Mining Frequent Patterns from Data streams using Dynamic DP-tree

Shaik.Hafija,
M.Tech(CS),
JNTU Kakinada.

J.V.R. Murthy,
Phd.
Professor,
CSE Department,
JNTU Kakinada.

Y.Anuradha,
M.Tech, (Ph.D),
JNTU Kakinada.

M.Chandra Sekhar
M.Tech(IT)
JNTU Kakinada,

ABSTRACT
A DataStream is a real time continuous, ordered sequence of items. It is impossible to control the order in which items arrive, nor it is feasible to locally store a stream in reality. By short it is a rapid flow of continuous ordered data. By these specific characteristics static models and static two pass algorithms are not suitable to data streams. Data stream mining have following three challenges one every item is examined only once. Second the storage space should control even there is a large amount of data, third the mining results have to be produced as early as possible. In this paper we propose a novel method to mine the frequent items over data streams by dividing data as no of windows and mine frequent item sets over window using a very compact data structure DP-Tree and placing the every DP-Tree safely in disk space so that we can retrieve the tree structure for pruning as and when we require. More over we propose methods to dynamically construct and update the DP-Tree

Keywords
Data streams mining, Frequent Patterns, Tilted time window model, DP-Tree (Decreased Pattern Tree).

1. INTRODUCTION
In data streams transactions arrive continuously and the volume of transactions can be potentially infinite. Formally a data stream can be defined as a sequence of transactions. D={t1,t2,t3,….} where t_i is the ith arrived transaction . to process and mine the streams , different window models are often used. A window is subsequence between i-th and j-th arrived transactions, denoted by W[i,j]=(ti , ti+1, ti+2,……,tj)., i<j The order of the stream can be maintained either implicitly by arrival time and explicitly by time stamps.

There have been many algorithms developed for fast mining of frequent patterns, which can be classified into two categories. The first category, candidate generation and test approach, such as Apriori as well as many subsequent studies, are directly based on an anti-monotone Apriori property if a pattern with k items is not frequent, any of its super-pattern with (k+1) or more items can never be frequent.

2. PREVIOUS WORK
There are different Stream window models Sliding window, Damped window, Tilted time window and weighted sliding window. Manku and matwani (2002) propose lossy counting algorithm for computing an approximate set of FIs over the entire history of a stream. The stream is divided into a sequence of buckets and each bucket consists of B = ⌈1/ϵ⌉ transactions.Each bucket is also given an ID and the ID of the current bucket, bidτ , is assigned as bidτ = ⌈N/B⌉, where N is the number of transactions currently in the stream. Lossy Counting processes a batch of transactions arriving on the stream at a time, whereeach batch contains β buckets of transactions. Giannellia , Han.. proposed a technique for computing and maintaining all the frequent item sets in the datastreams, frequent patterns are maintained under a titied time window framework in order to answer time sensitive queries.They incrementally maintain tilted_time windows for each pattern at multiple time granularities.Moreover, inspired by the fact that the FP-tree provides an effective data structurefor frequent pattern mining, they develop Fp-stream, an effective FP-tree-based modelfor mining frequent patterns from data streams. Yun Chi, Haixum wang…et al propose a moment algorithm and in-memory data structure ,The closed enumeration tree, to monitor dynamically selected small set of item sets that enable usto answer the queries“ what are the current closed frequent item sets?”Syed Khiruzzaman Tanbeer, Chowdary Farhan Ahmed proposed CP-Tree the one pass algorithm to mine frequent item sets in large databases.

2.1 Motivation for proposed system
Apriori based algorithms take more time to mine frequent patterns because they need lot of temporary candidate item set generations. More over FP-tree is a two pass and static algorithm which required pre knowledge about the stream which may not be possible in data streams. So our proposed model Dynamic DP-tree offers an algorithm which creates and modifies the tree dynamically. Experimental results show that this will give better performance than the previous ones.

3. PROPOSED SYSTEM
3.1 Problem Statement
The problem of frequent pattern mining is to find the complete set of frequent patterns in a given transaction database with respect to a given support threshold

3.2 Overview of Proposed Work

After data has arrived, it will divide the data stream into partitions. First take the first partition and construct the DP-Tree(Dynamic Tree) which is a Prefix-Tree data structure. DP-Tree is efficient due to its support count descending order. DP–Tree further maintains the DFS order of the nodes. And as well as it maintains the link of same nodes. Because of these new features, DP-Tree becomes an efficient prefix tree for FP-growth algorithm to prune frequent patterns. More-over each such DP-Tree will be generated for each window of the data stream. Once a DP-Tree is generated and pruned it is stored in efficient disk based data structure called “Disk based time location table “ for future references. This mechanism of

storing the DP-Tree in disk will be useful when we want to retrieve the previous window for frequent pattern mining, instead of storing data as set of transactions we are storing a DP-Tree as it is in disk memory so that whenever one want to generate frequent patterns of previous windows they can just retrieve the DP-Tree from disk and apply the DP-growth algorithm to it.

3.3 Over View of DP-Tree

Let L={i1,i2,i3,….,in} be set of literals, called items that have ever been used as a unit information of an application domain. A set $X = \{ij, \ldots, ik\} \subseteq L$, ($j \leq k$ and $1 \leq j, k \leq n$) is celled a pattern. A transaction T={tid,Y} is a tuple where tid is a transaction id and Y is a pattern.

If X⊆Y it is said that T contains X or X occurs in T.

A transactional database DB over L is a set of transactions and |DB| is a size of the DB. Total number of transactions in DB. The support of pattern X in DB is the number of transactions in DB that contains X. A pattern is called frequent if its support is not less than the user specified support threshold min_sup, ϵ. Given a ϵ and a DB, discovering the complete set of frequent patterns in DB, say FDB is called the frequent Pattern mining problem.

Here we discuss the basics and step wise building of DP-Tree. In general, DP-tree achieves a frequency –descending structure by capturing part by part data from the database and dynamically restructuring itself after each part by using efficient restructuring mechanism. Like FP –tree, for tree traversal it maintains the Item List(I-list). The construction task mainly consists of two phases. First Insertion phase, That inserts transactions into DP-Tree as well as maintain the Node numbers in DFS order. The items of each transactions are inserted in the descending order of their support counts in I-list. More over DP-Tree contains the Depth First Order number for every node. And Item Location Table contains the item Ids and Their corresponding Link of nodes numbers as well as their support count SC in descending order of their support count. And Second phase is a Restructuring phase: That re arranges the Item list(I-list) according to the frequency-descending order of items and restructures the tree nodes according to new I-list . These two phases are executed alternatively at the end of DB. But in between after one partition over the tree is safely stored in the Disk using Disk based Time Location Table. So the this can later retrieve as much efficient as we can take the date from main memory.

3.4 Tree Restructuring

One of the two performance factor is the tree restructuring mechanism. Existing path adjusting method (PAM) , sorts the nodes of a pre-fix tree by using bubble sort method. We propose a new tree restructuring technique called branch sorting method (BSM) that unlike PAM, restructures by sorting unsorted paths in the tree one by one in frequency descending order of I-List. The time complexity to PAM is largely depends on distance among the items that has to be swap because it follows bubble sort algorithm, the time complexity is in O(n2). On the other hand BSM uses merge sort algorithm with the time complexity of O(n log 2 n); so BSM is efficient one

3.5 Item Location Table and Disk Time Location Table

This paper after constructing a tree smoothly handles that tree for pruning as well as preserving it in disk. For smooth pruning it maintains the Item Location table wich is a Hash table and Link list combined data structure. In Hash table it contains the Item id and their support count and link to the chain of nodes where that item located in DP-Tree. This data structure can be used to efficiently handling the pruning phase. More over if we use top down FP-Growth algorithm

Then the pruning time will also be reduced.

Disk Time Location table smoothly stores the tree after each partition (or window) in Disk. In a file. And file id are maintained in a list called F-list. So that whenever we want to retrieve Frequent items in past, we just retrieve the tree and apply pruning algorithm to it.

Fig 1 (b) represents the structure of the tree after inserting transactions 10, 20 and 30.in item appearance order. This will initiate the restructuring phase. The restructuring phase first sorts the I-list in descending order. And then rearranges the node according to the new I-list order. Fig 1(c) represents that all items having higher count is arranged at the upper part of the tree. DP-Tree at this stage is the frequency descending tree. The next insertions phase inserts transactions 40, 50 and 60 in the order of new I-list that is (a,b,c,d,e,f)
Instead of previous order(a,e,c,b,d,f). Fis 1(d) represents the tree after insertion phase. And Fig 1(e) represents the tree after restructuring phase.

3.6 Algorithms

Construction DP-Tree

Input: DB

Output: DP-Tree

Procedure:

1. For each transaction from DB one by one.
2. Make root of DP-Tree as null.
3. Call insertDP-Tree(t , DP-tree, I-list)
4. Call restructure DP-Tree(t, DP-Tree, I-list)

InsertDP-Tree (t, T, I-list)

i. For each item I in t

   If I-list already contains I increment its count otherwise add I to I-list and initialize count as 1.

ii. Take the items in the order of I-list let the order is Trans[p|P], where p is the first element of the transaction,

P is the remaining list.

iii. Call insert-tree([p|P], T), which is performed as follows, if T has a child N such that T.item-name=p.item-name, then increment the N's

Count by 1. Else create a new node N, and let it count be 1 , its parent link be linked to T and its node link to the node with the same

Item-name via the node link structure. If P is non-empty , call insert-treee(P,N) recursively

4. Restructure DP-Tree(t, T , I-list)

i. Sort the I-list in descending order of count value

ii. Traverse the tree in DFS take each path P of Tree and rearrange the items of P in I-list order

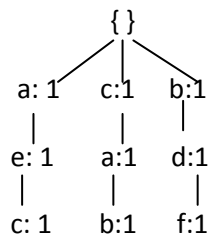
Trans DB

TID	ItemSet
10	a, e, c
20	c , a ,b
30	b, d , f
40	f, e, d
50	a, e, f
60	b, e d

I-list
a:2
e:1
c:2
b:2
d:1
f:1

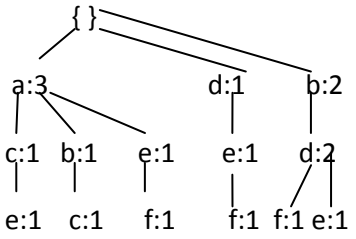
(a)

I-list
a: 2
e:1
c:2
b:2
d:1
f:1



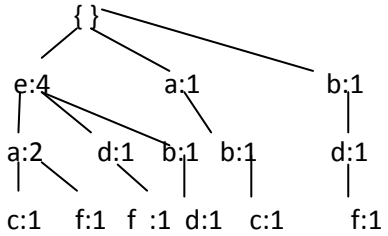
(b) After inserting TID 10-30

I-list
a: 3
b: 3
c:2
d:3
e:4
f:3



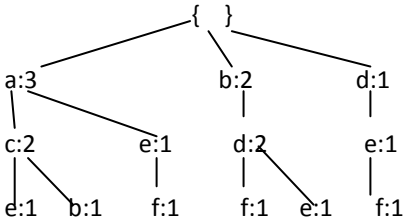
(d) After inserting TID 40-60

I-list
e: 4
a: 3
b: 3
d:3
f:3
c:2



(e) after restructuring TID 40-60

I-list
a:3
b:3
c:2
d:3
e:4
f:3



(f) CanTree in lexicographic order

Fig 1(a, b, c, d, e, f) : Construction of DP-Tree and Can Tree

Item Location Table

Item	Support Count	DFS Node numbers
a	2	1
b	2	4,6
c	2	2,5
d	1	7
e	1	3
f	1	8

DTLT

TreeId	File ID
Tree0	File1
Tree1	File2
Tree2	File3

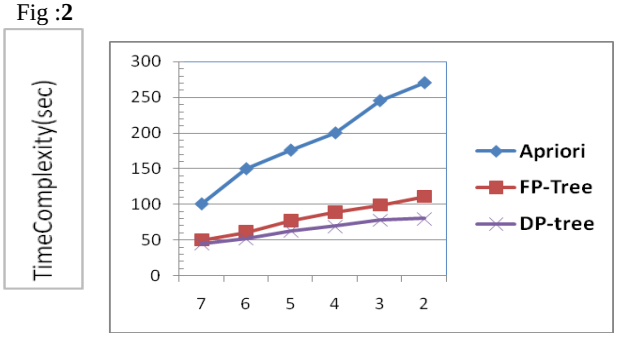
5. EXPERIMENTAL RESULTS

To measure the performance of DP-tree , we conducted several experiments on Intel(R) core (TM) i3-233M CPU with 2.20GHz. And 4GB memory, 32 bit operating system. Windows -7. All the programs are implemented using Java and JSP, The method used to generate synthetic transactions is similar to the one used in apriori (Agrawal & Srikanth, 1994) table 1 summarizes the parameters used in this experiment. The dataset is generated by setting N=1000 and |L|=2000, we use Ta.lb.Dc to represent that |T|=a, |I|=b and |D|=c x 1000.

To simulate the data streams in the this tilted window model , transactions in the synthetic data are processed in sequence. Foe simplicity we assume that the memory size is 20 transactions . Fig 2 compare the efficiency of the DP-algorithm with apriori (WSW), FP-Tree, and CP-Tree for different support count value.

Fig 3 shows the time complexity for different number of transactions shows the time complexity vs average size of the transaction.

Fig :**2**



Support Count

Fig:3

Average size of the transaction

6. CONCLUSION

This paper we propose a efficient algorithm DP-Tree based on Tilted time window model. This algorithm consists of two phases insertion and restructuring and also storing the trees smoothly in disk using Disk Time Location Table.

7. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template.

8. REFERENCES

[1] Pauray S.M. Tsai, 2009, Mining Frequent Itemstes in data streams using the weighted sliding window model.

[2] James Cheng , Yiping Ke, Wilfred Ng, A survey on Algorithms for Mining Frequent Itemsets over Data streams.

[3] Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, Philip S.Yu, Mining Frequent Patterns in Data Streams at Multiple Time Granularities

[4] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee CP-Tree\: A Tree Structure for Single Pass Frequent Pattern Mining

[5] Chih-Hsiang Lin Ding-Ying Chiu Yi-Hung Wu, Arbee L. P. Chen Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window

[6] M. Deypir , M.H. Sadreddini ,Frequent Patterns Mining over Data Stream Using an Efficient Tree Structure.

[7] William Cheung and Osmar R. Zaiane, Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint

AUTHUR'S PROFILE

Sri Dr. J.V.R Murthy, who is guide of author, is currently working as professor in computer science department of JNTU Kakinada. He completed B.Tech from JNTUK in 1982, M.Tech from IIT kharagpur in 1990 and received Ph.D from JNTU Kakinada in 2004. His achieved a Momento by KESPAN ENERGY New York for successful implementation of people soft HRMS. His areas of interests are Database management Systems and E.R.P. His research paper is "porting details from Motorola 68020 to Intel 80386 "

Shaik. Hafija is a Author, who completed MCA from GITAM college, Visakhapatnam in 2008 and per suing M.Tech in Computer Science Branch from JNTU Kakinada, AP, India. And I have good encouragement from my guide Sri Dr.J.V.R Murthy and Y.Anuradha madam.

Mr. M. Chandrasekhar, per suing M.Tech in Information Technology from JNTU Kakinada and Completed B .Tech (I.T) from Vignan Engg College, Guntur, AP. Helped a lot for preparing current menuscript.