Analyzing Transaction database using Fast Online Dynamic-Growth (FOLD-growth) algorithm

May Me Me Hlaing, Nang Saing Moon Kham
University of Computer Studies, Yangon
maymimihlaing@gmail.com

Abstract

Data mining is the process of analyzing large data sets in order to find patterns. Mining frequent patterns is a fundamental and crucial task in data mining problems. The association rule mining is needed in order to search for interesting relationship among items from a very large database. FP-Growth algorithm that allows mining of the frequent itemsets without candidate generation. In this paper, implement the pattern growth approach (FOLD-Growth) algorithm is used to analyze transaction database in the real world.

1. Introduction

Data mining is the process of analysis of raw data in the database and synthesizing it into information that is useful for affective decision making. Association rule mining finds interesting association or correlation relationships among a large set of data items. Frequent patterns are patterns that appear in a dataset frequently .A set of items that appear frequently together in a transaction dataset is a frequent itemset. The supported-Order Tree is constructed by extracting 1-itemsets and 2-itemsets from all transactions. It can be constructed online , meaning that each time a new transaction arrives, and Support-Ordered Tree can be incrementally updated.

In this paper, there are six sections. Section 1 describes the introduction of this paper. In section 2, related works for this presented system. Section 3 describes the Background Theory used by this system. Section 4 is the description of proposed system architecture. In section 5, the implementation of this system described. Section 6 is the conclusion of this paper. This paper aims to analyze which car type and car parts are most frequently repaired by user and to understand how to extract the frequent pattern from transaction database using FOLD-Growth theory.

2. Related work

Mining frequent patterns are a fundamental and essential problem in many data mining applications[2]. This process analyses customer buying habits by finding associations between the different items that customers place in their “shopping basket”. Association rule mining introduced by Rakesh Agrawal is an important research problem in data mining field.[4] Association rule mining aims at detecting the relationship of tupels in transaction database and serving decision making. The basic algorithm for mining frequent itemsets is Apriori algorithm. [3].

It generates C_k (candidate itemsets) in a pass k (number of itemset) using only L_{k-1} (frequent itemsets) in the previous pass . Hence, C_k can be generated by joining L_{k-1} deleting those that contain any subset that is not frequent[3]. In many cases, the Apriori candidate generate–and–test method significantly reduces the sizes of candidate sets, leading to good performance gain. However, it can suffer from two nontrivial costs. It may need to generate a huge number of candidate sets and need to repeatedly scan the database and check a large set of candidates by pattern matching.[4].

Frequent–pattern growth (FP-growth) which adopts a divide-and-conquer strategy as follows. First, it compress the database representing items into frequent pattern which retain the itemset association information. It then divides the compressed database into a set of conditional database, each associated with one frequent item or ”pattern fragment”, and mines each such database separately[4]. FP-growth method shows that is efficient and scalable for mining both long and short frequent pattern, and is about an order of magnitude faster than Apriori algorithm[5].

The Support-Ordered Tree is constructed by extracting 1-itemsets and 2-itemsets from all transactions and using them to update the Support-Ordered Tree. The Support-Ordered Tree, all transactions are mapped onto a tree structure. Tree is ordered by support count (speed up search) and reduces construction time. To find frequent itemsets, the structure is simply traversed using depth-first search.[6]

3. Data Mining

Data mining refers to extracting or mining knowledge from large amounts of data. Data mining is knowledge mining from database, knowledge

extraction, data/pattern analysis, data archaeology and data dredging. In data mining, there is another popularly used term, knowledge discovery in database or KDD. Knowledge discovery as a process consists of an interactive sequence of seven steps. Data cleaning (to remove noise and inconsistent data) , Data integration (where multiple data sources may be combined) , Data selection (where data relevant to the analysis task are retrieved from database) , Data transformation (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance) , Data mining (an essential process where intelligent methods are applied in order to extract data patterns) , Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures) and Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user).

Data mining involves an integration of techniques from multiple disciplines such as database and data warehouse technology, statistics machine learning, high performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing, and spatial or temporal data analysis.[1]

3.1 Mining frequent patterns and Associations

Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including itemsets, subsequences, and substructures. A frequent itemset typically refers to a set of items that frequently appear together in a transactional data set. Mining frequent patterns leads to the discovery of interesting association.

Association rule mining finds interesting association or correlation relationships among a large set of data items. The discovery of interesting association relationships among huge amounts of business transaction records can help in many business decision making process. Association rule mining is a two- step process.

1 Find all frequent itemsets : By definition, each of these itemsets will occur at least as frequently as a pre-determined minimum support count.

2 Generate strong association rules from the frequent itemsets: By definition, these rules satisfy minimum support and minimum confidence.

Algorithms that calculate association rules work in two phases. In the first phase, all combinations of items that have the required minimum support (called the "frequent itemsets") are discovered. In the second phase, rules of the form X → Y with the specified minimum confidence are generated form frequent itemsets.

Support of a rule is a measure of how frequently the items involved in it occur together. Using probability notation, support (A→B) = P(AUB). Confidence of a rule is the conditional probability of B given A. Using probability notation, confidence (A→B) = P(B/A) , which equal to P (AUB) / P (A). These statistical measures can be used to rank the rules.

3.2 FOLD-growth algorithm

FOLD-growth algorithm is the hybrid version of frequent pattern (FP-growth) algorithm. FOLD-growth, 1-itemsets (L_1) and 2-itemsets (L_2) can be found promptly by using the Support-Ordered Tree algorithm because there is no need to scan the database. We apply FP-growth to discover the rest of the frequent itemsets. The FOLD-growth algorithm is as follows:

1. Use the SOTrieIT to quickly discover L_1 and L_2
2. if L_1= Ø and V L_2 = Ø
3. then Terminate algorithm
4. end if
5. for transaction T Є D do
6. Remove items that will not contribute to L_k, where k>2,using L_1 and L_2
7. Sort items in support descending order
8. Construct/Update the FP-tree with trimmed and sorted T
9. end for
10. Run FP-Growth algorithm on constructed FP-tree

With the SOTrieIT, L_1 and L_2 can be quickly found and they can be used to further prune the transactions that are used to constructed FP-tree. Hence, only one database scan is needed to start building the FP-tree. If L_2 is not found, we can terminate the algorithm immediately because all possibly frequent itemsets, n this case L_1 are already found. Therefore, since FOLD-growth uses SOTrieIT, it can be said to be more incremental than FP-growth to a certain extent even through FOLD-growth itself is not incremental.

3.2.1 Support-Ordered Trie Itemsets (SOTrieIT)

The Support-Ordered Tree is constructed by extracting 1-itemsets and 2-itemsets from all transactions and using them to update the Support-

Ordered Tree. Figure 1 shows the Support-Ordered Tree built from a sample database with just four transactions. The tree is ordered by support count. The bracket number beside a node's label denotes the support count. It nodes are support-ordered and have two levels of nodes (excluding the special ROOT node).

TID	Items
100	AC
200	BC
300	ABC
400	ABCD

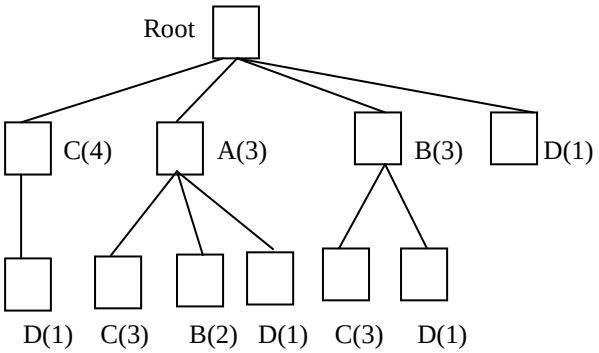

Figure 1: Support-Ordered Tree constructed from sample database

The main strength of the Support-Ordered Tree lies in its speed in discovering L_1 (1-itemsets) and L_2 (2-itemsets). L_1 and L_2 can be found promptly because there is no need to scan the database. In addition, the search (depth-first) can be stopped at a particular level the moment a node representing a non-frequent itemset is found because the nodes are all support-ordered. Another advantage of the SOTrieIT is that it can be constructed online, meaning that each time a new transaction arrives, and the SOTrieIT can be incrementally updated. It requires for less storage space than a tire because it is only two levels deep and can be easily stored in both memory and files. [2]

3.2.2 Frequent Pattern Growth(FP-growth)

Frequent-pattern tree (FP-tree) structure is an extended prefix-tree structure for sorting compressed, crucial information about frequent patterns. An efficient FP-tree based mining method: FP-growth is mining the complete set of frequent patterns by pattern fragment growth.

The first scan of the database which derives the sets of frequent items (1-itemsets) and their support counts (frequencies). The set of frequent items is sorted in order of descending support count. An FP-tree is then constructed is follows: First, create the root of the tree, labeled with “null”. Scan a database is second time. The items in each transaction are processed in L order (i.e., sorted according to descending support count), and a branch is created for each transaction. In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly. The tree obtained after scanning all of the transactions with the associated node-links. In this way, mining frequent patterns in databases is transformed to that of mining the FP-tree. The FP-tree is mined as follows. Start from each frequent length-1 patterns (as an initial suffix pattern), construct its conditional pattern base (a “sub-database,” which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern), then construct its conditional FP-tree, and perform mining recursively on such a tree. The pattern growth is achieved by concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

The FP-growth method transforms the problem of finding long frequent patterns to searching for shorter ones recursively and concatenating the suffix. It uses the least of frequent items as a suffix, offering good selectivity.

4. Flow of the proposed system

This system aims to analyze which car type and car parts are most frequently repaired by user and to understand how to extract the frequent pattern mining from transaction database using FOLD-Growth theory. Figure 2 shows flow of the proposed system.

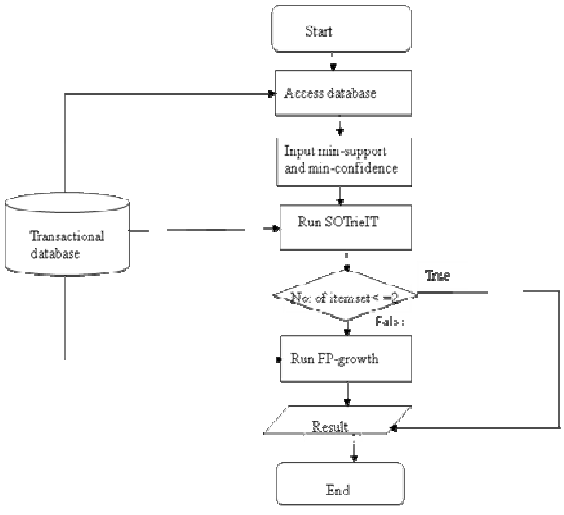

Figure 2: Flow of the proposed system

In Figure 2, accessing database phase consists of inserting new data and deleting transactions. In analysis phase, user input minimum support and minimum confidence thresholds. Then, the system runs the Support-Ordered Tree (SOTrieIT) algorithm. If the user want to know 1-itemsets or 2-itemsets, the system show result that satisfy minimum support and minimum confidence without using the FP-growth algorithm. The SOTrieIT allows 1-itemsets and 2-itemsets to be discovered without computation and without database scans and regardless of the support threshold. So, the SOTrieIT algorithm can quickly discover the access pattern.

If the user want to know more than 2-itemsets, the system apply FP-growth algorithm using the result of SOTrieIT and show result that satisfy minimum support and minimum confidence. The method substantially reduces the search costs.

5. Implementation of the system

The system is implemented for discovering and analyzing car types and car parts which are frequently repaired by the users. It is developed by using Microsoft C#.Net and MySQL Server 2005 for data storage. The system based on transactions using between 1-year. The transaction database consists of transaction ID, Car type, Usage type and Repair parts.

In this system, there are four analysis types such as car type, repair part, car type and repair part and frequent patterns (pair of frequently repaired parts). In Analysis phase, user input minimum support and minimum confidence thresholds. If the user want to know analysis by car type or analysis by repair part or analysis car type and repair part, the system run the SOTrieIT algorithm.

First, all transactions are mapped onto a tree structure. The nodes in the first level are car types and the nodes in the second level are repair parts associated car types in the first level. This mapping involves the extraction of the transaction items and the updating of the tree structure. The tree structure contains all the support counts of items beside the tree node label.

To find frequent itemsets, the structure is simply traversed using depth-first search. If the users want to know car type, the system shows nodes in the first level as result . If the users want to know repair part, the system shows nodes in the second level as result. Otherwise, the users want to know car type and repair part, the system show result by using depth-first search theory. The results of the system must satisfy user defined minimum support and minimum confidence.

If the user analyze car type and repair part, the system displays the layout of the analysis page by running the SOTrieIT algorithm in Figure 3.

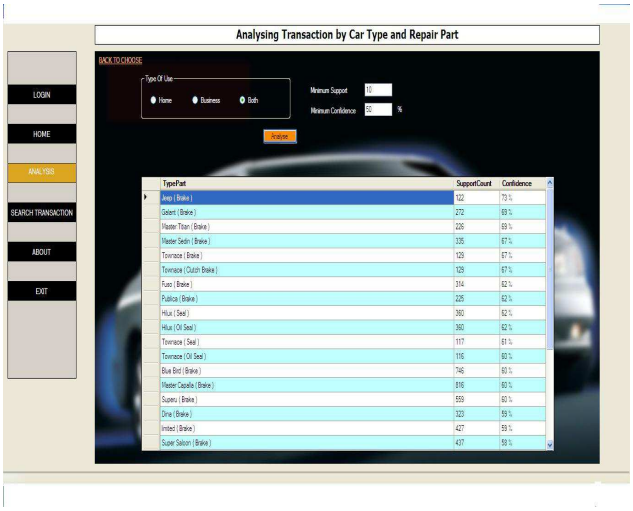

Figure 3: Analysis of car type and repair part

If the user want to know frequent itemsets (pair of frequently repaired parts) for a car type, the system run FP-growth algorithm by using the result of SOTrieIT algorithm. With the SOTrieIT, 1-itemsets and 2-itemsets can be quickly found and they can be used to construct the FP-tree. Hence, only one database scanning is needed to construct the FP-tree. After database scanning, items in the transactions are sorted according to the order of 1-itemsets.These sorted items are used to construct the FP-tree. FP-growth then proceeds to recursively mine FP-tree of decreasing size to generate frequent itemsets without candidate generation and database scanning. Conditional pattern base of the FP-tree which consists of the set of frequent itemsets occurring with the suffix pattern. The layout of the association rules for the analysis by frequent pattern in Figure 4.

Figure 4: Analysis of frequent pattern

The result of the system change depending upon user defined minimum support and minimum confidence values. With minimum support as 10 and

minimum confidence as 50%, the result is as shown in Figure 4. If the user changes minimum support as 15 and minimum confidence is 60%, the result may also change as shown in Figure 5.

Figure 5: Analysis of frequent pattern

Table 1: Experimental results for car type as Corona and usage type as Business

Minimum support	Minimum confidence	Number of association rules
10	**50 %**	**23**
10	**60 %**	**11**
10	**70 %**	**9**
15	**60 %**	**8**
20	**50 %**	**6**

6. Conclusion

This paper proposed Fast frequent itemsets discovery known as the SOTrieIT to enhance the performance of association rule mining. Besides being support independent, incremental, and space efficient, it allows FOLD-growth, tailor-made algorithm for the SOTrieIT to perform faster than FP-growth. The system give user the result for the fast discovery of frequently repaired car type and car parts, The system provide 1-itemsets and 2-itemsets quickly using SOTrieIT algorithm without using FP-growth algorithm. Otherwise, the system gives result for more than 2-itemsets by using FP-growth theory.

7. References

[1] A.Sawmi, R. Agrawal,and T. Imielinski, “Mining association rules between sets of items in large databases.” In proc. of the ACM SIGMOD Conference on Management of Data, pages 207-216, May 1993.

[2] B.Goethals and M.J.Zaki. “Advances in frequent itemset mining implementations”: introduction to fimi03. *In proceeding of the 1st IEEE ICDM* Workshop on Frequent Itemsets Mining Implementation (FIMI'03), Nov 2003.

[3] “Generating a Condensed Representation for Association Rules”, *Journal of Intelligent Information Systems*, 24:1, 29–60, 2005. 2005 Springer Science Business Media, Inc. Manufactured in the Netherlands

[4] Jiawei Han and Micheline Kamber, “Data Mining Concepts and Techniques”

[5] J.Han, J.Pei, and Y.Yin, “Mining Frequent patterns without Candidate Generation,” Proc.ACM SIGMOD Conf.,pp.1-12,2000.

[6] Yew-Kwong Woon, "A Support-Ordered Trie for Fast Frequent Itemset Discovery" Wee-Keong Ng, Member, IEEE Computer Society, and Ee-Peng Lim, Senior Member, IEEE