**BAYES**

import java.util.\*;

import java.io.\*;

class element

{

double p[][];

element(int n,int m)

{

p = new double[n][m];

}

}

class Classifier

{

int no\_attr,no\_rows=0;

String fileArray[][];

String values[][];

double class\_p[];

int count;

element a[];

void readFile(String fname)throws IOException

{

FileInputStream in = null;

try

{

File f = new File(fname);

in = new FileInputStream(f);

}

catch (Exception e){}

BufferedReader br = new BufferedReader(new InputStreamReader(in));

String input="";

input = br.readLine();

StringTokenizer line = new StringTokenizer(input);

no\_attr=line.countTokens()-1;

fileArray = new String[100][no\_attr+1];

for(int i=0;i<=no\_attr;i++)

fileArray[no\_rows][i]=line.nextToken();

while(true)

{

input = br.readLine();

if(input==null)

break;

line = new StringTokenizer(input);

no\_rows++;

for(int i=0;i<=no\_attr;i++)

fileArray[no\_rows][i]=line.nextToken();

}

getAllvalues();

createTable();

newEntry();

}

boolean in\_values(int col\_no,String temp)

{

for(int i=0;i<=count;i++)

if(values[i][col\_no]!=null && values[i][col\_no].equals(temp))

return true;

else

continue;

return false;

}

void getAllvalues()

{

values = new String[100][no\_attr+1];

count=0;

String temp;

for(int i=0;i<=no\_attr;i++)

{

for(int j=1;j<=no\_rows;j++)

{

temp=fileArray[j][i];

if(in\_values(i,temp))

continue;

values[count++][i]=temp;

}

count=0;

}

}

int getlen(int col\_no)

{

int i=0;

while(values[i][col\_no]!=null)

i++;

return i;

}

void display()

{

for(int i=0;i<=no\_rows;i++)

{

for(int j=0;j<=no\_attr;j++)

System.out.println(values[i][j]+" ");

System.out.println();

}

}

int getcount(String temp,int col\_no)

{

int tc=0;

for(int i=1;i<=no\_rows;i++)

if(fileArray[i][col\_no].equals(temp))

tc++;

return tc;

}

void createTable()

{

int tp = getlen(no\_attr);

class\_p=new double[tp];

for(int i=0;i<tp;i++)

{

for(int j=1;j<=no\_rows;j++)

{

if(values[i][no\_attr].equals(fileArray[j][no\_attr]))

class\_p[i]++;

}

class\_p[i]/=no\_rows;

System.out.println("P("+values[i][no\_attr]+")= "+class\_p[i]);

}

int tc=0;

a=new element[no\_attr];

for(int i=0;i<no\_attr;i++)

{

a[i] = new element(getlen(i),getlen(no\_attr));

for(int j=0;j<getlen(i);j++)

{

for(int k=0;k<getlen(no\_attr);k++)

{

for(int x=1;x<=no\_rows;x++)

{

if(values[j][i].equals(fileArray[x][i]) && values[k][no\_attr].equals(fileArray[x][no\_attr]))

tc++;

}

a[i].p[j][k]=(double)tc/getcount(values[k][no\_attr],no\_attr);

tc=0;

}

}

}

}

void newEntry()

{

Scanner s = new Scanner(System.in);

System.out.println("Enter new Entry");

String entry[] = new String[no\_attr];

double p\_entry[] = new double[getlen(no\_attr)];

String X="X=< ";

for(int i=0;i<no\_attr;i++)

{

System.out.println("Enter "+fileArray[0][i]);

entry[i]=s.next();

X+=entry[i]+" ";

}

X+=">";

System.out.println("The unseen sample is "+X+"\n");

double large=0.0;

int pos=-1;

for(int i=0;i<getlen(no\_attr);i++)

{

double product=1.0;

for(int j=0;j<no\_attr;j++)

{

product\*=a[j].p[getindex(j,entry[j])][i];

}

p\_entry[i]=class\_p[i]\*product;

System.out.println("P(X|"+values[i][no\_attr]+").P("+values[i][no\_attr]+")="+p\_entry[i]);

if(p\_entry[i]>large)

{

large=p\_entry[i];

pos=i;

}

}

System.out.println("The DECISION is "+values[pos][no\_attr]);

}

int getindex(int col\_no,String temp)

{

for(int i=0;i<getlen(col\_no);i++)

{

if(values[i][col\_no].equals(temp))

return i;

}

System.out.println("INVALID ENTRY");

return 32676;

}

}

class Bayes

{

public static void main(String args[])throws Exception

{

Scanner s = new Scanner(System.in);

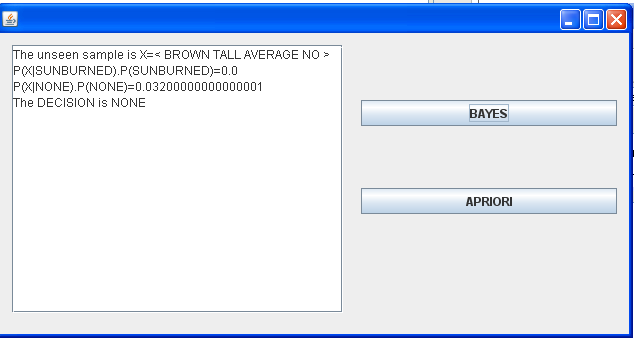
Classifier c= new Classifier();

System.out.println("Enter the name of the input file with its extension ");

c.readFile(s.next());

}

}



**APRIORI**

import java.io.\*;

import java.util.\*;

public class Apriori {

public static void main(String[] args) {

AprioriCalculation ap = new AprioriCalculation();

ap.aprioriProcess();

}

}

class AprioriCalculation

{

Vector<String> candidates=new Vector<String>(); //the current candidates

String configFile="config.txt"; //configuration file

String transaFile="transa.txt"; //transaction file

String outputFile="apriori-output.txt";//output file

int numItems; //number of items per transaction

int numTransactions; //number of transactions

double minSup; //minimum support for a frequent itemset

String oneVal[]; //array of value per column that will be treated as a '1'

String itemSep = " "; //the separator value for items in the database

public void aprioriProcess()

{

Date d; //date object for timing purposes

long start, end; //start and end time

int itemsetNumber=0; //the current itemset being looked at

//get config

getConfig();

System.out.println("Apriori algorithm has started.\n");

//start timer

d = new Date();

start = d.getTime();

//while not complete

do

{

//increase the itemset that is being looked at

itemsetNumber++;

//generate the candidates

generateCandidates(itemsetNumber);

//determine and display frequent itemsets

calculateFrequentItemsets(itemsetNumber);

if(candidates.size()!=0)

{

System.out.println("Frequent " + itemsetNumber + "-itemsets");

System.out.println(candidates);

}

//if there are <=1 frequent items, then its the end. This prevents reading through the database again. When there is only one frequent itemset.

}while(candidates.size()>1);

//end timer

d = new Date();

end = d.getTime();

//display the execution time

System.out.println("Execution time is: "+((double)(end-start)/1000) + " seconds.");

}

public static String getInput()

{

String input="";

//read from System.in

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

//try to get users input, if there is an error print the message

try

{

input = reader.readLine();

}

catch (Exception e)

{

System.out.println(e);

}

return input;

}

private void getConfig()

{

FileWriter fw;

BufferedWriter file\_out;

String input="";

//ask if want to change the config

System.out.println("Default Configuration: ");

System.out.println("\tRegular transaction file with '" + itemSep + "' item separator.");

System.out.println("\tConfig File: " + configFile);

System.out.println("\tTransa File: " + transaFile);

System.out.println("\tOutput File: " + outputFile);

System.out.println("\nPress 'C' to change the item separator, configuration file and transaction files");

System.out.print("or any other key to continue. ");

input=getInput();

if(input.compareToIgnoreCase("c")==0)

{

System.out.print("Enter new transaction filename (return for '"+transaFile+"'): ");

input=getInput();

if(input.compareToIgnoreCase("")!=0)

transaFile=input;

System.out.print("Enter new configuration filename (return for '"+configFile+"'): ");

input=getInput();

if(input.compareToIgnoreCase("")!=0)

configFile=input;

System.out.print("Enter new output filename (return for '"+outputFile+"'): ");

input=getInput();

if(input.compareToIgnoreCase("")!=0)

outputFile=input;

System.out.println("Filenames changed");

System.out.print("Enter the separating character(s) for items (return for '"+itemSep+"'): ");

input=getInput();

if(input.compareToIgnoreCase("")!=0)

itemSep=input;

}

try

{

FileInputStream file\_in = new FileInputStream(configFile);

BufferedReader data\_in = new BufferedReader(new InputStreamReader(file\_in));

//number of items

numItems=Integer.valueOf(data\_in.readLine()).intValue();

//number of transactions

numTransactions=Integer.valueOf(data\_in.readLine()).intValue();

//minsup

minSup=(Double.valueOf(data\_in.readLine()).doubleValue());

//output config info to the user

System.out.print("\nInput configuration: "+numItems+" items, "+numTransactions+" transactions, ");

System.out.println("minsup = "+minSup+"%");

System.out.println();

minSup/=100.0;

oneVal = new String[numItems];

System.out.print("Enter 'y' to change the value each row recognizes as a '1':");

if(getInput().compareToIgnoreCase("y")==0)

{

for(int i=0; i<oneVal.length; i++)

{

System.out.print("Enter value for column #" + (i+1) + ": ");

oneVal[i] = getInput();

}

}

else

for(int i=0; i<oneVal.length; i++)

oneVal[i]="1";

//create the output file

fw= new FileWriter(outputFile);

file\_out = new BufferedWriter(fw);

//put the number of transactions into the output file

file\_out.write(numTransactions + "\n");

file\_out.write(numItems + "\n\*\*\*\*\*\*\n");

file\_out.close();

}

//if there is an error, print the message

catch(IOException e)

{

System.out.println(e);

}

}

private void generateCandidates(int n)

{

Vector<String> tempCandidates = new Vector<String>(); //temporary candidate string vector

String str1, str2; //strings that will be used for comparisons

StringTokenizer st1, st2; //string tokenizers for the two itemsets being compared

//if its the first set, candidates are just the numbers

if(n==1)

{

for(int i=1; i<=numItems; i++)

{

tempCandidates.add(Integer.toString(i));

}

}

else if(n==2) //second itemset is just all combinations of itemset 1

{

//add each itemset from the previous frequent itemsets together

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

{

st1 = new StringTokenizer(candidates.get(i));

str1 = st1.nextToken();

for(int j=i+1; j<candidates.size(); j++)

{

st2 = new StringTokenizer(candidates.elementAt(j));

str2 = st2.nextToken();

tempCandidates.add(str1 + " " + str2);

}

}

}

else

{

//for each itemset

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

{

//compare to the next itemset

for(int j=i+1; j<candidates.size(); j++)

{

//create the strigns

str1 = new String();

str2 = new String();

//create the tokenizers

st1 = new StringTokenizer(candidates.get(i));

st2 = new StringTokenizer(candidates.get(j));

//make a string of the first n-2 tokens of the strings

for(int s=0; s<n-2; s++)

{

str1 = str1 + " " + st1.nextToken();

str2 = str2 + " " + st2.nextToken();

}

//if they have the same n-2 tokens, add them together

if(str2.compareToIgnoreCase(str1)==0)

tempCandidates.add((str1 + " " + st1.nextToken() + " " + st2.nextToken()).trim());

}

}

}

//clear the old candidates

candidates.clear();

//set the new ones

candidates = new Vector<String>(tempCandidates);

tempCandidates.clear();

}

private void calculateFrequentItemsets(int n)

{

Vector<String> frequentCandidates = new Vector<String>(); //the frequent candidates for the current itemset

FileInputStream file\_in; //file input stream

BufferedReader data\_in; //data input stream

FileWriter fw;

BufferedWriter file\_out;

StringTokenizer st, stFile; //tokenizer for candidate and transaction

boolean match; //whether the transaction has all the items in an itemset

boolean trans[] = new boolean[numItems]; //array to hold a transaction so that can be checked

int count[] = new int[candidates.size()]; //the number of successful matches

try

{

//output file

fw= new FileWriter(outputFile, true);

file\_out = new BufferedWriter(fw);

//load the transaction file

file\_in = new FileInputStream(transaFile);

data\_in = new BufferedReader(new InputStreamReader(file\_in));

//for each transaction

for(int i=0; i<numTransactions; i++)

{

//System.out.println("Got here " + i + " times"); //useful to debug files that you are unsure of the number of line

stFile = new StringTokenizer(data\_in.readLine(), itemSep); //read a line from the file to the tokenizer

//put the contents of that line into the transaction array

for(int j=0; j<numItems; j++)

{

trans[j]=(stFile.nextToken().compareToIgnoreCase(oneVal[j])==0); //if it is not a 0, assign the value to true

}

//check each candidate

for(int c=0; c<candidates.size(); c++)

{

match = false; //reset match to false

//tokenize the candidate so that we know what items need to be present for a match

st = new StringTokenizer(candidates.get(c));

//check each item in the itemset to see if it is present in the transaction

while(st.hasMoreTokens())

{

match = (trans[Integer.valueOf(st.nextToken())-1]);

if(!match) //if it is not present in the transaction stop checking

break;

}

if(match) //if at this point it is a match, increase the count

count[c]++;

}

}

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

{

// System.out.println("Candidate: " + candidates.get(c) + " with count: " + count + " % is: " + (count/(double)numItems));

//if the count% is larger than the minSup%, add to the candidate to the frequent candidates

if((count[i]/(double)numTransactions)>=minSup)

{

frequentCandidates.add(candidates.get(i));

//put the frequent itemset into the output file

file\_out.write(candidates.get(i) + "," + count[i]/(double)numTransactions + "\n");

}

}

file\_out.write("-\n");

file\_out.close();

}

//if error at all in this process, catch it and print the error messate

catch(IOException e)

{

System.out.println(e);

}

//clear old candidates

candidates.clear();

//new candidates are the old frequent candidates

candidates = new Vector<String>(frequentCandidates);

frequentCandidates.clear();

}

}

