A Novel Approach for Finding Frequent Item Sets Done by Comparison based Technique

Meghna Utmal
HOD, MCA
Gyan Ganga College of Technology, Jabalpur, (M.P.)Pin-482003 , India

Shailendra Chourasia
M.Tech, Scholar,
Gyan Ganga College of Technology, Jabalpur (M.P.)
Pin- 482003, India

Rashmi Vishwakarma
Lecturer, MCA Department,
Gyan Ganga Institute Of Technology& Sciences, Jabalpur, (M.P.)Pin-482003,India

ABSTRACT
Frequent pattern mining has been a focused theme in data mining research for over a decade. Abundant literature has been dedicated to this research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining, associative classification, and frequent pattern-based clustering, as well as their broad applications. In this paper, we develop a new technique for more efficient pattern mining. Our method find frequent 1-itemset and then uses the heap tree sorting we are generating frequent patterns, so that many. We present efficient techniques to implement the new approach.

Key words
Frequent pattern mining, MAXHEAP, Data mining, Data Structure..

1. INTRODUCTION

Agrawal et al. (1993) was the first man who has proposed frequent pattern mining for market basket analysis in the form of association rule mining. In this he analyses customers shopping basket to finding buying habits by finding associations between the different purchased items by customers.

In this paper, we perform a high-level overview of frequent pattern mining methods, extensions and applications. We proposed new method in the place of FP-Growth tree using MAXHEAP tree, in this tree we are trying to find most frequent item which is already sorted and present in root of heap tree. we organize our discussion into the following three themes: (1) Efficient and scalable FP-Tree Method for mining frequent patterns (2) Data Structue (3) Proposed Methods with example

2. EFFICIENT AND SCALABLE FP-TREE METHOD FOR MINING FREQUENT PATTERNS

2.1 Problem Specification

The concept of frequent itemset was first introduced for mining transaction databases (Agrawal et al. 1993). Let I = {I1, I2, . . . , In} be a set of all items. A k-itemset α, which consists of k items from I, is frequent if α occurs in a transaction database D no lower than θ |D| times, where θ is a user-specified minimum support threshold (called min_sup) , and |D| is the total number of transactions in D. The basic frequent itemset mining methodology FP-growth is introduced in next section.

2.2 FP-growth

In order to count the supports of all generated itemsets, FP-growth uses a combination of the vertical and horizontal database layout to store the database in main memory. Instead of storing the cover for every item the database, it stores the actual transactions from the database in a tree structure and every item has a linked list going through all transactions that contain that item. This new data structure is denoted by FP-tree (Frequent-Pattern tree) and is created as follows Again; we order the items in the database in support ascending order for the same reasons as before. First, create the root node of the tree, labeled with "null". For each transaction in the database, the items are processed in reverse order (hence, support descending) and a branch is created for each transaction. Every node in the FP-tree additionally stores a counter which keeps track of the number of transactions that share that node. Specifically, when considering the branch to be added for a transaction, the count of each node along the common prefix is incremented by 1, and nodes for the items in the transaction following the prefix are created and linked accordingly. Additionally, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links. Each item in this header table also stores its support. The reason to store transactions in the FP-tree in support descending order is that in this way, it is hoped that the FP-tree representation of the database is kept as small as possible since the more frequently occurring items are arranged closer to the root of the FP-tree and thus are more likely to be shared.

Example . Assume we are given a transaction database and a minimal support threshold of 2. First, the supports of all items is computed, all infrequent items are removed from the database and all transactions are reordered according to the support descending order resulting in the example transaction database in table .

Table 1 An example preprocessed transaction database.

Tid	List of Items
100	{Pen, Ink, Rubber}
200	{ Pen, Ink, Pencil }
300	{Pencil, Rubber}
400	{Ink, Pencil}
500	{Register, Ink, Pen}

The FP-tree for this database is shown in Figure

Fig. Header Table

Item	SC	Node
Ink	4	
Pen	3	
Pencil	3	
Rubber	2	
Register	1	

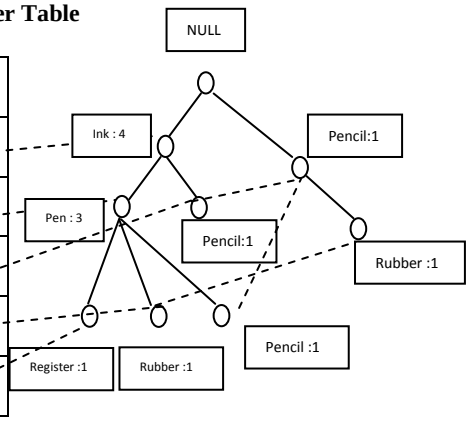


Table 2 Result of fp-growth tree

Item	Conditional Pattern Base	Conditional FP Tree	Frequent Pattern Generates
Register	{Ink , Pen :1 }	-	-
Rubber	{Ink, Pen : 1} {Pencil :1)	-	-
Pencil	{Ink , Pen :1 } {Ink : 1}	{Ink :2}	{Ink , Pencil :2}
Pen	{Ink :3 }	{Ink :3 }	{Ink , Pen :3 }

Algorithm FP-growth

Input: D,σ, I ⊆ I

Output: F[I](D, σ)

1: F[I] := {}

2: for all i ∈ I occurring in D do

3: F[I] := F[I] ∪ {I ∪ {i}}

4: // Create D^i

5: D^i := {}

6: H := {}

7: for all j ∈ I occurring in D such that j > i do

8: if support(I ∪ {i, j}) ≥ σ then

9: H := H ∪ {j}

10: end if

11: end for

12: for all (tid ,X) ∈ D with i ∈ X do

13: $D^i := D^i \cup \{(tid, X \cap H)\}$

14: end for

15: // Depth-first recursion

16: Compute $F[I \cup \{i\}](D^i,\sigma)$

17: F[I] := F[I] ∪ F[I ∪ {i}]

18: end for

3 Data structures

The data structures are user defined data types specifically created for the manipulation of data in a predefined manner. There are two types of data structure Linear and Non Linear. Array, stacks, queues and Linked List are Linear Data structure whereas trees and graphs are non linear data structure.

3.1 Tree

Trees are very flexible, versatile and powerful data structures .A tree is a non linear data structure in which items are arranged in a sorted sequence. It is used to represent Hierarchical relationship existing amongst several data structures.

There are different types of trees like Binary tree, B-Tree, B+Tree ,AVL Tree, Extended Tree ,Heap Tree etc. All trees manage data in different types. Here we will explain the functioning of Heap Tree.

3.2 Heap

The Heap is used in an elegant sorting algorithm called heapsort. Suppose H is a complete binary tree with n elements is maintain in memory using the sequential representation of H. Then H is called the Heap.

Heap is of Two types Max Heap and Min Heap. If root is greater than or equal to their left and right child then it is called Max Heap.If root is less than or equal to their left and right child then it is called Min Heap.

3.2.1 Operations on Max Heap

We can perform different types of operations on Max Heap i.e. Insertion, Deletion, Sorting, Traversing and Searching.

Once a heap has been built, Heapsort can simply remove the maximum value (root node) and create the output, sorted array

one item at a time. This process is somewhat akin to the Selection Sort but much more efficient.

When the root node in a heap is removed to become part of the final, ordered data set, the last item on the heap is promoted to fill the vacancy at the root position. Clearly, in many cases, this last item will now be out of place (that is, it may be smaller than one of its new children). To ensure that the modified heap retains the max-heap property it becomes necessary to ``push down'' the newly promoted root item until it is back in the right place. This pushing down process entails examining the node's key value and comparing it with the key value of the node's greatest child. If the node's greater child is larger in value than the node itself, a swap is performed. The process repeats, following the node from the root down through demotion, until no swap is needed. At this point the heap is back in order, the new root may be popped off, and the sorting process can continue.

The process of removing the root, promoting the last node, and re-heapifying continues until the heap is exhausted.

4 MAX HEAP Pattern Mining Algorithm :

We are introducing a new algorithm for finding the frequent pattern mining which is based on MAXHEAP.

4.1 Scan :

1. L1 = Scan-Database-for-Support-Count

2. for each ITEM.ID ∈ ITEM

3. {

4. for each transaction t ∈ D

5.{

6. if (id ∈ t)

7. ITEM..ID.count++

8. } ITEM = {ID.count >= min_sup}

9.}

4.2 Algorithm: InsertHeap (TREE, N, ITEM)

A Heap H with N elements is sorted in the array of structure TREE and an ITEM of information is given. This procedure inserts item as a new element of H. PTR gives the location of ITEM as it rises in the tree, and PAR denotes the location of the parent of ITEM.

1. [Add new node to H and initialize PTR.]

Set N= N+1 and PTR = N

2. [Find location to insert ITEM]

Repeat Steps 3 to 6 while PTR<1.

3. Set PAR = ⌊PTR/2⌋ [Location of parent node]

4. If ITEM = TREE[PAR] then

Set TREE[PTR] = ITEM and Return

[End of If structure]

5. Set TREE[PTR]=TREE[PAR] [Move node down]

6. Set PTR = PAR [Updates PTR]

[End of step 2 loop]

7. [Assign ITEM as the root of H.]

Set TREE[1] = ITEM

8. Return

4.2 Iterative Traversal

The term traversal means visit/read each node at least ones. In iterative traversing we will traverse the tree and scan the frequent itemset according to the iteration.

Consider the following transactional database having transaction id (tid) and List of items.

Table 3

Tid	**List of Items**
100	{Pen,Ink, Rubber}
200	{ Pen,Ink, Pencil }
300	{Pencil,Rubber}
400	{Ink, Pencil}
500	{Register,Ink,Pen}

As our algorithm first we need to scan database for the Support Count, that will generate following output

Table 4

Item	Support Count
Pen	3
Ink	4
Rubber	2
Pencil	3
Register	1

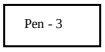
Now we will discard the Items that have support count less than Minimum Support Count. In our case Minimum Support Count is 2.

So after discarding Item that has less than 2 support count, the output will be as follows;

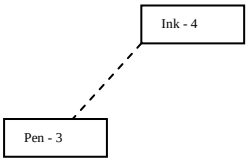
Table 5

Item	Support Count
Pen	3
Ink	4
Rubber	2
Pencil	3

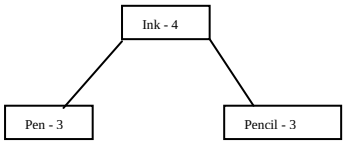
Now we use **InsertHeap(TREE,N,ITEM)** procedure to create Max-Heap.



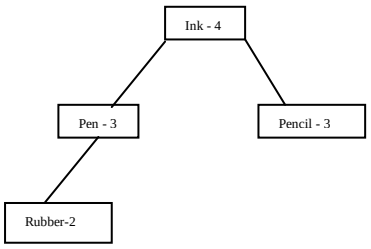
(i) ITEM = Pen-3



(ii) ITEM = Ink-4



(iii) ITEM = Pencil-3



(iv) ITEM = Rubber-2

Now by **Iterative Traversing** procedure we generate frequent n-Itemset where n = 1,2,3 … n by using a this traversing the output of the following HeapTree will be

1. Frequent 1 Itemset

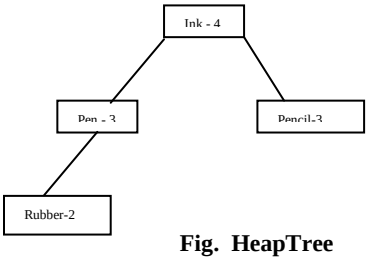


Fig. HeapTree

Iterative Traversal: The output of the above HeapTree by a this method will be as follows & find most frequent item;

1. Iteration 1

Frequent 1 Itemset will generate

{Ink-4, Pen-3, Pencil-3,Rubber-2}

2. Iteration 2

Frequent 2 Itemset will generate

{Ink-4 Pen-3, Ink-4 Pencil-3, Pen-3 Rubber-2}

3. Iteration 3

Frequent 3 Itemset will generate

{Ink-4 Pen-3 Pencil-3}

5. CONCLUSION

In this paper, we present a overview of FP-Growth Tree. With over a decade of extensive research, have been hundreds of research publications and tremendous research, development and application activities in this domain. We are trying to exploring a new approach on the frequent pattern mining. It is impossible for us to give a complete coverage on this topic with limited space and our limited knowledge. Hopefully, this short overview may provide a rough outline of our recent work and give just idea of a general view of the MAXHEAP tree by using comparison based technique. In general, we feel that as a young research field in data mining, frequent pattern mining has achieved tremendous progress and claimed a good set of applications. However, in-depth research is still needed on several critical issues so that the field may have its long lasting and deep impact in data mining applications.

6. REFERENCES

[1] Afrati FN, Gionis A, Mannila H (2004) Approximating a collection of frequent sets. In: Proceedings of the 2004 ACM SIGKDD international conference knowledge discovery in databases (KDD'04), Seattle,WA, pp 12–19.

[2] Agarwal R, Aggarwal CC, Prasad VVV (2001) A tree projection algorithm for generation of frequent itemsets. J Parallel Distribut Comput 61:350–371.

[3] Aggarwal CC, Yu PS (1998) A new framework for itemset generation. In: Proceedings of the 1998 ACM symposium on principles of database systems (PODS'98), Seattle,WA, pp 18–24.

[4] Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM-SIGMOD international conference on management of data (SIGMOD'98), Seattle, WA, pp 94–105.

[5] Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993ACM-SIGMODinternational conference on management of data (SIGMOD'93), Washington, DC, pp 207–216.

[6] Agrawal R, Shafer JC (1996) Parallel mining of association rules: design, implementation, and experience. IEEE Trans Knowl Data Eng 8:962–969.

[7] Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 1994 international conference on very large data bases (VLDB'94), Santiago, Chile, pp 487–499.

[8] Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the 1995 international conference on data engineering (ICDE'95), Taipei, Taiwan, pp 3–14.

[9] Ahmed KM, El-Makky NM, Taha Y (2000) A note on "beyond market basket: generalizing association rules to correlations". SIGKDD Explorations 1:46–48.

[10] AsaiT, AbeK,Kawasoe S, ArimuraH, SatamotoH,Arikawa S (2002) Efficient substructure discovery from large semi-structured data. In: Proceedings of the 2002 SIAM international conference on data mining (SDM'02), Arlington, VA, pp 158–174.

[11] Aumann Y, Lindell Y (1999) A statistical theory for quantitative association rules. In: Proceeding of the 1999 international conference on knowledge discovery and data mining (KDD'99), San Diego, CA, pp 261–270.

[12] Bayardo RJ (1998) Efficiently mining long patterns from databases. In: Proceeding of the 1998 ACM-SIGMOD international conference on management of data (SIGMOD'98), Seattle,WA, pp 85–93

[13] Beil F, EsterM, Xu X (2002) Frequent term-based text clustering. In: Proceeding of the 2002 ACM SIGKDD international conference on knowledge discovery in databases (KDD'02), Edmonton, Canada, pp 436–442.

[14] Bettini C, SeanWang X, Jajodia S (1998) Mining temporal relationships with multiple granularities in time sequences. Bull Tech Committee Data Eng 21:32–38.

[15] Beyer K, RamakrishnanR(1999) Bottom-up computation of sparse and iceberg cubes. In: Proceeding of the 1999ACM-SIGMODinternational conference on management of data (SIGMOD'99), Philadelphia, PA, pp 359–370.

[16] Blanchard J, Guillet F, Gras R, Briand H (2005) Using information-theoretic measures to assess association rule interestingness. In: Proceeding of the 2005 international conference on data mining (ICDM'05), Houston, TX, pp 66–73.

[17] Frequent pattern mining: current status and future directions Bonchi F, Giannotti F, Mazzanti A, Pedreschi D (2003) Exante: anticipated data reduction in constrained pattern mining. In: Proceeding of the 7th European conference on principles and pratice of knowledge discovery in databases (PKDD'03), pp 59–70.

[18] Bonchi F, Lucchese C (2004) On closed constrained frequent pattern mining. In: Proceeding of the 2004 international conference on data mining (ICDM'04), Brighton, UK, pp 35–42

[19] Borgelt C, Berthold MR (2002) Mining molecular fragments: finding relevant substructures of molecules. In: Proceeding of the 2002 international conference on data mining (ICDM'02), Maebashi, Japan, pp 211–218.

[20] Brin S, Motwani R, Silverstein C (1997) Beyond market basket: generalizing association rules to correlations. In: Proceeding of the 1997 ACM-SIGMOD international conference on management of data (SIGMOD'97), Tucson, AZ, pp 265–276.

[21] Brin S, Motwani R, Ullman JD, Tsur S (1997) Dynamic itemset counting and implication rules for market basket analysis. In: Proceeding of the 1997 ACM-SIGMOD international conference on management of data (SIGMOD'97), Tucson, AZ, pp 255–264.

[22] Bucila C, Gehrke J, Kifer D, White W (2003) DualMiner: a dual-pruning algorithm for itemsets with constraints. Data Min knowl discov 7:241–272.

[23] Burdick D, Calimlim M, Gehrke J (2001) MAFIA: a maximal frequent itemset algorithm for transactional databases. In: Proceeding of the 2001 international conference on data engineering (ICDE'01), Heidelberg, Germany, pp 443–452.

[24] Calders T,Goethals B (2002) Mining all non-derivable frequent itemsets. In: Proceeding of the 2002 European conference on principles and pratice of knowledge discovery in databases (PKDD'02), Helsinki, Finland, pp 74–85.

[25] Calders T, Goethals B (2005) Depth-first non-derivable itemset mining. In: Proceeding of the 2005 SIAM international conference on data mining (SDM'05), Newport Beach, CA, pp 250–261.

[26] Cao H, Mamoulis N, Cheung DW (2005) Mining frequent spatio-temporal sequential patterns. In: Proceeding of the 2005 international conference on data mining (ICDM'05), Houston, TX, pp 82–89.

[27] Zaki MJ (2002) Efficiently mining frequent trees in a forest. In: Proceeding of the 2002 ACM SIGKDD international conference on knowledge discovery in databases (KDD'02), Edmonton, Canada, pp 71–80

[28] Zaki MJ,Hsiao CJ (2002) CHARM: an efficient algorithm for closed itemset mining. In: Proceeding of the 2002SIAMinternational conference on data mining (SDM'02),Arlington,VA, pp 457–473.

[29] Zaki MJ, Lesh N,OgiharaM(1998) PLANMINE: sequencemining for plan failures. In: Proceeding of the 1998 international conference on knowledge discovery and data mining (KDD'98), New York, NY, pp 369–373.

[30] Zaki MJ, Parthasarathy S, Ogihara M, Li W (1997) Parallel algorithm for discovery of association rules. data mining knowl discov, 1:343–374.

[31] Zhang X, Mamoulis N, Cheung DW, Shou Y (2004) Fast mining of spatial collocations. In: Proceeding of the 2004 ACM SIGKDD international conference on knowledge discovery in databases (KDD'04), Seattle,WA, pp 384–393.

[32] Zhang H, Padmanabhan B, Tuzhilin A(2004) On the discovery of significant statistical quantitative rules. In: Proceeding of the 2004 international conference on knowledge discovery and data mining (KDD'04), Seattle,WA, pp 374–383.

[33] Zhu F, Yan X, Han J, Yu PS, Cheng H (2007) Mining colossal frequent patterns by core pattern fusion. In: Proceeding of the 2007 international conference on data engineering (ICDE'07).