Performance Analysis of Apriori and FP-Growth Algorithms  
(Association Rule Mining)  

1Alhassan Bala  
Jodhpur National University, India  
alumgarko200@gmail.com  

2Mansur Zakariyya Shuaibu  
Jodhpur National University, India  
mansurzshuaibu@gmail.com  

3Zaharaddeen Karami Lawal  
Jodhpur National University, India  
deenklawal13@gmail.com  

4Rufa’i Yusuf Zakari  
Jodhpur National University, India  
rufaig6@gmail.com  

ABSTRACT  
Association rule mining has become popular among marketers and organizations. In fact, an example of association rule mining is referred to as market basket analysis. The task is to find which items are frequently purchased together. This knowledge can be used by professionals to plan where to place items that are frequently bought together closely to each other, thus helping to improve the sales. It involves the relationships between items in a data set. Association rule mining finds out item sets which has minimum support and are represented in a relatively high number of transactions. These transactions are simply known as frequent item sets. The algorithms that use association rules are divided into two stages, first is to find the frequent sets and the second is to use these frequent sets to generate the association rules. In this paper we used Weka to compare two algorithms (Apriori and FP-growth) based on execution time and database scan parameters used are; number of instances, confidence and support levels it is categorically clear that FP-Growth algorithm is better than apriori algorithm.  

Keywords: - Association, Instances, Support, Confidence, Weka  

1. INTRODUCTION  
Data mining can be seen as the process of digging out important information from different data sources for the purpose of business or organizations decision making based on analysis. It is a process of Knowledge Discovery in Databases process also termed as KDD. It is an interdisciplinary subfield of computer science. It is the process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, statistics and database systems. The overall objective of the data mining process is to mine information from a data set and transform it into an understandable structure for further use [1]. Data mining is the non-trivial mining of implicit previously unknown and potentially useful information about data. Data mining technology offers a user-oriented approach to novel and unknown patterns in the data [2]. Data mining is the computational process of discovery patterns in large data sets using methods like artificial intelligence, machine learning and statistics. The main aim of the data mining process is to mine information from a database and transform it into a
simpler form which can be further used. It involves the raw data analysis step, database and data management aspects, data pre-processing, model and inference considerations, metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. It comprises many techniques like clustering, classification, association rule mining etc [3]. Data mining technology and its application is a hot topic in the international arena currently, and it has been applied in many industries, which demonstrates its advantages and development potential. In the area of information management, it is the only way to achieve development of knowledge retrieval and knowledge management, as long as we integrate data mining technology with artificial intelligence technology, access to user knowledge, literature and other kinds of knowledge. Data mining is to discover and extract hidden information from large databases and vast network information space in the digital library, and the purpose is to help information workers search for the potential association between the data and find the neglected elements, which is very useful to predict trends and make decision [4].

1. Association Rule

In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness [5] [6]. Association rules mining is also used to discover the associations and relations among large data items. Association rules mining is an important tool of data mining research. Currently, association rules mining plays an important role in the area of artificial intelligence, information retrieval, information science, finding patterns in biological databases, mining of knowledge from software engineering metrics, clinical database, web personalization, text mining, statistic, telecommunication, market and risk management, inventory control and many other fields. What can competently catch the significant relationships among data are simple forms of association rules and easily to explanation and understanding [7]. Mining association rules problems from large database has become the most advanced, important and dynamic research contents. The selection of association rule is based on support and confidence. The confidence factor indicates the strength of the implication rules, i.e. the confidence for an association rule is the ratio of the number of transactions that contain \( X \cup Y \) to the number of transactions that contain \( X \); whereas the support factor indicates the frequencies of the occurring patterns in the rule. i.e., the support for an association rule is the percentage of transactions in the database that contain \( X \cup Y \). Given the database \( AB \), the problem of mining association rules involves the generation of all association rules among all items in the given database \( AB \) that have support and confidence greater than or equal to the user specified minimum support and minimum confidence. Typically, large confidence values and a smaller support are used. Rules that satisfy both minimum support and minimum confidence are called strong rules. Since the database is large and users concern about only those frequently purchased items, usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful [7].

Examples: -

(1) Let a database contains 500,000 transactions and 20,000 of those transactions contain both \( X \) and \( Y \) therefore we can conclude that the support of the association \( S = 20,000/500,000 \) which equals to 4%.
(2) if 500,000 of the transactions contain X and 30,000 out of those 50,000 transactions contain Y then we can conclude that the confidence of the association C = 30,000/50000 which equals to 60%.

**Basic example of association rule mining**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{rice, millet}</td>
</tr>
<tr>
<td>2</td>
<td>{rice, oil, firewood, millet}</td>
</tr>
<tr>
<td>3</td>
<td>{beans, oil, rice, millet}</td>
</tr>
<tr>
<td>4</td>
<td>{oil, beans, firewood}</td>
</tr>
<tr>
<td>5</td>
<td>{rice, millet, oil}</td>
</tr>
</tbody>
</table>

- **Itemset**
  - A collection of one or more items
    - Example: {rice, millet, oil}
- **k-itemset**
  - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. \(σ\{rice, millet, oil\} = 3\)
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a \(\text{minsup}\) threshold.

**Association Rule**

- An implication expression of the form \(X \rightarrow Y\), where X and Y are itemsets.
- Example:
  \(\{\text{rice, millet,}\} \rightarrow \{\text{oil}\}\)

**Rule Evaluation Metrics**

- **Support (s)**
  - Fraction of transactions that contain both X and Y
  - From the above support is: -
  \(S = \frac{σ\{\text{rice, millet, oil}\}}{T} = \frac{3}{5} =: 60\%\)
- **Confidence (c)**
  - Measures how often items in Y appear in transactions that contain X.
  - From the above confidence is:
  \(C = \frac{\{\text{rice, millet, oil}\}}{\{\text{rice, millet}\} = \frac{3}{4} = 75\%}\).
2. Apriori Algorithm

Apriori algorithm is the most classical and important algorithm for mining frequent itemsets, proposed by R. Agrawal and R. Srikant in 1994 [8]. Apriori is used to find all frequent itemsets in a given database DB. The key issue of Apriori algorithm is to make multiple passes over the database. It employs an iterative approach known as a breadth-first search (level-wise search) through the search space, where k-itemsets are used to explore (k+1) itemsets. The working of Apriori algorithm is fairly dependent upon the Apriori property which states that “All nonempty subsets of a frequent itemsets must be frequent”. It also described the antimonotonic property which says if the system cannot pass the minimum support test, all its supersets will fail to pass the test. Therefore, if the one set is infrequent then all its supersets are also frequent and vice versa. This property is used to prune the infrequent candidate elements. In the beginning, the set of frequent 1-itemsets is found. The set of that contains one item, which satisfy the support threshold, is denoted by L. In each subsequent pass, we begin with a seed set of itemsets found to be large in the previous pass. This seed set is used for generating new potentially large itemsets, called candidate itemsets, and count the actual support for these candidate itemsets during the pass over the data. At the end of the pass, we determine which of the candidate itemsets are actually large (frequent), and they become the seed for the next pass. Therefore, L is used to find L!, the set of frequent 2-itemsets, which is used to find L and so on, until no more frequent k-itemsets can be found [8]. Following are basic steps for mining frequent elements: -

- **Generate and test:** In this first you find the 1-itemset of frequent elements named L through scanning the database and eliminating all the elements from C which in turn cannot satisfy the minimum proposed support criteria.

- **Join step:** To achieve the next level elements Ck join the former frequent elements through self-join that is Lk-1* Lk-1 well-known as Cartesian product of Lk-1. thus This phase produces new candidate k-itemsets based on joining Lk-1 by itself which has obtained in the previous iteration. Let Ck indicate candidate k-itemset and Lk serve as the frequent k-itemset.

- **Prune step:** in this step Ck is the superset of Lk therefore members of Ck may or may not be frequent but all K ' 1 frequent itemsets are incorporated in Ck thus prunes the Ck to get K frequent itemsets with the support of Apriori property. Thus this step removes some of the candidate k-itemsets by means of the Apriori property. A scan of the database to obtain the count of each and every candidate in Ck would result in the determination of the table or level of Lk (that is all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk). Ck, can be huge, and so this could involve grave computation. By contracting the size of Ck, Apriori property will be used as follows. Any (k-1)-itemset that is not frequent cannot be used a subset of a frequent k-itemset. therefore, if any (k-1)-subset of candidate k-itemset is not in Lk-1 then the candidate cannot be frequent therefore can be removed from Ck. Step 2 and 3 will be repeated till no other candidate set new is generated. There is no doubt that Apriori algorithm effectively finds the all the frequent elements from the database. But as the size of the database growths with the number of items then:
• More search space is required and the cost of I/O will surely increase.
• Number of database scan is increased resulting in candidate generation thereby increasing the computational cost.

3. FP-Growth Algorithm

FP-growth algorithm is an efficient method of mining all frequent itemsets without candidate generation. FP-growth utilizes a combination of the vertical and horizontal database layout to store the database in main memory. Instead of storing the cover for every item in the database, it stores the actual transactions from the database in a tree structure and every item has a linked list going through all transactions that contain that item. Every node additionally stores a counter, which keeps track of the number of transactions that share the branch through that node. Also a link is stored, pointing to the next occurrence of the respective item in the FP-tree, such that all occurrences of an item in the FP-tree are linked together. Additionally, a header table is stored containing each separate item together with its support and a link to the first occurrence of the item in the FP-tree [9]. In the FP-tree, all items are ordered in support descending order, because in this way, it is hoped that this representation of the database is kept as small as possible since all more frequently occurring items are arranged closer to the root of the FP-tree and thus are more likely to be shared.

The algorithm mines the frequent itemsets by using a divide and-conquer strategy as follows: FP-growth first compresses the database representing frequent itemset into a frequent-pattern tree, or FP-tree, which retains the itemset association information as well. The next step is to divide a compressed database into set of conditional

All nodes correspond to items have a counter

- FP-Growth reads 1 transaction at a time and maps it to a path

Fixed order is used, so paths can overlap when databases (a special kind of projected database), each associated with one frequent item. Finally, mine each such database separately. Particularly, the construction of FP-tree and the mining of FP-tree are the main steps in FP-growth algorithm [9].

FP-Growth: allow you to discover frequent itemset without candidate itemset generation. It consists of two step approach:

Step 1: Build a compact data structure called the FP-tree
- Built using 2 passes over the dataset.

Step 2: Extracts frequent itemsets directly from the FP-tree
- Traversal through FP-Tree

General Data Structure of FP-Tree

- transactions share items (when they have the same prefix).
- In this case, counters are incremented
- Pointers are maintained between nodes containing the same item, creating singly linked lists (dotted lines)
- The more paths that overlap, the higher the compression. FP-tree may fit in memory.
- Frequent itemsets extracted from the FP-Tree.

Below are clear examples of how FP-Growth works: -
Advantages of FP-Growth
- It requires only 2 passes on a data-set
- Data-set are compressed so as to reduce the size consume.
- It requires no candidate generation

Disadvantages of FP-Growth
- FP-Tree is expensive to build
- if support threshold is high time is wasted since pruning is done on a single item.

4. Methodology
These two association rule mining algorithms (Apriori and FP-Growth) will be implemented in WEKA. Weka (Waikato Environment for Knowledge Analysis) is a popular collection of...
machine learning software developed in Java, at the University of Waikato. WEKA is an open source software accessible under the GNU General Public License. The Weka workbench comprises of a collection of algorithms and visualization tools for predictive modelling and data analysis, together with graphical user interfaces for easy access to this functionality. Weka software comprise of tools for data pre-processing, regression, association rules, classification, clustering, and visualization tools. The Supermarket and voter datasets will be used for this experimentation. These datasets contain 4627 instances and 217 attributes respectively. The performance of Apriori and FP-Growth algorithms will be analyzed based on execution time and number of scans for different instances.

Figure 1: Pre-processing of data sets. This is the stage where editing and loading of data in the Weka platform takes place.
Figures below show the execution time of both apriori and FP-Growth based on the number of instances.

Figure 2: This phase is the result of data set implemented using (463) number of instances.

Figure 3: This phase is the result of data set implemented using (925) number of instances.
Figure 4: This phase is the result of data set implemented using (1541) number of instances.
Figure 5: This is the environment that shows the time taken by both Apriori Algorithm and Fp-growth Algorithm using the above number of instance.
Table 1: shows that when the number of instances increases, execution time also increases. The Apriori and FP-Growth algorithms take 23 seconds and 2 seconds when number of instances are 463, 30 seconds and 2 seconds when number of instances are 925 also 107 seconds and 13 seconds when number of instances are 1541. These results categorically cleared show that, FP-Growth Algorithm is faster than Apriori Algorithm in terms of the number of instances.

Figures below show the execution time of both apriori and FP-Growth based on different confidence level.

Figure 6: This phase is the result of data set implemented using different confidence level.
Figure 7: This is the environment that shows the time taken by both Apriori Algorithm and Fp-growth Algorithm using the above number of confidence levels.
Table 2: shows the execution time using different confidence level based on the result of the above table and implementation figure it also shows that, FP-Growth Algorithm outperform Apriori Algorithm.

**Figures below show the execution time of both apriori and FP-Growth based on different support level.**

![Figure 8](image.png)

*Figure 8: This phase is the result of data set implemented using different support level.*
Figure 9: This is the environment that shows the time taken by both Apriori Algorithm and FP-growth Algorithm using the above number of support levels.

<table>
<thead>
<tr>
<th>Support level</th>
<th>Execution Time per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Apriori</td>
</tr>
<tr>
<td>20%</td>
<td>161</td>
</tr>
<tr>
<td>50%</td>
<td>34</td>
</tr>
<tr>
<td>60%</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 2: shows the execution time using different support level based on the result of the above table and implementation figure it also shows that, FP-Growth Algorithm outperform Apriori Algorithm.
5. Conclusion

This paper focused on comparison and analysis of association rule mining based on certain parameters which include number of instances, different confidence levels and of course different support levels implemented in Weka taking supermarket data set as the sample data. Two association rule techniques were used which include FP-growth Algorithm and Apriori algorithm. It was categorically cleared that, FP-growth Algorithm is faster in terms of execution time as compared to Apriori Algorithm. This shows that the amount of time it taken to run to completion is less than the amount of time needed by Apriori algorithm.

References


