Mining Maximum Frequent Item Sets Over Data Streams Using Transaction Sliding Window Techniques

Neeraj And Anuradha

Student/ Software engineering/M.Tech ITM University Gurgaon, 122017, India
neeraj.yadav28 @gmail.com
Faculty/ Software engineering ITM University Gurgaon, 122017, India
anuradha@itmindia.edu

ABSTRACT

*As we know that the online mining of streaming data is one of the most important issues in data mining. In this paper, we proposed an efficient one- .frequent item sets over a transaction-sensitive sliding window), to mine the set of all frequent item sets 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 the performance significant faster and consume less memory than existing algorithms for mining frequent item sets over recent data streams. In this paper our theoretical analysis and experimental studies show that the proposed algorithm is efficient and scalable and perform better for mining the set of all maximum frequent item sets over the entire history of the data streams***.**

Keywords

Recently frequent item sets, Maximum frequent item set, Transaction sliding window, Data stream.

1. INTRODUCTION

We know that the frequent item sets mining play an essential role in many data mining tasks. With years of research into this research, several data stream mining problems have been discussed, such as frequent item set mining **[5, 6, 7, 8, 9],** closed frequent structure mining **[19,24,29].** The frequent item set mining over data streams is to find an approximate set of frequent item sets in transaction with respect to a given support and threshold. It should support the flexible trade-off between processing time and mining accuracy. It should be time efficient even when the user-specified minimum support threshold is small. The objective was to propose an effective algorithm which generates frequent patterns in a very less time. Our approach has been developed based on improvement and analysis of MFI algorithm. Frequent patterns can be very meaningful in data streams. In network monitoring, frequent patterns can correspond to excessive traffic which could be an indicator for network attack. In sales transactions, frequent patterns relate to the top selling products in a market, and possibly their relationships. Due to numerous applications of this sort mining frequent patterns from data stream has been a hot research area. If we consider that the data stream consists of transactions, each being a set of items, then the problem definition of mining frequent patterns can be written as - Given a set of transactions, find all patterns with frequency above a threshold s. Traditional mining algorithms assume a finite dataset over which the algorithms are allowed to scan multiple times. Therefore it is necessary to modify these algorithms in order to apply these algorithms to transactional data streams. An important property that distinguishes data stream mining from traditional data mining is the unbounded nature of stream data, which precludes multiple-scan algorithms. Traditional frequent item sets mining algorithms are thus not applicable to a data stream **[11].**

Stream mining algorithms are being developed to discover useful knowledge from data as it streams in -- a process that isn't as straightforward as it might seem. Scientists have to deal with the fact that the data rate of the stream isn't constant, leading to a condition called business, and the patterns of the data stream are continuously evolving. Online mining of frequent item sets over a stream sliding window is one of the most important problems in stream data mining with broad applications. It is also a difficult issue since the streaming data possess some challenging characteristics, such as unknown or unbound size, possibly a very fast arrival rate, inability to backtrack over previously arrived transactions, and a lack of system control overtheorderinwhichthedataarrive.Inthispaper,weproposeaneffectivebit-sequencebased,one pass algorithm, called MFI-TransSW (Mining Frequent Item sets within a Transaction-sensitive Sliding Window), to mine the set of frequent item sets from data streams within a transaction-sensitive sliding window which consists of a fixed number of transactions**[2,28,30]**

Figure first shows the categories of data streams are divided into three parts: Landmark windows, sliding windows, damped windows. Further the sliding windows also divided into two more parts: transaction sensitive windows and time sensitive sliding windows **[2, 21, 22, 23].**

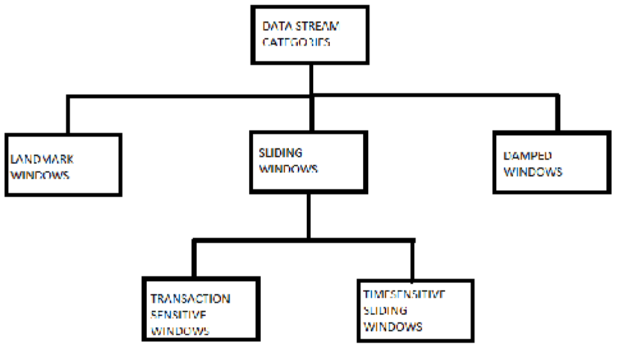


Fig 1.Categories of data stream.

Pei, Yin, & Mao, 2004 is used to maintain the frequent item sets and their supports stored in titled-time windows. In a sliding window model, knowledge discovery is performed over a fixed number of recently generated data elements which is the target of data mining. 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 transaction-sensitive sliding window is an expired transaction while the basic unit of window sliding of time-sensitive sliding window is a time unit, such as a minute or 1h**. [3, 20, 25, 26, 27]**

According to the stream processing model, the research of mining frequent item sets in data streams can be divided into three categories, snapshot windows, sliding windows and Landmark windows. In snapshot window model the frequent items are detected in a fixed range of time so that model is not used frequently. In landmark window model knowledge discovery is performed based on the values between the specific timestamp called landmark and the present. In the sliding window model knowledge discovery is performed over a fixed number of recently generated data elements which is the target of data mining. Sliding window model is a widely used model to perform frequent item set mining since it considers only recent transactions and forgets obsolete ones. Due to this reason, a large number of sliding window based algorithms have been devised **[11][12] [13][14][15][17].** However, only a few of these studies adaptively maintain and update the set of frequent item sets **[13][14][15]** and others only store sliding window transactions in an efficient way using a suitable synopsis structure and perform the mining task when the user requests.

The purpose of this paper is to mining the frequent item set over online data streams with the help of the transaction sensitive sliding window. The experiment shows that the purposed algorithm MFI-TransSW not only attained high accurate mining data but also increase the speed as well as the memory uses is less than the existing mining algorithm. The problem of mining frequent item set is defined and the algorithm is purposed also in this paper.

2. PROBLEM DEFINITION

One of the most important data mining problems is mining maximal frequent item sets from a large database [2, 6, 7, 14, 25].This problem of mining maximal frequent item sets was first of all given by Bayardo **[4].**

Let W= {i1, i2,..., im} be a set of items. A transaction T= (tid, x1x2...xn), xi 2W, for is a set of items, while n is called the size of the transaction, and tid is the unique identifier of the transaction. An item set is a non empty set of items. An item set with size k is called a k-item set. A transaction data stream TDS=T1,T2,...,TN is a continuous sequence of transactions, where N is the tid of latest incoming transaction TN.A transaction-sensitive sliding window (TransSW) in the transaction data stream is a window that slides forward for every transaction. The window at each slide has a fixed number, w, of transactions, and w is called the size of the window. Hence, the current transaction- sensitive sliding windows TransSWNw+1=[TN□w+1,TN□w+2,...,TN],where□w+1 is the window identifier of current TransSW.The support of an item set X over TransSW, denoted as sup(X) TransSW, is the number of transactions in TransSW containing X as a subset. An item set X is called a frequent item set (FI) if sup(X) TranSWPs □w, where s is a user defined minimum support threshold (MS) in the range of [0, 1, 2]. The value s □w is called the frequent threshold of TranSW (FTTranSW) given a transaction-sensitive sliding window TransSW, and a MST s, the problem of online mining of frequent item sets in recent transaction data streams is to mine the set of all frequent item sets by one scan of the TransSW.

EXAMPLE: Suppose we have six transaction in a transaction data stream be <T1,(1 3 4 7)>,<T2,(2 3 5 6 8)>,<T3,(1 2 3 5 7) >,<T4,(2 5 8)>,<T5,(1 3 7 8)>,<T6,(2 3 7)>, where T1,T2,T3,T4,T5,T6 are the transactions and the 1,2,3.,4,5,6,7,8 are the item sets in the transactions .Let the size of the sliding window be 3 and the minimum support is 2.

TID	Item sets
T1	1,3,4,7
T2	2,3,5,6,8
T3	1,2,3,5,7
T4	2,5,8
T5	1,3,7,8
T6	2,3,7

Item sets	Support
1	3
2	4
3	5
5	3
7	4
8	3

Item sets	Support
{1,3}	3
{1,7}	3
{2,3}	3
{2,5}	3
{3,7}	4

Item sets	Support
{1,3,7}	3

Fig 2: An example of transaction data stream and find out maximum frequent item set.

Online mining of frequent item sets within a transaction-sensitive sliding window:

In this part we describe our algorithm that is a single pass mining algorithm called MFI-Trans SW and its bit representation of items with the help of the example .In this we compared our algorithm with the existing sliding window based mining techniques and improve the speed as well as memory by dynamically maintenance in the current sliding window by the effective bit sequence representation.

1) *Bit-Sequence Representation of Items*

Transactions	Items	Bit-Sequence
T1<1,3,4,5>	1	101
T2<2,3,5,6,7>	2	011
T3<1,2,3,5,7>	3	111
	4	100
	5	011
	6	010
	7	101
	8	010

101 &111=101
This is the bit
representation
for (1,3)

Fig: 3 Bit-Sequence Representations of Items.

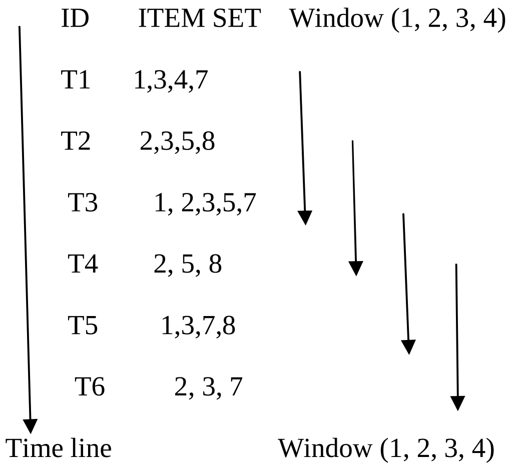
The above figure 3 shows that the bit sequence representation. How we can represent our item set in bit sequence representation. The computation of frequent item sets can be done efficiently by representing all the items in the transactions of the data stream by bit sequence representation **[12]**. If there are three transactions in the given window size then the length of the bit sequence representation will be three. If there are four transactions then the length will be four. The first bits represent the first transaction, second bit represent the second transaction and so on. If the sliding window is containing three transaction and they contain different item set as shown in the above fig. The concept of bit sequence representation are illustrated in the above figure 2 where the bit sequence representation for items (1,3) is obtained by performing bit AND operator on bit the sequence representation of item 1 and the sequence representation of item 3.The same procedure is repeated for deriving the bit sequence representation of other item set of length 2 .Once the 2 item set are available then easily we can illustrate the three bit representation of item three and so on. The bit sequence representation enables the computation of frequent item sets easily because counting the number of 1's in the bit sequence will give us the exact count of number of transactions that contain the item set. This count can be compared with the support threshold to find if the item set is frequent or not.

3. MFI- TransSW ALGORITHIM

3.1 Window Initialization Phase

This phase is activated when the number of transaction data stream is less than or equal to a user defined sliding window size. In this phase every transaction is converted into bit sequence representation.

3.2 Window Sliding Phase



ID	ITEM SET
T1	1,3,4,7
T2	2,3,5,8
T3	1, 2,3,5,7
T4	2, 5, 8
T5	1,3,7,8
T6	2, 3, 7

Fig: 4 running example of window size 3.

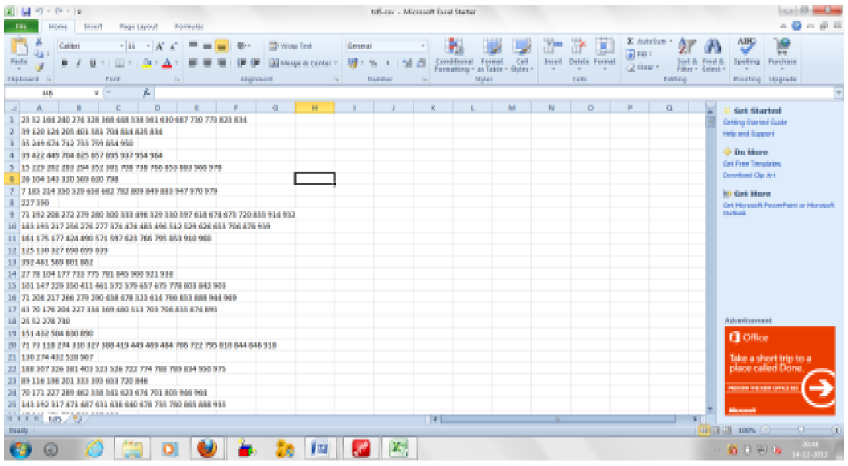
```
Input: TDS (a transaction data stream),ms( user defined minimum support in the range of
[0,1,2],and ws (the user defined sliding window size)
Output: FI-Output (set of frequent item set)
Begin
/*the following algorithm is for sliding window how the transactions are going according
to GWS ( given window  size) */
  Item_count=0,t_count=0,GWS(given window
  size),TDS Load
  Tdsw=NULL;      /*consist of number of
  transactions */
  For (i=0; i>=EOF TDS;i++)     /*for each
  transaction */
  T_count++;
  If (t_count>=GWS) then
  W1=TDS (1) to TDS(GWS)
  Same repetitions for W2
        end
  if (w2 > GWS)  then
  go to w3
     end
/*the following algorithm is for candidate generation phase and find out the maximum
frequent item set */
    do item = item of transaction
   For (i=o; i<=EOF TDS;i++)
  If (item ==TDS (i)) then
    I_count =I_count+1;
       end
  if (I_count >=MS)  then
  CG1 = w2 (FI (k-1)/*candidate item generation step */
  if (CG1>=MS) then
  CG2=w3=FI (K)   {CG2>=MS}
  Item = w3;
  Go to step 1;
  While EOF TDS
  end if
  end for
  Output = MFI (maximum frequent item set)
```

4. EXPERIMENTS

In this experiment we shows the result of proposed MFITSW (maximum frequent item set transaction sensitive sliding window) algorithm **.**All the program are implemented using Microsoft visual studio 2010(DOT NET FRAMEWORK 4.0) and performed on a 4 GHz Intel i3 PC machine with 512 MB memory running on windows 7.

From the description of the algorithm, it is clear that the mining MFI maximum frequent item set over online is very efficient by maintaining the various item set in the various transactions. We have to implemented our sliding window algorithm program in the DOT NET FRAMEWORK 4.0 MICROSOFT VISUAL STUDIO 2010 .We have to compare and testing frequent item set mining over MFITSW maximum frequent item set transaction sensitive sliding window algorithm implementation using with IBM Synthetic Data Generator proposed by Agrawal and Srikant.

Fig:2 shows the synthetic data stream, denoted by T5.I4.D1000K of size of 1 million transactions has an average of 5 of items(T5) with average maximum frequent item set size of 4 items (I4).

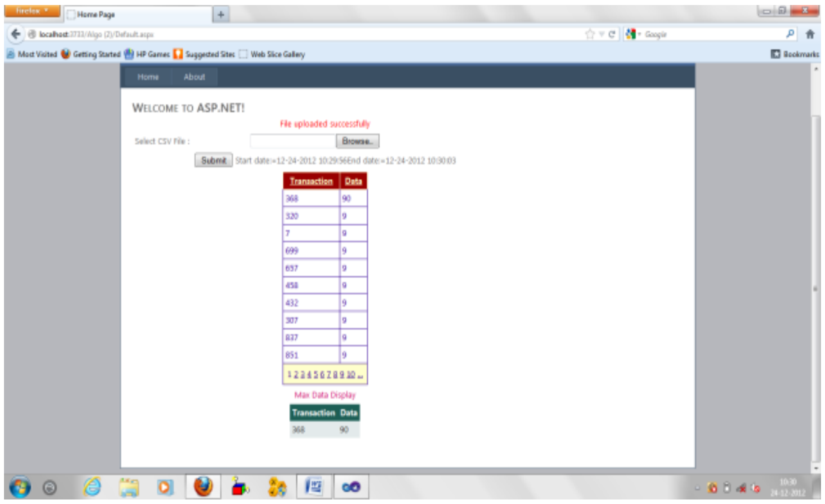


```
1  21 32 184 240 274 328 368 448 538 561 630 687 730 775 825 834
2  39 120 124 205 401 581 704 814 825 834
3  35 249 674 712 733 759 854 950
4  39 422 449 704 825 857 895 937 954 964
5  15 229 282 283 294 352 381 708 738 750 853 883 900 979
6  26 104 143 320 369 620 798
7  7 183 214 350 529 658 682 782 809 849 883 947 970 979
8  227 390
9  71 192 208 272 279 280 300 333 496 529 530 597 618 674 675 720 855 914 932
10 183 193 217 256 276 277 374 474 483 496 512 529 626 653 706 878 939
11 161 175 177 424 490 571 597 623 766 795 853 910 960
12 125 130 327 698 699 839
13 392 461 569 801 862
14 27 78 104 177 733 775 781 845 900 921 938
15 101 147 229 350 411 461 572 579 657 675 778 803 842 903
16 71 208 217 266 279 290 458 478 523 614 766 853 888 944 969
17 43 70 176 204 227 334 369 480 513 701 708 835 874 895
18 24 52 278 740
19 151 432 504 830 890
20 71 73 118 274 310 327 368 419 449 469 484 766 722 795 810 844 846 918
21 130 274 432 528 967
22 188 307 326 381 403 523 526 722 774 788 789 834 950 975
23 89 116 198 201 333 393 658 720 846
24 70 171 227 289 462 338 561 623 674 701 800 960 961
25 143 192 317 471 487 631 638 640 678 735 780 865 888 935
```

Fig:-5 transactions data stream.

Fig:3 shows the significantly outperforms for memory consumption and CPU cost and take less time to take out the maximum frequent item set of an online data stream. The sliding windows are based upon the user requriments.If the user gives the value of GWS (given window size).

In this experiment first of download the synthetic data steams using IBM synthetic data generator purposed by Agrawal and Srikant. After that converts the data stream into the CSV format and run this CSV data stream into Microsoft visual studio framework 2010 and first of all design the program in the design file drag and drops the label, text and button from the toolbox and make the code or programs in the source file after that running the programs and get the output. Select the CSV file of online transaction data streams and click on the submit button for find out the maximum frequent item set as an output.



WELCOME TO ASP.NET!

File uploaded successfully

Select CSV File :

Start date:=12-24-2012 10:29:56End date:=12-24-2012 10:30:03

Transaction	Data
368	90
320	9
7	9
699	9
657	9
458	9
432	9
307	9
837	9
851	9
1 2 3 4 5 6 7 8 9 10 ...	

Max Data Display

Transaction	Data
368	90

Fig:-6 output as maximum frequent item set.

Experiment 1

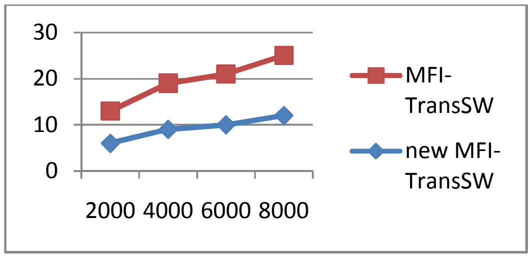


Fig: 7 number of transactions and processing time in ms.

Figure 7 shows that the graph between the processing time in mille second and the number of incoming transaction. The new MFI-TransSW takes less time than the MFI-TransSW.

Experiment 2

In the experiment 2 we compare our algorithm on the basis of the memory (MB) and the number of incoming sliding window in the generating frequent item set phase. This experiment shows that purposed algorithm take little bit less memory then the existing one. We have performed my algorithm in Dot Net framework. This algorithm increase the processing time as well as take less time for generating

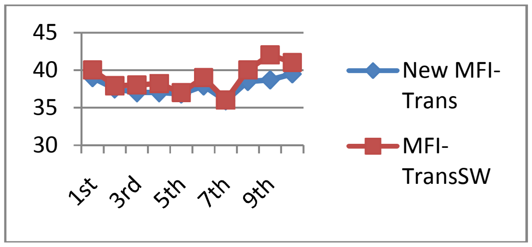


Fig: 8 incoming sliding window and memory uses infrequent item set generation phase.

5. CONCLUDING REMARKS

As we know that find out the maximum frequent item set of online transaction data stream is very important concept .In this paper proposes a method for finding recently frequent item sets over a data stream based on MFITSW mining frequent item set transaction sensitive sliding window algorithm. In order to support various needs of data stream analysis, experiment shows that the proposed not only attained the highly accuracy mining result but also run significant faster and consume less memory then the existing algorithms for mining frequent item sets. In this paper the interesting recent range of a data stream is defined by the size of a window. The proposed method can be employed to monitor the recent change of embedded knowledge in a data stream. The proposed algorithm gave a guarantee of the output quality and also a bound on the memory usage.

REFERENCES

QuestDataMiningSyntheticDataGenerationCode.Available:http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/datamining/datasets/syndata.html:-.

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J.Widom. Models and Issues in Data Stream Systems. In Proc. of the 2002 ACM Symposium on Principles of Database Systems (PODS 2002), ACM Press, 2002.

[2] H.-F. Li, S.-Y. Lee/Expert Systems with Applications 36 (2009) 1466–1477.

[3] Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining and Knowledge Discovery, 8(1), 53–87.

[4] Roberto Bayardo. Efficiently Mining Long Patterns from Databases. In ACM SIGMOD Conference, 1998.

[5] J. Chang and W. Lee. Finding Recent Frequent Item sets adaptively over Online Data Streams. In Proc. of the 9th ACM SIGKDD International Conference & Data Mining (KDD-2003), 2003.

[6] V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining Data Streams under Block Evolution. SIGKDD Exploration, 3(2):1-10, Jan. 2002.

[7] C. Giannella, J. Han, J. Pei, X. Yan and P. S. Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In Proc. of the NSF Workshop on Next Generation Data Mining, 2002.

[8] G. S. Manku and R. Motwani. Approximate Frequency Counts Over Data Streams. In Proc. of the 28th VLDB conference, 2002.

[9] W.G. Teng, M.-S. Chen and P. S. Yu. A Regression Based Temporal Pattern Mining Scheme for Data Streams.

[10] International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.2, No.6, November 2012.

[11] J. Han, H. Cheng, D. Xin, & X. Yan. (2007) "Frequent pattern mining: current status and future directions", Data Mining and Knowledge Discovery, vol. 15(1), pp. 55–86.

[12] Hua-Fu Li, Chin-Chuan Ho, Man-Kwan Shan, and Suh-Yin Lee, (October 8-11, 2006) " Efficient Maintenance and Mining of Frequent Item sets over Online Data Streams with a Sliding Window", IEEE international Conference on Systems, Man, and Cybernetics Taipei, Taiwan.

[13] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy,(2005) "Resource aware Mining of data streams", Journal of universal computer science,vol II ,pp 1440-1453.

[14] K.FJea,C ,W Li,T P Chang, "An efficient approximate approach to mining frequent item sets over high speed transactional data streams " ,Eighth International conference on intelligent system design and applications, DOI 10.1109/ISDA2008.74 .

[15] Jia Ren, Ke LI , (July 2008) " Online Data Stream Mining Of Recent Frequent Item sets Based On Sliding Window Model ",Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming.

[16] J.H. Chang and W.S. Lee, (2003).Finding recent frequent item sets adaptively over online data streams. In Proc. of the 9th ACM SIGKDD, pp. 487-492.

[17] D Lee, W Lee. (2005). "Finding Maximal Frequent Item sets over Online Data Streams Adaptively", Fifth Intl. Conference on Data Mining.

[18] Mozafari, H. Thakkar, and C. Zaniolo. Verifying and mining frequent patterns from large windows over data streams. In Proc. Int. Conf. on Data Engineering, pages 179–188, Los Alamitos, CA, 2008. IEEE Computer Society.

[19] Caixia Meng, (2009)"An efficient algorithm for mining frequent patterns over high speed data streams", World congress on software engineering.

[20] H.-F. Li, S.-Y. Lee, M.-K. Shan, An efficient algorithm for mining frequent item sets over the entire history of data streams, in: Proc. International workshop on Knowledge Discovery in Data Streams, 2004.

[21] C.K.-S. Leung, Q.I. Khan, and DSTree: a tree structure for the mining of frequent sets from data streams, in: Proc. ICDM, 2006, pp. 928–932. [23] C.K.-S. Leung, Q.I. Khan, Efficient mining of constrained frequent patterns from streams, in: Proc. 10th International Database Engineering and Applications Symposium, 2006.

[22] H.-F. Li, S.-Y. Lee, Mining frequent item sets over data streams using efficient window sliding techniques, Expert Systems with Applications 36 (2009) 1466–1477.

[23] G.S. Manku, R. Motwani, Approximate frequency counts over data streams, in: Proc. VLDB, 2002, pp. 346–357.
[24] B. Mozafari, H. Thakkar, C. Zaniolo, Verifying and mining frequent patterns from large windows over data streams, in: Proc. ICDE, 2008 pp. 179–188.
[25] J. Li, D. Maier, K. Tuftel, V. Papadimos, P.A. Tucker, No pane, no gain: efficient evaluation of sliding-window aggregates over data streams, SIGM ODRecord 34 (1) (2005) 39–44.
[26] C.-H. Lin, D.-Y. Chiu, Y.-H. Wu, A.L.P. Chen, Mining Frequent item sets from data streams with a time-sensitive sliding window, in: Proc. SIAM International Conference on Data Mining, 2005.
[27] J.X. Yu, Z. Chong, H. Lu, Z. Zhang, A. Zhou, a False negative approach to mining frequent item sets from high speed transactional data streams, Information Sciences 176 (14) (2006) 1986–2015.
[28] J.X. Yu, Z. Chong, H.Lu, A. Zhou, False Positive or false negative: mining frequent item sets from high speed transactional data streams, in: Proc. VLDB, 2004, pp. 204–215.
[29] S.K. Tanbeer, C.F. Ahmed, B.-S. Jeong, Y.-K. Lee, CP-tree: A tree structure for single-pass frequent pattern mining, in: T. Washio Et al. (Eds.), Proc. PAKDD, 2008, pp.1022–1027.
[30] Y.J. Tsay, J.Y. Chiang, An efficient cluster and decomposition algorithm for mining association rules, Information Sciences 160(1–4) (2004) 161–171.

.

AUTHORS

Neeraj received B.E (Information Technology) from DAV College of Engineering and Technology from Kanina (Haryana), INDIA. During 2006-2010 & pursuing M-tech (Software Engineering) from ITM University Gurgaon (INDIA) during 2011-2013.