Data Structures Final Report

Search Engine

Group 1

Members

105204027 會計四 王思勻

105303021 會計四 王奕涵

107306021 資管二 李昱瑾

1. Introduction

(1) Topic

- Master Search - Search Engine for Graduate Schools in Taiwan
- Expected home page:

- Actual home page:

(2) Target User

- graduates or someone that have work experience, who want to study for a master's degree in Taiwan

(3) Motivation

a. Background

- Recently, more and more college students want to study for a master's degree to possess advanced knowledge in the field of their professional, improve their ability to do analysis, be able to solve complex problems, and learn how to think independently. Meanwhile, more and more jobs or internship require a master's degree or to be currently studying for it. If we want to stand out from others, we may need a master's degree.

b. Problems

- When we try to find some information about a master's degree by google search engine, for example, when searching for "資訊管理", the results show up lots of websites related to work but not graduate schools. But the information that users need are how to write the application packages, or how to prepare for the interview or tests. Users may have to spend more time checking each website to find the one they truly need.

(4) Goal

- For the above reasons, our goal is to create a search engine only for information about graduate schools or master's degrees, including website of each graduate school, articles written by people who have experience in applying to a graduate school and so on. This will help our target users have more conveniency and clarity when looking for the information of they need. Most importantly, saving time.

KEYWORD	WEIGHT	KEYWORD	WEIGHT
研究所	10	口試	3
碩士	8	考研	3
推甄	8	筆試	3
入學考試	8	出路	-3
大四	5	職涯	-5
正取	5	工作	-5

2. Classes that we create

(1) Keyword: Creating a keyword to store keywords.
(String name + int weight).

```
public class Keyword {
    public String name;
    public int weight;
    public int count;

    public Keyword(String name,int weight){
        this.name = name;
        this.weight = weight;
    }

    @Override
    public String toString(){
        return "["+name+","+weight+"]";
    }
}
```

(2) Result: Creating a result to store the sorted searching results.
(String name + String url).

```
public class Result {
    public String name;
    public String url;

    public Result(String name,String url){
        this.name = name;
        this.url = url;
    }

    @Override
    public String toString(){
        return "["+name+","+url+"]";
    }
}
```

(3) WordCounter: Counting the number of times the keywords appear on the website.

```java
import java.io.BufferedReader;

public class WordCounter {
    private String urlStr;
    private String content;

    public WordCounter(String urlStr){
        this.urlStr = urlStr;
    }

    private String fetchContent() throws IOException{
        URL url = new URL(this.urlStr);
        URLConnection conn = url.openConnection();
        conn.setRequestProperty("User-agent", "Chrome/7.0.517.44");
        InputStream in = conn.getInputStream();
        BufferedReader br = new BufferedReader(new InputStreamReader(in));

        String retVal = "";

        String line = null;

        while ((line = br.readLine()) != null){
            retVal = retVal + line + "\n";
        }

        return retVal;
    }


    public int countKeyword(String keyword) throws IOException{
        if (content == null){
            content = fetchContent();
        }

        //To do a case-insensitive search, we turn the whole content and keyword into upper-case:
        content = content.toUpperCase();
        keyword = keyword.toUpperCase();

        int retVal = 0;
        int fromIdx = 0;
        int found = -1;

        while ((found = content.indexOf(keyword, fromIdx)) != -1){
            retVal++;
            fromIdx = found + keyword.length();
        }

        return retVal;
    }
}
```

(4) WebPage: Counting the score of the website
(keyword's weight * the number WordCounter return).

```
import java.io.IOException;

public class WebPage {
    public String url;
    public String name;
    public WordCounter counter;
    public int score;

    public WebPage(String url,String name){
        this.url = url;
        this.name = name;
        this.counter = new WordCounter(url);
    }

    public void setScore(ArrayList<Keyword> keywords) throws IOException{
        score = 0;
        for(Keyword k : keywords){
            try{
            score += counter.countKeyword(k.name)* k.weight;
            }
            catch(IOException ex){
                return;
            }
        }

    }
```

(5) WebTree:

```
import java.io.IOException;


public class WebTree {
    public WebNode root;

    public WebTree(WebPage rootPage){
        this.root = new WebNode(rootPage);
    }

    public void setPostOrderScore(ArrayList<Keyword> keywords) throws IOException{
        setPostOrderScore(root, keywords);
    }

    private void setPostOrderScore(WebNode startNode, ArrayList<Keyword> keywords) throws IOException{
        // 1. compute the score of children nodes
        // 2. setNode score of startNode

        for(WebNode child : startNode.children) {
            setPostOrderScore(child, keywords);
        }
        startNode.setNodeScore(keywords);
    }

    public void eularPrintTree(){
        eularPrintTree(root);
    }

    private void eularPrintTree(WebNode startNode){
         int nodeDepth = startNode.getDepth();

         if(nodeDepth > 1) System.out.print("\n" + repeat("\t", nodeDepth-1));
         System.out.print("(");
         System.out.print(startNode.webPage.name+","+startNode.nodeScore);

         for(WebNode child : startNode.children){
             eularPrintTree(child);
         }
         System.out.print(")");
         if(startNode.isTheLastChild()) System.out.print("\n" + repeat("\t", nodeDepth-2));

    }

    private String repeat(String str,int repeat){
         String retVal  = "";
         for(int i=0;i<repeat;i++){
             retVal+=str;
         }
         return retVal;
    }
}
```

(6) WebNode:

```
import java.io.IOException;

public class WebNode {
    public WebNode parent;
    public ArrayList<WebNode> children;
    public WebPage webPage;
    public double nodeScore;

    public WebNode(WebPage webPage){
        this.webPage = webPage;
        this.children = new ArrayList<WebNode>();
    }

    public void setNodeScore(ArrayList<Keyword> keywords) throws IOException{
        //this method should be called in post-order mode

        //1. compute webPage score
        //2. set webPage score to nodeScore

        //3. nodeScore.score += all childrens nodeScore
        webPage.setScore(keywords);
        nodeScore=webPage.score;
        for(WebNode child: children) {
            nodeScore+=child.webPage.score;
        }
    }

    public void addChild(WebNode child){
        // add the WebNode to its children list
        children.add(child);
    }

     public boolean isTheLastChild(){
         if(this.parent == null) return true;
         ArrayList<WebNode> siblings = this.parent.children;

         return this.equals(siblings.get(siblings.size() - 1));
     }

     public int getDepth(){
         int retVal = 1;
         WebNode currNode = this;
         while(currNode.parent!=null){
             retVal ++;
             currNode = currNode.parent;
         }
         return retVal;
     }
}
```

(7) KeywordList: Maintaining a KeywordList to store the sorted scores of websites.

```
import java.io.IOException;

public class KeywordList {
    private Keyword[] hash_table;
    private ArrayList<Keyword> keywords = new ArrayList<>();
    private int maxSize;

    public KeywordList(int slots){
        maxSize = slots;
        this.hash_table = new Keyword[maxSize];
    }

    private int hash(int key){
        return key % maxSize;
    }


    public void add(Keyword keyword){

        int pos = hash(keyword.weight);
        hash_table[pos] = keyword;
//      System.out.println("Done");
        pos = (pos + 1) % maxSize;

    }
```

```
    public String getKeyWordByCount(int count){

        int i = hash(count);
        if(hash_table[i] != null)
        {
            return(hash_table[i].name);
        }
        else return null;
    }

    public void output(){
        StringBuilder sb = new StringBuilder();
        for(Keyword k : keywords){
            sb.append(k.toString()+" ");
        }

        System.out.println(sb.toString().trim());
    }

    public void sort(){
        this.keywords = doQuickSort(this.keywords);

    }

    private ArrayList<Keyword> doQuickSort(ArrayList<Keyword> keywords) {
        if(keywords.size() <= 1){
            return keywords;
        }

        ArrayList<Keyword> lt = new ArrayList<>();
        ArrayList<Keyword> eq = new ArrayList<>();
        ArrayList<Keyword> gt = new ArrayList<>();
        ArrayList<Keyword> retVal = new ArrayList<>();
        int IndexOfList = keywords.size() / 2;
        Keyword point = keywords.get(IndexOfList);

        for(Keyword keyword : keywords){
            if(keyword.count > point.count){
                lt.add(keyword);
            }
            else if (keyword.count < point.count){
                gt.add(keyword);
            }
            else {
                eq.add(keyword);
            }
        }
        retVal.addAll(doQuickSort(gt));
        retVal.addAll(eq);
        retVal.addAll(doQuickSort(lt));

        return retVal;
    }
}
```

(8) GoogleQuery: Connecting to the internet to get the searching results on Google, counting the scores, and put the scores in descending order.

```
import java.io.BufferedReader;


public class GoogleQuery {

    public String searchKeyword;

    public String url;

    public String content;


    public GoogleQuery(String searchKeyword){

        this.searchKeyword = searchKeyword;

        this.url = "http://www.google.com/search?q="+searchKeyword+"&oe=utf8&num=10";

    }


     private String fetchContent() throws IOException{

         String retVal = "";

         URL u = new URL(url);

         URLConnection conn = u.openConnection();

         conn.setRequestProperty("User-agent", "Chrome/7.0.517.44");

         InputStream in = conn.getInputStream();

         InputStreamReader inReader = new InputStreamReader(in,"utf-8");

         BufferedReader bufReader = new BufferedReader(inReader);
         String line = null;

         while((line=bufReader.readLine())!=null){
             //retVal += line;
             retVal = retVal + line + "\n";
         }
         return retVal;
     }
```

```
    public ArrayList<Result> query() throws IOException{

        if(content==null){

            content= fetchContent();
//          System.out.println(content);
        }

        HashMap<String, String> searchResult = new HashMap<String, String>();

        Document doc = Jsoup.parse(content);
//      System.out.println(doc.text());
        Elements lis = doc.select("div");
        lis = lis.select(".ZINbbc");
//      System.out.println(lis.size());


        for(Element li : lis)
        {
            try
            {
                Elements select = li.select("a[title]");
                if (select!=null&&select.size()>0) {
                String linkHref = select.get(0).attr("href");//獲取href值
                String linkText = select.get(0).text();//獲取text

                searchResult.put(linkHref, linkText);
                System.out.println(linkHref + linkText);}


            } catch (IndexOutOfBoundsException e)  {
            e.printStackTrace();
            }
        }

        KeywordList result_table = calculate(searchResult);

        ArrayList<Result> finalResult = new ArrayList<Result>();

        for(int i = finalResult.size() ; i > 0 ; i--) {
            if(result_table.getKeyWordByCount(i) != null) {
                String name = result_table.getKeyWordByCount(i);
                String resultUrl = searchResult.get(name);
                finalResult.add(new Result(name,resultUrl));
            }
        }
        return finalResult;
```

```
    public KeywordList calculate(HashMap<String, String> searchResult) throws IOException{

        ArrayList<Keyword> keywords = new ArrayList<Keyword>();
        keywords.add(new Keyword("台灣",5));
        keywords.add(new Keyword("研究所",10));
        keywords.add(new Keyword("碩士",8));
        keywords.add(new Keyword("推甄",8));
        keywords.add(new Keyword("甄試",8));
        keywords.add(new Keyword("入學考試",8));
        keywords.add(new Keyword("大四",5));
        keywords.add(new Keyword("正取",5));
        keywords.add(new Keyword("筆試",3));
        keywords.add(new Keyword("口試",3));
        keywords.add(new Keyword("考研",3));
        keywords.add(new Keyword("出路",-3));
        keywords.add(new Keyword("資工",-3));
        keywords.add(new Keyword("職涯",-5));
        keywords.add(new Keyword("工作",-5));

        KeywordList result_table = new KeywordList(100);

        for(String key : searchResult.keySet()){
              String value = searchResult.get(key);
              WebPage rootPage = new WebPage(value,key);

              rootPage.setScore(keywords);
//            System.out.print(rootPage.name+","+rootPage.score);
              result_table.add(new Keyword(key,rootPage.score));
        }
        return result_table;
    }
```

(9) Main: Showing the optput of our search engine.

```
import java.io.IOException;

public class Main {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);
        while(scanner.hasNextLine()){
            GoogleQuery google = new GoogleQuery(scanner.next());
            ArrayList<Result> query = google.query();

            String[][] s = new String[query.size()][2];
            int num = 0;
            for(Result entry : query) {
                String key = entry.name;
                String value = entry.url;
                s[num][0] = key;
                s[num][1] = value;
                System.out.println(key + "     " +value);
                num++;
            }
        }

        scanner.close();
    }

}
```

(10) Web: Showing the interface of our search engine.

```java
import java.io.IOException;

/**
 * Servlet implementation class Web
 */
@WebServlet("/Web")
public class Web extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Web() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExcept
        // TODO Auto-generated method stub
        response.setCharacterEncoding("UTF-8");
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html");

        if(request.getParameter("keyword")==null) {
            String requestUri = request.getRequestURI();
            request.setAttribute("requestUri",requestUri );
            request.getRequestDispatcher("Search.jsp").forward(request, response);
            return;
        }

        GoogleQuery google = new GoogleQuery(request.getParameter("keyword"));
        ArrayList<Result> query = google.query();

        String[][] s = new String[query.size()][2];
        request.setAttribute("query", s);
        int num = 0;
        for(Result entry : query) {
            String key = entry.name;
            String value = entry.url;
            s[num][0] = key;
            s[num][1] = value;
            num++;
        }
        request.getRequestDispatcher("googleitem.jsp").forward(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExcept
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

3. JSP file that we create

(1) googleitem: home page

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> MasterSearch</title>
</head>
<body bgColor="#BBFFEE">
<center>
    <form action= '${requestUri}' method='get'>
        <br><br><br><br><br><br><br>
        <H1><font color=steelblue>Master Search</font></H1>
        <H3><font color=#008888>Search Engine for Graduate Schools in Taiwan</font></H3>
        <br>
        <input type='text' name='keyword' placeholder='關鍵字'/>
        <input type='submit' value='搜尋' />
        <br>
        <br>
        <input type="button" value="Dcard" onclick="location.href='https://www.dcard.tw/f/graduate_school
        <input type="button" value="PTT" onclick="location.href='https://www.ptt.cc/bbs/graduate/index.ht


    </form>
</center>
</form>
</body>
</html>
```

(2) Search: result page

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>SearchResult</title>
</head>
<body bgColor="#BBFFEE">

<b style="font-size: 16px; color: #000088;"><%="[Suggested results]"%></b>
<br>
    <%
String[][] orderList = (String[][]) request.getAttribute("query");
for (int i=0; i<orderList.length; i++){%>
    <a href='<%= orderList[i][1] %> '><%= orderList[i][0] %></a><br><h style="font-size:10px;">
    <%=orderList[i][1] %></h><br>
<%
}
%>
</body>
</html>
```

4. Challenges that we faced

We fail to use Jsoup to collect google search results correctly so that our search output is not perfect. We will keep making our code work, hope our master search engine can help junior in the future.