International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 1, Issue 1, May-June 2012 ISSN 2278-6856

An Overview of Mining Frequent Itemsets Over Data Streams Using Sliding Window Model

K Jothimani[1], Dr Antony Selvadoss Thanamani[2]

[1] Research Scholar, Research Department of Computer Science,
NGM College, 90, Palghat Road, Pollachi - 642 001
Coimbatore District, Tamilnadu, INDIA
Email: jothi1083@yahoo.co.in

[2] Professor and Head, Research Department of Computer Science,
NGM College, 90, Palghat Road, Pollachi - 642 001
Coimbatore District, Tamilnadu, INDIA
Email: selvdoss@yahoo.com

Abstract: ***Mining frequent itemsets over data streams is an emergent research topic in recent years. In data streams, new data are continuously coming as time advances. It is costly even impossible to store all streaming data received so far due to the memory constraint. 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. Storing large parts of the stream, however, is not possible because the amount of data passing by is typically huge. The previous approaches, accept only one minimum support and using fixed window length. In reality, the minimum support is not a fixed value for the entire stream of transactions. In this paper we propose a new method, which used multiple segments for handling different size of windows over data streams. By storing these segments in a data structure, the usage of memory can be optimized.***

Keywords: Data streams, Mining Frequent Itemsets, Sliding Window.

1. INTRODUCTION

Frequent itemset mining is a KDD technique which is the basic of many other techniques, such as association rule mining, sequence pattern mining, classification, clustering and so on. A data stream is a massive unbounded sequence of data elements continuously generated at a rapid rate. Due to this reason, it is impossible to maintain all the elements of data streams [1]. This rapid generation of continuous streams of information has challenged our storage, computation and communication capabilities in computing systems. The main challenge is that 'data-intensive' mining is constrained by limited resources of time, memory, and sample size.

Data Stream mining refers to informational structure extraction as models and patterns from continuous data streams. Data Streams have different challenges in many aspects, such as computational, storage, querying and mining. Based on last researches, because of data stream requirements, it is necessary to design new techniques to replace the old ones. Traditional methods would require the data to be first stored and then processed off-line using complex algorithms that make several pass over the data, but data stream is infinite and data generates with high rates, so it is impossible to store it [12].

Data from sensors like weather stations is an example of fixed-sized data, whereas again, market basket data are an example of variable size data, because each basket contains a different number of items. By contrast, sensor measurements have a fixed size, as each set of measurements contains a fixed set of dimensions, like temperature, precipitation, etc.

A typical approach for dealing is based on the use of so-called sliding windows.

The algorithm keeps a window of size W containing the last W data items that have arrived (say, in the last W time steps). When a new item arrives, the oldest element in the window is deleted to make place for it. The summary of the Data Stream is at every moment computed or rebuilt from the data in the window only. If W is of moderate size, this essentially takes care of the requirement to use low memory. The type of objects i.e. windows in a stream impacts the way the data stream is processed. This is due to two facts: We have to handle windows of different types differently, and we are able to tailor the processing to the specific properties of the objects. Hence, for our focus, interesting properties are the size of the windows, what information the objects represent and how they relate to other windows in the stream.

2. RELATED WORK

Recently proposed mining approaches for event logs have often been based on some well-known algorithm for mining frequent itemsets (like Apriori or FPgrowth).In this section we will discuss the frequent itemset mining problem and prominent algorithms for finding optimistic solution in addressing this problem.

The sliding window method processes the incoming stream data transaction by transaction. Each time when a new transaction is inserted into the window, the itemsets contained in that transaction are updated into the data structure incrementally. Next, the oldest transaction in

the original window is dropped out, and the effect of those itemsets contained in it is also deleted. The sliding window method [4] also has a periodical operation to prune away unpromising itemsets from its data structure, and the frequent itemsets are output as mining result whenever a user requests.

Two types of sliding widow, i.e., transaction- sensitive sliding window and time-sensitive sliding window, are used in mining data streams. The basic processing unit of window sliding of first type is an expired transaction while the basic unit of window sliding of second one is a time unit, such as a minute or an hour. In the damped window model, recent sliding windows are more important than previous ones.

As long as the window size is reasonably large, and the conceptual drifts in the stream is not too dramatic, most itemsets do not change their status (from frequent to non-frequent or from non-frequent to frequent).[18] In other words, the effects of transactions moving in and out of a window offset each other and usually do not cause change of status of many involved nodes.

To find frequent itemsets on a data stream, we maintain a data structure that models the current frequent itemsets. We update the data structure incrementally. The combinatorial explosion problem of mining frequent itemsets becomes even more serious in the streaming environment. As a result, on the one hand, we cannot afford keeping track of all itemsets or even frequent itemsets, because of time and space constraints. Thus, the challenge lies in designing a compact data structure which does not lose information of any frequent itemset over a sliding window.

3. PROBLEM DESCRIPTION

Let I = {x1, x2, …, xz} be a set of items (or attributes). An itemset (or a pattern) X is a subset of I and written as X =xi xj…xm. The length (i.e., number of items) of an itemset X is denoted by |X|. A transaction, T, is an itemset and T supports an itemset, X, if X⊆T. A transactional data stream is a sequence of continuously incoming transactions. A segment, S, is a sequence of fixed number of transactions, and the size of S is indicated by s. A window, W, in the stream is a set of successive w transactions, where w ≥ s.

A sliding window in the stream is a window of n number of most recent w transactions which slides forward for every transaction or every segment of transactions. We adopt the notation Il to denote all the itemsets of length l together with their respective counts in a set of transactions (e.g., over W or S). In addition, we use Tn and Sn to denote the latest transaction and segment in the current window, respectively. Thus, the current window is either W = < Tn-w+1, …, Tn > or W = < Sn-m+1, …, Sn >, where w and m denote the size of W and the number of segments in W, respectively.

A stream of itemsets F = $\{f_i\}_{i \square N}$ is an infinite sequence of itemsets that evolves continuously. It is to be assumed that only a small part of it can be kept in memory. The window W of size w at time t is the sequence W = $\{f_{i1}, f_{i2}, \ldots. f_{in}\}$ such that □ f_i € F \ W, i < t or i > t + w . The current window of size w of a stream F is the window beginning at time t - w + 1 where t is the position of the last itemset appeared in the stream. This window includes the most recent itemsets of the stream.

4. PROBLEM DESCRIPTION

The following definition describes the mining frequent itemsets over data streams.

Definition: 1 (Mining a stream of itemsets). Given a threshold α, we say that a sequence S is frequent in a window W of size w iff $supp_W(S) \geq \alpha$. Mining a stream of itemsets consists in extracting at every time instant, all the frequent sequences in the most recent sliding window. In approaches that considers multiple parallel streams [8, 16, 18], the support of a sequence is usually defined as the number of streams in which this sequence appears. Here we consider a single stream and the support of a sequence is the number of instances of this sequence in the current window corresponding to the most recent data.

In this research, we employ a prefix tree which is organized under the lexicographic order as our data structure, and also processes the growth of itemsets in a lexicographic-ordered away. A superset of an itemset X is the one whose length is above |X| and has X as its prefix. We define Growthl(X) as the set of supersets of an itemset X whose length are l more than that of X, where l ≥ 0. The number of itemsets in Growthl(X) is denoted by |Growthl(X)|.

We adopt the symbol cnt(X) to represent the count-value (or just count) of an itemset X. The count of X over W, denoted as cntW(X), is the number of transactions in W that support X. So cntS(X) represents the count of X over a segment S. Given a user-specified minimum support threshold (ms), where 0 < ms ≤ 1, we say that X is a Segment based Frequent Itemset (FI) over W if cntW(X) ms×w, otherwise X is an infrequent itemset (IFI). The SFI and IFI over S are defined similarly to those for W.

Given a data stream in which every incoming transaction has its items arranged in order, and a changeable value of ms specified by the user, the problem of mining SFIs over a sliding window in the stream is to find out the set of frequent itemsets over the window at different slides.

SFI Algorithm:
FREQUENT(*k*)
n ‹– 0;
T‹– ø;
foreach *i* **do**
n ‹– n + 1;
if *i* ∈*T* **then**
c_i ‹– c_i + 1;
else if |T| < *k–1* **then**
T‹– □ {i};
c_i ‹– 1;

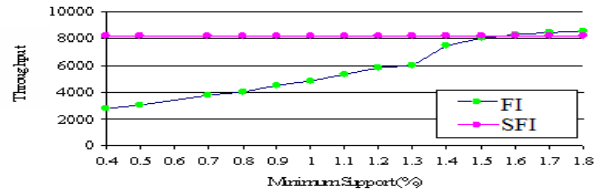
else forall $j \in T$ **do**
$c_j \leftarrow c_j - 1$;
if $cj = 0$ **then** $T \leftarrow T \setminus \{j\}$;

As mentioned in Section 3, we further divide the sliding window into *m* equal-size segments of *s* transactions, and process the sliding of window incrementally in a segment-based manner. The data structure we employ to maintain the summary information is a *lexicographic-ordered prefix tree* modified from the one in [12]. This tree structure mainly maintains *I*1, *I*2, and SFIs of 2-itemsets over the current window of a data stream, also in a segment-based fashion. For each itemset *X* belonging to *I*1 or *I*2, the corresponding node in the tree includes a circular array of size *m*, which corresponds to the *m* segments of the sliding window, and *X*'s count over the current window is recorded respectively in these *m* fields.

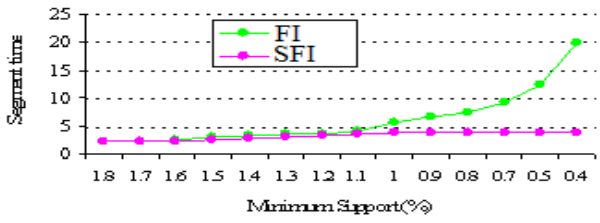
5. Experimental Results

The goal of our experiments is to examine the time and space behavior of our algorithm over a variety of parameter settings. More specially our goals are to determine the following: the effective processing rate, the existence of a time and space usage upper-bound, and the time and space sensitivity. Since the purpose of our algorithm is to process an unbounded data stream.
The test datasets adopted in our experiments. were downloaded from the website of *FIMI Repository* [14], while others were generated using the *IBM's synthetic data generator* [15]. Every dataset has 1000 different attributes and consists of 100 thousands of transactions. The size of sliding window in our experiments is set to 50,000 transactions for both methods.

The first experiment investigates the efficiency with respect to *throughput* and *average window-sliding time* of both methods. Here throughput is measured as the number of transactions processed per second by the algorithms. We report the experimental result in the following Fig.1



(a) Throughput



(b) Average window-sliding time
Figure 1 Scalability on T15.I6.D100K with varying minimum support threshold0

The second experiment evaluates the *scalability* of FI and SFI with varying the value of *ms*. We measure the throughput and average window-sliding time of both methods, which are similar to those in the previous experiment. The dataset adopted is T16.I8.D100K, and we vary *ms* from 1.5% down to 0.2%.

6. Conclusion

In this paper, we study the problem of mining frequent itemsets over the sliding window of a transactional data stream. Based on the improvement theory of *An Efficient approximate Inclusion–Exclusion*, we devise and propose an algorithm called SFI for finding frequent itemsets through an approximating approach.

SFI divides the sliding window into *segments* and handles the sliding of window in a segment-based manner. It capable of approximating itemsets dynamically by choosing different parameter-values for different itemsets to be approximated. We analyze all the factors which is related for mining frequent itemsets over datastreams. In future we can improve this algorithm for large and variable windows

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of the 1993 International Conference on Management of Data, pp. 207-216, 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499, 1994

[3] J.H. Chang & W.S. Lee, "A sliding window method for finding recently frequent itemsets over online data streams', Journal of Information Science and Engineering, 20(4), 2004, pp. 753–762

[4] Y. Zhu & D. Shasha, "Stat Stream: statistical monitoring of thousands of data streams in real time", Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 358–369.

[5] K Jothimani, Dr Antony Selvadoss Thanmani, "MS: Multiple Segments with Combinatorial Approach for Mining Frequent Itemsets Over Data Streams", IJCES International Journal of Computer Engineering Science, Volume 2 Issue 2 ISSN : 2250:3439.

[6] J.H. Chang & W.S. Lee, "A sliding window method for finding recently frequent itemsets over online data streams", Journal of Information science and Engineering, 20(4), 2004, pp. 753–762.

[7] J. Cheng, Y. Ke, & W. Ng," Maintaining frequent itemsets over high-speed data streams", Proc. 10th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Singapore, 2006, pp.462–467.

[8] C.K.-S. Leung & Q.I. Khan, "DSTree: a tree structure for the mining of frequent sets from data streams," Proc. 6th IEEE Conf. on Data Mining, Hong Kong, China, 2006, pp. 928–932.
[9] Y. Chi, H. Wang, P.S. Yu, & R.R. Muntz, "Moment: maintaining closed frequent itemsets over a stream sliding window", Proc. 4th IEEE Conf. on Data Mining, Brighton, UK, 2004, pp. 59–66.
[10] K.-F. Jea & C.-W. Li, "Discovering frequent itemsets over transactional data streams through an efficient and stable approximate approach, Expert Systems with Applications", 36(10), 2009, pp. 12323–12331.
[11] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference, was supported in part by the National Science Council in 2006.
[12] F. Bodon, "A fast APRIORI implementation", Proc. ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), 2003.
[13] Frequent Itemset Mining Implementations Repository (FIMI). Available: http://fimi.cs.helsinki.fi/
[14] Mahnoosh Kholghi, Mohammadreza Keyvanpour, "An Analytical Framework for Data Stream Mining Techniques Based on Challenges and Requirements", International Journal of Engineering Science and Technology (IJEST) ISSN: 0975-5462 Vol. 3 No. 3 Mar 2011.
[15] N. Jiang & L. Gruenwald, "CFI-Stream: mining closed frequent Itemsets in data streams", Proc. 12th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, Philadelphia, PA,USA, 2006, pp. 592–597.
[16] Quest Data Mining Synthetic Data Generation Code.
[17] Available:http://www.almaden.ibm.com/cs/projects /iis/hdb/Projects/datamining/datasets/syndata.htm
[18] H.F Li, S.Y. Lee, M.K. Shan, "An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams", In Proceedings of First International Workshop on Knowledge Discovery in Data Streams 9IWKDDS, 2004.
[19] P. Indyk, D. Woodruff, "Optimal approximations of the frequency moments of data streams", Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pp.202–208, 2005.

K Jothimani received her Bachelor degree in Computer Science from Bharathidasan University, Trichy in 2003. She received her Master degree in Computer Applications in 2008. She pursued her Master of Philosophy in Computer Science in the year 2009 from Vinayaka Missions University, Salem. Currently she is a research scholar of the Department of Computer Science, NGM College under Bharathiyar University, Coimbatore. She had three years of experience in the computer field in technical as well as non technical. She is a life member of Indian Society for Technical Education from the year 2009. Also she is a member of Computer Society of India(CSI). She has published more than ten papers in international and national conferences including standard journals. Her area of Interests Data mining, Knowledge Engineering and Image Processing.

Email: jothi1083@yahoo.co.in

Dr. Antony Selvadoss Thanamani is presently working as Professor and Head, Dept of Computer Science, NGM College, Coimbatore, India (affiliated to Bharathiar University, Coimbatore). He has published more than 150 papers in international/ national journals and conferences. He has authored many books on recent trends in Information Technology. His areas of interest include E-Learning, Knowledge Management, Data Mining, Networking, Parallel and Distributed Computing. He has to his credit 26 years of teaching and research experience. He is a senior member of International Association of Computer Science and Information Technology, Singapore and Active member of Computer Science Society of India, Computer Science Teachers Association, New York.

Email:- selvdoss@yahoo.com