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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

Association Rule Generation in Streams

Manisha Thool[1], Prof. Preeti Voditel[2]

Student of M-Tech, Computer Science & engineering, Ramdeobaba College of Engineering & Management, Nagpur, India[1]

HOD, MCA, Ramdeobaba College of Engineering & Management, Nagpur, India[2]

Abstract: Many applications involve the generation and analysis of a new kind of data, called stream data, where data flow in and out of an observation platform or window dynamically. Such data streams have the unique features such as huge or possibly infinite volume, dynamically changing, flowing in or out in a fixed order, allowing only one or a small number of scans. An important problem in data stream mining is that of finding frequent items in the stream. This problem finds application across several domains such as financial systems, web traffic monitoring, internet advertising, retail and e-business. This raises new issues that need to be considered when developing association rule mining technique for stream data.

In this paper, we propose an integrated online streaming algorithm for solving both problems of finding the top-k elements, and finding frequent elements in a data stream. Our Space-Saving algorithm reports both frequent and top-k elements with tight guarantees on errors. We also develop the notion of association rules in streams of elements. The Streaming-Rules algorithm is integrated with Space-Saving algorithm to report 1-1 association rules with tight guarantees on errors, using minimal space, and limited processing per element and we also implement the Apriori algorithm for static datasets and generated association rules and implement Streaming-Rules algorithm for pair, triplet association rules. We compare the top- rules of static datasets with output of stream datasets and find percentage of error.

Keywords: Association rule mining, frequent itemsets, Apriori algorithm, streaming-rules algorithm

I. INTRODUCTION

Data mining has attracted a great deal of attention in the information industry and in society as a whole in recent years, due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. Data mining means *extracting useful information or knowledge from large amounts of data*, has become the key technique to analyse and understand data. Typical mining include association mining [3], classification [4], and clustering [5]. These technique help find interesting pattern, regularities, and anomalies in the data. However, traditional data mining techniques cannot directly apply to data streams. This is because most of them require scans of data to extract the information which is unrealistic for stream data. The characteristics of the data stream can change over time and also need to consider the problem of resource allocation in mining data streams due to the large volume and the high speed of streaming data. [1]Many applications involve the generation and analysis of a new kind of data, called stream data, where data flow in and out of an observation platform or window dynamically. Such data streams have the unique features such as huge or possibly infinite volume, dynamically changing, flowing in or out in a fixed order, allowing only one or a small number of scans. Applications in which data is generated at very high rates in the form of transient *data stream.* An important problem in data stream mining is that of finding frequent items in the stream. This problem finds application across several domains such as financial systems, web traffic monitoring, internet advertising, retail and e-business [2].

There are generally two techniques: counter-based technique, and sketch-based technique. Counter-based algorithm maintains counters for and monitors a fixed number of elements of the stream. If an item arrives in the stream that is monitored, the associated counter is incremented; else the algorithm decides whether to discard the item or reassign an existing counter to this item. They maintain a summary of the items .The summary consists of a small subset of the items with associated counters approximating the frequency of the item in the stream. The counter-based algorithms include Sticky Sampling and Frequent (Freq), Lossy Counting (LC), and Space-Saving (SS).The Sketch-based technique is to maintain a sketch of the data stream using hashing, to map items to a reduced set of counters. Sketch-based techniques maintain approximate frequency counts of all elements in the stream. In Sketch-based techniques the algorithms include CountSketch (CCFC), GroupTest (CGT), and CountMin-Sketch (CM).

In this paper, we propose an integrated online streaming algorithm for solving both problems of finding the top-k

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

elements, and finding frequent elements in a data stream. Our Space-Saving algorithm reports both frequent and top-k elements with tight guarantees on errors. For general data distribution, Space-Saving answers top-k queries by returning k elements with roughly the highest frequencies in the stream and it use limited space for calculating frequent elements. In this paper, we develop the notion of association rules in streams of elements. The Streaming-Rules algorithm is developed to report association rules with tight guarantees on errors, using minimal space, and limited processing per element and then we compare the top-k rules of static datasets with output of stream datasets.

In rest of the paper is organized as follows. Section II highlights the related work. In Section III, we introduce the Apriori algorithm and Association Rule, and in Section IV, we introduce Space-Saving algorithm, and its associated data structure. The building blocks of Streaming algorithm are explained in Section V.

II. BACKGROUND AND RELATED WORK

In this section we provide the background information about static database and dynamic database, general issues in Data Stream Association Rule mining, techniques and why we used technique.

A. Database

Database can be static and dynamic.

Static Database

Static databases are those databases that do not change with time while dynamic databases, new transactions append a time advance. This may introduce new frequent itemsets and some existing frequent itemsets may become invalid.

Dynamic Database

Due to the continuous, unbounded, and high speed characteristic of dynamic data, there is huge amount of data and there is not enough time to rescan the whole database or perform a rescan as in traditional data mining algorithms whenever a new update occurs. Furthermore, there is not enough space to store all the data for its processing. [1].

B. General issues in Data Stream Association Rule mining

a) Two Sub-problems

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

b) Memory Management

Due to continuous, unbounded and high speed characteristics of data streams, there is huge amount of data in both offline and online data streams and thus there is not enough space to store all the frequent itemsets and their counts when a huge amount of data comes continuously. According to Karp in 2003, the most frequent items and their counts are stored in the main memory. This technique stored the most important information. There is not enough time to rescan the whole database or perform a multi-scan as in traditional data mining algorithm. Therefore a one scan of data and compact memory usage of the association rule mining technique are necessary. [17]

c) Handling the continuous flow of data

Due to continuous, high speed, and changing data distribution characteristics, the analysis results of data streams often keep changing as well. Therefore, mining of data streams should be an incremental process to keep up with the highly update rate, i.e. new iterations of mining results are built based on old mining results so that the results will not have to be recalculated each time a user's request is received.

d) Resource Aware

Owing to the unlimited amount of data stream and limited system resource, such as memory space and CPU power, a mining mechanism that adapts itself to available resource is needed; otherwise, the accuracy of the mining results will be decreased.

e) Real-time response

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

f) Unbounded memory requirements

Data are un-bounded but storage that can be used to discover or maintain the frequent item sets is limited. In comparison, storage that can be used to discover or maintain the frequent itemsets is limited.

g) Visualization of data mining results

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

Visualization of traditional data mining results on a desktop is still a research issue. Visualization in small screens of a PDA for example is a real challenge. Imagine a businessman and data are being streamed and analysed on his PDA.

C. Technique: Counter-based algorithms

a) Freq

According to [10, 9] the Frequent algorithm keeps count of k=$1/\emptyset$ number of items. This is based on the observation that there can be at the most $1/\emptyset$ items having frequency more than $\emptyset$N.

Freq keeps count of each incoming item by assigning a unique counter for each item, until all the available counters are occupied. The algorithm then decrement all counters by 1 until one of the counters becomes zero. It then uses that counter for newest item. This step deletes all the non-frequent item counters.

b) LC

The Lossy Counting algorithm was proposed by Manku and Motwani in 2002 [7] in addition to a randomized sampling-based algorithm and technique for extracting from frequent items to frequent itemsets. The algorithm maintains a data structure D, which is a set of entries of the form (e, f, Δ), where e is an element in the stream, f is an integer representing the estimated frequency and Δ is the maximum possible error in f. LC conceptually divides the incoming stream into buckets of width w=$1/\varepsilon$ transactions each. If an item arrives that already exists in D, the corresponding f is incrementing, and else a new entry is created. D is pruned by deleting some of the entries at the bucket boundaries.

The space requirement is O ($\frac{1}{\varepsilon} log\varepsilon N$) and time cost O (1).

c) Space-Saving Algorithm

According to [11] the deterministic Space-Saving algorithm uses a data structure called Stream-Summary. For each corresponding monitor the frequent items the Stream-Summary data structure consist of a linked list of a fixed number of counters. All counters with the same count are associated with a bucket which stores the count. Buckets are created and destroyed as new items arrive. They are stored as an always sorted doubly linked list. Each counters also stores the estimated error in the frequency count of the corresponding item, which is used later to provide guarantees about the accuracy of the frequency estimate returned by and error returned by the algorithm.

The space requirement is O ($\frac{1}{\varepsilon}$) and the counts of all stored items solve frequency estimation problem with error n/k.

D. Technique :Sketch-based algorithm

a) CGT

According to [10] the Combinational Group Testing algorithm is based on a combination of group testing and error correcting codes. Each item is assigned to groups using a family of hash functions. Within each group there is a group counter which indicates how many items are present in the group and a set of log M counters with M being the largest item in the dataset. The group counters and the counters which correspond to the bits 1 in the binary representation of the item are updated accordingly.

The space complexity is O ($\frac{1}{\emptyset}\log\frac{1}{\emptyset\delta}\log M$).

b) CountSketch

According to [6] CountSketch is an array of t hash tables each containing b buckets. There are two sets of hash functions are used one (h_1……..h_t) hashes items to buckets, and second set is (s_1……s_t) hashes items to the set {+1, -1}. Randomness of O ($t \log M$) required for implementation of these independent hash function. When an item arrives, the t buckets corresponding to that item are identified using first set, and in second set updated by adding +1 or -1.

Space complexity is O ($\frac{1}{\varepsilon^2}\log\frac{N}{\delta}$) and time is O ($\frac{N}{\delta}$).

c) CountMin Sketch

CountMin Sketch proposed by Cormode and Muthukrishnan [13] described similar to CountSketch. The algorithm maintains an array of $d \times w$ counters. When an item i arrives, one counter in each row is incremented, the counter is determined by the hash functions. The estimated frequency for any item is the minimum of the values of its associated counters. For each new item its estimated frequency is calculated, and if it is greater than the required threshold, it is added to a heap. At the end, all items whose estimated count is still above the threshold are output.

The Space complexity is O ($\frac{1}{\varepsilon}\log\frac{1}{\delta}$).

E. Why we use counter-based algorithm

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

We used Counter-based algorithm because we are interested to solve the only the frequent elements problem whereas Sketch-based algorithm act as general data stream summaries, and can be used for other types of approximate statistical analysis of the data stream ,apart from being used to find the frequent items.

Thus our application was strictly limited to discovering frequent items, counter-based algorithm would be preferable. With the observation [16] sketch-based algorithm require more space than counter-based algorithm.

We compare with other algorithm Space-Saving algorithm required less space i.e., $O(\frac{1}{\varepsilon})$ so we implemented Space-Saving Counter-based algorithm which only solve frequent item problem with minimizing space.

III. APRIORI ALGORITHM

Apriori algorithm proposed by R. Agrwal and R. Srikant in 1994 [3] for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties. Frequent itemsets generation and the creation of strong association rule from the frequent itemsets pattern are two basic steps in association rule mining.

According to [18] it first scans the database D and calculates the support of each single item in every record I in D, and denotes it as C_I. Out of the itemsets in C_I, the algorithm computes the set L_I containing the frequent 1-itemsets. In the k^{th} scan of the database, it generates all the new itemset candidates using the set L_{k-1} of frequent (k-1) itemsets discovered in the previous scanning and denotes it as C_k. And the itemsets whose support is greater than the minimum support threshold are kept L_k.

```
L1 = {large 1-itemsets}
for (k=2; Lk-1≠∅; k++) do begin
        Ck = apriori-gen(Lk-1);  // New candidates
        for all transactions t ∈ D do begin
                C't = subset (Ck, t) // Candidates contained in t
                For all candidates c ∈ Ct do c.count++
        end
        Lk = {c ∈ Ct | c.count ≥ minsup}
end
Return ∪k Lk
```

Fig. 1 Pseudo code of Apriori Algorithm

Generating Association rules from Frequent Itemsets

According to [19] the following two steps are required to augment the association rule generation.

i) For every frequent itemset "I", all non-empty subsets of "I" is required to be generated.

ii) For all non-empty subsets of I, if support_count (I) / support_count(s) >= min_conf (min_conf=minimum confidence threshold) then output the rule as "s → (l-s)".

According to [20] Apriori Algorithm is the algorithm to extract association rules from datasets. Apriori Algorithm is not an efficient algorithm as it in a time consuming algorithm in case of large datasets. With the time a number of changes proposed in Apriori to enhance the performance in term of time and number of database passes.

A. Associations Rules

In data mining, with the increasing amount of data stored in real application system, the discovery of association rule attracts more and more attention. Mining for association rules can help in business, and decision making. [3]

Association rule techniques are used for data mining if the goal is to detect relationship or association between specific values of categorical variables in large data sets. There may be thousands or millions of records that have to be read and to extract the rules for, but in the past user would repeat the whole procedure, which is time –consuming in addition to its lack of efficiency for new data, or there is a need to modify or delete some or all the existing set of data during the process of data mining [15]. Mining association rules is particularly useful for discovering relationship among items from large databases [13]. A standard association rule is a rule of the form X → Y which says that if X is true of an instance in a database, so is Y true of the same instance, with a certain level of significance as measured by two indicators, support and confidence. The goal of standard association rule mining is to output all rules whose support and confidence are respectively above some given support and coverage thresholds. These rules encapsulate the relational associations between selected attributes in the database, for instance, computer →antivirus software: 0.02 support, 0.70 coverage denotes that in the database, 70% of the people who buy computer also buy antivirus software, and these buyers constitute 2% of the database. The mining process of association rules can be divided into two steps.

1. Frequent Itemsets Generation: generate all sets of items that have support greater than a certain threshold, called minsupport.

2. Association Rule Generation: from frequent itemsets, generate all association rule that have

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

confidence greater than a certain threshold called minconfidence [14].

IV. Space-Saving algorithm

In this we briefly describe the *Space-Saving* algorithm. The algorithm proposed in [5] our counter-based *Space-Saving* algorithm and its associated *Stream-Summary* data structure.
The underlying idea is to maintain partial information of interest i.e.., to keep counters for m elements only. Each counter, at any time, is assigned a specific element to monitor. The counter is updated in a way that accurately estimates the frequencies of the significant elements. A lightweight data structure, stream-Summary is utilized. To keep the monitored elements $e_1,e_2,\ldots e_i\ldots..e_m$, sorted by their estimated frequencies. Therefore, if any monitored elements *count* (e_i), receives a hit, then its counter *count* (e_i) will be incremented, and counter will be moved its right position in the list. Among all monitored elements $\mathbf{e_1}$ is the element with highest estimated frequency and e_m is the element with the lowest estimated frequency .If an element is not monitored; its estimated frequency is *0.*
The algorithm is straightforward. The algorithm is sketched in Fig. 2.

```
Algorithm Space-Saving (Stream-Summary (m))
Begin
 For each element, x,in the stream S{
  If x is monitored {
   Let Count (ei) be the counter of x
   Count (ei) ++;
  } else {
   //The replacement step
   Let em be the element with least hits, min
   Replace em with x
   Assign e(x) the value min;
   Count(x) ++;
  }
}// end for
End;
```

Fig. 2 The Space-Saving algorithm

If there is a counter, *Count* (e_i), assigned to the observed element, *x*, i.e.., $e_i=x$, then *Count* (e_i), is incremented. If the observed element , *x*, is not monitored, i.e., no counters is assigned to it ,give it the benefit of doubt, and replace e_m, the element that currently has the least estimated hits, *min*, with *e*. The new element, *e*, could have actually occurred between *1* and *min + 1* times. For each monitored element, e_i, keep track of its maximum possible over-estimation, $\varepsilon(e_i)$, resulting from the initialization of its counter when inserted into the list. That is, when starting to monitor *x*, set $\varepsilon(x)$ to the counter value that was evicted. When queried, the elements of the *Stream-Summary* are traversed in order of their estimated frequency, and all the elements are output, until an element is reached that does not satisfy *minsup*. The reader also referred [17] for a full description.
The basic intuition is to make use of the skewed property of the data, since usually a minority of the elements, the more frequent ones, gets the majority of the hits. Frequent elements will reside in the counter of bigger values, and will not be distorted by the ineffective hits of the infrequent elements, and thus will never be replaced out of the monitored counters.

V. THE STREAMING-RULES ALGORITHM

The algorithm proposed in [21], given a stream $q_1,q_2\ldots q_I\ldots.q_N$, and maxspan is δ. The algorithm maintains a *Stream-Summary* data structure for *m* elements. For each element e_i , of these *m* counters, the algorithm maintains a consequent *Stream-Summarye*$_i$ *data* structure of *n* elements. The j^{th} element in *Stream-Summarye*$_i$ will be denoted e_i^j , and will be monitored by counter *Count* (e_i, e_j), whose error bound will be $\varepsilon(e_i, e_j)$. Each element, q_I, in the current window has a consequent set s_I. In addition, the last observed element has an antecedent set t_I.. For each element, q_I, in the data stream, if there is a counter, *Count* (e_i), assigned to q_I ,i.e., $e_i = q_I$,increment *Count* (e_i). Otherwise, replace e_m , the element that currently has the least estimated hits, *min*, with q_I; assign *Count*(q_I) the value *min +1;* set $\varepsilon(q_I)$ to min ;re- initialize *Stream-Summary*. Delete the consequent set, $s_{I-\delta-1}$, of the expired element, $q_{I-\delta-1}$ .Assign an empty consequent s_I set to q_I . Delete the antecedent set t_{I-1} and create an empty antecedent set t_I for q_I. Scan the current window $q_{I-\delta}$ to q_{I-1}. For each scanned element q_J, the algorithm checks if q_I has been inserted into s_J, and whether q_J has been inserted into t_I. If both condition do not hold, insert q_I into s_J ;and q_J into t_I.. If $\mathbf{q_J}$ is monitored, say at $\mathbf{e_j}$, i.e., Stream-Summarye$_j$ is Stream-Summaryq$_I$, and then insert $\mathbf{q_I}$ into Stream-Summarye$_j$ as follows. If there is a counter, **Count** $(\mathbf{e_j}, \mathbf{q_I})$,assigned to $\mathbf{q_I}$ in Stream-Summarye$_j$, increment it .If **Count** $(\mathbf{e_j}, \mathbf{q_I})$does not exist ,let $\mathbf{e_j^n}$ be the element with currently the least estimated hits, $\mathbf{min_j}$ in Stream-Summarye$_j$ .Replace $\mathbf{e_j^n}$ with $\mathbf{q_I}$;set **Count** $(\mathbf{e_j}, \mathbf{q_I})$ to $\mathbf{min_j + 1}$and set $\boldsymbol{\varepsilon}$ $(\mathbf{e_j}, \mathbf{q_I})$to $\mathbf{min_j}$.
If q_I has been inserted into s_J, or q_J has been inserted into t_I, or q_J is not monitored in Stream-Summary, the algorithm skips to q_{J+1}. Streaming-Rules is sketched in Fig.4.
From algorithm Streaming-Rules, we know that the processing per element involve mainly incrementing its

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

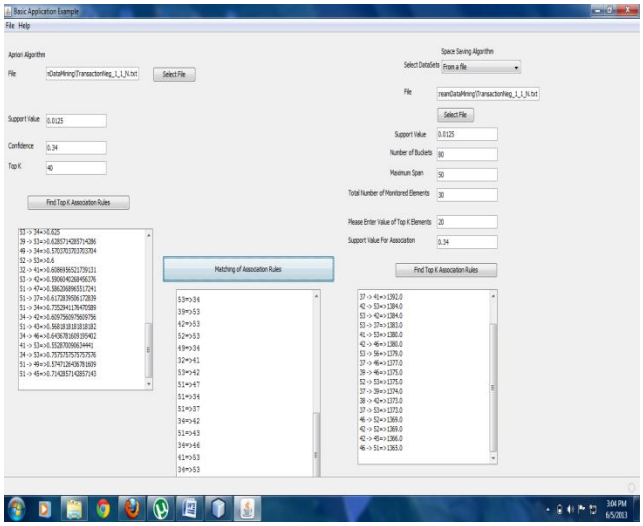
counter in the antecedent Stream-Summary and incrementing multiple counters for associating it with elements in the current window. Notice that incrementing a counter in Stream-Summary takes O (1) amortized cost if the Stream-Summary is stored in a hash table, and O (1) worst case cost if it is stored in associate if it is stored in associative memory. For finding frequent elements, Streaming-Rules inherits from Space- Saving the fact the number of counters to guarantee an error rate of ε is bounded by [$\frac{1}{\varepsilon}$]. Thus, in estimating the frequency of the rule antecedents, the error rate will be less than $\frac{1}{m}$, where m is the number of elements in Stream-Summary. We are generated one to one association by using Streaming-Rules Algorithm.

Algorithm streaming-Rules (nested Stream-Summary (**m, n**))
begin
For each element,$q_{1,}$ in the stream S {
If q_I is monitored {
Let Count (ei) be the counter of q_I
Count (ei) ++;
} else {
//The replacement step
Let e_m be the element with least hits, min
Replace e_m with q_I
Assign ε (q_I)the value min;
Count (e_m) ++;
Re-initialize Stream-Summaryq_I;
};
Delete $s_{I-\delta-1}$ of the expired element, $q_{I-\delta-1;}$
Create an empty set s_I for $q_{I;}$
Delete the set $t_{I-1;}$
Create an empty set t_I for $q_{I;}$
For each element ,q_J in the stream S, where (I-δ)≤J<I{
If q_J is monitored **AND** $q_I \in s_J$ **AND** $q_J \in t_I$ {
Insert q_I into s_J;
Insert q_J into $t_{I;}$
//The association counting step
Let q_J be monitored at e_j
If q_I is monitored in Stream-Summarye_j {
Let Count ($e_{j,}$ q_I) be the counter of
Count ($e_{j,}$ q_I) ++;
} else {
//The nested replacement step
Let e_j^n be the element with least hits,min_j
Replace e_j^n with q_I;
Assign ε ($e_{j,}$ q_I) the value min_j ;
Count ($e_{j,}$ q_I) ++;
}
}
} //end for
}//end for
End;

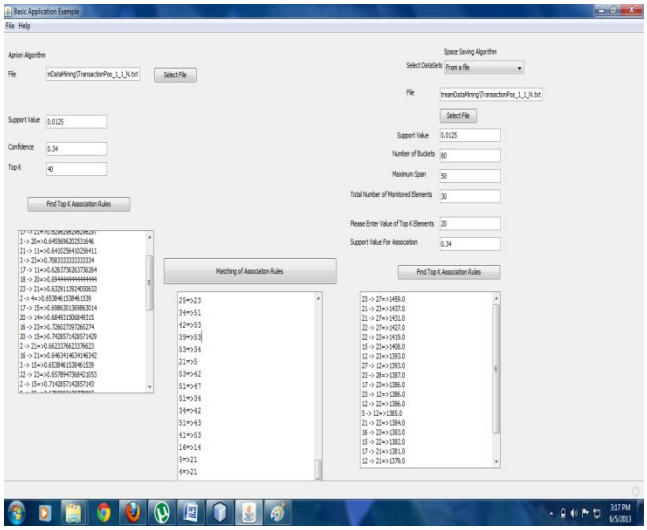
Fig. 3 The Streaming-Rules algorithm

V. EXPERIMENTAL RESULT V. EXPERIMENTAL RESULT

We are compare only 1-1 association rules and generate results. We are consider datasets TransactionNeg_1_1 TransactionPos_1_1 and TransactionPosNeg_1_1 with support 0.0125 and confidence 0.34 .The result for TransactionNeg_1_1 is ,



For TransactionPos_1_1



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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

VI. CONCLUSION

We are generated association rules by using Apriori algorithm and Streaming –Rules Algorithm with datasets TransactionNeg_1_1,TransactionPos_1_1,TransactionPosNeg_1_1. Apriori Algorithm generates Static Rules while Streaming-Rules Algorithm generates dynamic. We compare the top-k rule of both algorithms. The static rules are matched with dynamic but some of the errors are in dynamic database. We are using streaming rules i.e. dynamic in applications such as sensor network, in web block data for advertisement where memory is small and processor speed is slow.

REFERENCES

[1] Elena Ikonomovska, Suzana Loskovska, and Dejan Gjorgjevik," A survey of stream data mining",2005.
[2] Hebah H. O. Nasereddin, "Stream Data Mining", International Journal of Web Applications, , June 2011.pp 90 Volume 3,Number 2.
[3] Rakesh Agrawal, Ramakrishnan Srikant,"Fast Algorithms for Mining Association Rules in Large Databases" VLDB 1994: 487-499 Proceeding of the 20th Conference on Very Large Data Bases.
[4] Bing Liu, Wynne Hsu, Yiming Ma, "Integrating Classification and Association Rule Mining", 1998 Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining .
[5] Wang H, Yang J,Wang W,Yu P,"Clustering by pattern similarity in large data sets." Proceedings of the 2002 ACM SIGMOD international conference on Management of data Pages 394-405
[6] S. Muthukrishnan. Data streams," Algorithms and applications", 2006.
[7] G. S. Manku and R. Motwani.,"Approximate frequency counts over data streams", 2002 In Proc. Of the 28th Int'l Conference on Very Large Databases pages 346–357.
[8] Ferry Irawan Tantono, Nishad Manerikar, Themis Palpanas: "Efficiently Discovering Recent Frequent Items in Data Streams". SSDBM 2008: Pages 222-239
[9] R. M. Karp, S. Shenker, and C. H. Papadimitriou,"A simple algorithm for finding frequent elements in streams and bags". 2003 ACM Transactions on Database Systems, vol.28,pp.51-55
L. Golab, D. DeHaan, E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro,"Identifying frequent items in sliding windows over on-line
[10] packet streams", 2003 In Proceedings of the Internet Measurement Conference, pp.173-178
[11] A. Metwally, D. Agrawal, and A. E. Abbadi, "An integrated efficient solution for computing frequent and top-k elements in data streams", 2006 ACM Trans. Database Syst., vol. 31, no. 3, pp. 1095–1133.
[12] M. Charikar, K. Chen, and M. Farach-Colton," Finding frequent items in data streams", 2002 In Procedings of the International Colloquium on Automata, Languages and Programming (ICALP).
[13] G. Cormode and S. Muthukrishnan," An improved data stream summary: the count-min sketch and its applications.J. Algorithms ", 2005 55(1):58–75.
[14] Li Y. C., Yeh J. S., Chang, C. C ,"Efficient algorithms for mining share-frequent itemsets", 2005 In Proceedings of the 11th World Congress of Intl. Fuzzy Systems Association, pp. 543–539.
[15] Raymond Chi-Wing Wong, Ada Wai-Chee Fu,"Assocation Rule mining and its applications".
[16] Frequent items in streaming data: An experimental evaluation of the state-of-the-art.Nishad Manerikar, and Themis Palpanas. Data Knowl. Eng. 68(4):415-430 (2009)
[17] G. Cormode and M. Hadjieleftheriou.,"Finding frequent items in data streams" , 2008 PVLDB, 1(2):1530–1541.
[18] Nan Jiang, Le Gruenwald," Research issues in data stream association rule mining " , (March 2006) SIGMOD Rec., Vol. 35, No. 1.
[19] Vikarm Singh and Sapna Nagpal Integrating User's Domain Knowledge with Association Rule Mining in March 2010 IJCSI Vol. 7, Issue 2, No 5.
[20] Biswaranjan Nayak and Srinivas Prasad,"A Pragmatic Approach on Association Rule Mining and its Effective Utilization in Large Database" in May 2012 IJCSI, vol. 9, Issue 3, No 1.
[21] Ahmed Metawally, Divyakant Agrawal and Amr EI Abbadi,"Using Association Rules for Fraud Detection in Web Advertising Networks". In Proceedings of the 31st International Conference on Very Large Databases 2005.