

Certificate

This is to certify that Mr. Sanjit Sadhukhan of Guru Nanak Institute of Technology, registration number: 151430110084 of 2015-2016, has successfully completed a project on Market Basket Analysis using Bigdata (Hadoop) under the guidance of Mr. Titas RoyChowdhury.

Titas RoyChowdhury
Globsyn Finishing School

Titas RoyChowdhury



Certificate

This is to certify that Mr. Subham Banerjee of Guru Nanak Institute of Technology, registration number: 151430110106 of 2015-2016, has successfully completed a project on Market Basket Analysis using Bigdata (Hadoop) under the guidance of Mr. Titas RoyChowdhury.

Titas RoyChowdhury
Globsyn Finishing School



Certificate

This is to certify that Ms. Sayantony Dey of Guru Nanak Institute of Technology, registration number: 151430110089 of 2015-2016, has successfully completed a project on Market Basket Analysis using Bigdata (Hadoop) under the guidance of Mr. Titas RoyChowdhury.

Titas RoyChowdhury
Globsyn Finishing School

Titas Roy Chowdhury



Certificate

This is to certify that Mr. Aveepsit Chowdhury of Dr. B. C. Roy Engineering College, registration number: 161200110026 of 2016-2017, has successfully completed a project on Market Basket Analysis using Bigdata (Hadoop) under the guidance of Mr. Titas RoyChowdhury.

Titas RoyChowdhury
Globsyn Finishing School

Aveepsit Chowdhury



Certificate

This is to certify that Ms. Kheya Mondal of Dr. B. C. Roy Engineering College, registration number: 171200120003 of 2017-2018, has successfully completed a project on Market Basket Analysis using Bigdata (Hadoop) under the guidance of Mr. Titas RoyChowdhury.

Titas RoyChowdhury
Globsyn Finishing School

Titas RoyChowdhury



ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas RoyChowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time carried me a long way in the journey of this project.

Aveepsit Chowdhury, Dr. B. C. Roy Engineering
College, 161200110026 of 2016 - 2017

Kheya Mondal, Dr. B. C. Roy Engineering College,
171200120003 of 2017 - 2018

Sanjit Sadhukhan, Guru Nanak Institute of
Technology, 151430110084 of 2015 - 16

Subham Banerjee, Guru Nanak Institute of
Technology, 151430110106 of 2015-16

Sayantony Dey, Guru Nanak Institute of Technology,
151430110089 of 2015-16



CONTENTS

Serial No.	Topic	Page No.
1.	Introduction	1
2.	Objectives	2
3.	Data Source	3
4.	Tools/Technologies Used	4 - 11
4.1.	Association Rule Mining	
4.2.	Hadoop Cluster	
4.3.	Map/Reduce in Hadoop	
5.	Data Processing	12 - 13
5.1.	Flow-Chart of Association Rule Mining Algorithm	
5.2.	Association Rule Mining Algorithm	
5.3.	Mapper-Reducer Decomposition Napster	
6.	Code	14 - 29
6.1.	Mapper 1 - MBAMapper.java	
6.2.	Reducer 1 - MBAReducer.java	
6.3.	Mapper 2 - MBAMapper2.java	
6.4.	Mapper 3 – MBAMapper3.java	
6.5.	Driver – MBADriver.java	
7.	Implementations of market basket analysis	30
8.	Future implementation	32
9.	Conclusion	33

y. Prashant Choudhary



1. INTRODUCTION

Market Basket Analysis is one of the most common and useful types of data analysis for marketing and retailing. The purpose of market basket analysis is to determine what products customers purchase together. It takes its name from the idea of customers throwing all their purchases into a shopping cart (a "market basket") during grocery shopping. Knowing what products people purchase as a group can be very helpful to a retailer or to any other company. A store could use this information to place products frequently sold together into the same area, while a catalogue or World Wide Web merchant could use it to determine the layout of their catalogue and order form. Direct marketers could use the basket analysis results to determine what new products to offer their prior customers.

In some cases, the fact that items sell together is obvious – every fast-food restaurant asks their customers "Would you like fries with that?" whenever they go through a drive-through window. However, sometimes the fact that certain items would sell well together is far from obvious. A well-known example is that a supermarket performing a basket analysis discovered that diapers and beer sell well together on Thursdays. Though the result does make sense – young couples stocking up on supplies for themselves and for their children before the weekend starts – it's not the sort of thing that someone would normally think of right away. The strength of market basket analysis is that by using computer data mining tools, it's not necessary for a person to think of what products consumers would logically buy together – instead, the customers' sales data is allowed to speak for itself. This is a good example of data-driven marketing.

Y. P. Chowdhury



2. OBJECTIVES

The primary objective of Market Basket Analysis is to improve the effectiveness of marketing and sales tactics using customer data collected during the sales transaction. It can also be used to optimize and facilitate business operations particularly with regards to inventory control and channel optimization.

The aim of Market Basket Analysis is to discover patterns in apparently random data, and use all this information to better understand trends, patterns, correlations, and ultimately predict customer behaviour, market and competition trends, so that the company uses its own data more meaningfully to better position itself on the new waves.

The primary project goals consist of:

1. Generate the association rule from the item sets.
2. Calculate the various metrics of Association Rule - Support, Confidence and Lift for the transactions performed.

y. Prashant Chowdhury



3. DATA SOURCE

Input to the program will be a file which contains the list of transactions with each transaction being separated by a new line character. Each transaction contains a list of items bought by a customer in one go. The items inside a transaction being space separated. Each item is a distinct entity referring to one single distinct product inside the shop.

```
hering corned_b olives ham turkey bourbon ice_crea
baguette soda hering cracker heineken olives corned_b
avocado cracker artichok heineken ham turkey sardines
olives bourbon coke turkey ice_crea ham peppers
hering corned_b apples olives steak avocado turkey
sardines heineken chicken coke ice_crea peppers ham
olives bourbon coke turkey ice_crea heineken apples
corned_b peppers bourbon cracker chicken ice_crea baguette
soda olives bourbon cracker heineken peppers baguette
corned_b peppers bourbon cracker chicken bordeaux hering
baguette sardines apples peppers avocado steak turkey
baguette hering avocado artichok heineken apples corned_b
hering corned_b apples olives steak sardines heineken
baguette sardines apples peppers avocado steak ice_crea
hering corned_b olives ham turkey coke apples
olives bourbon coke turkey ice_crea artichok ham
baguette hering avocado artichok heineken coke turkey
sardines heineken chicken coke ice_crea corned_b apples
...
```

The sample file of transactions we have used for this project has been provided to us by Globsyn Finishing School.



Document sign date : Nov 17, 2018

4. TOOLS/TECHNOLOGIES USED

For the purpose of processing the transaction data we have implemented Association Rule Mining algorithm using Hadoop map-reduce code.

4.1. Association Rule Mining

Association rule mining is a procedure which is meant to find frequent patterns, correlations, associations, or causal structures from data sets. It is the data mining process of finding the rules that may govern associations and causal objects between sets of items.

So in a given transaction with multiple items, it tries to find the rules that govern how or why such items are often bought together. For example, peanut butter and jelly are often bought together because a lot of people like to make PB&J sandwiches. Also, surprisingly, diapers and beer are bought together because, as it turns out, that dads are often tasked to do the shopping while the moms are left with the baby.

Mathematically it can be interpreted as:

Let $I = \{i_1, i_2, \dots\}$ be a set of binary attributes called items.

Let $D = \{t_1, t_2, \dots\}$ be a set of transactions called the database.

Each transaction in D contains a subset of items in I .

Simple rule looks like -























$t_1 \Rightarrow t_2$ (Here, t_i is generally a single item or a set of items)

* t_1 : Antecedent, t_2 : Consequent

Given a set of transactions, association rule mining aims to find the rules



which enable us to predict the occurrence of a specific item based on the occurrences of the other items in the transaction.

Transaction 1	   
Transaction 2	  
Transaction 3	 
Transaction 4	 
Transaction 5	   
Transaction 6	  
Transaction 7	 
Transaction 8	 

Measure 1: Support. This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. In Table 1 below, the support of {apple} is 4 out of 8, or 50%. Itemset can also contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%.

$$\text{Support} \{\text{apple}\} = \frac{4}{8}$$

Table 1. Example Transactions

If you discover that sales of items beyond a certain proportion tend to have a significant impact on your profits, you might consider using that proportion as your *support threshold*. You may then identify itemset with support values above this threshold as significant itemset.

yProz Chowdhury



Measure 2: Confidence. This says how likely item Y is purchased when item X is purchased, expressed as {X → Y}. This is measured by the proportion of transactions with item X, in which item Y also appears. In Table 1, the confidence of {apple → beer} is 3 out of 4, or 75%.

$$\text{Confidence} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎}, \text{🍺}\}}{\text{Support} \{\text{🍎}\}}$$

One drawback of the confidence measure is that it might misrepresent the importance of an association. This is because it only accounts for how popular apples are, but not beers. If beers are also very popular in general, there will be a higher chance that a transaction containing apples will also contain beers, thus inflating the confidence measure. To account for the base popularity of both constituent items, we use a third measure called lift.

Measure 3: Lift. This says how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is. In Table 1, the lift of {apple → beer} is 1, which implies no association between items. A lift value greater than 1 means that item Y is *likely* to be bought if item X is bought, while a value less than 1 means that item Y is *unlikely* to be bought if item X is bought.

$$\text{Lift} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎}, \text{🍺}\}}{\text{Support} \{\text{🍎}\} \times \text{Support} \{\text{🍺}\}}$$

yProz Chowdhury

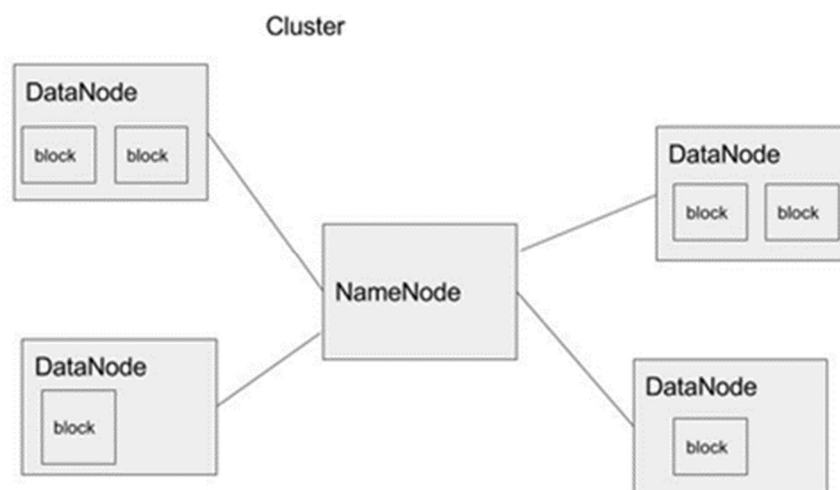


4.2. Hadoop Cluster

A Hadoop cluster is a special type of computational cluster designed specifically for storing and analysing huge amounts of unstructured data in a distributed computing environment or SUDO distributed environment.

In Hadoop clusters, first we need to define two terms: cluster and node. A cluster is a collection of nodes. A node is a process running on a virtual or physical machine or in a container. We say process because a code would be running other programs beside Hadoop.

When Hadoop is not running in cluster mode, it is said to be running in local mode. That would be suitable for, say, installing Hadoop on one machine just to learn it. When you run Hadoop in local node it writes data to the local file system instead of HDFS (Hadoop Distributed File System). Hadoop is a master-slave model, with one master (albeit with an optional High Availability hot standby) coordinating the role of many slaves. Yarn is the resource manager that coordinates what task runs where, keeping in mind available CPU, memory, network bandwidth, and storage.



One can scale out a Hadoop cluster, which means add more nodes. Hadoop is said to be linearly scalable. That means for every node you add you get a corresponding boost in throughput. More generally if you have n nodes then

Yash Chowdhury



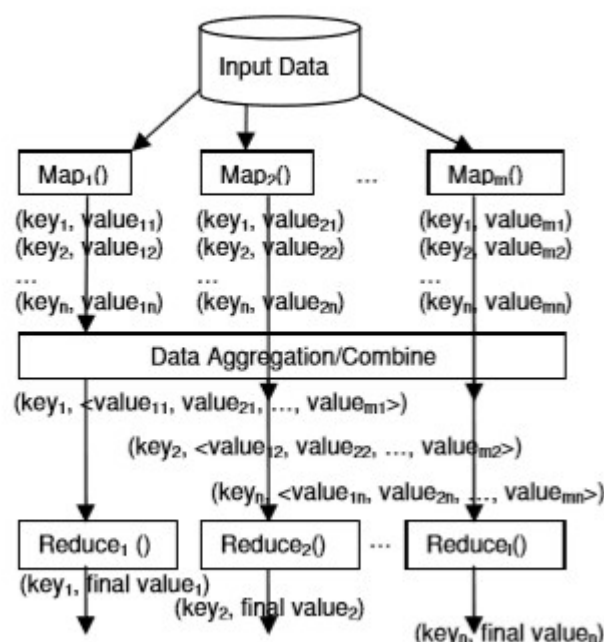
adding 1 node give you $(1/n)$ additional computing power. That type of distributed computing is a major shift from the days of using a single server where when you add memory and CPUs it produces only a marginal increase in throughput.

4.3. Map/Reduce in Hadoop

Map/Reduce is an algorithm used as functional programming. It has been received the highlight since re-introduced by Google to solve the problems to analyse huge volumes of data set in distributed computing environment. It is composed of two functions to specify, “Map” and “Reduce”. They are both defined to process data structured in (key, value) pairs.

4.3.1. Map/Reduce in parallel computing

Map/Reduce programming platform is implemented in the Apache Hadoop project that develops open-source software for reliable, scalable, and distributed computing. Hadoop can compose hundreds of nodes that process and compute peta- or tera-bytes of data working together. Hadoop was inspired by Google's MapReduce and GFS as



y Pro Chowdhury



Google has had needs to process huge data set for information retrieval and analysis. It is used by a global community of contributors such as Yahoo, Facebook, and Twitters. Hadoop's subprojects include Hadoop Common, HDFS, MapReduce, Avro, Chukwa, HBase, Hive, Mahout, Pig, and ZooKeeper etc.

The map and reduce functions run on distributed nodes in parallel. Each map operation can be processed independently on each node and all the operations can be performed in parallel. But in practice, it is limited by the data source and/or the number of CPUs near that data. The reduce functions are in the similar situation because they are from all the output of the map operations. However, Map/Reduce can handle significantly huge data sets since data are distributed on HDFS and operations move close to data for better performance. Hadoop is restricted or partial parallel programming platform because it needs to collect data of (key, value) pairs as input and parallelly computes and generates the list of (key, value) as output on map/reduce functions. In map function, the master node parts the input into smaller subproblems, and distributes those to worker nodes. Those worker nodes process smaller problems and pass the answers back to their master node. That is, map function takes inputs $(k1, v1)$ and generates $\langle k2, v2 \rangle$ where $\langle \rangle$ represents list or set. Between map and reduce, there is a combiner that resides on map node, which takes inputs $(k2, \langle v2 \rangle)$ and generates $\langle k2, v2 \rangle$. In reduce function, the master node takes the answers to all the sub-problems and combines them in some way to get the output, the answer to the problem [1, 2]. That is, reduce function takes inputs $(k2, \langle v2 \rangle)$ and generates $\langle k3, v3 \rangle$. Figure illustrates Map/Reduce control flow where each value is simply 1 and gets accumulated for the occurrence of items together in the proposed Market Basket Analysis Algorithm.

y. Prasad Chowdhury



4.3.2. Database for Big Data

Input/output files are processed on HDFS instead of using HBase DB in the paper. However, as HBase is interesting and will be integrated with the algorithm in the future, the section briefly introduces HBase. There are some drawbacks when we use RDBMS to handle huge volumes of data, like impossible deleting, slow inserting, and random failing. HBase on HDFS is distributed database that supports structured data storage for horizontally scalable tables. It is column oriented semi-structured data store. It is relatively easy to integrate with Hadoop Map/Reduce because HBase consists of a core map that is composed of keys and values - each key is associated with a value. Users store data rows in labelled tables. A data row has a sortable key and an arbitrary number of columns. The table is stored sparsely, so that rows in the same table can have different columns. Using the legacy programming languages such as Java, PHP, and Ruby, we can put data in the map as Java JDBC does for RDBMS. The file storage of HBase can be distributed over an array of independent nodes because it is built on HDFS. Data is replicated across some participating nodes. When the table is created, the table's column families are generated at the same time. We can retrieve data from HBase with the full column name in a certain form. And then HBase returns the result according to the given queries as SQL does in RDBMS.

y. Prasad Chowdhury



4.3.3. The Issues of Map/Reduce

Although there are advantages of Map/Reduce, for some researchers and educators, it is:

1. A giant step backward in the programming paradigm for large-scale data intensive applications
2. Not new at all - it represents a specific implementation of well-known techniques developed tens of years ago, especially in Artificial Intelligence
3. Data should be converted to the format of (key, value) pair for Map/Reduce, which misses most of the features that are routinely included in current DBMS
4. Incompatible with all of the tools or algorithms that have been built.

However, the issues clearly show us not only the problems but also the opportunity where we can implement algorithms with Map/Reduce approach, especially for big data set. It will give us the chance to develop new systems and evolve IT in parallel computing environment. It started a few years ago and many IT departments of companies have been moving to Map/Reduce approach in the states.

y. Prashant Chowdhury

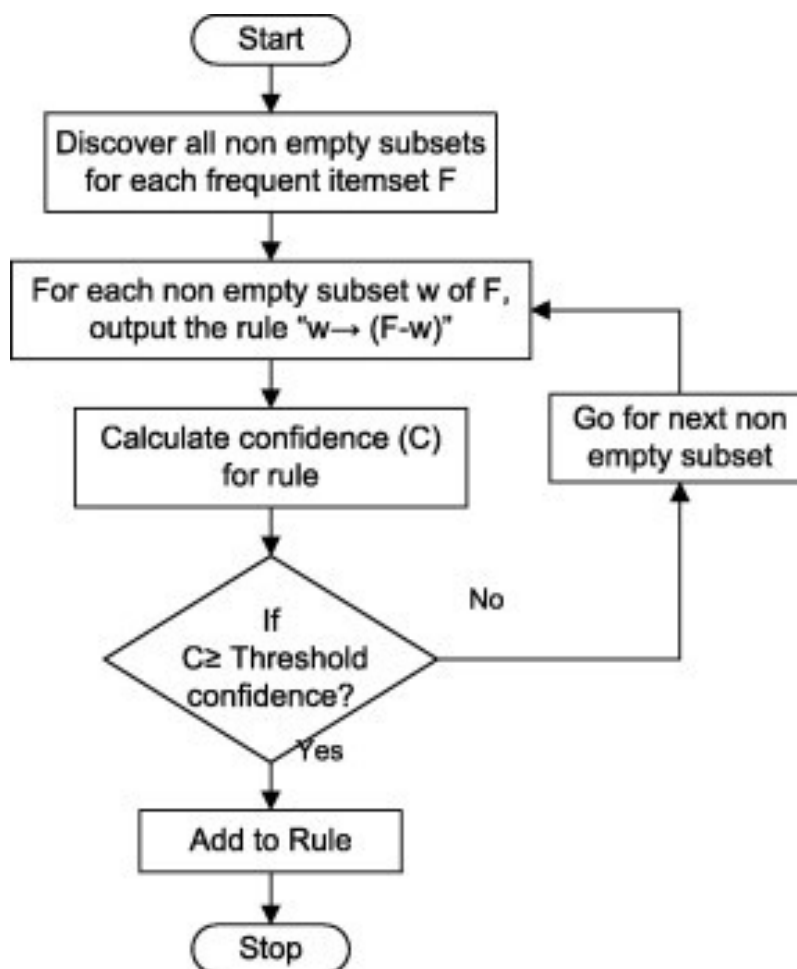


5. DATA PROCESSING

Our project incorporates the use of Hadoop Ecosystem installed on CentOS as Client. The Map-Reduce program involve the use of three mappers spanning across three different subsequent jobs. The first mapper being associated to a reducer for clubbing. The three jobs are set such that second job initialized upon the output of the first job. And subsequently the third job on the completion of the second job.

5.1. Flow-Chart of Association Rule Mining Algorithm:

The following flow chart describes the flow of data process performed in classification and calculation of the association rules for the given data set.



y. Prashant Chowdhury



5.2. Association Rule Mining Algorithm:

1. Take an input transaction text T such that, $T = \{t_1, t_2, t_3 \dots t_n\}$
2. Read every transaction from the input file and generate the data set D such that, $D = \{d_1, d_2, d_3 \dots d_n\}$
3. Generate item list for each line from transaction set for each transaction
4. Generate set of elements E such that, $E = \{e_1, e_2 \dots e_n\}$ where $e_1 = (\text{item}, \text{frequency})$ pair
5. All pairs are reducing by each key and values are summed.
6. Resultant key value pair generated.
7. Generated Key Value Pair are formed for getting Associations.
7. For each Key Value Pair the Support, Confidence and Lift are Calculated.

5.3. Mapper-Reducer Decomposition:

The algorithm used for decomposing the problem into the Association Rules involves the following steps:

1) Job 1:

- Mapper 1: Forms all possible not-null subsets of the transaction items
- Reducer 1: Calculates the total number of occurrence of occurring each subset

2) Job 2:

- Mapper 2: Associates the occurrences of the subsets with their respected transactions

3) Job 3:

- Mapper 3: Generates the association rules for each of the subsets and calculates the Support, Confidence and Lift.

y. Prashant Chowdhury



6. CODE

In this 1st Mapper-Reducer program we have calculated the total no. of sets possible from the given transaction.

Here in Mapper.java we have written a method getItemSets() where we have generated the subset of the item sets possible from the transactions by masking the data items and also counted the number of transactions by counting the no. of lines taken in the Mapper program. By overwriting the method cleanup() we write the Total Transaction at first so that we can process it first in the next Mapper programs.

In the Reducer.java we have simply counted the same kind of data set occurring frequently.

The output of the Mapper and Reducer class is Text, IntWritable. So, we are getting the output as a tab separated “part-r-” file.

6.1 Mapper 1 - MBAMapper.java:

```
package org;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MBAMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private Text itemset = new Text();
    Text keyEmit = new Text("Total Transactions");
    IntWritable valEmit = new IntWritable();
    int partialSum = 0;

    @Override
```

Y. Prashant Chowdhury



```

public void map(LongWritable key, Text value, Context context)
    throws IOException ,InterruptedException{

    partialSum++;
    String transaction = value.toString();
    List<String> itemsets = getItemSets(transaction.split(" "));
    for(String itmset : itemsets){
        itemset.set(itmset.replaceAll(" ", ""));
        context.write(itemset, new IntWritable(1));
    }
}

```

@Override

```

protected void cleanup(Context context)
    throws IOException,InterruptedException {

    valEmit.set(partialSum);
    context.write(keyEmit, valEmit);

}

```

```

private static List<String> getItemSets(String[] items) {

    List<String> itemsets = new ArrayList<String>();
    int n = items.length;
    int[] masks = new int[n];

    for (int i = 0; i < n; i++)
        masks[i] = (1<<i);

    for (int i = 0; i < (1 << n); i++){
        List<String> newList = new ArrayList<String>(n);
        for (int j = 0; j < n; j++){
            if ((masks[j] & i) != 0){
                newList.add(items[j]);
            }
            if(j == n-1 && newList.size() > 0 && newList.size() < 8){
                itemsets.add(newList.toString());
            }
        }
    }
    return itemsets;
}

```

yProz Chowdhury



}

6.2 Reducer 1 - MBAReducer.java:

```
package org;

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class MBAReducer extends Reducer<Text, IntWritable, Text,
IntWritable>
{
    private IntWritable outvalue = new IntWritable();
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
        Iterator<IntWritable> itr = values.iterator();
        int sum = 0;
        while(itr.hasNext()) {
            IntWritable val = itr.next();
            int x = val.get();
            sum = sum+x;
        }
        outvalue.set(sum);
        context.write(key, outvalue);
    }
}
```

y. Prashant Chowdhury



Output from Job 1:

```
Total Transactions  1001
[apples,apples] 5
[apples,artichok,artichok] 2
[apples,artichok,avocado] 1
[apples,artichok,chicken] 1
[apples,artichok,cracker] 1
[apples,artichok,heineken] 1
[apples,artichok,ice_crea] 1
[apples,artichok,sardines] 1
[apples,artichok,soda] 1
[apples,artichok] 31
[apples,avocado,artichok,artichok] 1
[apples,avocado,artichok,cracker] 1
[apples,avocado,artichok,heineken] 1
...
```

yProz Chowdhury



In this 2nd Mapper program we have generated all possible LHS and RHS combination sets of association rule from the given transaction.

Here in Mapper2.java again we have written the method getItemSets() where we have generated the subset of the item sets possible from the output of previous Mapper-Reducer output file by masking the data items except the first line that starts with T (i.e. Total Transaction). We write the Total Transaction value as it is, to calculate the support in the next program. Here we simply generated only the possible LHS and RHS combinations and significantly marked the count of main data item sets generated from 1st Mapper-Reducer program by “0;”. This will help us to detect the item set from which we have generated the association rules. The output of the Mapper class is Text, Text. So, we are getting the output as a text “part-m-” file.

6.3 Mapper 2 - MBAMapper2.java :

```
package org;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MBAMapper2 extends Mapper<LongWritable, Text, Text, Text>
{
    private Text outkey=new Text();
    private Text outvalue=new Text();
    private Text okey=new Text();
    private Text osub=new Text();
    private static List<String> getItemSets(String[] items) {

        List<String> itemsets = new ArrayList<String>();
```

y Pro Chowdhury



```

int n = items.length;
int[] masks = new int[n];
for (int i = 0; i < n; i++)
    masks[i] = (1<<i);
for (int i = 0; i < (1 << n); i++){
    List<String> newList = new ArrayList<String>(n);
    for (int j = 0; j < n; j++){
        if ((masks[j] & i) != 0){
            newList.add(items[j]);
        }
    }
    if(j == n-1 && newList.size() > 0 && newList.size() < 5){
        itemsets.add(newList.toString());
    }
}
return itemsets;
}

@Override
protected void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    String line = value.toString();
    if(line !=null){

        String items[]=line.split("\\t");
        if(!items[0].startsWith("T")) {
            items[0]=items[0].replaceAll(" ", "");
            String out="0;" +items[1].trim();
            String keyout=items[0];
            String count=items[1];
            String parts[]=items[0].replace("[", "").replace("]",
                "").split(",");
            outkey.set(keyout.replaceAll(" ", ""));
            outvalue.set(out);
            context.write(outkey, outvalue);

```

yPro Chowdhury



```

List<String> list=getItemSets(parts);
for(String item:list){
    String sub = item;
    String s=keyout+";" +count;
    okey.set(s);
    osub.set(sub.replaceAll(" ", ""));
    context.write(osub, okey);
}
}
else {
    context.write(new Text(items[0]), new
                    Text(items[1]));
}
}
}
}
}

```

Output from Job 2:

```

Total Transactions 1001
[apples,apples] 0;5
[apples][apples,apples];5
[apples] [apples,apples];5
[apples,apples] [apples,apples];5
[apples,artichok,artichok] 0;2
[apples][apples,artichok,artichok];2
[artichok] [apples,artichok,artichok];2
[apples,artichok] [apples,artichok,artichok];2
[artichok] [apples,artichok,artichok];2
[apples,artichok] [apples,artichok,artichok];2
[artichok,artichok] [apples,artichok,artichok];2
[apples,artichok,artichok] [apples,artichok,artichok];2
...

```

yProz Chowdhury



In this 3rd Mapper program we have generated all possible association rules from the given transaction and calculated the 3 parameters Support, Confidence and Lift.

Here in Mapper3.java we have overridden the method setup() where we have stored the distributed cache file (i.e. The output of the 1st Mapper-Reducer program) and matched the data item set with the LHS and RHS of the transaction to find the number of occurrence of the item sets in total transaction so that we can calculate the Confidence and Lift for that particular association rule.

Here, we have defined a user defined method getRHS() in which we have generated the RHS of the association rule by comparing it with the LHS and those items which are not present in the LHS are only present in the RHS, i.e. here we are generating the actual association rules. To calculate the Support, we divide the total occurrence of the association rule by the total no. of transaction that we get at the very first line of the input of mapper which we stored in a local variable “val” and then using the support and distributed cache file, the Confidence and Lift value where to calculate Confidence we matched the LHS of the association rule with the Distributed cache file and got the no. of occurrence of that particular set. Similarly to calculate the Lift, we matched the RHS with the Distributed cache file and got the occurrence of that set of items. And then we generated the output as :

Association Rule

Occurrence:

Support:

Confidence:

Lift:

In this manner. The output of the Mapper class is Text, Text. So we are getting the output as a text “part-m-” file.

y Pro Chowdhury



6.4 Mapper 3 – MBAMapper3.java:

```
package org;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;

public class MBAMapper3 extends Mapper<LongWritable, Text, Text, Text>
{

    private Text outkey = new Text();
    private Text outvalue = new Text();
    private int tno;
    private ArrayList<String> main_String = new ArrayList<String>();
    private ArrayList<Integer> count = new ArrayList<Integer>();

    @Override
    public void map(LongWritable key, Text value, Context context)
        throws IOException ,InterruptedException {

        String line = value.toString();

        if(line!=null){
```

y. Prashant Chowdhury



```

String items[]=line.split("\\t");
if (items[0].startsWith("T")) {
    tno = Integer.parseInt(items[1]);
}
if(!items[0].startsWith("T")) {
    String lhs = items[0];
    String rhs = getRHS(items,tno);
    if(!(rhs.equals("") || rhs.startsWith("0"))){
        outkey.set(lhs+"--->");
        outvalue.set(rhs);
        context.write(outkey, outvalue);
    }
}
/*else {
    context.write(new Text(items[0]), new
        Text(items[1]));
}*/
}
}

```

@Override

protected void setup(Context context) throws IOException,
InterruptedException {

```

Path path[] = context.getLocalCacheFiles();
//checking all local computers and returns all the files that have
    all distributed cache file
if(path != null && path.length>0) {
    for(Path p:path) {
        String strpath = p.toString();
        FileReader fr = new FileReader(strpath);
        BufferedReader breader = new BufferedReader(fr);
        while(true) {
            String line = breader.readLine();
            if(line == null)
                break;

```

y Pro Chowdhury



```

        String word[] = line.split("\\t");

        main_String.add(word[0]);
        count.add(Integer.parseInt(word[1]));
    }//while
    breader.close();
} //for
} //if
}

```

```

private String getRHS(String[] items, int tno) {
    String word0 = items[0];
    String word1 = items[1];
    String p = "", finOut = "";

    if(!word1.startsWith("0;")){

        String item0 = word0.replace("[", "").replace("]", "
").replace(", ", " ").trim();
        String item1 = word1.replace("[", "").replace("]", "
").replace(", ", " ").trim();

        String[] modword0 = item0.split(" ");
        String[] modword1 = item1.split(" ");

        Set<String> s1 = new HashSet<String>();

        for(String x : modword0)
            s1.add(x);

        for(String x : modword1)
            ,

```

y. Prashant Chowdhury



```

        if(!s1.contains(x))
            p = p+" "+x;
    }
    String out = p.replaceFirst(" ", "[").replace(" ", ",");
    finOut = out.replace(";", ", ");
    finOut = finOut.replace("[", ";");

    double sup;

    String[] support = finOut.split(";");
    int val = Integer.parseInt(support[1]);

    sup = (double)(val*100)/(double)tno;
    double confidence = (double)(val*100)/(double)
        (count.get(main_String.indexOf(word0)));
    double lift =0;
    try {
        lift = confidence/(double)
            (count.get(main_String.indexOf(
                support[0].replaceAll("'", ""))));
    } catch(Exception e){}
    finOut = support[0]+"\\tOccurence :
"+support[1]+"\\tSupport : "+String.valueOf(sup)+"\\tConfidence :
"+Double.valueOf(confidence)+"\\tLift : "+Double.valueOf(lift);
    }
    return finOut;
}
}

```

y. Prashant Chowdhury



Output from Job 3:

```
[apples]---> [artichok,artichok]
Occurence : 2
Support = 0.001998001998001998
Confidence : 0.006269592476489028
Lift : 0.001044932079414838
[artichok]---> [apples]
Occurence : 2
Support = 0.001998001998001998
Confidence : 0.006430868167202572
Lift : 2.0159461339193018E-5
[artichok]---> [apples]
Occurence : 2    Support = 0.001998001998001998
Confidence : 0.006430868167202572
Lift : 2.0159461339193018E-5
[artichok,artichok]---> [apples]
Occurence : 2
Support = 0.001998001998001998
Confidence : 0.3333333333333333
Lift : 0.001044932079414838
[apples]---> [artichok,avocado]
Occurence : 1
Support = 9.99000999000999E-4
Confidence : 0.003134796238244514
Lift : 7.836990595611285E-4
[artichok]---> [apples,avocado]
Occurence : 1
Support = 9.99000999000999E-4
Confidence : 0.003215434083601286
Lift : 3.0050785828049403E-5
[apples,artichok]---> [avocado]
Occurence : 1
Support = 9.99000999000999E-4
Confidence : 0.03225806451612903
Lift : 8.789663355893469E-5
[avocado]---> [apples,artichok]
Occurence : 1
Support = 9.99000999000999E-4
Confidence : 0.0027247956403269754
Lift : 8.789663355893469E-5
...
```

y. Prof Chowdhury



6.5 Driver – MBADriver.java:

```
package org;

import java.io.IOException;
import java.net.URI;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MBADriver {

    /**
     * @param args
     * @throws IOException
     * @throws InterruptedException
     * @throws ClassNotFoundException
     */

    public static void main(String[] args) throws IOException, ClassNotFoundException,
    InterruptedException {

        // TODO Auto-generated method stub
```

Y. Prashant Chowdhury



```
Configuration conf = new Configuration();
```

```
Job job1 = Job.getInstance(conf);
```

```
job1.setJarByClass(MBADriver.class);
```

```
job1.setMapperClass(MBAMapper.class);
```

```
job1.setReducerClass(MBAReducer.class);
```

```
job1.setOutputKeyClass(Text.class);
```

```
job1.setOutputValueClass(IntWritable.class);
```

```
job1.setNumReduceTasks(1);
```

```
FileInputFormat.addInputPath(job1,new Path("MBAdir"));
```

```
FileOutputFormat.setOutputPath(job1, new Path("MBAout1"));
```

```
boolean success = job1.waitForCompletion(true);
```

```
if(success) {
```

```
    Job job2 = Job.getInstance(conf);
```

```
    job2.setJarByClass(MBADriver.class);
```

```
    job2.setMapperClass(MBAMapper2.class);
```

```
    job2.setMapOutputKeyClass(Text.class);
```

```
    job2.setMapOutputKeyClass(Text.class);
```

y Pro Chowdhury



```
job2.setOutputKeyClass(Text.class);
job2.setOutputValueClass(Text.class);

job2.setNumReduceTasks(0);

FileInputFormat.addInputPath(job2,new Path("MBAout1"));

FileOutputFormat.setOutputPath(job2, new Path("MBAout2"));

boolean success2 = job2.waitForCompletion(true);

if(success2) {

    Job job3 = Job.getInstance(conf);

    Path setpath = new Path("MBAout1/part-r-00000");
    URI uri = setpath.toUri();

    job3.addCacheFile(uri);
    job3.setJarByClass(MBADriver.class);
    job3.setMapperClass(MBAMapper3.class);

    job3.setMapOutputKeyClass(Text.class);
    job3.setMapOutputValueClass(Text.class);
    job3.setOutputKeyClass(Text.class);
    job3.setOutputValueClass(Text.class);
```

y. Prashant



```
job3.setNumReduceTasks(0);

FileInputFormat.addInputPath(job3,new
    Path("MBAout2"));

FileOutputFormat.setOutputPath(job3, new
    Path("MBAout3"));

job3.waitForCompletion(true);

}

}

}

}
```

y. Prashant Choudhary



7. IMPLEMENTATIONS OF MARKET BASKET ANALYSIS

When one hears Market Basket Analysis, one thinks of shopping carts and supermarket shoppers. It is important to realize that there are many other areas in which Market Basket Analysis can be applied. An example of Market Basket Analysis for a majority of Internet users is a list of potentially interesting products for Amazon. Amazon informs the customer that people who bought the item being purchased by them, also reviewed or bought another list of items. A list of applications of Market Basket Analysis in various industries is listed below:

1. Store Layout:

Based on the insights from market basket analysis you can organize your store to increase revenues. Items that go along with each other should be placed near each other to help consumers notice them. This will guide the way a store should be organized to shoot for best revenues. With the help of this data you can eliminate the guesswork while determining the optimal store layout.

2. Marketing Messages:

Whether it is email, phone, social media or an offer by a direct salesman, market basket analysis can improve the efficiency of all of them. By using data from MBA, you can suggest the next best product which a customer is likely to buy. Hence you will help your customers with fruitful suggestions instead of annoying them with marketing blasts.

3. Maintain Inventory:

Based on the inputs from MBA you can also predict future purchases of customers over a period of time. Using your initial sales data, you can predict which item would probably fall short and maintain stocks in optimal quality. This will help you improve the allocations of resources to different items of the inventory.

4. Content Placement:

In case of e-commerce businesses, website content placement is very important. If goods are displayed in right order than it can help boost conversions. MBA

y. Prashant Chowdhury



can also be used by online publishers and bloggers to display content which consumer is most likely to read next. This will reduce bounce rate, improve engagement and result in better performance in search results.

5. Recommendation Engines:

Recommendation engines are already used by some popular companies like Netflix, Amazon, Facebook, etc. If you want to create an effective recommendation system for your company then you will also need market basket analysis to efficiently maintain one. MBA can be considered as the basis for creating a recommendation engine.

y. Prashant Chowdhury



8. FUTURE IMPLEMENTATION

This research essentially deals with effective recommendation system for the supermarket and agriculture for identifying the most frequently purchased items and suitable crop respectively using association rule techniques. These proposed techniques are much useful in both supermarket and agriculture. The proposed Fast Adaptive

Association Rule Mining approach provides the best result among the proposed techniques. In order to improve the performance of these proposed approaches, some future enhancements have to be done. The main aim of the future enhancements would be to increase the effectiveness of the recommendation system.

The future work would be some modification of other algorithms to generate the association rules that can be adopted on existing recommendation system to make them functionally more effective.

Better and effective rule mining techniques can be used for better performance of the recommendation system.

Effective Neural Network techniques can be incorporated with the Association Rule Mining to increase the accuracy of the recommendation system.

Y. Prasad Chowdhury



9. CONCLUSION

Frequent pattern mining has been a focused theme in data mining research for over a decade. Abundant literature has been dedicated to this research and tremendous progress has been made, ranging from efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining, associative classification, and frequent pattern-based clustering, as well as their broad applications. It is believed that frequent pattern mining research has substantially broadened the scope of data analysis and will have deep impact on data mining methodologies and applications in the long run. However, there are still some challenging research issues that need to be solved before frequent pattern mining can claim a cornerstone approach in data mining applications.

HDFS and MapReduce play an important role in reducing the processing time for large datasets. However, most of the algorithms have limitation of processing speed. In this paper, Hadoop based distributed approach is presented which processes the transactional dataset into partitions and transfers the task to all participating nodes. The purpose of this, is to reduce inter node message passing in the cluster. The DFPM algorithm generates a smaller candidate set and uses a less message passing than CDA and FDM algorithm, thus the execution time of the DFPM algorithm is less as compare to others. In the first phase, the DFPM algorithm generates distributed frequent itemset, and, association rules are generated from those frequent itemset. In the second phase, the proposed MR-CIRD algorithm is used to detect consistent and inconsistent rules when the data is distributed geographically. This will help the organization to improve the marketing strategy for the zone where the inconsistent rules are more. Performance studies have shown that the distributed computing tasks scale linearly with the number of nodes. It is observed that for some region the number of inconsistent rules is relatively less even though the number of consistent rules is more. The proposed algorithm is more flexible, scalable and efficient for mining huge amount of data.

The time efficiency of the algorithm may be improved by using FP-tree based data structures for the candidate itemset generation. Further, the work can be extended by considering the different weights for each interestingness measures and find weighted interesting association rule.

Y. Pro. Chowdhury

