Relative Competence Centered Scrutiny and Implementation of Apriori, FP – Growth and Mapreduce Algorithms

Manpreet Kaur¹, Prof.(Dr.)Vishal Goyal²

¹Research Scholar (M.Phil.), Dept. of Comp. Science, Punjabi University, Patiala, Punjab, India.
²Professor, Dept. of Computer Science, Punjabi University, Patiala, Punjab, India.

*Corresponding Author: Manpreet Kaur, Research Scholar (M.Phil.), Dept. of Comp. Science, Punjabi University, Patiala, Punjab, India.

Abstract: The major rise in data collection and storage has raised the necessity for much more powerful data analysis tools. The data collected in huge databases needs to be handled effectively and efficiently. The important and highly critical decisions are made not on the basis of information rich data stored in databases but instead on a decision maker’s instinct merely because of the absence of the tools capable of extracting the valuable knowledge from vast amount of the data. Currently expert systems depends on users to manually input knowledge into knowledge bases. This process is often time consuming, expensive, and bias. The problem with data mining algorithms are their non-capability of dealing with non-static, and unbalanced data. There is a need for constantly updating the models to handle data velocity or new incoming data.

The objectives of the research paper is to implement the three popular data mining algorithms (Apriori algorithm, FP – Growth algorithm, and Map Reduce algorithm) using appropriate programming tool (preferably Java). The paper also perform comparative analysis of the three algorithms under study via measuring efficiency in terms of time. The paper also elaborates on analysis of all three algorithms on the basis of performance evaluation using accuracy metric.

Keywords: Apriori algorithm, data mining, FP – Growth algorithm, Map Reduce algorithm.

1. INTRODUCTION

Data mining deals with the kind of patterns that can be mined. On the basis of the kind of data to be mined, there are two categories of functions involved in data mining mentioned as under.

- Descriptive
- Classification and Prediction

1.1. Descriptive Function

The descriptive function deals with the general properties of data in the database and are mentioned as under [1].

1.1.1. Class/Concept Description

Class/Concept refers to the data to be associated with the classes or concepts. For example, in a company, the classes of items for sales include computer and printers, and concepts of customers include big spenders and budget spenders. Such descriptions of a class or a concept are called class/concept descriptions. These descriptions can be derived by the following two ways.

- Data Characterization – Data characterization refers to summarizing data of class under study. This class under study is called as Target Class.
- Data Discrimination – It refers to the mapping or classification of a class with some predefined group or class.
1.1.2. **Mining of Frequent Patterns**

Frequent patterns are those patterns that occur frequently in transactional data (FP - Growth). Here is the list of kind of frequent patterns.

- **Frequent Item Set** – It refers to a set of items that frequently appear together, for example, milk and bread.
- **Frequent Subsequence** – A sequence of patterns that occur frequently such as purchasing a camera is followed by memory card.
- **Frequent Sub Structure** – Substructure refers to different structural forms, such as graphs, trees, or lattices, which may be combined with item-sets or subsequences.

1.1.3. **Mining of Association**

Associations are used in retail sales to identify patterns that are frequently purchased together. This process refers to the process of uncovering the relationship among data and determining association rules.

For example, a retailer generates an association rule that shows that 70% of time milk is sold with bread and only 30% of times biscuits are sold with bread.

1.1.4. **Mining of Correlations**

It is a kind of additional analysis performed to uncover interesting statistical correlations between associated-attribute-value pairs or between two item sets to analyze that if they have positive, negative or no effect on each other.

1.1.5. **Mining of Clusters**

Cluster refers to a group of similar kind of objects. Cluster analysis refers to forming group of objects that are very similar to each other but are highly different from the objects in other clusters.

1.2. **Classification and Prediction**

Classification is the process of finding a model that describes the data classes or concepts [2]. The purpose is to be able to use this model to predict the class of objects whose class label is unknown. This derived model is based on the analysis of sets of training data. The derived model can be presented as classification (if-then) rules, decision trees, mathematical formulae, and neural networks [3].

The list of functions involved in these processes are mentioned as under.

1.2.1. **Classification**

It predicts the class of objects whose class label is unknown. Its objective is to find a derived model that describes and distinguishes data classes or concepts. The Derived Model is based on the analysis set of training data i.e. the data object whose class label is well known.

1.2.2. **Prediction**

It is used to predict missing or unavailable numerical data values rather than class labels. Regression Analysis is generally used for prediction. Prediction can also be used for identification of distribution trends based on available data [3].

1.2.3. **Outlier Analysis**

Outliers may be defined as the data objects that do not comply with the general behavior or model of the data available.

1.2.4. **Evolution Analysis**

Evolution analysis refers to the description and model regularities or trends for objects whose behavior changes over time.
2. DATA MINING ALGORITHMS

The research work covers the detailed study and implementation of three data mining algorithms: Apriori algorithm, FP-Growth algorithm, and MapReduce algorithm.

2.1. Apriori Algorithm

In data mining, Apriori algorithm [2, 3] is a traditional algorithm used for learning association rules. Association rules are referred to those statements which are used to find the relation between the data items of the database. Apriori algorithm is used to mine the frequent data items and corresponding association rule in the database of the transactions. Apriori algorithm is based upon the bottom up strategy in which the common subset of data items is expanded to add one more item at a time and then it is checked against the minimum support. Minimum support is the minimum value used to search frequent patterns that satisfy this restriction [4, 5]. The mining of association rule from huge amount of data assist the companies in taking important decisions regarding their business. This rule is used in many fields such as storage planning, analysis of customer shopping, good shelves design etc [4, 12].

Flowchart for Apriori algorithm

The working of Apriori algorithm is shown in the flowchart below depicted in Fig. 1.

![Flowchart for Apriori algorithm](image-url)
Example of Apriori Algorithm

The data in the Table 1 is taken as input where “T.Id” refers to Transaction_Id, “Items bought” shows the items bought together. The minimum support for example under study is set to 3.

Table 1. Table shows the Input data

<table>
<thead>
<tr>
<th>T.Id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cookies, tea, cake</td>
</tr>
<tr>
<td>2</td>
<td>Bread, tea, butter</td>
</tr>
<tr>
<td>3</td>
<td>Cookies, Bread, tea, butter</td>
</tr>
<tr>
<td>4</td>
<td>Bread, butter</td>
</tr>
<tr>
<td>5</td>
<td>pan cakes</td>
</tr>
</tbody>
</table>

Calculate the number of times each item appears in the table.

Table 2. Table displays the items against frequency of its occurrence

<table>
<thead>
<tr>
<th>Items bought</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookies</td>
<td>2</td>
</tr>
<tr>
<td>Bread</td>
<td>3</td>
</tr>
<tr>
<td>Tea</td>
<td>3</td>
</tr>
<tr>
<td>Butter</td>
<td>3</td>
</tr>
<tr>
<td>Pan cakes</td>
<td>1</td>
</tr>
<tr>
<td>Cake</td>
<td>1</td>
</tr>
</tbody>
</table>

Only items having occurrence equal to or greater than 3 are moved to next stage

Table 3. Table shows the qualified items

<table>
<thead>
<tr>
<th>Items bought</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>3</td>
</tr>
<tr>
<td>Tea</td>
<td>3</td>
</tr>
<tr>
<td>Butter</td>
<td>3</td>
</tr>
</tbody>
</table>

Reassemble the three items with possible combinations.

Table 4. Table shows the possible combinations of shortlisted items

<table>
<thead>
<tr>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread, tea</td>
</tr>
<tr>
<td>Bread, butter</td>
</tr>
<tr>
<td>Tea, butter</td>
</tr>
</tbody>
</table>

Calculate the occurrences of combinations in Table 4

Table 5. Table shows the occurrences of combinations

<table>
<thead>
<tr>
<th>Items bought</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread, tea</td>
<td>2</td>
</tr>
<tr>
<td>Bread, butter</td>
<td>3</td>
</tr>
<tr>
<td>Tea, butter</td>
<td>2</td>
</tr>
</tbody>
</table>

Discard the products having minimum support less than 3.

Table 6. Final output

<table>
<thead>
<tr>
<th>Items bought</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread, butter</td>
<td>3</td>
</tr>
</tbody>
</table>

Only one item set with frequent item set is left with support 3.

2.2. FP-Growth Algorithm

Frequent patterns are item sets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a frequent item.
set. Pattern mining can be applied on various types of data such as transaction databases, sequence databases, streams, strings, spatial data, graphs, etc. Frequent patterns are those patterns that occur frequently in transactional data [4, 5].

The most popular algorithm for pattern mining is the FP-Growth algorithm. The main idea of the algorithm is to use a divide and conquer strategy. Compress the database which provides the frequent sets; then divide this compressed database into a set of conditional databases, each associated with a frequent set and apply data mining on each database. It is designed to be applied on a transaction database to discover patterns in transactions made by customers in stores. But it can also be applied in several other applications [6, 11, 13].

**Flowchart for FP-Growth algorithm**

The flowchart for FP-Growth algorithm is shown below in Fig. 2.

![Flowchart](image)

**Fig2. The flowchart depicts the working of FP-Growth algorithm**

**Pros and cons of FP-Growth**

The pros and cons related to FP-Growth algorithm are mentioned as under [10].

**Pros**

- The major advantage of the FP-Growth algorithm is that it takes only two passes over the data set.
- The FP-Growth algorithm compresses the data set because of overlapping of paths.
The candidate generation is not required.

The working of the FP-Growth algorithm is much faster as compared to the Apriori algorithm.

Cons

- The FP-Growth algorithm may not fit into the memory.
- The FP-Growth algorithm is expensive to construct. It consumes time to build. But once it is done with construction, itemsets can be read off easily.
- Enormous time is wasted when support threshold is high as pruning can be practiced only on single items.
- The process of calculating the support can be carried out only after the entire data set is added to the FP-Tree.

2.3. Mapreduce Algorithm

MapReduce is parallel programming paradigm that enables the distributed processing of massive data sets across the large cluster of commodity servers. The concept of MapReduce is easily understandable. The data which is given as input is usually very large in size and to complete it in specific time, it has to be distributed over the thousands of servers [6, 7, 9].

The Processing of MapReduce algorithm divides into six steps:

2.3.1. Job Submission

When the user writes a basic program for the creation of new JobClient, the JobClient send the request to JobTracker to get a new JobID. Then the JobClient will check whether the input and output directories are correct. After this, the JobClient will store the resources like the number of input data fragmentations, the configuration files and mapper/reducer JAR files to HDFS. Basically, JAR files will be keep as several backups. After all of this, the JobClient will submit a job request to JobTracker [8].

2.3.2. Job Initialization

JobTracker is the master of the system so it will take many JobClient requests. All the requests are placed in a queue which is managed by the job scheduler. Once the JobTracker starts to initialize, its job is to make a JobInProgress case to signify a job. The JobTracker must retrieve the input data from HDFS and to decide on the number of the map tasks. The reduce tasks and TaskInProgress are determined by the parameters in the configuration files.

2.3.3. Task Allocation

Firstly, the TaskTracker has to be launched which is responsible for the map and reduce tasks. The TaskTracker will send the message to the JobTracker for the completion of the task. When the job queue of the JobTracker is not empty, the TaskTracker will receive the tasks to do. Because of the shortage of TaskTracker computing capability, it can handle limited tasks. The TaskTracker basically have two task slots i.e. map task and reduce task. During task allocation, the JobTracker initially use the map task. Once the map task slot is empty it will receive another job task. When it is full, then the reduce task will receive the tasks to do.

2.3.4. Map Tasks Execution

In the map TaskTracker, there is a series of operations for the completion of the tasks. Initially, the map TaskTracker will make a TaskInProgress object to schedule and monitor the tasks. Secondly, the map TaskTracker will copy the JAR files and linked configuration files from HDFS to the local working directory. When all these things are completed, the TaskTracker will create a new TaskRunner to run the map task. The TaskRunner can launch a distinct JVM and will begin the map task within to execute map() function. During the execution, the map task can communicate with TaskTracker to report task progress until all the tasks are completed. At that point, all the computing results are stored within the local disk.
2.3.5. Reduce Tasks Execution

When the task execution of map tasks is completed, the JobTracker will follow the same procedure with reduce TaskTracker to allocate the tasks. The reduce TaskTracker also execute the reduce() function in separate JVM. At that point, the reduce task will download the results from map TaskTracker. When all the map tasks completed their execution, the JobTracker notify the reduce TaskTracker to start the execution. The same way, reduce task will communicate about the progress with TaskTracker until all the tasks are finished.

2.3.6. Job Completion

At each stage of reduce execution, all the results of reduce task will stored in the temporary file in HDFS. When the execution of all reduce tasks is completed, all these temporary files are combined together into the final output file. The JobTracker received the message of completion and the JobClient notify the user and display the required information.

2.4. Algorithm For Mapreduce Algorithm

The step wise working of MapReduce algorithm in mentioned below [14].

- The incoming data can be alienated into n number of modules which depends upon the amount of input data and processing power of the individual unit.
- All these fragmented modules are then passed over to mapper function where these modules undergo simultaneous parallel processing.
- Thereafter, shuffling is conducted in order to gather similar looking patterns.
- Finally, reducer function is called which is responsible for getting the ultimate output in a reduced form.
- Moreover, this technique is scalable and depending upon increase in the data to be processed, the processing units can be further extended.

The working of MapReduce algorithm is shown in the flowchart depicted in Fig. 3 below.

![Flowchart](image.png)

**Fig 3. The flowchart depicts the detailed working of MapReduce algorithm**
3. CONTRIBUTION AND IMPLEMENTATION

A database titled “commondatabase.data” shown in Fig. 4 and Fig. 5 has been constructed which consists of 3196 rows and 37 columns i.e. each entry consists of 37 numbers.

Fig 4. The figure displays the snapshot of constructed database

Fig 5. The figure displays the second snapshot of constructed database
Evaluating working of Apriori algorithm and FP – Growth algorithm at minimum support of 80% (minsup>=0.8)

The constructed database is given as input to the Apriori algorithm program developed in Java to find out the frequent itemsets of sizes ranging from 1 to 14 with minimum support value of 80% (minsup=0.8%). As the total number of entries in the database is 3196, the 80% of this value is 2556.8 (3196 * .80). So the extracted answer will contain only those itemsets whose support value occurrence is above 2556.8.

Fig. 6 shows the result obtained in accordance with itemsets of size 1. The first row extracted is as follows.

\[3\] <0.88829787234042562839>

Here [3] refers to the item been scanned.

0.8882978723404256 shows the support value of item [3] which is 88.82978% and is clearly above the minimum support value of 80% or is above minsup value of .80.

\[9, 66\] <0.84918469310387992714>

Here [9, 66] refers to the itemset been scanned.

Fig. 7 shows the result obtained in accordance with itemsets of size 2. The first row in the result extracted is as follows.

\[9, 66\] <0.889924906132665832874>

Here [9, 66] refers to the itemset been scanned.
0.8494993742177722 shows the support value of itemset [9, 66] which is 84.94993742% and is clearly above the minimum support value of 80% or is above minsup value of .80.

Fig. 7 shows the result obtained in accordance with itemsets of size 2.

Fig. 8 shows the result obtained in accordance with itemsets of size 3. The first row in the result extracted is as follows.

[48, 52, 66] <0.895494367959952862>

Here [48, 52, 66] refers to the itemset been scanned.

0.89549436795995 shows the support value of itemset [48, 52, 66] which is 89.549436795995% and is clearly above the minimum support value of 80% or is above minsup value of .80.
Relative Competence Centered Scrutiny and Implementation of Apriori, FP – Growth and Mapreduce Algorithms

Fig. 8 shows the result obtained in accordance with itemsets of size 3.

Fig. 9 shows the result obtained in accordance with itemsets of size 4. The first row in the result extracted is as follows.

\[ [29, 48, 56, 66] < 0.8416770963704631 \ 2690] \]

Here \([29, 48, 56, 66]\) refers to the itemset been scanned. 0.8416770963704631 shows the support value of itemset \([29, 48, 56, 66]\) which is 84.16770963704631% and is clearly above the minimum support value of 80% or is above minsup value of .80.
Similarly, the frequent itemsets of size 5 to 10 can be obtained.

Fig. 10 shows the result obtained in accordance with itemsets of size 10. The first row in the result extracted is as follows.

[7, 29, 36, 40, 48, 52, 58, 60, 62, 66] <0.8050688360450563 2573>

Here [7, 29, 36, 40, 48, 52, 58, 60, 62, 66] refers to the itemset been scanned.

0.8050688360450563 shows the support value of itemset [7, 29, 36, 40, 48, 52, 58, 60, 62, 66] which is 80.50688360450563 % and is clearly above the minimum support value of 80% or is above minsup value of .80. Fig. 10 also indicates that there are 2 unique itemsets of size 11 created from itemsets of size 10.

**Fig9. Shows the result obtained in accordance with itemsets of size 4**
The total time taken in the entire process is recorded as 103.432 seconds (103432 milliseconds) at minsup = .80.

**Fig. 10.** Shows the result obtained in accordance with itemsets of size 10

The minsup value can be dynamically altered as desired and the operation can be conducted accordingly.

The same database shown in Fig. 4 and Fig. 5 is given as input to the source code of FP – Growth algorithm designed in Java platform. The minsup value has been set to .80%.

Fig. 11 shows the result obtained in accordance with itemsets of size 1. The first row in the result extracted is as follows.

```
[3] < 0.88829787234042562839
```

Here [3] refers to the itemset been scanned. 0.8882978723404256 shows the support value of itemset [3] which is 88.82978723404256 % and is clearly above the minimum support value of 80% or is above minsup value of .80. Fig. 5.13 also indicates that there are 76 itemsets of size 1 out of which 19 itemsets of size 1 qualified the set condition of minimum support.
Fig. 11. Shows the result obtained in accordance with itemsets of size 1

Fig. 12 shows the result obtained in accordance with itemsets of size 2. The first row in the result extracted is as follows.

\[0.81257822277784731\]

Here [29, 44] refers to the itemset been scanned.

0.81257822277784731 shows the support value of itemset [29, 44] which is 81.257822277784731 % and is clearly above the minimum support value of 80% or is above minsup value of .80. Fig. 12 also indicates 141 itemsets of size 2 qualified the set condition of minimum support i.e. equal to or greater than minsup. The itemset [29, 44] appeared frequently 2597 times in the database under study.

Fig. 12. Shows the result obtained in accordance with itemsets of size 2
Relative Competence Centered Scrutiny and Implementation of Apriori, FP – Growth and Mapreduce Algorithms

Fig. 13 shows the result obtained in accordance with itemsets of size 3. The first row in the result extracted is as follows.

\[[48, 52, 66] \quad <0.84599436795995 \quad 2862>\]

Here [48, 52, 66] refers to the itemset been scanned.

0.84599436795995 shows the support value of itemset [48, 52, 66] which is 84.599436795995 % and is clearly above the minimum support value of 80% or is above minsup value of .80. Fig. 5.15 also indicates 566 itemsets of size 3 qualified the set condition of minimum support i.e. equal to or greater than minsup. The itemset [48, 52, 66] appeared frequently 2862 times in the database under study.

Fig. 13. Shows the result obtained in accordance with itemsets of size 3

Similarly frequent itemsets can be obtained for size 4 to size 11.

Fig. 14 shows the result obtained in accordance with itemsets of size 10. The first row in the result extracted is as follows.

\[[7, 29, 36, 48, 52, 56, 58, 60, 66] \quad <0.802565708468572>\]

Here [7, 29, 36, 48, 52, 56, 58, 60, 66] refers to the itemset been scanned.

0.802565708468572 shows the support value of itemset [7, 29, 36, 48, 52, 56, 58, 60, 66] which is 80.2565708468572 % and is clearly above the minimum support value of 80% or is above minsup value of .80. Fig. 14 also indicates 78 itemsets of size 10 qualified the set condition of minimum support i.e. equal to or greater than minsup. The itemset [7, 29, 36, 48, 52, 56, 58, 60, 66] appeared frequently 2565 times in the database under study.

Fig. 14 also shows that 2 itemsets of size 11 also qualify the set condition of minimum support.
Fig 14. Shows the result obtained in accordance with itemsets of size 10

The total time taken to conduct frequent mining using FP – Growth algorithm is 103.107 seconds (103107milliseconds).

Table 7 below show the figures obtained at minimum support of 80% (minsup=0.8) via running Apriori algorithm on database under study “commondatabase.dat”.

Table 7. Displays the figures obtained on running Apriori algorithm at minsup= 0.8

<table>
<thead>
<tr>
<th>Size</th>
<th>Created Itemsets</th>
<th>Frequent Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>76</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>171</td>
<td>141</td>
</tr>
<tr>
<td>3</td>
<td>821</td>
<td>566</td>
</tr>
<tr>
<td>4</td>
<td>2360</td>
<td>1383</td>
</tr>
<tr>
<td>5</td>
<td>4478</td>
<td>2130</td>
</tr>
<tr>
<td>6</td>
<td>5583</td>
<td>2104</td>
</tr>
<tr>
<td>7</td>
<td>4445</td>
<td>1314</td>
</tr>
<tr>
<td>8</td>
<td>2189</td>
<td>481</td>
</tr>
<tr>
<td>9</td>
<td>617</td>
<td>85</td>
</tr>
<tr>
<td>10</td>
<td>78</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Relative Competence Centered Scrutiny and Implementation of Apriori, FP – Growth and Mapreduce Algorithms

**Fig15.** Figure displays the figures obtained on running Apriori algorithm at minsup = 0.8

Table 8 below show the figures obtained at minimum support of 80% (minsup=0.8) via running FP-Growth algorithm on database under study “commondatabase.dat”.

**Table8.** Displays the figures obtained on running Apriori algorithm at minsup = 0.8

<table>
<thead>
<tr>
<th>Size</th>
<th>Created Itemsets</th>
<th>Frequent Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>76</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>171</td>
<td>141</td>
</tr>
<tr>
<td>3</td>
<td>821</td>
<td>566</td>
</tr>
<tr>
<td>4</td>
<td>2360</td>
<td>1383</td>
</tr>
<tr>
<td>5</td>
<td>4478</td>
<td>2130</td>
</tr>
<tr>
<td>6</td>
<td>5583</td>
<td>2104</td>
</tr>
<tr>
<td>7</td>
<td>4445</td>
<td>1314</td>
</tr>
<tr>
<td>8</td>
<td>2189</td>
<td>481</td>
</tr>
<tr>
<td>9</td>
<td>617</td>
<td>85</td>
</tr>
<tr>
<td>10</td>
<td>78</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Fig16.** Figure displays the figures obtained on running Apriori algorithm at minsup = 0.8
Table 9 shows the created itemsets and frequent itemsets values obtained at different sizes by both the algorithms.

Table 9. Comparative table at minsup >= 0.8

<table>
<thead>
<tr>
<th>Size</th>
<th>Apriori Created itemsets</th>
<th>Apriori Frequent Itemsets</th>
<th>FP-Growth Created itemsets</th>
<th>FP-Growth Frequent Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>76</td>
<td>19</td>
<td>76</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>171</td>
<td>141</td>
<td>171</td>
<td>141</td>
</tr>
<tr>
<td>3</td>
<td>821</td>
<td>566</td>
<td>821</td>
<td>566</td>
</tr>
<tr>
<td>4</td>
<td>2360</td>
<td>1383</td>
<td>2360</td>
<td>1383</td>
</tr>
<tr>
<td>5</td>
<td>4478</td>
<td>2130</td>
<td>4478</td>
<td>2130</td>
</tr>
<tr>
<td>6</td>
<td>5583</td>
<td>2104</td>
<td>5583</td>
<td>2104</td>
</tr>
<tr>
<td>7</td>
<td>4445</td>
<td>1314</td>
<td>4445</td>
<td>1314</td>
</tr>
<tr>
<td>8</td>
<td>2189</td>
<td>481</td>
<td>2189</td>
<td>481</td>
</tr>
<tr>
<td>9</td>
<td>617</td>
<td>85</td>
<td>617</td>
<td>85</td>
</tr>
<tr>
<td>10</td>
<td>78</td>
<td>4</td>
<td>78</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 17 shows the graphical representation of created itemsets and frequent itemsets values obtained at different sizes by both the algorithms.

![Comparative representation at minsup >= 0.8](image)

Fig 17. Shows the graphical representation of created itemsets and frequent itemsets values at different sizes of both the algorithms

4. PERFORMANCE EVALUATION OF A PRIORI ALGORITHM AND HADOOP BASED MAPREDUCE ALGORITHM

Among the two data mining algorithms, Apriori algorithm and FP – Growth algorithm, the Apriori algorithm dominates in performance when evaluated in terms of time taken. This section of the research paper compares the efficiency of Apriori algorithm with the MapReduce algorithm in terms of time consumed. The source code for Apriori algorithm has been constructed using C language. A small excel file titled “numbers.csv” shown in Fig. 18 is provided as input to both the algorithms.
The comparative graph of Apriori and Hadoop is shown below in Fig. 19. The result shows that MapReduce algorithm is much more speedy and efficient in mining as compared to Apriori algorithm.

So, it can be concluded that the working of MapReduce algorithm is much better than Apriori algorithm.

5. Conclusion

The research work conducted has proved that among the two conventional data mining algorithms, Apriori algorithm and FP – Growth algorithm, the performance of Apriori algorithm is much better than FP – Growth algorithm when we talk about efficiency in terms of time taken. The test has been conducted on three minimum support values of 80%. The Apriori algorithm has proved its worth upon FP – Growth algorithm as evaluated and proved in section 3.

Thereafter, the comparison of Apriori algorithm is done with MapReduce algorithm to conduct the performance evaluation of both the algorithms. It is proved in the evaluation that MapReduce algorithm takes much less time in completing the operation as compared to Apriori algorithm.

Therefore, it can be concluded that out of three algorithms under study in this research paper, the MapReduce turns out to be the best in efficiency in terms to completing any particular operation relevant to mining.
REFERENCES


