TRP RANK FINDER

package netflix;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

import java.util.TreeMap;

/\*\*

\*

\* @author Soft

\*/

public class netflix {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*NETFLIX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

String dataset="";

try

{

File f =new File("netflix.csv");

FileInputStream fis=new FileInputStream(f);

byte data[]=new byte[fis.available()];

fis.read(data);

fis.close();

dataset=new String(data);

System.out.println(dataset.trim());

System.out.println();

List<String> aa = new ArrayList();

}

catch(Exception e)

{

e.printStackTrace();

}

System.out.println("\*\*\*\*\*\*\*\*\*K-Neearest Neighbour Algorithm\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

try

{

ArrayList title=new ArrayList();

String st[]=dataset.trim().split("\n");

for(int i=1;i<st.length;i++)

{

String row[]=st[i].trim().split("\t");

if(!(title.contains(row[0].trim())))

{

title.add(row[0].trim());

}

}

int k=0;

for(int j=0;j<title.size();j++)

{

String t=title.get(j).toString().trim();

int count=0;

for(int i=1;i<st.length;i++)

{

String row[]=st[i].trim().split("\t");

if(t.trim().equals(row[0].trim()))

{

count++;

}

}

if(j==0)

{

k=count;

}

else

{

if(count>k)

{

k=count;

}

}

}

for(int j=0;j<title.size();j++)

{

String t=title.get(j).toString().trim();

for(int i=1;i<st.length;i++)

{

String row[]=st[i].trim().split("\t");

if(t.trim().equals(row[0].trim()))

{

System.out.println(st[i].trim());

}

}

System.out.println();

}

System.out.println();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*Singular Value Decomposition\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

System.out.println(st[0].trim()+"\tTRP");

//ArrayList trponly=new ArrayList();

//ArrayList trpwithTitle=new ArrayList();

for(int j=0;j<title.size();j++)

{

String t=title.get(j).toString().trim();

int viewCount=0;

String svd="";

double trp=0;

for(int i=1;i<st.length;i++)

{

String ro[]=st[i].trim().split("\t");

if(t.trim().equals(ro[0].trim()))

{

double fortrp=(Double.parseDouble(ro[3].trim())+Double.parseDouble(ro[5].trim())+Double.parseDouble(ro[6].trim()))/(double)3;

trp=trp+fortrp;

viewCount++;

if(viewCount==1)

{

svd=st[i].trim();

//System.out.println("\t"+t);

//System.out.println("\t"+fortrp);

}

}

}

System.out.println();

System.out.println(svd.trim()+"\t"+trp);

}

System.out.println();

System.out.println();

System.out.println();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TRP FINDED \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

HashMap<Double, String> rank = new HashMap<Double, String>();

for(int j=0;j<title.size();j++)

{

String t=title.get(j).toString().trim();

int viewCount=0;

String svd="";

double trp=0;

for(int i=1;i<st.length;i++)

{

String ro[]=st[i].trim().split("\t");

if(t.trim().equals(ro[0].trim()))

{

double fortrp=(Double.parseDouble(ro[3].trim())+Double.parseDouble(ro[5].trim())+Double.parseDouble(ro[6].trim()))/(double)3;

trp=trp+fortrp;

viewCount++;

if(viewCount==1)

{

svd=st[i].trim();

System.out.println(t+"\t"+fortrp);

rank.put(fortrp, t);

}

}

}

}

System.out.println();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TRP RANKING FOR CHANNEL\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println();

Map<Double, String> map = new TreeMap<Double, String>(rank);

Map<Double, String> reverseSortedMap = new TreeMap<Double, String>(Collections.reverseOrder());

reverseSortedMap.putAll(map);

Set set = reverseSortedMap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext())

{

Map.Entry me = (Map.Entry)iterator.next();

System.out.println(me.getValue()+ " : "+me.getKey());

}

}

}

}