International Journal for Modern Trends in Science and Technology, 6(8S): 149-155, 2020
Copyright © 2020 International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTSTCIET29
Available online at: http://www.ijmtst.com/ncracse2020.html

Robust Frequent Item set Mining on Massive Data

Dr. B. Indira[1] | M. Surya Teja[2]

[1]Assistant Professor, Department of MCA, CBIT(A), Gandipet, Hyderabad.
[2]Student, MCA III year, CBIT(A), Gandipet, Hyderabad.

To Cite this Article
Dr. B. Indira and M. Surya Teja, "Robust Frequent Item set Mining on Massive Data", *International Journal for Modern Trends in Science and Technology*, 6(8S): 149-155, 2020.

Article Info
Received on 16-July-2020, Revised on 15-August-2020, Accepted on 25-August-2020, Published on 28-August-2020.

ABSTRACT

Frequent item set mining is one of the significant operationswhich returns all item sets in the transaction table, that occur as a subset of specified fraction of the transactions. The existing algorithms fail to compute frequent item sets efficiently on massive data, since they either require multiple-pass scans on the table or construct complex data structures which normally exceed the available memory on massive data. This paper proposes a novel pre computation-based frequent item set mining (PFIM) algorithm to compute the frequent item sets quickly on massive data. PFIM treats the transaction table as two parts: a large old table storing historical data and a relatively small new table storing newly generated data. PFIM first pre- constructs the quasi-frequent item sets on the old table whose supports are above the lower-bound of the practical support level. Given the specified support threshold, PFIM can quickly return the required frequent item sets on the table by utilizing the quasi-frequent item sets. Three pruning rules are presented to reduce the size of the involved candidates. An incremental update strategy is devised to efficiently re-construct the quasi-frequent item sets when the tables are merged. The extensive experimental results, conducted on synthetic and real-life datasets, show that PFIM has a significant advantage over the existing algorithms and runs two orders of magnitude faster than the latest algorithm.

KEYWORDS--*Frequent item sets, Pre-computation based frequent item set mining, quasi-frequent item set, minimum support Threshold*

INTRODUCTION

In practical applications like as data mining, software bug detection, spatiotemporal data analysis and biological analysis we use the important operation Frequent itemset mining which is used to mine the frequent itemsets. In the proposed algorithm given transaction table contains a set of items, frequent itemset mining returns all sets of items whose frequenciesare above a given threshold. The existing frequent itemset mining algorithms can be classified into two groups:candidate-generation-based algorithms and pattern-growth-based algorithms. In frequent itemset mining, the number of the frequent itemsets normally is sensitive to the value of the support threshold which is very small there will be a large number of frequent itemsets and it is difficult for the users to make efficient decisions. if the support threshold is large, there may be no frequent itemsets or the interesting itemsets may be missed. So,a proper support threshold is mandatory for the practical frequent itemset mining and the users often need to perform frequent itemset mining for several times before the satisfactory support threshold is determined. the given transaction table, data set can be divided into two parts: the much larger old data set storing the historical data, and the relatively small new data set storing the newly generated data. This paper proposes precomputation-based PFIM algorithm to compute frequent itemsets on massive data.Three pruning rules are proposed in this paper to reduce the number of the candidate frequent itemsets.An incremental update strategy is used to re-construct the quasi-frequent itemsets quickly.

SCOPE AND OBJECTIVE

This paper proposes a novel pre-computation based PFIM (Precomputation-based Frequent Itemset Mining) algorithm to compute frequent itemsets on massive data efficiently. Three pruning rules are proposed in this paper to reduce the number of the candidate frequent itemsets. An incremental update strategy is devised to reconstruct the quasi-frequent itemsets quickly. The experimental results show that PFIM has a significant advantage over the existing algorithms. we want to devise a highly efficient algorithm to mine the frequent itemsets on massive data quickly. Some of the existing algorithms, such as FP-tree-based methods or vertical representation-based methods, re use the work that has already been done previously in the current frequent itemset mining, so they can discover frequent itemsets faster.

LITERATURE REVIEW

Rakesh et.al., [1] presentedtheir perspective of data base mining as the confluence of machine learning techniquesandtheperformanceemphasisofdatabas etechnology.Theydescribed the threeclasses of database mining problems involving classification, associations, and sequences, and argued that these problems can be uniformly viewed as requiring discovery of rules embedded in massive data. They described a model and some basic operations for the process of rule discovery.Theyshowedhowthedatabaseminingprob lemscan bemappedtothismodeland how they can be solved by using the proposed basic operations. And they gave an example of an algorithm for classification obtained by combining the basic rule discovery operations. This algorithm not only is efficient in discovering classification rules but also has accuracy comparable to ID3, one of the current best classifiers.

"Clustering by Pattern Similarity in Large Data Sets" by Haixun Wang et.al., [2], defined Clustering as the process of grouping a set of objects into classes of similar objects. Although definitions of similarity vary from one clustering model to another, in most of these models the concept of similarity is based on distances, e.g., Euclidean distance or cosine distance. In other words, similar objects are required to have close values on at least a set of dimensions. In their paper, they explored a more general type of similarity. Under the pCluster model they proposed that two objects are similar if they exhibit a coherent pattern on a subset of dimensions. For instance, in DNA microarray analysis, the expression levels of two genes may rise and fall synchronously in response to a set of environmental stimuli. Although the magnitude of their expression levels may not be close, the patterns they exhibit can be very much alike. Discovery of such clusters of genes is essential in revealing significant connections in gene regulatory networks. Clustering has been extensively studied in many areas, including statistics, machine learning, pattern recognition, and image processing. Recent eff orts in data mining have focused on methods for efficient and effective cluster analysis in large databases. Much active research has been devoted to areas such as the scalability of clustering methods and the techniques for high- dimensional clustering.Clustering in high dimensional spaces is often problematic as theoretical results questioned the meaning of closest matching in high dimensional spaces. Recent research work has focused on discovering clusters embedded in the subspaces of a high dimensional data set. This problem is known as subspace clustering. In their paper, they explored a more general type of subspace clustering which uses pattern similarity to measure the distance between two objects.They applied the pCluster algorithm on the yeast gene microarray. And showed that a majority of maximum dimension sets are eliminated after the 1st and 2nd round. The overall running time is around 200-300 seconds, depending on the user parameters. They showed that their algorithm has performance advantage over the bicluster algorithm [3], as it takes roughly 300-400 seconds for the bicluster algorithm to find a single cluster.

Mutlu Yüksel Avcilar and Emre Yakut[4] in their paper "Association Rules in Data Mining: An Application on a Clothing and Accessory Specialty Store" discussed about retailers who provide important functions that increase the value of the products and services they sell to consumers. Retailers value creating functions are providing assortment of products and services: breaking bulk, holding inventory, and providing services. For a long time, retail store managers have been interested in learning about within and cross-category purchase behavior of their customers, since valuable insights for designing marketing and/or targeted cross-selling programs can be derived. Especially, parallel to the development of information processing and communication technologies, it has become possible to transfer customers shopping information into databases with the help of barcode technology. Data mining is the technique presenting significant and useful information using of lots of data. Association rule mining is realized by using market basket analysis to discover relationships among items purchased by customers in transaction databases. In this study, association rules were estimated by using market basket analysis and taking support, confidence and lift measures into consideration. In the process of analysis, by using of data belonging to the year of 2012 from a clothing and accessory specialty store operating in the province of Osmaniye, a set of data related to 42,390 sales transactions including 9,000 different product kinds in 35 different product categories (SKU) were used. Analyses were carried out with the help of SPSS Clementine packet program and hence 25,470 rules were determined.Retailing is the set of business activities that adds value to the products and services sold to consumers for their personal or family use. Retailers are business that manages to satisfy the consumers' needs and wants by offering the right product assortment, at the right quantity, at the reasonable price, at the desired time and place. Thus, they add value to products and services. In their study, specialty store customers' shopping information were analyzed by using association rules mining with Apriori algorithm. As a result of the analyses, strong and useful association rules were determined between the product groups with regard to understanding what kind of purchase behavior customers exhibit within a certain shopping visit from both in-category and different product categories for the specialty store in question. By utilizing the association rules which are discovered as a result of the analyses, the retail store manager will be able to develop and apply effective marketing and sales promotion strategies.

"Direct Discriminative Pattern Mining for Effective Classification" by Hong Chenget.al.,[5] proposed a direct discriminative pattern mining approach, DDPMine, to tackle the efficiency issue arising from the two-step approach: frequent pattern (or classification rule) mining followed by feature selection (or rule ranking). However, this two-step process could be computationally expensive, especially when the problem scale is large or the minimum support is low. It was observed that frequent pattern mining usually produces a huge number of "patterns" that could not only slow down the mining process but also make feature selection hard to complete. DDPMine performs a branch-and- bound search for directly mining discriminative patterns without generating the complete pattern set. Instead of selecting best patterns in a batch, they introduced a "feature-centered" mining approach that generates discriminative patterns sequentially on a progressively shrinking FP-tree by incrementally eliminating training instances. The instance elimination effectively reduces the problem size iteratively and expedites the mining process. Empirical results show that DDPMine achieves orders of magnitude speedup without any downgrade of classification accuracy. It outperforms the state-of-the-art associative classification methods in terms of both accuracy and efficiency.Frequent pattern-based classification has been explored in recent years and its power was demonstrated by multiple studies in several domains, including associative classification on categorical data, where a classifier is built based on high-support, high confidence association rules; and frequent pattern-based classification on text or data with complex structures such as sequences and graphs, where discriminative frequent patterns are taken as features to build high quality classifiers. A frequent itemset (pattern) is a set of items that occur in a dataset no less than a user-specified minimum support (min sup). Frequent patterns have been explored widely in classification tasks. These studies achieve promising classification accuracy and demonstrate the success of frequent patterns (or association rules) in classification.

Jiawei Han et.al.,[6] in their paper "Frequent pattern mining: current status and future directions" discussed about various basic mining technologies and their extensions. Frequent pattern mining has been focused 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 article, they provided a brief overview of the current status of frequent pattern mining and discussed a few promising research directions. They believed that frequent pattern mining research has substantially broadened the scope of data analysis and will have deep impact on data mining methodologies and applications in the long run. However, they opinioned that there are still some challenging research issues that need to be solved before frequent pattern mining.

ALGORITHM

The five main steps used in the algorithm are:

a. Intuitive Idea
b. Pre-computation Operation
c. Basic Process
d. Pruning Operation
e. Update Operation

a.Intuitive Idea:

The number of frequent itemsets is very sensitive to the value of minsup. If the value of minsup is too small, the number of frequent itemsets will be so large that the users can become overwhelmed with too many results and it is difficult for users to find the really useful information from them. Therefore, in this paper, we assume that there exists a lower-bound for the value of minsup in practical applications. The lower-bound is denoted by ω. The value of ω can be determined by some domain experts, or the lowest value of the support used in the past frequent itemset mining. On massive data, the existing algorithms often cannot meet the users' requirement, they either need to scan the table multiple times, or need a complex data structure and a high memory consumption.

b.Pre-Computation Operation:

Pre-computation operation generate the required item sets on the large old transaction table, $T_{0,}$ whose supports are no less than ω, referred to as quasi-frequent item sets, different from the frequent item sets with the support threshold minsup specified by users. Let tn_o be the number of transactions in T_0 and tn_1 be the number of transactions in T_1. The process of pre-computing the quasi-frequent item sets consists of two stages: candidate generation and result refinement.

In candidate generation, the transactions in $T_0$are retrieved sequentially and maintained in an in-memory buffer BUF, whose size is set according to the size of the allocated memory. If BUF is full, compute the local quasi-frequent itemsets in BUF by the current vertical frequent itemset mining algorithms and keep in a file. Then empty the BUF and continue the sequential scan for the next iteration.

In result refinement, first read all the local quasi-frequent itemsets into the memory. Then perform sequential scan on T_0 to compute support count, i.e., the absolute occurrence number, for each local quasi-frequent itemset. Then the local quasi-frequent itemsets whose support counts are no less than $[tn_o \times \omega]$ are maintained as the global quasi-frequent itemsets, which are stored in a file F_{qf} and will be used for frequent itemset mining.

c. Basic Process:

- **Sequential scan on new table**:

PFIM first retrieves the transactions in T_1. $\forall\ t_1 \in T_1$, let t1 be the currently retrieved transaction. $\forall i \in t_1$, i is an item in t_1, we increase the count of i by 1 (initial value is 0). Due to its relatively small size of T_1 and the simple computation, this sequential scan can be executed quickly. We use an array cnt1 to keep these counts. $\forall i \in U$, $cnt_1[i]$ is the count of item i in T_1, $cnt_1[i] = 0$ if i does not appear in any transaction in T_1. The value of mas_1 is the maximum support count for all items in T_1.

- **Sequential scan on quasi-frequent itemsets**

Then, PFIM begins to retrieve F_{qf} .$\forall t \in F_{qf}$, let t be the currently retrieved quasi-frequent itemset in F_{qf}. The quasi-frequent item sets in F_{qf} can be divided into three classes:

(1) definitely belonging to the frequent item sets,

(2) definitely not belonging to the frequent item sets,

(3) possibly belonging to the frequent item sets.

Given $|t.IS| = 1$ and t.IS = {i}, if $t.SUP+cnt1[i] \geq [n\times minsup]$, t.IS is frequent, otherwise, t.IS is not frequent. Given $|t.IS| \geq 2$, if $t.SUP \geq [n\times minsup]$, t.IS is frequent obviously, otherwise, if $t.SUP+mas_1< [n\times minsup]$, t.IS certainly not a frequent itemset. In other cases, t may be a frequent itemset, depending on the transactions in T_1, and PFIM maintains t in a set ST_{CAD}.

- **Increase supports for itemsets**

When all quasi-frequent item sets are retrieved already, PFIM needs to increase the support counts of quasi-frequent item sets in ST_{CAD} by their counts in T1, this can be done by a sequential scan on T_1., PFIM retrieves B transactions from T_1. For the current iteration, the transactions maintained in memory are transformed into vertical representation, i.e. each item is associated with the list of identifiers (TID) of transactions containing the item. $\forall t \in ST_{CAD}$ and t.IS = $\{i_{j1}, i_{j2}, \ldots, i_{ja}\}$, the number of transactions in BUF1 containing t.IS is $|Ta = 1\ i_{jb}\ .tlist|$, where i_{jb}.tlist is the TID list corresponding to the item i_{jb}.

- **Compute new frequent itemsets in new table if required**

F_{qf} contains the quasi-frequent item sets whose support counts in T_0 is at least $[t_{no} \times \omega]$. Therefore, for any itemset which is not contained in F_{qf}, its support count in T_0 is at most $([t_{no} \times \omega]-1)$. The value of mas_1, the maximum count for all items in T1, acquired already. This means that the maximum support count of any itemset in T_1 cannot exceed mas1. If $([t_{no} \times \omega]-1) + mas_1 < [n \times minsup]$, there are no new frequent item sets generated from T_1 that are not contained in F_{qf}, and PFIM does not need to compute new item sets in T_1. Conversely, if $([t_{no} \times \omega]-1) + mas_1 < [n \times minsup]$, PFIM still needs to discover required frequent item sets from T_1 whose support counts in T_1 are at least $[n \times minsup] - ([t_{no} \times \omega]-1)$. This corresponds to a new frequent itemset on T_1 with the specified support threshold minsup1 = $[n \times minsup] - ([t_{no} \times \omega]-1)$.

d. PRUNING Operation

PFIM can reuse the pre-computation result of T_0 and reduce the execution cost significantly. In this part, we discuss how to improve PFIM further to speed up its execution by pruning operation.

- **PRUNING operation in step2**

One main part of the cost in PFIM is to compute the support counts of the item sets of ST_{CAD} in T_1. Therefore, if we can reduce the number of item sets in ST_{CAD} in step 2, the counting cost in T_1 can be decreased. Use the maximum count mas_1 of the single item in T_1 to determine the support count range of the possible frequent item sets. Obviously, if we can narrow down the support count range, the size of ST_{CAD} can be reduced. As described in the process of step 2, PFIM can determine directly whether the quasi-frequent 1-itemsets in F_{qf} are frequent item sets. Therefore, ST_{CAD} only needs to maintain the quasi-frequent item sets which contain at least two items. At the end of step2, PFIM maintains the possible frequent item sets in ST_{CAD}

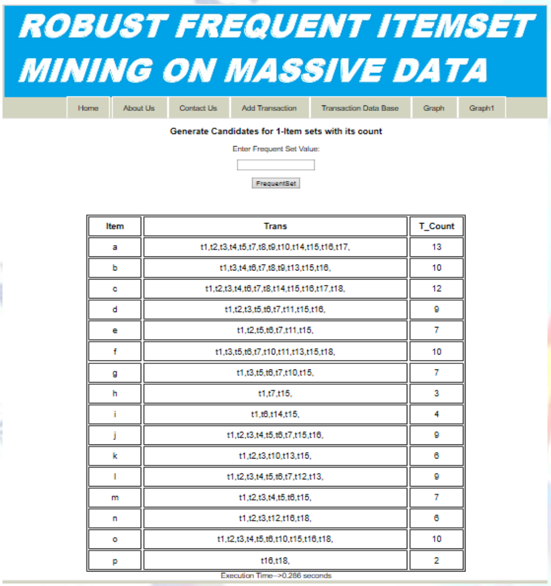
- **PRUNING operation in step4**

Another possible main part of the cost in PFIM is the operation to generate the new frequent item sets in T_1 if required, i.e. the step 4. If $minsup_1$ is low, the computation on T_1 and T_O can take a relatively long execution time, this will affect the high efficiency of PFIM. Therefore, in step 4, we also need a pruning operation to speed up its execution. The pruning operation is performed on the generation of the local frequent item sets in T_1 because, if the number of local frequent item sets is decreased, the cost in support counting can be reduced also. PFIM first determines the local frequent 1-itemsets LF_1, Given the local frequent k-itemsets $LF_{k,1}(k \geq 1)$ the pruning operation is executed. Itemsets in $LF_{k,1}$ is sorted in the lexicographical order. The itemsets in $LF_{k,1}$is first grouped according to the first $(k - 1)$ items, i.e. each group contains the itemsets sharing the same first $(k-1)$ items. After the grouping operation, assume that we can obtain h groups $G_1, G_2, \ldots, G_h$. The pruning operation in step 4 has two rules: pruning rule 2 (PR2) and pruning rule 3 (PR3). PR2: $\forall 1 \leq b \leq h$, if G_b contains only one k-itemset, G_b can be removed. PR3: $\forall 1 \leq b \leq h$, if the itemset is contained in F_{qf}, G_b can be removed. The intuition behind PR2 is that, if G_b contains only one k-itemset, $(k + 1)$-itemsets cannot be generated from G_b. The intuition behind PR3 is that, if the itemset is contained in F_{qf}, the possible itemset has already been considered in the previous steps, we do not need to consider it again.

e.UPDATE operation:

When the size of T_1 reaches a certain threshold, for example, 5% of the size of T_O, the transactions in T_1 and T_O are merged. At this point, the quasi-frequent item sets in F_{qf} needs to be updated also. But the total re-construction can be expensive. Therefore, in this paper, a new an incremental update strategy is proposed, which utilizes the existing information computed already, to speed up the update operation.

The goal of the update operation is to generate the quasi-frequent item sets on T given the support level ω. The local quasi-frequent item sets of T_O are kept in $F_{qf,\ O}$, First we need to add the occurrences of the local quasi-frequent item sets of $F_{qf,\ O}$ in T_1. Then, the local quasi-frequent itemsets in $F_{qf,\ O}$, are written into the new file F_{qf}, the local quasi-frequent itemsets of T_1 are kept in $F_{qf,1}$. In order to avoid duplicate computation, the local quasi-frequent itemsets in $F_{qf,1}$, which have been contained in $F_{qf,O}$, are removed before the support counting. The support counts of the local quasi-frequent itemsets in $F_{qf,1}$ are calculated by another scan on T_O and T_1. The number of the itemsets in $F_{qf,1}$ can be reduced significantly by the containment checking in $F_{qf,O}$. The computation cost of adding the support counts of itemsets in T_1 and T_O can be lowered accordingly. Therefore, the incremental update strategy can run much faster than the total re-construction strategy, which also is verified in the experiments.



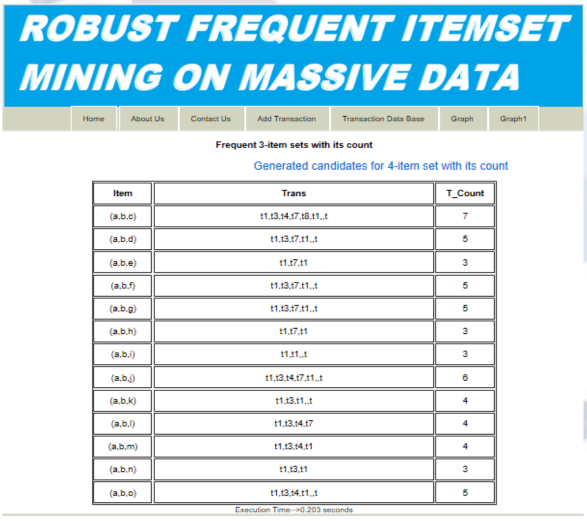
Generate Candidates for 1-Item sets with its count

Enter Frequent Set Value:

Item	Trans	T_Count
a	t1,t2,t3,t4,t5,t7,t8,t9,t10,t14,t15,t16,t17,	13
b	t1,t3,t4,t6,t7,t8,t9,t13,t15,t16,	10
c	t1,t2,t3,t4,t6,t7,t8,t14,t15,t16,t17,t18,	12
d	t1,t2,t3,t5,t6,t7,t11,t15,t16,	9
e	t1,t2,t5,t6,t7,t11,t15,	7
f	t1,t3,t5,t6,t7,t10,t11,t13,t15,t18,	10
g	t1,t3,t5,t6,t7,t10,t15,	7
h	t1,t7,t15,	3
i	t1,t6,t14,t15,	4
j	t1,t2,t3,t4,t5,t6,t7,t15,t16,	9
k	t1,t2,t3,t10,t13,t15,	6
l	t1,t2,t3,t4,t5,t6,t7,t12,t13,	9
m	t1,t2,t3,t4,t5,t6,t15,	7
n	t1,t2,t3,t12,t16,t18,	6
o	t1,t2,t3,t4,t5,t6,t10,t15,t16,t18,	10
p	t16,t18,	2

Execution Time-->0.286 seconds

Fig.1: Generating candidate with 1 itemset



Frequent 3-item sets with its count

Generated candidates for 4-item set with its count

Item	Trans	T_Count
(a,b,c)	t1,t3,t4,t7,t8,t1,,t	7
(a,b,d)	t1,t3,t7,t1,,t	5
(a,b,e)	t1,t7,t1	3
(a,b,f)	t1,t3,t7,t1,,t	5
(a,b,g)	t1,t3,t7,t1,,t	5
(a,b,h)	t1,t7,t1	3
(a,b,i)	t1,t1,,t	3
(a,b,j)	t1,t3,t4,t7,t1,,t	6
(a,b,k)	t1,t3,t1,,t	4
(a,b,l)	t1,t3,t4,t7	4
(a,b,m)	t1,t3,t4,t1	4
(a,b,n)	t1,t3,t1	3
(a,b,o)	t1,t3,t4,t1,,t	5

Execution Time-->0.203 seconds

Fig.2: Generating candidate for 4 item sets

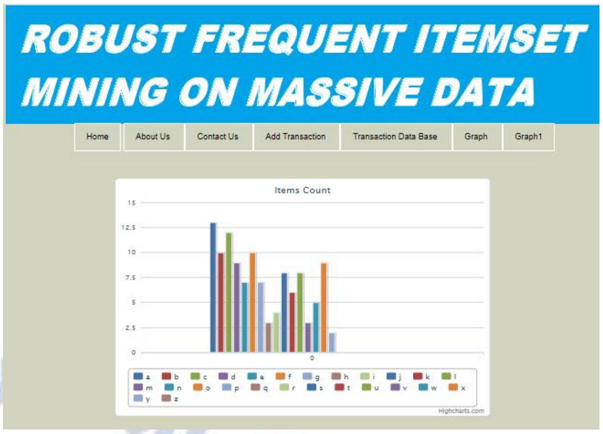


Fig.3: Graph showing itemset based on count

CONCLUSION AND FUTURE SCOPE

The problem of computing frequent item sets on massive data is considered in this paper. It is found that the existing algorithms cannot perform frequent item set mining on massive data efficiently. This paper utilizes the idea of reusing the work done previously and devises a pre computation based PFIM algorithm to quickly acquire the frequent item sets on massive data. The transaction table consists of two part: the large old table and the relatively small new table. By the quasi-frequent item sets pre-computed on the old table, PFIM can report the frequent item sets on massive data efficiently. Three pruning rules are proposed in this paper to speed up the execution of PFIM. The incremental update strategy is presented to re-construct the quasi-frequent item sets quickly when merging the old table and the new table. The extensive experimental results show that PFIM has a significant performance advantage over the existing algorithms. Hence the proposed algorithm is better for finding frequent itemset mining in massive databases.

REFERENCES

[1] Rakesh Agrawal Tomasz Imielinski Arun Swami, "Database Mining: A Performance Perspective", IBM Almaden Research Center 650 Harry Road San Jose, CA 95120-6099.

[2] Haixun Wang Wei Wang Jiong Yang Philip S. Yu "Clustering by Pattern Similarity in Large Data Sets", IBM T. J. Watson Research Center 30 Saw Mill River Road Hawthorne, NY 10532 {haixun, ww1, jiyang, psyu}@us.ibm.com

[3] Y. Cheng and G. Church." Biclustering of expression data", In Proc. of 8th International Conference on Intelligent System for Molecular Biology, 2000.

[4] Mutlu Yüksel Avcilar and Emre Yakut in their paper "Association Rules in Data Mining: An Application on a Clothing and Accessory Specialty Store", Canadian Social Science Vol. 10, No. 3, 2014, pp. 75-83 DOI:10.3968/4541, ISSN 1712-8056[Print] ISSN 1923-6697[Online]

[5] Hong Cheng, Xifeng Yan , Jiawei Han, Philip S. Yu , "Direct Discriminative Pattern Mining for Effective Classification",

#University of Illinois at Urbana-Champaign Urbana, IL, USA 1hcheng3@uiuc.edu 3hanj@cs.uiuc.edu ∗IBM T. J. Watson Research Center Hawthorne, NY, USA 2xifengyan@us.ibm.com 4psyu@us.ibm.com

[6] Jiawei Han , Hong Cheng · Dong Xin · Xifeng Yan , "Frequent pattern mining: current status and future directions", Data Min Knowl Disc (2007) 15:55–86 DOI 10.1007/s10618-006-0059-1