ABSTRACT
Association mining aspire to extort frequent patterns, interesting correlations, associations or informal structures between the sets of items in the transaction databases or further data repositories. It plays a essential role in spawning frequent item sets from big transaction databases. The finding of interesting association relationship between business transaction records in various business decision making process such as catalog decision, cross-marketing, and loss-leader analysis. It is also utilized to extort hidden knowledge from big datasets. The Association Rule Mining algorithms such as Apriori, FP-Growth needs repeated scans over the whole database. All the input/output overheads that are being generated through repeated scanning the whole database reduce the performance of CPU, memory and I/O overheads. In this paper we have equaled many classical Association Rule Mining algorithms and topical algorithms.

Keywords
Data Mining, Association Rule Mining (ARM), Association rules, Apriori algorithm, Frequent pattern.

1. INTRODUCTION
The quick development of computer technology, specially enhance capacities and reduce costs of storage media, has led businesses to accumulate large amounts of external and internal information in big databases at low cost. Mining helpful information and useful knowledge from these big databases has thus evolved into a significant research area [3, 2, 1].

Association rule mining (ARM) [18] has become one of the core data mining tasks and has concerned tremendous interest amongst data mining researchers. ARM is an un-directed or un-supervised data mining method which works on variable length data, and generates clear and understandable results. Association Rule Mining (ARM) algorithms [17] are defined into two categories; namely, algorithms respectively with applicant generation and algorithms with no applicant generation. The initial category, those algorithms which are similar to Apriori algorithm for applicant generation are considered. Eclat may also be considered in initial category [8]. In the second category, the FP-Growth algorithm is the best known algorithm. Following table defines the comparison among these three algorithms [9].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scan</th>
<th>Data Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori</td>
<td>M+1</td>
<td>HashTable &amp;Tree</td>
</tr>
<tr>
<td>Eclat</td>
<td>M+1</td>
<td>HashTable &amp;Tree</td>
</tr>
<tr>
<td>FP-Growth</td>
<td>2</td>
<td>PrefixTree</td>
</tr>
</tbody>
</table>

The major disadvantage of above discussed algorithms is the repetitive scans of large database. This may be a cause of decrement in CPU performance, memory and increase in I/O overheads. The performance and efficiency of ARM algorithms mostly depend on three factors; namely applicant sets generated, data structure utilized and details of implementations [8].

The remainder of this paper is organized as follow: Section 2 provides a short review of the related work. In Section 3 we describe Frequent Item set and Association Rule Mining through Apriori Algorithm. In Section 4, we have explained the problem in topical algorithm and how efficiency of similar algorithm can be measured and how speed up is decided. In section 5 we have concluded our study.

2. RELATED WORK
One of the mainly well-known and popular data mining methods is the Association rules or frequent item sets mining algorithm. The algorithm was originally proposed by Agrawal et al. [4] [5] for market basket investigation. Because of its important applicability, various revised algorithms have been introduced since then, and Association rule mining is still a broadly researched area.

Agrawal et al. presented an AIS algorithm in [4] which generates applicant item sets on-the-fly during every pass of the database scan. Large item sets from earlier pass are checked if they are present in the current transaction. Thus new item sets are formed by extending obtainable item sets. This algorithm turns out to be useless because it generates too
many applicant item sets. It needs more space and at the same
time this algorithm needs too many passes over the entire
database and also it generates rules with one consequential
item.

Agrawal [5] developed several versions of Apriori algorithm
such as Apriori, AprioriTid, and AprioriHybrid. Apriori and
AprioriTid produce item sets utilizing the big item sets found
in the previous pass, without considering the transactions.
AprioriTid progress Apriori by utilizing the database at the
initial pass. Counting in consequent passes is done using
encodings created in the initial pass, which is much smaller
than the database. This lead to a dramatic performance
development of three times faster than AIS.

Scalability is another significant area of data mining because
of its large size. Hence, algorithms must be capable to “scale
up” to handle big amount of data. Eui-Hong et al. [16] try
to create data distribution and applicant distribution scalable
by Intelligent Data Distribution (IDD) algorithm and Hybrid
Distribution (HD) algorithm respectively. IDD addresses the
problems of communication overhead and unnecessary
computation by using aggregate memory to partition
applicants and move data efficiently. HD progress over IDD
by dynamically partitioning the applicant set to maintain good
load balance. Different works are reported in the literature to
modify the Apriori logic so as to progress the efficiency of
generating rules. These systems even though focused on
reducing time and space, in real time still requires
improvement.

3. FREQUENT ITEM SET AND
ASSOCIATION RULE

The objective of Association rule mining is exploring
relations and essential rules in large datasets. A dataset is
considered as a series of entries consisting of attribute values
also called as items. A set of such item sets is known an item
set. Frequent item sets are sets of items which are visited
frequently mutually in a single server session.

Let \( I = \{I_1, I_2, \ldots, I_m\} \) be a set of items. Let D, the task-
relevant data, is a set of database transactions where every
transaction T is a set of items such that \( T \subseteq I \). Every
transaction is associated with an identifier, known as TID. Let
A be a set of items. A transaction T is said to contain A if and
only if \( A \subseteq T \). An association rule is an insinuation of the
form \( A \Rightarrow B \), where \( A, B \subseteq I \) and \( A \cap B = \emptyset \). The rule \( A \Rightarrow B \)
hold in the transaction set D with support \( s \), where s is the
percentage of transactions in D that contain A
\( \Rightarrow \) B (i.e., the
union of sets A and B, or say, both A and B). This is taken to
be the possibility, \( P(A \Rightarrow B) \). The rule \( A \Rightarrow B \) has confidence c
in the transaction set D, where c is the percentage of
transactions in D containing A that also contain B. This is
taken to be the conditional possibility, \( P(B|A) \). That is, the
support \( (A \Rightarrow B) = P(A \Rightarrow B) \)……………..\( (2.1) \)

confidence \( (A \Rightarrow B) = P(B|A) \)……………..\( (2.2) \)

A set of items is refers to as an item set. An item set that
contains k items is a k-item set. The set (bread, butter) is a 2-
item set. The occurrence frequency of an item set is the
number of transactions that contains the item set; it is also
called as the frequency, or support count. If the virtual support
of an item set I satisfy a pre specified least support threshold
then I is a frequent item set. The set of frequent k-item sets is
normally denoted by Lk. From Equation (2.2), we have

\[
\text{confidence}(A \Rightarrow B) = \frac{\text{support}(A) \cdot \text{support}(B)}{\text{support}(A)}
\]

Let \( \tau = 11, 12, \ldots \) Im be a set of binary attributes, called items.
Let T be a database of transactions. Each transaction t is
represented as a binary vector, with \( t[k] = 1 \) if t bought
the item Ik, and \( t[k] = 0 \) otherwise. There is one tuple in the
database for each transaction. Let X be a set of some items in
\( \tau \). We say that a transaction t satisfies X if for all items Ik in
X, \( t[k] = 1 \).

By an association rule, we mean an implication of the form
\( X \Rightarrow Ij \), where X is a set of some items in \( \tau \), and Ij is a single
item in \( \tau \) that is not present in X. The rule X \( \Rightarrow Ij \) is satisfied in the
set of transactions T with the confidence factor 0 \( \leq c \leq 1 \)
if at least c% of transactions in T that satisfy X also satisfy Ij.
We will use the notation X \( \Rightarrow Ij \) c to specify that the rule X
\( \Rightarrow Ij \) has a confidence factor of c.[3]

3.1 Apriori Algorithm

The Apriori algorithm is one of the most famous algorithms
for mining frequent patterns and association rules [4]. It
introduces a system to generate applicant item sets Ck in the
pass k of a transaction database utilizing only frequent item
set Lk−1 in the earlier pass. The idea rests on the fact that
any subset of a frequent item set must be frequent as well.
Hence, Ck can be created by joining two item sets in Lk−1
and pruning those that contain any subset that is not frequent
as shown in Figure

### Figure 1. Apriori algorithm

1. **Step 1**: Scan the transaction database to get the support S of each I
   itemset, compares with min sup and get a set of frequent I
   itemset L;

2. **Step 2**: Use Lk+1 to join Lk-1 to generate a set of candidate k-
   itemset and use apriori
   property to prune the unfrequented k-itemset from this set

3. **Step 3**: For every non empty subset s of I
   output the rule "(1-s) if confidence C of rule "(1-s)"
   = support S of I supports s’ min confidence

4. **Step 4**: The candidate set null set

5. **Step 5**: For each frequent itemset generate all nonempty
   subset of I

\[
\text{confidence}(A \Rightarrow B) = \frac{\text{support}(A) \cdot \text{support}(B)}{\text{support}(A)}
\]
In the mainly straight forward version of the algorithm, each item set present in any of the tuples will be measured in one pass, terminating the algorithm in one pass. In the worst case, this approach will need setting up 2^m counters equivalent to all subsets of the set of items D, where m is number of items in D. This is, of course, not only infeasible (m can easily be further than 1000 in a supermarket setting) but also unnecessary. Certainly, most likely there will extremely few large item sets containing more than l items, where l is small. Hence, a lot of those 2^m combinations will turn out to be small in any case.

### 3.2 Bottlenecks of the Apriori Algorithm

In Apriori algorithm there are two bottlenecks.

- One is the complex applicant generation process that utilizes most of the time, space and memory.

- Another bottleneck is the multiple scan of the database. Based on Apriori algorithm.

Above instance shows the working of Apriori algorithm. In every pass of the algorithm item sets of diverse size are generated. To analyze support count for each item set multiple passes to the dataset is necessary so the time taken by process to evaluate support count is more and is keep on increasing as the size of the dataset enhanced.

### 4. TOPICAL ALGORITHM FOR FREQUENT ITEM SET

Topical algorithm [17] are Integrated approach of Parallel Computing and ARM for mining Association Rules in Generalized data set that is basically different from all the earlier algorithms in that it utilizes database in transposed form and database transposition is done utilizing Parallel transposition algorithm (Mesh Transpose) so to generate every important association rules number of passes required is decreased. We will evaluate proposed algorithm with Apriori algorithm for frequent item sets generation. The CPU and I/O overhead can be decreased in our proposed algorithm and it is much quicker than other Association Rule Mining algorithms.

#### Table1: Comparison of Apriori with Topical Algorithm

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data Preprocessing</th>
<th>Scan</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori</td>
<td>No Facility</td>
<td>Repeated</td>
<td>Boolean</td>
</tr>
<tr>
<td>Topical Algorithm</td>
<td>Parallel Preprocessing</td>
<td>One Time</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

#### Transaction database

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>T2</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>T3</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>T4</td>
<td>0</td>
<td>30</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>

#### Parallel Transposition

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1*A2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A1*A3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A1*A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A1*A5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A2*A3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A2*A4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A2*A5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A3*A4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A3*A5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A4*A5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Logical Dot Product

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Rowwise Sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. PROBLEM IDENTIFICATION

We can summarize the working of topical algorithm as follows.

![Diagram of Topical Algorithm Working]

Figure 4. Topical Algorithm Working

Topical algorithms utilize Boolean data set as input but the transaction data set are not in Boolean data type hence there is separate application necessary that will convert Generalized data set into Boolean data set which will reduce the overall efficiency of topical algorithm. As we are seen in Table 1 topical algorithm is efficient than classical Apriori algorithm. The most important benefit of topical proposed algorithm is as follows:

- Applicant generation becomes easy and fast.
- Association rules are produced much quicker, since retrieving a support of an item set is quicker.
- The original document is not influenced by the pruning process where its role ends as soon as data is stored in 2-d array.
- The retrieval of support of an item set is faster.

Topical algorithm utilizes Parallel Mesh Transpose for transposition of 2d array. Since speeding up computations appears to be the main cause behind our interest in building parallel algorithm, the most significant measure in calculating a parallel algorithm is therefore its running time. This is defined as the time taken by the algorithm to resolve a problem on a parallel computer, that is, the time elapsed from the moment the algorithm starts to the moment it terminates.

In calculating a parallel algorithm for a given problem, it is quite natural to do it in terms of the best accessible sequential algorithm for that problem. Thus a superior indication of the quality of a parallel algorithm is the speed up it produces. This is defined as

\[
\text{Speed Up} = \frac{\text{Worst \ running \ time \ of \ fastest \ known \ sequential \ algorithm \ for \ problem}}{\text{Worst \ running \ time \ of \ parallel \ algorithm}}
\]

We that topical algorithm utilizes Mesh Parallel transpose for 2D array transposition. MESH TRANSPOSE calculate the transpose of an n x n matrix in O(n) time. We also noted that this running time is the quickest that can be obtained on a mesh with one data element per processor. However, since the transpose can be calculate sequentially in O(n^2) time, the speed up accomplished by procedure MESH TRANSPOSE is only linear. This speed up may be considered rather small since the process utilizes a quadratic number of processors i.e. if same number of processors arranged in a unlike geometry can transpose a matrix in logarithmic time.

6. CONCLUSION

ARM algorithms are significant to find out frequent item sets and patterns from big databases. In this paper, we have studied classical and topical algorithms for generation of frequent item sets all are similar to Apriori algorithm. Topical algorithm can progress the efficiency of Apriori algorithm and it is observed to be extremely fast. Still there are few problems which we have discussed in Problem identification part i.e. Topical algorithms utilizes Boolean data set as input but transaction data set are not in Generalized data type and hence there is separate application needed that will convert Generalized data set into Boolean data set which will reduce the overall efficiency of topical algorithm and we can also enhance the efficiency of parallel transposition algorithm.

7. REFERENCES


16] Sanjeev Kumar Sharma and Ugrasen Suman “A Performance Based Transposition Algorithm for Frequent Itemsets Generation” International Journal of Data Engineering (IJDE), Volume (2) : Issue (2) : 2011

17] Sujni Paul “An Optimized Distributed Association Rule Mining Algorithm In Parallel and Distributed Data

18] Sujni Paul “An Optimized Distributed Association Rule Mining Algorithm In Parallel and Distributed Data Mining With Xml Data For Improved Response Time”.International Journal Of Computer Science And Information Technology, Volume 2, Number 2, April 2010

19] Manoj Bahel and Chhaya Dule “Analysis of frequent item set generation process in Apriori & RCS (Reduced Candidate Set) Algorithm” National Conference on Information and Communication Technology, Bangalore April 2010