Mining Interesting Association Rules with a Heterogeneous Environment

P.Asha¹ and Dr.T.Jebarajan²

¹Research Scholar,Computer Science and Engineering Department, Sathyabama University, Chennai,Tamilnadu,India.  ashapandian225@gmail.com
²Principal, Kings College of Engineering,Chennai, Tamilnadu,India.  drtjebarajan@gmail.com

Abstract: Association Rule Mining (ARM) algorithms aims at mining rules that is finding the frequent item sets from large databases. This paper discusses various Association Rule Mining algorithms and the Interestingness measures such as Subjective and Objective. Every ARM Algorithm generates large set of rules based on the Support and Confidence, yet sometimes for some specific users the generated rules may not be much interesting. So to filter out only the interested pattern, various Interestingness Parameters have been considered and based on it, the best rules have been filtered.

Keywords: Data Mining, Heterogeneity, Frequent Pattern, Rule Generation, Interestingness Measures.

I. INTRODUCTION

Data mining is a widely used approach to transform data into useful pattern. Handling and mining huge amount of records seems to be difficult with traditional data mining techniques. The process of mining data into useful patterns took longer duration. After finding out the association rules, filtering out the best rules is really challenging. There exists an imbalance in the current data mining techniques from the performance perspective: [2]

* Algorithm imbalance: Many published algorithms vs. several really workable in the business environment,
* Pattern imbalance: Many patterns mined vs. a small portion or none of them satisfying business expectations,
* Decision imbalance: Many patterns identified vs. effectively very few of them can be taken over for business use.

Due to these imbalance, the exact and apt solution lags, that is there exists,

* Deviation between a research issue and its actual business problem,
* Deviation between academic objectives and business goals,
* Deviation between technical significance and business interest,
* Deviation between identified patterns and business expected deliverables.

Therefore, to find the best rules out of the huge rule set, several Rule Interestingness measures were considered. There are two basic Interestingness Measures, Subjective and Objective measures.
**A. Interestingness Measures**

Subjective measures of Interestingness states the belief of the user. It is categorized into two, namely Actionability and Unexpected. Unexpected measures states that the pattern found while discovery may be much astonishing and useful to the user. Actionability measure is that the user act over the pattern to gain advantage of it [4],[6].

Beliefs are classified into two, Hard and Soft Beliefs.

**Hard Beliefs:**
- Belief does not change even after viewing new evidences.
- Varies between users.

**Soft Beliefs:**
- Belief changes with new evidences.
- It is associated with a parameter Degree, which declares how strong the belief is.

Objective measures work on the data and the structure of rule in a discovery procedure. Support and Confidence are objective measures. It generates best rules but the user may not be much interested in these rules. So, to obtain the best and highly interesting rule, it is important to have a combined format, which is the combination of Subjective and Objective measures [4],[6].

If data were present at the same system the mining and filtering the best rules becomes difficult. Accessing data from system with different platform and different database would be extremely hard after which filtering out the best pattern is really challenging. In the proposed system, data mining and extraction of rules as well as filtering the best rules were carried out in a Heterogeneous Environment.

**B. Association Rule Mining Algorithms (ARM)**

Few of the ARM algorithms are:
- Apriori Algorithm
- Dynamic Hashing and Pruning
- Dynamic Item Set Counting
- Common Candidate Partitioned Database
- SEAR and SPEAR, PEAR
- Eclat, MaxEclat, Clique, MaxClique
- Apriori - Count and Data Distribution.
- Frequent Pattern Tree Algorithm.

**✓ Apriori Algorithm**

It finds frequent item sets based on iterative bottom-up approach by generating the candidates. The Apriori Algorithm is an influential algorithm for mining frequent item sets for Boolean association rules [5],[7].

**Key Concepts:**
- Frequent Item sets: The sets of item which has minimum support.
- Apriori Property: Any subset of frequent item set must be frequent.
  1. Join step: Generate all possible candidate item sets Ck of length k
  2. Prune step: Remove those candidates in Ck that cannot be frequent.

Find the frequent item sets: the sets of items that have minimum support
- A subset of a frequent item set must also be a frequent item set i.e., if {AB} is a frequent item set, both {A} and {B} should be a frequent item set
- Iteratively find frequent item sets with cardinality from 1 to k (k-item set)
- Use the frequent item sets to generate association rules.

For each frequent item set X, and for each proper nonempty subset A of X, let B = X – A.

Then, A → B is a association rule if

Confidence (A → B) ≥ minconf,

Support (A → B) = support (A∪B) = support (X)
Confidence (A → B) = support (A∪B) / support (A)

**✓ Dynamic Itemset Counting**

The database is divided into p equal-sized partitions. For every partition, it obtains its local support count with single and 2-item sets. This algorithm counts candidate k-item sets while processing partition k in the initial scan of database. After processing all partitions the processing goes back to initial partition. A
candidate support will be identified when the processing wraps around the database and reaches the partition where it was initially generated [10].

**Common Candidate Partitioned Database**
The Hash Tree stores the logically partitioned database and the candidates which are generated in parallel by the processors. Now every processor scans the database partition and updates the candidate’s count [1], [8].

**Eclat based Algorithm**
It is a vertical mining algorithm. The algorithm provides good scalability. Here the database is divided into horizontal partitions among all processes and each process scans and counts the items in the local database partition and then all processes exchange to get the global counts and finds till frequent 2-itemsets. Then partition them into equivalence classes by prefixes and assigns these classes to processes. Next transform local database partition into vertical tid-lists and then exchange them to get the global tid-lists and now recursively mine all frequent item sets by joining pairs of item sets from the same equivalence class and intersect their corresponding tid-lists [9].

**FP-Growth**
Frequent Pattern Growth mines the frequent item sets without generating the candidates in contrast with Apriori like in which they generate the candidates and finds the frequent item sets. In FP-Growth algorithm, it builds a compact data structure called FP tree from the transactional database and then it extracts the frequent item sets directly from the FP-Tree [5].

II. FILTERING THE INTERESTING RULES

**A. Piatetsky-Shapiro Rule Interestness (RI) measure**
According to Gregory Piatetsky-Shapiro,
RI is defined as, \( RI = N_{\text{both}} - (N_{\text{left}} \cdot N_{\text{right}}) / N_{\text{total}} \), where \( N_{\text{left}} \) be instances matching left, \( N_{\text{right}} \) be instances matching right, \( N_{\text{both}} \) be instances matching both left and right and \( N_{\text{total}} \) be the total number of instances.

RI measures the difference between the actual number of matches and the expected number if the left and right hand sides of the rule were independent [4]. Usually RI value is positive.

If RI = 0, then the rule is no better than chance.

RI = negative, then the rule is less successful than chance.

**B. Finding the Best N Rules**

- The support of the rule is the number of cases that contain both the L (Left hand side rule) and the R (Right hand side rule).
- The coverage of the rule is the number of cases that contain the L.
- The strength is the support divided by the coverage. This represents the proportion of the cases that contain the L that also contain the R.
- Lift of a rule, \( L \rightarrow R \) measures how many more times the items in L and R occur together in transactions than would be expected if the item sets L and R were statistically independent. Lift values greater than 1 implies interesting. They indicate that transactions containing L tend to contain R more often than transactions that do not contain L.
  \[ \text{Lift} (L \rightarrow R) = \frac{\text{count}(L \cup R)}{\text{count}(L) \cdot \text{support}(R)} \]
- Leverage of a rule, \( L \rightarrow R \) measures the difference between the support for LUR (items in L and R occurring together in the database) and the support that would be expected if L and R were independent. The value of leverage of a rule is always less than its support. The number of rules satisfying the support>=minsup and confidence>= minconf constraints can be reduced by setting a leverage>=0.0001 corresponding to an improvement in support of one occurrence per 10000 transactions in a database.
  \[ \text{Leverage} (L \rightarrow R) = \text{support}(L \cup R) - \text{support}(L) \cdot \text{support}(R) \]
- To find the best rule a quality measure named J-measure is used to measure the information content of a rule. Given a rule of the form, if \( Y=y \) then \( X=x \), the information content of the rule measured in bits of information, is denoted by \( J(X;Y=y) \), called the J-measure (or Cross Entropy) for the rule.
  \( J\text{-measure} \) is defined by,
  \[ J(X;Y=y) = p(x|y) \cdot \log_2 \left( \frac{p(x|y)}{p(x)} \right) + (1- p(x|y)) \cdot \log_2 \left( \frac{1- p(x|y)}{1- p(x)} \right) \]
where,

- \( p(x) \) is the probability that RHS of the rule will be satisfied if we have no other information.
- \( p(x|y) \) is the probability that RHS of the rule will be satisfied if we know that the LHS is satisfied.
- \( p(x) = \frac{N_{\text{right}}}{N_{\text{total}}} \)
- \( p(y) = \frac{N_{\text{left}}}{N_{\text{total}}} \)
- \( p(x|y) = \frac{N_{\text{both}}}{N_{\text{left}}} \)

\[ J_{\text{max}} = p(y) \cdot \max \left\{ p(x|y) \cdot \log_2 \left( \frac{1}{p(x)} \right) , (1 - p(x|y)) \cdot \log_2 \left( \frac{1}{1-p(x)} \right) \right\} \]

Therefore if a given value is known to have a \( J \) value of say 0.352 bits and \( J_{\text{max}} \) of say 0.352, there is no benefit to be gained by adding further terms to the LHS as far as information content is concerned[9].

III. IMPLEMENTATION

Apriori is the basic algorithm used for frequent association rule mining. In the proposed system we overcome the difficulties faced in Apriori algorithm by applying Apriori TID and FP-Growth algorithm. In Improved Apriori the database need not be scanned each time when a new candidate is generated. In FP-Growth there is no need for candidates to be generated. The database is read only once so the process of collecting data at each point is reduced. The usage of clustering gives the advantage of grouping items of similar kind into groups under a common factor.

Association rules are then generated which calculates the percentage of each item as a single as well as combination with other items [3]. This tells the probability in which an item is buyable. The proposed system is applied to a Library Management system database. The process of searching book and finding related as well as most frequently accessed books is implemented with the help of FP-Growth algorithm. The FP-Tree algorithm never breaks a long pattern of a transaction; it preserves complete information for frequent pattern mining. It reduces irrelevant information there by infrequent item is removed or they are not taken into consideration, more frequent item are more likely to be shared. If \( D \), a transaction database \( \text{min}_\text{sup} \) the minimum support count threshold are the inputs given to the system then the output will be the complete set of frequent patterns. The functioning happens in a heterogeneous environment where data are stored as well as accessed from different database system located at different nodes.

The work consists of the various phases:

- Finding Frequent Item set and Association Rules, filtering best rules
- Clustering of items of similar kind.
- Performance measurement

The contents of library management system include:

- Registration, Login
- Admin (add & delete books, validation, elapse, profile maintenance, ordinary & advanced search, reservation status)
- User(view profile & account, ordinary & advanced search, reservation)

The first system consists of the data mining algorithm with Microsoft Windows XP in it. The next system has a individual library system in oracle database with Windows OS. Next two systems with individual library system in IBM DB2 database. The next system has another separate library system in MySql with Linux Ubuntu OS. Next two systems with individual library system in MySql database with Linux Fedora OS. Hence a heterogeneous environment was created and tested.

The main idea here is the user accessing a particular library in search for a book can reserve it in the same library if available or he can also access the db of another library and perform all operation on it as local.

In the invoiced system the user uses System-1 and access db in System-2 and if he doesn’t get the book he wanted he accesses System-3(another library) in search of his need.

The concept of data mining is applied in Search engine of the LMS which is described in Fig. 1. On a strike of an alphabet or a word, top ten books that contain the entered input value are retrieved based on their sup_count value and ordered in descending order. This demonstrates the usefulness of data mining in a simpler but effective manner.

In data mining, association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. This concept is applied in the LMS to display the list of books that was taken in combination with the book reserved by the user (in Fig.2). It helps the user by guiding
Fig. 1: Listing Top Ranked Books using Data Mining

him/her with what other book he/she may refer to in addition. To accomplish this, the Apriori algorithm was taken where candidates were generated from the values stored in the db and association rule was applied. The total time taken by this algorithm was 52ms.

In order to decrease this time and improve the efficiency Apriori TID Algorithm was utilized which overcame the difficulties faced with the Apriori Algorithm and yielded a better result of 17ms. Finally the FP-Growth algorithm was analyzed which gave an even better result of 7ms. Hence FP-Growth algorithm is been used for generation of this result.

Fig. 2: Closely associated Book

Clustering in the computer science world is the classification of data or object into different groups. It can also be referred to as partitioning of a data set into different subsets. Each data in the subset ideally shares some common traits.
Every book is given a rank. Clusters are formed based on this rank and on the department to which the books belong to. That is if a department is chosen and clustering process is applied then the result would be three clusters with books belonging to that particular department based on its ranking. We use k-means clustering algorithm for this purpose.

The project is not restricted to be used only via computer. A user visiting the library finds that all systems are occupied he need not wait for the system; he just can use his mobile and perform all actions as he does with the system. This increases user friendliness in the system.

The Performance measurement that is the time taken for the Apriori algorithm is 52ms and Apriori TID is 17ms and FP-Growth is 7ms.

<table>
<thead>
<tr>
<th>Data file: jc.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 cases / 0 holdout cases / 68 items</td>
</tr>
<tr>
<td>Sat Jan 05 00:06:24 2013</td>
</tr>
<tr>
<td>Search for rules</td>
</tr>
<tr>
<td>Search by leverage</td>
</tr>
<tr>
<td>Filter out rules that are unsound.</td>
</tr>
<tr>
<td>Maximum number of attributes on LHS = 4</td>
</tr>
<tr>
<td>Maximum number of rules = 160</td>
</tr>
<tr>
<td>Minimum leverage = 1.9</td>
</tr>
<tr>
<td>Minimum support = 0.2</td>
</tr>
<tr>
<td>Minimum confidence = 0.1</td>
</tr>
<tr>
<td>Minimum strength = 0.1</td>
</tr>
<tr>
<td>All values allowed on LHS</td>
</tr>
<tr>
<td>All values allowed on RHS</td>
</tr>
</tbody>
</table>

Only 8 rules satisfy the specified constraints.

- 'C,GOPLAM' --> 'NUTRITIVE VALUE OF INDIAN FOODS' [Coverage=0.21; Support=0.21; Strength=1.000; Lift=4.14; Leverage=0.1831]
- 'ADVANCES IN ONCOLOGY' --> 'RACETTE' [Coverage=0.101; Support=0.101; Strength=1.000; Lift=1.29; Leverage=0.2112]
- 'ANATOMY' --> 'MEDICAL' [Coverage=0.529; Support=0.529; Strength=0.923; Lift=1.92; Leverage=0.2304]
- 'MEDICAL' --> 'GYNAC' [Coverage=0.224; Support=0.224; Strength=1.000; Lift=4.46; Leverage=0.1739]
- 'C,GOPLAM' --> 'NUTRITION' [Coverage=0.480; Support=0.480; Strength=0.792; Lift=2.08; Leverage=0.1976]
- 'HUMAN PHYSIOLOGY' --> 'KETIE' [Coverage=0.741; Support=0.741; Strength=1.000; Lift=2.07; Leverage=0.1240]
- 'NUTRITION' --> 'JOHN WILEY' [Coverage=0.481; Support=0.481; Strength=0.599; Lift=2.07; Leverage=0.1240]
- 'GEORGE M SCOTT' --> 'NUTRITIVE VALUE OF INDIAN FOODS' [Coverage=0.320; Support=0.320; Strength=0.731; Lift=1.92; Leverage=0.1024]

After finding the frequent item set and generating association rules, the next step is to find the best rules out of all rules generated. From Fig.3. Out of 43 rules, based on the above said Interestingness measures only 8 best and top rules has been generated. The above mentioned computation was done sequentially. Then the same task was tested by computing it parallelly using shared memory parallel programming.

The entire work can be parallelized so as to improve the performance, that is minimizing the execution time. Either the shared memory programming or distributed memory programming can be used. Shared memory parallelization can be done by using openmp programming paradigm. For distributed parallelization, use Message Passing Interface (MPI).

The data mining task has been parallelized using openmp where the task was made to execute using dual core processor in which the total execution time reduced a lot from milliseconds to nanoseconds, than the sequential one.

The computation time before parallelization was 36483516 ns and after parallelization was 23681318 ns. The Run time difference, before and after Parallelization is 12802198 ns. Effective parallelization has been done and hence it shows a better performance.

V. CONCLUSION

A comparative study on various association rule mining algorithms have been done and concluded that the Frequent Pattern Growth behaves effectively in the above discussed Case Study and Environment. Various Interestingness measures were applied to filter out valid and essential patterns and the pattern generated finally sounds good. As a future work, the algorithm may be effectively parallelized and will be tested for its behavior in shared and distributed memory models.
REFERENCES