www.ijemr.net ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Volume-7, Issue-3, May-June 2017

International Journal of Engineering and Management Research

Page Number: 379-382

Frequent Itemset Mining Algorithms for Knowledge Discovery – A Compendious Review of Various Approaches

Nafisur Rahman[1], Samar Wazir[2]

[1, 2] Department of Computer Science and Engineering, School of Engineering Sciences and Technology, Jamia Hamdard, New Delhi, INDIA

ABSTRACT

***Frequent Itemset Mining* is a *Data Mining* task that has drawn the attention of researchers over the years. This concept is used in *Market Basket Analysis* in particular and *Decision Support* problems in general. In this paper, we have focused on the developments in this area so far. We start with an account of the algorithms that generate *Candidate Itemsets*. This class of algorithms proves costly, particularly in cases where there exist a large number of *Itemsets*. Then we describe the tree based *Frequent Pattern* algorithm that does not require *Candidate Itemset Generation*, thereby bringing the cost down. After that, we introduce lattice based algorithms in which fewer database scans are needed and hence I/O cost gets reduced. Then we discuss an algorithm that uses a single recursive function and simplifies its structure without worrying too much about the speed. Then we move on to have a look at an algorithm that uses hyperlinks and saves time and space. Then we describe computationally efficient algorithms for *Closed Itemsets*. Finally, we discuss some algorithms that leverage the inherent advantages of some special data representations to enhance efficiency. While keeping the nub and essence intact, we have avoided the original algorithmic and mathematical notations to keep it perspicuous, coherent, and comprehensible. However, a simplified block diagram has been given, wherever the nature of the algorithm permits it, to summarize the functioning of the algorithm briefly.**

Keywords--- Data Mining, Decision Support, Itemsets, Frequent Itemsets, Frequent Itemset Mining, Market Basket Analysis

I. INTRODUCTION

Data Mining aims at discovering valuable information, which is not obvious, from a huge collection of data. The concept of *Frequent Itemsets*, in the context of *Data Mining*, refers to interesting pattern discovered from databases. An *Itemset* is essentially a collection or set of one or more items. If an *Itemset* has a support equal to or beyond a minimum support threshold, it is termed as a *Frequent Itemset*. The original motivation for *Frequent Itemset Mining* was its usefulness in Market Basket Analysis. It continues to find its application in various decision support problems to help different types of businesses and enterprises. Approaches employed for *Frequent Itemset Mining* have been evolving over the years. The very fundamental algorithms viz. *Apriori* and *AprioriTid* depend on generating *Candidate Itemsets* and making multiple passes on the given data. *FP-Growth* is a tree-based approach based on partition based divide and conquer strategy that does not require generating *Candidate Itemsets*. *Eclat* makes use of the structural properties of the *Frequent Itemsets* and decomposes them into lattices and sublattices for independent processing. *dEclact* employs a special way of vertical representation known as *Diffset* to enhance performance by tracking the difference between the transaction IDs of *Candidate Itemsets* from *Frequent Itemsets*. *Relim* uses a single recursive function to ensure simplicity. *H-Mine* employs a data structure that uses hyperlinks that dynamically adjust in mining. *LCMFreq* enumerates *Closed Itemsets* avoiding duplications and uses hypercube decomposition. *PrePost* and *PrePost*$^+$ use a vertical data representation called *N-list* that helps store important information about *Frequent Itemsets*. *FIN* makes use of *Nodesets* to save memory along with ensuring speed.

A short holistic description of the algorithms mentioned above can be found in the sections that follow.

II. APRIORI

Proposed by *Agrawal* and *Srikant* in 1994 [1], this algorithm follows the principle that supersets of *infrequent Itemsets* should not be generated or tested. It aims to find interesting patterns by discovering *Frequent Itemsets* after generating *Candidate Itemsets* by selecting only those *Candidate*s whose support is over a minimum support

threshold. It counts the occurrences of combinations of attributes of a data set and reports the count if it is above a minimum support threshold. The *Frequent Itemset* thus found is then used to generate the new *Candidate Itemset* from this subset. The process of scanning, generation of *Candidate Itemsets*, and discovery of *Frequent Itemsets* continues iteratively until it stops generating any more *Frequent Itemsets*. This is faster than Brute-force approach [10], [11].

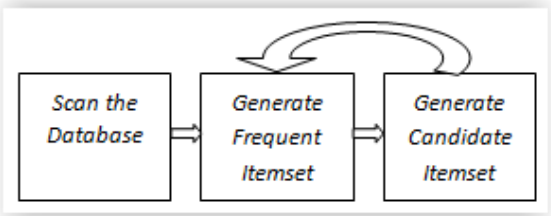

Figure 1: Simplified Block Diagram for Apriori

III. APRIORITID

This algorithm is a variant of standard *Apriori* with a difference that after the first pass, it does not require the original database for counting support. Instead, it uses the *Candidate Itemsets* that keep the Transaction IDs associated with the *Candidate*s. For large sized databases, this potentially reduces the entries in such *Itemsets* than the number of transactions in the actual database. *Apriori* and *AprioriTID*[1] can be combined as *Apriori*Hybrid for performance scale-up.

IV. FP-GROWTH

Given by *Han* et al. in 2004 [2], this algorithm uses a frequent pattern tree to store important information about frequent patterns. The tree has nodes only for frequent items. The arrangement of the nodes ensures that more frequent nodes get a higher chance of node sharing than the less frequent nodes. In this mining approach, only conditional pattern base is examined, its frequent pattern tree is constructed, and mining is performed recursively with this tree. The suffix patterns get concatenated to achieve pattern growth. Since this method requires only restricted test and does not require the generation of *Candidate Itemsets*, it is usually much faster than the *Apriori*-like algorithms that require generation and test. For searching, *FP-Growth* uses partition based divide and conquer approach thereby reducing the search space drastically [10], [11].

V. ECLAT

Eclat is one of the six scalable algorithms for association rule mining proposed by *Zaki* in 2000 [3]. It uses a vertical database format where each *Itemset* is associated with a list of transactions of its occurrences. It considers the original search space as a lattice and decomposes the same into smaller sublattices which can independently be processed in the memory. The *Frequent Itemsets* are enumerated within the sublattices by employing bottom-up, top-down, and hybrid search strategies. This approach requires fewer database scans and hence reduces the I/O costs.

VI. dEclat

Despite the obvious advantages of popular vertical databases, for very frequent items, intersection time and the size of the intermediate sets of transaction IDs increase excessively. *Zaki* and *Gauda* addressed these issues by introducing a different type of vertical data representation *Diffset* in 2001 [4]. It is confined to tracking the differences in the transaction IDs of *Candidate Itemsets* from *Frequent Itemsets*. This helps in cutting down the memory requirements to a large extent.

VII. RELIM

Proposed by *Borgelt* in 2005 [5], *Relim* aims at ensuring simplicity. Speed is not its primary concern though it is fairly fast. This algorithm performs all the work in a simple recursive function. The transaction database is preprocessed by scanning and determining the number of occurrences of the items, discarding the infrequent items, and sorting the items in ascending order. The original transaction database gets refined into a set of transaction lists with one list representing one item. These lists are stored as arrays. These arrays are traversed from left to right to process the transactions in a list in a call to the recursive function in order to find all the *Frequent Itemsets*. After recursively processing a list, its elements are then assigned to the remaining lists, or they are discarded, depending on the transactions represented by them. After all such assignments, the array will finally be empty.

VIII. H-MINE

Given by *Pei* et al. [6], *H-Mine* is an algorithm that takes advantage of a data structure H-struct which uses hyperlinks. The links get dynamically adjusted during the course of mining. H-struct organizes the *Frequent Item Projections* of the transactions in the database. Item-IDs and hyperlinks are used to represent the items in the *Frequent Item Projections*. At the time of creation of the H-struct, the header table of the structure contains the heads of the queues of *Frequent Item Projections*. These heads are linked by the hyperlinks. Building an H-struct requires only two scans of a transaction database. Having a polynomial space complexity, *H-Mine* is more efficient

than their pattern-growth counterparts. This algorithm is not only highly efficient for memory based *Frequent Itemset Mining*, but it can also be extended to vast disk-based databases by employing some optimization techniques.

IX. LCMFREQ

LCMFreq [7] employs database reduction to reduce the cost of counting the number of occurrences of frequent items, checking closedness, and checking maximality. Database reduction is achieved by removing each item included in less than a fixed number of transactions or by removing each item included in all the transactions or by merging the identical transactions into a single transaction. *LCMFreq* computes the number of occurrences of frequent items in linear time by using occurrence deliver method. It uses prefix preserving closure extension and generates Closed *Itemsets* without memory or duplication. By prefix preserving closure extension, the time complexity is bounded by a linear function in the number of frequent Closed *Itemsets*. The algorithm uses hypercube decomposition for fast enumeration by equivalence class grouping of *Frequent Itemsets*.

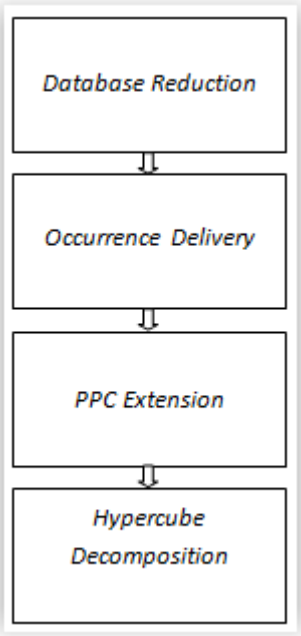

Figure 2: Simplified Block Diagram for LCMFreq

X. PREPOST

Proposed by *Deng* et al. in 2012 [8], *PrePost* uses a vertical data structure called *N-list* which comes from a prefix tree called PPC-tree that stores important information about *Frequent Itemsets*. PPC-tree consists of a root and a set of prefix subtrees. As the nodes of the PPC-tree are shared by the transactions with common prefixes, it ensures the compactness of *N-list*. Each node is encoded with a pre-order and post-order code. The algorithm follows the sequence of constructing a PPC-tree, constructing *N-list* of each *Frequent Itemset*, scanning the PPC-tree to find all *Frequent Itemsets*, and then mining all *Frequent Itemsets*. By using the single path property of *N-list*, in some cases, *PrePost* algorithm can directly discover *Frequent Itemsets* without having to generate *Candidate Itemsets*.

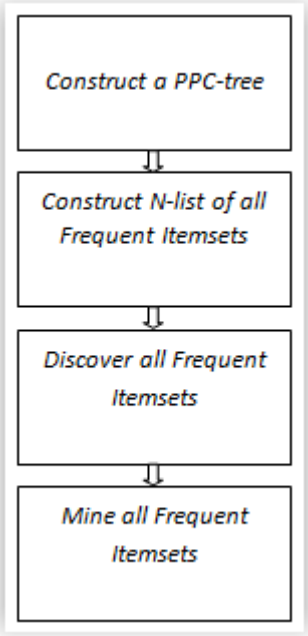

Figure 3: Simplified Block Diagram for PrePost

XI. PREPOST$^+$

Even with the adoption of single path property of *N-list* data structure, *PrePost* still suffers from the problem of too many *Candidate*s as its approach is similar to that of *Apriori*. *PrePost*$^+$ [8] addresses this issue by employing *N-list* to find *Frequent Itemsets* using set enumeration search tree and uses Children-Parent Equivalence pruning for search space reduction. The basic difference between *PrePost*+ and *PrePost* is that *PrePost*+ adopts superset equivalence for pruning while *PrePost* adopts single path property of *N-list* for pruning.

XII. FIN

Deng and *Sheng* [9] presented a fast *Frequent Itemset Mining* Algorithm *FIN* that uses a data structure *Nodeset* which is different from *N-list* in that it requires either pre-order or post-order code of each node but not both. Thus, *Nodesets*, which are based on POC-tree, save

half of the memory in comparison to *N-list* which requires both pre-order and post-order code for each node of the PPC-tree. *FIN* is usually faster than *PrePost* and *PrePost*$^{+}$ and consumes far too less memory for dense databases.

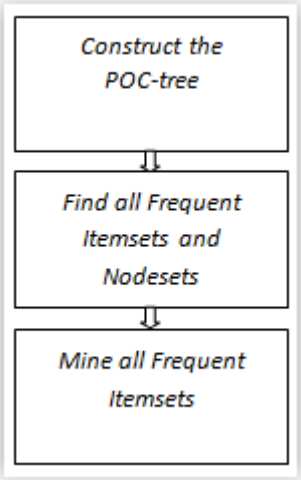

Figure 4: Simplified Block Diagram for FIN

XII. CONCLUSIONS

In this paper, we have covered the gist of the significant works done so far in the area of Frequent Itemset Mining. Researchers have shown and continue to show tremendous interest in this area. Going by the advancements in the field over the years coupled with ever increasing demands of saving time and space, the future of the research in this area promises to be of great interest.

REFERENCES

[1] R. Agrawal, and R. Srikant, Fast Algorithms for Mining Association Rules, Proc. of the 20th VLDB Conference, Santiago, Chile, 1994, pp. 487-499.
[2] Jiawei han , Jian Pei, Yiwen Yin Runying Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowledge Discovery, 8, 53–87, 2004 Kluwer Academic Publishers
[3] Mohammed J. Zaki, Scalable Algorithms for Association Mining, IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 3, May/June 2000
[4] Mohammed J. Zaki and Karam Gouda, Fast Vertical Mining Using Diffsets. In 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Aug 2003.
[5] Christian Borgelt, Keeping Things Simple: Finding Frequent Item Sets by Recursive Elimination
Workshop Open Source Data Mining Software (OSDM'05, Chicago, IL), 66-70
ACM Press, New York, NY, USA 2005
[6] Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang and Dongqing Yang, H-Mine: Fast and space-preserving frequent pattern mining in large databases IIE Transactions (2007) 39, 593–605
[7] Takeaki Uno, Masashi Kiyomi, Hiroki Arimura, LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets, ICDM 2004
[8] Deng Zhi Hong, Wang Zhong Hui & Jiang Jia Jian, A new algorithm for fast mining frequent itemsets using N-lists, Information Sciences, September 2012 Vol. 55
[9] Zhi-Hong Deng, Sheng Long Lv Fast Mining frequent itemsets using nodesets, Expert sytems with applications 41 (2014).
[10] Han J. and Kamber M. (2006) Data Mining Concepts and Techniques. Second edition, Morgan Kaufmann Publishers.
[11] Agarwal C. C. and Han J. (2014) Frequent Pattern Mining. Springer, 2014.