ISSN 22773061

TIFIM: Tree based Incremental Frequent Itemset Mining over Streaming Data

V.sidda Reddy, Dr T.V.Rao, Dr A.Govardhan
Professor, Sagar Institute of Technology,Hyderabad, A.P, India
siddareddy.v@gmail.com
Professor, School of Computing, K.L. University, A.P, India
tv_venkat@yahoo.com
Professor, School of Information Technology,JNTUH, A.P, India
govardhan_cse@yahoo.co.in

ABSTRACT

Data Stream Mining algorithms performs under constraints called space used and time taken, which is due to the streaming property. The relaxation in these constraints is inversely proportional to the streaming speed of the data. Since the caching and mining the streaming-data is sensitive, here in this paper a scalable, memory efficient caching and frequent itemset mining model is devised. The proposed model is an incremental approach that builds single level multi node trees called bushes from each window of the streaming data; henceforth we refer this proposed algorithm as a Tree (bush) based Incremental Frequent Itemset Mining (TIFIM) over data streams.

Keywords

Data Mining, Data Streams, Frequent itemset, Frequent Itemset Mining, Data Stream Mining

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 10, No 5
editor@cirworld.com
www.cirworld.com, member.cirworld.com

ISSN 22773061

1. INTRODUCTION

Frequent Itemsets, are the set of items appear together in given transactions over a given threshold number of times. The process of mining frequent itemsets is included in divergent mining methods such as formation of Association Rules for Market -Basket analysis, classification and clustering of documents, pages, and text in text mining, web mining, which enables users to uncover hidden relationships and patterns in large datasets. In a wide range of emerging applications, data is in the form of an enormous, continuous stream where the speed at which the data is produced outstrips the rate at which it can be mined [1]. This is in direct contrast to traditional static databases; thus data stream mining therefore is substantially deviant from conventional data mining in numerous aspects. Firstly, the absolute volume of data embedded in a data stream over its lifespan can be overwhelmingly huge [2]. Secondly, due to resource bottlenecks, generating timely responses by keeping response time to queries on such data streams is necessary [3]. Because of the issues stated above, data stream mining has become the subject of intense research and the problem of obtaining timely and accurate association rules is a contemporary research topic. There is a critical need to switch from traditional data mining schemes to those methods that are able to operate on an open-ended, high speed stream of data [4]. Due to the inherent nature of a data stream, any mining scheme faces the challenges [2] such as, since the data streams, the traditional approach of scanning the database multiple times for model creation is no longer feasible. The performance of the approaches such as association rules, clustering and classification are focus on the optimality of the frequent itemset mining. They're predicated on the calculations that need the alteration of data from one depiction to other, and so excessively use resources and incur hefty CPU overhead. This paper projects a tree based incremental frequent itemset mining model that is resource efficient and scalable as it performs with less memory requirements and fewer computational cost. It characterizes the tradeoffs among data depiction, computation, I/O and heuristics. The projected algorithm uses one level multi node tree based item storage for the candidate and frequent itemsets.

2. RELATED WORK

Syed Khairuzzaman Tanbeer et.al [5] devised a dynamic tree reformation system called CPS-tree to handle the stream data, which is based on prefix tree structure. This dynamic tree reformation approach reforms the tree upon receiving a new item, which leads to huge memory usage. Since the reformation process is continuous, the computational time and cost also high.

The weighted sliding window (WSW) algorithm that devised by Pauray S.M. Tsai [6] calculates the weight of each transaction in each window. The WSW model accompanied with candidature maintenance, which not a feasible process as in terms of memory and time usage. For candidate generation, an apriori algorithm is used. MFI-Trans SW (Mining Frequent Item sets with in a Transaction Sensitive Sliding window) is bit-sequence based algorithm devised by Hue-Fu Li et.al [7], which worked on three phases. And observations indicating that “memory usage is proportional to window size”, “the process time is also proportional to window size in phase1 and 2”. Weighted Support Frequent Item sets mining (WSFI-mine) with normalized weight over data stream is devised by Yo Unghee Kim et.al [8], which discovers frequent itemsets in a single pass. High Utility Pattern Mining in Stream data (HUPMS) is explored by Chowdhury Farhan Ahmed et. al [9], which is a sliding window approach and limited to interactive mining. Mining frequent patterns from streaming data from multiple streams is discussed by Jing Guo,Peng Zhang et. al [10]. The resultant patterns of frequent pattern mining over multiple streaming data are collaborative and comparative patterns. Privacy preserving in frequent itemset mining over data streams is discussed and explored novel model by anushree Gowtham Ringe et. al [11]. The model devised by Fan Guidan et al [12] is based on matrix in sliding window over data streams, which is conceptually relative to our proposed TIFIM. The algorithm in [12] used two 0-1 matrices to store transaction and 2-itemsets, then we could get frequent item sets through some relative operation of the two matrices. Hence the experiment results empirically analyzing the performance of the proposed TIFIM with model devised in [12].

3. TREE (BUSH) BASED INCREMENTAL FREQUENT ITEMSET MINING (TIFIM)

3.1 Caching of the Transactions

Let A_{set} be the attribute set contains set of attributes, and a transaction t is formed by set of attributes such that $t \subset A_{set}$.

Let Stream S said to be sliced as windows, such that each window w represents a transaction t

Upon receiving first transaction:

Let Project a set of trees with one hierarchy, such that each tree b contains a pair of attributes $\{a_i, a_j \mid a_i \in t, a_j \in t, a_i \in A_{set} \text{ and } a_j \in A_{set}\}$ as root and current transaction t_c, which is first transaction as child. Since these trees are one level multi child models, hence there after we refer each tree as bush, and building a tree as bush formation.

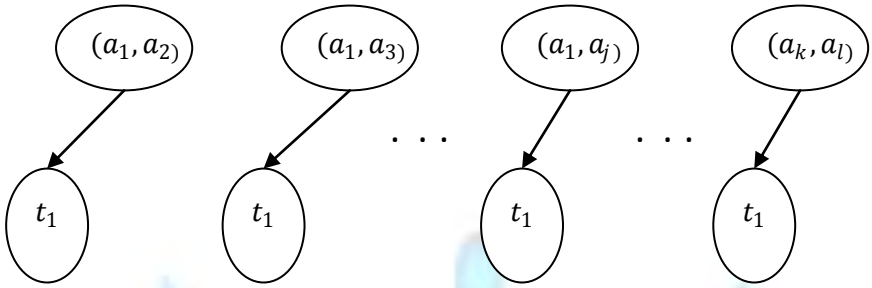

Fig 1: Adding transactions to bush denoted by frequent itemset

Upon receiving a window w_c, extract pairs of attributes from the transaction t_c represented by w_c, and then verifies that a bush is already formed by that pair or not, if not then forms a bush with current pair as root and current transaction t_c as child. A sample bush formation can be addressed by the Fig 2.

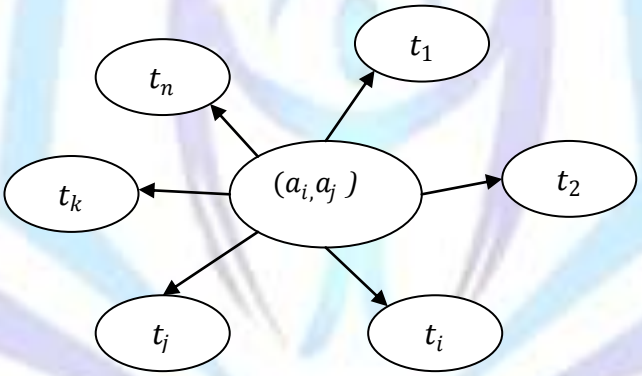

Fig 2: A bush representation of the attribute pair and set of transactions with that pair

3.2 Finding Frequent Itemsets

The primary representation of the transactions streamed through S is as described above. An asynchronous parallel process runs to find frequent itemsets in incremental manner.

A bush represents an itemset with two attribute pair and transactions contain that pair. Assume the case of total number of transactions unpredictable, the coverage can be considered to measure the frequency of the itemsets. The coverage of two attribute itemset can be the count of number of Childs in a bush represented by that pair of attributes.

An asynchronous parallel process called frequent itemset finder (FIF) performs as follow:

Initially picks the bushes with coverage more than given coverage threshold COV .

Prepare new bushes from each two bushes by union the roots and intersects the Childs, and retains it if new bush coverage is greater or equal to COV else discards.

This continues until no new bush formed.

3.3 The pruning process

A bush b_i said to be sub-bush to bush b_j if $r_{b_i} \subseteq r_{b_j}$ and $\text{cov}_{(b_i)} \leq \text{cov}_{(b_j)}$. Since sub-bush b_i represented by b_j, then bush b_i can be pruned from the bush-set B.

3.4 Find frequent items

At an event of time, frequent itemsets can be found as follows

The roots of the bushes with coverage more than given cov can be claimed as frequent itemsets.

A bush 'b_i' coverage can be find as follows

If a bush b_j found to be such that $b_i \subseteq b_j$ and coverage value of b_j is higher than any other bush b_k such that $b_i \subseteq b_k$, then the coverage of b_i said to be $\text{cov}_{(b_j)} + \text{cov}_{(b_i)}$.

3.5 An algorithmic representation of the caching processes

Input: At an event of time a window w_i with transaction t_i received

For each transaction t_i :

Let set of attributes $\{(a_1, a_2, a_3, ...a_i) \forall (a_1, a_2, a_3, ...a_i) \in t_i \wedge (a_1, a_2, a_3, ...a_i) \subseteq A_{set}\}$

For each pair of attributes $\{(a_m, a_n) \forall (a_m, a_n) \in t_i\}$, if found a bush $\{b_i \exists (a_m, a_n)\ as\ root\}$ then add transaction t_i as node to bush b_i, else prepare a bush such that $\{b_i \exists (a_m, a_n)\ as\ root \wedge t_i\ as\ node\}$

3.6 An algorithmic approach of FIF

The bush set B prepared by caching process is said to be input to FIF

For each bush $\{b_i \forall b_i \in B\}$ perform the following:

For each bush $b_{i+c} \exists (c := 1, 2, 3.....n) \wedge b_{i+c} \in B)$

Forms a bush $\{b_{(i \cup i+c)} \exists b_{(i \cup i+c)} \notin B\}$ by Union the roots of the 'b_i' and 'b_{i+c}' ($r_{(b_i)} \bigcup r_{(b_{i+c})}$) and intersects nodes of b_i and b_{i+c} ($ts_{(b_i)} \bigcap ts_{(b_{i+c})}$).

4. PERFORMANCE ANALYSIS

4.1 Datasets

Acquiring a real life dataset is quite difficult due to the fact that many organizations refuse to part with their data because of the sensitivity and confidentiality of the data. Therefore, artificial synthetic data generators such as IBM are very commonly used by researchers to evaluate and benchmark their algorithms' Performances. The experimentation is carried out with the help of synthetic datasets that are generated through the use of a dataset generator that is publically available [13].

4.2 Performance Metrics

The TIFIM is evaluated against certain commonly used performance metrics such as Accuracy (in terms of Recall and Precision), Computational performance (in terms of time taken to process the dataset under divergent support values), and Memory consumption in terms of number of bushes maintained in memory. Recall and precision can be defined.

$$\text{Recall} = \frac{|afi \cap tfi|}{|afi|}$$

$$\text{precision} = \frac{|afi \cap tfi|}{|tfi|}$$

Here in the equations, afi indicates the actual frequent itemsets, tfi indicates traced frequent itemsets.

4.3 Experimental results

We compare our algorithm with frequent itemset mining model for data streams devised in [12], which is a matrix based frequent itemsets mining algorithm for data streams. The implementation of TIFIM and model devised in [12] done by using java 7 and set of flat files as streaming data sources. The streaming environment is emulated using java RMI and parallel process involved in proposed TIFIM is achieved by using java multi threading concept. The three parameters of each synthetic dataset are the total number of transactions, the average length, and divergence count of items, respectively. The synthetic dataset that used here is labeled as TIFIM9999.9994 and it contains 9999 transactions and 999 different items, and each transaction consists of 4 items in average. Each transaction of a dataset is scanned only once in our experiments to simulate the environment of data streams. In regard to measure the computational cost and scalability, the algorithms run under divergent coverage values in the range of 10 to 2.

Fig 3, 4 shows the comparison of the Memory usage, execution time under divergent coverage values range given between 2 to10 respectively. The figure 5 and 6 explores the scalability of TIFIM over Matrix based FIM under divergent streaming data sizes respectively, In Fig 3 and 4, the horizontal axis is the coverage given and the vertical axis is the memory in unit of mega bytes and time in unit of seconds respectively. In Fig 5 and 6, the horizontal axis is the streaming data size given in unit of transactions and the vertical axis is the execution time in unit of seconds and percentage of elapsed time in unit od seconds respectively. The experimental data is the synthetic dataset generated by IBM Quest data generator. As the coverage value decreases the average increment in memory usage for matrix based FIM and TIFIM are 2.29 and 0.9 respectively (see Fig 3) and average execution time increment for matrix based FIM and TIFIM are 83.2 and 49.9 respectively (see Fig 4). The results obtained here clearly indicating that the performance of TIFIM is miles ahead than the matrix based FIM. The performance of TIFIM is scalable as matrix based FIM is taking average of 17.16% elapsed time under uniform increment of streaming data size with 1500 transactions (see Fig 6).

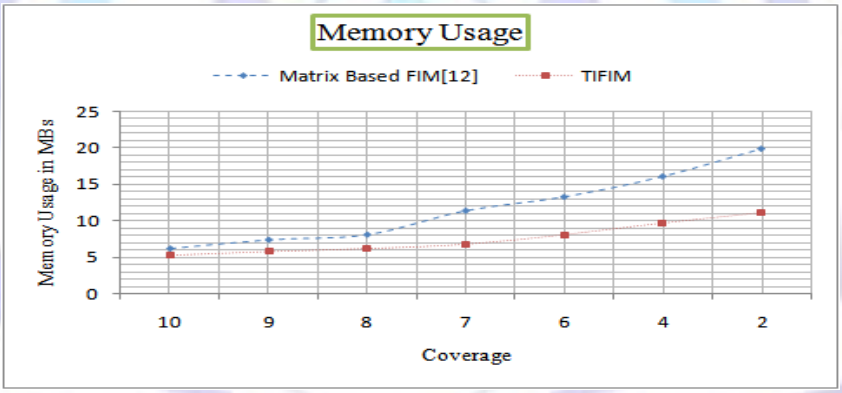

Fig 3: Memory usage comparison report

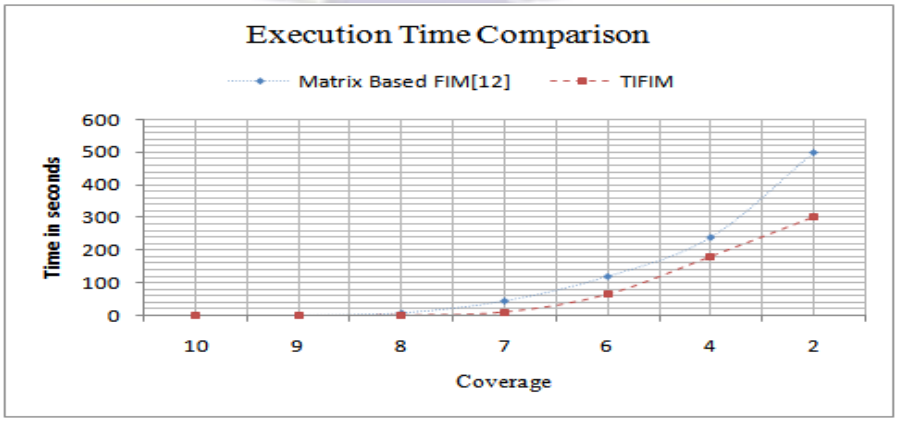

Fig 4: Performance analysis in terms of execution time

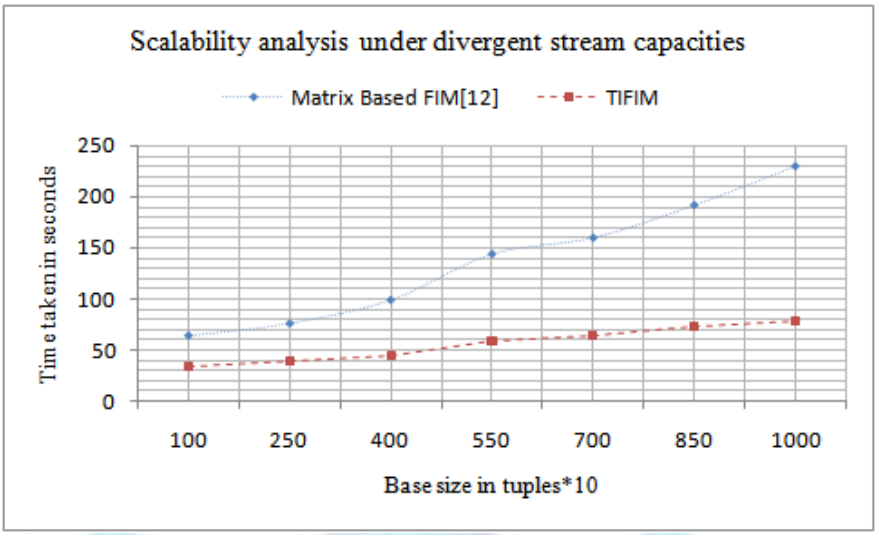

Fig 5: Scalability check under divergent streaming data size

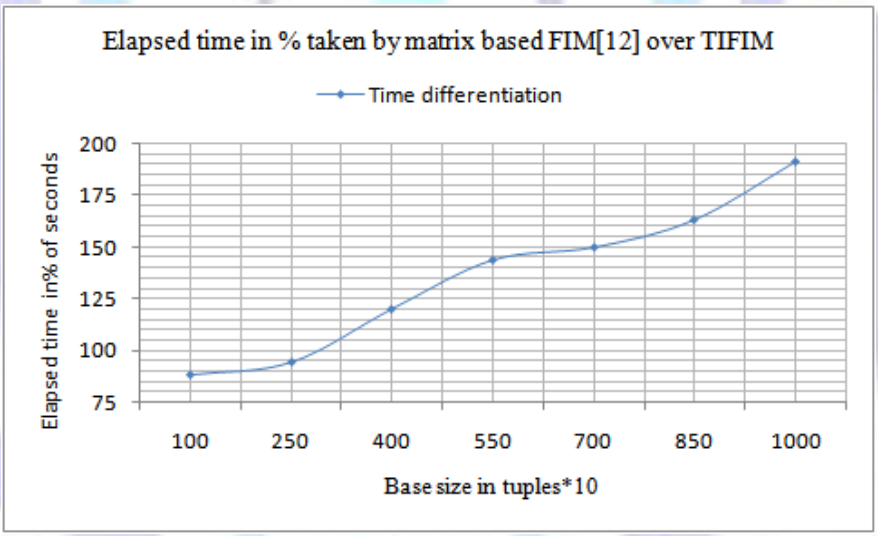

Fig 6: Elapsed time in % taken by Matrix based FIM [12] over TIFIM

5. CONCLUSION

We explored a novel approach for mining the frequent itemsets from a data stream. We have implemented an efficient tree based incremental frequent itemset mining model, which caches itemsets as single level tree called bush. A parallel process that determines frequent itemsets from the cached bush structures and prunes these bush structures for memory efficiency. We developed an incremental frequent itemset mining algorithm with efficient memory usage, execution time. The experiment results confirm that the TIFIM is scalable under divergent streaming data size and coverage values. In future this model can be extended to perform utility based frequent itemset mining over data streams.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments. We also thank the authors of all references for helping us setup the paper.

REFERENCES

[1] Charikar, M ., Chen, K., Colton, M . (2004). Finding frequent items in data streams. Theoretical Computer Science, 1-11.

[2] Gaber, M ., Zaslavsky, A., Krishnaswamy, S.(2005). Mining data streams: A review. ACM SIGM OD Record, 34(2), 18-26.

[3] Jiang, N., Gruenwald, L. (2006). CFI-stream: Mining closed frequent itemsets in data streams.Paper presented at the KDD, Philadelphia, USA.

[4] Manku, G., Motvani, R. (2002). Approximate frequency counts over data streams.Paper presented at the 28th International Conference on Very Large Data Bases, Hong Kong, China.

[5] Syed Khairuzzaman Tabeer, Chowdary Farha ahmed, Byeong-Soo Jeong, Young Koo Lee"Efficient frequent pattern mining over data streams" 2008.

[6] Pauray S.M.Tsai, "Mining frequent item sets in data streams using the weighted sliding window model", Elsevier publication 2009.

[7]] Hue-Fu Li, Suh-Li "Mining frequent item sets over data streams using efficient window sliding technique", Elsevier publication. 2009.

[8] www.borgelt.net/slides/fpm.pdf.

[9] Chowdary Farha ahmed, Byeong-Soo Jeong, "Efficient mining of high utility patterns over data streams with a sliding window model". Springerlink.com, 2011.

[10] Jing Guo, Peng Zhang, Jianlong Tan and li Guo "Mining frequent patterns across multiple data streams", 2011.

[11] Anushree Gowtham Ringe, Deeksha Sood and Turga Toshniwa"Compression and privacy preservation of data streams using moments",Information journal of machine learning and computing. 2011.

[12] Fan Guidan; Yin Shaohong, "A Frequent Itemsets Mining Algorithm Based on Matrix in Sliding Window over Data Streams," Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on , vol., no., pp.66,69, 16-18 Jan. 2013; doi: 10.1109/ISDEA.2012.23.

[13] R. Agrawal, A. Arning, T. Bollinger, M. Mehta, J. Shafer, and R. Srikant. The Quest data mining system. In Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, August 1996.