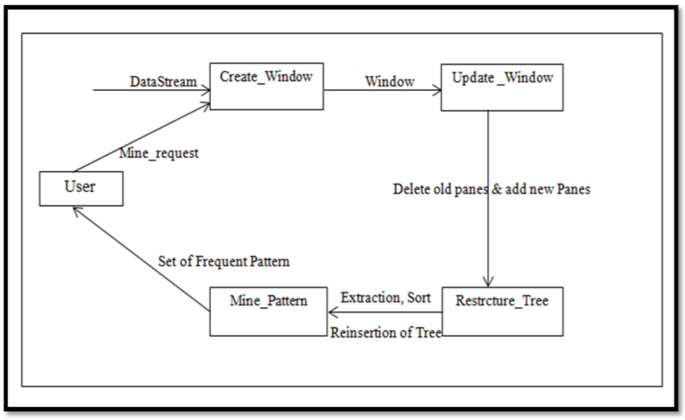

Figure : 1

In system architecture when user make the mine request then system first load the transactions from data stream within a single scan, after the successful database connection, the transactions are imported into compact sliding window tree with support value, the imported data pass as input to create window module which create the window and specify the each item count and it’s weight, our system support for mining latest frequent pattern from data stream due to this reason, here we use sliding window in the next update window module which take the input from create window module and delete the old panes and add new panes to mine latest frequent patterns then updated window pass as input to restructure module which perform the extraction, sorting and reinsertion operation then pass the sorted, extracted data as input to final mine module which mine all the transactions over dynamic data streams and find the latest frequent patterns which are greater than threshold value and finally display the set of latest frequent patterns over the dynamic data stream [9].

C. Data Stream Mining for Frequent Itemsets [DSMFI]

A new single-pass algorithm, called DSMFI (Data Stream Mining for Frequent Itemsets), to mine all frequent itemsets over the entire history of data streams. DSM-FI has three major features, namely single streaming data scan for counting itemsets frequency information, extended prefix-tree-based compact pattern representation, and top-down frequent itemset discovery scheme [10]. Our performance study shows that

DSM-FI outperforms the well-known algorithm Lossy Counting in the same streaming environment. The algorithm *DSM-FI* (data stream mining for frequent itemsets) for online mining of frequent itemsets in a landmark window of a continuous data stream. The DSM-FI algorithm consists of four steps.

The DSM-FI algorithm reads a basic window of transactions from the buffer in main memory, and sorts the items of transaction in lexicographical order .DSM-FI algorithm constructs and maintains an in-memory prefix-tree base summary data structure, called SFI-forest (summary frequent itemset forest)[6] . DSM-FI algorithm prunes the infrequent information from the current SFI-forest. DSM-FI finds the frequent itemsets from the current SFI-forest.

D. Mining Frequent Itemsets over Transaction Sliding Window [MFI-TransSW]

An efficient one-pass algorithm, called MFI-TransSW (Mining Frequent Itemsets over Transaction Sliding Window), to mine the set of all frequent itemsets in data streams with a transaction-sensitive sliding window. An effective bit-sequence representation of items is used in the proposed algorithm to reduce the time and memory needed to slide the windows. The experiments show that the proposed algorithm not only attain highly accurate mining results, but also run significant faster and consume less memory than existing algorithms for mining frequent itemsets over recent data streams. An effective bit- sequence representation of items is used to reduce the time and memory needed to slide the windows [1]. Each transaction in the data stream can be examined only once, making traditional multiple-scan approaches infeasible. The consumption of memory space should be confined in a range. The data characteristic of incoming stream may be unpredictable, the mining task should proceed normally and offer acceptable quality of results. Compared with other sliding window based mining techniques, we save memory and improve speed by dynamically maintaining all transactions in the current sliding window by using an effective bit-sequence representation of items[4]. The proposed MFI-TransSW algorithm consists of three phases: window initialization, window sliding and pattern generation.

The proposed bit-sequence representation of item is used to reduce the time and memory needed to slide the windows in the following phases. Second, MFI-TransSW uses the left bit-shift technique to slide the windows efficiently in the window sliding phase. Finally, the complete set of frequent itemsets within the current sliding window is generated by a level-wise method in the pattern generation phase [1]. Experimental studies show that the proposed algorithm not only attain highly accurate mining results, but also run significant faster and consume less memory than do existing algorithms for mining frequent itemsets over data streams with a sliding window.

III. COMPARATIVE STUDY

A brief summary of important data stream mining algorithms are compared in below table .

SR NO	***Algorithm Name***	***Window Used***	***Benefits***	***Drawbacks***
1.	BFI-Stream	Sliding window	BFI-stream uses BFI-tree, which is prefix-tree based and highly compact, to maintain all accurate frequent itemsets and even mining process becomes simple	An additional scan is required on all transactions in the sliding window to compute their frequencies
2.	CSW	Sliding window	CSW uses bottom-up FPgrowth Mining technique to generate the set of recent frequent patterns. Thus the mining operation is highly efficient.	Every time a new item is arrived it reconstructs the tree. So it requires high execution time.
3.	DSMFI	Land-mark window	It mines all frequent itemsets over the entire history of data streams	The algorithm uses landmark window . This technique is based on time slots & managing timing becomes tedious task.

SR NO	***Algorithm Name***	***Window Used***	***Benefits***	***Drawbacks***
4.	MFI-TransSW	Sliding window	MFI-TransSW applies levelwise candidate-generation to find accurate set of recent frequent patterns from the current window	MFI-TransSW's uses bit sequences representation ,for mining all frequent itemsets which will cost much more time

STREAM MINING ALGORITHMS ARE COMPARED IN TABLE1

IV. CONCLUSION

We have isolated a number of issues in data streams, the purpose of this paper is to introduce the process of mining frequent patterns in data streams and particularly analyze the algorithms for frequent pattern mining in data streams . We have studied the concept of data streams and how the frequent patterns are mined over data streams. In addition to this, we have analyzed the different existing research works of frequent pattern mining over data streams. Merits and demerits of the existing works are also discussed. By comparing these algorithms, we will be able to select which kind of algorithm is suitable for mining frequent patterns over data stream. In future, we can develop new techniques and algorithms for finding frequent patterns over data streams which helps to overcome the drawbacks of the existing techniques.

REFERENCES

1. Mr. Velusamy., Ms. Shobana. , Ms. Saranya. "Efficient Mechanism to Discover Frequent Pattern over Online Data Streams" International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 12, December – 2013

2. R. AGRAWAL, T. Imielinski, and A. Swami. "Mining Association Rules between Sets of Items in Large Databases". In Proceedings of the 2008 International Conference on Management of Data, pp. 207-216, 2008

3. S. Muthukrishnan , "Data streams: algorithms and applications". Proceedings of the fourteenth annual ACMSIAM symposium on discrete algorithms, 2009.

4. Li, H.-F., Lee, S.-Y., Shan, M.-K. (2005a). "Online mining (recently) maximal frequent itemsets over data streams". In Prooceedings of the IEEE RIDE.

5. Chang, J., & Lee, "A sliding window method for finding recently frequent itemsets over online data streams". Journal of Information Science and Engineering, 2005.
6. Yang, C., Li, Y., Zhang, C., & Hu, Y., "A novel algorithm of mining maximal frequent pattern based on projection sum tree", Fuzzy Systems and Knowledge Discovery, vol. 1, pp.458–462, 2007

7. S. Muthukrishnan , "Data streams: algorithms and applications". Proceedings of the fourteenth annual ACMSIAM symposium on discrete algorithms, 2009.

8. Kun Li,, Yong-yan Wang, Manzoor Ellahi, Hong-an Wang "Mining Recent Frequent Itemsets in Data Streams" Fifth International Conference on Fuzzy Systems and Knowledge Discovery , IEEE-2008.
9. Rahul Anil Ghatage "Frequent Pattern Mining Over Data Stream Using Compact Sliding Window Tree & Sliding Window Model" nternational Research Journal of Engineering and Technology (IRJET) Volume: 02 , July-20

10. Hua-Fu Li , Man-Kwan Shan , Suh-Yin Lee. "DSM-FI : an efficient algorithm for mining frequent itemsets in data stream" Knowl Inf Syst (2008) 17:79–97Springer