A comparative study of Frequent pattern mining Algorithms: Apriori and FP Growth on Apache Hadoop

Ahilandeeswari.G.\textsuperscript{1}
\par
\textsuperscript{1}Research Scholar, Department of Computer Science, NGM College, Pollachi, India,

Dr. R. Manicka Chezian\textsuperscript{2}
\par
\textsuperscript{2}Associate Professor, Department of Computer Science, NGM College, Pollachi, India,

Abstract— In Data Mining Research, Frequent pattern (itemset) mining plays an important role in association rule mining. The Apriori and FP-growth algorithms are the most famous algorithms which can be used for Frequent Pattern mining. The analysis of literature survey would give the information about what has been done previously in the same area, what is the current trend and what are the other related areas. This paper explains the concepts of Frequent Pattern Mining and two important approaches candidate generation approach and without candidate generation. The paper describes methods for frequent item set mining and various improvements in the classical algorithms “Apriori” and “FP growth” for frequent item set generation. Apache Hadoop is a major innovation in the IT market place last decade. From humble beginnings Apache Hadoop has become a world-wide adoption in data centers. It brings parallel processing in hands of average programmer. This paper presents a literature review of frequent pattern mining algorithms on Hadoop.

Keywords: Data mining, Association rule, Frequent itemset mining, Apriori and FP growth algorithm.

1. INTRODUCTION

Frequent pattern mining \cite{1} plays a major field in research since it is a part of data mining. Many research papers, articles are published in the field of Frequent Pattern Mining (FPM). This chapter details about frequent pattern mining algorithm, types and extensions of frequent pattern mining, association rule mining algorithm, rule generation, suitable measures for rule generation. 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. Frequent pattern mining was first proposed by Agarwal et. al. \cite{1} for market basket analysis in the form of association rule mining. Frequent Itemset mining came into existence where it is needed to discover useful patterns in customer’s transaction database. A customer’s transaction database is a sequence of transactions (T=t1…tn), where each transaction is an itemset (ti \subseteq I). An itemset with k elements is called a k-itemset. An itemset is frequent if its support is greater than a support threshold, denoted by \text{min sup}. The frequent itemset problem is to find all frequent itemset in a given transaction database. The first and most important solution for finding frequent itemsets, is the Apriori algorithm. The fundamental frequent pattern algorithms are classified into two ways as follows:

1. Candidate generation approach (E.g. Apriori algorithm, AprioriTID, Apriori Hybrid)

2. Without candidate generation approach (E.g. FP Growth algorithm)
Understanding the FP Tree Structure:
The frequent-pattern tree (FP-tree) is a compact structure that stores quantitative information about frequent patterns in a database. One root labeled as “null” with a set of item-prefix subtrees as children, and a frequent-item-header table.[2].

2. BASIC APPROACH’S OF FREQUENT PATTERN MINING

2.1 Candidate Generation Approach

Apriori: Apriori proposed by R. Rakesh[1] is the fundamental algorithm. It searches for frequent itemsets browsing the lattice of itemsets in breadth. The database is scanned at each level of lattice. Additionally, Apriori uses a pruning technique based on the properties of the itemsets, which are: If an itemset is frequent, all its sub-sets are frequent and not need to be considered.

AprioriTID: AprioriTID algorithm uses the generation function in order to determine the candidate item sets. The only difference between the two algorithms is that, in AprioriTID algorithm the database is not referred for counting support after the first pass itself.

Apriori Hybrid: Apriori Hybrid uses Apriori in the initial passes and switches to AprioriTid when it expects that the candidate item sets at the end of the pass will be in memory[3].

2.1.1 Implementation of Apriori Algorithm

<table>
<thead>
<tr>
<th>Find all frequent itemsets:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
</tr>
<tr>
<td>Items whose occurrence in database is greater than or equal to the min.support threshold.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
</tr>
<tr>
<td>Generate candidates from frequent items.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
</tr>
<tr>
<td>Rules which satisfy the min.support and min.confidence threshold.</td>
</tr>
</tbody>
</table>

Fig 1: High level design of Apriori

Fig 2: Apriori Algorithm
2.2 Without Candidate Generation Approach

FP-growth: The principle of FP-growth method [5] is to found that few lately frequent pattern mining methods being effectual and scalable for mining long and short frequent patterns. FP-tree is proposed as a compact data structure that represents the data set in tree form.

2.2.1 Implementation of FP Growth Algorithm

![Activity design of FP growth](image)

**Step 1** The dataset is scanned to determine the support of each items. The infrequent items are discarded in FP tree. All frequent items are ordered based on their support.

**Step 2** The algorithm does the second pass over the data and construct the FP tree.

![Algorithm for FP Growth](image)

2.3. Comparative analysis

The two algorithms discussed above are widely studied algorithms for frequent pattern mining. The apriori algorithm works by generating candidate itemsets while the FP-growth algorithm works without generating the candidate sets.

The apriori algorithm has the following bottlenecks:

1. Difficult to handle huge number of candidate itemsets. The candidate generation can be very costly with the increasing size of database.
2. It is tedious to repeatedly scan the huge databases.

The FP-growth algorithm is quite a different algorithm from its predecessors. It works by generating a prefix-tree data structure known as FP-tree from two scans of the database. This algorithm doesn’t need to scan the database multiple times. The main drawbacks of the apriori algorithm are removed with the introduction of the FPgrowth algorithm. The
Table 1: Differentiation between Apriori and FP Growth

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Apriori</th>
<th>FP Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technique</td>
<td>Use Apriori property and join and prune property</td>
<td>Constructs FP-tree and conditional pattern base satisfying minimum support.</td>
</tr>
<tr>
<td>Memory utilization</td>
<td>Large memory space for candidate itemsets</td>
<td>Lesser memory due to compact structure</td>
</tr>
<tr>
<td>No of scans</td>
<td>Multiple scans of database.</td>
<td>Scans the database twice only</td>
</tr>
<tr>
<td>Time</td>
<td>Execution time is large because of candidate itemsets generation</td>
<td>Execution time is smaller.</td>
</tr>
</tbody>
</table>

2.4. Apache Hadoop

The Apache™ Hadoop® develops open-source software for reliable, scalable, distributed computing [6]. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. Hadoop is a Java-based programming framework that supports the processing of large data sets in a distributed computing environment and is part of the Apache project sponsored by the Apache Software Foundation. Hadoop was originally conceived on the basis of Google's Map Reduce, in which an application is broken down into numerous small parts [9]. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures[15]. Hadoop framework is popular for HDFS and Map Reduce. The Hadoop Ecosystem also contains different projects which are discussed below [7] [8]: The Hadoop includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules. It contains libraries and utilities needed by other Hadoop modules.

- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data. HDFS is the Hadoop file system and comprises two major components: namespaces and blocks storage service. The namespace service manages operations on files and directories, such as creating and modifying files and directories. The block storage service implements data node cluster management, block operations and replication.

- **Hadoop YARN**: A framework for job scheduling and cluster resource management. YARN is a resource manager that was created by separating the processing engine and resource management capabilities of Map Reduce as it was implemented in Hadoop 1. YARN is often called the operating system of
Hadoop because it is responsible for managing and monitoring workloads, maintaining a multi-tenant environment, implementing security controls, and managing high availability features of Hadoop.

- **Hadoop Map Reduce**: A YARN-based system for parallel processing of large data sets. The Hadoop Map Reduce framework consists of one Master node termed as Job Tracker and many Worker nodes called as Task Trackers.

### 3. REVIEW ON SEVERAL IMPROVEMENTS OF APRIORI AND FP GROWTH ALGORITHM ON APACHE HADOOP

The table clearly summarizes the essential information of all the algorithms discussed in the paper. The main purpose of the table is to highlight the application of all the above stated algorithms. In this section we are going to discuss content of the paper, proposed system, research gape and observed parameters of different papers.

#### Table 2: Represents a Comparative study of Algorithms

<table>
<thead>
<tr>
<th>Author</th>
<th>Technique</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khurana, K., and Sharma</td>
<td>Apriori Hybrid: Used where Apriori and AprioriTID used.</td>
<td>Better than both Apriori and AprioriTID. [3]</td>
</tr>
<tr>
<td>Borgelt, C.</td>
<td>Apriori: Best for closed item sets.</td>
<td>1. Fast 2. Less candidate sets, 3. Generates</td>
</tr>
<tr>
<td>Hunyadi, D. Borgelt, C.</td>
<td>FP Growth: Used in cases of large problems as it doesn’t require generation of candidate sets.</td>
<td>1. Only 2 passes of dataset., Compresses data set. 2. No candidate set generation required so better than éclat, Apriori. [11][10]</td>
</tr>
</tbody>
</table>

In table 3, Different algorithm’s proposed system and their limitations are discussed which give us a research gape in that papers.

#### Table 3: Comparative Analysis of Apriori and FP Growth Algorithm on Apache Hadoop
Ahilandeeswari.G., Dr. R. Manicka Chezian

<table>
<thead>
<tr>
<th>Author</th>
<th>Content and Proposed System</th>
<th>Research Gape</th>
<th>using hashing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Othman Yahya, Osman Hegazy, Ehab Ezat.</td>
<td>Authors implemented an efficient MapReduce Apriori algorithm (MRApriori) based on Hadoop-MapReduce model which needs only two phases (MapReduce Jobs) to find all frequent itemsets.</td>
<td>In this paper author implement Apriori algorithm on a single machine or can say a stand-alone mode so there are some chance to implement on multiple node.[12]</td>
<td>Sandy Moens, Emin Aksehirli, and Bart Goethals. In this paper, there are two new methods introduce for FIM: Dist-Eclat focuses on speed while BigFIM is optimized to run on really large datasets [14].</td>
</tr>
<tr>
<td>Pallavi Roy.</td>
<td>In this thesis association rule mining was implemented on hadoop. An association rule mining helps in finding relation between the items or item sets in the given data. [13]</td>
<td>Limitation of this algorithm is that it can generated too many association rules and also small dataset may not give good performance. There are some chance to make this algorithm faster by using hashing.</td>
<td></td>
</tr>
</tbody>
</table>

In table 4, different parameters are observed in time of performance evolutions which summarize in this table.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Execution time</th>
<th>Efficiency</th>
<th>Load Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
4. CONCLUSION
In recent years the size of database has increased rapidly. Therefore require a system to handle such huge amount of data. In this paper, algorithmic aspects of association rule mining are dealt with. From a broad variety of efficient algorithms the most important ones are compared. The algorithms are systemized and their performance is analyzed based on runtime and theoretical considerations. Despite the identified fundamental differences concerning employed strategies, runtime shown by algorithms is almost similar. The comparison table shows that the Apriori algorithm outperforms other algorithms in cases of closed item sets whereas FP growth displayed better performance in all the cases. The overall goal of the frequent item set mining process helps to form the association rules for further use. This paper gives a brief survey on frequent pattern mining algorithm Apriori and FP Growth algorithm on Apache Hadoop.

4. REFERENCES


set mining implementations (FIMI 2003, Melbourne, FL, USA).


