IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

Mining Stream Data with Data Load Shedding Techniques using Self Adaptive Sliding Window Model

C.Nalini

Professor, Information Technology Department, Kongu Engineering College, Erode, India

Abstract: Frequent patterns are patterns that appear frequently in a data set. Frequent pattern mining searches for recurring relationships in a given data set. It plays an important role in mining associations and correlation analysis among data, is an important data mining task. This work focuses on discovering frequent item sets in data-stream environments which may suffer from data overload. Stream data refer to data that flow into a system in vast volumes, change dynamically and contain multidimensional features. The traditional frequent pattern algorithms are not suitable to find frequent patterns from stream data. This paper proposed a frequent pattern mining algorithm integrate two data overload handling mechanisms. It extracts basic information from streaming data i.e. frequency of data items and keeps as base information. On user requirement the frequent pattern mining algorithm generates frequent item set from base information by using approximate inclusion-exclusion technique to calculate the approximate counts of frequent item sets. Self adaptive sliding window time model has been implemented to process the data stream. When data overload exists, the algorithm chooses data overload mechanism based on the nature of the data. The experimental results showed that the mining algorithm performed well in data overload state and generated frequent item set.

Keywords: Data mining, Stream data, Frequent patterns, Approximate inclusion-exclusion, Adaptive sliding window model, Data overload handling mechanisms.

I. INTRODUCTION

Today, many commercial applications use online services. They have presented their data in the form of continuously transmitted stream, namely data streams. A stream data is ordered sequences of items that arrive continuously with high speed and have a data distribution that often changes with time [5]. Hence, there is a need to design an efficient frequent pattern mining algorithm for finding frequent item set with load shedding techniques. The association rule mining consists of two step processes. First, find frequent pattern from the database and then generate an association rule. Frequent pattern mining [1] helps to discover implicit, previously unknown, and potentially useful knowledge in the form of frequently occurring sets of items that are hidden in the data. Data stream mining is a technique to continuously discover useful information or knowledge from a large amount of running data elements. Data stream mining algorithms implements three kinds of time models[17] to process the stream data, i)Landmark window model ii) Damped/fading window model iii) Sliding window model. The landmark window model covers all data elements that have ever been received. In damped/fading window model, each data element is associated with a variable weight and recent elements have higher weights than previous ones. The sliding window model fixed window length which moves with time is given, and the range of mining covers the recent data elements contained within the window. Self adaptive sliding window model learns the sliding window control parameters dynamically adapt the window size when patterns are mined from the stream. A data stream mining system may suffer the problem of data overload. The transmission rate of a data stream is usually dynamic and its running speed may change with time. When the data transmission rate of the data-stream source exceeds the data processing rate of the mining algorithm of a mining system, e.g., during a peak period, the system is overloaded with data and can't handle all incoming data elements properly within a time-unit. And also produce flawed results or come into a crash. To solve this problem data overload mechanism should be incorporated within the algorithm to adequately deal with data overload. In this paper, two data overload mechanism has been proposed for discovering frequent patterns effectively in transactional data streams. The rest of this paper is organized as follows. In Section 2, related work regarding data-stream frequent-pattern mining and data-overload handling is described. Section 3 represents the problem definition. In Section 4, a mining algorithm together with two overload-handling mechanisms is proposed and explained in detail. Section 5 presents the analysis of experimental results. Finally, Section 6 concludes this work.

II. RELATED WORK

Apriori algorithm [1] is the first algorithm for mining frequent item sets. It has implemented an iterative approach known as a breadth-first search through the search space where k-item sets have used to explore (k+1)-item sets. It scanned the database each time to generate candidate item set. As a result, it has required more time complexity and space complexity to complete the task. Discovering frequent item set from data stream is more challenging as (i) data streams are continuous and unbounded (ii) data in the streams are not necessarily uniformly distributed. Frequent-pattern mining [2] from the data streams have initially limited to singleton items. Lossy Counting (LC) [13] is the first practical algorithm used to discover frequent item sets from transactional data streams. This algorithm has implemented under the landmark window model. In addition to the minimum-support parameter (ms), LC also uses an error-bound parameter ε to maintain those infrequent item sets having the potential to become frequent in the near future. LC has processed the stream data batch by batch, and each batch includes bucket(s) of transactions. With the use of parameter ε, when an item set newly found LC had known the upper-bound count of the frequent item set. In [3], the author has implemented LC algorithm using a sliding window model. The authors [14] suggested the Approximate Inclusion–Exclusion technique to obtain the frequent count of k-item set from frequent 2-item set. If all the subsets value had known, then the actual value of a union term would have been found easily. Apriori algorithm scanned the database multiple times to generate frequent item sets. However, it is not possible in data stream mining. Since the system couldn't retrieve the past data stream details from current data stream, has required more memory to store it.

The authors [11] proposed a data stream frequent pattern mining algorithm under the sliding window model, used DS-Tree data structure for capturing the contents of transactions in each batch of data within the current sliding window. It required more time to update the tree structure for every batch. According to the experimental results, the algorithm has attained 100% accuracy but not suitable for long stream transactions. Since this method should have enumerated item set for each transaction. CPS-Tree which has closely related to DS-Tree, the authors [16] used to discover frequent patterns from a data stream under sliding window model. Unlike DS-Tree, CPS-Tree stored the batch information only at the last node of each path to reduce the memory consumption.

The authors of [3] proposed Online Combinatorial Approximation (OCA) algorithm, use Apriori algorithm to find frequent 2-item sets. It generates higher order frequent item sets from frequent 2-item sets using Approximate Inclusion–Exclusion technique. It required less time to generate n-frequent item set compared with traditional algorithms, not suitable for mining frequent item sets over transactional data streams. The authors extended the algorithm and named as Data Stream Combinatorial Approximation algorithm (DSCA) [9] which has been implemented under the landmark window model and generates all item sets of length 1 and 2 existing in the data stream. The mining task used Approximate Inclusion-Exclusion technique to compute the approximate count of longer item sets during frequent pattern mining. It kept all the details in the base information to generate higher order frequent item sets.

The Loadstar system [5] measures the level of uncertainty in the classification model. The experimental results showed that offers a good solution to data-stream classification with the presence of system overload. The authors of [7] proposed a load controllable frequent pattern mining system which used LC under sliding window model. The system effectively handles the workload during peak periods. Three kinds of load shedding mechanisms proposed in [4] to address the data overload like i) Efficiency oriented policy ii) Complexity oriented policy iii) Accuracy oriented policy. The proposed work incorporates these load shedding mechanisms in frequent pattern mining task to handle load at peak periods.

III. PROBLEM DEFINITION

Let I ={x_1, x_2, . . ., x_z} be a set of attributes, and each x_i in I be a single item. An item set (or a pattern) X is a subset of I and written as X = x_i x_j. . .x_m. The length of an item set is the number of items it contains, and an item set of length l is called an l-item set. A transaction T consists of a set of ordered items, and T supports an item set X if X€T. The support of an item set X in a group of transactions is the number of occurrences of X within the group. An item set X of length l which is a subset of another item set Y or a transaction T is called a l-subset of Y or T. A transaction data stream on I is a continuous sequence of elements where each element is a transaction.

A threshold of minimum support (ms) and a size of the sliding window (sw) are two parameters specified by the user. The objective of frequent-pattern mining is to find frequent pattern from data elements within the current sliding window. There are two basic strategies have used to design a sliding window model in data stream mining i)count based window model ii)time based window model. Assume that there is a mining algorithm A running on a mining system and a data-stream source D. Let the processing rate of A on stream elements be z transactions in a time-unit, and the data transmission rate of D be w transactions at a time. A data stream is continuous and dynamic; its transmission rate is dynamic. The data load L at a point is defined to be the ratio of transmission rate to processing rate at that point,(i.e) L = w/z. As the value of w becomes greater and closer to that of z, the mining system which operates A will get busier in processing the continuously arriving data.

When the transmission rate of D becomes higher than the data processing rate of A (i.e., w > z or L is over 100%), the mining system is said to be data overloaded. In a data-overloaded situation, the system will receive a batch of w stream transactions within a time-unit whose quantity goes beyond its processing capability of z transactions. Data overload may cause a mining system to perform abnormally. It is an important issue for a data-stream mining algorithm which should not be neglected. The goal of this research work is to develop a mining algorithm together with mechanisms for overload management to support frequent-item set discovery under self adaptive sliding window model in a dynamic data-stream environment.

3.1 Self Adaptive Sliding Window Model

The proposed algorithm uses self adaptive sliding window model for processing the data stream. In this model, the total window (W) was divided into two sub-windows as W_0 and W_1. The length of an item sets in W_0 and W_1 is denoted as n_0 and n_1.respectively. The total number of transactions in the data stream was denoted as n. The harmonic mean(m) is calculated by using equation(1)

$$m = \frac{1}{\frac{1}{n_0} + \frac{1}{n_1}} \qquad (1)$$

The approximate confidence value was calculated with respect to user specified confidence value and the total number of transactions n. The approximate confidence value δ^{ι} is calculated by using equation(2)

$$\delta^{\iota} = \frac{\delta}{n} \qquad (2)$$

The window size of the self adaptive sliding window varies each and every time based on the batch that enters into the sliding window The ε_{cut} is used for splitting the sub-windows dynamically based on the batch size. The ε_{cut} value was obtained by harmonic mean and an approximate confidence value. The Equation 3 denotes the ε_{cut} value based on the current batch in the sliding window.

$$\varepsilon_{cut} = \sqrt{\frac{1}{2m} . \ln \frac{4}{\delta}} \qquad (3)$$

IV. FREQUENT PATTERN MINING ALGORITHM WITH LOAD SHEDDING MECHANISMS

4.1 Transaction data stream

The transaction data stream consists of set of transaction with its respective item set of market basket data. Table 1 denotes the sample transaction dataset. It consists of ten transactions. The item set I has 4 items like {milk, bread, cheese, butter}

Table 1 Sample Transaction Dataset

TID	ITEMS
T01	{milk, cheese, butter}
T02	{milk, bread, cheese, butter}
T03	{cheese, butter}
T04	{milk, bread, butter}
T05	{bread, butter}
T06	{bread, cheese}
T07	{milk, butter}
T08	{milk, bread, cheese}
T09	{milk, bread, butter}
T10	{milk, bread, cheese, butter}

4.2 Frequent 2- Item Set Generation

At first step, frequent item set mining algorithm computes the frequency count of 1 and 2-item sets by scanning the transaction from the data stream. The base information contains the frequency count of all 1-item sets and 2-item sets existing in the data stream by using Apriori algorithm.

4.1.3 Frequent N-Item Set Generation by using Approximate Inclusion-Exclusion Technique

4.1.3.1 Skip and Complete Technique

The skip and complete technique accelerates the speed of processing and increases the throughput of the frequent mining algorithm. For every incoming transaction, it finds all 1-item set, 2-item set and records their counts and stores the same in the base information. Assume that every transaction is sorted in order with its items in pattern-growth manner. In skip and complete technique, 3-item set is generated by joining a 2-item set with a 1-item, where a 2-itmset is called a prefix item set while a 1-item is called a stem item. Fig. 1 illustrates the process of skip and complete technique.

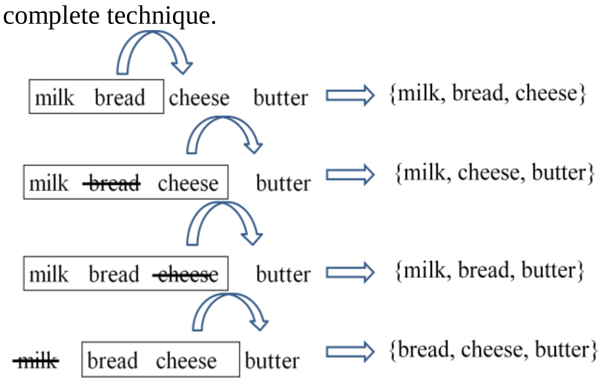

Fig. 1 Skip and complete technique

There is a transaction which consists of 4 items. For finding all 3-item set subsets involved in this transaction, a basic and intuitive way is to fix a certain prefix item set first, and then sequentially finds all stem items with the members of that prefix item set. Every found stem item joining with the prefix item set will result in a 3-item set. For the transaction in Fig. 1, it starts with the least prefix item set {milk,bread} to generate 3-item sets. When prefix {milk,bread} is done, it then proceeds with prefix {milk,cheese}, prefix {bread,cheese}, and so forth.

The basic idea behind this technique is that, even if it does not access all stem items of all prefix item sets in a transaction, it can still find all the 3-item set subsets contained in that transaction completely and correctly. Since it access all stem items of every prefix item set, the saving in access time may reduce the process time of handling every incoming transaction, hence the throughput of the incoming stream processing is improved.

4.1.3..2 Group Count Technique

The group count technique dealt with the intention to reduce the memory consumption of mining algorithm. If the user wishes to keep information about the entire item set of length 3 in the lexicographic order tree, it needs more than 166 million additional nodes for recording. This amount of nodes will consume a great deal of memory space, perhaps more than 1 GB. In the consideration between the approximate accuracy and the memory usage, it is designed a way to record the 3-item sets' counts in grouped form. 3-item sets which belong to the same 2-item set prefix are grouped by a number of n, and then their counts are recorded together. That is, it makes every 3-item sets to form a group, and increase the group's count when any member in that group is found in a transaction.

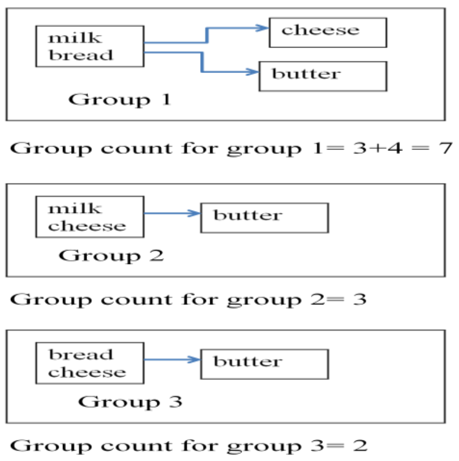
milk
bread
cheese
butter
Group 1
Group count for group 1= 3+4 = 7
milk
cheese
butter
Group 2
Group count for group 2= 3
bread
cheese
butter
Group 3
Group count for group 3= 2

Fig. 2 Group count technique

Fig. 2 illustrates the process of group count technique, for all 3-item sets which have 2-item set prefix {milk, bread} in the data stream, it make every four item sets to form a group which belongs to {milk, bread} and the number of all groups of {milk, bread} in proper order. When any member of {milk, bread, cheese},{milk, bread, butter} is found in a transaction , it increases the count of the group it belongs to. With this technique, it can keep the information about 3-item set with less memory usage.

4.1.3.3 Approximate Inclusion-Exclusion Technique

The frequent item set of length n can be calculated by using the approximate inclusion exclusion technique without continual scanning the data stream. The approximate inclusion exclusion technique is based on principle of inclusion exclusion technique. With this technique, approximate count of each frequent item set can be generated. The Principle of Inclusion and Exclusion is under combinatorial mathematics.

$$\left|A_1 \cup A_2 ... \cup A_m\right| = \sum \left|A_i\right| - \sum_{i<j} \left|A_I \cap A_j\right| + \sum_{i<j<k} \left|A_I \cap A_j \cap A_k\right| + ... + (-1)^{m+1} \left|A_1 \cap A_2 \cap ... A_m\right| \quad (4)$$

The Equation 4 is applied for calculating the support count of item sets $A_1 \cup A_2 ... \cup A_m$. It derives the support count for m-item set, only if all the subsets are given. If any of the subsets are not available, only the approximate count can be calculated.

$$|A_1 \cup A_2 ... \cup A_m| = \begin{cases} 1+O\left(e^{-2k/\sqrt{m}}\right) \sum_{|s| \le k} \alpha_{|s|}^{k,m} \left|\cap_{i \in s} A_i\right| & \text{if } k \ge \Omega\left(\sqrt{m}\right), \\ \left(O\left(\frac{m}{k^2}\right)\right) \sum_{|s| \le k} \alpha_{|s|}^{k,m} \left|\cap_{i \in s} A_i\right| & \text{if } k < O\left(\sqrt{m}\right) \end{cases} \quad (5)$$

The Equation 5 has been derived from the principle of approximate inclusion-exclusion which is used for calculating the approximate count. Here k denotes the base information size, m denotes the candidate item sets length and S denotes the subsets.

In Equation 5, $\sum_{|s| \le k} \left|\cap_{i \in s} A_i\right|$ denotes the summation of counts for all m-subsets with length m and $\alpha_{|s|}^{k,m}$ represents the coefficients of the linearly transformed Chebyshev polynomial which refers to the variety of items including first or most significant item set whose values is derived as follows.

$$\alpha_j^{k,m} = \sum_{i=1}^{k} t_i^{k,m} . a_{ij} \quad (6)$$

In Equation 6, i,j,k,m represents the item set, length of an item set, base information size, target item set length respectively, where m=k+1. a_{ij} represents the i th item set of j th item set and t represents the constant value with respect to i,k,m.

$$t_i^{k,m} = 1 + (-1)^i \times 2 / \left(\left(\frac{-(m+1)}{m-1} + \sqrt{\left(\frac{-(m+1)}{m-1}\right)^2 - 1} \right)^k + \left(\frac{-(m+1)}{m-1} - \sqrt{\left(\frac{-(m+1)}{m-1}\right)^2 - 1} \right)^k \right) \quad (7)$$

$$\alpha_{ij} = \begin{cases} (-1)^{i+j} C_i^j & \text{if } j \ge i \\ 0 & \text{if } j < i \end{cases} \quad (8)$$

The Equation 8 is used to calculate i th item set of j th item set. C represents the combination in combinatorial mathematics and C_i^j represents the combination of i item set from j item set.

$$|A_1 \cup |A_2 \cup ... \cup A_m = \sum_{\backslash s \backslash \le k} \alpha_{|s|}^{k,m} \left| \bigcap_{i \in s} A_i \right| \quad (9)$$

The Equation 9 is used for calculating the approximate count of m item set with the help of m-union item set value which is obtained from Equation 4.1.Here, down closure property is used to prune the item sets.

4.1.4 Overload Handling Mechanism

The algorithm has higher performance on processing incoming stream elements than ε-deficient mining or exact stream mining methods, since it keeps base information and carries out the mining task. It calculates a count approximation for candidate item sets based on the kept information. The base information consists of k orders of item sets out on all orders of item sets. So, the performance of the algorithm is better than ε-deficient mining or exact stream mining methods. As a result, the algorithm easily handles the data overload .The processing rate and processing time of an algorithm used to measure the performance of the algorithm. The proposed algorithm uses processing acceleration mechanism and adaptive sliding window model to raise the data processing rate of the mining algorithm. Approximate Inclusion–Exclusion technique is used to implement processing acceleration mechanism.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

When the data transmission rate of a data stream D becomes higher than the data processing rate of a mining algorithm A, the mining system receives stream elements within a time-unit by the amount w beyond its processing capability of z elements. Assume that the average length of stream elements in a batch is n items per transaction. To solve this problem the proposed algorithm proposes priority based data load shedding techniques which prunes the received data. The discarded data may be an attribute or a transaction. Dropping the unprocessed data without any care may bring about effects on the performance of mining because the discarded data may have some useful information. It proposes two policies i) Frequency oriented policy ii) Accuracy oriented policy.

4.1.4.1. Frequency Oriented Policy

This policy prunes the unprocessed data in terms of attribute. If the amount of attributes in a data source is decreased, the possible number of pattern combinations will be correspondingly diminished. The priority function utilizes down closure property of an item set (i.e) the frequency count in the previous batch to decide the priority value of each attribute. Since an attribute with low frequency has less influence on the frequent pattern generation than an attribute with high frequency, discard the attributes which are having low frequencies. An array is used to record the counts and priority values of the attributes, which is called as priority table. In these transactions, those attributes having the lowest priority values are eliminated from the transactions.

4.1.4.2. Accuracy Oriented Policy

The accuracy oriented policy aims at preserving the accuracy of mining after load shedding. The load shedding mechanism with this policy trims the unprocessed data by deleting some transactions inside it. This policy use counts-bounding technique proposed by Jea and Li (2009), to approximate the count of an item set X. A transaction containing more 2-item sets which are frequent is considered to be more important than a transaction containing less frequent 2-item sets. The transaction having lowest count will be given less priority and is discarded. Because the function of frequent 2-item sets is to limit the count ranges of 1-item sets, a transaction containing more 2-item sets which are frequent is considered to be more important than a transaction containing less frequent 2-item sets. This policy requires some feedback information from the previous batch. If a transaction contains less potential frequent 2-item sets, it has lower priority-value and will be dropped with a higher possibility. It is a time consuming problem. So the algorithm use group count technique to quickly calculate the maximum possible number of potential frequent 2-item sets (as subsets) in a transaction.

Algorithm:

Input: A transactional data stream (D), the minimum-support value (ms), initial sliding-window size (sw)

Output: A set of frequent item sets F

```
Method:
Build an empty trie P;
while (data of D is still streaming )
 do begin
     Clear the contents in F;
      while (there is no request from the user )
     do begin
         Receive from D a batch B of transactions;
         Calculate n the average length of transactions in B;
         Compute the window size by using Eq.(3.3) and
          update window size
         Set the base-information size k for a value
          satisfying k ≥√n
              for (i = 1; i ≥ k; i++)
               do
                  Extract all i-itemsets whose supports in B
                  satisfy the base threshold ˇ;
                   Update entries of the found i-itemsets in P;
                 end for
           update the base information
   end while
for (i = 1; i ≥ k; i++) do
     Find all large i-itemsets in P and insert them as $F_i$ into
F;
end for
for (m = k + 1; $F_{m-1} \neq$ NULL; m++)
do
     foreach candidate m-itemsets X
     do
     Calculate the counts of X by Eq. (4.1) based on the
base information;
           if  X's approximate count _ sw×ms then
     Insert X as a member of  $F_m$ into F;
     end if
    end foreach
end for
Return F as the mining outcome;
end while
```

V. EXPERIMENTAL ANALYSIS

This section shows empirical evidence about the proposed method. The experiments were conducted to evaluate the performance of the algorithm as well as the overload-handling mechanisms. All experiments were carried out on a platform of personal computer with an Intel 2.80 GHz dual-core processor with 2 GB of available memory space in WindowsXP operating system. The programs of the mining algorithm and the overload-management mechanisms are implemented using NETBEANS. The testing data include both synthetic and real-life datasets. Synthetic datasets are created by using IBM's synthetic-data generator and each dataset is generated with the parameters $T_s.I_t.D_u.A_v$, where s, t, u, and v respectively represent the average length of transactions, the average length of potentially frequent item sets, the amount of transactions, and the amount of attributes in the dataset. Every adopted synthetic dataset consists of 700 thousand of transactions. The real-life dataset used in the experimental evaluation is BMS-POS, which can be acquired from the well-known website namely FIMI. This dataset contains a several-year quantity of point-of-sale data from a large electronics retailer. These datasets are used to simulate transactional data streams.

The sliding window in the experiments is set for the size of 700k of transactions. Initially it is divided into 70 equal-sized segments. In next iteration the window size is calculated automatically based on the size of the stream. The base threshold ˇis set to be 5 occurrences in a batch. The algorithm is evaluated based on mining performance and mining accuracy, where performance is measured in terms of run-time and accuracy is measured in terms of precision, recall and F_β measure is used to evaluate the accuracy of the algorithm, measures the effectiveness of retrieval with respect to a user β times as importance to recall as precision. It is based onVan- Rijsbergen's effectiveness measure

$$E = 1 - \left(\tfrac{\alpha}{P} + \tfrac{1-\alpha}{R}\right)^{-1} \qquad (10)$$

$$F_\beta = 1 - E \qquad (11)$$

Where $\alpha = \frac{1}{1+\beta^2}$, β is user defined value (i.e.) at what percentage the user give importance to recall. Implementing Lossy Counting (LC) algorithm under self adaptive sliding window model using NETBEANS for comparison. The minimum support (ms) in the experiment is in the values ranging from 0.1% to 1.0%; the parameter ε of Lossy Counting is set to be 0.1*ms, which is suggested by the authors. Given a test dataset, mining algorithms are run on the dataset several times with different values of ms and find the optimal value. First generate a T15.I6 dataset. Since the average length n of transactions in this dataset is 15, it can be calculated that $\sqrt{n}$ =4.The default size k of base information to be 5.

5.1 Experiment 1: Frequent Data Stream Mining without Data Load Shedding Techniques

Fig. 3 and Fig. 4 show the mining performance of the proposed algorithm and Lossy counting algorithm. The run-time of the proposed algorithm has less impact with different values of ms. But, Lossy Counting runs as fast as the algorithm at larger values of ms but its run-time gets longer as smaller ms are given. This fact is due to the number of frequent item set is high when ms has low value and the value of ε dependent on ms. The processing acceleration mechanism useful for reducing the processing time of the proposed algorithm.

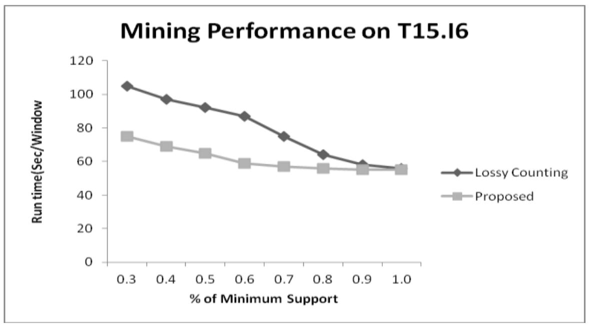

Fig. 3 Mining Performance on T15.16

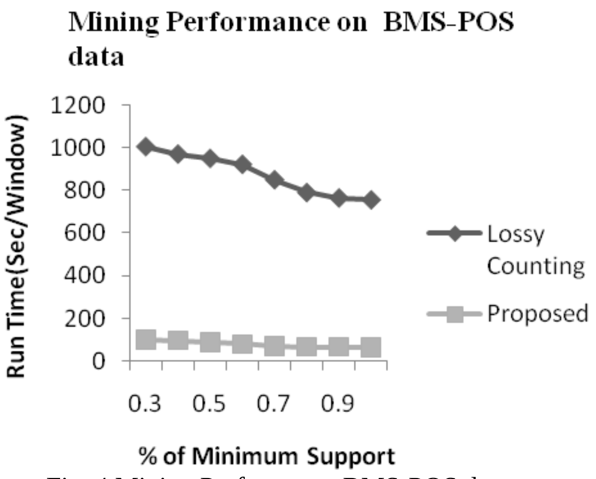

Fig. 4 Mining Performance BMS-POS data set

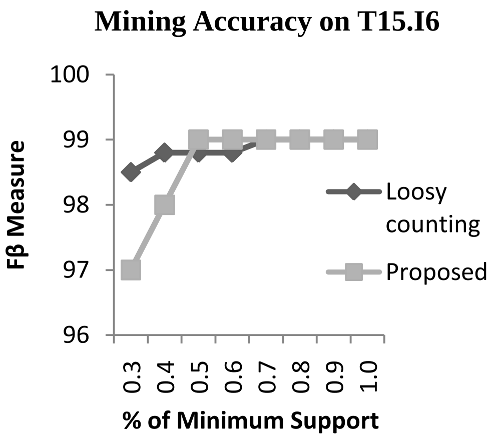

Fig. 5 Mining accuracy on T15.16

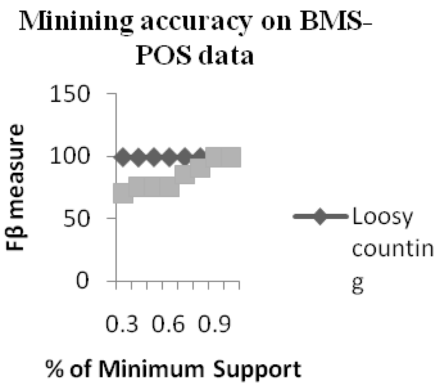

Fig. 6 Mining Accuracy on BMS-POS

Fig. 5 and Fig. 6 illustrate the mining accuracy of the algorithms on T15.16 and BMS-POS data set. From the results observed that the mining accuracy of the proposed algorithm is slightly lower than the LC algorithm at low ms values. The proposed algorithm gives 75% importance to recall than precision. But, its average score of F_β-measure is greater than 94%. It shows that generate more accurate result than LC method. From the result, the processing acceleration mechanism (i.e.) self adaptive sliding window model and dynamic k size lend a hand to improve the throughput of the algorithm.

5.2 Experiment II: Algorithm with Data Load Shedding Techniques

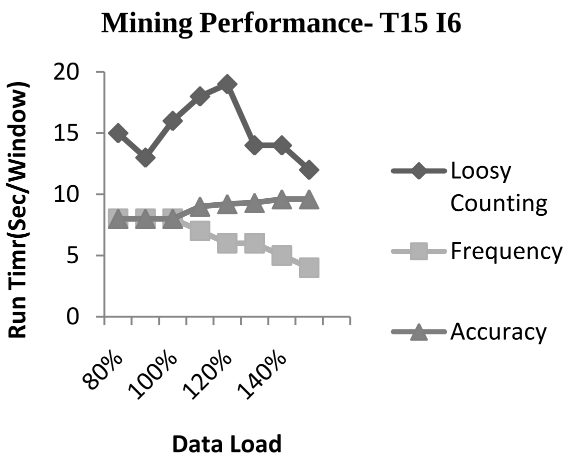

Fig. 7 Mining performance of proposed alogithm with load shedding techniques on T15.16

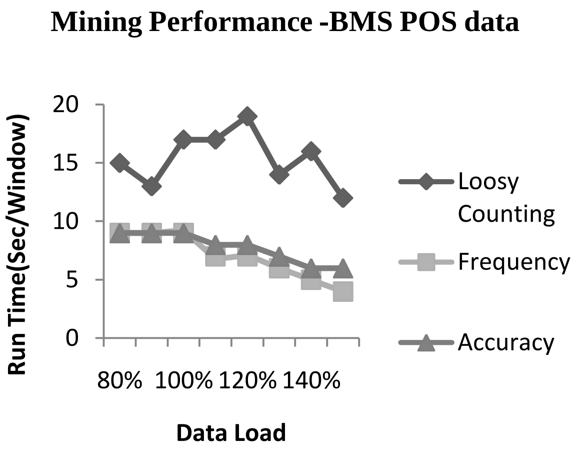

Fig. 8 Mining performance of proposed alogithm with load shedding techniques on BMS-POS data

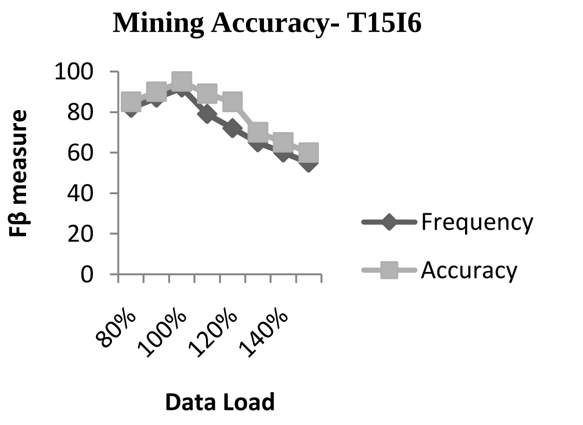

Fig. 9 Mining accuracy of proposed alogithm with load shedding techniques on T15.16

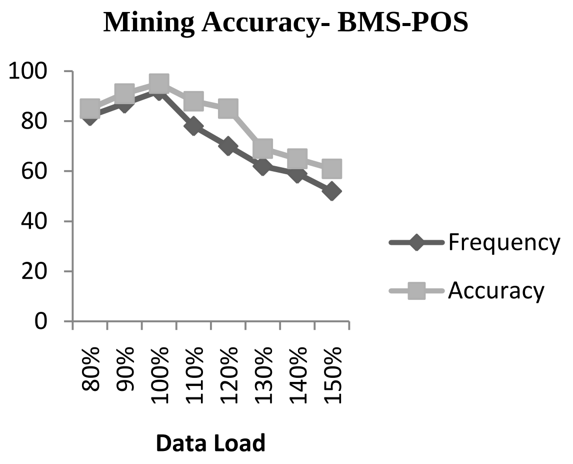

Fig 10 Mining accuracy of proposed alogithm with load shedding techniques on BMS-POS data

In this experiment, the performance of the algorithm is evaluated with the load shedding mechanism during data overloading situations. The two proposed policies for the mechanism are separately tested and then compared together. The performance is tested on different data-load degrees, ranges from 80% (i.e., normal) to 150% (i.e., highly overloaded) with 10% intervals.

Lossy Counting requires more run time than the proposed method across different data loads. Since it has no data overload shedding function. So, it needs to process and maintain larger amount of item sets. When the data load shedding mechanism is disabled, the proposed algorithm needs more run time. If the data load shedding mechanism is enabled, then the run time of the proposed algorithm is reduced based on the policy being adopted. The frequency oriented policy reduces the number infrequent item set based on down closure property. So the number of item set combination is reduced in the trimmed data set. The accuracy-oriented policy trims the unprocessed data by deleting some transactions inside it by using counts bounding technique to approximate the count of an item set X. This policy requires some feedback information from the previous batch. It consumes more run time compared with frequency oriented policy. The mining accuracy of this policy is preserved by counts bounding technique. The choice of data load shedding policy is fully dependent on the application needs. Since the different applications have different priorities of quality and performance requirements. From the analysis I concluded that, the accuracy-oriented policy is most suitable where the application give more importance to the quality of mining outcome. The frequency oriented policy is preferred where the hardware resources are limited.

VI. CONCLUSION

In real-life, the data transmission rate of data streams is usually varies with time. The mining algorithm leads to a serious issue of data overload. The work proposes a feasible solution to frequent-pattern discovery in dynamic data streams which are prone to data overload. It proposes two dedicated mechanisms for overload management. It extracts the item set from the received data stream and stores it in the base information. The frequent mining task uses approximate count technique to find the frequency of larger item set. To prevent data overload and increase data processing rate of data mining task, it implements self adaptive sliding window model as processing acceleration mechanisms. To increase throughput and handle data overload, it proposes frequency oriented policy and accuracy oriented policy. The results showed that the data processing rate is good compared with the existing algorithm. The accuracy-oriented policy is most suitable where the application gives more importance to the quality of mining outcome. The frequency oriented policy is preferred where the processor power is low.

REFERENCES

[1]. R.Agrawal., R.Srikant, "Fast algorithms for mining association rules", *Proceedings of the 20th international conference on VLDB*, 1994,pp. 487–499.

[2]. M.Charikar., K.Chen, M.Farach-Colton," Finding frequent items in data streams", *Theoretical Computer Science*, Vol. 312,No.1, 2004, pp..3–15.

[3]. J.H Chang, W.S.Lee, "A sliding window method for finding recently frequent item sets over online data streams", *Journal of Information Science and Engineering* , Vol.20, No.4, 2004, pp.753–762.

[4]. Chao-Wei Li., Kuen-Fang Jea., Ru-Ping Lin., Ssu-Fan Yen., Chih-Wei Hsu., "Mining frequent patterns from dynamic data streams with data load management", *Journal of Systems and Software*, Vol.85, No. 6, 2012 pp. 1346-1362.

[5]. Y.Chi, P.S. Yu, H.Wang, R.R.Muntz, "Loadstar: a load shedding scheme for classifying data streams", *Proceedings of the 5th SIAM Conference on Data Mining*, 2005,pp. 346–357

[6]. Chung Laung Liu., "Introduction to Combinatorial Mathematics", *McGraw-Hill, New York,*1968.

[7]. K.F. Jea,C.W. Li, C.W. Hsu,R.P. Lin,S.F. Yen, " A load controllable mining system for frequent-pattern discovery in dynamic data streams", *Proceedings of the 9th Conference on Machine Learning and Cybernetics,* 2010 pp. 2466–2471.

[8]. N. Jiang,L. Gruenwald,"Research issues in data stream association rule mining", *SIGMOD Record*, Vol.35, No.1, 2009, pp. 14–19.

[9]. K.F Jea., C.Wei Li., "Discovering frequent item sets over transactional data streams through an efficient and stable approximate approach", *Expert Systems with Applications*, Vol. 36, No.10, 2009, pp. 12323–12331.

[10]. K.F Jea., M.Y. Chang., K.C.Lin., "An efficient and flexible algorithm for online mining of large item sets", *Information Processing Letters*, 2004, pp. 311–316.

[11]. C.K.S. Leung, Q.I. Khan, "DSTree: a tree structure for the mining of frequent sets from data streams", *Proceedings of the 6th Conference on Data Mining*,2006, pp. 928–932.

[12]. C.K.S. Leung, B. Hao," Mining of frequent item sets from streams of uncertain data", *Proceedings IEEE ICDE* ,2009,pp. 1663–1670

[13]. Manku Garofalakis., Motwani., "Approximate frequency counts over data streams", *Proceedings of the 28th Conference on Very Large Data Bases,* 2002, pp. 346–357.

[14]. Nathan Linial., Noam Nisan., "Approximate Inclusion Exclusion". *Combinatoria*, Vol. 10, No.4, 1990, pp. 349-365.

[15]. Sudipto Guha, Nick Koudas, Kyuseok Shim,"Data Streams and Histograms", *ACM Symposium on Theory of Computing*, ,2001

[16]. S.K. Tanbeer, C.F Ahmed, B.S. Jeong, Y.K. Lee, "Sliding window-based frequent pattern mining over data streams", Information Sciences , Vo.179,No.22, 2009,pp. 3843–3865.

[17]. Y. Zhu, D. Shasha, "StatStream: statistical monitoring of thousands of data streams in real time", Proceedings of the 28th Conference on Very Large Data Bases, 2002, pp. 358–369.