Dalhousie University

Improvised Apriori algorithm using frequent pattern tree for real time applications in data mining

CS 6405: Data Mining Project

April 20, 2017

Submitted to:

Dr. Qigang Gao

Submitted by:

Tushar Bhalla, B00759300
Arun Athisamy, B00762770

Table of Contents

Abstract 3
  Technology Used: 3
Introduction 4
  Limitations of Apriori Algorithm 4
  Project Goals 5
Related Work 5
Data Preparation 6
  Data Understanding 6
  Data Cleaning 7
System Design 8
  Improved Apriori Algorithm 8
  Pseudo Code for C_k generation 8
  Pseudo Code of the algorithm 9
  Outcome from this algorithm 9
  Example given in the paper 9
  High Level Design 11
    Data Cleaning 11
    Data Mining Engine 11
    Result Comparison 11
    Data Visualization 11
  System Architecture 11
  Code Architecture 13
    I-Apriori 13
    Original Apriori 14
    Running the program 15
    Data Structure 15
    User Interface of I-Apriori 15
    Output File of I-Apriori 16
Experimental Results 17
Conclusion 19
References 19

Abstract

Apriori Algorithm is one of the most important algorithm which is used to extract frequent itemsets from large database and get the association rule for discovering the knowledge. It basically requires two important things: minimum support and minimum confidence. First, we check whether the items are greater than or equal to the minimum support and we find the frequent itemsets respectively. Secondly, the minimum confidence constraint is used to form association rules. Based on this algorithm, the paper which we selected indicates the limitation of the original Apriori algorithm of wasting time and space for scanning the whole database searching on the frequent itemsets, and present an improvement on Apriori. In our project, we implemented the improved algorithm which was proposed by the author and compared the results with the paper's result. We have used Student Alcohol Consumption dataset which is freely available online to test the newly designed algorithm, compare its performance, and visualizing.

Technology Used:

Tools: Visual Studio 2015, SSIS and Weka.
Language: C#.Net from C Family
Application: Console Application

Introduction

Apriori Algorithm is one of the most popular algorithm in data mining for learning the concept of association rules. It is being used by so many people specifically for transaction operations and it can be used in real time applications (for instance, grocery shop, general store, library etc.) by collecting the items bought by customers over the time so that frequent item sets can be generated. Frequent itemsets (itemsets with frequency greater than or equal to a user specified minimum support) can be found very easily because of its combinatorial explosion. Once they are obtained, it is simply easy to generate association rules with confidence greater than or equal to a user specified minimum confidence.

With the invent in technology of information and the need for extracting useful information of business people from dataset, data mining and its techniques is appeared to achieve the above goal. As large amount of data is stored in data warehouses, on line analytical process, databases and other repositories of information. If a person tries to search for the information, it can be done manually which may take Huge (exponential) amount of time. This is not at all optimum and efficient, so data mining approach is the best way by which this problem can be solved very easily. It is the process in which hidden and kind of interesting patterns are generated from huge amount of data which certainly limits the running time. This data may reach to more than terabytes. In some places Data mining can be termed as knowledge discovery in databases as it generates hidden and interesting patterns, and it also comprises of the amalgamation of methodologies from various disciplines such as statistics, neural networks, database technology, machine learning and information retrieval, etc. Interesting patterns are extracted at reasonable time by using the techniques of knowledge discovery in databases(KDD). Frequent itemsets can be found in various ways: using hash-based technique, partitioning, sampling and using vertical data format since it reduces the running time of the algorithm as it finds the itemsets concurrently.

Limitations of Apriori Algorithm

Apriori algorithm suffers from some weaknesses despite being clear and simple. The main limitation is costly wasting of time to hold vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets. For instance, if there are 104 frequent l-itemsets, it may need to generate more than 107 candidates into 2-length which in turn will be tested and accumulated.4 Furthermore, to detect frequent pattern of size 1000 (e. g.) V1,V2...V1000, it will have to generate 21000 candidate itemsets1 that yields costly wasting of time in candidate generation as it checks for many more sets from candidate itemsets, also it will scan database many times repeatedly for finding candidate itemsets. Apriori will be very low and inefficient when memory capacity is limited with large number of transactions. The proposed approach in this paper reduces the time spent for searching in the database and performing transactions for frequent itemsets.

Project Goals

In our project, we have implemented the proposed improvised Apriori algorithm to reduce the time complexity of the original Apriori algorithm by reducing the number of scans of the large number of transaction. The project is predominantly concentrating on reducing the time complexity of original Apriori and compare the results to the results claimed by Author in the paper.

Related Work

Mining of frequent itemsets is an important phase in association mining which discovers frequent itemsets in transactions database. It is essential in many tasks of data mining that try to find interesting patterns from datasets, such as association rules, episodes, classifier, clustering and correlation, etc. Over the time, many algorithms are proposed to find frequent itemsets, but all of them can be catalogued into two classes: candidate generation or pattern growth. Apriori is a representative of the candidate generation approach. It generates the candidate itemsets of length (k+l) based on the frequent itemsets of length (k). The itemset frequency can be defined by counting their occurrences in transactions. Frequent Pattem(FP) - growth, is proposed by Han in 2000, where he stated some useful facts about the FP tree. T.Tassa et al6 proposed secure mining of Association Rules which is based on the Fast Distributed Mining Algorithm and successfully implementd and developed the techniques and methodologies to solve the problem of distributed association rule mining when items are ordered vertically,[6] he also solved the problem of mining generalized association rules, and the problem of discovering subgroups in the horizontal setting. Mahesh Balaji and G Subrahmanya VRK Rao et al[7] in their paper for IEEE proposed Adaptive Implementation Of Apriori Algorithm for Retail Scenario in Cloud Environment which solves the time consuming problem for retail transactional databases. It aims to reduce the response time significantly by using the approach of mining the frequent itemsets. Yanbin Ye, Chia Chu Chiang et al8 in their paper proposed A Parallel Apriori Algorithm for Frequent Itemsets Mining. They modified Bodon’s implementation and converted it into a parallel approach where the input transactions can be read by a parallel computer. The immediate outcome of a parallel computer on this modified implementation is very well presented. Quiang Yang, Yanhong Hu et al in their paper Applications of Improved Apriori Algorithm on Educational Information identified the main problems in the applications and introduced an improved algorithm. Then this algorithm was used for the data education mining. Ketan Shah, Sunita Mahajan et al [10] in their paper Maximizing the efficiency of Parallel Apriori Algorithm suggested how to maximize the efficiency of the parallel Apriori Algorithm. The paper records and observe the performance of the algorithm over different datasets and over 11 processors on a commodity cluster of machines. The experiments conducted showed that the parallel algorithm scaled well to the number of processes and also improved the efficiency by effective load balancing. F eng Wang, Yong-Hua Li et al11 in their paper An Improved Apriori Algorithm Based on the Matrix suggested an improved apriori algorithm based on the matrix. It used the matrix effectively indicating the operations in the database and used the "AND operation" to deal with the matrix to generate the largest frequent itemsets. It doesn’t need to scan the database again and again to perform operations and therefore takes less time, and it also reduced the number of candidates of frequent itemsets greatly.

Data Preparation

This dataset is taken from UCI Machine Learning Repository, the creator of this dataset is Paulo Cortez, and this dataset can be found at:

https://archive.ics.uci.edu/ml/datasets/Student+Performance#

Data Understanding

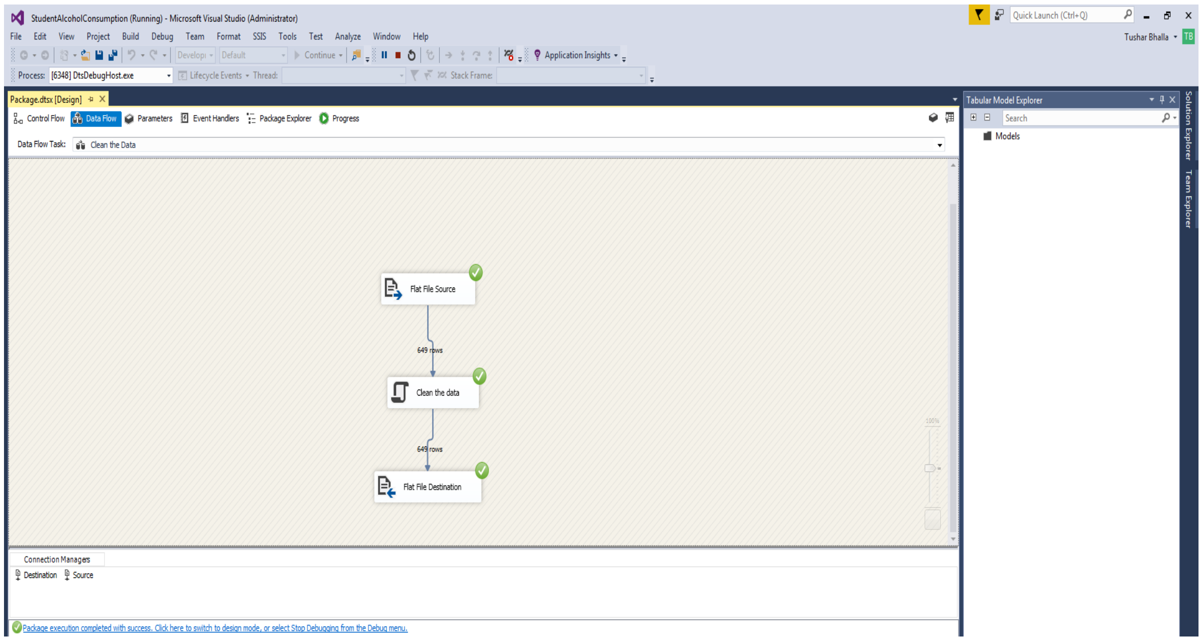
The database was built from two sources: school reports that included few attributes like grades and number of classes missed and questionnaires that included a set of predefined questions that included demographics and social/emotional and school related questions. The data was finally integrated into two datasets related to Mathematics and Portuguese language.
The table below contains attributes for both the datasets and some students may belong to both the classes namely to Mathematics and Portuguese language.

Attribute No	Attribute Name	Attribute Type
1	student’s school	binary
2	student’s sex	binary
3	student's age	numeric
4	student's home address type	binary
5	family size	binary
6	parent's cohabitation status	binary
7	mother's education	numeric
8	father's education	numeric
9	mother's job	nominal
10	father's job	nominal
11	reason to choose this school	nominal
12	student's guardian	nominal
13	home to school travel time	numeric
14	weekly study time	numeric
15	number of past class failures	numeric
16	extra educational support	binary
17	family educational support	binary
18	extra paid classes within the course subject (Math or Portuguese)	binary
19	extra-curricular activities	binary
20	attended nursery school	binary
21	wants to take higher education	binary
22	Internet access at home	binary
23	with a romantic relationship	binary
24	quality of family relationships	numeric
25	free time after school	numeric

26	going out with friends	numeric
27	workday alcohol consumption	numeric
28	weekend alcohol consumption	numeric
29	current health status	numeric
30	number of school absences	numeric
31	G1 - first period grade	numeric
32	G2 - second period grade	numeric
33	G3 - final grade	numeric

Data Cleaning

Special characters are being removed from the dataset and the columns are being converted into tab separated format. For data cleaning, we’ve used SSIS ETL process. We’ve used Script task component here in the data flow to remove special characters with the use of regular expressions.

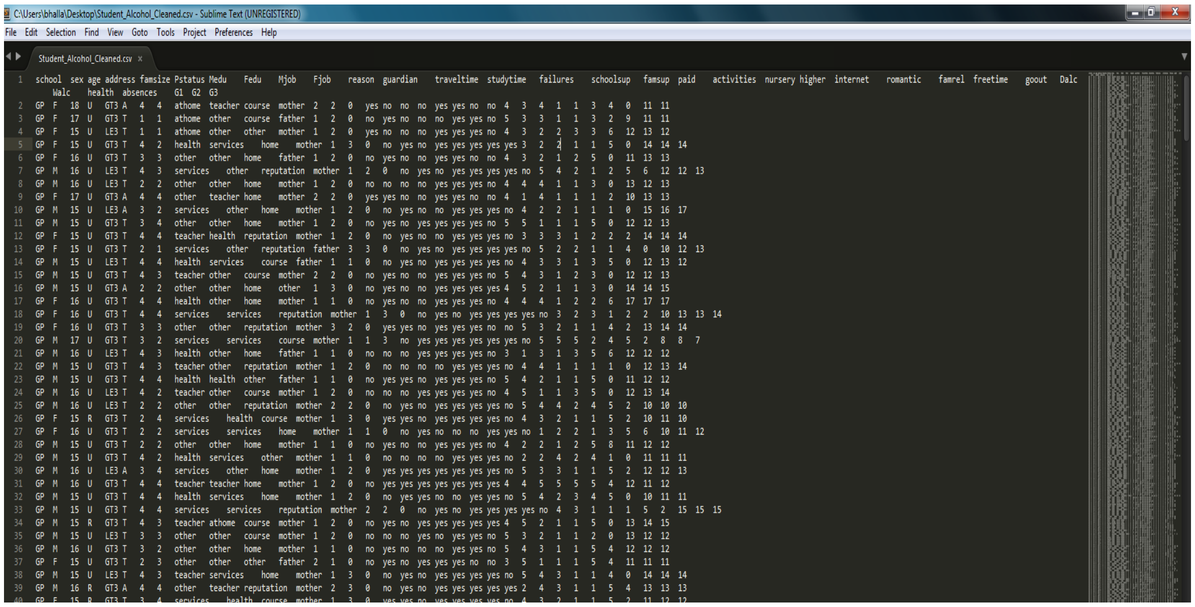


Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

Figure 1. Data preprocessing using SSIS Tool

Using SSIS tool, we have cleaned the raw data by removing double-quotes, flaws, missing or incomplete data. Following are the steps followed for cleaning the data.

- Reading Data: Read the raw dataset from the source.
- Cleaning Data: Used Regex patterns to clean the unwanted characters and double quotes in the dataset
- Writing Data: After cleaning, exported the result to a new CSV file.


C:\Users\bhalla\Desktop\Student_Alcohol_Cleaned.csv - Sublime Text (UNREGISTERED)

```
school  sex age address famsize Pstatus Medu    Fedu    Mjob    Fjob    reason  guardian    traveltime  studytime   failures    schoolsup   famsup  paid    activities  nursery higher  internet    romantic    famrel  freetime    goout   Dalc
    Walc    health  absences    G1  G2  G3
GP  F   18  U   GT3 A   4   4   athome  teacher course  mother  2   2   0   yes no  no  no  yes yes no  no  4   3   4   1   1   3   4   0   11  11
GP  F   17  U   GT3 T   1   1   athome  other   course  father  1   2   0   no  yes no  no  no  yes yes no  5   3   3   1   1   3   2   9   11  11
GP  F   15  U   LE3 T   1   1   athome  other   other   mother  1   2   0   yes no  no  no  yes yes yes no  4   3   2   2   3   3   6   12  13  12
GP  F   15  U   GT3 T   4   2   health  services    home    mother  1   3   0   no  yes no  yes yes yes yes yes 3   2   2   1   1   5   0   14  14  14
GP  F   16  U   GT3 T   3   3   other   other   home    father  1   2   0   no  yes no  no  yes yes no  no  4   3   2   1   2   5   0   11  13  13
GP  M   16  U   LE3 T   4   3   services    other   reputation  mother  1   2   0   no  yes no  yes yes yes yes no  5   4   2   1   2   5   6   12  12  13
GP  M   16  U   LE3 T   2   2   other   other   home    mother  1   2   0   no  no  no  no  yes yes yes no  4   4   4   1   1   3   0   13  12  13
GP  F   17  U   GT3 A   4   4   other   teacher home    mother  2   2   0   yes yes no  no  yes yes no  no  4   1   4   1   1   1   2   10  13  13
GP  M   15  U   LE3 A   3   2   services    other   home    mother  1   2   0   no  yes no  no  yes yes yes no  4   2   2   1   1   1   0   15  16  17
GP  M   15  U   GT3 T   3   4   other   other   home    mother  1   2   0   no  yes no  yes yes yes yes no  5   5   1   1   1   5   0   12  12  13
GP  F   15  U   GT3 T   4   4   teacher health  reputation  mother  1   2   0   no  yes no  no  yes yes yes no  3   3   3   1   2   2   2   14  14  14
GP  F   15  U   GT3 T   2   1   services    other   reputation  father  3   3   0   no  yes no  yes yes yes yes no  5   2   2   1   1   4   0   10  12  13
GP  M   15  U   LE3 T   4   4   health  services    course  father  1   1   0   no  yes no  yes yes yes yes no  4   3   3   1   3   5   0   12  13  12
GP  M   15  U   GT3 T   4   3   teacher other   course  mother  2   2   0   no  yes no  no  yes yes yes no  5   4   3   1   2   3   0   12  12  13
GP  M   15  U   GT3 A   2   2   other   other   home    other   1   3   0   no  yes no  no  yes yes yes yes 4   5   2   1   1   3   0   14  14  15
GP  F   16  U   GT3 T   4   4   health  other   home    mother  1   1   0   no  yes no  no  yes yes yes no  4   4   4   1   2   2   6   17  17  17
GP  F   16  U   GT3 T   4   4   services    services    reputation  mother  1   3   0   no  yes no  yes yes yes yes no  3   2   3   1   2   2   10  13  13  14
GP  F   16  U   GT3 T   3   3   other   other   reputation  mother  3   2   0   yes yes no  yes yes yes no  no  5   3   2   1   1   4   2   13  14  14
GP  M   17  U   GT3 T   3   2   services    services    course  mother  1   1   3   no  yes yes yes yes yes yes no  5   5   5   2   4   5   2   8   8   7
GP  M   16  U   LE3 T   4   3   health  other   home    father  1   1   0   no  no  no  yes yes yes yes no  3   1   3   1   3   5   6   12  12  12
GP  M   15  U   GT3 T   4   3   teacher other   reputation  mother  1   2   0   no  no  no  no  yes yes yes no  4   4   1   1   1   1   0   12  13  14
GP  M   15  U   GT3 T   4   4   health  health  other   father  1   1   0   no  yes yes no  yes yes yes no  5   4   2   1   1   5   0   11  12  12
GP  M   16  U   LE3 T   4   2   teacher other   course  mother  1   2   0   no  no  no  yes yes yes yes no  4   5   1   1   3   5   0   12  13  14
GP  M   16  U   LE3 T   2   2   other   other   reputation  mother  2   2   0   no  yes no  yes yes yes yes no  5   4   4   2   4   5   2   10  10  10
GP  F   15  R   GT3 T   2   4   services    health  course  mother  1   3   0   yes yes no  yes yes yes yes no  4   3   2   1   1   5   2   10  11  10
GP  F   16  U   GT3 T   2   2   services    services    home    mother  1   1   0   no  yes no  no  no  yes yes no  1   2   2   1   3   5   6   10  11  12
GP  M   15  U   GT3 T   2   2   other   other   home    mother  1   1   0   no  yes no  no  yes yes yes no  4   2   2   1   2   5   8   11  12  12
GP  M   15  U   GT3 T   4   2   health  services    other   mother  1   1   0   no  no  no  no  yes yes yes no  2   2   4   2   4   1   0   11  11  11
GP  M   16  U   LE3 A   3   4   services    other   home    mother  1   2   0   yes yes yes yes yes yes yes no  5   3   3   1   1   5   2   12  12  13
GP  M   16  U   GT3 T   4   4   teacher teacher home    mother  1   2   0   no  yes yes yes yes yes yes yes 4   4   5   5   5   5   4   12  11  12
GP  M   15  U   GT3 T   4   4   health  services    home    mother  1   2   0   no  yes yes no  no  yes yes no  5   4   2   3   4   5   0   10  11  11
GP  M   15  U   GT3 T   4   4   services    services    reputation  mother  2   2   0   no  yes no  yes yes yes yes no  4   3   1   1   1   5   2   15  15  15
GP  M   15  R   GT3 T   4   3   teacher athome  course  mother  1   2   0   no  yes no  yes yes yes yes yes 4   5   2   1   1   5   0   13  14  15
GP  M   15  U   LE3 T   3   3   other   other   course  mother  1   2   0   no  no  no  yes no  yes yes no  5   3   2   1   1   2   0   13  12  12
GP  M   16  U   GT3 T   3   2   other   other   home    mother  1   1   0   no  yes no  no  no  yes yes no  5   4   3   1   1   5   4   12  12  12
GP  F   15  U   GT3 T   2   3   other   other   other   father  2   1   0   no  yes no  yes yes yes no  no  3   5   1   1   1   5   4   11  11  11
GP  M   15  U   LE3 T   4   3   teacher services    home    mother  1   3   0   no  yes no  yes yes yes yes no  5   4   3   1   1   4   0   14  14  14
GP  M   16  R   GT3 A   4   4   other   teacher reputation  mother  2   3   0   no  yes no  yes yes yes yes yes 2   4   3   1   1   5   4   13  13  13
GP  F   15  R   GT3 T   3   4   services    health  course  mother  1   3   0   yes yes no  yes yes yes yes no  4   3   2   1   1   5   2   11  12  12
```

Figure 2. Output of Cleaned Dataset

System Design

We have implemented proposed Improved Apriori algorithm to reduce the time complexity of the Original Apriori algorithm.

Improved Apriori Algorithm

Apriori works by taking two inputs:
- Support
- Confidence

Let $T = \{T_1T_2,\ldots,T_m\}$, (m,1) is a set of transactions, $T_i = \{I_1,I_2,\ldots\ldots,I_n\}$,(n,1) is the set of items, and K-itemset = $\{i_1,i_2,\ldots\ldots,i_k\}$, (k,1) is also the set of K items, and k-itemset.

Let (itemset) is the support count of itemset or the frequency of occurrence of an itemset in transactions.

Let C_k is the candidate itemset of size k, and F_k is the frequent itemset of size k.

Pseudo Code for C_k generation

Step1: Scan all transactions to generate F_1 table. F_1 (items, support, transaction ids)
Step2: Construct by self-join.
Step3: Use F_1 to identify the target transactions for.
Step4: Scan the target transactions to generate.

Pseudo Code of the algorithm

Stepl: //Partition the database by n.
Now, select partition one at a time.
Step2: //Generate items, their items' support, transaction ids.
Step3: F] = find_frequent_l_itemsets (T);
Step4: For (k = 2; Fk_1 75 (I); k++)
Step5: //Generate the Ck from the Fk_1
Step6: Ck Z candidates generated from FM;
Step7: // get the item IW with minimum support in Ck using F], (l S w 5 k).
Step8: x = Get _ item_ min_ sup (Ck, F1);
Step9: // get the target transaction IDs that contain item x.
Stepl0: Target = get_ Transaction_ ID(x);
Step11: For each transaction t in Target Do
Step12: Increment the count of all items in Ck that are found in Target;
Step13: Fk = items in Ck min_ support;
End;

Outcome from this algorithm

- Frequent rules generation in less amount of time than the original Apriori algorithm

Example given in the paper

Assume that a large supermarket tracks sales data by stock – keeping unit (SKU) for each item, such as:
- Butter
- Jam
- Coffee
- Cheese
- Milk

The supermarket has a database of transactions where transaction is a set of SKUs that were bought together. Firstly, scan all the transactions to get frequent 1-itemset which contains the items and their support count and the transactions ids that contain these items, and then eliminate the candidates that are infrequent or their support are less than the minimum support. The next step is to generate candidate 2-itemset from L1 split each itemset in 2-itemset into two elements then use l1 table to determine the transactions where you can find the itemset in, rather than searching for them in all transactions. For example, let's take the first item in table.4 (Milk, Cheese), in the original Apriori we scan all 7 transactions to find the item (Milk, Cheese); but in our proposed improved algorithm we will split the item (Milk, Cheese), into Milk and Cheese and get the minimum support between them using L1 has the smallest minimum support. After that we search for itemset (Milk, Cheese) only in the transactions T1 the minimum confidence, and then generate all candidate association rules. In the previous example, if we count the number of scanned transactions to get (1, 2, 3)-itemset using the original Apriori and our improved Apriori, we will observe the obvious difference between number of scanned transactions with our improved Apriori and the original Apriori. From the table 6, number of transactions in1-itemset is the same in both of sides, and whenever the k of k-itemset increase, the gap between our improved Apriori and the original Apriori increase from view of time consumed, and hence this will reduce the time consumed to generate candidate support count.

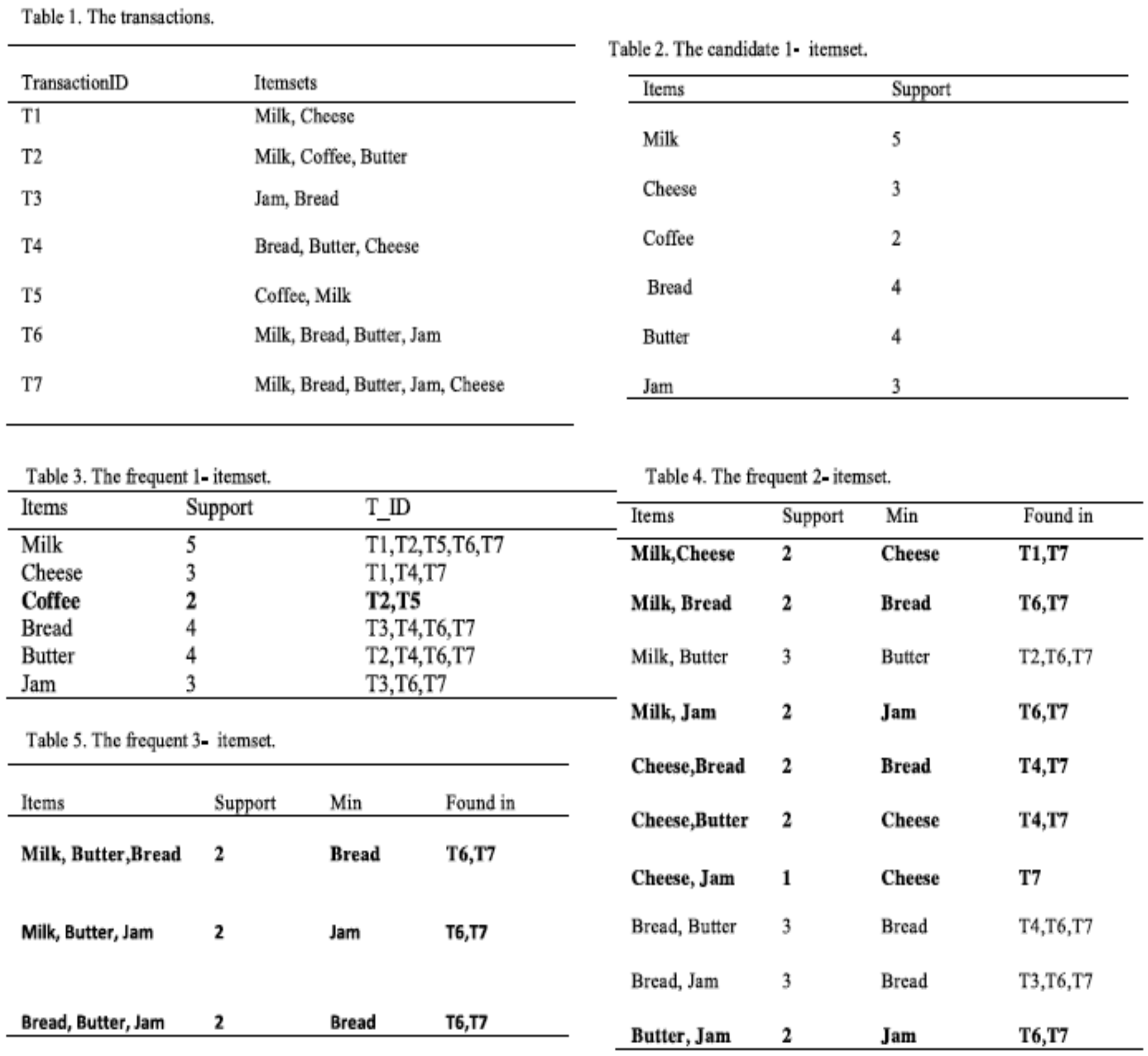


Table 1. The transactions.

TransactionID	Itemsets
T1	Milk, Cheese
T2	Milk, Coffee, Butter
T3	Jam, Bread
T4	Bread, Butter, Cheese
T5	Coffee, Milk
T6	Milk, Bread, Butter, Jam
T7	Milk, Bread, Butter, Jam, Cheese

Table 2. The candidate 1- itemset.

Items	Support
Milk	5
Cheese	3
Coffee	2
Bread	4
Butter	4
Jam	3

Table 3. The frequent 1- itemset.

Items	Support	T_ID
Milk	5	T1,T2,T5,T6,T7
Cheese	3	T1,T4,T7
Coffee	**2**	**T2,T5**
Bread	4	T3,T4,T6,T7
Butter	4	T2,T4,T6,T7
Jam	3	T3,T6,T7

Table 4. The frequent 2- itemset.

Items	Support	Min	Found in
Milk,Cheese	**2**	**Cheese**	**T1,T7**
Milk, Bread	**2**	**Bread**	**T6,T7**
Milk, Butter	3	Butter	T2,T6,T7
Milk, Jam	**2**	**Jam**	**T6,T7**
Cheese,Bread	**2**	**Bread**	**T4,T7**
Cheese,Butter	**2**	**Cheese**	**T4,T7**
Cheese, Jam	**1**	**Cheese**	**T7**
Bread, Butter	3	Bread	T4,T6,T7
Bread, Jam	3	Bread	T3,T6,T7
Butter, Jam	**2**	**Jam**	**T6,T7**

Table 5. The frequent 3- itemset.

Items	Support	Min	Found in
Milk, Butter,Bread	**2**	**Bread**	**T6,T7**
Milk, Butter, Jam	**2**	**Jam**	**T6,T7**
Bread, Butter, Jam	**2**	**Bread**	**T6,T7**

Figure 3. Example for I- Apriori Algorithm [16]

High Level Design

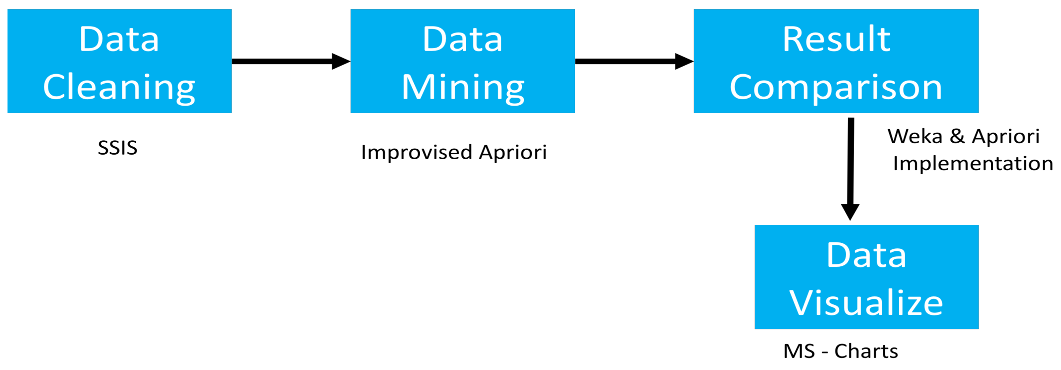


Figure 4. High Level system design

The High-level system design has the following main components: data cleaning, data mining engine, result comparison module and finally visualizing the result in chart.

Data Cleaning
In our project, we have used SSIS tool to clean the raw data and then used in the algorithm to get efficient association rules.

Data Mining Engine
This is most important component of the system. It is a simple console application. The implementation of Improvised Apriori and standard Apriori is done and executed simultaneously to compare the results. The cleaned dataset is taken as input into this engine, processed to generate frequent itemsets. The generated Itemsets are then pruned to generate the association rules. The improvised Apriori algorithm overcomes the limitation of traditional Apriori algorithm by reducing the time complexity. It reduces number of scans while generating candidate k-itemset from frequent 1-itemset.

Result Comparison
This module is used to compare the results from Standard Apriori, Improvised Apriori and Weka Apriori algorithms. The comparison can be found in the upcoming section.

Data Visualization
Results are visualized in the form of chart and evaluated. WE have used MS-chart to visualize the results.

System Architecture

Figure 5 shown below, represents the system architecture of our project. Under data preprocessing module we have used SSIS tool to remove the special characters and double quotes using SSIS scripts. We have used Regex function in SSIS scripts to remove those characters. Also, the program is bound to run only on space/tab separated dataset, hence we have modified the CSV dataset which we took from source [15] in to a space separated value file.

After data preprocessing, the cleaned data set is fetched and given as input to the data mining engine. Data mining engine is used to fetch the dataset and load the data in to data table data structure which we used in this project. Once loading the dataset, we will execute the improved Apriori algorithm and generate the rules. The rules are generated and written in an output text file. The output text file will have the information of number of rules generated, number of rows scanned, minimum support and confidence rate and the time taken to build the rules.

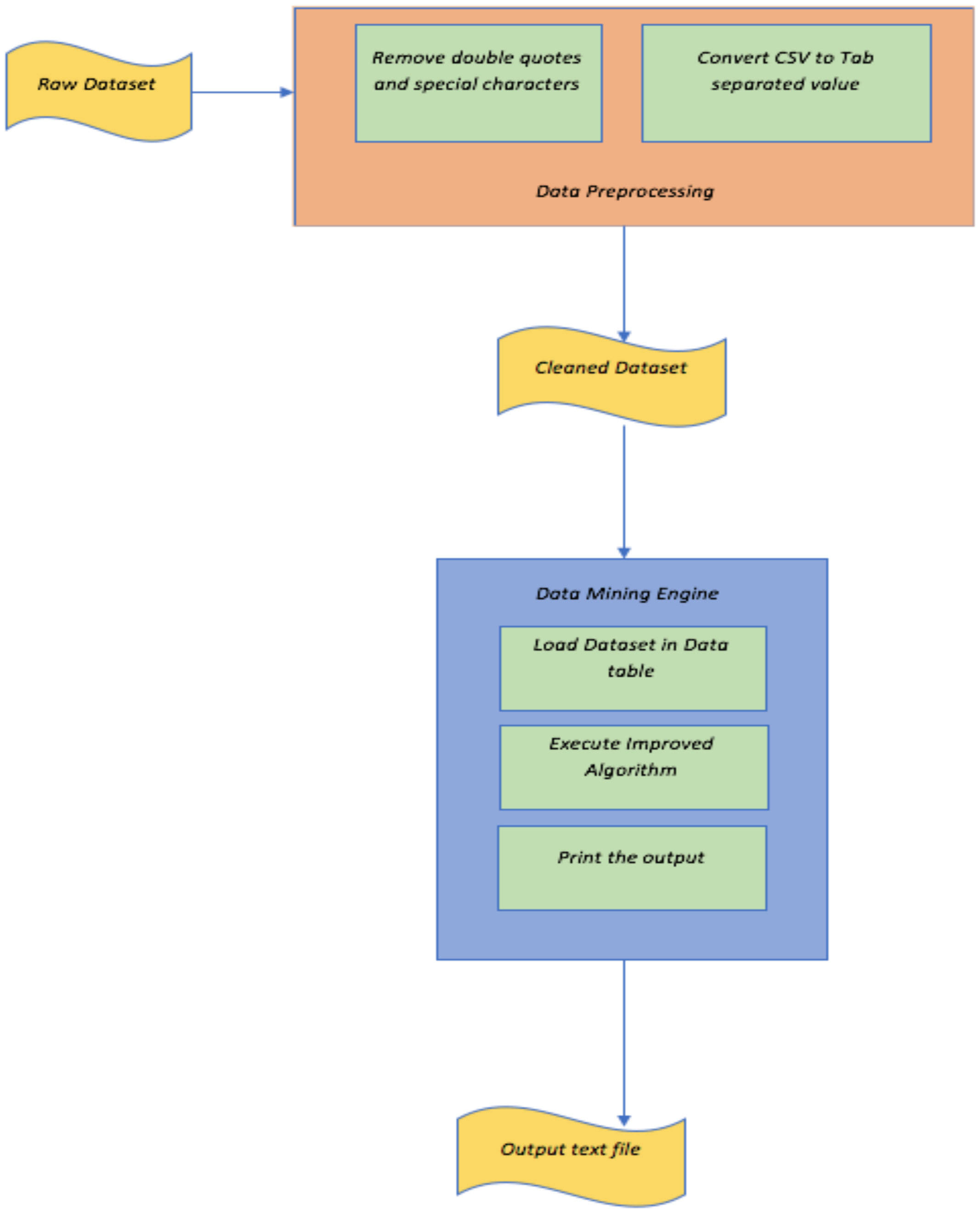

Raw Dataset
Remove double quotes and special characters
Convert CSV to Tab separated value
Data Preprocessing
Cleaned Dataset
Data Mining Engine
Load Dataset in Data table
Execute Improved Algorithm
Print the output
Output text file

Figure 5. System Architecture

Code Architecture

The program is an implementation of the Improvised Apriori algorithm proposed in the paper using C#. We have reduced the number of scans while generating the candidate k-itemset from frequent k-1 itemset in the Improvised algorithm in order to reduce the time complexity. While generating Candidate 1- itemset, we will collect each unique itemset's target transaction IDs. From candidate 1- itemset, we will remove the unique items which has support count less than the minimum support count to generate frequent 1- itemset. For generating candidate K- itemset, we will first self join the frequent k-1 itemset. During self join, we will take item with less minimum support and its target transaction IDs to scan only those target transactional IDs instead of scanning the whole DB again. This will be repeated until the frequent itemset get null and then association rules are generated. On the other hand, Original Apriori scans the entire Dataset every time to generate candidate k-itemset. Below are the code skeleton of both the algorithm.

I-Apriori

It is a simple console application which is separated as Business Logic region and model region

- Business Logic => Program.cs() - Main class for executing Apriori main method and businessLogics
- Models => There are 3 Model classes : Data Rows, Itemset, Result Class

There are 11 functions used in Program.cs :

- Main() - Main method to invoke other methods.
- LoadDB() - Used to read the CSV into List
- generateCandidate1Itemset() - Used to generate unique items in the dataset (Stores unique items, support count and Target Transactions)
- generateFrequent1ItemSet() - Used to generate frequent 1- itemset
- generateFrequentKItemSet() - Used to generate frequent k-itemset
- generateCandidateKItemset() - Used to generate Candidate K-Itemset (Scans only Target Transactions)
- generateCandiateItemSetFromTwoSubsets() - Used to combine two subsets
- GenerateAssociationRule() - Used to generate final association rules
- PrintRules() - Used to write the Rules to the output file
- GetSubsets() - Used to find the subset of an itemset
- Increment() - Used to increment count of the parent element

The following is the program structure:

```
main()
        -->LoadDB()
        -->generateCandidate1Itemset()
        -->generateFrequent1ItemSet()
        -->generateFrequentKItemSet()
                -->  generateCandidateKItemset()
                        -->generateCandiateItemSetFromTwoSubsets()
        -->GenerateAssociationRule()
                -->GetSubsets()
                        -->Increment()
        -->PrintRules()
```

Original Apriori

It is a simple console application which is separated as Business Logic region and model region

- Business Logic => Program.cs() - Main class for executing Apriori main method and businessLogics
- Models => There are 3 Model classes : Data Rows, Itemset, Result Class

There are 11 functions used in Program.cs :

- Main() - Main method to invoke other methods.
- LoadDB() - Used to read the CSV into List
- generateCandidate1Itemset() - Used to generate unique items in the dataset (Stores unique items, support count)
- generateFrequent1ItemSet() - Used to generate frequent 1- itemset
- generateFrequentKItemSet() - Used to generate frequent k-itemset
- generateCandidateKItemset() - Used to generate Candidate K-Itemset (Scans Entire DB)
- generateCandiateItemSetFromTwoSubsets() - Used to combine two subsets
- GenerateAssociationRule() - Used to generate final association rules
- PrintRules() - Used to write the Rules to the output file
- GetSubsets() - Used to find the subset of an itemset
- Increment() - Used to increment count of the parent element

The following is the program structure:

```
main()
        -->LoadDB()
        -->generateCandidate1Itemset()
        -->generateFrequent1ItemSet()
        -->generateFrequentKItemSet()
                -->  generateCandidateKItemset()
                        -->generateCandiateItemSetFromTwoSubsets()
        -->GenerateAssociationRule()
                -->GetSubsets()
                        -->Increment()
        -->PrintRules()
```

Running the program

I have tried executing my .exe file in bluenose. But due to security reasons, it looks like they have restricted .exe files to be executed in bluenose. I have Uploaded the entire Project in the bluenose.
To run the Improved Apriori algorithm, Download the .exe file present the in the path:

1)/users/grad/athisamy/Proj/ImprovedAprioriAlgorithm.exe

(or)

2)/users/grad/athisamy/Proj/SourceCode/ImprovedApriori/bin/Debug/ImprovedApriori.exe

As we used standard Apriori algorithm to compare the performance and results, I have uploaded the source code for the original Apriori in bluenose as well.
To run the Standard Apriori algorithm, Download the .exe file present the in the path:

1)/users/grad/athisamy/ Proj /AprioriAlgorithm.exe

(or)

2)/users/grad/athisamy/Proj/SourceCode/AprioriAlgorithm/bin/Debug/AprioriAlgorithm.exe

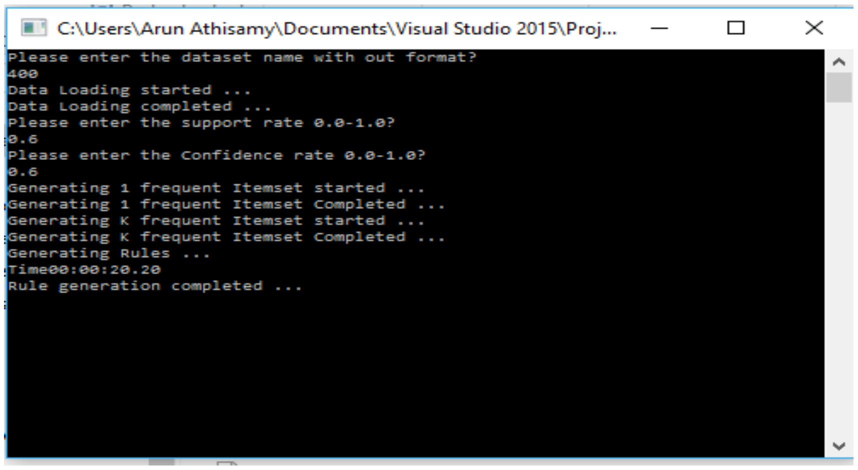
The result is stored in the file "output.txt". The result file is generated at the same directory as that of your execution for both the algorithms.

Data Structure

Author in his implementation used FP tree as data structure to have compactness of the dataset. In our project implementation, we have used Data Table as our data structure to store dataset and still found results matching close to that of Author's claim which is discussed in the below sections.

User Interface of I-Apriori

As mentioned above it is a simple console application, where user can input the dataset along with their support and confidence rate. In below execution, 400 is the dataset name. User is given the notification after completion of each step. The UI also specifies the time taken by the algorithm as well as the completion of rule generation at the end.

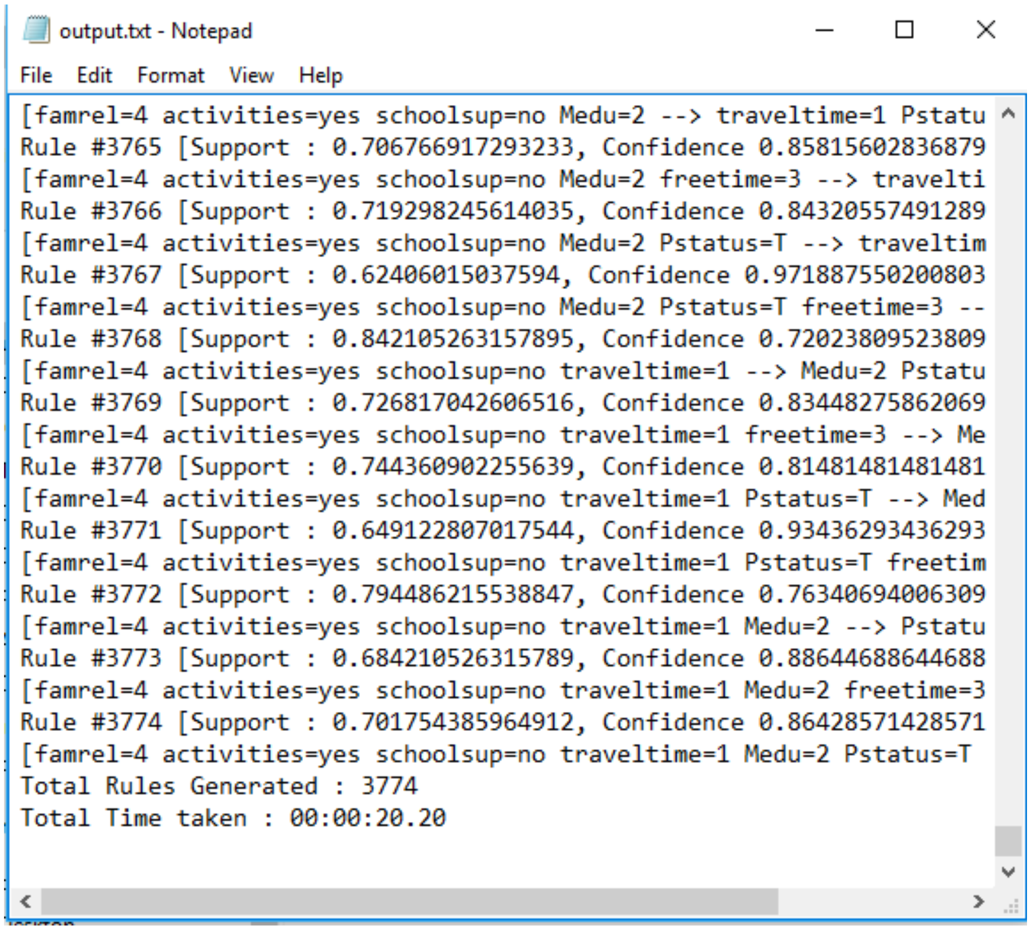


```
Please enter the dataset name with out format?
400
Data Loading started ...
Data Loading completed ...
Please enter the support rate 0.0-1.0?
0.6
Please enter the Confidence rate 0.0-1.0?
0.6
Generating 1 frequent Itemset started ...
Generating 1 frequent Itemset Completed ...
Generating K frequent Itemset started ...
Generating K frequent Itemset Completed ...
Generating Rules ...
Time00:00:20.20
Rule generation completed ...
```

Figure 6. UI of Improvised Apriori Algorithm

Output File of I-Apriori

Output file contains the information of how many instances parsed, time taken to build the rules and number of rules generated.



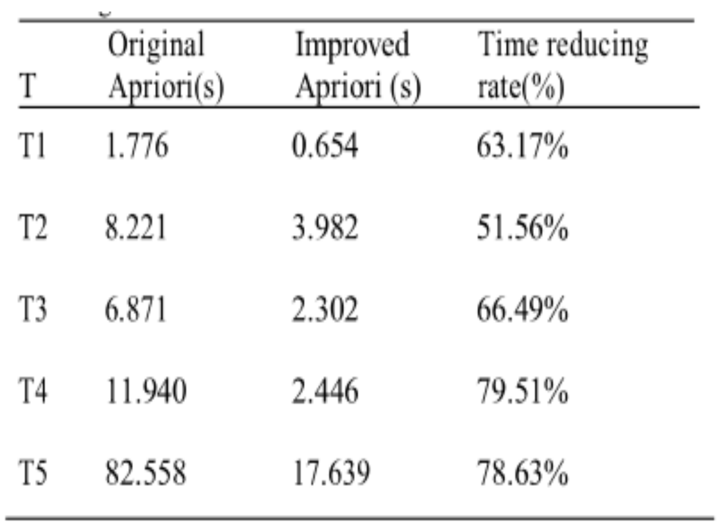
```
[famrel=4 activities=yes schoolsup=no Medu=2 --> traveltime=1 Pstatu
Rule #3765 [Support : 0.706766917293233, Confidence 0.85815602836879
[famrel=4 activities=yes schoolsup=no Medu=2 freetime=3 --> travelti
Rule #3766 [Support : 0.719298245614035, Confidence 0.84320557491289
[famrel=4 activities=yes schoolsup=no Medu=2 Pstatus=T --> traveltim
Rule #3767 [Support : 0.62406015037594, Confidence 0.971887550200803
[famrel=4 activities=yes schoolsup=no Medu=2 Pstatus=T freetime=3 --
Rule #3768 [Support : 0.842105263157895, Confidence 0.72023809523809
[famrel=4 activities=yes schoolsup=no traveltime=1 --> Medu=2 Pstatu
Rule #3769 [Support : 0.726817042606516, Confidence 0.83448275862069
[famrel=4 activities=yes schoolsup=no traveltime=1 freetime=3 --> Me
Rule #3770 [Support : 0.744360902255639, Confidence 0.81481481481481
[famrel=4 activities=yes schoolsup=no traveltime=1 Pstatus=T --> Med
Rule #3771 [Support : 0.649122807017544, Confidence 0.93436293436293
[famrel=4 activities=yes schoolsup=no traveltime=1 Pstatus=T freetim
Rule #3772 [Support : 0.794486215538847, Confidence 0.76340694006309
[famrel=4 activities=yes schoolsup=no traveltime=1 Medu=2 --> Pstatu
Rule #3773 [Support : 0.684210526315789, Confidence 0.88644688644688
[famrel=4 activities=yes schoolsup=no traveltime=1 Medu=2 freetime=3
Rule #3774 [Support : 0.701754385964912, Confidence 0.86428571428571
[famrel=4 activities=yes schoolsup=no traveltime=1 Medu=2 Pstatus=T
Total Rules Generated : 3774
Total Time taken : 00:00:20.20
```

Figure 7. Sample Output File

Experimental Results

Experiments were performed between standard Apriori algorithm and improvised Apriori algorithm implemented in C# by splitting up the dataset in to different amount transactions like T250, T400 and T650 with support and confidence rate as 60% respectively. We have also used Weka tool to run all the three Transactions in order to find the response time.

Author Claims that the average time reducing rate of Improved Apriori algorithm as 67% with 5 set of Transactions. We are splitting the dataset into just 3 parts, hence we are taking the average time reducing rate of first 3 transactions used by Author. We find that the average time reducing rate of first 3 transaction is 60%.



T	Original Apriori(s)	Improved Apriori (s)	Time reducing rate(%)
T1	1.776	0.654	63.17%
T2	8.221	3.982	51.56%
T3	6.871	2.302	66.49%
T4	11.940	2.446	79.51%
T5	82.558	17.639	78.63%

Figure 8. Results claimed by Author [16]

Figure 7 shows the time reducing rate of Improved Apriori algorithm over original Apriori algorithm claimed by author. Table 1 represents the Time reducing rate for each set of transactions executed in Original and Improved Apriori implemented in C#.

T	Original Apriori (s)	Improved Apriori (s)	Time reducing rate (%)
T1	22.80	16.35	40
T2	08.64	05.18	66
T3	12.07	07.64	59

Table 1. Results from Original and Improved Apriori algorithm implemented in C#

The average reducing time rate from our implementation is 55% which is almost as close that of Author's claim. Below is the chart, which depicts the time taken by both Original and Improved Apriori algorithm.

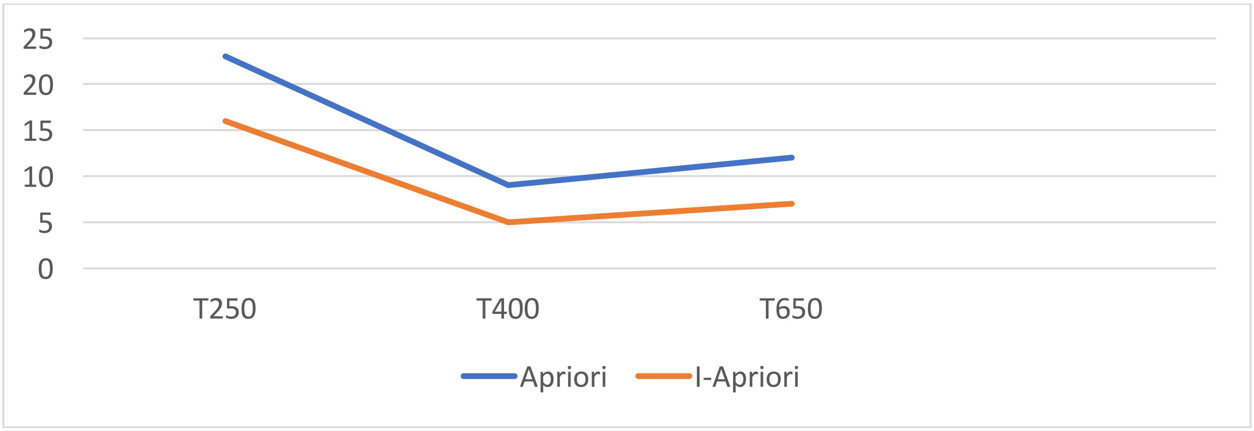
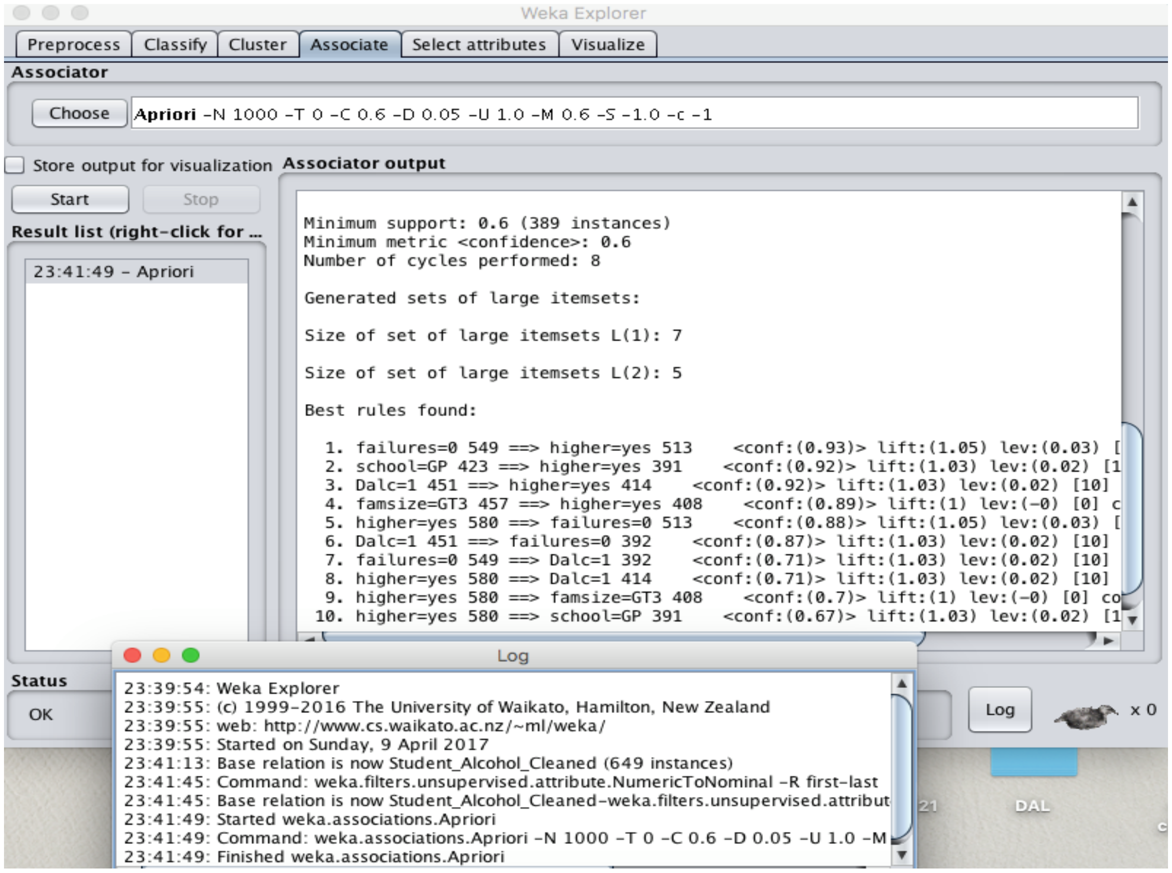


Figure 9. Graph between Original and Improved Apriori – Time comparison for different groups of transactions

From Figure 8, it is evident that the improvised Apriori algorithm takes less time than that of Original Apriori. We have also run our dataset in Weka and found that Weka works with a concept of cycles for generating rules in Apriori algorithm which is not used in our implementation.



```
Apriori -N 1000 -T 0 -C 0.6 -D 0.05 -U 1.0 -M 0.6 -S -1.0 -c -1
```

```
Minimum support: 0.6 (389 instances)
Minimum metric <confidence>: 0.6
Number of cycles performed: 8

Generated sets of large itemsets:

Size of set of large itemsets L(1): 7

Size of set of large itemsets L(2): 5

Best rules found:

  1. failures=0 549 ==> higher=yes 513    <conf:(0.93)> lift:(1.05) lev:(0.03) [
  2. school=GP 423 ==> higher=yes 391    <conf:(0.92)> lift:(1.03) lev:(0.02) [1
  3. Dalc=1 451 ==> higher=yes 414    <conf:(0.92)> lift:(1.03) lev:(0.02) [10]
  4. famsize=GT3 457 ==> higher=yes 408    <conf:(0.89)> lift:(1) lev:(-0) [0] c
  5. higher=yes 580 ==> failures=0 513    <conf:(0.88)> lift:(1.05) lev:(0.03) [
  6. Dalc=1 451 ==> failures=0 392    <conf:(0.87)> lift:(1.03) lev:(0.02) [10]
  7. failures=0 549 ==> Dalc=1 392    <conf:(0.71)> lift:(1.03) lev:(0.02) [10]
  8. higher=yes 580 ==> Dalc=1 414    <conf:(0.71)> lift:(1.03) lev:(0.02) [10]
  9. higher=yes 580 ==> famsize=GT3 408    <conf:(0.7)> lift:(1) lev:(-0) [0] co
 10. higher=yes 580 ==> school=GP 391    <conf:(0.67)> lift:(1.03) lev:(0.02) [1
```

```
23:39:54: Weka Explorer
23:39:55: (c) 1999-2016 The University of Waikato, Hamilton, New Zealand
23:39:55: web: http://www.cs.waikato.ac.nz/~ml/weka/
23:39:55: Started on Sunday, 9 April 2017
23:41:13: Base relation is now Student_Alcohol_Cleaned (649 instances)
23:41:45: Command: weka.filters.unsupervised.attribute.NumericToNominal -R first-last
23:41:45: Base relation is now Student_Alcohol_Cleaned-weka.filters.unsupervised.attribut
23:41:49: Started weka.associations.Apriori
23:41:49: Command: weka.associations.Apriori -N 1000 -T 0 -C 0.6 -D 0.05 -U 1.0 -M
23:41:49: Finished weka.associations.Apriori
```

Figure 10. Results from Weka with time stamp

On our observation, we found that the rules generated by Weka are present in the output file of improvised Apriori algorithm. Hence we verified that our rules are correct by comparing it with

Weka. Since Weka runs in different logic, time comparison is done with the Original Apriori algorithm which is also implemented on the same platform using same data structure.

Conclusion

On our observation, the time consumed to generate candidate support count in our improved Apriori is less than the time consumed in the original Apriori; our improved Apriori reduces the time consuming by 55%. Hence, this approach is far more efficient than the original Apriori algorithm. It can be successfully used in the real-time applications especially in the library as it can save a lot of time by giving all the information about those books which are frequently read. Having used our own data structure instead of the data structure which was proposed by author, we achieved the claim of the Author.

In future, the time and memory complexity can be reduced by using parallel algorithm with FP tree data structure. As we are talking about the distributed processing concept, cloud computing can be used to improve the performance for real time applications.

References

[1] Wu X, Kumar V, Ross Quinlan J, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou Z-H, Steinbach M, Hand DJ and Steinberg D, Top 10 algorithms in data mining, Knowledge and Information Systems, vol. 14, no. 1, pp. 1-37, Dec. 2007.

[2] Mohammed Al-Maolegil, Bassam Arkok, Computer Science, Jordan University of Science and Technology, Irbid, Jordan.

[3] International Journal on Natural Language Computing (IJNLC) Vol. 3, No.1, February 2014. Han J, Kamber M, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, Book, 2000.

[4] Rao S, Gupta R, Implementing Improved Algorithm Over APRIORI Data Mining Association Rule Algorithm, International Journal of Computer Science And Technology, pp. 489-493, Mar. 2012.

[5] Srikant R, Fast algorithms for mining association rules and sequential patterns, University of Wisconsin, 1996.

[6] Tassa T, Secure Mining of Association Rules in Horizontally Distributed Database, in IEEE Transactions on knowledge and data Engineering, Vol. 26, No. 4, April 2014.

[7] Balaji Mahesh, VRK Rao G Subrahmanya, An Adaptive Implementation Case Study of Apriori Algorithm for a Retail Scenario in a Cloud Environment, ccgrid, pp.625-629, 2013 13th IEEE/A CM International Symposium on Cluster, Cloud, and Grid Computing, 2013.

[8] Ye Yanbin, Chiang Chia-Chu, A Parallel Apriori Algorithm for Frequent Itemsets Mining, sera, pp.87-94, Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), 2006.

[9] Yang Qiang, Hu Yanhong, Application ofImproved Apriori Algorithm on Educational Information, icgec, pp.330-332, 2011 Fifth International Conference on Genetic and Evolutionary Computing, 2011.

[10] Shah K, Mahajan , Maximizing the Efficiency of Parallel Apriori Algorithm, artcom, pp.107-109, 2009 International Conference on Advances in Recent Technologies in Communication and Computing, 2009.

[11] Wang Feng, Li Yong-hua, An Improved Apriori Algorithm Based on the Matrix, fbie, pp.152- 155, 2008 International Seminar on Future BioMedical Information Engineering, 2008.

[12] Das D, Misra R, Raj A, Approximating geographic routing using coverage tree heuristics for wireless network, Springer Wireless Networks ,DOI: 10.1007/s11276-014-0837-4.

[13] Das D and Misra R, Improvised k-hop Neighbourhood Knowledge Based Routing in Wireless Sensor Networks, 2nd International Conference on Advanced Computing, Networking and Security (ADCONS), 2013, p. 136-141.

[14] Agrawal R, Imieli ski T, and Swami A, “Mining association rules between sets of items in large databases,” in Acm Sig Mod Record, vol. 22, pp. 207-216, 1993.

[15] "UCI Machine Learning Repository: STUDENT ALCOHOL CONSUMPTION Data Set", *Archive.ics.uci.edu*, 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/STUDENT+ALCOHOL+CONSUMPTION. [Accessed: 10- Apr- 2017].

[16] Akshita Bhandaria, Ashutosh Guptaa and Debasis Dasa. Improvised apriori algorithm using frequent pattern tree for real time applications in data mining. International Conference on Information and Communication Technologies (ICICT 2014) Procedia Computer Science 46 (2015) 644 – 651, 2014.