A LITERATURE SURVEY OF MODERN TECHNIQUES USED FOR FREQUENT ITEM SET MINING

[1]Rupesh Panwar, [2]Prof. Abhishek Raghuvanshi

Abstract: Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Frequent item set mining is useful in many real world applications. It has a huge array of applications ranging from market basket analysis to disease prediction. This paper presents a comprehensive survey of modern techniques for frequent item set mining.

1. INTRODUCTION:

Dramatic advances in data capture, processing power, data transmission, and storage capabilities are enabling organizations to integrate their various databases into data *warehouses*. Data warehousing is defined as a process of centralized data management and retrieval. Data warehousing, like data mining, is a relatively new term although the concept itself has been around for years. Data warehousing represents an ideal vision of maintaining a central repository of all organizational data. Centralization of data is needed to maximize user access and analysis. Dramatic technological advances are making this vision a reality for many companies and equally dramatic advances in data analysis software are allowing users to access this data freely. The data analysis software is what supports data mining.

Frequently, the data to be mined is first extracted from an enterprise data warehouse into a data mining database or data mart. There is some real benefit if your data is already part of a data warehouse. The problems of cleansing data for a data warehouse and for data mining are very similar. If the data has already been cleansed for a data warehouse, then it most likely will not need further cleaning in order to be mined. Furthermore, we will have already addressed many of the problems of data consolidation and put in place maintenance procedures. The data mining database may be a logical rather than a physical subset of our data warehouse, provided that the data warehouse DBMS can support the additional resource demands of data mining. If it cannot, then you will be better off with a separate data mining database [4].

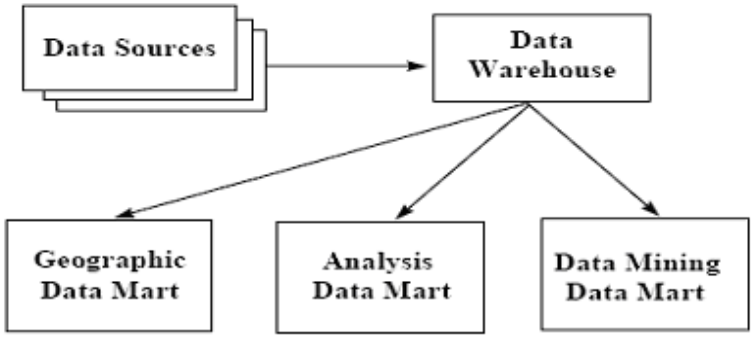

Figure 1 Data Warehouse and its Relations with Other Streams

This is where the classic beer/diapers bought together analysis came from. It finds groupings. Basically, this technique finds relationships in product or customer or wherever you want to find associations in data. The process of grouping a set of physical or abstract objects into classes of similar object is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group in many applications. Association analysis is the discovery of association rule showing attribute-value conditions that occur frequently together in a given set of data. Association analysis is widely used for market based or transaction data analysis. Association rules identify collections of data attributes that are statistically related in the underlying data. An association rule is of the form X => Y where X and Y are disjoint conjunctions of attribute-value pairs. The confidence of the rule is the conditional probability of Y given X, Pr(Y|X), and the support of the rule is the prior probability of X and Y, Pr(X and Y). Here probability is taken to be the observed frequency in the data set.

2. Literature Survey:

Dynamic Itemset Counting (DIC):

This algorithm [1] also used to reduce the number of database scan. It is based upon the downward disclosure property in which adds the candidate itemsets at different point of time during the scan. In this dynamic blocks are formed from the database marked by start points and unlike the previous techniques of Apriori it dynamically changes the sets of candidates during the database scan. Unlike the Apriori it cannot start the next level scan at the end of first level scan, it start the scan by starting label attached to each dynamic partition of candidate sets.

In this way it reduce the database scan for finding the frequent itemsets by just adding the new candidate at any point of time during the run time. But it generates the large number of candidates and computing their frequencies are the bottleneck of performance while the database scans only take a small part of runtime.

Assumption [4, 5]: The performance of all the above algorithms relies on an implicit assumption that the database is homogenous and thus they will not generate too many extra candidates than Apriori algorithm does. For example, if all partitions in Partition algorithm are not homogenous and nearly completely different sets of local frequent itemsets are generated from them, the performance cannot be good.

Improved Apriori algorithm

It was absorbed in [7] [5] that the improved algorithm is based on the combination of forward scan and reverse scan of a given database. If certain conditions are satisfied, the improved algorithm can greatly reduce the iteration, scanning times required for the discovery of candidate itemsets.

Suppose the itemset is frequent, all of its nonempty subsets are frequent, contradictory to the given condition that one nonempty subset is not frequent, the itemset is not frequent.

Based on this thought, proposes an improved method by combining forward and reverse thinking: find the maximum frequent itemsets from the maximum itemset firstly, then, get all the nonempty subsets of the frequent itemset. We know they are frequent on the basis of Apriori's property. Secondly, scan the database once more from the lowest itemset and count the frequent. During this scanning, if one item is found out being excluded in the frequent set, it will be processed to judge whether the itemsets associated with it is frequent, if they are frequent, they will be added in the barrel-structure (include frequent itemsets).we get all the frequent itemsets. The key of this algorithm is to find the maximum frequent itemset fast.

COFI-Tree Algorithm

COFI tree [8] generation is depends upon the FP-tree however the only difference is that in COFI tree the links in FP-tree is bidirectional that allow bottom up scanning as well [2,9]. The relatively small tree for each frequent item in the header table of FP-tree is built known as COFI trees [9]. Then after pruning mine the each small tree independently which minimise the candidacy generation and no need to build he conditional sub-trees recursively. At any time only one COFI tree is present in the main memory thus in this way it overcome the limitations of classic FP-tree which can not fit into main memory and has memory problem.

COFI tree is based upon the new anti-monotone property called global frequent/local non frequent property [20]. It states that all the nonempty subsets of frequent patterns with respect to the item X of an X-COFI tree must also be frequent with respect to item X. In this approach trying to find the entire frequent item set with respect to the one frequent item sets. If the itemset participate in making the COFI tree then it means that item set is globally frequent but this doesn't mean that item set is locally frequent with respect to the particular item.

CT-PRO Algorithm

CT-PRO is also the variation of classic FP-tree algorithm [10]. It is based upon the compact tree structure [10, 11]. It traverses the tree in bottom up fashion. It is based upon the non-recursive based technique. Compress tree structure is also the prefix tree in which all the items are stored in the descending order of the frequency with the field index, frequency, pointer, item-id [11]. In this all the items if the databases after finding the frequency of items and items whose frequency is greater than minimum support are mapped into the index forms according to the occurrence of items in the transaction. Root of the tree is always at index '0' with maximum frequency elements. The CT-PRO uses the compact data structure known as CFP-tree i e. compact frequent pattern tree so that all the items of the transactions can be represented in the main memory [10, 11, 12].

H-mine Algorithm

H-mine [3] algorithm is the improvement over FP-tree algorithm as in H-mine projected database is created using in-memory pointers. H-mine uses an 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 data sets that can fit into main memory. It has polynomial space complexity therefore more space efficient then FP-growth and also designed for fast mining purpose. For the large databases, first in partition the database then mine each partition in main memory using H-struct then consolidating global frequent pattern [3]. If the database is dense then it integrates with FP-Growth dynamically by detecting the swapping condition and constructing the FPtree.

This working ensures that it is scalable for both large and medium size databases and for both sparse and dense datasets [6]. The advantage of using in-memory pointers is that their projected database does not need any memory the memory required only for the set of in-memory pointers .

3. Conclusion:

Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors. Frequent pattern mining is the most common data mining technique, which is used for pattern discovery. This paper presented a survey of frequent item set mining techniques.

References:

[1] Brin.S, Motwani. R, Ullman. J.D, and S. Tsur. "Dynamic itemset counting and implication rules for market basket analysis". In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD), May 1997, pages 255–264.
[2] C. Borgelt. "An Implementation of the FP- growth Algorithm". Proc. Workshop Open Software for Data Mining, 1–5.ACMPress, New York, NY, USA 2005.
[3] Pei.J, Han.J, Lu.H, Nishio.S. Tang. S. and Yang. D. "H-mine: Hyper-structure mining of frequent patterns in large databases". In Proc. Int'l Conf. Data Mining (ICDM), November 2001.
[4] Yiwu Xie, Yutong Li, Chunli Wang, Mingyu Lu. "The Optimization and Improvement of the Apriori Algorithm". In Proc. Int'l Workshop on Education Technology and Training & International Workshop on Geoscience and Remote Sensing 2008.
[5] "Data mining Concepts and Techniques" by By Jiawei Han, Micheline Kamber, Morgan Kaufmann Publishers, 2006.
[6] S.P Latha, DR. N.Ramaraj. "Algorithm for Efficient Data Mining". In Proc. Int'l Conf. on IEEE International Computational Intelligence and Multimedia Applications, 2007, pp. 66-70.
[7] Dongme Sun, Shaohua Teng, Wei Zhang, Haibin Zhu. "An Algorithm to Improve the Effectiveness of Apriori". In Proc. Int'l Conf. on 6th IEEE Int. Conf. on Cognitive Informatics (ICCI'07), 2007.
[8] M. El-Hajj and O. R. Zaiane. "Inverted matrix: Efficient discovery of frequent items in large datasets in the context of interactive mining". In Proc. Int'l Conf. on Data Mining and Knowledge Discovery (ACM SIGKDD), August 2003.
[9] M. El-Hajj and O. R. Zaiane. "COFI-tree Mining:A New Approach to Pattern Growth with Reduced Candidacy Generation". Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA, CEUR Workshop Proceedings, vol. 90, pp. 112-119, 2003.
[10] Y. G. Sucahyo and R. P. Gopalan. "CT-ITL: Efficient Frequent Item Set Mining Using a Compressed Prefix Tree with Pattern Growth". Proceedings of the 14th Australasian Database Conference, Adelaide, Australia, 2003.
[11] Y. G. Sucahyo and R. P. Gopalan. "CT-PRO: A Bottom Up Non Recursive Frequent Itemset Mining Algorithm Using Compressed FP-Tre Data Structure". In proc Paper presented at the IEEE ICDM Workshop on Frequent Itemset Mining Implementation (FIMI), Brighton UK, 2004.
[12] A.M.Said, P.P.Dominic, A.B. Abdullah. "A Comparative Study of FP-Growth Variations". In Proc. International Journal of Computer Science and Network Security, VOL.9 No.5 may 2009.