Проект 1.9

по

Мрежово Програмиране

Изготвила Диана Иванова

ФН: 81426

Съдържание:

1.Тема....................................................................................1  
2. Теоретично описание на решението на проблема........1  
3. Проектиране......................................................................3  
4. Реализация ........................................................................3  
5. Ръководство за инсталация. .............................................4  
6. Резултати - резултати от работата на приложението…...4  
7. Литература .........................................................................  
8. Приложение…...................................................................

1.Тема  
Клиент и сървър приложения чрез TCP протокола за търсене на зависимост на поведение на потребител.

Клиентът изпраща до сървър лог файл със структура:

Time;Event context;Component;Event name;Description;Origin;IP address

Сървърът анализира поведението на потребителите на база данните от файла с помощта на откриване на шаблон на данните чрез реализирането на NegFin алгоритъм за извличането на данни.

Извеждат се зависимости: какви ресурси и/или действия предпочита дадения потребител, кои са най-честите дейности извършени от потребителите и др.  Информацията се предоставя на клиента във вид удобен за визуализиране на зависимостите.

2.Теоретично описание на решението на проблема

Полученият от сървъра файл с логове е първоначално обработен, така че на всеки един елемент от файла (под елемент се има предвид информация от тип Time;Event context;Component;Event name;Description;Origin;IP address) да съответсва числово означение под формата на мапинг – за всеки елемент съответства цяло число. Нов файл е създаден само с цифровите еквиваленти на съответните елементи. Всеки ред от файла се сортира независимо от другите, така че да бъде във възходящ ред, създавайки нов файл. Новият файл се предава за обработка на SMPF алгоритъма, който в случая е NegFin алгоритъм. Алгоритъма от своя страна създава файл, който съдържа вече обработените данни, като всеки ред представлява група от елементи, последвани от техният “support”, което е колко пъти се среща тази група елемнти във всички транзакции.   
За да бъде лесно четяема и разбираема за потребителя, получаната информация е обработена отново, като този път числовите стойности се конвертират в техните съответни текстови еквиваленти отново чрез мап.  
Така накрая информацията от последният получен файл е предадена на клиента.

3.Проектиране

Създават се класове за клиент и сървър. Сървърът е многонишков за да може да работи с повече от един клиент едновременно. Сървърът слуша на порт 4444, а когато accept-не заявката на клиента, се установява комуникация.  
Първоначално се изисква клиента да въведе “minsup”, това е прага/минимума срещания на коя да е било група елементи в транзакциите. Тази информация се праща на сървъра, като тя се използва за изпълнението на алгоритъма.  
След което клиента и изпраща лог файла чрез PrintWriter стрийм.   
Сървъра получава информацията и я запазва във файл, след което информацията от файла я “кодира” чрез Map<String, Integer> чрез функцията convertText и създава нов файл само от числа.   
За сортирането на всеки един ред се ползва функцията sortFile(File file).   
Новополученият файл е вече подходящ за обработвящият алгоритъм.  
чрез функцията runAlgorithm(input, minsup, output) се подава файлът и minsup коефициента и в output се записва името на новият файл с намерените зависимости и изчислени честите срещания на дадени групи елементи.  
Този файл от своя страна се подава на convertSPMF функцията и оттам се извеждат текстовите еквиваленти на числата, така че се създава нов текстов файл, със вече ясно четими стойности и резултати от алгоритъма. Така вече чрез стрийм се предават обратно към клиента резултатите.

4. Реализация  
За алгоритъмът NegFIN е използвана готова реализация от SPMF библиотеката за data mining.

Входът на NegFIN е **база данни** за **транзакции** (и праг с име ***minsup*** (стойност между 0 и 100%). **Базата данни за транзакции** е набор от транзакции. Всяка **транзакция** е набор от елементи. **NegFIN** е алгоритъм за откриване на елементи (група от елементи), които се срещат често в база данни за транзакции ( **чести позиции** ). Честият набор от елементи е набор от елементи, който се появява поне minsup транзакции от база данни за транзакции, където minsup е параметър, зададен от потребителя.

В резултатите всеки набор от елементи е анотиран с неговият support. Support на набор от елементи е колко пъти се показва наборът от елементи в базата данни за транзакции.

**Входният файлов формат,** използван от NegFIN, се дефинира както следва. Това е текстов файл. Елементът е представен с положително цяло число.Транзакцията е ред в текстовия файл. Във всеки ред (транзакция) елементите са разделени от пауза. Предполага се, че всички елементи в една и съща транзакция (ред) са сортирани (напр. Възходящ ред) и че нито един елемент не може да се появи два пъти в рамките на същия ред.

Форматът на **изходния файл** се дефинира както следва. Това е текстов файл, където всеки ред представлява често срещан набор от елементи. Във всеки ред, елементите от набора от елементи първо се изброяват. Всеки елемент е представен с цяло число и е последвано от пауза. След като се появят всички елементи, се появява ключовата дума "#SUP:", която е последвана от цяло число, обозначаващо support-a на набора от елементи, изразено като брой транзакции.

5. Ръководство за инсталация

Нужно е Server класа и Client класа предвартиелно да бъдат компилирани както следва:

javac Sever.java

javac Client.java

Забележка: Първа трябва да сме в тяхната директория, а именно src директорията.  
  
Първо се стартира сървъра както следва  
java Server

После се отваря нов терминал, където се стартира клиента  
java Client

6.Резултати - резултати от работата на приложението  
Приложението изкарва на конзолата/терминала списък със набори от елементи с техният съпорт  
  
7.Литература:  
1. <http://www.philippe-fournier-viger.com/spmf/NegFIN.php>

8.Приложение  
**Ползвани са допълнителните класове MemoryLogger, AlgoNegFIN от библиотеката spmf**

**SERVER:**

import java.io.\*;

import java.net.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

//import network.connection.AlgoNegFIN;

public class Server {

public static final int SERVER\_PORT = 4444;

public static void main(String[] args) {

System.out.println("Server started ...");

try (ServerSocket ss = new ServerSocket(SERVER\_PORT);) {

while (true) {

Socket s = ss.accept();

ServerThread t = new ServerThread(s);

t.start();

}

} catch (IOException e) {

System.out.println("There is a problem with the server socket");

e.printStackTrace();

}

}

}

class ServerThread extends Thread {

Socket socket;

BufferedReader in;

PrintStream out;

String inputFile;

String outputFile;

public ServerThread(Socket s) throws IOException {

socket = s;

outputFile = ".//output.txt";

inputFile = ".//input.txt";

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintStream(socket.getOutputStream());

}

public Map<Integer, String> convertTEXT(String charset, String input) throws IOException {

// This map will be used to store mapping from item id (key) to words (value).

Map<Integer, String> mapItemsIDToWords = null;

mapItemsIDToWords = new HashMap<Integer, String>();

// A map that store the corresponding Item ID for each word

// An entry in the map is :

// key = a word

// value = Integer (item id)

Map<String, Integer> mapWordsToItemIDs = new HashMap<String, Integer>();

// object for writing the output file

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("Tmp.txt"), charset));

// Now we will read the input file

BufferedReader myInput = null;

try {

// Create some objects to read the file

FileInputStream fin = new FileInputStream(new File(input));

myInput = new BufferedReader(new InputStreamReader(fin, charset));

// Create a string builder to store the current sentence

StringBuilder currentSentence = new StringBuilder();

// Variable to be used to assign item ids (integers) to words

int nextItemID = 1;

boolean isFirstWordOfSentence;

// For each line in the input file

String thisLine;

myInput.readLine();

while ((thisLine = myInput.readLine()) != null) {

// if the line is too short (e.g emptylines), skip it

if (thisLine.length() < 1) {

continue;

}

isFirstWordOfSentence = true;

// split the line into words

String words[] = thisLine.split(",");

// for each word

for (int i = 2; i < words.length; i++) {

if (words[i].length() != 0) {

// Convert the word to an item

Integer item = mapWordsToItemIDs.get(words[i]);

if (item == null) {

// Give a new ID to this item

item = nextItemID++;

// Remember the ID

mapWordsToItemIDs.put(words[i], item);

if (mapItemsIDToWords != null) {

mapItemsIDToWords.put(item, words[i]);

}

}

// First we will save the word in the output file

// If it is not the first word we will add a space.

if (isFirstWordOfSentence) {

isFirstWordOfSentence = false;

} else {

currentSentence.append(" ");

}

currentSentence.append(item);

}

}

// write the current sentence to the file

writer.write(currentSentence.toString());

writer.newLine();

currentSentence.setLength(0);

}

// close output file

writer.flush();

writer.close();

} catch (Exception e) {

e.printStackTrace();

} finally {

if (myInput != null) {

myInput.close();

}

}

return mapItemsIDToWords;

}

public void sortFile(String file) {

FileInputStream fin;

try {

fin = new FileInputStream(new File(file));

BufferedReader myInput = new BufferedReader(new InputStreamReader(fin));

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("Tmp2.txt"), "UTF-8"));

String sentence;

Integer value;

List<Integer> line = new ArrayList<>();

while ((sentence = myInput.readLine()) != null) {

String split[] = sentence.split(" ");

for (String s : split) {

value = Integer.valueOf(s);

line.add(value);

}

line.sort(null);

for (Integer element : line) {

writer.write(Integer.toString(element));

if (line.indexOf(element) != line.size() - 1) {

writer.write(" ");

}

}

writer.newLine();

line.clear();

}

writer.flush();

myInput.close();

writer.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

void convertSPMF( Map<Integer, String> mapItemsIDToWords)

{

String fileInput;

String value;

BufferedReader buffer;

BufferedWriter writer;

try {

buffer = new BufferedReader(new InputStreamReader(new FileInputStream(new File("output.txt")), "UTF-8"));

writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("forUser.txt"), "UTF-8"));

while ((fileInput = buffer.readLine()) != null) {

String split[] = fileInput.split(" #SUP:");

String subsplit[]=(split[0]).split(" ");

for(String s : subsplit)

{ value = mapItemsIDToWords.get(Integer.parseInt(s));

writer.write(value+" ");

}

writer.write("SUP:");

writer.write(split[1]);

writer.newLine();

}

writer.flush();

writer.close();

buffer.close();

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

} catch (NumberFormatException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public void run() {

String msgreceive;

String minsup;

try (BufferedWriter writer = new BufferedWriter(

new OutputStreamWriter(new FileOutputStream(new File(inputFile))));) {

minsup = in.readLine();

msgreceive = in.readLine();

while (!(msgreceive = in.readLine()).equals("end")) {

writer.write(msgreceive);

writer.newLine();

}

writer.flush();

Map<Integer, String> mapItemsIDToWords = convertTEXT("UTF-8", inputFile);

sortFile("Tmp.txt");

String input = "Tmp2.txt";

double doublePrim = Double.parseDouble(minsup);

// Applying the algorithm

AlgoNegFIN algorithm = new AlgoNegFIN();

algorithm.runAlgorithm(input, doublePrim, outputFile);

algorithm.printStats();

convertSPMF(mapItemsIDToWords);

BufferedReader buffer = new BufferedReader( new InputStreamReader(new FileInputStream(new File(".//forUser.txt")), "UTF-8"));

String fileInput;

while ((fileInput = buffer.readLine()) != null) {

out.println(fileInput);

}

out.println("end");

out.flush();

out.close();

in.close();

writer.close();

buffer.close();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*public static String fileToPath(String filename) throws

UnsupportedEncodingException{ URL url =

MainTestFIN.class.getResource(filename); return

java.net.URLDecoder.decode(url.getPath(),"UTF-8"); }

\*\*/

}

Client:

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintStream;

import java.net.\*;

public class Client {

public static final int SERVER\_PORT = 4444;

;

public static void main(String[] args) {

System.out.println("Client started ...");

String host= "localhost";

File fileName = new File(".//logs\_BCS37\_20181103\_UTF-8.csv");

try {

Socket socket= new Socket(host, SERVER\_PORT);

BufferedReader rcvMsg= new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintStream sendMsg= new PrintStream(socket.getOutputStream());

BufferedReader fromFile = new BufferedReader(new InputStreamReader(new FileInputStream(fileName)));

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

String fileInput;

System.out.println("Please, enter minimal support");

fileInput = stdIn.readLine();

stdIn.close();

sendMsg.println(fileInput);

//sending the log file

while ((fileInput = fromFile.readLine())!=null) {

sendMsg.println(fileInput);

}

sendMsg.println("end");

sendMsg.flush();

//the information from the server

while (!((fileInput = rcvMsg.readLine()).equals("end"))) {

System.out.println(fileInput);

}

fromFile.close();

rcvMsg.close();

fromFile.close();

sendMsg.close();

socket.close();

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}