Experiment-6B: Apriori Algorithm

Source Code:

```
import java.io. * ;

import java.util. * ;

public class AprioriDataMining {

                Vector < String > candidates = new Vector < String > ();

                 List < String > itemSet = new ArrayList < String > ();

                 List < String > finalFrequentItemSet = new ArrayList < >();

              HashMap < String,

                Integer > frequentItems = new HashMap < String,

             Integer > ();

               String newLine = System.getProperty("line.separator");

 int itemsCount, countItemOccurrence = 0,
displayFrequentItemSetNumber = 2,

              displayTransactionNumber = 1;

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

                                BufferedReader br = new BufferedReader(new InputStreamReader(System. in

));

                        int noOfTransactions,
minimumSupport;

                          double minimumConfidence;

                          System.out.println("Reached till here!");

                          String sampleFile = args[0];

                          System.out.println(sampleFile);

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

```
String newLine = System.getProperty("line.separator");
System.out.println(newLine + "'APRIORI ALGORITHM'"); System.out.print("Enter the
Minimum Support = "); minimumSupport = Integer.parseInt(br.readLine());
System.out.print("Enter the Minimum Confidence (in %) = "); minimumConfidence =
Double.parseDouble(br.readLine());  minimumConfidence = minimumConfidence / 100;

File file = new File(sampleFile);

Scanner sc = new Scanner(file);

while (sc.hasNextLine()) { String str =
sc.nextLine();  transactions.add(str);

}

noOfTransactions = transactions.size();

ApprioriDataMining a = new ApprioriDataMining();

a.display(noOfTransactions, transactions, minimumSupport, minimumConfidence);

}

public void display(int noOfTransactions, List < String > transactions, int

minimumSupport, double minimumConfidence) {

for (int i = 0; i < noOfTransactions; i++) {

String str = transactions.get(i); String[] words =
str.split(" ");  int count = words.length;

for (int j = 0; j < count; j++) {     if (i == 0) {

itemSet.add(words[j]);

}

else {

if (! (itemSet.contains(words[j]))) {

itemSet.add(words[j]);

}

}

}

}

itemsCount = itemSet.size();

System.out.println(newLine + "No of Items = " + itemsCount);

System.out.println("No of Transactions = " + noOfTransactions);
```

```
 System.out.println("Minimum Support = " + minimumSupport);
System.out.println("Minimum Confidence = " + minimumConfidence + newLine);

        System.out.println("'Items present in the Database'"); for (String i:
                itemSet) {

                        System.out.println(" -----> " + i);

                }

        System.out.println(newLine + "TRANSACTION ITEMSET"); for (String i:
transactions) {

                                System.out.println("Transaction " + displayTransactionNumber + " = "

+ i);

                        displayTransactionNumber++;

                }

 firstFrequentItemSet(noOfTransactions, transactions, minimumSupport, minimumConfidence);


        }


        public void firstFrequentItemSet(int noOfTransactions, List < String > transactions, int
minimumSupport, double minimumConfidence) {

                 System.out.println();

 System.out.println("Frequent Itemset 1"); for (int items = 0; items <
itemSet.size(); items++) {

                        countItemOccurrence = 0;

  String itemStr = itemSet.get(items);  for (int t = 0; t < noOfTransactions;
t++) {    String transactionStr = transactions.get(t);    if
(transactionStr.contains(itemStr)) {

                                        countItemOccurrence++;

                        }

                }


  if (countItemOccurrence >= minimumSupport) {    System.out.println(itemStr + "
=> support = " + countItemOccurrence);

                        finalFrequentItemSet.add(itemStr);
frequentItems.put(itemStr, countItemOccurrence);

                }

        }
```

```
                aprioriStart(noOfTransactions, transactions, minimumSupport,
minimumConfidence);
        }

            public void aprioriStart(int noOfTransactions, List < String > transactions, int
minimumSupport, double minimumConfidence) {
                int itemsetNumber = 1;
        for (int i = 0; i < finalFrequentItemSet.size(); i++) { String str =
                finalFrequentItemSet.get(i);
                        candidates.add(str);
                }
                do {
                        itemsetNumber++;
                       generateCombinations(itemsetNumber);
checkFrequentItems(noOfTransactions, transactions, minimumSupport);
                }
                  while ( candidates . size () > 1);

                    System.out.println("Association Rules for Frequent Itemset" + newLine);

                generateAssociationRules(noOfTransactions, transactions, minimumConfidence);
        }

          private void generateCombinations(int itr) {
                    Vector < String > candidatesTemp = new Vector < String > ();
                String s1,
                s2;
                  StringTokenizer strToken1,
                strToken2;
                 if (itr == 2) {
  for (int i = 0; i < candidates.size(); i++) {     strToken1 = new
StringTokenizer(candidates.get(i));     s1 = strToken1.nextToken();
                                        for (int j = i + 1; j < candidates.size(); j++) {
```

```
                                        strToken2 = new
StringTokenizer(candidates.elementAt(j));
                                          s2 = strToken2.nextToken();
                                        String addString = s1 + " " + s2;
candidatesTemp.add(addString);
                                }
                        }
                }
                else {
                        for (int i = 0; i < candidates.size(); i++) {
                                for (int j = i + 1; j < candidates.size(); j++) {
                                s1 = new String();
s2 = new String();
                                        strToken1 = new StringTokenizer(candidates.get(i));
                                        strToken2 = new StringTokenizer(candidates.get(j));
                                      for (int s = 0; s < itr - 2; s++) {
                        s1 = s1 + " " + strToken1.nextToken();                  s2
                = s2 + " " + strToken2.nextToken();
                                      }
   if (s2.compareToIgnoreCase(s1) == 0) {      String addString = (s1 + " " +
strToken1.nextToken() + " " + strToken2.nextToken()).trim();
                                              candidatesTemp.add(addString);
                                      }
                                }
                        }
                }
                 candidates.clear();
                    candidates = new Vector < String > (candidatesTemp);
                 candidatesTemp.clear();
                 System.out.println();
        }
```

```
 public void checkFrequentItems(int noOfTransactions, List < String > transactions, int
minimumSupport) {

                        List < String > combList = new ArrayList < String > ();

  for (int i = 0; i < candidates.size(); i++) {  String str =
candidates.get(i);   combList.add(str);

                    }

                       System.out.println("Frequent Itemset " + displayFrequentItemSetNumber);

                       for (int i = 0; i < combList.size(); i++) {

                            String str = combList.get(i);
          String[] words = str.split(" ");

   int count = words.length;    int flag = 0,

                             itemSetOccurence = 0;

   for (int t = 0; t < noOfTransactions; t++) {     String transac =
transactions.get(t);     for (int j = 0; j < count; j++) {      String wordStr =
words[j];      if (transac.contains(wordStr)) {       flag++;

                                                  }

                                        }

                                         if (flag == count) {

                                                 itemSetOccurence++;

                                        }

                                         flag = 0;

                              }

                                 if (itemSetOccurence >= minimumSupport) {

                                          System.out.println(str + " => support = " + itemSetOccurence);

                                   frequentItems.put(str, itemSetOccurence);

                                   finalFrequentItemSet.add(str);

                              }

                               itemSetOccurence = 0;

                    }

                     displayFrequentItemSetNumber++;

          }
```

```
public void generateAssociationRules(int noOfTransactions, List < String > transactions,
double minimumConfidence) {

                double confidence,

                confidence1;


                  for (int i = 0; i < finalFrequentItemSet.size(); i++) {

                        int spring2019count = 0;

                         String itemSetStr = finalFrequentItemSet.get(i);

                         double value = frequentItems.get(itemSetStr);

                       String str = "",
        str1 = "";

  String[] words = itemSetStr.split(" "); int wordCountInString =
words.length; if (wordCountInString == 2) /* for
FrequentItemSet = 2*/

                              {

   double s = frequentItems.get(words[0]);     confidence = value / s;

                                    spring2019count++;

                                      if (confidence >= minimumConfidence) {

                                              System.out.println(words[0] + " -> " + words[1] + " =
Confidence = " + confidence * 100 + " and Support = " + (int) value + "");


                                    }

                                   double s1 = frequentItems.get(words[1]);
        confidence = value / s1;

                                     spring2019count++;

                                      if (confidence >= minimumConfidence) {

                                              System.out.println(words[1] + " -> " + words[0] + " =
Confidence = " + confidence * 100 + " and Support = " + (int) value + "");

                                    }

                              }

                              else

                               /* for FrequentItemSet > 2 */

                              {
```

```
                                    for (int a = 0; a < wordCountInString - 1; a++) {

                                        if (a == 0) {
                                    str = str + words[a];

                                            spring2019count++;

                                        }

                                        else {

                                            str = str + " " + words[a];
                        spring2019count++;

                                        }

                                            for (int j = a + 1; j < wordCountInString; j++) {

                                                {

                                                    str1 = str1 + " " + words[j];
                                    spring2019count++;

                                                }

                                        }

    double s = frequentItems.get(str);      confidence = value / s;      String st = str1.trim();
double s1 = frequentItems.get(st);      confidence1 = value / s1;      if (confidence >=
minimumConfidence) {       System.out.println(str + " -> " + str1 + " =

Confidence = " + confidence * 100 + " and Support = " + (int) value + "");


                                        }

    if (confidence1 >= minimumConfidence) {       System.out.println(st + " -> " + str + " =

Confidence = " + confidence1 * 100 + " and Support = " + (int) value + "");


                                        }

                                        str1 = "";

                                    }

   str = "";     str1 = "";

                                }

                        }

                }

}
```

Output:

```
'APRIORI ALGORITHM'
Enter the Minimum Support = 2
Enter the Minimum Confidence (in %) = 70

No of Items = 5
No of Transactions = 9
Minimum Support = 2
Minimum Confidence = 0.7


TRANSACTION ITEMSET
Transaction 1 = i1 i2 i5
Transaction 2 = i2 i4
Transaction 3 = i2 i3
Transaction 4 = i1 i2 i4
Transaction 5 = i1 i3
Transaction 6 = i2 i3
Transaction 7 = i1 i3
Transaction 8 = i1 i2 i3 i5
Transaction 9 = i1 i2 i3

Frequent Itemset 1
i1 => support = 6
i2 => support = 7
i5 => support = 2
i4 => support = 2
i3 => support = 6
```

```
Frequent Itemset 2
i1 i2 => support = 4
i1 i5 => support = 2
i1 i3 => support = 4
i2 i5 => support = 2
i2 i4 => support = 2
i2 i3 => support = 4

Frequent Itemset 3
i1 i2 i5 => support = 2
i1 i2 i3 => support = 2

Frequent Itemset 4

Frequent Itemset 5
Association Rules for Frequent Itemset

i5 -> i1 = Confidence = 100.0 and Support = 2
i5 -> i2 = Confidence = 100.0 and Support = 2
i4 -> i2 = Confidence = 100.0 and Support = 2
i2 i5 -> i1 = Confidence = 100.0 and Support = 2
i5 -> i1 i2 = Confidence = 100.0 and Support = 2
```