ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

An Efficient Algorithm for Finding 1-itemsets and Their Respective Counts in Frequent Pattern Mining Using Red-Black Tree

Kumari Priyanka Sinha, Rajeshwar Puran
Central University of Bihar

Abstract*— Data mining deals with turning data into useful information and knowledge. It relies heavily on the concept of association rule mining, which is one of the most extensively researched topics of data mining. Association rule mining, in turn leads to the concept of frequent pattern mining. Frequent pattern mining seeks to extract interesting patterns among different sets of items in databases. There are scores of widely used algorithms such as Apriori algorithm and FP-tree algorithm for frequent pattern mining. Almost all these algorithms require finding 1- item sets and their corresponding counts. This task in itself snowballs into a tedious task considering the hugeness of modern databases available. No explicit strategy has been outlined in these algorithms to perform the aforesaid task. In this paper, we have proposed a simple and easy-to-implement algorithm to perform this task. This algorithm employs the concept of red-black trees. This data structure is readily available in the form of map in C++ and Tree Map in Java , thus requires no extra effort in implementation. This algorithm can be easily embedded into any of the existing algorithms aimed at frequent pattern mining. The experiments carried out by us and results so obtained testify our algorithm.

Index Terms— **Association rule mining; Frequent pattern Mining; Red-black tree; 1-itemsets.**

I. INTRODUCTION

This section starts with a brief discussion on frequent pattern mining and association rule mining. It also provides a brief literature survey on them. Later, it discusses red-black trees.

A. Frequent pattern Mining [1]

Association rule mining [2] is one of the key tasks of data mining. It is basically an iterative process that works on a voluminous data to extract valid and meaningful rules. In order to accomplish its objective, it explores interesting relationships among item sets[3] in a given database.

Let I= {, ,…} be a Item set, D be a set of database transactions where each transaction T is a set of items such that T⊆I. Each transaction is associated with an identifier called transaction Id (TID). Suppose A and B be two item sets. T is said to be containing A If and only if A⊆T. An association rule is an implication of the form A→ B, where A⊂I, B⊂I and A∩B=ϕ. The rule holds in the transaction set D with support s where s denotes the percentage of transactions in D that contains A∪B. This is equal to the probability P(A∪B). The rule A→ B has the confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B.

Support (A → B) = P (A ∪ B) . (1)

Confidence (A → B) = P (B|A) . (2)

The task of association rule mining can be divided into two subtasks [4]. The first task is to find out those item sets whose frequency of occurrences exceeds or equals minimum support count. The item sets extracted at this stage are known as frequent item sets or patterns. Hence, this stage is known as frequent pattern mining. The second task in association rule mining is to generate strong association rules from the frequent item sets. Frequent pattern mining is one of the most active research areas of data mining. It owes its emergence from [5], which introduced an algorithm known as AIS algorithm which was able to generate all significant association rules between items in the database. This algorithm was followed by a huge pile of algorithms [5, 8, 9, 10, 11, 12, 13]. Apriori algorithm [6] is one of the most widely used algorithms for frequent pattern mining. Frequent item sets at a level are used to construct the frequent item sets at the other level. However this algorithm needs multiple scans of the database adversely affecting its speed in case of huge databases. FP-tree algorithm [7] is yet another widely used algorithm in the field of frequent pattern mining. Unlike Apriori algorithm [6], it does not require candidate item sets. It is basically a divide-and-conquer approach converting the problem of mining into finding smaller patterns recursively. It uses a pattern fragment growth method to avoid the costly process of candidate generation and testing used by Apriori. However when the database is relatively large, the task of constructing a main

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

memory-based FP-tree becomes too costly to handle. In past 4-5 years there has been a huge leap in the research on association rule mining and frequent pattern mining. Some of the most recent works include [16, 17, 18, 19, 20].

B. Red-Black Tree

Red-black tree is a self-balancing binary search tree with one extra bit of storage per node [1]. This data structure was introduced by Rudolf Bayer in 1972. Although complex data structure, it has a one of the best worst case running time for dynamic set operations. It takes O(logn) time for search, insertion and deletion , where n is the total number of elements in the tree. Each node of this tree has a color either red or black associated with it. A red-black tree, like all other binary search trees, allows in-order traversal, Left-Root-Right, of their elements.

In addition to the ordinary requirements imposed on the binary search trees [15], a red-black tree has the following additional requirements associated with it [1]. Every node is either red or black. The root is black. Every leaf (NIL) is black. If a node is red, then both its children are black. For each node, all paths form the node to descendent leaves contains the same number of black nodes.

These constraints instigate an important characteristic of red-black trees that the longest path from the root to any leaf is at most two times as long as the shortest path from the root to any other leaf in that tree, thus rendering the tree roughly balanced. If there are n nodes in the tree, the height of the tree is at most 2log(n+1). Since operations such as insertion, deletion and searching etc. require a time proportional to the height of the tree, this upper bound on the height allows the red-black trees to be extremely efficient in terms of worst-case complexity, unlike ordinary binary search trees [15] where the worst case complexity is O(n).

The following picture depicts a typical red-black tree. adopters indicate that security and privacy is the primary concern for it adoption.

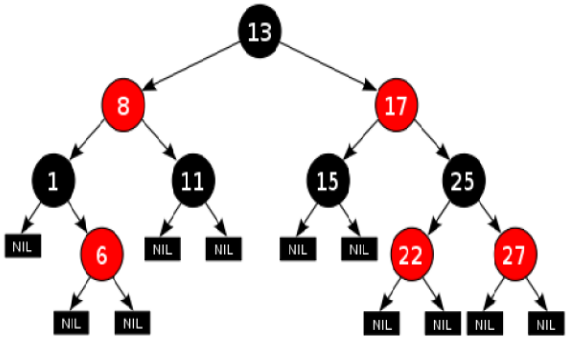

Fig. 1. Red-Black Tree

II. PROBLEM DESCRIPTION

A set of items that appears frequently together in a transaction dataset is known as a frequent item set. Moreover, the item set containing 1 item is called a 1-itemset.Let us take an example. The table below is a transaction database D containing 9 transactions i.e. |D| = 9. There are many algorithms available for frequent pattern mining [5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20]. Apriori algorithm [6] and FP-tree algorithm [7] are the classic examples. Almost all of these algorithms generate 1- item sets and their corresponding counts at some stage in their implementation. This task in itself is a time- demanding task considering the enormity of modern databases. However, no explicit strategy has been suggested in the above-mentioned algorithms to perform this task.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

Table 1. TID and Itemset

TID	List of Itemset
T_1	i_1, i_2, i_5
T_2	i_2, i_4
T_3	i_2, i_3
T_4	i_1, i_2, i_4
T_5	i_1, i_3
T_6	i_2, i_3
T_7	i_1, i_3
T_8	i_1, i_2, i_3, i_5
T_9	i_1, i_2, i_3

Table 2. 1- Itemsets and count

List of Itemset	Count
i_1	6
i_2	7
i_3	6
i_4	2
i_5	2

There are many algorithms available for frequent pattern mining [5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 20]. Apriori algorithm [6] and FP-tree algorithm [7] are the classic examples. Almost all of these algorithms generate 1-item sets and their corresponding counts at some stage in their implementation. This task in itself is a time-demanding task considering the enormity of modern databases. However, no explicit strategy has been suggested in the above-mentioned algorithms to perform this task. In this paper, we have proposed a simple and easy-to-implement algorithm to address this problem.

III. PROPOSED ALGORITHM

A. Pseudo code of the proposed algorithm

Step1: Instantiate a red-black tree with no element (It can be a map in C++ or a Tree Map in Java. One can also instantiate his own implementation of a red-black tree, although it may require a great deal of work).
Step 2: Scan the whole database. For each element of the database, loop through step 3.
Step 3: Insert the element in the instantiated red-black tree. If the element is not there in the tree, it will be inserted in the tree as a key with the corresponding value automatically initialized by 1. If the element already exists in the tree, its corresponding value will be incremented by 1.
Step 4: Extract each key and their corresponding values of the red-black tree. These will be the 1-itemsets and their corresponding count respectively.
The looping through the whole database takes O(n) time, where n is the total number of elements in the database. Also for each element of the database, it takes O(logn) time to insert the element into the red-black tree. Thus the overall time complexity of the proposed algorithm comes out to be O(nlogn). Had an array, which is a widely used and a natural data structure, been used in lieu of red-black tree, the time complexity would have been as high as O().

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

IV.EXPERIMENTS AND ANALYSIS

To affirm the time-efficiency of our algorithm, we have carried out experiments wherein we have compared the scalabilities of traditional FP-tree algorithm [7] and the FP-tree algorithm armed with our algorithm. For this purpose, we have used the connect-4 dataset obtained from UCI directory [14]. It is a multivariate and spatial dataset. It contains 67557 transactions and 42 attributes. The attributes are of categorical type. We have used a machine possessing 1 GB main memory and a 1.83 GHz dual core processor with windows XP service pack 2 as the operating system. It is noteworthy that the runtime mentioned here denotes the total execution time i.e. the time elapsed between feeding input to the system and obtaining output form the system.

In order to test the efficiency of our algorithm, we choose FP-tree algorithm as a testing algorithm. To compare traditional FP-tree algorithm and the FP-tree algorithm employing our algorithm, we have coded both the algorithms using Java programming language. We have chosen this language owing to its unique feature of portability. The codes prepared by us, hence, can be run on any operating system, be it windows XP, Vista, Fedora, Ubuntu or Macintosh. In the traditional FP-tree algorithm, we have used arrays to hold 1-itemsets, whereas our algorithm utilizes red-black trees as stated earlier. The scalability of traditional FP-tree algorithm and the FP-tree algorithm using our algorithm, as the support threshold varies from 3.5% to 0.5%, have been shown hereunder.

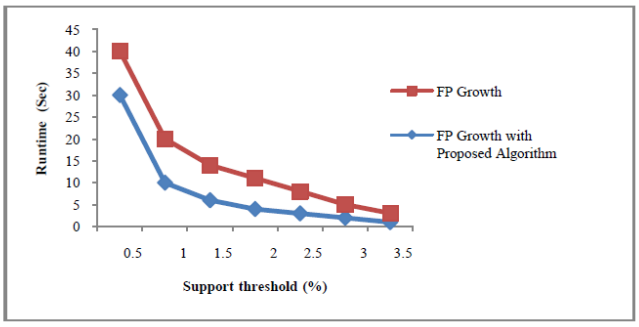

Fig. 2. Comparison between the performances of FG-Growth and FG-Growth with proposed algorithm on the basis of minimum support threshold.

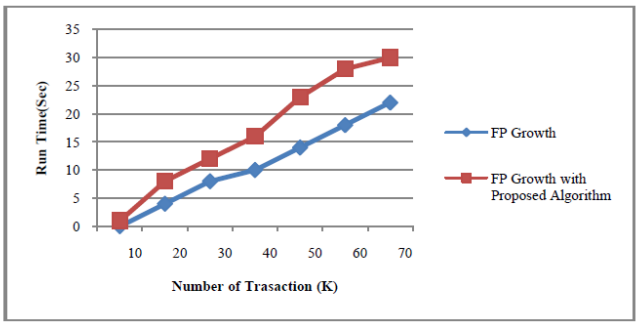

Fig. 3. Comparison between the performances of FG-Growth and FG-Growth with proposed algorithm on the basis of number of transactions

Next, we analyze the scalability of both the versions of FP-tree algorithm with respect to the number of transactions. We vary the number of transactions from 10K to 67K. The result is depicted in the following graph. It is obvious from the above two graphs that our algorithm can speed up any frequent pattern mining algorithm. Also it takes almost no time to embed our algorithm into any of the existing algorithm.

V. CONCLUSION AND FUTURE WORK

In the algorithms aimed at frequent pattern mining, extracting 1-itemsets and their corresponding counts is a computationally demanding task in case of huge databases. In this paper, we attempt to propose a simple and easy-to-implement algorithm to deal with this task. This algorithm employs red-black trees in its implementation. This data structure is readily available in the form of map in C++ and Tree Map in Java. The proposed algorithm works in O(nlogn) time, where n is the total number of elements in the database. This is a big achievement over the naïve method which is expected to wok is O(n2) time. This algorithm can be easily embedded into any of the existing algorithms aimed at association rule mining in order to extract 1-itemsets and their corresponding counts.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

We have carried out experiments to verify our assertion, wherein we have compared the scalabilities of FP-tree algorithm and the FP-tree algorithm using our algorithm. The results so obtained bolster our algorithm.

Our algorithm can generate only 1-itemsets and their corresponding count. However, some algorithms, such as Apriori algorithm, need k-item sets where k can be any number greater than or equal to 1 depending on the data under consideration. Our algorithm cannot extend too much help to such algorithms. Thus, further research need to be carried out in order to dig out efficient algorithms to generate those item sets as well.

REFERENCES

[1] BALDAN, P., CORRADINI, A., ESPARZA, J., HEINDEL, T., KONIG, B., KOZIOURA, V.: VERIFYING RED-BLACK TREES. IN: PROCEEDING OF COSMICAH (2005).

[2] Schildt, H.: C++: The Complete Reference. 4th edition, McGraw-Hill, Berkeley (2003).

[3] Schildt, H.: Java: The Complete Reference. 7th edition, McGraw-Hill, Berkeley (2007).

[4] Han, J., Kamber,M.: Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2006).

[5] Agrawal, R., Imielinski, T., Swami, A. N.: Mining association rules between sets of items in large datasets, In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 207-216 (1993).

[6] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th international conference. Very Large Data Bases, pp. 487-499 (1994).

[7] Han, J., Pei, J.: Mining frequent patterns by pattern-growth: methodology and implications, ACM SIGKDD Explorations Newsletter 2, 2, 14-20 (2000).

[8] Das, A., Ng, W.-K.,Woon, Y.-K.: Rapid association rule mining. In: Proceedings of the tenth international conference on Information and knowledge management. ACM Press, pp. 474-481(2001).

[9] Psaila, G., Lanzi, P. L.: Hierarchy-based mining of association rules in data warehouses, In: Proceedings of the ACM symposium on Applied computing .ACM Press, pp. 307-312 (2000).

[10] Wu, X., Zhang, C., Zhang, S.: Efficient Mining of Both Positive and Negative Association Rules, ACM Transactions on Information Systems, Vol. 22, No. 3, pp. 381–405 (2004).

[11] Wang, C., Tjortjis, C.: PRICES: An Efficient Algorithm for Mining Association Rules. Lecture Notes in Computer Science, Volume 3177, pp. 352 - 358 (2004).

[12] Yuan, Y., Huang, T.: A Matrix Algorithm for Mining Association Rules. Lecture Notes in Computer Science, Volume 3644, pp. 370 - 379 Sep (2005).

[13] Tang, P., Turkia, M.: Parallelizing frequent itemset mining with FP-trees, Technical Report, Department of Computer Science, University of Arkansas at Little Rock, (2005).

[14] UCI Machine Learning Repository: Connect Data Set, HTTP://ARCHIVE.ICS.UCI.EDU/ML/DATASETS/CONNECT-4.

[15] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.: Introduction to Algorithms. 2nd edition, MIT Press, McGraw-Hill, Cambridge (2001).

[16] Moreno, M. N., Ramos, I., García, F. J., Toro, M.: An association rule mining method for estimating the impact of project management policies on software quality, development time and effort. Expert Systems with Applications, pp. 522–529 (2008).

[17] Alatas, B., Akin, E., Karci, A.: MODENAR: Multi-objective differential evolution. algorithm for mining numeric association rules. Applied Soft Computing, pp. 646 – 656(2008).

[18] Claude, T. J., Sheng, Y , chuang, L., Kaia, X. : An optimal class association rule algorithm. In: Proceedings of the fourth International Symposium, ISICA, Huangshi, pp. 344-350 (2009).

[19] M.Anandhavalli , M.K.Ghose , K.Gauthaman : Fast Association Rule Mining Algorithm for Spatial Gene Expression Data ,Global Journal of Computer Science and Technology ,Volume 9 issue 5, pp. 36-40 (2010).

[20] Sekhavat, Y. A., Gholamian, M. R., Fathian, M., Alizadeh, S.: Mining important association rules based on the RFMD technique, International Journal of Data Analysis Techniques and Strategies, Volume 2, pp. 1-21 (2010).

AUTHOR BIOGRAPHY

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 4, July 2013

Kumari Priyanka Sinha, Working as a Assistant Professor in Central University of Bihar, She got M.Tech degree from Indian Institute of Information Technology.

Rajeshwar Puran, Working as a Assistant Professor in Central University of Bihar, She got M.Tech degree from Indian Institute of Information Technology.