Efficient Implementation of Leap Traversal for FP-Tree Mining

M.Shashi[1], Ch. Sujatha[2], M. Sudeepth[3]
[1]Department of Computer Science & Systems Engineering, Andhra University, Visakhapatnam,
smogalla2000 @ yahoo.com
[2]Department of Computer Science & Systems Engineering, Andhra University, Visakhapatnam,
chatrasi_sujatha@ yahoo.com
[3]Department of Computer Science, ANITS, Visakhapatnam,
sudeepthi_mogalla@ yahoo.co.in

Abstract

Most of the research on frequent itemset mining at early stages explored the itemset lattice breadth-wise. There after, with the advent of FP-Tree representation of transac-tional database, depth-wise exploration is found to be prom-ising. Recently COFI* algorithm introduced Leap Traversal approach to extract (maximal) frequent itemsets. We propose a new array-based, data structure called Conditional Pattern Lattice (CPL) to represent the conditional pattern base suit-able for Leap Traversal. A non-recursive algorithm for imple-menting Leap Traversal using CPL is developed and its per-formance is found to be better than the other state of art min-ing methods.

Key words

Association mining, maximal frequent itemsets, FP-Tree, leap traversal, Conditional pattern lattice

1. Introduction

An association rule of the form X=>Y suggests the occurrence of Y in the context of X. When applied to Market Basket Analysis, X and Y represent non-overlapping and non-empty subsets of universal item set, I, for a given transaction database. In a simple case of a five element universal item set, I= $\{i_1, i_2, i_3, i_4, i_5\}$ the association rule of the form $\{i_3, i_5\}$ => $\{i_2\}$ sug gests the occurrence of item i_2 in a transaction involving items i_3 and i_5.This can be used to predict possible customers for item i_2.

Association Mining aims at discovering frequently co-occurring items referred to as frequent itemsets/patterns in a transaction database in order to generate association rules. The first phase of association mining deals with the extrac-tion of frequent itemsets/patterns from an exponentially (in the number of items) large collection of candidate itemsets and thereby requires efficient algorithms. The second phase of association mining makes use of the outcome of the first phase to generate association rules which is comparatively trivial and is beyond the scope of this paper.

Most of the earlier developments towards the extraction of frequent itemsets are based on the Apriori property. According to the Apriori property, all the supersets of an itemset (infrequent) can be eliminated from the candidate itemsets as soon as the itemset is found to be infrequent. Apriori algorithm [1],[2] makes a level-wise exploration of the itemset lattice, starting from single itemsets. Before scanning the transaction database for counting the support of a candi-date k-itemset, it is made sure that all its (k- 1)-item subsets are frequent. Thus Apriori algorithm requires as many database scans as there are elements in the largest frequent itemset. In case of very large databases because of limited main memory, each database scan, in turn, requires swap in/out of database partitions one after the other. Due to this, the basic Apriori algorithm is not suitable for handling large transaction data-bases. Many variations [3],[5],[10] to the basic approach to apriori algorithm are developed to reduce the number of data-base scans. Most of these approaches generate a large collection of candidate itemsets in the process of identifying frequent itemsets.

A different approach to frequent itemset mining known as Frequent Pattern Growth, does not require to generate can-didate itemsets. It makes use of a data structure, FP-Tree, to limit the number of database scans to two. After identifying frequent items in the first database scan, the transaction data-base is compressed and represented in the form of FP-Tree in the next scan. A path in the FP-Tree represents an itemset/ pattern along with its frequency of occurrence. Reduction in the size of FP-Tree compared to the corresponding transac-tion database is two fold. Primarily, the level of granularity is increased from transaction level to itemset/pattern level in FP-Tree. The other factor is that, multiple itemsets with common prefix share a single path in the FP-Tree. FP-Tree, the compact form of transaction database in terms of active pat-terns and their frequencies, is memory resident and hence, no more I/O scans are required during the mining process.

The active portion of itemset/pattern lattice is divided into different partitions such that each partition is associated with a particular focused frequent item and it contains pat-terns involving the co-occurrence of the focused item along with more frequent items compared to itself. Such a partition is called Conditional Pattern Base of the focused item. Frequent Pattern Growth approach mines each of these condi-tional pattern bases separately by building conditional FP-Trees [9] recursively. Starting from the conditional pattern base of the frequent item with the least support count, this algorithm makes a depthwise exploration of the itemset lattice to find the possibility of a pattern's growth once it is found to be frequent. Being a recursive algorithm, FP-Growth requires a lot of stack space for extracting long frequent patterns. In order to improve the efficiency of mining process, various attempts are made [6], [11] towards devising a non-recursive approach to mining FP-Tree. Co-Occurrence Frequent Item (COFI) Tree mining is an efficient non-recursive algorithm [7] which builds a separate COFI-Tree rooted with a focused frequent item for mining its conditional pattern base. Each node of COFI-Tree maintains a participation count, in addition to the support count of the item represented by the node. COFI-Tree mining requires comparatively less memory since a COFI-Tree associated with a focused item is built, mined and discarded before the next COFI-Tree is build.

Some of the recent developments concentrated on extracting maximal frequent itemsets/patterns rather than extracting all frequent patterns. A frequent itemset is said to be maximal frequent if none of its supersets is also frequent in the transaction database. Since all the subsets of a maximal frequent itemset are obviously frequent, a small set of maximal frequent itemsets can be considered as a concise representation of the set of all frequent itemsets. In addition to association rule mining, identification of maximal frequent patterns finds its application in clustering, web mining, classification [4], [12] etc.

COFI* algorithm[8] is one of the most recent algorithms for finding the maximal frequent itemsets/patterns. The algo-rithm uses the FP-Tree with bi-directional parent-child links to extract conditional pattern bases for each of the frequent items in a given transactional database. The algorithm introduces a new data structure called Ordered-Partitioning-Bases **(OPB)** to group the patterns of a conditional pattern base by their lengths. This non-recursive algorithm constructs a separate COFI-Tree for mining maximal frequent patterns involving each of the frequent 1-itemset. It introduces the concept of Leap Traversal for mining COFI-Trees.

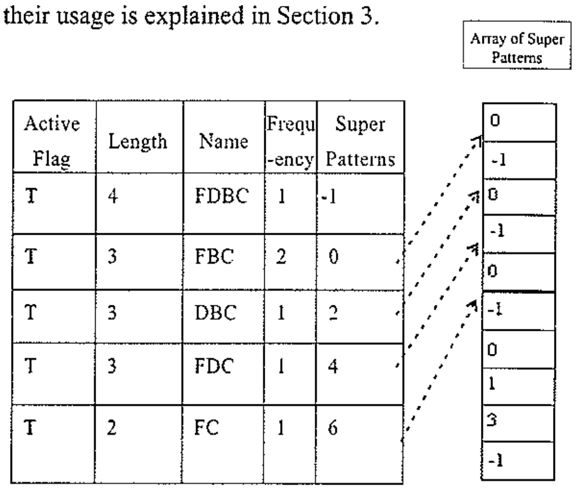
In this paper, a simplified approach to leap traversal of itemset lattice for finding maximal frequent itemsets is pro-posed. A new data structure namely Conditional Pattern **Lat-tice (CPL)** is devised to represent the reduced conditional pattern base of a focused frequent item. This data structure explicitly represents the "Contained-in" (Í) relationship among the patterns which is essential for mining process. A non-recursive algorithm that makes use of this data structure, CPL, to effectively extract the frequent itemsets is proposed. The proposed mining algorithm requires very less memory compared to the other proven algorithms used for FP-Tree mining.

The rest of the paper is organized as follows. In Section 2 the design of Conditional Pattern Lattice is presented. Suitability of Conditional Pattern Lattice for leap traversal of itemset lattice is discussed in section.3. Section 4 provides the algorithm in full length and explains the algorithm with a simple running example. Section 5 tabulates the experimental results and compares the proposed algorithm with established FP-Growth algorithm and the recent COFI* algorithm. Con-clusions are given in Section 6.

2. Conditional Pattern Lattice

This is a new data structure devised by us to sup-port mining of a conditional pattern base associated with a focused frequent item. The conditional pattern base is a part of itemset/pattern lattice, whose elements are partially related by "Contained-in" (Í) relation, which is essential for mining process. This partial relation, "Contained-in", plays an impor-tant role in the mining process in the sense that the support of a pattern is determined as the summation of the frequencies of all its super patterns. The data structure, Conditional Pattern Lattice(CPL), provides a simple way of accommodating the "Contained-in" relationship among the itemsets of conditional pattern base. It represents each pattern in terms of name of the pattern, its length, frequency of co-occurrence, active flag and an index to the list of its super patterns. An array of such patterns constitutes Conditional Pattern Lattice associated with a focused frequent item. A global array of indices is used to accommodate the list of super patterns of each entry of CPL one after the other with -1 as delimiter.

For example, let the conditional pattern base of a focused frequent item H be {<FDBC: 1>, <FBC:2>, <DBC: 1>, <FDC:1>, <FC:1>}. The corresponding Conditional Pattern Lattice is shown in Figure 1. The first pattern FDBC is repre-sented by the 0th entry of the CPL. The pattern contains 4 items and it co-occurs only once with the focused item. The '-1' in the last field indicates that there are no super patterns to it. Similarly the last pattern FC is represented by the 4th entry of the CPL and its list of super patterns is stored in 6th to 9th entries of the array of super patterns. All active flags are ini-tialized to true and their usage is explained in Section 3.



Active Flag	Length	Name	Frequ -ency	Super Patterns
T	4	FDBC	1	-1
T	3	FBC	2	0
T	3	DBC	1	2
T	3	FDC	1	4
T	2	FC	1	6

Array of Super Patterns
0
-1
0
-1
0
-1
0
1
3
-1

Figure 1: Conditional Pattern Lattice of H.

3. Leap Traversal Using Conditional Pattern Lattice

A simple non-recursive algorithm to extract maximal frequent itemsets/patterns from a transaction database is proposed. The algorithm uses the data structure FP-Tree with bi-directional parent-child links to represent the essence of the transaction data-base compactly. Once the FP-Tree is constructed, the mining of frequent patterns starts with focus on the frequent item with least support count and proceeds to focus on the more frequent items as was suggested in FP-Growth algorithm. FP-Tree is traversed in bottom-up manner for finding the conditional pattern bases of frequent items one by one. Conditional pattern base of a focused frequent item consists of patterns/itemsets involving the co-occurrence of a focused frequent item with other items having more glo-bal frequency than itself. However, it is possible that an item i, which is globally frequent enough, might not have sufficient number of co-occurrences with the focused frequent item, 'f'. It implies that no frequent pattern involving 'f' also contains such an item, 'i'. So the patterns constituting the F-conditional pattern base are processed to find co-occurrence fre-quencies of relevant items and accordingly the patterns are trimmed/reduced by removing infrequent items. The collection of trimmed patterns in the conditional pattern base are represented as a Conditional Pattern Lattice associated with a focused frequent item, 'f'.

Starting from the longest pattern to the shortest pattern, the trimmed patterns are represented by successive entries of a partially ordered (length-wise) array that stands for the Conditional Pattern Lattice. An entry of the CPL accom-modates a reduced pattern in terms of its name, length, active flag, co-occurrence frequency and a pointer to the list of its super patterns. The length of a pattern is the number of items constituting it and it also indicates the level of the pattern in the itemset lattice. The super patterns of each pattern/entry are found by performing subset-checking of the present pattern in the patterns already entered in the Conditional Pattern Lattice at higher levels (longer patterns). Subset checking involves comparing the corresponding items of the given pair of patterns as long as they match. Since the items of a pattern are in the ascending order of their global frequencies, most of the itemsets which are not related through "Contained-in" relationship are identified at an early stage of subset-check-ing process. This process results in an ordered list of CPL indices to super patterns of the present pattern (except itself) and it is attached to the corresponding entry of the CPL through a pointer. A pattern is active until and unless it is recognized as either a maximal frequent itemset or a subset of a maximal frequent itemset. The active flags of all entries are initialized to True.

The frequency of a pattern indicates the number of co-occurrences of the corresponding itemset (as a whole and on its own) with the focused item. In addition to this, a pattern inherits co-occurrence frequencies of all its super patterns, if any, in the CPL. Hence the support of a pattern is the summa-tion of the frequencies of all its super patterns including itself. The list of CPL indices to super patterns is used in order to find the support of a pattern.

Extraction of local maximal frequent patterns is taken up in two passes over the entries of the CPL. In the first pass, the support count of each of the active patterns is found and if the support is more than the given support threshold, the pattern is entered into the list of Maximal Frequent Itemsets. The active flag of the corresponding entry of CPL is made false. Once a pattern is recognized as a maximal frequent pat-tern, none of its sub-patterns can be maximal frequent. Hence, the process of counting the support of a pattern terminates upon finding an inactive pattern in the list of its super pat-terns and the pattern is made inactive.

In the second pass, the leap traversal of the active part of the Conditional Pattern Lattice is carried out to extract the hidden maximal frequent patterns/itemsets. Unlike the member patterns of the Conditional Pattern Lattice, a hidden pat tern may not confine itself to a single prefix path in the FP-Tree. Instead, a hidden pattern occurs on two or more prefix paths. The common items of the active member patterns representing these prefix paths constitute a hidden pattern which is possibly maximal frequent. The successive entries of CPL representing active patterns are processed as follows. A new non-null itemset is generated as the intersection of the present pattern with a lower indexed, overlapping active pattern. A pair of patterns which are not related through "Contained-in" (c) relation are referred to as overlapping patterns. The or-dered list of super patterns associated with the present pattern is used to identify overlapping patterns. For example if the list of super patterns of i th entry in CPL is (k,k+2,...) where k<i the (k+1) th entry of CPL is possibly an overlapping pattern of i th pattern and if its active flag is true, the intersection of i th and (k+l) th pattern, if non-null, can be considered as a candidate for hidden maximal frequent pattern. The support count of the new itemset is calculated as the summation of the occurrence frequencies inherited from all patterns of CPL con-taining the new itemset.

If the support count of the new itemset is greater than support threshold, then the new itemset is entered into the set of local maximal frequent patterns. Otherwise, the pattern generated as an intersection of the new itemset with the remaining patterns in the CPL should be explored in the similar way. This step aims at finding hidden patterns resulting from intersection of two or more member patterns. For example while finding fre-quent itemsets (with minimum support count, σ =5) three conditional patterns namely ABCDEF, ABEFGH and ABCDGH, each having a frequency of 2 result in a hidden pattern AB with a frequency of 6. Such pattern which is formed as an intersection of more than two conditional patterns could be recognized as detailed below: The intersection of ABCDEF and ABEFGH forms a new itemset ABEF whose support count is only 4. Due to lack of support count the new itemset ABEF should be further explored by taking its intersection with the remaining pattern ABCDGH and there by the resultant itemset AB is found to be having a support count of 6 and entered into local maximal frequent itemsets.

Some more hidden patterns are possibly generated as an intersection of i th pattern with the next overlapping active pattern identified by processing the remaining portion of the list of super patterns of i th entry of CPL. In this way the second pass over the entries of CPL, brings out the hidden maximal frequent patterns and adds them to the list of local maximal frequent patterns of the focused item. Once a maximal frequent itemset is identified, the support count of its sub-patterns is calculated as the summation of co-occurrence frequencies of member patterns (available in CPL) that contain the sub-pattern. Accordingly the support counts of all frequent itemsets co-occurring with the focused item 'f' are calculated and stored in the list of frequent itemsets with prefix 'f'.

Similarly the conditional pattern bases of the remaining frequent items are mined one by one.

ABCDEFGHI
BCDEKJ
CEFGH
BCDEM
FCBAHI
FCDI
BCDH
BML
AM
CMB
ABCFHI
ACDFHI

C	10
B	8
D	6
F	6
H	6
A	5
I	5
E	4
M	4

Figure 2a: Transaction database & Header Table

4. Proposed Algortthm For Fp-Tree Mining Input:

a) Transaction database, T

b) Support threshold, ó

Output: List of frequent itemsets with their support counts, FIS.

Uses: : , MFIS to accommodate the list of maximal frequent itemsets co-occurring with a focused frequent item, f.

Algorithm

Step 1. Scan the transaction database T to find the support counts of individual items. Based on the minimum support threshold, σ, insert the frequent 1 -itemsets into FIS.

Step 2. Scan the transaction database again and process each transaction as given below to construct the FP-Tree.

a) Reduce a transaction by removing infrequent items and then re-arrange the frequent items of the transaction in non-descending order of their support count.

b) Construct FP-Tree (a trie-like structure) to accommodate the reduced transactions in the form of shared prefix paths along with a header table to accommodate the frequent items, their support counts and a pointer to the first occurrence of the item in the FP-Tree [**11**]. A node in the FP-Tree has a count (in addition to the name of an item) that indicates the frequency of occurrence of an itemset represented by its path. Nodes representing a reduced transaction are linked with bi-directional parent-child relation.

Multiple occurrences of an item in terms of nodes on different paths are also linked so that all itemsets involving a selected focused frequent item are readily accessible starting from the corresponding entry of the header table.

c) FP-Tree represents the transaction database in a com-pressed form in terms of various patterns/ itemsets and their frequency of occurrences.

Figure 2.(b) depicts the FP-Tree representation of the transaction database given in **Figure 2.(a)** with support threshold of 3.

Step 3. Select a frequent item with the least support count from the header table and refer to it as focused frequent item, 'f'. Extract the local maximal frequent patterns and all frequent itemsets involving 'f' as detailed below.

a) Traverse the FP-Tree upwards from each occurrence of item 'f to find the conditional pattern base of 'f. For the transaction database given in **Figure 2,** the conditional pat-tern base of the frequent item 'H' is found to be {<FDBC:1>, <FBC:2>,<DBC: 1>,<FDC: 1}.

b) Find the co-occurrence frequency of each relevant item

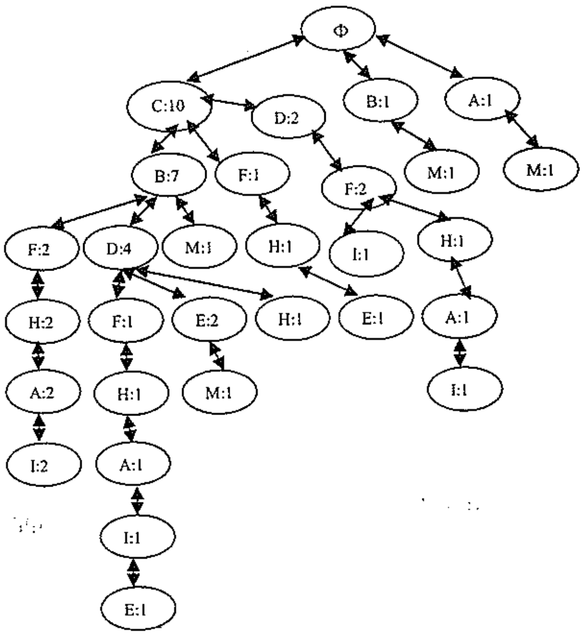


Figure 2b: FP-Tree

and note the frequent items with prefix 'f in the FIS along with their support count.

c) Trim the patterns of the conditional pattern base by re-moving the items with insufficient co-occurrence frequency. If the length of a pattern is one discard it.

d) Enter the reduced/trimmed patterns whose length is more than one, into the successive entries of the array representing Condition Pattern Lattice in the non-ascending order of their length/level as detailed below.

Set the active flag and fill the other fields.

Length of a pattern is the number of items contained in it. Frequency of a pattern is given by the count of node 'f along the path.

i) List of super patterns is NULL for the patterns at the highest level.

ii) While adding the patterns at lower levels, find the super patterns of the present pattern and attach the list of super patterns in terms of their indices to the corresponding entry of the Conditional Pattern Lattice.

Refer to Figure 1 for the Conditional Pattern Lattice of item'H'.

e) Starting from the highest level pattern to the lowest level in the CPL, find the support of a pattern by adding the frequencies of its active super patterns (found in the list attached) to the frequency of itself. If the support is greater than or equal to the support threshold, inactivate the pattern and insert it in the MFIS. During the process of support aggregation, if a pattern is found to have an inactive super pattern, mark the pattern as inactive and continue the process of support aggregation for the next active pattern in the conditional pattern lattice. In our running example, at the end of first pass over the CPL entries the pattern FBC is found to be maximal fre-quent and the active flag of FBC and its sub-pattern FC are marked False. At this stage, MFIS= {FBC}.

f) For each active pattern in the CPL starting with the second pattern referred to as the present pattern, find the lower indexed active patterns which are not included in the list of super patterns of the present pattern. These patterns are referred to as overlapping patterns of the present pattern.

Repeat the following steps on each of the overlapping patterns.

i) Generate a new itemset as an intersection of the present pattern with the selected overlapping pattern.

ii) Find support count of the new itemset as the sum mation of frequencies of all member patterns containing the itemset.

iii) If the support is greater than or equal to ó then insert the new itemset in local MFIS and continue processing with next overlapping pattern. Else take intersection of the new itemset with the next pattern in the CPL and make the resultant as the new itemset.

iv) Repeat the steps (ii) and (iii) on successive patterns in the CPL

In the running example, the itemset DC is formed as an intersection of FDC and DBC and since its support is greater than or equal to ó, DC is inserted into local MFIS. At this stage, MFIS={FBC,DC}.

g) Generate all subsets (size>l) of each of the maximal frequent itemset as and when it is identified and determine their support counts as the summation of the frequencies of all the patterns of CPL which contain them. Insert the itemset with prefix 'f into the FIS. At this stage FIS includes

{<C:10>, <B:8>, <D:6>, <F:6>, <A:5>, <I:5>, <E:4>, <M:4>, <HC:6>, <HB:4>, <HD:3>, <HF:5>, <HFBC:3>, <HFC:3>, <HFB:3 >, <HBC:4 >, <HDC:5>}

h) Re-initialize the list of local maximal frequent itemset, MFIS to NULL. Remove the conditional pattern lattice and its associated array of indices to super patterns.

Step 4. Repeat the Step 3 for mining the conditional pattern bases of all frequent items except the most frequent item. This results in the collection of global frequent itemsets along with their supports stored in FIS.

5. Experimental Results

FP-Growth algorithm is a highly accepted and successful algorithm for extracting frequent itemsets based on the pattern-growth approach. Due to its recursive nature it fails to extract very long patterns from huge transaction databases on simple systems with memory limitations. COFI* is one of the recently developed promising algorithm. It has introduced the concept of Leap Traversal for COFI-Tree mining.

The proposed algorithm is an efficient implementation of Leap Traversal of itemset lattice for extracting local maximal frequent itemsets and there by all frequent itemsets. The proposed algorithm is expected to give better results in terms of memory requirements due to the following reasons.

1) It is non-recursive in nature so it handles long patterns more efficiently compared to recursive mining algorithms.
2) It uses a compact data structure, Conditional Pattern Lattice that accommodates "Contained-in" relationships among patterns which is essential for leap traversal.

FP-Growth algorithm, COFI* algorithm and proposed algorithm are implemented in Java and their performance is tested on a simple PC with Intel Celeron processor of 1.7GHz speed and 256MB RAM. Experimental results are tabulated in Tables 1, 2 and 3 for different datasets namely Chess, Mushroom and Retail databases.

Support(%)	Time in seconds		
	FP-Growth	COFI*	Propos algorith
95	1.5	0.75	0.5
90	4.5	1.25	1
85	33	10.75	9
80	Memory space	121.25	96
75	exception	746	575

Table 1. Chess dataset
# trans=3196 # items=75

Support(%)	Time in seconds		
	FP-Growth	COFI*	Proposed algorithm
90	3	2.55	2
80	3	2.55	2
70	3	2.55	2
60	3	2.55	2
50	3	2.55	2
40	4.5	3.75	3.5
30	21	17.5	17

Table 2. Mushroom dataset
#trans=8124 # items=119

Support(%)	Time in seconds		
	FP-Growth	COFI*	Proposed algorithm
10	15	1.25	1
5	16.75	2.5	2
2	17.58	2.75	2.35
1	41.9.5	37.5	77

Table 3. Retail datase
t# trans=88162 # items= 16469

6. Conclusions

The proposed algorithm adopts divide and conquer strategy for mining a given transaction database in the form of FP-Tree. Patterns representing the co-occurrence of a focused frequent item with its more frequent items are grouped as a conditional pattern base and are accommodated in a new data structure Conditional Pattern Lattice. An effective implementation of Leap Traversal for finding frequent itemsets using Conditional Pattern Lattice is presented. This algorithm is expected to require very less memory space because the conditional pattern lattice for each of the frequent item is built, mined and discarded before the next frequent item is dealt with.

The expectation is rightly supported by the experimental results. Our implementation of Leap Traversal requires much less memory space while handling sparse as well as dense databases. The proposed implementation out performs the state of art methods in view of both time and memory space requirements.

In order to obtain better response time, multiple threads can be employed to handle the conditional pattern base of different frequent items simultaneously.

7. References

[1] R.Agrawal,T.Imielinski, A.Swamy. *Mining association rules between set of items Marge databases.* In Proceed-ings of the ACM SIGMOD Conference on Management of Data, (1993), 207-216.

[2] R.Agrawal, R.Srikant. *Fast algorithms for mining asso-ciation rules.* In Proceedings of the International Confer-ence on Very Large Databases, (September 1994), 487-499.

[3] RAgrawal, R.Mannila, R.Srikant, H.Toivonen, Verkano. *Fast discovery of association rules.* In Advances in Knowledge Discovery and Data Mining, (1996), 307-328.

[4] MAntonie, O.R.Zaine. *Text Document Categorization by Term Association.* In Proceedings of IEEE International Conference on Data Mining, (December 2002), 19-26.

[5] R.J.Bayardo Jr. *Efficiently mining long patterns from databases.* In Proceedings of the ACM SIGMOD Inter-national Conference on Management of Data, ACM Press,(1998),85-93.

[6] B.Goethals. *Survey on frequent pattern mining.* http://www.cs.helsinki.fi/u/goethals/publications/surveys.ps

[7] M.E.Hajj, O.R.Zaiane. *Non-recursive generation of fre-quent k-itemsets from frequent pattern tree representa-tions.* In Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery DaWaK, (September 2003), 371-380.

[8] M.E.Hajj, O.R.Zaiane. *COFI approach for mining fre-quent item-sets revisited.* In Proceedings of International Conference on Data Mining and Knowledge Discovery, (June 2004).

[9] J.Han, J.Pei, Y.Yin. *Mining frequent patterns without can-didate generation.* In Proceedings of ACM SIGMOD In-ternational Conference on Management of Data, (July 2000), 1-12.

[10] J.Han, M.Kamber. *Data Mining: Concepts and Tech-niques,* Morgan Kaufmann /Elsevier Science India, (2002), 225-278.

[11] J.Han, J.Pei, Y.Yin, R.Mao. *Mining frequent patterns with-out candidate generation: A Frequent Pattern Tree ap-proach.* Data Mining and Knowledge Discovery, 8, (2004), 53-87.

[12] B.Lent, A.Swamy, J.Widom. *Clustering Association rules.* In Proceedings of International Conference on Data Engineering ICDM, (April 1997), 220-231.