A Survey on Frequent Item sets Mining for Big Data

دراسة استقصائية عن تعدين مجموعه عناصر متكررة فى البيانات الكبيرة

Engy A. El-Shafaiy[1], Ali I. El-Desouky[2] and Yousry M. AbdulAzeem[3]

[1]Computers and Systems Department, Faculty of Engineering, Mansoura University, Egypt, Email: engy_eng@yahoo.com

[2] Computers and Systems Department, Faculty of Engineering, Mansoura University, Egypt

[3] Computers and Systems Department, Faculty of Engineering, Mansoura University, Egypt, Email: yousry@mans.edu.eg

الملخص

" بيانات كبيرة " مرتبطه بالكميات الضخمة والمعقدة وزيادة مجموعات البيانات متعددة من مصادر مستقلة. وفى الوقت الحاضر تجرى بسرعة كبيرة البيانات تتوسع فى جميع مجالات العلوم والهندسة بسبب التطور السريع للبيانات وتخزين البيانات وجمع الشبكات. بسبب تقلب المناخ الحجم والسرعة و" التنقيب عن البيانات كبيرة " يتمتع بقدرة استخراج المعلومات البناءة من تدفقات البيانات او مجموعات. استخراج البيانات يتضمن تحليل كميات كبيرة من البيانات من اجل تحديد مختلف القوالب الكبيرة البيانات. "تعدين مجموعه عناصر متكررة" هو واحد من اهم المهام لاستكشاف انماط المفيدة من مجموعات كبيرة من البيانات. استخراج قواعد الرابطة انماط متكررة من استخراج البيانات ذات الاهمية للعديد من الصناعات يمكن ان تقدم توجيهات فى عمليات صنع القرار مثل التسويق سلة الاسواق تحليل مجموعة متنوعة...الخ. تقنيات اكتشاف قواعد الرابطة من البيانات تركز تقليديا على تحديد العلاقة بين البندين التنبؤ ببعض جوانب السلوك البشرى سلوك الشراء عادة. وتقدم هذه الورقة استعراض مختلف لتقنيات التعدين لمجموعه العناصر المتكررة.

Abstract

Big Data" connects large-volume, complex, and increasing data sets with multiple independent sources. Nowadays, Big Data are speedily expanding in all science and engineering domains due to the rapid evolution of data, data storage, and the networking collection capabilities. Due to its variability, volume, and velocity, "Big Data mining" enjoys the ability of extracting constructive information from huge streams of data or datasets. Data mining includes exploring and analyzing big quantities of data in order to locate different molds for big data. "Frequent item sets Mining" is one of the most important tasks for discovering useful and meaningful patterns from large collections of data. Mining of association rules from frequent patterns from big data mining is of interest for many industries, for it can provide guidance in decision making processes; such as cross marketing, market basket analysis, promotion assortment, ...etc. The techniques of discovering association rules from data have traditionally focused on identifying the relationship between items predicting some aspect of human behavior; usually buying behavior. This paper provides a review on different techniques for mining frequent item sets.

Keyword

Association Rule Mining, Data Mining, Frequent Item sets, Big Data, Frequent Pattern Mining, Apriori, FP-Growth.

1. Introduction

"Data Mining" is gaining importance due to the available huge amount of data. Retrieving information from the warehouse is not only tedious, but is also difficult in some cases [1]. The most important usage of Data Mining is customer segmentation in marketing, shopping cart analyzes, management of customer relationship, campaign management, Web usage mining, text mining, player tracking ...etc. In Data Mining, "Association Rule Mining" is one of the important techniques for discovering meaningful patterns from large collections of data [2]. Discovering frequent item sets plays an important role in mining association rules,

Accepted: 22 June, 2015

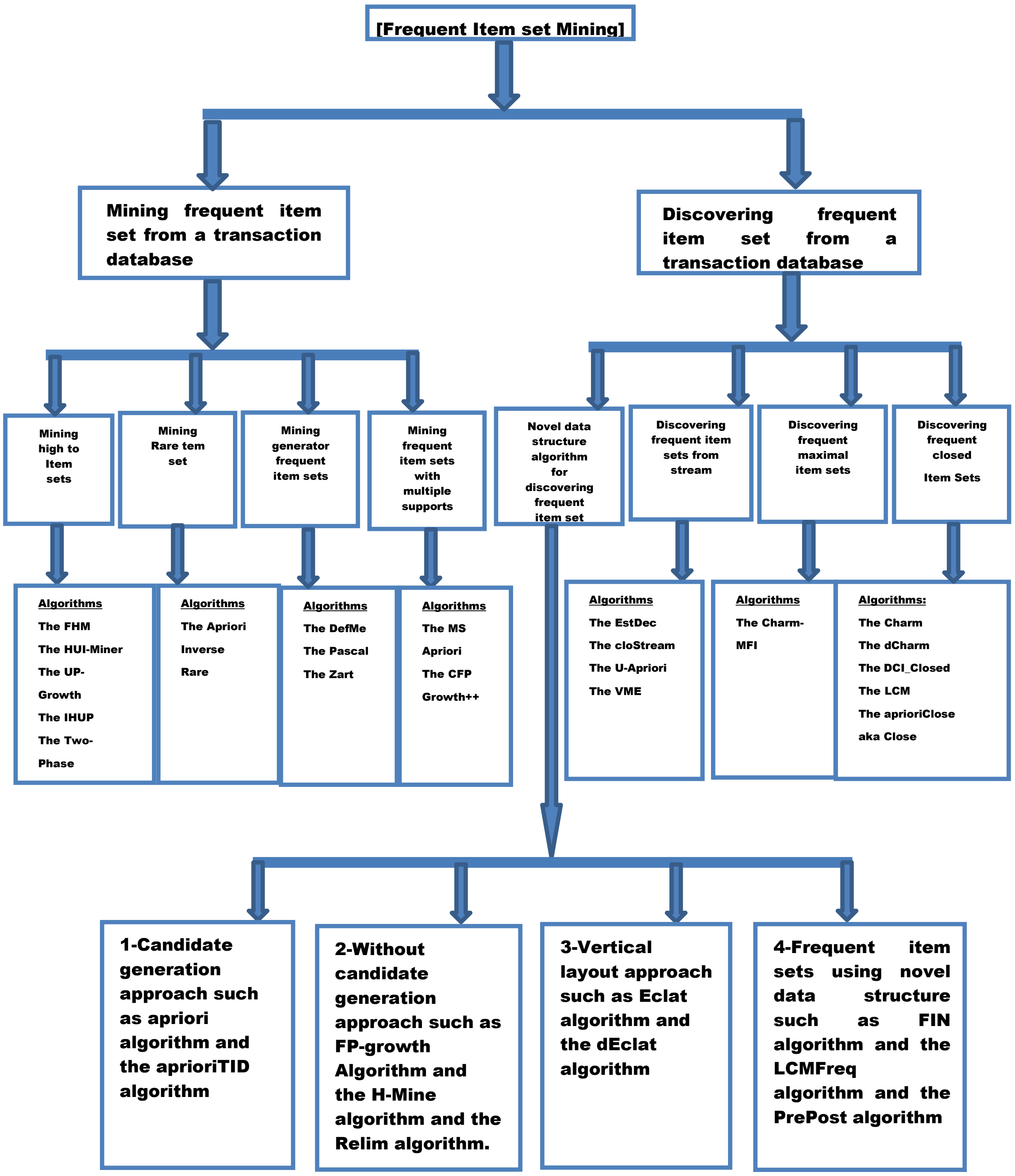
web log mining, and many other interesting patterns among complex data. Besides being the discovery of hidden information found in databases, Data Mining can be viewed as a step in the knowledge discovery process [3]. Data mining functions include clustering, classification, prediction, and link analysis (associations). One of the most important data mining applications is that of mining association rules. Association rules are first introduced by Agarwal[4]. Association rules are helpful for analyzing customer behavior in retail trade, banking system...etc. Association rule can be defined as {X, Y} => {Z}. In retail stores if customer buys X, Y, he is likely to buy Z. The concept of "Association Rule" is applied today in many fields, such as intrusion detection, biometrics, production planning ...etc. Association Rule Mining means to find out association rules that satisfy the predefined minimum support and confidence from a given database. If an item set is said to be frequent, that item set supports both minimum support and confidence. The problem of finding the association rules can be divided into two parts

1. Find all frequent item sets: Frequent item sets will occur at least as frequently as a pre- determined minimum support count, i.e. they must satisfy the minimum support.

2 Generate strong association rules from the frequent item sets: These rules must satisfy minimum support and minimum confidence values. Frequent Pattern Mining is the process of mining data in a set of items or some patterns from a large database. The resulted frequent set data supports the minimum support threshold. A frequent pattern is a pattern that occurs frequently in a dataset. A frequent item set should appear in all the transactions of that data base. Frequent pattern mining [5] plays a major field in research, since it is a part of data mining. Many research papers and articles are published in the field of Frequent Pattern Mining (FPM). This Paper provides details about frequent pattern mining algorithm, types and extensions of frequent pattern mining, association rule mining algorithm, rule generation, and suitable measures for rule generation. This paper describes about various existing FPM algorithms and data mining algorithm for big data by applying frequent pattern mining algorithm and suitable measures [6]. Frequent pattern mining is fundamental in data mining. The goal is to compute on huge data efficiently. Finding frequent patterns plays a fundamental role in association rule mining, classification, clustering, and other data mining tasks.

2. Frequent Data Item set Mining

Frequent patterns, such as frequent item sets, substructures, sequences term-sets, phrase sets, and sub graphs, generally exist in real-world databases [7]. Identifying frequent item sets is one of the most important issues faced by the knowledge discovery and data mining community. Frequent item set mining plays an important role in several data mining fields as association rules, warehousing, correlations, clustering of high-dimensional biological data, and classification. The frequent item set mining is motivated by problems such as market basket analysis. A tuple in a market basket database is a set of items purchased by customer in a transaction. An association rule mined from market basket database states that if some items are purchased in transaction, then it is likely that some other items are purchased as well. As frequent data item sets mining are very important in mining the association rules. Therefore there are various techniques proposed for generating frequent item sets so that association rules are mined efficiently [8].


[Frequent Item set Mining]
Mining frequent item set from a transaction database
Discovering frequent item set from a transaction database
Mining high to Item sets
Mining Rare tem set
Mining generator frequent item sets
Mining frequent item sets with multiple supports
Novel data structure algorithm for discovering frequent item set
Discovering frequent item sets from stream
Discovering frequent maximal item sets
Discovering frequent closed Item Sets
Algorithms The FHM The HUI-Miner The UP-Growth The IHUP The Two-Phase
Algorithms The Apriori Inverse Rare
Algorithms The DefMe The Pascal The Zart
Algorithms The MS Apriori The CFP Growth++
Algorithms The EstDec The cloStream The U-Apriori The VME
Algorithms The Charm-MFI
Algorithms: The Charm The dCharm The DCI_Closed The LCM The aprioriClose aka Close
1-Candidate generation approach such as apriori algorithm and the aprioriTID algorithm
2-Without candidate generation approach such as FP-growth Algorithm and the H-Mine algorithm and the Relim algorithm.
3-Vertical layout approach such as Eclat algorithm and the dEclat algorithm
4-Frequent item sets using novel data structure such as FIN algorithm and the LCMFreq algorithm and the PrePost algorithm

(FIG 1: Techniques of Frequent Itemsets Mining.)

Using Novel Data structure Algorithm for Frequent Itemsets are divided into basic four techniques as show in Fig 1:

1. Candidate generation approach (E.g. Apriori algorithm)
2. without candidate generation approach (E.g. FP-growth algorithm)
3. Vertical layout approach (E.g. Eclat algorithm
4. Using novel data structure approach (E.g. FIN algorithm)

One of the challenges of frequent pattern mining is that a large number of redundant patterns are often mined [9]. For example, the subset of a frequent pattern is also guaranteed to be frequent and by mining a maximal item set, one is assured that the other frequent patterns can also be generated from this smaller set. Therefore, one possibility is to mine for only maximal item sets. However, the mining of maximal item sets loses information about the exact value of support of the subsets of maximal patterns. Therefore, a further refinement would be to find closed frequent item sets [10, 11]. Closed frequent item sets are defined as frequent patterns, no superset of which have the same frequency as that item set. By mining closed frequent item sets, it is possible to significantly reduce the number of patterns found, without losing any information about the support level. Closed patterns can be viewed as the maximal patterns from each group of equi-support patterns (i.e., patterns with the same support). All maximal patterns are, therefore, closed. The depth-first method has been shown to have a number. There are number of algorithms used to mine frequent item sets. The most important algorithms are briefly explained here. The algorithms vary in the generation of candidate item sets and support count. The approaches of generating frequent item sets are divided into basic four techniques as show in Fig 1

2.1 Algorithms for Mining with Candidate generation approach

Algorithms for Mining with Candidate, each row of database represents a transaction which has a transaction identifier (TID), followed by a set of items [12].

2.1.1 Apriori Algorithm:

Apriori algorithm is, the most classical and important algorithm for mining frequent itemsets. Apriori is used to find all frequent itemsets in a given database DB. Apriori algorithm is given by Agrawal. The apriori algorithm uses the apriori principle, which says that the item set I containing item set X is never large if item set X is not large or all the non-empty subset of frequent item set must be frequent also[13].

Based on this principle, the apriori algorithm generates a set of candidate item sets whose lengths are (k+1) from the large k item sets and prune those candidates, which does not contain large subset. Then, for the rest candidates, only those candidates that satisfy the minimum support threshold (decided previously by the user) are taken to be large (k+1)-item sets. The apriori generate item sets by using only the large item sets found in the previous pass, without considering the transactions. Steps involved are:

1. Generate the candidate 1-itemsets (C_1) and write their support counts during the first scan. 2. Find the large 1-itemsets (L_1) from C_1 by eliminating all those candidates which does not satisfy the support criteria.
2. Join the L_1 to form C_2 and use apriori principle and repeat until no frequent itemset is found.

2.1.2 Partitioning Algorithm:

Partitioning algorithm is to find the frequent elements on the basis of dividing database into n partitions. It overcomes the memory problem for large database which do not fit into main memory because small parts of database easily fit into main memory. The algorithm executes in two phases. In the first phase, the Partition algorithm logically divides the database into a number of non-overlapping partitions. The partitions are considered one at a time and all large itemsets for that partition are generated [14]. At the end of phase I, these large itemsets are merged to generate a set of all potential large itemsets. In phase II, the actual support for these itemsets is generated and the large itemsets are identified. The partition sizes are chosen such that each partition can be accommodated in the main memory so that the partitions are read only once in each phase.

2.1.3 The Apriori TID algorithm:

This is an alternative to Apriori Itemset Generation. In this itemsets are dynamically added and deleted as transactions are read. This algorithm also used to reduce the number of database scan. It is based upon the downward disclosure property in which this adds the candidate itemsets at different point of time during the scan. It reduces the database scan for finding the frequent itemsets by just adding the new candidate at any point of time during the run time [15].

2.1.4 The Apriori Hybrid algorithm:

The Apriori still examines every transaction in the database. On the other hand, rather than scanning the database, AprioriTid scans Ck for obtaining support counts, and the size of Ck has become smaller than the size of the database. Based on these observations Apriori Hybrid algorithm has been designed to execution times for Apriori and Apriori Tid for different passes. In the earlier passes, Apriori does better than AprioriTid. However, AprioriTid beats Apriori in later passes, the reason for which is as follows. Apriori and AprioriTid use the same candidate generation procedure and therefore count the same itemsets. In the later passes, the number of candidate itemsets reduces However; uses Apriori in the initial passes and switches to AprioriTid in the later passes. Apriori performs better than AprioriTid in the initial passes but in the later passes AprioriTid has better performance than Apriori [16].

(Table 2: Comparison of Apriori, AprioriTid and AprioriHybrid.)

SNO	PROPERTIES	Apriori Algorithm	Apriori Tid Algorithm	Apriori Hybrid Algorithm
1	**Candidate generation**	Candidate item- sets are generated using only the large item-sets of the previous pass without considering the transactions in the database.	The database is not used at all for counting the support of candidate item- sets after the first pass.	**Hybrid Algorithm** can be designed that uses **Apriori** in the initial passes and switches to AprioriTid in the later passes.
2	**The methodology**	Uses Join &prune steps.	Uses Join &prune as well as Tids.	Uses **Apriori** + AprioriTid
3	**Database scans**	Multiple scan over the database.	Uses the database only one.	Uses **Apriori** + AprioriTid
4	**Memory usage**	It takes more space and memory for the candidate generation process.	In the Kth pass AprioriTid needs memory for Lk-1 and Ck-1 during candidate. An additional cost is incurred if it cannot completely fit into the memory.	An extra cost is incurred when shifting from Apriori to AprioriTid.
5	**Execution Time**	It takes more execution time for the candidate generation process.	For small problem it's better than Apriori but it takes more time for Large problem.	It's better than Apriori and AprioriTid.

2.2 Algorithms for Mining without candidate generation approach

This type of database uses divide and conquer strategy to mine itemsets therefore it counts the support more efficiently then Apriori based algorithms. Tree projected layout based approaches use tree structure to store and mines the itemsets. The projected based layout contains the record id separated by column then record. Tree Projection algorithms based upon two kinds of ordering breadth-first and depth-first [17].

2.2.1 FP-Growth Algorithm:

FP-Growth Algorithm using a compact data structure called FP-tree and extracts frequent itemsets directly from this structure. This algorithm is based upon the recursively divide and conquers strategy; first the set of frequent 1-itemset and their counts are discovered. Starting from each frequent pattern, Construct the conditional pattern base, and then its conditional FP-tree is constructed (which is a prefix tree). Until the resulting FP-tree is empty, or contains only one single path [18] [19]. (Single path will generate all the combinations of its sub- paths, each of which is a frequent pattern). The items in each transaction are processed in L order. (i.e. items in the set were sorted based on their frequencies in the descending order to form a list).

2.2.2 H-mine Algorithm:

A memory-based, efficient pattern-growth algorithm, H-mine (Mem), is for mining frequent patterns for the datasets that can fit in (main) memory. A simple, memory-based hyper structure, H-struct, is designed for fast mining. H-mine (Mem) has polynomial space complexity and is thus more space efficient than pattern growth methods like FP-growth and tree projection when mining sparse datasets, and also more efficient than apriori-based methods which generate a large number of candidates. H- mine has very limited and exactly [20].

Predict table space overhead and is faster than memory-based a priori and FP-growth. H-mine uses H-struct new data structure for mining purpose known as hyperlinked structure. It is used upon the dynamic adjustment of pointers which helps to maintain the processed projected tree in main memory. Therefore, H-mine proposed for frequent pattern data mining for datasets that can fit into main memory**.**

2.2.3 The Relim algorithm:

This algorithm for finding frequent item sets, which is strongly inspired by the FP-growth algorithm and very similar to the H-mine algorithm. It does its work without prefix trees or any other complicated data structures, processing the transactions directly. Its main strength is not its speed (although it is not slow, even outperforms Apriori and Eclat on some data sets), but the simplicity of its structure. Basically, all the work is done in one simple recursive function [21].

2.3 Algorithms for Mining from, Vertical Layout Database

The vertical pattern mining algorithms use a vertical representation of the transaction database to enable more efficient counting. In vertical layout data set, each column corresponds to an item, followed by a TID list, which is the list of rows that the item appears [22]. A vertical organization means that the layout of the data is column-wise as shown in Figure 2. Advantages of vertical achieved the best possible runtime performance.

Table 2: Comparison of FP-Growth, H-mine and The Relim.

SNO	Properties	FP-Growth Algorithm	H-mine Algorithm	The Relim algorithm
1	**Technique**	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support.	It uses the hyperlink pointers to store the partitioned projected database in main memory.	Relim (Recursive elimination) processes the transactions directly, based on FP-Growth algorithm without the prefix tree.
2	**Memory utilization**	Due to compact structure and no candidates generation require less memory.	Memory is utilized according to needs and partitions of projected database.	It save a lot of memory for storing the transaction. Relim algorithm deletes all items from the transaction database that has least frequent items. Relim is better when min support is low.
3	**Databases**	Suitable for large and medium datasets.	Suitable for sparse and dense datasets.	Suitable for large datasets
4	**Execution Time**	Execution time is large due to complex compact data structure.	Execution time is large then FP-tree and others because of partition the database.	Execution time is large.

Horizontal Layout

LIST OF ITEMS			
M	S	I	W
1	1	2	1
2	2	4	4
3	3	5	5
4	5		

Vertical Layout

TID	LIST OF ITEMS
1	**M, S, W**
2	**M , I, S**
3	**M, S**
4	**I, M, W**
5	**I, W, S**

(Fig 2: Vertical Layout Database)

2.3.1 Eclat algorithm:

Equivalence Class Clustering and bottom up Lattice Traversal is known as ECLAT algorithm. This algorithm is also used to perform item set mining [23]. It uses TID set intersection that is transaction id intersection to compute the support of a candidate item set for avoiding the generation of subsets that does not exist in the prefix tree. For each item store a list of transaction id. In this type of algorithm, for each frequent itemset i new database is created D_i.

This can be done by finding 'j' which is frequent corresponding to 'I' together as a set Then j is also added to the created database i.e. each frequent item is added to the output set. It uses the join step like the Apriori only for generating the candidate sets but as the items are arranged in ascending order of their support thus less amount of intersection is needed between the sets.

2.3.2 The d Eclat algorithm

Algorithm developed to generate all frequent itemsets in a depth-first manner is the Eclat (Equivalence Class Transformation) algorithm. If the database is stored in the vertical layout, the counting of support can be much easier by simply intersecting the covers of two its subsets that together give the set itself. The Eclat algorithm essentially used this technique in the Apriori algorithm. Always this is not possible since the total size of all covers at a certain iteration of the local set generation procedure could exceed 28 main memory limits. It is usually more efficient to first find the frequent items and frequent 2-sets separately and use the Eclat algorithm only for all larger sets [24] [25].

Table 3: Comparison of Eclat and The dEclat.

SNO	PROPERTIES	Eclat algorithm	The dEclat algorithm
1	**Technique**	Use intersection of transaction ids list for generating candidate itemsets	Use set intersection. Of transaction ids list for Generating candidate itemsets
2	**Memory utilization**	Require less amount of memory compare to apriori if itemsets are small	Suitable for large databases
3	**Databases**	Suitable for medium and dense datasets but not suitable for small datasets	Suitable for medium and large datasets,
4	**Execution Time**	Execution time is small then apriori algorithm	Execution time is small then Apriori algorithm

2.4 Algorithms for Mining using Nodesets

Node-list and N-list, two novel data structure proposed in recent years, have been proven to be very efficient for mining frequent itemsets. A node-set contains two tables: one called mapping and the other called itemlist. The main problem of these structures is that they both need to encode each node of a PPC-tree [26] with pre-order and post-order code. This causes that they are memory- consuming and inconvenient to mine frequent itemsets. [28] propose Nodeset, a more efficient data structure, for mining frequent itemsets. Nodesets require only the pre-order (or post-order code) of each node, which makes it saves half of memory compared with N-lists and Node-lists. Based on Nodesets, an efficient algorithm called FIN to mining frequent itemsets.These three algorithms were proposed by Deng et al [26] [27] [28], and are based on three novel data structures called Node-list [26], N-list [27], and Nodeset [28] respectively for facilitating the mining process of frequent itemsets. They are based on a FP-tree with each node encoding with pre-order traversal and post-order traversal. Compared with Node-lists, N-lists and Nodesets are more efficient. This causes the efficiency of PrePost and FIN is higher than that of PPV.

2.4.1 N-list

A novel vertical data representation called N-list, which originates from an FP-tree-like coding prefix tree called PPC-tree that stores crucial information about frequent itemsets. Based on the N-list data structure it develop an efficient mining algorithm, PrePost, for mining all frequent itemsets. Efficiency of PrePost is achieved by the following three reasons. First, N-list is compact since transactions with common prefixes share the same nodes of the PPC-tree. Second, the counting of itemsets' supports is transformed into the intersection of N-lists and the complexity of intersecting two N-lists can be reduced to O (m + n) by an efficient strategy, where m and n are the cardinalities of the two N-lists respectively. Third, PrePost can directly find frequent itemsets without generating candidate itemsets in some cases by making use of the single path property of N-list. N-list have experimentally evaluated PrePost against four state-of-the-art algorithms for mining frequent itemsets on a variety of real and synthetic datasets. The experimental results show that the PrePost algorithm is the fastest in most cases. Even though the algorithm consumes more memory when the datasets are sparse, it is still the fastest one.

2.4.2 Node-list

Node-list and N-list, two novel data structure proposed in recent years, have been proven to be very efficient for mining frequent itemsets. The main problem of these structures is that they both need to encode each node of a PPC-tree with pre-order and post-order code. This causes that they are memory- consuming and inconvenient to mine frequent itemsets. Node-list Based on PPC-tree and efficiently mining frequent patterns in large datasets. First, PPV obtains the Node-list of each frequent item. Then, PPV obtains Node-lists of the candidate patterns of length (k+1) by intersecting Node-lists of frequent patterns of length k and thus discovers the frequent patterns of length (k+1). The advantages of PPV are that it transforms the mining of frequent patterns into the intersecting of Node-lists, which makes mining process easier, and adopts an efficient method for intersecting two Node-lists, which has an average time complexity of O(m+n) [26].

2.4.3 Nodeset

Nodeset, a more efficient data structure, for mining frequent itemsets. Nodesets require only the pre-order (or post-order code) of each node, which makes it saves half of memory compared with N-lists and Node-lists. Based on Nodesets, who present [28] an efficient algorithm called FIN to mining frequent itemsets. For evaluating the performance of FIN. A novel structure called Nodeset to facilitate the process of mining frequent itemsets. Based on Nodesets, The advantage of Nodeset lies in that it encodes each node of a POC-tree with only pre-order (or post-order) constructed. The extensive experiments show that the Nodeset structure is efficient and FIN run faster than PrePost on the whole. Especially, FIN consumes

Much less memory than PrePost on dense datasets. These three algorithms were Proposed by Deng et al [26] [27] [28], and are based on three novel data structures called Node-list [26], N-list [27], and Nodeset [28] respectively for facilitating the mining process of frequent itemsets. They are sets of nodes in a FP-tree with each node encoding with pre-order traversal and post-order traversal. Compared with Node-lists, N-lists and Nodesets are more efficient. This causes the efficiency of PrePost and FIN [28] is higher than that of PPV[27].

Table 4: Comparison of N-list, Node-list and The Nodeset.

SNO	PROPERTIES	N-list	Node-list	Nodeset
1	**Technique**	PrePost employs a novel data structure, N-list, to represent itemsets. N-list stores all crucial information about itemsets. By combining the search approach of candidate set generation-and-test and the search approach of mining frequent itemsets directly without candidate generation, PrePost achieves very high efficiency in mining frequent itemsets.	A novel vertical algorithm called PPV for fast frequent pattern discovery. PPV works based on a data structure called Node-lists, which is obtained from a coding prefix-tree called PPC-tree.	Nodesets require only the pre- order (or post-order code) of each node, which makes it saves half of memory compared with N-lists and Node-lists. Based on Nodesets, we present an efficient algorithm called FIN to mining frequent itemsets.
2	**Memory utilization**	Algorithm consumes more memory when the datasets are sparse.	Memory is utilized according to needs.	It saves a lot of memory for storing the transaction. it saves half of memory compared with N- lists and Node-lists.
3	**Databases**	Suitable for sparse.	Suitable for large and medium datasets.	Suitable for large datasets.
4	**Execution Time**	PrePost runs faster under all minimum supports.	Execution time is large then N-list	FIN is faster than PrePost.

3. Conclusion

Frequent itemsets play an important role in many real-world applications. In this paper, we provide a survey of research on Frequent Itemsets Mining. The focus on frequent itemset mining and have tried to cover both early and recent literature related to mining frequent itemsets (FIs) and Frequent Itemsets Mining for Big Data (FIM). In particular, we have discussed in detail a number of algorithms [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, and 28] on mining FIs or FIM. Moreover, in this paper gives a brief survey on different approaches for mining frequent itemsets using association rules. The performance of algorithms depends on support level, nature and size of the datasets.

References.

[1] Zeng, Li, et al. "Distributed data mining: a survey." Information Technology and Management 13.4 (2012): 403-409.

[2] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. motoda, G.J. MClachlan, A. Ng, B. Liu, P.S. Yu, Z. Zhou, M. Steinbach, D. J. Hand,D. Steinberg,―Top 10 Algorithms in Data Mining,Knowl Inf Syst (2008) 141-37.

[3] Aggaraval R; Imielinski.t; Swami. A. 1993. Mining Association Rules between Sets of Items in Large Databases. ACM SIGMOD Conference. Washington DC, USA.

[4] R.C. Agarwal, C.C. Aggarwal, and V.V.V. Prasad. A tree projection algorithm for generation of frequent itemsets. Journal of Parallel and Distributed Computing, 61(3):350–371, March 2001.

[5] S.Vijayarani el.al., "Mining Frequent Item Sets over Data Streams using Éclat Algorithm" International Conference on Research Trends in Computer Technologies (ICRTCT-2013).

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, Proceedings 20th International Conference on Very Large Data Bases, pages 487–499. Morgan Kaufmann, 1994.

[7] Savasere A, Omiecinski E, Navathe S (1995) an efficient algorithm for mining association rules in large 687 databases. In: Proceedings of international conference on very large data bases, pp. 688–692

[8] R. Agrawal, T. Imielinski, and A.N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, volume 22(2) of SIGMOD Record, pages 207–216. ACM Press, 1993.

[9] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent Pattern Mining: Current Status and Future Directions, Data Mining and Knowledge Discovery, 15(1), pp. 55–86, 2007.

[10] C. Lucchese, S. Orlando, and R. Perego. Fast and memory efficient mining of frequent closed itemsets. IEEE TKDE Journal, 18(1), pp. 21–36, January 2006.

[11] F Pan, A. K. H. Tung, G. Cong, X. Xu. COBBLER: Combining column and Row Enumeration for Closed Pattern Discovery. SSDBM, 2004.

[12] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. In Data Mining, ICDM Conference, 2001.

[13] Q. Lan, D. Zhang, and B.Wu. A new Algorithm For Frequent Itemsets Mining Based On Apriori And FP-Tree, IEEE International Conference on Global Congress on Intelligent Systems, pp. 360–364, 2009.

[14] Savasere E. Omiecinski and Navathe S., "An efficient algorithm for mining association rules in large databases," In Proc. Int'l Conf. Very Large Data Bases (VLDB), pp: 432– 443, 199 5.

[15] http://associationrule.blogspot.in/2008/09/apriori-aprioritid-and-apriori-hybrid.html.

[16] Guofeng Wang, Xiu Yu, Dongbiao Peng, Yinhu Cui, Qiming Li, Research of Data Mining Based on Apriori algorithm in Cutting Database, IEEE, 2010 pp.

[17] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. Naughton, and P. A. Bernstein, editors, 2000 ACM SIGMOD Intl. Conference on Management of Data, pages 1–12. ACM Press, 05 2000.

[18] E. Azkural and C. Aykanat. A Space Optimization for FP-Growth, FIMI workshop, 2004.

[19] B. Racz. nonordfp: An FP-Growth Variation without Rebuilding the FP-Tree, FIMI Workshop, 2004.

[20] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: Hyper-structure mining of frequent patterns in

large databases. In Data Mining, ICDM Conference, 2001.

[21] http://www.borgelt.net/relim.html http://www.borgelt.net/doc/relim/relim.html

[22] Laila A. Abd EI. Megid et al, “Vertical Mining of Frequent Patterns using Diffset Groups”, International. Conference on Intelligent systems Design and Applications 2007.

[23] M. J. Zaki. Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering, 12(3), pp. 372–390, 2000.

[24] M. Zaki and K. Gouda. Fast vertical mining using diffsets. ACM KDD Conference, 2003.

[25] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. KDD Conference, pp. 283– 286, 1997.

[26] Deng, Z. & Wang, Z. A New Fast Vertical Method for Mining Frequent Patterns [1]. International Journal of Computational Intelligence Systems, 3(6): 733 - 744, 2010.

[27] Deng, Z.; Wang, Z. & Jiang, J. A New Algorithm for Fast Mining Frequent Itemsets Using N-Lists [2]. SCIENCE CHINA Information Sciences, 55 (9): 2008 - 2030, 2012.

[28] Deng, Z. & Lv, S. Fast mining frequent itemsets using Nodesets. Expert Systems with Applications, 41(10): 4505–4512, 2014