Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 4, April 2017, pg.134 – 137

REAL-TIME STREAM DATA MINING to FIND FREQUENT ITEM-SET

Prathamesh S. Pimpale, Nayan V. Rakhecha, Mayur A. Saindane, Harshal S. Sawant

K.K.Wagh Institute of Engineering Education & Research,
Savitribai Phule Pune University, Nashik
pimpaleprathamesh76@gmail.com, rakhecha30@gmail.com,
mayursai24@gmail.com, hrshlswnt143@gmail.com

Abstract—Frequent sets play an essential role in many Data Mining tasks that try to find interesting patterns from databases, such as association rules. The mining of association rules is one of the most popular problems of all these. Compared to mining from a static transaction dataset, the streaming case has far more information to track and far greater complexity to manage. In-frequent items can become frequent later on and hence cannot be ignored. The storage structure needs to be dynamically adjusted to react the evolution of item-set frequencies overtime. In this project, we will use a novel data mining algorithm, called as Can-tree and G-tree, which discovers the complete frequent item- sets from real-time transactions based on sliding-windows.

The algorithm uses two data structures: Can-tree and G-tree. Can-tree compactly represents all transactions in a sliding-window by one scan, and serves as a base-tree. G-tree finally gives output as the frequent item-set. In this system, we will implement the Can-tree and G-tree algorithm to find the complete frequent item-sets with minimum time.

Keywords— Data Management system, Data Scan, Data Processing System, Data Analytics, Data storage.

INTRODUCTION

Researchers have proposed many algorithms for discovering frequent Item-sets from a given data set. The data set can be static or dynamic. Most algorithms to discover frequent Item sets focus on a static data set. However, static data mining algorithms cannot be applied to a dynamic data set, which dynamically shrinks and expands. The development of sensors and their increasing use in applications has led to a flood of data. Data streams from sensors have become the target of data mining. Hence it is necessary to find the algorithms which can efficiently find the frequent Item-sets from the real time data streams.

The real time data stream are to be processed for the frequent item sets they have. Mostly with the algorithms available it is time consuming .The idea of the project is to find frequent item sets from the real time data stream. The different algorithms used are Can-tree and G-tree. These are efficient algorithms as compared to present algorithms used and can have one scan operation so equally perform good with the real time data stream.

The sensors usually provide the real time data streams. The basic idea to perform operation on the real time streams is that the operation should be completed in one scan as the data is getting changed after every specified time. The Can-tree and G-tree algorithm combination that are being used to perform this operation have the highest efficiency to find frequent Item-set. The data provided by the sensors is sometimes important for humans, hence it is important to find all the frequent item sets not just single frequent pattern. The above algorithms gives all the frequent item-sets from the input data stream.

RELATED WORK

In particular, the Lambda Architecture pattern distinguishes the components that process recent data only in real-time (speed layer). In practice, such data items come continuously from so-called data streams. To prove effectiveness of the designed topology a sample big data analytics task was solved (the task was to build up a predictive model for programming technologies trends based on the data stream from GitHub and Twitter). The implemented topology demonstrated enough flexibility for the sample task. Therefore, it can be evolved in order to be applied for resolving more complex and valuable problems. Apache Mahout provides effective machine learning algorithms adopted to data stream mining purposes. Furthermore, we can take advantages of its framework to build custom improvements and extensions. [1]

Random decision tree model method uses Hoeffding Bounds inequality and information entropy instead of random selection to determine the split point, and it uses the threshold determined by Hoeffding Bounds inequality detect concept drift. It has better classification accuracy for data stream, and it provides a new practical approach for coal mine safety evaluation. The method not only has good classification accuracy in real data set, but also has good classification accuracy in other simulated data sets. Generally there is missing data and abnormal data in coal mine monitoring data. [2]

Data streams have several unique properties due to which stream classification is more difficult compared to traditional classification. Primarily, data streams have huge, infinite length of data, which makes it unfeasible to store on the disk and even though if the data get stored on the disk it is unfeasible to scan the data recursively in order to identify the beneath patterns .These developments would be realized over the next few years as research issues that would speed up the science discovery in physical and astronomical applications in addition to business and financial one. [3]

Optimization model for clustering categorical data streams, in the model, an objective function is proposed which simultaneously considers the clustering validity on new sliding windows and difference of cluster structures between windows. Furthermore, a validity index for the drifting-concept detection is derived from the optimization model. Therefore, optimization model for clustering categorical data streams this new method can effectively avoid ignoring the effect of the clustering validity on the detection result. [4]

A new single-pass algorithm, called DSMFI (Data Stream Mining for Frequent Item-sets), to mine all frequent Item-sets over the entire history of data streams. DSM-FI has three major features, namely single streaming data scan for counting Item-sets frequency information, extended prefix-tree-based compact pattern representation, and top-down frequent Item-set discovery scheme. The performance study shows that DSM-FI outperforms the well-known algorithm Lossy. Counting in the same streaming environment. [5]

METHODOLOGY

In this project, the methodology of problem solving used is algorithmic approach. The algorithm used are Can-tree and G-tree algorithms.

Can-tree Algorithm:-

Pre-processed data is given to the can tree data structure .The data is filtered for the garbage values and the missing values .Then the i-table and l-table are constructed which gives the batch-wise data. Can-tree organize the data in the canonical order .Base tree is constructed by the Can-tree. Further the output of the Can-tree is given as the input to the G-tree.

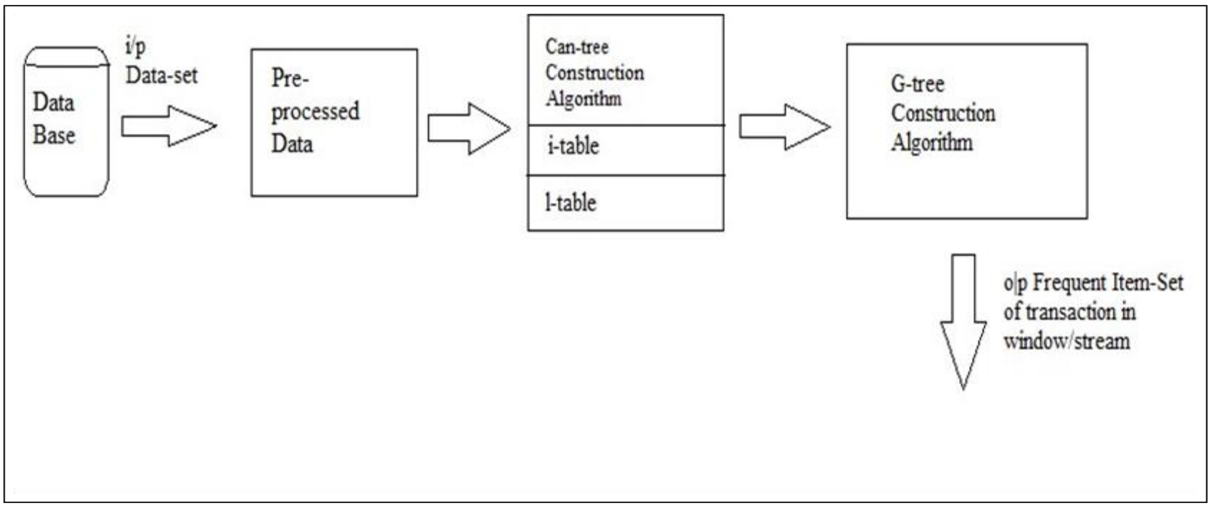
G-tree Algorithm:-

The output of the Can-tree, i-table and l-table is given as the input to the G-tree .G-tree algorithm finds out the frequent item sets in the given stream batch. The frequent item sets are given with all the sets in given batch.

```
Can-tree G-tree Algorithm:-
1) Input:
A sequences of batches

2) Method:
Can-tree-G-tree()
{
    While(true)
    {
        Get a new batch;
        If a window is full of batches,
        Delete the oldest batch from Can-tree;
        Add the new batch to Can-tree;
        For each data item D in Can-tree,
        Construct G-tree from Can-tree;
        Item-set=null;
        Call Preorder Traverse(G-tree, item-set);
    }
}
Preorder Traverse(G-tree, item-set)
{
    If (G-tree is null) return;
    New-Item=item-set + item(root(G-tree));
    Add new-Item to list of Item-sets;
    For each data item X in (I(G-tree)-{root of G-tree})
    {
        If the support of X is greater or equal to minimum support
        {
            Construct G-tree from Can-tree;
            Call Preorder Traverse (G-tree, new-Item);
        }
    }
}
```

ARCHITECTURAL DESIGN


Data Base
i/p Data-set
Pre-processed Data
Can-tree Construction Algorithm
i-table
l-table
G-tree Construction Algorithm
o|p Frequent Item-Set of transaction in window/stream

The input is given as the Data base containing the numerical Datasets. Later the pre-processing of the data is performed. The pre-processed data is given as input to the Can-tree. Can-tree stores the transaction and operate on them. i-table contains the support count of the nodes in the base tree. l-table contains the numbers of the last node of the transactions. The i-table , l-table along with the output of the Can-tree is given as input to the G-tree .G-tree then give the frequent item set/sets as the output. The data can be sent or divided into batches that can be operated as static. This technique do not have a memory overhead as we operate only on the Current data

in the batch in the given window. Only with a single scan we can obtain the frequent item-sets from the current data.
The dataset used for the primary purpose testing is the KDD dataset. With the static data testing the effects in the results are observed. Even the one scan operation is useful in this case. Having the stream data for the operation is quite difficult of operating as we have to save the data also and then operate on it. But with this setup for Can-tree and G-tree we can operate on the data with ease.
Consider any given numeric data application which produces the stream data .With the given traditional algorithms, the data has to be saved first and then operated with the mining operations. The stock market is the example in which the data is continuously changing with few moments .So it has to be operated within the given time or the value changes and the operations become useless. For such kind of problem, the proposed algorithms that are Can-tree and G-tree are very useful. They can operate on this data in a continuous way finding the frequent item-sets in it.

CONCLUSION

We proposed the Can-tree and G-tree algorithm for real-time stream data mining. When sliding-window moves (or new transactions arrive), the algorithm uses a data structure Can-tree to add the new transactions to the original database (called a base-tree), without restructuring the base-tree. Then, we used G-tree as a projection-tree to find frequent item-sets. G-tree contains only frequent items, and the algorithm traverses its nodes just once. This means that Can-tree and G-tree algorithm uses G-tree to represent a projection-DB. When representing a base-tree, Can-tree generally needs more nodes than FP-tree. However, G-tree is more effective as a projection-tree than FP-tree. This is because G-tree is already ordered by canonical order, and only one scan of Can-tree (or parent G-tree) is sufficient to construct G-tree. Through the various performance evaluations with Can-tree-Can-tree algorithm, Can-tree-FP-tree algorithm, and CPS-tree algorithm, we have verified that Can-tree-G-tree algorithm outperforms the other algorithms. Whenever recent knowledge is quickly needed, Can-tree- G-tree algorithm can be a good choice to discover the complete frequent item-sets from the data stream environment. In future study, we hope to develop more efficient data structure than Can-tree.

REFERENCES

[1] “Apache Storm Based on Topology for Real-Time Processing of Streaming data from Social Networks , IEEE First International Conference on Data Stream Mining Processing23-27 August 2016, Lviv, Ukraine
[2] `A Coal Mine Safety Evaluation Method Based on Concept Drifting Data Stream Classification', 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)
[3] " A Review on Method of Stream data classification through Tree based approach", IEEE Sponsored World Conference on Futuristic Trends in Research and Innovation for Social Welfare (WCFTR16)
[4] "An Optimization Model for Clustering Categorical Data Streams with Drifting Concepts”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, REVISION.
[5] “An Efficient Algorithm for Mining Frequent Item-sets over the Entire History of Data Streams”, Hua-Fu Li, Suh-Yin Lee and Man-Kwan Shan
[6]”Mining Frequent Patterns in Data Streams at Multiple Time Granularities,”.ChrisGiannella, Jiawei Han, Jian Pei, XifengYan, Philip S. Yu.
[7] New Algorithms for Finding Approximate Frequent Item Sets, Christian Borgelt1, Christian Braune1,2, Tobias Kotter3 and Sonja Grun4,5
[8] Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree, SujathaDandu, B.L.DeekshatuluPriti Chandra
[9] An Efficient Algorithm for Closed Item set Mining, Mohammed J. ZakiandChing-Jui Hsiao
[10] Mining Frequent Item sets Using Genetic Algorithm,Soumadip Ghosh, Sushanta Biswas, Debasree Sarkar, ParthaPratim Sarkar.