ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st&22ndMarch, Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Mining Frequent Patterns with Screening of Null Transactions Using Different Models

B.Subbulakshmi, A. Periya Nayaki, Dr. C. Deisy

Assistant Professor, Dept of Computer Science & Engineering, Thiagarajar College of Engineering, Madurai. Tamil Nadu, India.

PG Student, Dept of Computer Science & Engineering, Thiagarajar College of Engineering, Madurai. Tamil Nadu, India.

Associate Professor, Dept of Computer Science & Engineering, Thiagarajar College of Engineering, Madurai, Tamil Nadu, India.

Abstract— Frequent Pattern Mining plays an important role in the field of data mining community today. The concept of frequent pattern mining can be extended to dynamic databases and data streams. A data stream represents a massive input data that arrives at high speed and is unbounded. There are various data processing models in data streams. The challenge in frequent pattern mining is the presence of null transactions. Null transaction is a transaction that does not contain any itemset being examined. Most of the existing streaming algorithms did not consider the overhead of null transactions. Hence, they fail to discover the frequent patterns faster and occupy lot of memory space to represent frequent items. To overcome this issue, a new algorithm called Screening of Null Transactions-Frequent Pattern Mining over Data Streams (SNT-FPMoDS) has been proposed which extracts frequent patterns using landmark and sliding window models. Experimental results using real datasets on different models show that our proposed algorithm saves lot of computation time and memory.

Keywords— Data Stream, Landmark Window Model, Sliding Window Model

I. INTRODUCTION

Data Mining is used to discover the patterns from huge databases. In the knowledge discovery process, frequent pattern mining [3] is one of the fundamental and interesting problems [10] to find the frequent patterns within the dataset. The problem of frequent pattern mining is not limited to static databases and also extended to dynamic databases [9] and data streams.

data stream is a massive input data that arrives at high speed and is unbounded. Examples of data stream include telecommunication, sensors, stock market analysis, web click streams data, etc. In data streams, concept change is one of the important phenomena because data is not static here. So, whenever a significant change occurs, it may require more memory and processing power and it may produce inaccurate results.

There are three data processing models [7] namely Landmark Model, Damped Model or Time Fading Model, Sliding Window Model. Landmark Model extracts the frequent patterns over the entire history of data streams.

Time Fading Model or Damped Model brings the frequent patterns with respect to time or based on the weight assigned to each transaction. Sliding Window Model processes only the recent transactions and gives the recent frequent patterns in the result. One of the three models should be chosen for stream mining process based upon the application selected.

In addition to that, a single item in a transaction does not give any association for pattern mining. This transaction is known as null transaction. The performance degrades due to the presence of null transactions in a dataset.

Main Contribution of this paper is depicted as follows. The major work is to mine the frequent patterns over data streams using different models. A new algorithm called SNT-FPMoDS (Screening of Null Transactions-Frequent Pattern Mining over Data Streams) has been proposed and it was implemented using Landmark and Sliding Window models. Screening of null transactions contributes towards the reduction in number of frequent patterns, memory storage and executing time.

II. RELATED WORKS

In 2002 [2], Manku and Motwani proposed an algorithm called lossy counting for frequent items mining and then extended it for frequent itemset mining. In this work, all frequent itemsets are outputted with an error bound and there are no false negatives in their result. In 2004, Chris Gianella et al. [5] proposed a new algorithm using tilted time window model to mine the complete set of frequent patterns over data streams. In 2004, Li and Lee [4] proposed an algorithm called DSM-FI for approximate mining of frequent itemsets over an entire history of data streams. Here, lots of tree traversals are required to collect the frequency information. In 2006, Leung et al. [8] proposed an algorithm called DSTree, prefix tree based data structure is used to maintain recent frequent patterns. Here, Sliding window model is used to mine the complete set of recent frequent patterns. Each node of prefix tree contains the transaction information. In 2008, Mozafari et al. [11] proposed an algorithm for mining frequent patterns using sliding window model. In 2009, Tanbeer et al [12] proposed an algorithm CPS-tree, a prefix based data structure is used to maintain the recent and exact information. Insertion and Restructuring phases are repeatedly executed while data stream processing. In 2009, Li et al. [13] proposed the efficient sliding window techniques called MFI-TransSW and MFI-TimeSW for frequent pattern mining over data streams. In 2010, Calders et al. [14] proposed an approximate algorithm for mining top-k frequent items with max-frequency. In 2011, Binesh Nair et al. [15] proposed an algorithm called CFIM-P, to mine the closed frequent patterns over static data with the elimination of null transactions.

III. PRELIMINARIES AND DEFINITIONS

Let S be a stream of transactions and I= $\{i_1, i_2,\ldots, i_m\}$ be the set of items. For an itemset Y, which is a subset of I, a transaction T in S which contains an itemset Y if $Y \subseteq T$. The Support of Y is defined as the fraction of received transactions that contains the itemset Y. If the support (Y) is greater than or equal to the user given minimum support threshold value, then the item is said to be frequent.

Definition 1: (Landmark Window Model)

Landmark window model is a data processing model in data stream which maintains the history information from the landmark starting point (t_{sp}) to the current point (t_{cp}). If transaction t_m is valid,

$$t_{sp} \leq t_m \leq t_{cp}$$

Definition 2: (Sliding Window Model)

Sliding Window Model is one of the data processing model in data streams which process and maintains only the recent transactional data,

$$t_{n-|w|+1} \leq t_m \leq t_i$$

Where $t_{n-|w|+1}$ and t_i are the window's identifier and i^{th} received transaction.

Problem Statement: Given the data stream s, size of window and minimum support threshold, the problem is to find the all frequent itemsets using Landmark and Sliding Window model by eliminating the null transactions.

IV. PROPOSED METHOD

The method proposed here is based on Eclat algorithm which is used for mining all frequent itemsets operating on the vertical layout of database [6]. A new algorithm has been proposed with elimination of null transactions for mining all frequent patterns over data streams using landmark window and sliding window models. The modules identified are Elimination of Null Transactions, Window Initialization, Pane Insertion and Frequent Patterns Maintenance. First three modules are common in both of these models. Maintenance of frequent patterns is done differently by each of the models.

Elimination of Null Transactions: Null Transaction is a transaction which contains a single item in a dataset. This transaction does not give any information for association. An attempt has been made to eliminate the null transactions [15] in order to reduce the processing time for finding k-frequent itemsets. It saves lot of memory when the patterns are maintained in the tree.

Window Initialization: The window initialization phase is activated while the number of transactions generated so far in a transaction data stream is equal to the window size (ws).

Pane Insertion: Adding a single transaction or a batch of transaction to an existing window is called as pane insertion. Due to efficiency issues, addition or deletion of transactions from window is in batch wise. The number of transactions are added to a window is equal to the pane size (ps).

These three phases are common for mining frequent patterns with elimination of null transactions over data streams using landmark ($SNT\text{-}FPMoDS_{LW}$) and sliding window ($SNT\text{-}FPMoDS_{SW}$) models but differ in frequent patterns maintenance phase. This phase is briefly explained along with algorithm for different models in following subsections.

A. $SNT\text{-}FPMoDS_{LW}$ – Proposed Method of Landmark window model:

In this section, the concept of our proposed method $SNT\text{-}FPMoDS_{LW}$ (Screening of Null Transactions-Frequent Pattern Mining over Data Streams using Landmark Window Model) is briefly explained. This model is used for users who are interested in historical data from a landmark time. The modules of the algorithm are as follows:

1. Elimination of Null Transactions
2. Window Initialization
3. Pane Insertion
4. Maintenance of Frequent Pattern Tree for Landmark Window (FPT-LW).

Window is initialized after eliminating the null transactions and frequent patterns are mined using Eclat algorithm [1]. This algorithm is operating on the vertical

layout of database [16, 19]. This layout contains the tidlists of items in the database. The support of all single items is directly extracted from the vertical layout of dataset in a single scan. If the support value is greater than or equal to the user defined minimum support threshold then those single items are considered as frequent items. Then, we do the intersection operation over tidlists of frequent single items to obtain the support value for candidate 2-itemsets. The process is repeated until all the frequent patterns are extracted from the initial window. When a pane or batch of transactions is inserted to the window, the frequent patterns are extracted in the same way from pane and the results are updated.

i) Maintenance of Frequent Pattern Tree for Landmark Window (FPT-LW)

A new data structure namely FPT-LW has been proposed to maintain the frequent patterns during data stream processing. The extracted frequent patterns from the initial window are inserted into FPT-LW tree. After every pane insertion, frequent patterns are updated to this tree.

Procedure for FPT-LW Updating
Input: FPT-LW tree structure (Window)
Output: Updated FPT-LW tree

1. Search for node of item in FPT-LW tree
2. **if** node does not exists **then**
   Create a node of item in that tree
   else
   Update the tid's of item <p.tid's> in corresponding node
   //p.tid's= Item tid's after pane insertion
   end if

Fig. 1 Procedure for FPT-LW Updating

Fig. 1 shows the procedure for updating the FPT-LW tree. FPT-LW tree contains two attributes: {itemset (Y), tid's of itemset}. From this FPT-LW, it is possible to get history of information about frequent patterns in a stream of data.

The sample structure of FPT-LW tree is shown in Fig 2.

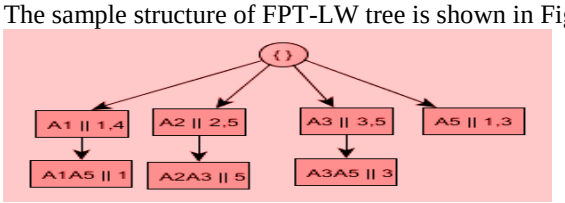


Fig. 2 FPT-LW Tree (Initial Window)

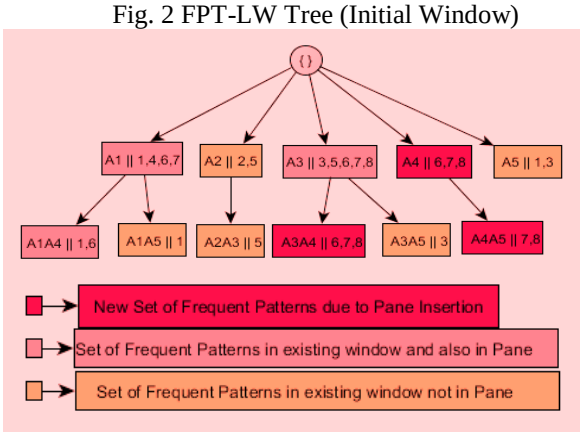


Fig. 3 FPT-LW Tree Updation

Initially, the root node is null. The frequent patterns corresponding to the initial window are stored in upcoming levels with their tidsets. Fig. 3 shows the updated FPT-LW tree after the insertion of pane into the initial window. Due to pane insertion, some frequent patterns may newly emerge and some frequent patterns in the initial window may become infrequent or they continue to be frequent as such. They are shown in Fig 3. in different colors.

The algorithm **SNT-FPMoDS$_{LW}$** is presented in Fig. 4. In this algorithm, ws, ps and σ are the parameters given by the user. In Step 1, the null transactions are dropped from the experimental dataset. In Step 2, the window is initialized using window size. Transactional window has been filled with transactions upto this window size. The set of frequent patterns are mined using Eclat algorithm and are stored in Frequent Patterns Set (FPSet). These frequent patterns are then inserted into the Frequent Pattern Tree-Landmark Window (FPT-LW). In Step 3, the pane size is initialized. After a pane is inserted into the initial window, the frequent patterns are mined incrementally. The results are updated in FPT-LW tree. This algorithm maintains the whole set of frequent patterns from the landmark time.

Input: Data Stream (S), Window Size(ws), Pane Size (ps),Minimum Support Threshold % (σ)
Output: a set of frequent patterns
Procedure:
Step 1: Scan the original dataset and drop the null transactions.
for each transaction (T) in S **do**
  if (tid → single item(i))
    delete (T[tid,i]);
  end if
end for
Step 2: Window Initalization (ws,σ)
Initialize FPT-LW = {empty}

for each transaction T_i in S **do**
  insert transaction (T_i) in TW
  if (TW_{ws}=FULL) **then**
    FPSet = Eclat (TW, σ)
    insert (FPT-LW,FPSet)
  end if
end for
Step 3: Pane Insertion (ps, σ)
Initialize p=1;
while (p≤psize)
  insert transaction (T_i) in TW
  p++;
end while
for each transaction in TW not in ws **do**
  if (TW_{ps} = FULL) **then**
    //Compute Support
    Update FPSet (TW_{ps} , σ)
    Update (FPT-LW, FPSet)
  end if
end for

Fig. 4 SNT-FPMoDS $_{LW}$ Algorithm

B. SNT-FPMoDS$_{SW}$ - Proposed Method of Sliding Window Model:

In this section, the Screening of Null Transactions-Frequent Pattern Mining over Data Streams using Sliding Window (SNT-FPMoDS$_{SW}$) algorithm is explained briefly. Sliding window model finds the frequent patterns over data streams by considering only the recent set of transactions. When the window slides, the old invalid transactions are deleted and new valid transactions are inserted into the window. The frequent patterns are updated in the tree because of this deletion. There are two types of sliding window: i) Transaction-Sensitive Sliding Window ii) Time-Sensitive Sliding Window. The mining process used in this paper is based on transaction-sensitive sliding window model which maintains the fixed size of transactions in a window. The sliding window model is appropriate for those users who are interested to know only the recent information.

The modules of this algorithm are as follows:

1. Elimination of Null Transactions
2. Window Initialization
3. Pane Insertion
4. Maintenance of Frequent Pattern Tree for Sliding Window (FPT-SW)

The concept of first three modules is same as that of landmark window model except the frequent patterns maintenance phase.

i) Maintenance of Frequent Pattern Tree for Sliding Window (FPT-SW)

A new data structure called FPT-SW is used to store the frequent patterns mined using sliding window model. Initially, the extracted frequent patterns from the initial window are stored in FPT-SW tree. Then after every pane insertion, window sliding phase is activated. During this window sliding phase, old information is deleted from the FPT-SW tree and new patterns are inserted into the tree. In Fig. 5, the procedure for updating the frequent patterns after pane insertion is given.

Procedure for FPT-SW Updating
Input: FPT-SW tree structure (Window)
Output: Updated FPT-SW tree

1. Search for node of item in FPT-SW tree
2. **if** node does not exists **then**
   Create a node of item in that tree
   else
   Modify the tid's of item <p.tid's> in corresponding node
   //p.tid's= Item tid's after pane insertion
   end if

Fig. 5 Procedure for FPT-SW Updating

Each node in a FPT-SW tree includes 2 fields: {itemset (Y), a set of transactions that contains the itemset (Y)}. The sample FPT-SW tree is given below:

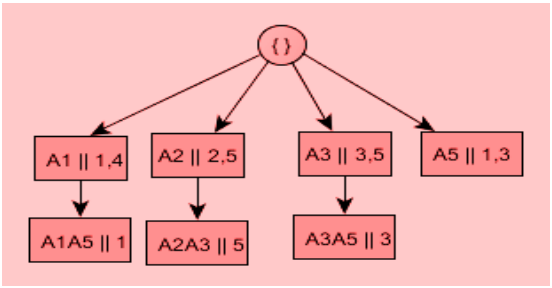


Fig. 6 FPT-SW Tree (Initial Window)

Fig. 6 stores the set of frequent patterns for initial window. Each node contains an itemset with their corresponding tidlists.

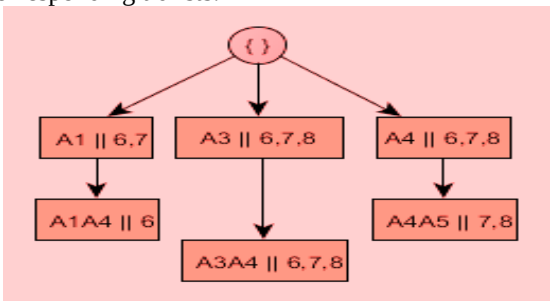


Fig. 7 FPT-SW Tree Updation

In Fig. 7, the updated frequent patterns are shown after a pane was inserted and processed. The old information in FPT-SW tree is deleted because the sliding window model considers only the recent transactions.

Input: Data Stream (S), Window Size(ws), Pane Size (ps),Minimum Support Threshold % (σ)
Output: a set of frequent patterns
Procedure:
Step 1: Scan the original dataset and drop the null transactions.

```
for each transaction (T) in S do
        if (tid → single item(i))
                delete (T[tid,i]);
        end if
end for
```

Step 2: Window Initalization (ws,σ)
Initialize FPT-SW = {}, CP= Last-Tid (TSW)

```
for each transaction Ti in S do
        insert transaction (Ti) in TSW
        if (TSWws=FULL) then
                FPSet = Eclat(TSW, σ)
                insert (FPT-SW,FPSet)
        end if
end for
```

Step 3: Pane Insertion (ps, σ)
Initialize p=1;

```
while (p≤psize)
        insert transaction (Ti) in TSW
        p++;
end while
for each transaction in TSW not in ws do
        if (TSWps = FULL) then
                //Compute Support
                Update FPSet (TSWps , σ)
                Stale = Cut (CP,TSW)
                DeleteEffect[FPT-SW]
```

Update(FPT-SW, FPSet)
end if
end for

Fig. 8 SNT-FPMoDS$_{SW}$ Algorithm

The **SNT-FPMoDS$_{SW}$** algorithm is presented in Fig.8. In Step1, all the null transactions from the dataset are removed. In Step2, window size is initialized and checkpoint is marked at the last transaction id (tid) of initial window. The mining of frequent patterns was as similar to the landmark window model. The frequent patterns (FPSet) are inserted into FPT-SW tree.

After inserting the batch of transactions, the set of frequent patterns are updated in FPSet. The stale transactions denoting previous concept upto the checkpoint was deleted by removing corresponding tidlists from window and effect of these transactions are also deleted from the FPT-SW tree. Now the FPT-SW tree stores only the updated / recent frequent patterns.

V. PERFORMANCE ANALYSIS

In this section, we first describe the experimental setup and then illustrate the results of the proposed algorithms.

A. Experimental Setup

The algorithms were written in Java and compiled using Netbeans IDE 7.3 version. The operating system used to run these algorithms is windows7 and the processor was Intel® Core i5-3230M, CPU@2.60GHz with 4GB of RAM.

The experiments were done on two real life benchmark datasets namely BMS-WebView-1 and BMS-WebView-2. Table1. gives some features of above mentioned datasets as follows:

TABLE I. DATASET CHARACTERISTICS

Name of Dataset	No. of Transactions	No.of Unique Items	Average TL
BMS-WebView-1	59,602	497	2.50
BMS-WebView-2	77,512	3340	5.0

BMS-WebView-1 and BMS-WebView-2 are the real life dataset from a small dot com company named as Gazelle.com. It contains several months of click stream data from an ecommerce website. These datasets were considered as stream data because clicking behaviors of the customers was changed over a time. We executed experiments on these datasets for mining frequent patterns by setting up the following parameters: Window Size (ws), Pane Size (ps), and Minimum Support (σ). The parameter which is used for mining frequent pattern is σ (minimum support). The pattern which is greater than or equal to σ is considered as a frequent pattern.

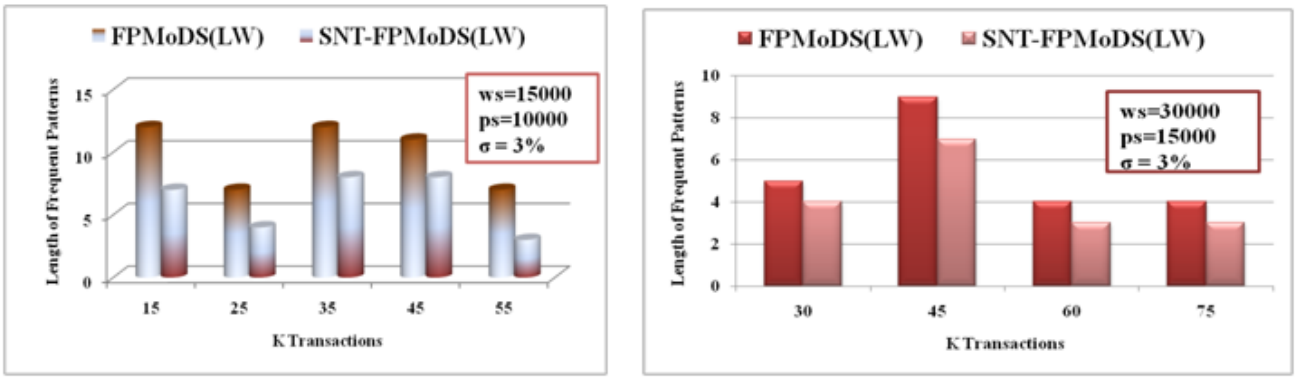
In this paper, the performance of the proposed algorithm SNT-FPMoDS is compared with FPMoDS on landmark and sliding window models.

B. Experimental Results- Performance for SNT-FPMoDS$_{LW}$ & SNT-FPMoDS$_{SW}$

In this section, the performance evaluation of proposed algorithms SNT-FPMoDS$_{LW}$ and SNT-FPMoDS$_{SW}$ is presented. First, we compared the number of frequent patterns produced by proposed and existing algorithms for the landmark model (FPMoDS$_{LW}$ & SNT-FPMoDS$_{LW}$) under varying minimum support thresholds using BMS WebView-1 and BMS WebView-2 datasets and the results are shown in Fig. 9. In the second experiment, we compared the running time of two algorithms for landmark model and it is shown in Fig. 10.

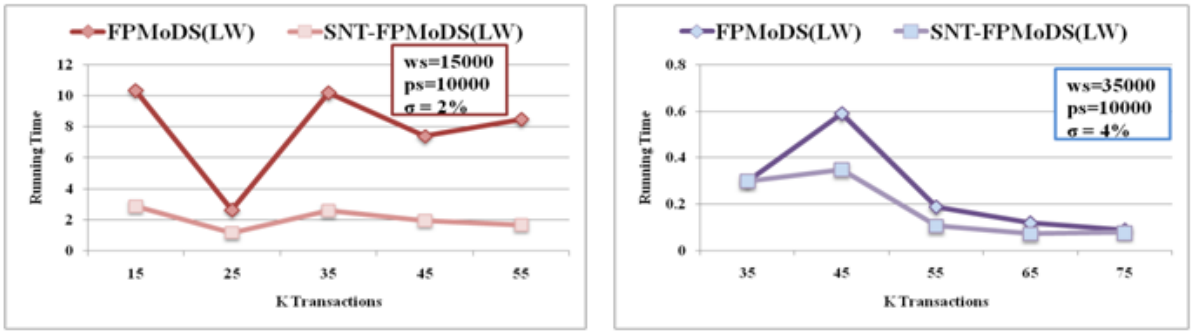
Fig. 11 shows the comparison of number of frequent patterns and memory storage for the sliding window model with different window size and pane size. It is observed from the figure that screening of null transactions gives better result than the existing algorithm. We also compared the execution time of the proposed algorithm with the existing algorithm in Fig. 12 for the sliding window model.

Depending upon the type of the application, the landmark or sliding window model can be used to mine frequent patterns by eliminating all null transactions earlier.



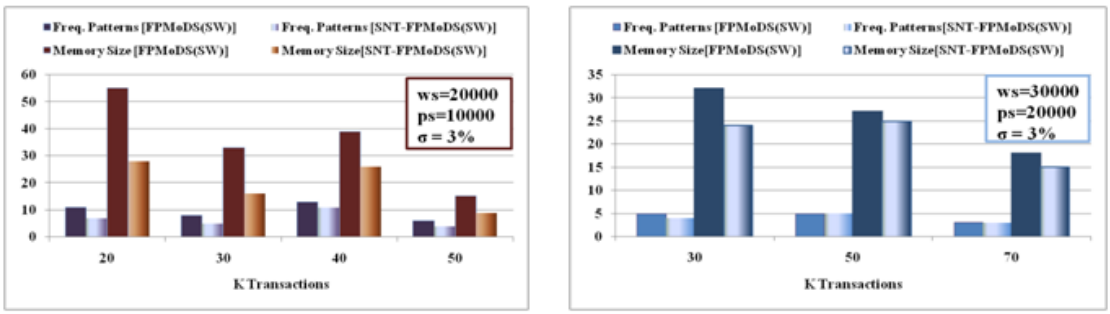
(a) BMS Web View-1 (b) BMS Web View-2

Fig. 9 Performance Comparison of Length of Frequent Patterns



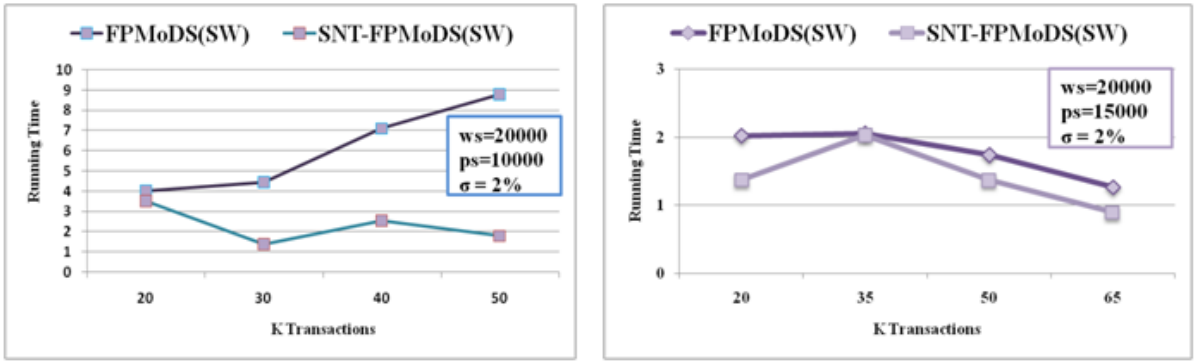
(a) BMS Web View-1 (b) BMS Web View-2

Fig. 10 Comparison of Execution Time



(a) BMS Web View-1 (b) BMS Web View-2

Fig. 11 Comparison of Frequent Patterns and Memory Size



(a) BMS Web View-1 (b) BMS Web View-2

Fig. 12 Run Time Comparison

VI. CONCLUSION

In this paper, we proposed new methods namely SNT-FPMoDS (SNT-FPMoDS$_{LW}$ & SNT-FPMoDS$_{SW}$) to efficiently mine the frequent patterns from data streams using landmark and sliding window models. The selection of algorithm and the data processing model is based upon the application requirements. Landmark model is best suited for applications where users are interested in mining patterns over a period of time whereas sliding window model is used to analyze frequent patterns genera

-ted from the recent information. However, the presence of null transactions in a stream of transactions degrades the performance of pattern discovery process. The proposed work has attempted to resolve this issue by eliminating those null transactions in a data stream. The SNT-FPMoDS algorithm is experimentally evaluated on real-life datasets with different window size and minimum support thresholds on both models. From the experimental results, it is observed that the SNT-FPMoDS algorithm mines the complete set of frequent patterns than the FPMoDS with less memory usage and runtime for both landmark window and sliding window models.

REFERENCES

[1] M. Zaki, "Scalable Algorithms for Association Mining," *IEEE Transactions on Knowledge and Data Engineering*, no. 12, pp. 372- 390, 2000.

[2] G. S. Manku and R. Motwani, "Approximate Frequency Counts over Data Streams," *in proceedings of the 28th international conference on very large databases, Hong Kong*, pp. 346-357, 2002.

[3] R. Agarwal, R. Srikant, "Fast Algorithms for mining association rules in large databases," *in proceedings of the 20th international conference on very large databases*, pp. 487-499, 2004.

[4] 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*, 2004.

[5] Chris Gianella, Jiawei Han, Jian Pei, Xifeng Yan, Philip S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities," *In: Proceedings of Data Mining: next generation challenges and future directions, MIT/AAAI Press*, pp. 191-212, 2004.

[6] Bart Geothals, "Frequent Set Mining," *Data Mining and Knowledge Discovery Handbook*, pp. 377-397, 2005.

[7] Nan Jiang and Le Gruenwald, "Researh Issues in Data Stream Association Rule Mining," *ACM SIGMOD Record*, vol. 35, no. 1, pp. 14-19, 2006.

[8] C.K.-S. Leung, Q.I. Khan, "DSTree: a tree structure for the mining of frequent sets from data streams," *in Proceedings of ICDM*, pp. 928-932, 2006.

[9] S. Zhang, J. Zhang, C. Zhang, "EDUA: an efficient algorithm for dynamic database mining," *Information Sciences*,vol. 177, pp. 2756-2767, 2007.

[10] J. Han, H. Cheng, D. Xin, X. Yan, "Frequent Pattern Mining: Current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp . 55-86, 2007.

[11] B. Mozafari, H. Thakkar, C. Zaniolo, "Verifying and mining frequent patterns from large windows over data streams," *in proceedings of international conference on ICDE*, pp. 179-188, 2008.

[12] S. K. Tanbeer, C. F. Ahmed, B. S. Jeong, Y. K. Lee, "Sliding window-based frequent pattern mining over data streams," *Information Sciences*, vol. 179, pp. 3843-3865, 2009.

[13] H. F. Li and S. Y. Lee, "Mining frequent itemsets over data streams using efficient window sliding techniques," *Expert Systems with Applications*, vol. 36, pp. 1466-1477, 2009.

[14] H. T. Lam and T. Calders, "Mining top-k frequent items in a data stream with flexible sliding windows," *in proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 283-292, 2010.

[15] Binesh Nair, Amiya Kumar Tripathy, "Accelerating Closed Frequent Itemset Mining by Elimination of Null Transactions," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2, no. 7, pp. 317-324, July 2011.

[16] M. Deypir, M. H. Sadreddini,"EclatDS: An efficient sliding window based frequent pattern mining method for data streams," *Intelligent Data Analysis*, vol. 15, no. 4, pp. 571-587, 2011.

[17] Bart Goethals, *Frequent Itemset Mining Dataset Repository*, Available: http://fimi.ua.ac.be/data/

[18] M. Deypir, M. H. Sadreddini, S. Hashemi, "Towards a variable size sliding window model for frequent itemset mining over data streams," *Computers & Industrial Engineering*, vol. 63, pp. 161-172, 2012.

[19] M. Deypir, M. H. Sadreddini, "A dynamic layout of sliding of frequent itemset mining over data streams," *The Journal of Systems and Software*, vol. 85, pp. 746-759, 2012.