DATA STREAM MINING ALGORITHMS – A REVIEW OF ISSUES AND EXISTING APPROACHES

A.Mala
Asst.Prof.,Dept. of Information Technology,
KNSK College of Engineering, Nagercoil
Email: mala.elango@gmail.com

F.Ramesh Dhanaseelan
Prof., Dept. of Computer Applications,
St. Xavier's Catholic College of Engg.,Nagercoil
Email: message_to_ramesh@yahoo.com

Abstract

More and more applications such as traffic modeling, military sensing and tracking, online data processing etc., generate a large amount of data streams every day. Efficient knowledge discovery of such data streams is an emerging active research area in data mining with broad applications. Different from data in traditional static databases, data streams typically arrive continuously in high speed with huge amount and changing data distribution. This raises new issues that need to be considered when developing association rule mining techniques for stream data. Due to the unique features of data stream, traditional data mining techniques which require multiple scans of the entire data sets can not be applied directly to mine stream data, which usually allows only one scan and demands fast response time.

Keywords
Closed frequent item set, Data Streams, Frequent item set, Mining Streams, Stream Mining algorithms

1.Introduction

A data stream is an ordered sequence of items that arrives in timely order. Different from data in traditional static databases, data streams [1] are continuous, unbounded, usually come with high speed and have a data distribution that often changes with time.

As the number of applications on mining data streams grows rapidly, there is an increasing need to perform association rule mining on stream data. For most data stream applications, there are needs for mining frequent patterns and association rules from data streams. Some key applications in various areas are listed below.

Performance monitoring*:* Monitor network traffic and performance, detect abnormality and intrusion, and predict frequency estimation of Internet packet streams sometimes used to find alarming incidents from data streams.

Transaction monitoring*:* Monitor transactions in retail stores, ATM machines, and financial markets Log record mining, mine patterns from telecommunication calling records, Web server log, etc. Association rule mining can also be applied to monitor manufacturing flows [2] to predict failure or generate reports based on web log streams

Sensor network mining*:* Mine patterns in streams coming from sensor networks or surveillance cameras and also estimate missing data [3].

2. Data Stream Classifications

Data streams can be classified into offline streams and online streams. Offline streams are characterized by regular bulk arrivals [4]. Among the above examples, generating reports based on web log streams can be treated as mining offline data streams because most of reports are made based on log data in a certain period of time. Other offline stream examples include queries on updates to warehouses or backup devices. Queries on these streams are allowed to be processed offline.

Online streams are characterized by real-time updated data that come one by one in time. Among the above examples, predicting frequency estimation of Internet packet streams is an application of mining online data streams because Internet packet streams is a real-time one packet by one packet process. Other online data streams are stock tickers, network measurements and sensor data. They have to be processed online and must keep up with the rapid speed of online queries. Once they arrive they are processed and then immediately discarded. In addition, unlike with offline data streams, bulk data processing [5] is not possible for online stream data.

Research on data streams started around 2000 when several data stream management systems were originated (e.g., the Brandeis AURORA project, the Cornell COUGAR project, and the Stanford STREAM project) to solve new challenges in applications such as network traffic monitoring, transaction data management, Web click streams monitoring, sensor networks, etc.

Because association rule mining plays an important role in these data stream applications, along with the development of data stream management systems, developing association rule mining algorithms for data streams has also become an important research topic.

3. General Issues in Data Stream Association Rule Mining

3.1 Two Sub-problems

Algorithms for association rule mining usually consist of two steps. The first step is to discover frequent item sets. In this step, all frequent item sets that meet the support threshold are discovered. The second step is to derive association rules. In this step, based on the frequent item sets discovered in the first step, the association rules that meet the confidence criterion are derived. Because the second step, deriving association rules, can be solved efficiently in a straightforward manner, most of researches focus mainly on the first step, i.e., how to efficiently discover all frequent item sets in data streams. Therefore, in the rest of this article, the focus will be on frequent item set mining in data streams.

3.2 Key Challenges

Frequent item set mining in general is already a challenging problem. For example, due to combinatorial explosion, there may be huge number of frequent item sets, and a main challenge is how to efficiently enumerate, discover, and store frequent item sets. Data streams, because of their unique features, have further posed many new challenges to frequent item set mining.

Recently, the data generation rates in some data sources have become faster than ever before. This rapid generation of continuous streams of information has challenged our storage, computation and communication capabilities in computing systems. The storage, querying and mining of such data sets are highly computationally challenging tasks. Mining data streams is concerned with extracting knowledge structures represented in models and patterns in non stopping streams of information.

Data stream mining is a stimulating field of study that has raised challenges and research issues to be addressed by the database and data mining communities [5]. Some of these new challenges are described here.

3.2.1 Handling the continuous flow of data

This is a data management issue. Traditional database management systems are not capable of dealing with such continuous high data rate. As a consequence, in many cases it is impractical to store all data in persistent media and in other cases it is too expensive to randomly access data multiple times. The challenge is to discover frequent item sets while the data can only be accessed once. Novel indexing, storage and querying techniques are required to handle this non stopping, fluctuating flow of information streams.

3.2.2 Minimizing energy consumption

Large amounts of data streams are generated in resource-constrained environments. Senor networks represent a typical example. These devices have short life batteries. The design of techniques that are energy efficient is a crucial issue given that sending all the generated stream to a central site is energy inefficient in addition to its lack of scalability problem.

3.2.3 Unbounded memory requirements

Another feature of data streams is that data are un-bounded but storage that can be used to discover or maintain the frequent item sets is limited. Machine learning techniques represent the main source of data mining algorithms. Most machine learning methods require data to be resident in memory while executing the analysis algorithm. Due to the huge amounts of the generated streams, it is absolutely important to design space efficient techniques that can have only one look or less over the incoming stream. Another consequence of un-bounded data is that whether an item set is frequent depends on time. The challenge is to use limited storage to discover dynamic frequent item sets from unbounded data.

3.2.4 Transferring data mining results

Knowledge structure representation is another essential research problem. After extracting models and patterns locally from data stream generators, it is essential to transfer these structures to the user. Kargupta et al. [6] have addressed this problem by using Fourier transformations to efficiently send mining results over limited bandwidth links.

3.2.5 Modeling changes of results over time

In some cases, the user is not interested in mining data stream results, but how these results are changed over time. For example of the number of clusters generated by a data stream changes over time, it might represent some changes in the dynamics of the arriving stream. Dynamics of data streams using changes in the knowledge structures generated would benefit many temporal-based analysis applications like video-based surveillance, emergency response, disaster recovery etc.

3.2.6 Real-time response

Because data stream applications are usually time-critical, there are requirements on response time. For some restricted scenarios, algorithms that are slower than the data arriving rate are useless.

3.2.7 Visualization of data mining results

Visualization of traditional data mining results on a desktop is still a research issue. Visualization in small screens of a PDA for example is a real challenge. Imagine a businessman and data are being streamed and analyzed on his PDA. Such results should be efficiently visualized in a way that enables him to take a quick decision.

4. Literature Review

In this paper, we survey a number of representative state-of-the-art algorithms on mining frequent item sets, frequent maximal item sets [15], or frequent closed item sets [16] over data streams. We organize the algorithms into three categories based on the window model that they adopt: the landmark window, the damped window or the sliding window. Each window model is then classified as time-based or count-based. According to the number of transactions that are updated each time, the algorithms are further categorized into update per transaction or update per batch. Then, we classify the mining algorithms into two categories: exact or approximate. We also classify the approximate algorithms according to the results they return: the false-positive approach or the false-negative approach. The false-positive approach [8-13,17] returns a set of item sets that includes all frequent item sets but also some infrequent item sets, while the false-negative approach [14] returns a set of item sets that does not include any infrequent item sets but misses some frequent item sets. We discuss the different issues raised from the different window models and the nature of the algorithms. We also explain the underlying principle of the ten algorithms and analyze their merits and limitations in data streams.

4.1 Landmark-window based mining

In landmark window model, frequent pattern mining is performed based on the values between a specific time-stamp called landmark and the present.

4.1.1 Lossy Counting BTS algorithm

Manku and Motwani [13] propose the *Lossy Counting* algorithm for computing an approximate set of FIs over the entire history of a stream. Therefore, an estimation mechanism is proposed in the Lossy Counting algorithm [18] which is a classic algorithm for mining frequent item sets in data streams and is based on the well-known Apriori property [19].: if any length k pattern is not frequent in the database, its length (k+1) super-patterns can never be frequent, to harmonize such a conflict. The Lossy Counting algorithm mines frequent item sets in data streams, when a maximum allowable error ε as well as a minimum support θ is given. The information about the previous mining result up to the latest block operation is maintained in the data structure called lattice.

Merits and limitations:

A distinguishing feature of the Lossy Counting algorithm is that it outputs a set of item sets that have the following guarantees:

– All FIs are outputted. There are no false-negatives.
– No item set whose actual frequency is less than $(\sigma - \varepsilon) N \approx 0$is outputted.
– The computed frequency of an item set is less than its actual frequency by at most εN.

However, using a relaxed minimum support threshold, ε, to control the quality of the approximation of the mining result leads to a dilemma. The smaller the value of ε, the more accurate is the approximation but the greater is the number of sub-FIs generated, which requires both more memory space and more CPU processing power. However, if ε approaches σ, more false-positive answers will be included in the result, since all sub-FIs whose computed frequency is at least $(\sigma - \varepsilon)/N \approx 0$ are outputted while the computed frequency of the sub-FIs can be less than their actual frequency by as much as σN. We note that this problem also exists in other mining algorithms [8-12, 17] that use a relaxed minimum support threshold to control the accuracy of the mining result.

4.1.2 StreamMining algorithm

StreamMining [20] is an in-core frequent item set mining algorithm based on BTS algorithm. The StreamMining uses a reduced set of 2-item frequent sets and uses the apriori property to reduce the number of i-itemsets, for i>2 and establishes a bound on the false positives. It is a single pass algorithm for frequent itemset mining in a streaming environment. This algorithm takes as input a parameter ε_. Given the desired support level θ, this one pass algorithm reports all item sets occurring with frequency level θ and does not include any item set occurring with frequency level less than (1-ε) θ. In the process, the memory requirements increase proportional to 1/ε.

Merits and limitations:

This is a one pass algorithm for streaming environment, which has deterministic bounds on the accuracy. Particularly, it is the first such algorithm which does not require any out-of-core memory structure and is very memory efficient in practice. One exception is that datasets with very large average length of an item set are not processed efficiently. Extra knowledge about maximal frequent item sets is required in such cases. The memory requirements of this algorithm are significantly lower than those of FP-tree. However, time-sensitive queries have not been considered.

4.1.3 DSM-FI Algorithm

DSM-FI algorithm [21] reads a basic window of transactions from the buffer in main memory and sorts them in lexicographical order. It constructs and maintains an in-memory prefix-tree based summary data structure called SFI-Forest (Summary Frequent Item set Forest). It prunes the infrequent information from the current SFI-Forest. It finds the frequent item sets from the SFI-Forest when it is needed. The frequency error guarantees provided by this algorithm is same as that of BTS algorithm. An efficient frequent item set search mechanism called ToDoFIS(Top Down Frequent Item set Search) is used to identify FIs from the SFI-Forest.
DSM-FI has three major features

i) single streaming data scan for counting frequency of item sets
ii) extended prefix-tree based compact pattern representation
iii) top-down frequent item set discovery scheme

Merits and limitations:

It is efficient on both dense and rare datasets. An SFI-forest is a more compact data structure than a prefix tree. We remark that the number of FCIs approaches that of FIs when the stream becomes large. Moreover, the compactness of the data structure is paid for by the price of a higher computational cost, since more tree traversals are needed to gather the frequency information of the item sets.

4.2 Damped landmark window based algorithms

In damped windows, the most recent windows are more important than the previous ones. In other words older transactions contribute less towards the item set frequencies.

4.2.1estDec algorithm

This is an approximate damped window based algorithm for mining data streams. The estDec method [8] examines each transaction in a data stream one by one without any candidate generation and keeps track of the occurrence count of an item set in the transactions generated so far by a prefix tree structure.

EstDec maintains a lattice for recording the potentially frequent item sets and their counts, and updates the lattice for each new transaction correspondingly. By recording additional information for each item set p, the time-point of the most recent transaction that contains p, the algorithm only needs to update the counts for the item sets which are the subsets of newly arrived transactions. It will reduce their counts using the constant factor d and then increase them by one.

Merits and limitations:

The use of a decay rate diminishes the effect of the old and obsolete information of a data stream on the mining result. However, estimating the frequency of an item set from the frequency of its subsets can produce a large error and the error may propagate all the way from the 2-subsets to the *n*-supersets, while the upper bound is too loose. Thus, it is difficult to formulate an error bound on the computed frequency of the resulting item sets and a large number of false-positive results will be returned, since the computed frequency of an item set may be much larger than its actual frequency. Moreover, the update for each incoming transaction (instead of a batch) may not be able to handle high-speed streams.

4.2.2 Instant

Instant [24] is a single-phase algorithm for finding MFIs over a data stream. It stores all of the currently maximal item sets for a data stream. Given a predefined minimum support count ð(ð>=1), there are ð-1 distinct arrays. Each array U[i], 1 <= I <= ð -1 stores those maximal infrequent item sets whose support counts are the same as i. In addition, the array F maintains all MFIs regardless of the support of each MFI.

Merits and demerits:

A new MFI can be displayed as soon as it becomes a frequent item set. However, the algorithm has some serious drawbacks. First of all, the algorithm employs a number of arrays to maintain all MFIs and maximal infrequent item sets so that the required size of memory space is very large. In addition, since there is no efficient superset/subset checking mechanism for a newly identified MFI against the item sets of each array, the number of comparisons among item sets is increased and the required amount of memory space is enlarged rapidly when the average length of transactions becomes longer. Although correctness and completeness of the resulting set of MFIs can be guaranteed for every incoming transaction, the algorithm should sacrifice both processing time and memory usage so that the algorithm is not suitable for an online data stream.

4.3 Sliding window based algorithms

The inspiration behind sliding window is that the user is more concerned with the analysis of most recent data streams. Thus the detailed analysis is done over the most recent data items and summarized versions of the old ones. This idea has been adopted in many techniques. In the sliding window model, knowledge discovery is performed over a fixed number of recently generated data items which are the target of data mining.

4.3.1 EstWin

EstWin algorithm [9] is used for finding recently frequent item sets adaptively over an online transactional data stream when a minimum support *Smin* ϵ (0, 1), a significant support *Ssig* ϵ (0,*Smin*) and the size of a sliding window *w* are given. Significant item sets, are maintained in main memory by a lexicographic tree structure [7, 8, 14]. The effect of the information in an old transaction that becomes out of the range of the sliding window is eliminated by decreasing the occurrence count of each item set that appeared in the transaction. The estWIN algorithm can be employed to find the recent change of embedded knowledge in a data stream. The interesting recent range of a data stream is defined by the size of a sliding window.

Merits and limitations:

The total number of item sets monitored in main memory is minimized by employing *delayed-insertion* and *pruning* techniques. It is also useful in identifying the recent changes in a data stream. The sliding window size can be changed to suit the user's requirement regarding the range of recent change. The method provides flexible trade-off between processing time and mining accuracy. The old disregarded information is not available if in future there is a need to analyze them. No summary structure about the old transactions is maintained.

4.3.2 TransSW and TimeSW algorithms

MFI-TransSW [25] is used to mine the set of frequent item sets online and incrementally within a sliding window by using an effective bit-sequence representation of items. Based on the MFI-TransSW framework, an extended single-pass algorithm, called MFI-TimeSW (Mining Frequent Item sets within a Time-sensitive Sliding Window), [25] is used to mine the set of frequent item sets over data streams within a time-sensitive sliding window.

In MFI-TransSW algorithm, for each item X in the current transaction-sensitive sliding window TransSW, a bit sequence with w bits, denoted as Bit(X), is constructed. If an item X is in the ith transaction of current TransSW, the ith bit of Bit(X) is set to be 1; otherwise, it is set to be 0. The process is called bit-sequence transform. MFI-TransSW algorithm consists of three phases: window initialization, window sliding and frequent item sets generation.

The major differences between MFI-TransSW and MFI-TimeSW are the unit of data processing, the bit-sequence transformation of a time unit, the number of sliding transactions, and the dynamic frequent threshold of item sets. In the MFITransSW algorithm, the constant value s. W is the frequent threshold of item sets, where s is the user specified minimum support threshold in the range of [0, 1] and w is the size of sliding window. However, in the MFI-TimeSW algorithm, the value of frequent threshold is s .|TimeSW| where the second term is a dynamic value.

Merits and limitations:

MFI-TransSW is an efficient single-pass algorithm for mining the set of frequent item sets over data streams with a transaction-sensitive sliding window. An effective bit-sequence representation of items is developed to enhance the performance of MFI-TransSW. MFI-TransSW and MFI-TimeSW not only attain highly accurate mining results, but also run significant faster and consume less memory than do existing algorithms, such as SWF, and Moment for mining frequent item sets from data streams within a sliding window. They maintain all frequent item sets which is quite a large number. Instead they can go for storing only the closed frequent item sets which will reduce the space requirement. Also these algorithms employ the candidate generation concept of apriori algorithm which can be improved upon by employing methods which do not go for candidate generation.

4.3.3 Moment

Chi proposed a new algorithm, Moment, [26] to mine and store all closed frequent- item sets using a sliding window which holds the most recent samples in a data-stream. The recent samples are stored using memory-structure called the Closed-Enumeration-Trees (CET). The CET is constructed using a depth first process which maintains every item set in lexicographical order and if the item set is found frequent then it is added to the tree as a node. Non-frequent item sets are also stored as they may become frequent in the future.

This approach distinguishes between 4 types of nodes, namely, Infrequent gateway node (infrequent nodes whose parents and/ siblings are frequent), Unpromising gateway node (infrequent node who has a superset having same support as itself), Intermediate node (frequent node who has a superset having same support as itself), and Closed node (node which has no children with greater or equal support as itself).

During the sliding window timeframe, whenever a transaction arrives, Moment classifies the nodes present in the transaction into one of the 4 categories mentioned previously. Whenever the user requests for mined results the CET (which observes only closed nodes and boundary nodes) is traversed to mine the top ‘n’ closed-frequent item sets assumed that all the interesting changes occur at the boundary of closed-frequent item sets and rest of the item sets which turns out to be true most of the time.

Merits and Limitations:

Unlike prefix trees, Moment’s CET maintains only closed item sets and nodes that mark the boundary between closed item sets and the rest of the item sets. This reduces the number of nodes to a large extent when compared to prefix trees. But still the CET data structure (cost-enumeration-trees) operations such as tree creation and maintenance is very time-consuming (since it scans the constructed tree in a depth-first manner), the storage and computational overheads consumed by Moment are very high. Thus, it can be surmised that, although algorithms based on exact methods guarantee better accuracy they suffer greatly in terms of computation time and memory requirements. Therefore, several approximation based methods were initiated to provide satisfactory performance by keeping computation costs under control.

5. Conclusion

Many organizations today have more than very large databases; they have databases that grow without limit at a rate of several million records per day. Mining these continuous data streams brings unique opportunities, but also new challenges.

Recently, mining data streams with concept drifts for actionable insights has become an important and challenging task for a wide range of applications including credit card fraud protection, target marketing, network intrusion detection, etc. Conventional knowledge discovery tools are facing two challenges, the overwhelming volume of the streaming data, and the concept drifts.

Pervasive computing applications, such as Internet-based control systems and large-scale mobile asset management, requires the monitoring and processing of data streams from highly distributed, mobile data sources. Middleware to support data stream processing must scale to potentially millions of streams. In many pervasive applications, streams represent rapid updates to stored data, which precludes the use of traditional scaling strategies such as mirroring or caching.

In this article we discussed the issues that need to be considered when designing a stream data association rule mining technique. We also discussed the existing approaches in mining data streams. From the above discussions, we can see that most of the current mining approaches adopt an incremental and one pass mining algorithm which are suitable to mine data streams, but few of them address the concept drifting problem. As more of these problems are solved and more efficient and user-friendly mining techniques are developed for the end users, it is quite likely that in the near future data stream mining will play a key role in the business world.

References

[1] Sudipto Guha, Nick Koudas, Kyuseok Shim; “Data Streams and Histograms”; ACM Symposium on Theory of Computing; 2001. p. 471-475

[2] Hillol Kargupta, Ruchita Bhargava, Kun Liu, Michael Powers, Patrick Blair, Samuel Bushra, James Dull, Kakali Sarkar, Martin Klein, Mitesh Vasa, David Handy; “VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring”; SIAM Int'l Conf. on Data Mining; 2004.

[3] Mihail Halatchev and Le Gruenwald; “Estimating Missing Values in Related Sensor Data Streams”; Int'l Conf. on Management of Data; January 2005. p. 83 – 94

[4] Gurmeet Singh Manku, Rajeev Motwani; “Approximate Frequency Counts over Data Streams”; Int'l Conf. on Very Large Databases; 2002 p.346-357

[5] Nan Jiang and Le Gruenwald, the University of Oklahoma, School of Computer Science, Norman, OK 73019, USA: “Research Issues in Data Stream Association Rule Mining”; SIGMOD Record, Vol. 35, No. 1, Mar. 2006 p. 14 - 19

[6] B. Park and H. Kargupta; “Distributed Data Mining: Algorithms, Systems, and Applications”. Published in the Data Mining Handbook. Editor: Nong Ye. 2002

[7] Tanner S., Alshayeb M., Criswell E., Iyer M., McDowell A., McEniry M., Regner K., “EVE: On-Board Process Planning and Execution”, Earth Science Technology Conference, Pasadena, CA, Jun. 11 - 14, (2002).

[8] Chang JH, Lee WS (2003) Finding recent frequent item sets adaptively over online data streams. In: Getoor L, Senator T, Domingos P, Faloutsos C (eds) Proceedings of the Ninth ACM SIGKDD international conference on knowledge discovery and data mining, Washington, DC, August 2003, pp 487–492

[9] Chang JH, Lee WS (2003) estWin: adaptively monitoring the recent change of frequent item sets over online data streams. In: Proceedings of the 2003 ACM CIKM international conference on information and knowledge management, New Orleans, Louisiana, USA, November 2003, pp 536–539

[10] Chang JH, Lee WS (2004) A sliding window method for finding recently frequent item sets over online data streams. J INF Sci Eng 20(4):753–762

[11] Giannella C, Han J, Pei J, Yan X, Yu P (2004) 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

[12] Li H, Lee S, ShanM (2004) An efficient algorithm for mining frequent item sets 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

[13] Manku GS, Motwani R (2002). Approximate frequency counts over data streams. In: Proceedings of the 28th international conference on very large data bases, Hong Kong, August 2002, pp 346–357

[14] Yu J, Chong Z, Lu H, Zhou A (2004) False positive or false negative: mining frequent item sets from high speed transactional data streams. In: Nascimento et al. (Eds) Proceedings of the thirtieth international conference on very large data bases, Toronto, Canada, September 3–August 31, 2004, pp 204–215

[15] Gouda K, Zaki M (2001). Efficiently mining maximal frequent item sets. In: Cercone N, Lin TY, Wu X (eds) Proceedings of the 2001 IEEE international conference on data mining, San Jose, 29 November – 2 December 2001, pp 163–170

[16] Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Discovering frequent closed item sets for association rules. In: Beeri C, Buneman P (eds) Proceedings of the 7th international conference on database theory, Jerusalem, Israel, January 1999, pp 398–416

[17] Lee D, LeeW (2005) Finding maximal frequent item sets over online data streams adaptively. In: Proceedings of the 5th IEEE international conference on data mining, Houston, Texas, USA, November 2005, pp 266–273

[18] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Bocca J, JarkeM, Zaniolo C (Eds) Proceedings of 20th international conference on very large data bases, Santiago de Chile, Chile, September 1994, pp 487–499

[19] Agrawal R, Srikant R (1995) Mining sequential patterns. In: Yu P, Chen A (Eds) Proceedings of the eleventh international conference on data engineering, Taipei, Taiwan, March 1995, pp 3–14

[20] An Algorithm for in-core frequent itemset mining on streaming data In: Proceedings of the 5th IEEE International Conference on data mining, Houston, TX, USA, pp 210-217

[21] H.-F., Lee, S.-Y., Shan, M.-K. (2004). An efficient algorithm for mining frequent item sets over the entire history of data streams. In: Proceedings of the IWKDDS

[22] Chernoff H (1952) A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann Math Stat 23(4):493–507

[23] EstMax: Tracing Maximal Frequent Item Sets Instantly over Online Transactional Data Streams Ho Jin Woo and Won Suk Lee, Member, IEEE In: IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No. 10, October 2009, pp 1418-1431

[24] G. Mao, X. Wu, X. Zhu, and G. Chen, “Mining Maximal Frequent Item sets from Data Streams,” J. Information Science, vol. 33, no. 3, pp. 251-262, 2007.

[25] Mining frequent item sets over data streams using efficient window sliding techniques. Expert Systems with Applications, 36(2P1), 1466–1477.

[26] Chi Y, Wang H, Yu P, Muntz R (2004) Moment: maintaining closed frequent item sets over a stream sliding window. In: Proc. of the 4th IEEE international conference on data mining, Brighton, UK, Nov. 2004, pp 59–66

[27] Nan Jiang, Le Gruenwald; CFI-Stream: Mining Closed Frequent Item sets in Data Streams; KDD; August 2006.

[28] Ranganath B.N., M. Narasimha Murty; “Stream-Close: Fast mining of closed Frequent Item sets in high speed data streams”; 2008 IEEE International Conference on Data Mining Workshops p. 516-525