Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 31 (2014) 976 – 984

Procedia Computer Science

2nd International Conference on Information Technology and Quantitative Management, ITQM 2014

Clustering Text Data Streams – A tree Based Approach with Ternary Function and Ternary Feature Vector

M.S.B.PhridviRaj[a], Chintakindi Srinivas[b], Dr.C.V.GuruRao[c]

[a,b]*Kakatiya Institute of Technology and Science, Warangal,INDIA*
[c] *Professor of CSE & Principal, S.R.Engineering College ,Warangal,INDIA*

Abstract

Data is the primary concern in data mining. Data Stream Mining is gaining a lot of practical significance with the huge online data generated from Sensors, Internet Relay Chats, Twitter, Facebook, Online Bank or ATM Transactions. The primary constraint in finding the frequent patterns in data streams is to perform only one time scan of the data with limited memory and requires less processing time. The concept of dynamically changing data is becoming a key challenge, what we call as data streams. In our present work, the algorithm is based on finding frequent patterns in the data streams using a tree based approach and to continuously cluster the text data streams being generated using a new ternary similarity measure defined.

Keywords : similarity , ternary vector ,cluster , data stream; frequent item

1. Introduction

Data mining is a knowledge discovery approach for finding hidden information and hidden patterns from currently available data. The difference between the data in the databases and a data warehouse is in a database the data is in the structured form whereas in the data warehouse the data may or may not be present in the structured format. The structure of the data may be defined to make it compatible for processing.

Handling static data is comparatively much easier than dynamically varying data. In the case of a static dataset, the entire data is available for the analysis in hand before performing processing and is generally not a time varying data.

* Corresponding author. Tel.: +0-000-000-0000 ; fax:+0-000-000-0000 .
E-mail address: prudviraj.kits@gmail.com

doi:10.1016/j.procs.2014.05.350

Dynamic data refers to high voluminous continuously varying information which is not a stand still data and hence not suitable for processing or analysing. This type of data is often called data stream. Data streams may be time series or temporal or spatio-temporal. The concept of clustering and classification is widely used and turned out as an area of typical interest for the researchers. In Section-2 of this paper we outline some of the related works of interest. In Section-3, we give the algorithm to find frequent patterns in data streams along with a case study. Finally, we conclude in Section 4.

2. Related Works

In case of data streams, the number of distinct features or items that exist would be so large, which makes even the amount of on cache memory or system memory available not suitable for storing the entire stream data. The main problem with data streams is the speed at which the data streams arrive is comparatively much faster than the rate at which the data can be stored and processed.

In the ACM KDD International conference held in 2010, the authors discuss the problem of finding the top-k most frequent terms in a data stream with flexible sliding widows [3]. The idea is to obtain only the top-k frequent items instead of considering all the frequent items. But the crucial factor or limitation that evolves here is the amount of memory that is required still for mining w.r.t to finding of top-k frequent items is still a bounding factor. The authors finally discusses that there exists however a memory efficient algorithms by making some assumptions. In [2], the authors focus, on developing a framework for classifying dynamically evolving data streams by considering the training and test streams for the dynamic classification of datasets. The objective is to develop a classification system in which a training system can adapt to quick changes of the underlying data stream.

The amount of memory available for mining stream data using one pass algorithms is very less and hence there is a chance for data loss. Also, it is not possible to mine the data online as and when it appears because of mismatch in speed and several other significant factors. In [4], the authors discuss the method of finding most frequent items by using a hash based approach. The idea is to use say ‘h’ hash functions and build the hash table by using linear congruencies. Data streams may be classified in to two types as 1. Offline data streams 2. Online data streams. In [6], Singular valued decomposition is used to find the correlation between multiple streams. The concept of SVD was particularly used to find offline data streams. Clustering text data streams is one of the topics which have evolved as an important challenge for data mining researchers. The problem of spam detection, email filtering, clustering customer behaviours, topic detection and identification, document clustering are a few of typical interest to data mining researchers.

In [7], Liu et.al, discuss on clustering text data streams. The idea is to extend the existing semantic smoothing model which works well with static data streams for clustering dynamic data streams. The authors propose two online clustering algorithms OCTSM and OCTS for clustering massive text data streams.

A tremendous amount of data is generated from the web every instant in various forms such as social networks, data from sensors, face book and twitter. The emerging data from web also called as the Text message stream that is generated from various instant message applications, Internet Relay Chat. This has become a prime topic which has become a prime topic of interest to researchers working in the area of data mining. Shen, Yang et.al [8], propose the method of detecting the threads in dynamic data streams. The paper discusses three methods of single pass clustering algorithm followed by a new approach of clustering based on linguistic features. A method of reducing the dimensionality of streaming data using a scalable supervised algorithm is proposed in [9].The limitations of PCA; LDA and MMC approaches are discussed. The authors point out the unsuitability of MMC for streaming data. A supervised incremental dimensionality reduction algorithm is proposed to meet the requirements of streaming data set. In [10] the authors show that the most cited result hoeffdings bound is invalid

3. Proposed Work

We consider the method of finding frequent items from data streams using sliding windows. Let S be the sliding window of Size n with I = $\{i_1, i_2, i_3 \ldots i_m\}$ as the set of all available items. Depending on the type of the transaction done, a transaction T_j may contain the entire item set denoted as I or only a proper subset of I as its items.

Let A and B be any two items. Then the binary vectors for A and B are denoted by Bin-Vector (A) and Bin-vector (B) respectively. As, the dimension of the sliding window is restricted to a count of n transactions, we restrict the representation of the binary vectors for item A and item B to be of the form, n-bit binary vector denoted as

Bin_Vector(A) = $A_1\, A_2\, A_3\, A_4 \ldots A_n$ and

Bin_Vector(B) = $B_1\, B_2\, B_3\, B_4 \ldots B_{n.}$ Where n is the size of sliding window.

If the item A is present in the transaction T_i then the corresponding bit of the Bin_Vector (A) is set to 1. Similarly, if the item A is not present in a given transaction T_i then the corresponding i^{th} bit of the Bin_ vector (A) is set to 0. This is shown as the first level nodes of the Frequent-Pattern-Generation-Tree called FPGT in fig.1.

Definition.1
Let Bit-1 and Bit-2 be two one bit ternary numbers. We define a function F over Bit-1 and Bit-2 as given in Table. 1

Definition.2
The ternary function F in the Table.1 takes input as two 1-bit ternary numbers and gives the output as a 1-bit value which is a 0 or 1 or Z. We extend the function F in Table.1 to compute over two ternary feature vector of item set by applying F for each corresponding bits of the feature vectors.

Table 1. Ternary Function

Bit -1	Bit-2	F(Bit-1,Bit-2)
0	0	Z
0	1	0
1	0	0
1`	1	1
0	Z	Z
Z	0	Z
Z	Z	Z
1	Z	Z
Z	1	Z

We assume the transaction item set to be static. However if the itemset is dynamic and gets added later we need to simply generate a new link from the root node.

Any node in the FPGT tree consists of four fields

1. First field represents the Item ID or name
2. Second field represents feature vector representation of the transaction item
3. Third field indicates the count of 1's in the feature vector.
4. Fourth filed stores the status of corresponding node in the tree indicating Live (L) or Dead (D).

The main problem in handling data streams is memory constraint because we are restricted to a single scan of the database. The algorithm defined below performs only one time scan of the database initially and uses the information to find frequent patterns using frequent pattern generation tree.

3.1 Algorithm

Let A be any item in a transaction, S be the sliding window, S_i is the ith sliding window and T = {T_1, T_2, T_3 ....T_n} be the transactions in current the sliding window.

Step.1: Start with the root node and generate a node for each item in the list of transactions of the sliding window. This is the first node in the frequent pattern generation tree which we call as start node or root node consisting of at most m links. Here m denotes the total number of items. The fields of the root node contain no information but are just links pointing to m items of the transaction data set. We perform the data scan of the entire database for the first time and store which item belongs to which transaction.

Step2: // Compute m-itemsets with i=2, 3, 4….m

For each item node created in the FPGT tree generated in Step1

For each of its corresponding siblings towards its right

Create a new quaternary node with four fields with the first field as item-set name, Second field as n-bit ternary feature vector ,Third field containing count of 1s in ternary feature vector, fourth field indicating status of the node as live or dead node.

// we call it as ternary feature vector because we have three values 0, 1, U and quaternary as each node has 4 fields.

```
        If (Support_value (itemset) of  E-node generated
           < user_threshold)
             Kill the corresponding  node of the tree and
              mark it as dead node.
         else
              Retain the node and mark the node as live.
               Endif
     Endfor
   Endfor
```

```
Step3:
 For each node generated in step-2 of the partial FPGT tree
    Consider only parent nodes of the node at level – (i+1).
       If ( there exists a node with itemset in level-i which
             is subset of current E-node at level i+1 and has
             same  support value )

                    Kill the node at level-i.

// this is because node at level-i is a subset and has no
   impact on its deletion.
       Else

        Retain the node for future tree generation.

Step-4:  Repeat step-3 till we get no new node is generated.

Step-5:  Display the nodes with first k-larger values which form top-k frequent items.
```

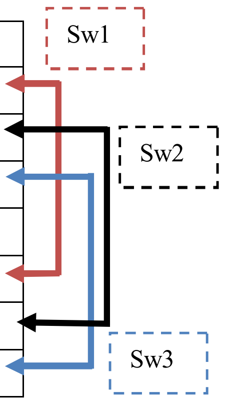
End of the algorithm

3.2 Case Study

Let the incoming transaction flow data stream is as shown below in Table.2. It may be viewed as a Data stream with seven transactions and sliding window of size=5.

Table 2. Transactions viewed as a Data stream

Transaction	Items
T_1	A B C
T_2	B C D
T_3	A B C
T_4	B C
T_5	B D
T_6	A B C D
T_7	C D



Consider A=10100. This represents that item A is present in transactions 1 and 3 only. In Binary vector, 1 indicates presence and 0 indicates absence. Let user defined threshold be 20%. This means support, S=20% = 0.2. In other words, as there are 5 transactions, each item must be present in at least one transaction. The support for each item computed is as given below

Support (A) = 2
Support (B) = 5
Support (C) = 4
Support (D) = 2

Now as all the items have the support more than 1. This means all the nodes are live as shown in Level-1 of the tree. The process starts with creating four nodes one for each item from the transactions present within the sliding window. This is shown in the FPGT tree at Level-1.

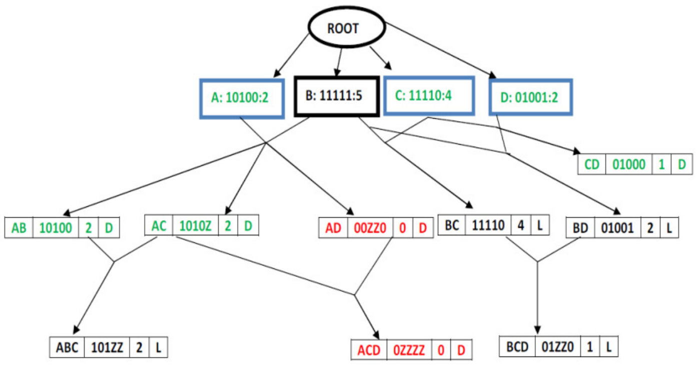


Figure 1. Creation of root node and handling SW1

Now, consider each node generated at Level-1 and its corresponding sibling node one at a time for each node at level-1. Create a new node at Level-2 with item name of new node equal to union of two item names from item name of node at level-1 and item name of its sibling. Obtain the feature vector by using the function defined in Table.1.

This gives 6 new nodes as AB, AC, AD, BC, BD and CD. Here AD is killed as its support is less than 20%. This is shown with node marked in red. The following figure is self explanatory and shows the trace of the proposed algorithm. In Figure 1, Figure 2, Figure 3 the nodes labeled black are live nodes and active. The nodes labeled green are not closed nodes or items. The nodes labeled red are killed nodes. The nodes at the first level are never killed, but just labeled green to indicate that they are not closed nodes.

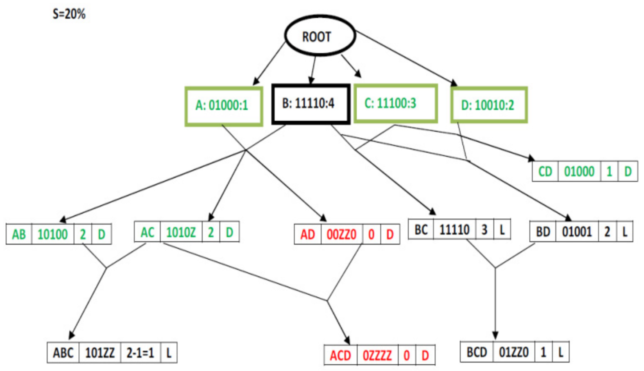


Figure 2. Deletion of nodes when SW1 slides and SW2 is active

Once we have all the frequent items (live nodes), we can find the top-k frequent patterns by displaying the first K-items from higher support values to lower support values.

If K=3 then the top-3 frequent terms are {B, BC, BD}.

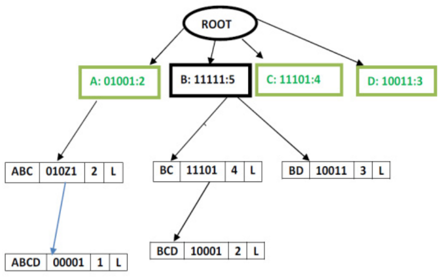


Figure 3. Final Frequent Pattern Generation Tree at sliding window2

Once we find frequent items we get the global feature vector as GFV= {B, C, D} and the transactions contain these frequent items only. Table.3 below lists Transactions and their corresponding feature vectors.

Table 3: Transactions with frequent items

Transaction	**Items**	**Feature_vector**
T_1	B C	<0110>
T_2	B C D	<0111>
T_3	B C	<0110>
T_4	B C	<0110>
T_5	B D	<0101>
T_6	B C D	<0111>
T_7	C D	<0011>

The similarity matrix formed for the above set of transactions is as shown below using the algorithm in [21] and ternary similarity measure defined in the present work. T_i

Table 4: Similarity Matrix Obtained from Table 3

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
T_1	x	2	2	2	1	2	1
T_2	x	x	2	2	1	3	1
T_3	x	x	x	2	1	2	1
T_4	x	x	x	x	1	2	1
T_5	x	x	x	x	x	2	1
T_6	x	x	x	x	x	x	2
T_7	x	x	x	x	x	x	x

Now since the element pair (2, 6) has higher similarity value, we place 2 and 6 into one cluster. Thus cluster1= {2, 6). Now choose next maximum value which is 2. Group all such pairs into one cluster i.e {1, 3} and {1, 4} into {1, 3, 4} as cluster2

Table 5: Reduced Similarity Matrix Obtained from Table 4

	T_1	T_3	T_4	T_5	T_7
T_1	x	**2**	**2**	1	1
T_3	x	**x**	**2**	1	1
T_4	x	x	x	1	1
T_5	x	x	x	x	1
T_7	x	x	x	x	x

Table 6: Reduced Similarity Matrix Obtained from Table 5

	T_5	T_7
T_5	x	1
T_7	x	x

Finally place {5,7} into one cluster and name it as cluster3.

The set of clusters so formed are

Cluster1: { T_2, T_6}

Cluster2: { T_1, T_3, T_4 }

Cluster3: { T_5, T_7}

4. Conclusion

The problem of handling streams for clustering, classification and topic detection is still a challenge and has a wide chance of exploration for data mining researchers to carry their work. In this paper, we find the frequent patterns from data streams which uses frequent pattern generation tree. A node in the tree consists of four fields. The algorithm uses the function defined to generate the ternary bit vectors. The workflow of the algorithm is traced. The algorithm lists all possible frequent patterns and can be used to find top-k frequent items. Finally, the proposed algorithm is extended to perform data stream clustering and classification by using the algorithm in [21].

References

[1] Albert Bifet, Geoff Holmes et.al. 2011. Mining Frequent Closed graphs on evolving data streams. In the Proceedings of 17th ACM SIGKDD International Conference on knowledge discovery and data mining.2011. 591-98.

[2] Charu C. Aggarwal, Jiawei Han, Philip S. Yu. 2004. On Demand Classification of Data Streams. In the proceedings of ACM KDD'04, August 2004, USA.

[3] Hoang Thanh Lam, Toon Calders. 2010. Mining Top-K Frequent Items in a Data Stream with Flexible Sliding Windows. Proceedings of in the proceedings of ACM KDD'10, July 2010, USA.

[4] Cheqing Jin et.al. 2003. Dynamically Maintaining Frequent Items over a Data Stream. In the proceedings of CIKM 2003.USA.

[5] Nan Jiang and Le Grunewald. 2006. Research Issues in Data Stream Association Rule Mining, SIGMOD Record, Vol. 35, No. 1, Mar. 2006.

[6] Sudipta Guha, D.Gunopulos, N.Kaudas.2003.Correlating synchronous and asynchronous data streams. In the proceedings of SIGKDD 2003 held from august 24th -27th, 2003, USA.

[7] Yu.Bao.Liu et.al. 2008. Clustering Text data streams. Journal of computer science and technology, volume 23, issue 1, pages 112-128, 2008.

[8] Dou Shen,Qiang Yang, Jian-Tuo-Sun, Zheng Chen.2003. Thread Detection in Dynamic Text Message Streams. In the proceedings of SIGIR from august 6th -11th, 2003, USA.

[9] Jun Yan et.al. 2006. A scalable supervised algorithm for dimensionality reduction on streaming data. Information Sciences, An International Journal, Published by Elsevier, Volume 176, 2042-65, 2006.

[10] L.Rutkowski et.al. 2013. Decision trees for mining data streams based on the McDiarmid's bound. IEEE Transactions on Knowledge and Data Engineering, Volume 25(6), 2013.

[11] Jun Yan et.al. 2006. Effective and Efficient Dimensionality Reduction for Large-Scale and Streaming Data Preprocessing. IEEE Transactions on Knowledge and Data Engineering, Volume 18(2), 2006.

[12] Graham Cormode et.al. 2003. Comparing Data Streams Using Hamming Norms (How to Zero In).IEEE Transactions on Knowledge and Data Engineering, Volume 15(3), 2003.

[13] Chen Ling, Zou Ling-Jun, Tu Li.2012. Clustering algorithm for multiple data streams based on spectral component similarity. Information Sciences, An International Journal, Published by Elsevier, Volume 183, 35-47, 2012.

[14] Panagiotis Antonellis, Christos Makris, Nikos Tsirakis. 2009. Algorithms for clustering click stream data. Information Processing Letters109, 381–385, 2009 published by Elsevier.

[15] Chang-Dong Wang, Dong Huang. 2013. A support vector based algorithm for clustering data streams. IEEE Transactions on Knowledge and Data Engineering, Volume 25, Issue 6, 2013.

[16] Shi Zhong. 2005. Efficient streaming text clustering, Neural Networks.Volume18, 2005, 790–798, published by Elsevier.

[17] Pedro Pereira Rodrigues, Joao Gama and Joao Pedro Pedroso.2008. Hierarchical Clustering of Time Series Data Streams. IEEE Transactions on Knowledge and Data Engineering, Volume 20, Issue 5, 2008.

[18] Vaneet Aggarwal, Shankar Krishnan. 2012. Achieving Approximate Soft Clustering in Data Streams", 2012. http://arxiv.org/abs/1207.6199.

[19] Haiyan Zhou, Xiaolin Bai, Jinsong Shan. 2011. A Rough-Set-based Clustering Algorithm for Multi-stream. Procedia Engineering 15 (2011) 1854-58.

[20] Mohamed Medhat Gaber.2012. Advances in Data stream mining. WIREs Data Mining Knowledge Discovery. Volume 2, 79–85, 2012. Doi: 10.1002/widm.52.

[21] Vangipuram Radhakrishna, C. Srinivas, C. V. Guru Rao. 2013. Document Clustering Using Hybrid XOR Similarity Function for Efficient Software Component Reuse. ITQM 2013: 121-128.

[22]. M.S.B.Phridviraj, C.V.GuruRao. Data Mining – Past, Present and Future – A Typical Survey on Data Streams. The 7th International Conference Interdisciplinarity in Engineering, INTER-ENG 2013, 10-11 October 2013, Petru Maior University of Tirgu Mures, Romania