Beijing University of Posts and Telecommunications

Information Retrieval & extraction course assignments

Design of search engine for animation otaku

(machine translation English version)

cour tyarsyst d em	Network Technology Research Institute	
Group members	Gao chuyang	2016111441
	Xue Yang	2016111434
	Feng Yimeng	2016010259
	Sun Chuanhao	2016110324
Time between	2016 7	23
Instructor	LEI LI	

catalogue

1、 Project introduction

In this course design, considering the basic requirements and practicability, we decided to make a course design work of otaku oriented search engine as an information retrieval and extraction system.

We quoted the famous secondary information website - mengniang encyclopedia （ https://zh.moegirl.org/Mainpage ）The experiment was carried out on the database.

图 1 Meng Niang Encyclopedia

In this experiment, we searched the content of mengniang encyclopedia website and used saber and other commonly used animation

Roles are searched as keywords, and the return results with practical value are obtained.

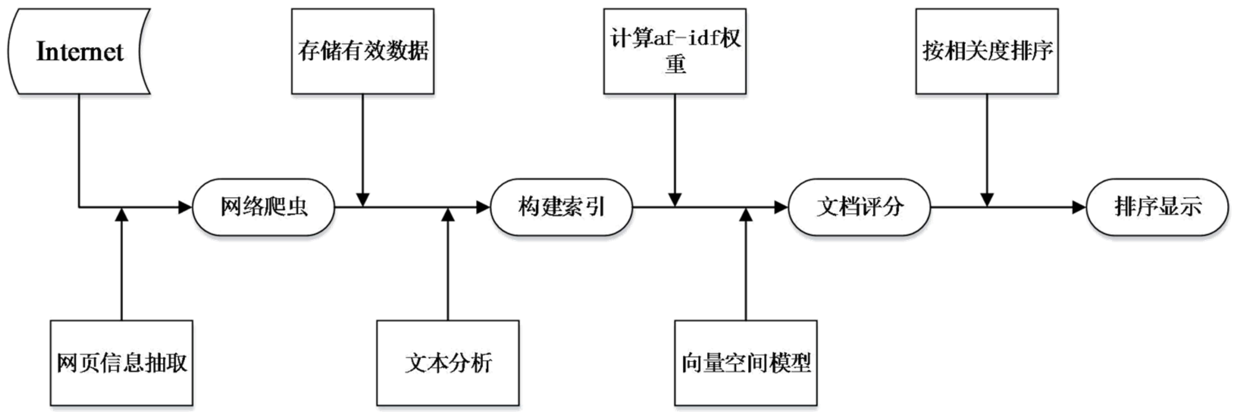
Next, we will explain in detail how we make the front-end interface, highlight it, and sort the index results in combination with our code.

We also made a simple comparison and analysis with the built-in search function on the page, and obtained some areas that need to be further improved.

2、 Overall architecture

Structure analysis: we divide the task into four parts: crawling of vocabulary interpretation entry data of mengniang encyclopedia, construction of inverted index, implementation of vector space model and front-end interface of information extraction.

It is mainly divided into four modules: web crawler, index building, document scoring and sorting display. There are some sub modules between modules, including web page information extraction, data storage, text analysis, TF IDF weight calculation, vector space model modeling, correlation ranking and so on. The following is the design structure diagram caused by the whole search:

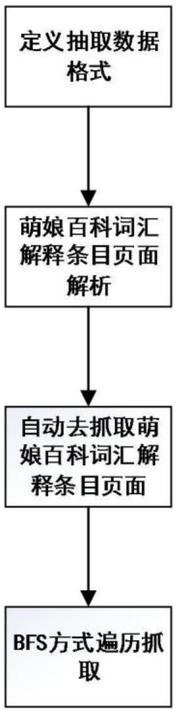


3、 Web crawler

In this job, the web crawler part uses the beautiful soup open source framework.

Beautiful soup is an HTML / XML parser written in Python. It can well handle non-standard tags and generate parsing trees. It is usually used to analyze web documents captured by crawlers. For irregular HTML documents, there are also many completion functions, saving developers' time and energy.

Flow chart of this module:

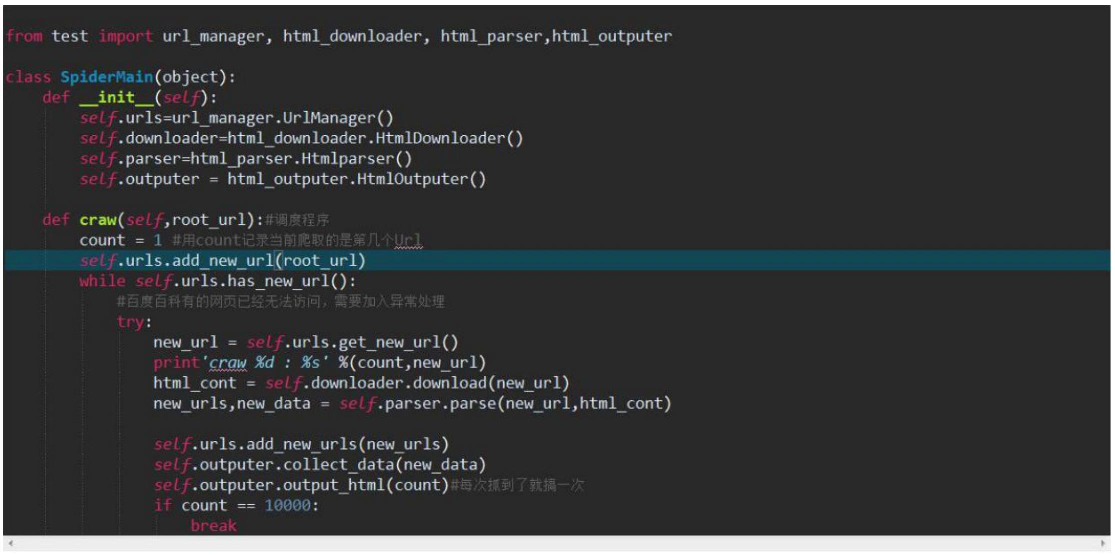

定义抽取数据格式
萌娘百科词汇解释条目页面解析
自动去抓取萌娘百科词汇解释条目页面
BFS方式遍历抓取

Specific implementation process of this module:

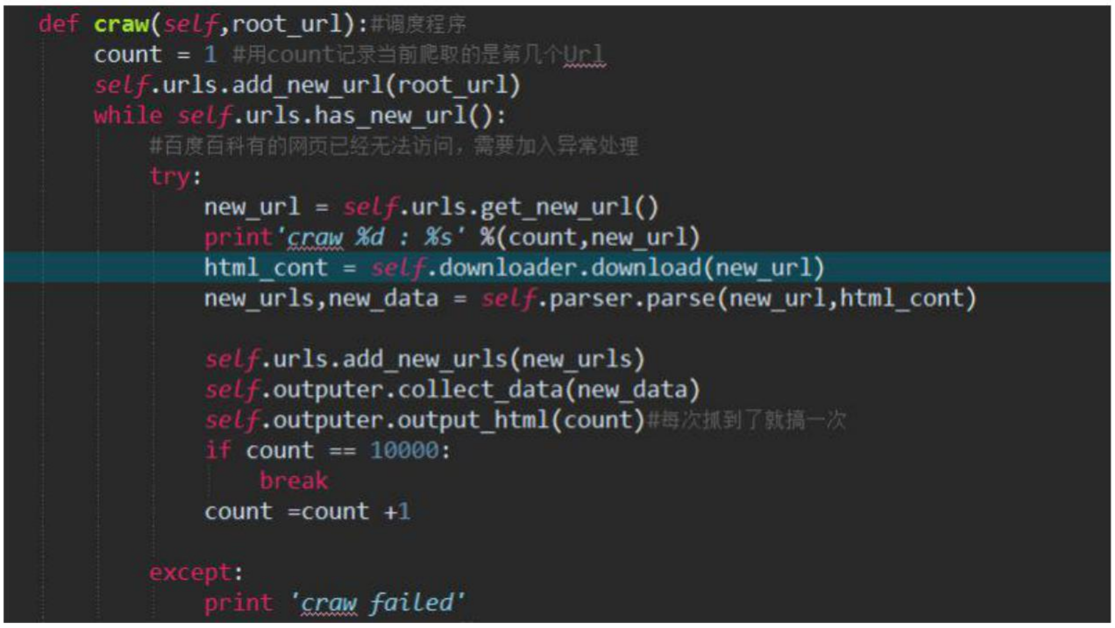
First, the source URL (https://zh.moegirl.org/Fate/stay_night) Load it into the URL manager, and the cycle starts. The judgment condition is whether there is a new URL in the URL manager. When there is a new URL, start crawling; Otherwise, end the crawler. After the crawler starts, it is necessary to use the HTML downloader to download all page contents in the source URL, and then use the HTML parser Htmlparser to parse the downloaded contents to obtain all valid URLs, body contents and titles in the body, and then load these newly obtained URLs into the URL manager again to start a new cycle. Finally, the obtained data is output by the outputter and saved as a JSON file. At the same time, we are in the process of crawling

An exception handler is also added. If an exception occurs, a "crawl failed" message will be sent to the console to continue the next cycle.

The main codes are as follows:



```
from test import url_manager, html_downloader, html_parser,html_outputer

class SpiderMain(object):
    def __init__(self):
        self.urls=url_manager.UrlManager()
        self.downloader=html_downloader.HtmlDownloader()
        self.parser=html_parser.Htmlparser()
        self.outputer = html_outputer.HtmlOutputer()

    def craw(self,root_url):#调度程序
        count = 1 #用count记录当前爬取的是第几个Url
        self.urls.add_new_url(root_url)
        while self.urls.has_new_url():
            #百度百科有的网页已经无法访问，需要加入异常处理
            try:
                new_url = self.urls.get_new_url()
                print'craw %d : %s' %(count,new_url)
                html_cont = self.downloader.download(new_url)
                new_urls,new_data = self.parser.parse(new_url,html_cont)

                self.urls.add_new_urls(new_urls)
                self.outputer.collect_data(new_data)
                self.outputer.output_html(count)#每次抓到了就搞一次
                if count == 10000:
                    break
```



```
    def craw(self,root_url):#调度程序
        count = 1 #用count记录当前爬取的是第几个Url
        self.urls.add_new_url(root_url)
        while self.urls.has_new_url():
            #百度百科有的网页已经无法访问，需要加入异常处理
            try:
                new_url = self.urls.get_new_url()
                print'craw %d : %s' %(count,new_url)
                html_cont = self.downloader.download(new_url)
                new_urls,new_data = self.parser.parse(new_url,html_cont)

                self.urls.add_new_urls(new_urls)
                self.outputer.collect_data(new_data)
                self.outputer.output_html(count)#每次抓到了就搞一次
                if count == 10000:
                    break
                count =count +1

            except:
                print 'craw failed'
```

3.1 define the format of extracted data

For the design of crawler, we first need to determine the format of extracted data. Through demand analysis, we decide to extract the following three kinds of data. Details are as follows:

Artistic: the text of news, which is used for text analysis and index construction;

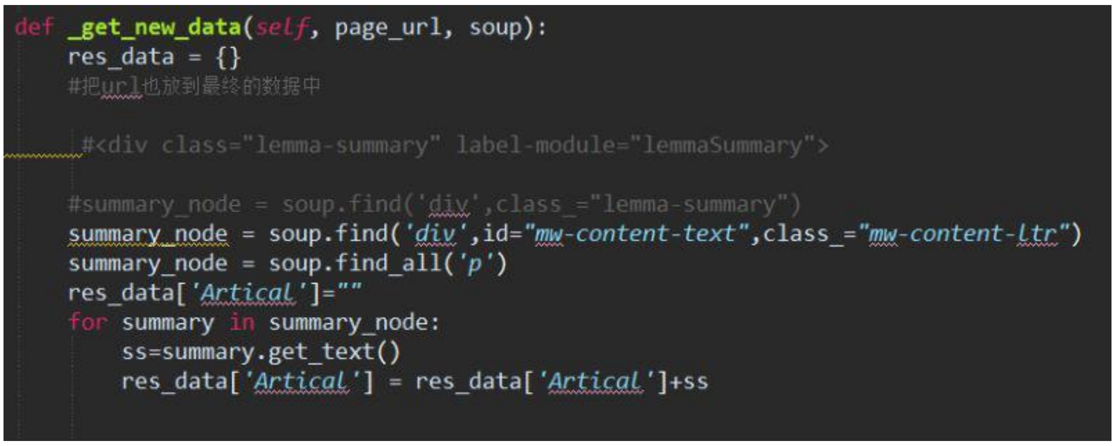
Title: the title of the news;

URL: the link of news, which is used for web display;

3.2 HTML parser

After determining the data format, next, grab a page of mengniang encyclopedia vocabulary interpretation entry, and then parse the data corresponding to the above data format. We extract page information by introducing an HTML parsing package Htmlparser in beautiful soup.

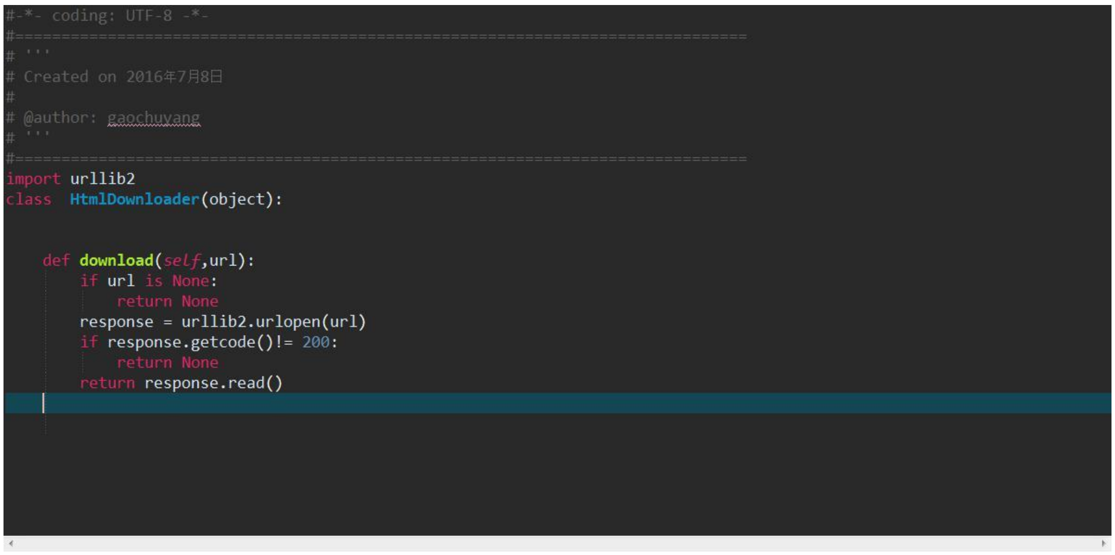
Specific implementation process: 1. Use HTML parsing package Htmlparser to parse the HTML inverted model; 2. Use the regular expression "/% +" to filter out the valid URL of the entry; 3. Use the find function in beautiful soup to extract the text and title in a specific < div >. The main codes are as follows:



```
def _get_new_data(self, page_url, soup):
    res_data = {}
    #把url也放到最终的数据中

    #<div class="lemma-summary" label-module="lemmaSummary">

    #summary_node = soup.find('div',class_="lemma-summary")
    summary_node = soup.find('div',id="mw-content-text",class_="mw-content-ltr")
    summary_node = soup.find_all('p')
    res_data['Artical']=""
    for summary in summary_node:
        ss=summary.get_text()
        res_data['Artical'] = res_data['Artical']+ss
```

3.3 HTML Downloader

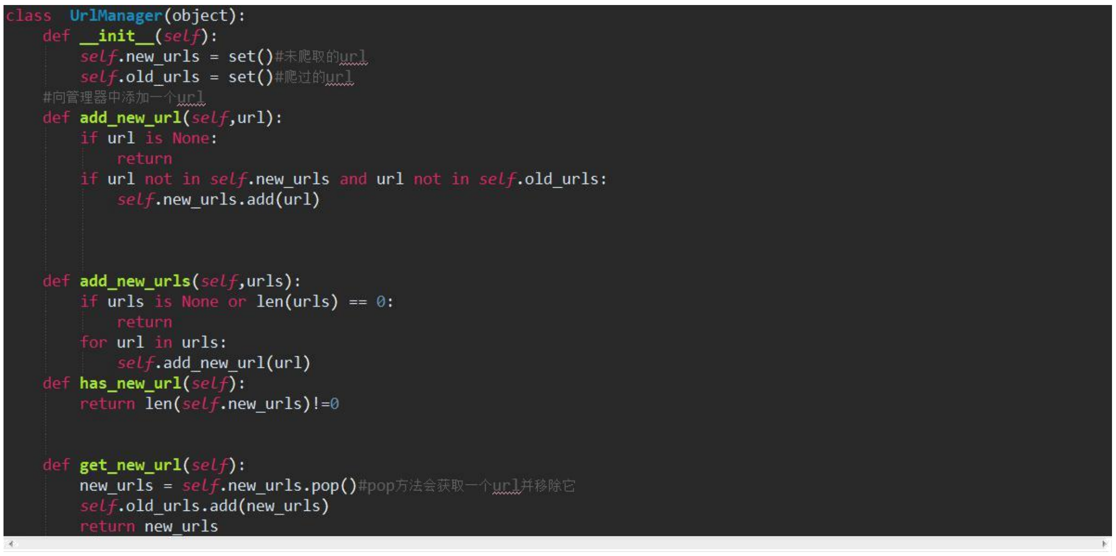
HTML downloader mainly uses urlib2 module to download documents. The specific implementation code is shown in the figure below:



```
#-*- coding: UTF-8 -*-
#==========================================================================
# '''
# Created on 2016年7月8日
#
# @author: gaochuyang
# '''
#==========================================================================
import urllib2
class  HtmlDownloader(object):


    def download(self,url):
        if url is None:
            return None
        response = urllib2.urlopen(url)
        if response.getcode()!= 200:
            return None
        return response.read()
```

3.4 URL Manager

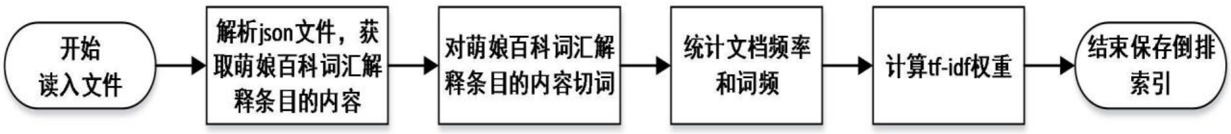
The URL manager can store all crawled URL addresses without repeated crawling. The specific implementation code is shown in the figure below:



```
class  UrlManager(object):
    def __init__(self):
        self.new_urls = set()#未爬取的url
        self.old_urls = set()#爬过的url
    #向管理器中添加一个url
    def add_new_url(self,url):
        if url is None:
            return
        if url not in self.new_urls and url not in self.old_urls:
            self.new_urls.add(url)


    def add_new_urls(self,urls):
        if urls is None or len(urls) == 0:
            return
        for url in urls:
            self.add_new_url(url)
    def has_new_url(self):
        return len(self.new_urls)!=0


    def get_new_url(self):
        new_urls = self.new_urls.pop()#pop方法会获取一个url并移除它
        self.old_urls.add(new_urls)
        return new_urls
```

4、 Construction of inverted index

Inverted index is an important part of information retrieval. This module mainly includes four key steps: extracting news content from JSON file, cutting news content into words, counting word frequency and document frequency, and calculating TF IDF weight.


开始
读入文件
解析json文件，获取萌娘百科词汇解释条目的内容
对萌娘百科词汇解释条目的内容切词
统计文档频率和词频
计算tf-idf权重
结束保存倒排索引

4.1 document segmentation

Among the common Chinese word segmentation tools, ikanalyzer is commonly used, with good effect and convenient expansion, which has been highly praised. Therefore, in our scheme, ikanalyzer word splitter is selected. Ikanalyzer is an open source, lightweight Chinese word segmentation toolkit based on Java language. Since the launch of version 1.0 in December 2006, ikanalyzer has launched four major versions.

Example of word segmentation effect: ikanalyzer 2012 supports fine-grained segmentation and intelligent segmentation. The following are demonstration examples of two segmentation methods.

Text original 1: ikanalyzer is an open source, lightweight Chinese word segmentation toolkit developed based on Java language. Since the launch of version 1.0 in December 2006, ikanalyzer has launched three major versions.

Intelligent word segmentation results: ikanalyzer is an open-source, lightweight, Chinese word segmentation toolkit based on Java language. It has launched version 1.0 since December 2006. Ikanalyzer has launched three major versions

Result of the most fine-grained word segmentation result: ikanalyzer result: the result of the fine-grained word segmentation result: the result of the ikanalyzer is the one that is the one that is the one that is the one that is the one that is the one that is the one that is the one that is the one that is the one that is the one that is the one that is the result of the most fine-grained word segmentation of the Chinese word segmentation.

Installation and deployment method: its installation and deployment is very simple. Deploy ikanalyzer2012.jar in the Lib directory of the project; Ikanalyzer.cfg.xml and stopword.dic files can be placed in the class root directory (for web projects, it is usually the WEB-INF / classes directory, which is the same as hibernate, log4j and other configuration files).

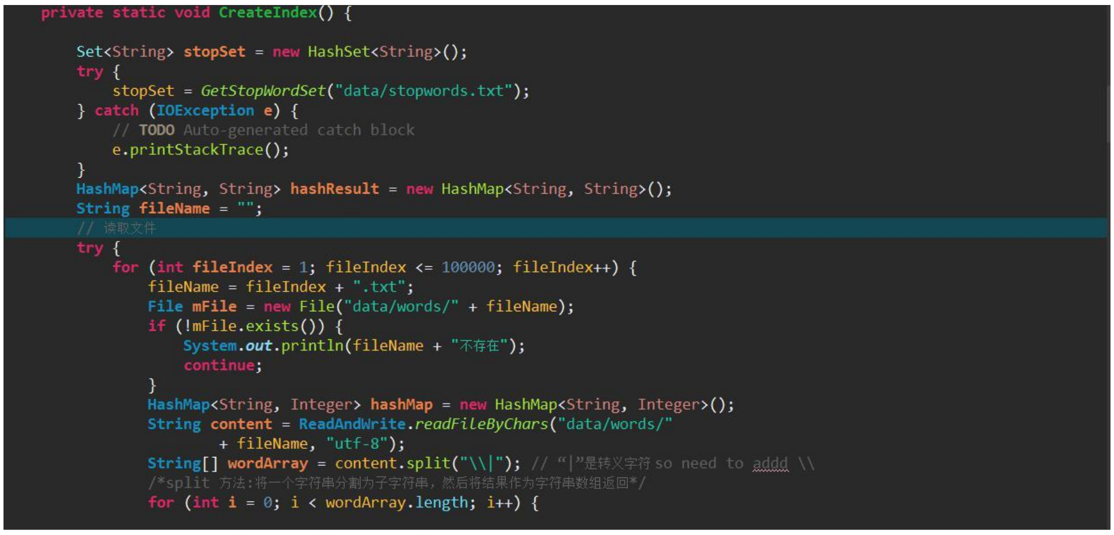
Main process: traverse all the TXT documents of the news subject extracted in the previous step, and save the word segmentation results in the local.

4.2 inverted index construction process

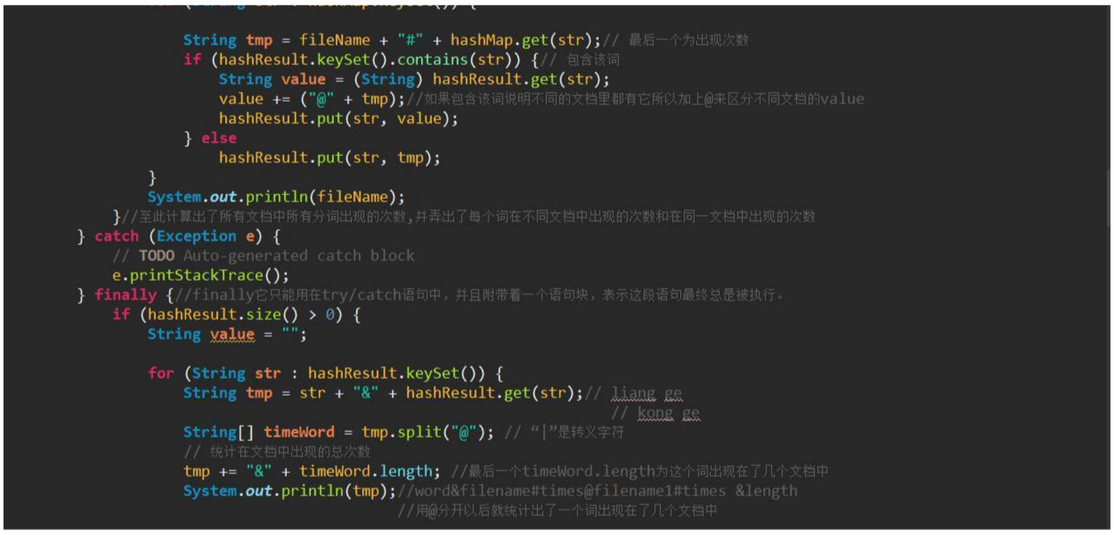
In this topic, we crawled 4287 documents, and the amount of data is not too large. Therefore, the memory based index construction method is adopted to put all the index construction process into memory for statistics, single scan, single statistical results and build the index.

Main process: traverse all 4287 documents of word segmentation in the previous step. In the loop, we count the position and number of occurrences of each word item in the document and the total number of occurrences in 4287 documents. In the specific implementation, we use two hashmaps to save the results. The HashMap uses word items as keys, and the document name, document location and other attributes are values (separated by special symbols in the middle). One is used to count the situation of each document and the other is used to count the situation of all documents.

The key codes are as follows:



```java
    private static void CreateIndex() {

        Set<String> stopSet = new HashSet<String>();
        try {
            stopSet = GetStopWordSet("data/stopwords.txt");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        HashMap<String, String> hashResult = new HashMap<String, String>();
        String fileName = "";
        // 读取文件
        try {
            for (int fileIndex = 1; fileIndex <= 100000; fileIndex++) {
                fileName = fileIndex + ".txt";
                File mFile = new File("data/words/" + fileName);
                if (!mFile.exists()) {
                    System.out.println(fileName + "不存在");
                    continue;
                }
                HashMap<String, Integer> hashMap = new HashMap<String, Integer>();
                String content = ReadAndWrite.readFileByChars("data/words/"
                        + fileName, "utf-8");
                String[] wordArray = content.split("\\|"); // “|”是转义字符 so need to addd \\
                /*split 方法:将一个字符串分割为子字符串，然后将结果作为字符串数组返回*/
                for (int i = 0; i < wordArray.length; i++) {
```



```java

                    String tmp = fileName + "#" + hashMap.get(str);// 最后一个为出现次数
                    if (hashResult.keySet().contains(str)) {// 包含该词
                        String value = (String) hashResult.get(str);
                        value += ("@" + tmp);//如果包含该词说明不同的文档里都有它所以加上@来区分不同文档的value
                        hashResult.put(str, value);
                    } else
                        hashResult.put(str, tmp);
                }
                System.out.println(fileName);
            }//至此计算出了所有文档中所有分词出现的次数,并弄出了每个词在不同文档中出现的次数和在同一文档中出现的次数
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {//finally它只能用在try/catch语句中，并且附带着一个语句块，表示这段语句最终总是被执行。
            if (hashResult.size() > 0) {
                String value = "";

                for (String str : hashResult.keySet()) {
                    String tmp = str + "&" + hashResult.get(str);// liang ge
                                                                 // kong ge
                    String[] timeWord = tmp.split("@"); // “|”是转义字符
                    // 统计在文档中出现的总次数
                    tmp += "&" + timeWord.length; //最后一个timeWord.length为这个词出现在了几个文档中
                    System.out.println(tmp);//word&filename#times@filename1#times &length
                                            //用@分开以后就统计出了一个词出现在了几个文档中
```

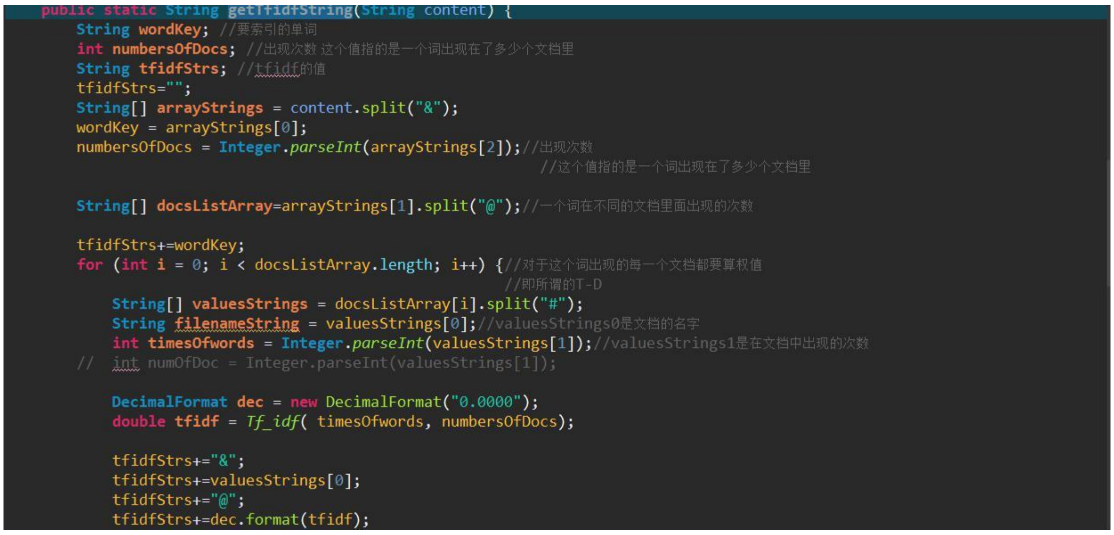
4.3 calculate TD IDF weight

The most famous weight calculation method in the field of information retrieval, TF IDF weight calculation formula:

$$w_{t,d} = (1 + \log tf_{t,d}) \times \log \frac{N}{df_t}$$

Where DFT is the number of documents with word item T, which is called inverse

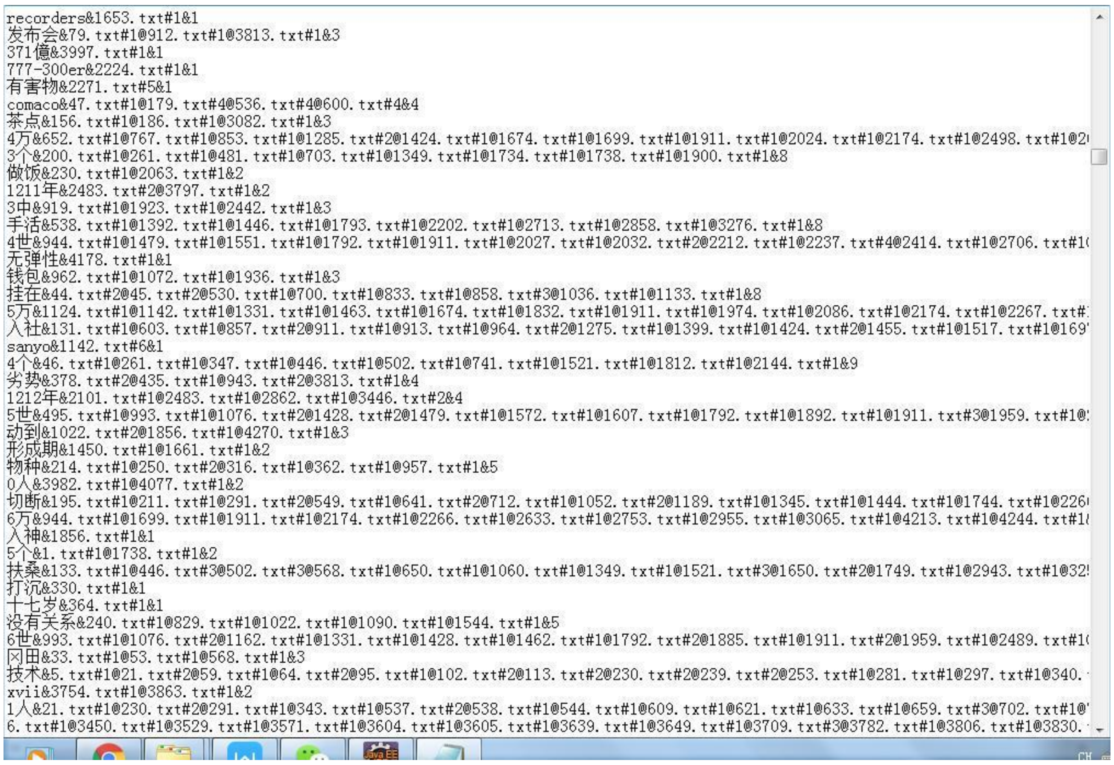
document frequency, TFT, D Is the number of times t occurs in D,
Is a quantity related to the document, which is called word item frequency. It increases with the increase of word frequency and the increase of word rarity.



```java
    public static String getTfidfString(String content) {
        String wordKey; //要索引的单词
        int numbersOfDocs; //出现次数 这个值指的是一个词出现在了多少个文档里
        String tfidfStrs; //tfidf的值
        tfidfStrs="";
        String[] arrayStrings = content.split("&");
        wordKey = arrayStrings[0];
        numbersOfDocs = Integer.parseInt(arrayStrings[2]);//出现次数
                                                          //这个值指的是一个词出现在了多少个文档里

        String[] docsListArray=arrayStrings[1].split("@");//一个词在不同的文档里面出现的次数

        tfidfStrs+=wordKey;
        for (int i = 0; i < docsListArray.length; i++) {//对于这个词出现的每一个文档都要算权值
                                                        //即所谓的T-D
            String[] valuesStrings = docsListArray[i].split("#");
            String filenameString = valuesStrings[0];//valuesStrings0是文档的名字
            int timesOfwords = Integer.parseInt(valuesStrings[1]);//valuesStrings1是在文档中出现的次数
        //  int numOfDoc = Integer.parseInt(valuesStrings[1]);

            DecimalFormat dec = new DecimalFormat("0.0000");
            double tfidf = Tf_idf( timesOfwords, numbersOfDocs);

            tfidfStrs+="&";
            tfidfStrs+=valuesStrings[0];
            tfidfStrs+="@";
            tfidfStrs+=dec.format(tfidf);
```

4.4 index file format

Each record of the index follows the following rules: wordkey & document name @ TFIDF value #next#& document name @ TFIDF finally forms an inverted index file, as shown in the following figure:



```
recorders&1653.txt#1&1
发布会&79.txt#1@912.txt#1@3813.txt#1&3
371億&3997.txt#1&1
777-300er&2224.txt#1&1
有害物&2271.txt#5&1
comaco&47.txt#1@179.txt#4@536.txt#4@600.txt#4&4
茶点&156.txt#1@186.txt#1@3082.txt#1&3
4万&652.txt#1@767.txt#1@853.txt#1@1285.txt#2@1424.txt#1@1674.txt#1@1699.txt#1@1911.txt#1@2024.txt#1@2174.txt#1@2498.txt#1@2
3个&200.txt#1@261.txt#1@481.txt#1@703.txt#1@1349.txt#1@1734.txt#1@1738.txt#1@1900.txt#1&8
做饭&230.txt#1@2063.txt#1&2
1211年&2483.txt#2@3797.txt#1&2
3中&919.txt#1@1923.txt#1@2442.txt#1&3
手活&538.txt#1@1392.txt#1@1446.txt#1@1793.txt#1@2202.txt#1@2713.txt#1@2858.txt#1@3276.txt#1&8
4世&944.txt#1@1479.txt#1@1551.txt#1@1792.txt#1@1911.txt#1@2027.txt#1@2032.txt#2@2212.txt#1@2237.txt#4@2414.txt#1@2706.txt#1
无弹性&4178.txt#1&1
钱包&962.txt#1@1072.txt#1@1936.txt#1&3
挂在&44.txt#2@45.txt#2@530.txt#1@700.txt#1@833.txt#1@858.txt#3@1036.txt#1@1133.txt#1&8
5万&1124.txt#1@1142.txt#1@1331.txt#1@1463.txt#1@1674.txt#1@1832.txt#1@1911.txt#1@1974.txt#1@2086.txt#1@2174.txt#1@2267.txt#
入社&131.txt#1@603.txt#1@857.txt#2@911.txt#1@913.txt#1@964.txt#2@1275.txt#1@1399.txt#1@1424.txt#2@1455.txt#1@1517.txt#1@169
sanyo&1142.txt#6&1
4个&46.txt#1@261.txt#1@347.txt#1@446.txt#1@502.txt#1@741.txt#1@1521.txt#1@1812.txt#1@2144.txt#1&9
劣势&378.txt#2@435.txt#1@943.txt#2@3813.txt#1&4
1212年&2101.txt#1@2483.txt#1@2862.txt#1@3446.txt#2&4
5世&495.txt#1@993.txt#1@1076.txt#2@1428.txt#2@1479.txt#1@1572.txt#1@1607.txt#1@1792.txt#1@1892.txt#1@1911.txt#3@1959.txt#1@
动到&1022.txt#2@1856.txt#1@4270.txt#1&3
形成期&1450.txt#1@1661.txt#1&2
物种&214.txt#1@250.txt#2@316.txt#1@362.txt#1@957.txt#1&5
0人&3982.txt#1@4077.txt#1&2
切断&195.txt#1@211.txt#1@291.txt#2@549.txt#1@641.txt#2@712.txt#1@1052.txt#2@1189.txt#1@1345.txt#1@1444.txt#1@1744.txt#1@226
6万&944.txt#1@1699.txt#1@1911.txt#1@2174.txt#1@2266.txt#1@2633.txt#1@2753.txt#1@2955.txt#1@3065.txt#1@4213.txt#1@4244.txt#1
入神&1856.txt#1&1
5个&1.txt#1@1738.txt#1&2
扶桑&133.txt#1@446.txt#3@502.txt#3@568.txt#1@650.txt#1@1060.txt#1@1349.txt#1@1521.txt#3@1650.txt#2@1749.txt#1@2943.txt#1@32
打沉&330.txt#1&1
十七岁&364.txt#1&1
没有关系&240.txt#1@829.txt#1@1022.txt#1@1090.txt#1@1544.txt#1&5
6世&993.txt#1@1076.txt#2@1162.txt#1@1331.txt#1@1428.txt#1@1462.txt#1@1792.txt#2@1885.txt#1@1911.txt#2@1959.txt#1@2489.txt#1
冈田&33.txt#1@53.txt#1@568.txt#1&3
技术&5.txt#1@21.txt#2@59.txt#1@64.txt#2@95.txt#1@102.txt#2@113.txt#2@230.txt#2@239.txt#2@253.txt#1@281.txt#1@297.txt#1@340.
xvii&3754.txt#1@3863.txt#1&2
1人&21.txt#1@230.txt#2@291.txt#1@343.txt#1@537.txt#2@538.txt#1@544.txt#1@609.txt#1@621.txt#1@633.txt#1@659.txt#3@702.txt#1@
6.txt#1@3450.txt#1@3529.txt#1@3571.txt#1@3604.txt#1@3605.txt#1@3639.txt#1@3649.txt#1@3709.txt#3@3782.txt#1@3806.txt#1@3830.
```

5、 Vector space modeling

5.1. Concept introduction

Vector space model (VSM) was proposed by Salton et al. In the 1970s, and

It is successfully applied to the famous smart text retrieval system. The processing of text content is simplified to vector operation in vector space, and it expresses semantic similarity by spatial similarity, which is intuitive and easy to understand.

The concept of VSM is simple, which simplifies the processing of text content into vector operation in vector space, and it expresses semantic similarity by spatial similarity, which is intuitive and easy to understand. When a document is represented as a vector in document space, the similarity between documents can be measured by calculating the similarity between vectors. The most commonly used similarity measure in text processing is cosine distance.

M An unordered feature item Ti, root / word / phrase / each other document DJ can be represented by a feature item vector

(A1j, A2j,..., AMJ) weight calculation, N training documents am * n = (AIJ) document similarity comparison 1) cosine calculation, the advantage of cosine calculation is that it is exactly a number between 0 and 1. If the vectors are consistent, it is 1, and if orthogonal, it is 1

是0, which conforms to the characteristic of similarity percentage. The calculation method of cosine is the product of vector inner product / module of each vector. 2)

Inner product calculation, direct calculation of inner product, low calculation intensity, but large error.

Vector space model is an algebraic model applied to information filtering, information retrieval, indexing and correlation evaluation. A document (Corpus) is regarded as a multiple meta vector space formed by index words (keywords). The set of index words is usually a phrase that appears at least once in the document. When searching, the input search term is also converted into a vector similar to a file

The model assumes that the correlation between the document and the search word can be known by comparing the angle deviation between each document (vector) and the search word (vector). A cosine of zero indicates that the search word vector is perpendicular to the document vector, that is, it does not match, that is, the document does not contain this search word.

Through the above vector space model, text data is transformed into structured data that can be processed by computer, and the similarity problem between two documents is transformed into the similarity problem between two vectors.

5.2 problem description

5.2.1 system tasks

The task of the text classification system is to automatically determine the category associated with the text according to the content of the text under a given classification system. From a mathematical point of view, text classification is a mapping process, which maps the text without category to the existing category. The mapping can be one-to-one mapping or one to many mapping. Because usually a text can be the same Multiple categories are associated and expressed by mathematical formula as follows: F: a - > b, where. A is the text set to be classified. B is the category set in the classification system.

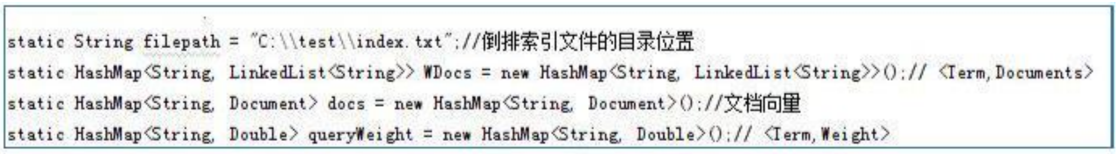
The mapping rules of text classification are the discriminant formulas and rules established by the system according to the data information of several samples of each type, and then determine the text related categories when encountering new text.

5.2 evaluation method

Because text classification is fundamentally a mapping process, the marks of evaluating text classification system are the accuracy of mapping and the speed of mapping. The speed of mapping depends on the complexity of mapping rules, and the reference for evaluating the accuracy of mapping

It is the classification result of the text after expert thinking and judgment (here, it is assumed that the manual classification is completely correct and personal thinking is excluded)

The closer the results are to those of manual classification, the higher the accuracy of classification is.

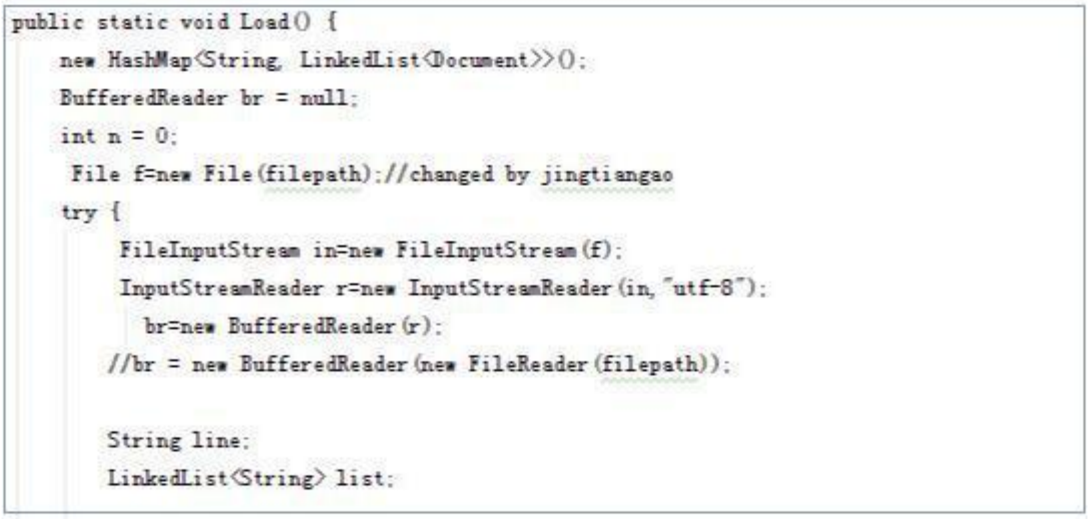


```
static String filepath = "C:\\test\\index.txt";//倒排索引文件的目录位置
static HashMap<String, LinkedList<String>> WDocs = new HashMap<String, LinkedList<String>>();// <Term,Documents>
static HashMap<String, Document> docs = new HashMap<String, Document>();//文档向里
static HashMap<String, Double> queryWeight = new HashMap<String, Double>();// <Term,Weight>
```

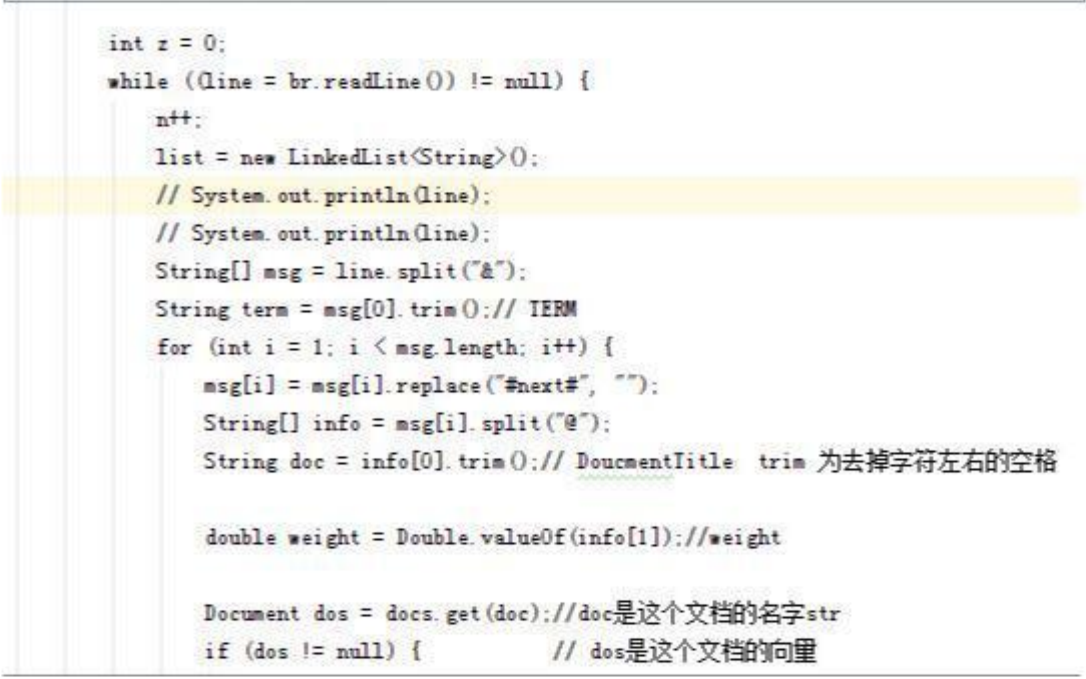
Define filepath of string type to point to the directory location of inverted index file, define wdocs of HashMap type as some attributes in the file, and define docs of HashMap type as document vector

Define docs of HashMap type as weight value

1. load( )

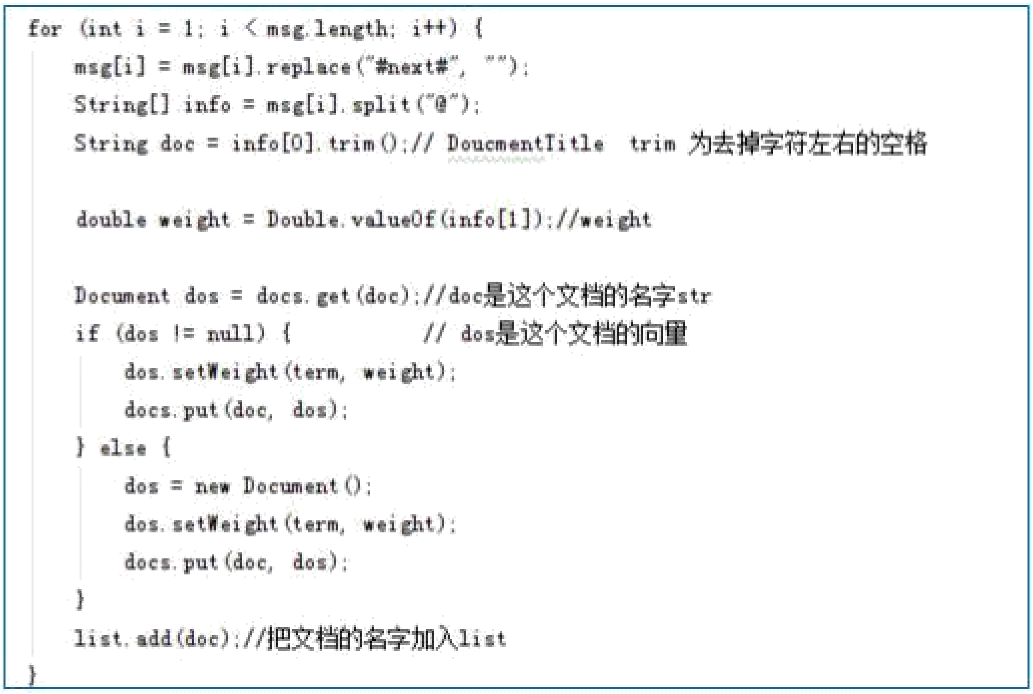


```
public static void Load() {
    new HashMap<String, LinkedList<Document>>();
    BufferedReader br = null;
    int n = 0;
     File f=new File(filepath);//changed by jingtiangao
    try {
         FileInputStream in=new FileInputStream(f);
         InputStreamReader r=new InputStreamReader(in,"utf-8");
           br=new BufferedReader(r);
       //br = new BufferedReader(new FileReader(filepath));

       String line;
       LinkedList<String> list;
```



```
       int z = 0;
       while ((line = br.readLine()) != null) {
           n++;
           list = new LinkedList<String>();
           // System.out.println(line);
           // System.out.println(line);
           String[] msg = line.split("&");
           String term = msg[0].trim();// TERM
           for (int i = 1; i < msg.length; i++) {
               msg[i] = msg[i].replace("#next#", "");
               String[] info = msg[i].split("@");
               String doc = info[0].trim();// DoucmentTitle  trim 为去掉字符左右的空格

               double weight = Double.valueOf(info[1]);//weight

               Document dos = docs.get(doc);//doc是这个文档的名字str
               if (dos != null) {           // dos是这个文档的向里
```

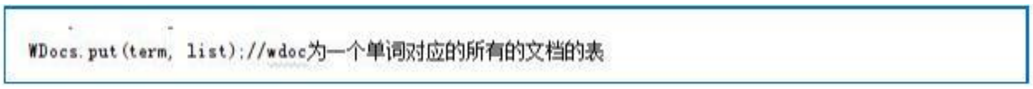
The main function of load() is word segmentation. First read in the file.



```
for (int i = 1; i < msg.length; i++) {
    msg[i] = msg[i].replace("#next#", "");
    String[] info = msg[i].split("@");
    String doc = info[0].trim();// DoucmentTitle  trim 为去掉字符左右的空格

    double weight = Double.valueOf(info[1]);//weight

    Document dos = docs.get(doc);//doc是这个文档的名字str
    if (dos != null) {            // dos是这个文档的向量
        dos.setWeight(term, weight);
        docs.put(doc, dos);
    } else {
        dos = new Document();
        dos.setWeight(term, weight);
        docs.put(doc, dos);
    }
    list.add(doc);//把文档的名字加入list
}
```

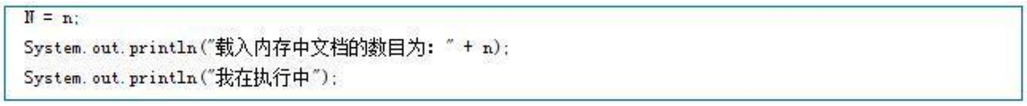
In this loop body, remove the spaces around the characters.

It is defined that DOS is the document vector and DOS is the document name str. as long as there is a new document vector, set the vector weight and add the document name to the list.



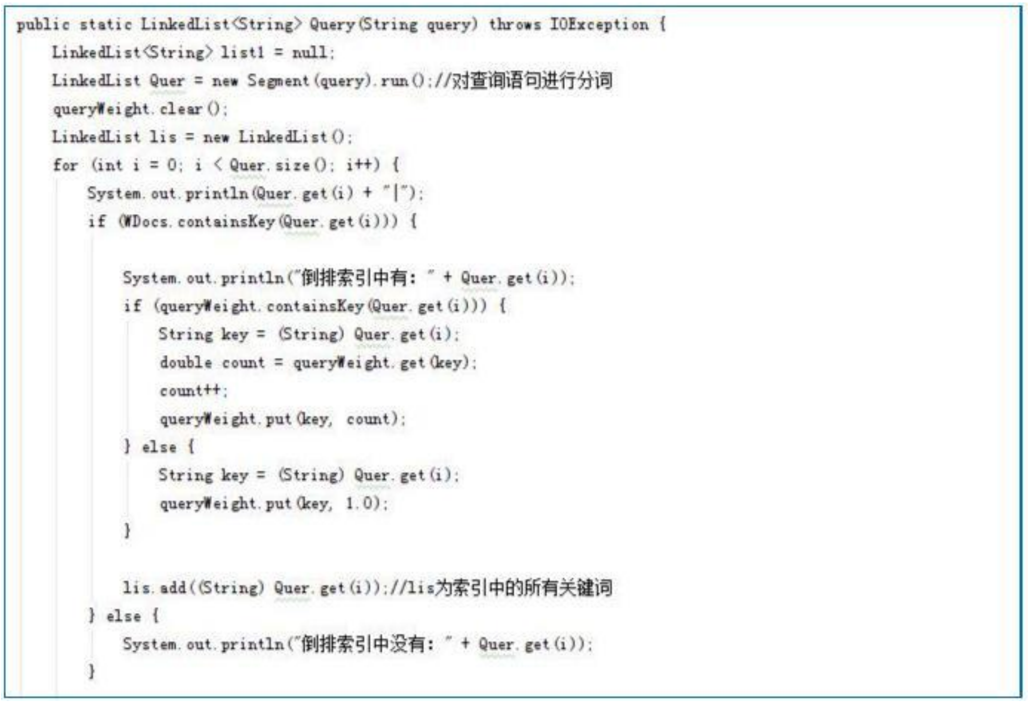
```
WDocs.put(term, list);//wdoc为一个单词对应的所有的文档的表
```

Wdoc is a table of all documents corresponding to a word.



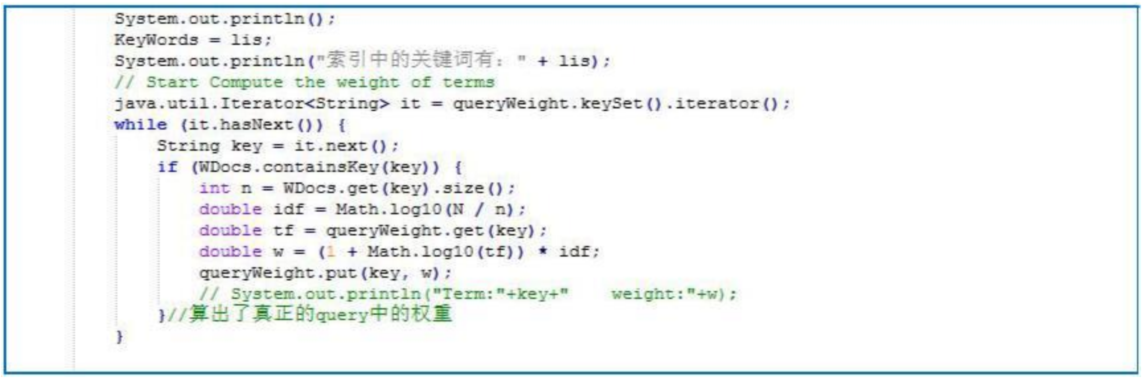
```
N = n;
System.out.println("载入内存中文档的数目为: " + n);
System.out.println("我在执行中");
```

2. query( )



```java
public static LinkedList<String> Query(String query) throws IOException {
    LinkedList<String> list1 = null;
    LinkedList Quer = new Segment(query).run();//对查询语句进行分词
    queryWeight.clear();
    LinkedList lis = new LinkedList();
    for (int i = 0; i < Quer.size(); i++) {
        System.out.println(Quer.get(i) + "|");
        if (WDocs.containsKey(Quer.get(i))) {

            System.out.println("倒排索引中有: " + Quer.get(i));
            if (queryWeight.containsKey(Quer.get(i))) {
                String key = (String) Quer.get(i);
                double count = queryWeight.get(key);
                count++;
                queryWeight.put(key, count);
            } else {
                String key = (String) Quer.get(i);
                queryWeight.put(key, 1.0);
            }

            lis.add((String) Quer.get(i));//lis为索引中的所有关键词
        } else {
            System.out.println("倒排索引中没有: " + Quer.get(i));
        }
```

Firstly, word segmentation is carried out to establish a new query index in the hash table.



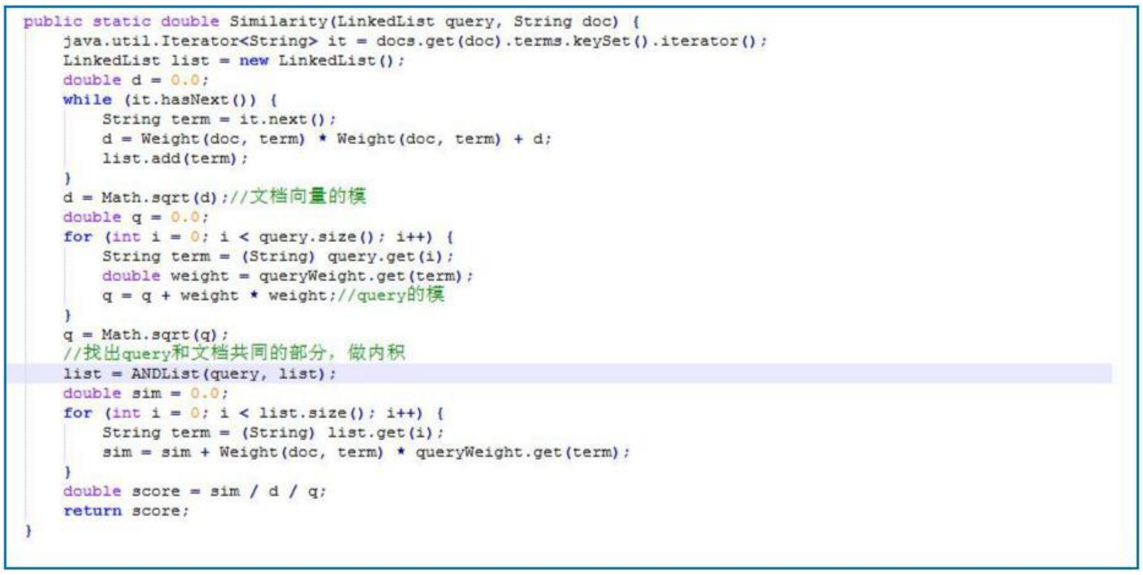
```java
        System.out.println();
        KeyWords = lis;
        System.out.println("索引中的关键词有: " + lis);
        // Start Compute the weight of terms
        java.util.Iterator<String> it = queryWeight.keySet().iterator();
        while (it.hasNext()) {
            String key = it.next();
            if (WDocs.containsKey(key)) {
                int n = WDocs.get(key).size();
                double idf = Math.log10(N / n);
                double tf = queryWeight.get(key);
                double w = (1 + Math.log10(tf)) * idf;
                queryWeight.put(key, w);
                // System.out.println("Term:"+key+"    weight:"+w);
            }//算出了真正的query中的权重
        }
```

Computer does not have human intelligence. After reading an article, people can have a fuzzy understanding of the content of the article according to their own understanding ability; while the computer can not easily "read" the article. Fundamentally, it only knows 0 and 1, so it must convert the text into a format that can be recognized by the computer. According to the "Bayesian hypothesis" , it is assumed that the words or words constituting the text are independent of each other in determining the text category. In this way, the set of words or words appearing in the text can be used to replace the text. It is self-evident that this will lose a lot of information about the content of the article, but this assumption can formalize the representation and processing of the text, and can achieve better results in text classification Previously, in the direction of information processing, the text is mainly represented by vector space model (VSM). The basic idea of vector space model is to represent the text by vector: (W1, W2,..., WN) , where porridge is the weight of the ith feature item. So what can be selected as the feature item? Generally, words, words or phrases can be selected. According to the experimental results, it is generally believed that selecting words as the feature item is better than words and phrases. Therefore, in order to represent the text as a vector in the vector space, we must first segment the text by taking these words as the dimension of the vector Represents the text. The original vector representation is completely in the form of 0, 1, that is, if the word appears in the text, the text vector

Is 1. Otherwise, it is 0.

Document vectorization method: expressed by TF-IDF weight.

(3)Similarity



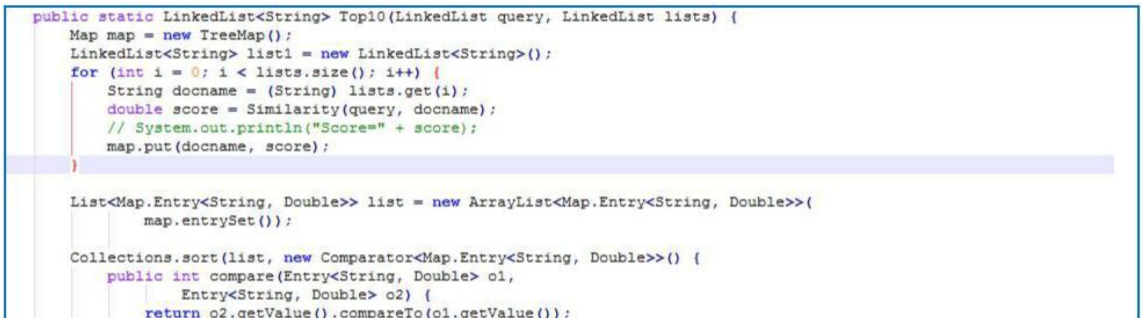
```
public static double Similarity(LinkedList query, String doc) {
    java.util.Iterator<String> it = docs.get(doc).terms.keySet().iterator();
    LinkedList list = new LinkedList();
    double d = 0.0;
    while (it.hasNext()) {
        String term = it.next();
        d = Weight(doc, term) * Weight(doc, term) + d;
        list.add(term);
    }
    d = Math.sqrt(d);//文档向量的模
    double q = 0.0;
    for (int i = 0; i < query.size(); i++) {
        String term = (String) query.get(i);
        double weight = queryWeight.get(term);
        q = q + weight * weight;//query的模
    }
    q = Math.sqrt(q);
    //找出query和文档共同的部分，做内积
    list = ANDList(query, list);
    double sim = 0.0;
    for (int i = 0; i < list.size(); i++) {
        String term = (String) list.get(i);
        sim = sim + Weight(doc, term) * queryWeight.get(term);
    }
    double score = sim / d / q;
    return score;
}
```

Calculate the similarity and add the weight of each file to the hash table.

Sorting method: sort according to the similarity calculation results according to the score. At the same time, it can be sorted according to time and time

Heat is simply sorted.

(4) LinkedList



```
public static LinkedList<String> Top10(LinkedList query, LinkedList lists) {
    Map map = new TreeMap();
    LinkedList<String> list1 = new LinkedList<String>();
    for (int i = 0; i < lists.size(); i++) {
        String docname = (String) lists.get(i);
        double score = Similarity(query, docname);
        // System.out.println("Score=" + score);
        map.put(docname, score);
    }

    List<Map.Entry<String, Double>> list = new ArrayList<Map.Entry<String, Double>>(
            map.entrySet());

    Collections.sort(list, new Comparator<Map.Entry<String, Double>>() {
        public int compare(Entry<String, Double> o1,
                Entry<String, Double> o2) {
            return o2.getValue().compareTo(o1.getValue());
```

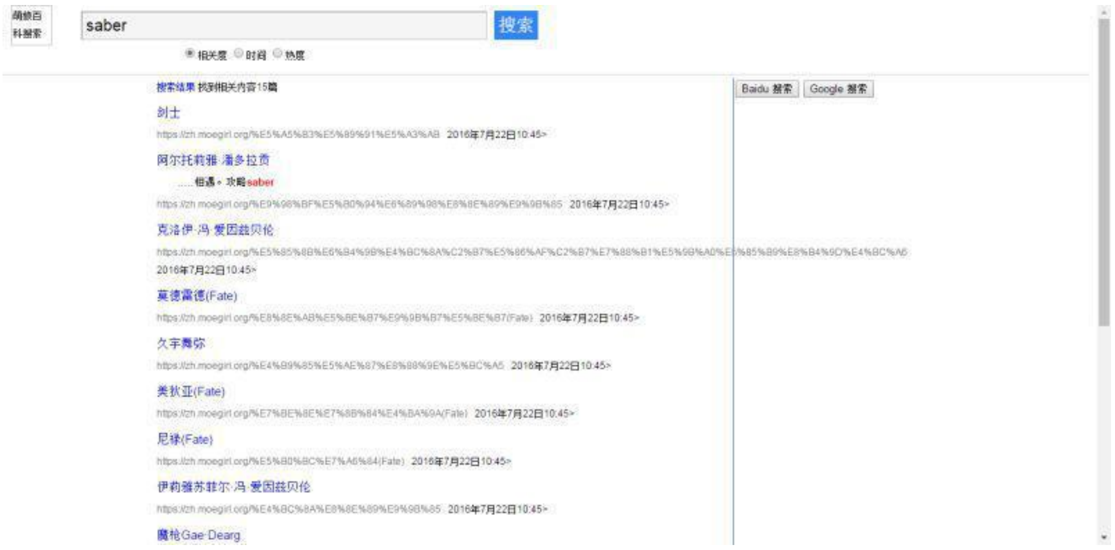
The final sorting result LinkedList is returned in the form of hash table.

6、 Search engine front end

6.1 effect display

This experiment searched the well-known animation consulting website mengniang encyclopedia. The reason why we experimented with this website is to design a tool that can help the otaku quickly obtain first-hand fresh consultation.

Here are the search results we returned:



搜索结果 找到相关内容15篇

剑士
https://zh.moegirl.org/%E5%A5%B3%E5%89%91%E5%A3%AB 2016年7月22日10:45>

阿尔托莉雅·潘多拉贡
……相遇。攻略saber
https://zh.moegirl.org/%E9%98%BF%E5%B0%94%E6%89%98%E8%8E%89%E9%9B%85 2016年7月22日10:45>

克洛伊·冯·爱因兹贝伦
https://zh.moegirl.org/%E5%85%8B%E6%B4%9B%E4%BC%8A%C2%B7%E5%86%AF%C2%B7%E7%88%B1%E5%9B%A0%E5%85%B9%E8%B4%9D%E4%BC%A6
2016年7月22日10:45>

莫德雷德(Fate)
https://zh.moegirl.org/%E8%8E%AB%E5%BE%B7%E9%9B%B7%E5%BE%B7(Fate) 2016年7月22日10:45>

久宇舞弥
https://zh.moegirl.org/%E4%B9%85%E5%AE%87%E8%88%9E%E5%BC%A5 2016年7月22日10:45>

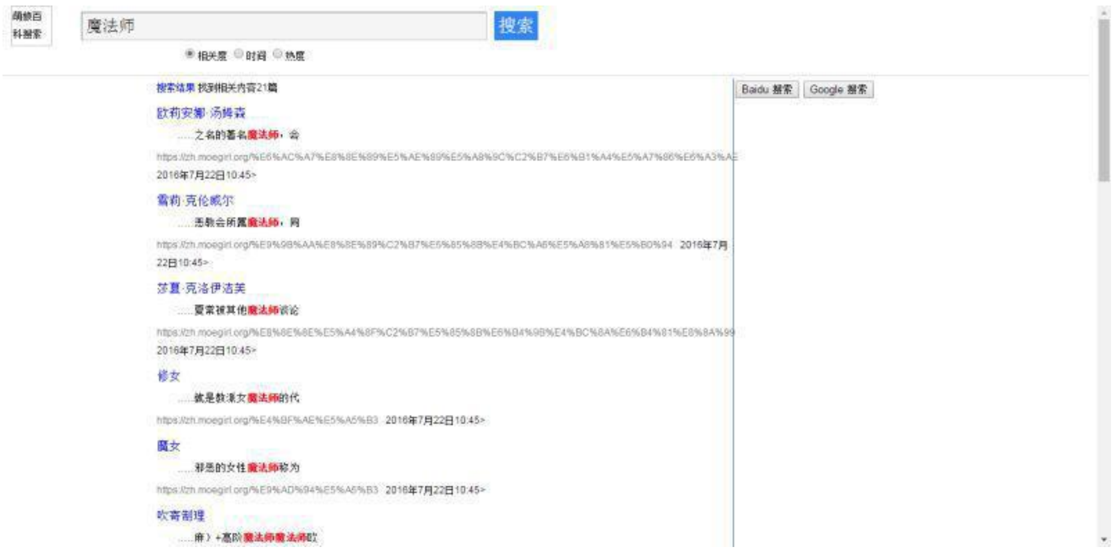
美狄亚(Fate)
https://zh.moegirl.org/%E7%BE%8E%E7%8B%84%E4%BA%9A(Fate) 2016年7月22日10:45>

尼禄(Fate)
https://zh.moegirl.org/%E5%B0%BC%E7%A6%84(Fate) 2016年7月22日10:45>

伊莉雅苏菲尔·冯·爱因兹贝伦
https://zh.moegirl.org/%E4%BC%8A%E8%8E%89%E9%9B%85 2016年7月22日10:45>

魔枪Gae·Dearg

Figure 1-1 English search results



搜索结果 找到相关内容21篇

欧莉安娜·汤姆森
……之名的著名魔法师，会
https://zh.moegirl.org/%E6%AC%A7%E8%8E%89%E5%AE%89%E5%A8%9C%C2%B7%E6%B1%A4%E5%A7%86%E6%A3%AE
2016年7月22日10:45>

露莉·克伦威尔
……恶魅会所属魔法师，同
https://zh.moegirl.org/%E9%9C%B2%E8%8E%89%C2%B7%E5%85%8B%E4%BC%A6%E5%A8%81%E5%B0%94 2016年7月
22日10:45>

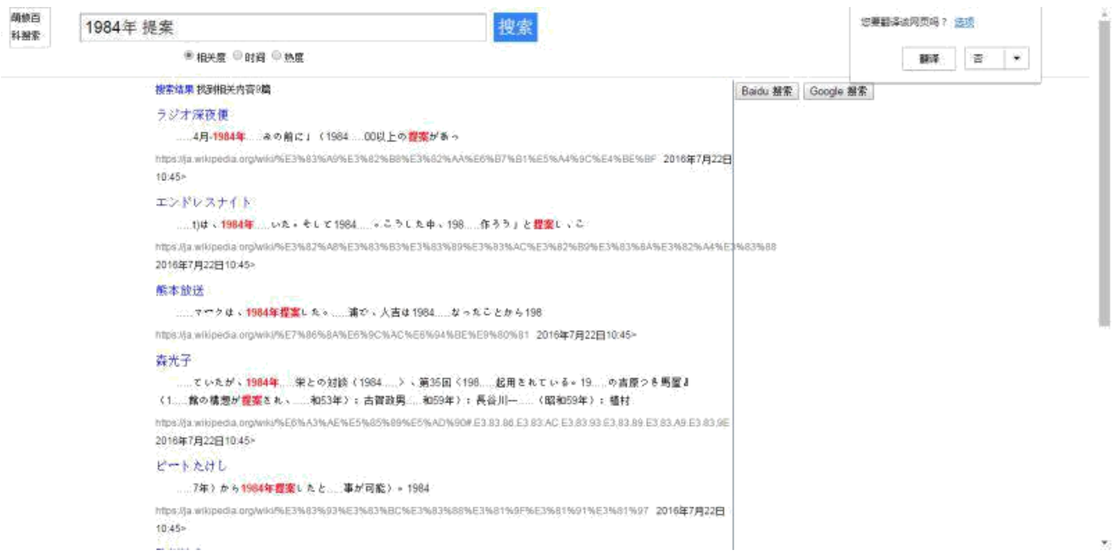
莎夏·克洛伊洁芙
……要常被其他魔法师谈论
https://zh.moegirl.org/%E8%8E%8E%E5%A4%8F%C2%B7%E5%85%8B%E6%B4%9B%E4%BC%8A%E6%B4%81%E8%8A%99
2016年7月22日10:45>

修女
……就是教派女魔法师的代
https://zh.moegirl.org/%E4%BF%AE%E5%A5%B3 2016年7月22日10:45>

魔女
……邪恶的女性魔法师称为
https://zh.moegirl.org/%E9%AD%94%E5%A5%B3 2016年7月22日10:45>

吹寄制理
……席）+高阶魔法师魔法师歌

Figure 1-2 Chinese search results


萌娘百科搜索

1984年 提案 搜索

◉ 相关度 ○ 时间 ○ 热度

搜索结果 找到相关内容9篇

ラジオ深夜便

.....4月-1984年.....その前に」（1984.....00以上の提案があっ

https://ja.wikipedia.org/wiki/%E3%83%A9%E3%82%B8%E3%82%AA%E6%B7%B1%E5%A4%9C%E4%BE%BF 2016年7月22日 10:45>

エンドレスナイト

.....1)は、1984年.....いた。そして1984.....。こうした中、198.....作ろう」と提案し、こ

https://ja.wikipedia.org/wiki/%E3%82%A8%E3%83%B3%E3%83%89%E3%83%AC%E3%82%B9%E3%83%8A%E3%82%A4%E3%83%88 2016年7月22日 10:45>

熊本放送

.....マークは、1984年提案した。.....蒲で、人吉は1984.....なったことから198

https://ja.wikipedia.org/wiki/%E7%86%8A%E6%9C%AC%E6%94%BE%E9%80%81 2016年7月22日 10:45>

森光子

.....ていたが、1984年.....栄との対談（1984.....）、第35回（198.....起用されている。19.....の吉原つき馬屋』（1.....館の構想が提案され、.....和53年）：古賀政男.....和59年）：長谷川一.....（昭和59年）：植村

https://ja.wikipedia.org/wiki/%E6%A3%AE%E5%85%89%E5%AD%90#.E3.83.86.E3.83.AC.E3.83.93.E3.83.89.E3.83.A9.E3.83.9E 2016年7月22日 10:45>

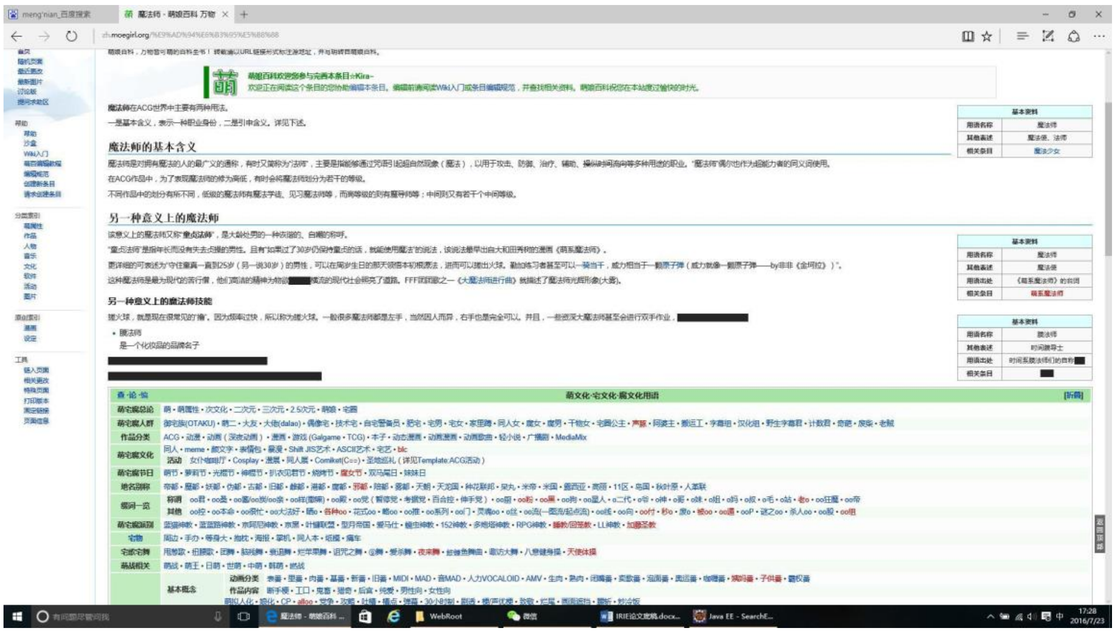
ビートたけし

.....7年）から1984年提案したと.....事が可能）。1984

https://ja.wikipedia.org/wiki/%E3%83%93%E3%83%BC%E3%83%88%E3%81%9F%E3%81%91%E3%81%97 2016年7月22日 10:45>

Figure 1-3 multi field search results

We can also take a look at the results returned by the same keywords using the search function of the web page, as shown in the figure below. When the search content is an entry, the page will automatically enter the explanation of the entry:


魔法师 - 萌娘百科 万物

zh.moegirl.org/%E9%AD%94%E6%B3%95%E5%B8%88

萌娘百科欢迎您参与完善本条目☆Kira~

欢迎正在阅读这个条目的您协助编辑本条目。编辑前请阅读Wiki入门或条目编辑规范，并查找相关资料。萌娘百科祝您在本站度过愉快的时光。

魔法师在ACG世界中主要有两种用法。

一是基本含义，表示一种职业身份，二是引申含义。详见下述。

魔法师的基本含义

魔法师是对拥有魔法的人的最广义的通称，有时又简称为"法师"，主要是指能够通过咒语引起超自然现象（魔法），以用于攻击、防御、治疗、辅助、操纵时间流向等多种用途的职业。"魔法师"偶尔也作为超能力者的同义词使用。

在ACG作品中，为了表现魔法师的修为高低，有时会将魔法师划分为若干的等级。

不同作品中的划分有所不同，低级的魔法师有魔法学徒、见习魔法师等，而高等级的则有魔导师等；中间则又有若干个中间等级。

另一种意义上的魔法师

该意义上的魔法师又称"童贞法师"，是大龄处男的一种诙谐的、自嘲的称呼。

"童贞法师"是指年长而没有失去贞操的男性。且有"如果过了30岁仍保持童贞的话，就能使用魔法"的说法，该说法最早出自大和田秀树的漫画《萌系魔法师》。

更详细的可表述为"守住童真一直到25岁（另一说30岁）的男性，可以在周岁生日的那天领悟本初根源法，进而可以搓出火球。勤加练习者甚至可以一骑当千，威力相当于一颗原子弹（威力就像一颗原子弹——by菲菲《金田拉》）"。

这种魔法师是最为现代的苦行僧，他们高洁的精神为物欲[redacted]横流的现代社会照亮了道路。FFF团团歌之一《大魔法师进行曲》就描述了魔法师光辉形象(大雾)。

另一种意义上的魔法师技能

搓火球，就是现在很常见的"撸"。因为频率过快，所以称为搓火球。一般很多魔法师都是左手，当然因人而异，右手也是完全可以。并且，一些资深大魔法师甚至会进行双手作业。[redacted]

- 魔法师

  是一个化妆品的品牌名子

[redacted]

基本资料	
用语名称	魔法师
其他表述	魔法使、法师
相关条目	魔法少女

基本资料	
用语名称	魔法师
其他表述	魔法使
用语出处	《萌系魔法师》的台词
相关条目	萌系魔法师

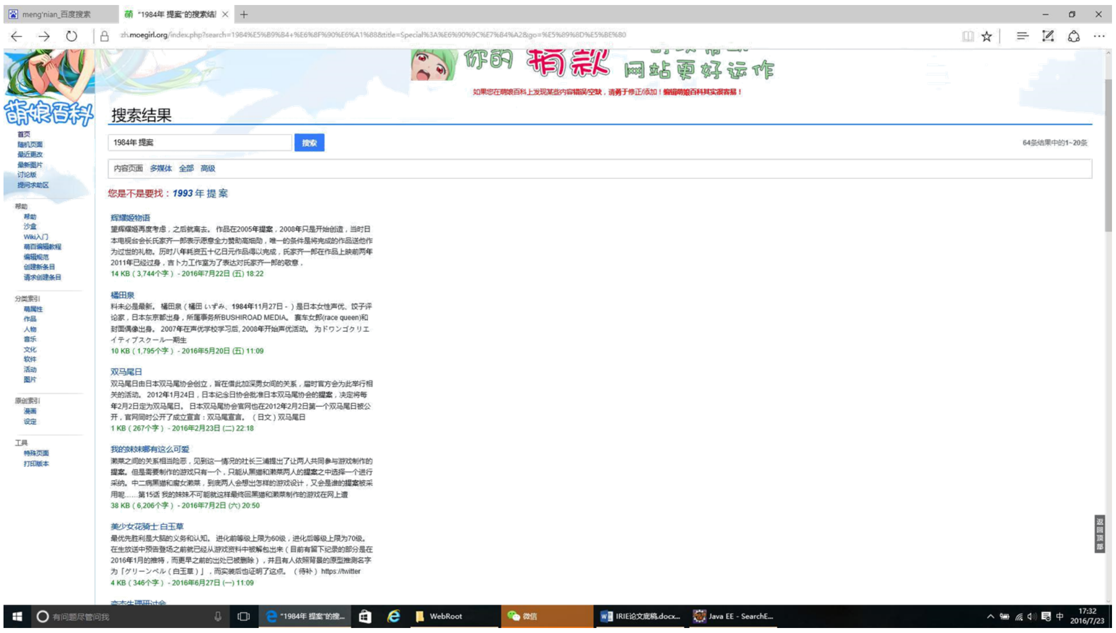
基本资料	
用语名称	膜法师
其他表述	时间膜导士
用语出处	时间系膜法师们的自称[redacted]
相关条目	[redacted]

查·论·编 萌文化·宅文化·腐文化用语 [折叠]

[illegible]

Figure 1-4 entry entry

When the input content is not a known term, the returned results are as follows:


搜索结果

1984年 提案

搜索

内容页面 多媒体 全部 高级

您是不是要找：*1993* 年 提 案

辉耀姬物语
望辉耀姬再度考虑，之后就离去。 作品在2005年提案，2008年只是开始创造，当时日本电视台会长氏家齐一郎表示愿意全力赞助高畑勋，唯一的条件是将完成的作品送他作为过世的礼物。历时八年耗资五十亿日元作品得以完成，氏家齐一郎在作品上映前两年2011年已经过身，吉卜力工作室为了表达对氏家齐一郎的敬意，
14 KB（3,744个字）- 2016年7月22日 (五) 18:22

橘田泉
料未必是最新。 橘田泉（橘田 いずみ、1984年11月27日 - ）是日本女性声优、饺子评论家，日本东京都出身，所属事务所BUSHIROAD MEDIA。 赛车女郎(race queen)和封面偶像出身。 2007年在声优学校学习后, 2008年开始声优活动。 为ドワンゴクリエイティブスクール一期生
10 KB（1,795个字）- 2016年5月20日 (五) 11:09

双马尾日
双马尾日由日本双马尾协会创立，旨在借此加深男女间的关系，届时官方会为此举行相关的活动。 2012年1月24日，日本纪念日协会批准日本双马尾协会的提案，决定将每年2月2日定为双马尾日。 日本双马尾协会官网也在2012年2月2日第一个双马尾日被公开，官网同时公开了成立宣言：双马尾宣言。 （日文）双马尾日
1 KB（267个字）- 2016年2月23日 (二) 22:18

我的妹妹哪有这么可爱
濑菜之间的关系相当险恶，见到这一情况的社长三浦提出了让两人共同参与游戏制作的提案。但是需要制作的游戏只有一个，只能从黑猫和濑菜两人的提案之中选择一个进行采纳。中二病黑猫和腐女濑菜，到底两人会想出怎样的游戏设计，又会是谁的提案被采用呢……第15话 我的妹妹不可能就这样最终回黑猫和濑菜制作的游戏在网上遭
38 KB（6,206个字）- 2016年7月2日 (六) 20:50

美少女花骑士:白玉草
最优先胜利是大脑的义务和认知。 进化前等级上限为60级，进化后等级上限为70级。在生放送中预告登场之前就已经从游戏资料中被解包出来（目前有留下纪录的部分是在2016年1月的推特，而更早之前的出处已被删除），并且有人依照背景的原型推测名字为「グリーンベル（白玉草）」，而实装后也证明了这点。 （待补） https://twitter
4 KB（346个字）- 2016年6月27日 (一) 11:09

图 1-5 search for ″1984″ ″proposal″

Obviously, the returned results are different from our engine. We have more returned results. The official search engine

We don't know what algorithm is used in. For example, they can sort and filter search results according to the number of visits of a page, which we can't realize.

In addition, the interface of the official engine is also better than ours, which is what we need to improve. We should beautify it according to the two-dimensional enthusiasts.

6.2 explanation of front end design code segment

The following is the relevant code of our front end:

```
TreeMap<String,String> map1 = new

TreeMap<String,String>(new Comparator(){

          public int compare(Object o1, Object o2)
              { // TODO Auto-generated method stub
              return o2.toString().compareTo(o1.toString());

          }


      });


        //Treemap by heat, newly added
     TreeMap<String,String> map2 = new

TreeMap<String,String>( new Comparator(){

          public int compare(Object o1, Object o2)
              { // TODO Auto-generated method stub
              return o2.toString().compareTo(o1.toString());

          }


          });
```

We first sort the retrieved results according to the ranking obtained by VSM algorithm, from high to low heat.

When the input keyword is not empty, the absolute position of the content containing the keyword is found.

The list variable is used to record the number of returned results. When the list is null, it will prompt "nothing found"

```
}else
        {
        %>
                <div class="bobyTitle">

                    <div class="bobyContent">
                         Nothing found!!!
                    </div>
                </div>
                  <%} %>
```

The design of the page is completed in HTML. Our display items include heat, number of results, etc. the implementation method is as follows:

```
<div id="head">
         < a herf = "index. JSP" id = "input_a" > < img ALT =
"mengniang Encyclopedia Search" class = "inputimg" ></a>
            <form id="fmSearch" method="post" action="index.jsp">
                <input type="text" name="keyWord" class="inputText"
value="<%=keyword %>"/>
           &nbsp;
           < input type = "submit" value = "search" class =
           "inputsubmit" / >
           <form method="post" action="index.jsp" class="formRadio">
               <div style="clear:both;margin-left:230px;">
              <input type="radio" name="radio" value="score"
Onclick = "save()" / > correlation
              <input type="radio" name="radio" value="time"
Onclick = "save()" / > time
              <input type="radio" name="radio" value="hottop"
Onclick = "save()" / > heat
              <! -- input type = "submit" value =
               "submit" / - > < / form >
              </div>
               </form>
```

According to the previous sorting, we will display the address link and profile information on the page one by one, and the corresponding code segment is shown in the figure below:

```
}
           for(String o:list)
           {
               //Engine.ResultModel mod = (Engine.ResultModel)o;
               //Engine.Result mod=new Result(o+".txt");
               //ucas.JsonObject myObject =
ucas.JsonUtil.readFile2JsonObject("/data/"+o+".json");
```

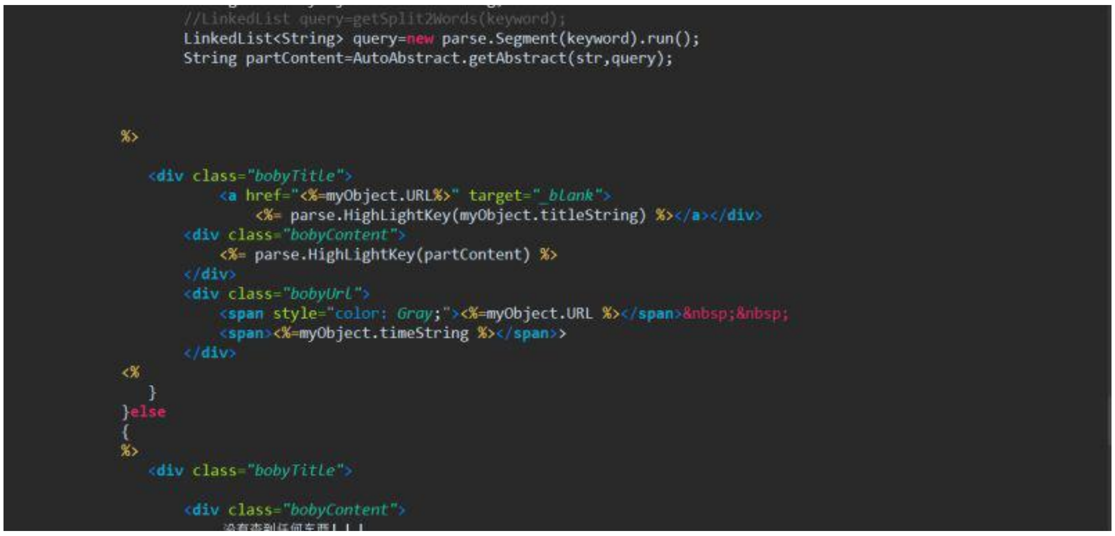
```
            String path= parse.Path(strPath,o);
               ucas.JsonObject myObject =
ucas.JsonUtil.readFile2JsonObject(path);
               String str = myObject.articalString;
               //LinkedList query=getSplit2Words(keyword);
               LinkedList<String> query=new
parse.Segment(keyword).run();
               String partContent=AutoAbstract.getAbstract(str,query);
        %>
       。。。。。。。。。。。。。。。。。。。。。。。。
```

Finally, as a function expansion, we also added Google and Baidu links to the page for everyone to use. After all, our database size is still very limited. The relevant codes are as follows:

```
    <form action="http://www.baidu.com/baidu">
                         <input type=hidden value='<%=keyword %>'
name=word>
                         < input type = "submit" value = "Baidu search" >
                         <input name=tn type=hidden value="bds">
                         <input name=cl type=hidden value="3">
                         <input name=ct type=hidden value="2097152">
                         <input name=si type=hidden
value="www.williamlong.info">
                      </form>
                      </td>
```

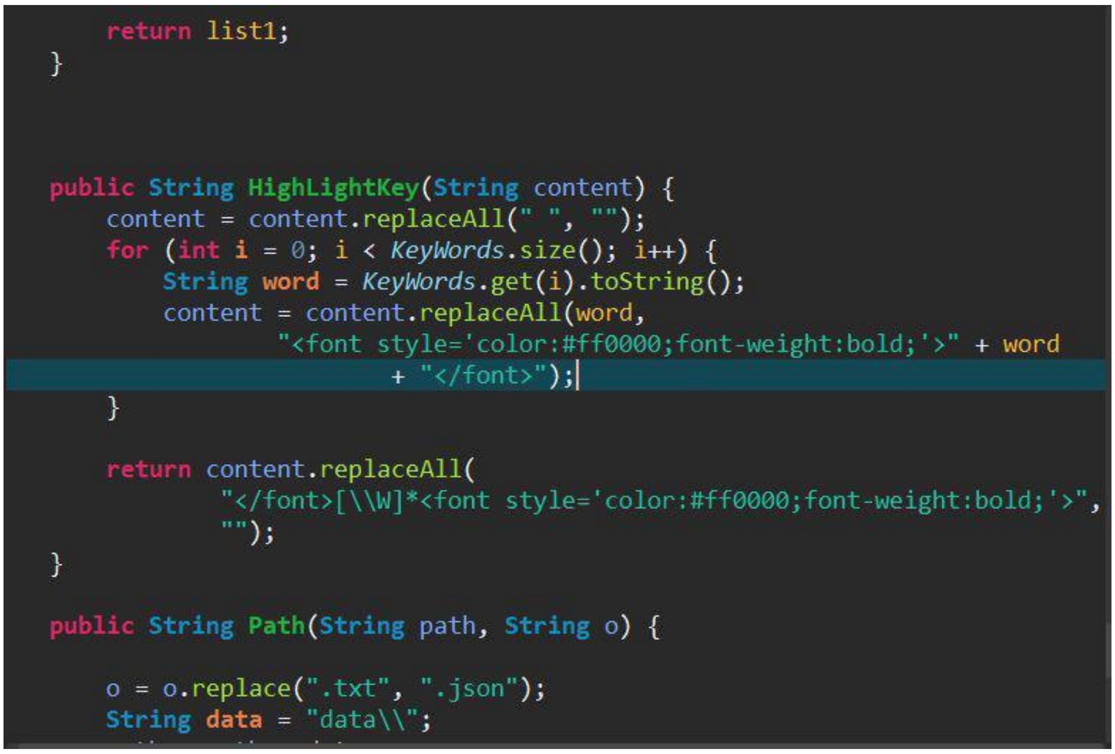
7、 Code description of information extraction display part

Core code display:



```
            //LinkedList query=getSplit2Words(keyword);
            LinkedList<String> query=new parse.Segment(keyword).run();
            String partContent=AutoAbstract.getAbstract(str,query);



        %>

          <div class="bobyTitle">
                <a href="<%=myObject.URL%>" target="_blank">
                    <%= parse.HighLightKey(myObject.titleString) %></a></div>
            <div class="bobyContent">
                <%= parse.HighLightKey(partContent) %>
            </div>
            <div class="bobyUrl">
                <span style="color: Gray;"><%=myObject.URL %></span>&nbsp;&nbsp;
                <span><%=myObject.timeString %></span>>
            </div>
        <%
          }
        }else
        {
        %>
          <div class="bobyTitle">

            <div class="bobyContent">
                没有查到任何东西！！！
```

Figure 2-1 code diagram



```
        return list1;
    }


    public String HighLightKey(String content) {
        content = content.replaceAll(" ", "");
        for (int i = 0; i < KeyWords.size(); i++) {
            String word = KeyWords.get(i).toString();
            content = content.replaceAll(word,
                    "<font style='color:#ff0000;font-weight:bold;'>" + word
                            + "</font>");
        }

        return content.replaceAll(
                "</font>[\\W]*<font style='color:#ff0000;font-weight:bold;'>",
                "");
    }

    public String Path(String path, String o) {

        o = o.replace(".txt", ".json");
        String data = "data\\";
```

Figure 2-2 schematic diagram of search code

The implementation of information retrieval is shown in the figure above. The overall idea is that we traverse all index files on the whole website, record the pages containing the input keywords, then display the upper and lower text, highlight the keywords, and record their URLs for users to access.

Our husband becomes a link to the target URL. The page displays the inclusion sentence with keywords, in which the keywords are highlighted, as shown in Figure 2-1.

The implementation of highlighting is relatively simple. We only need to modify the color and font of the keywords recorded earlier, as shown in Figure 2-2.

For the display of the text introduced below, we use the keyword itself as the benchmark to display the characters with fixed length before and after it (the specific length will be adjusted according to the number of keywords).

In this way, we have completed the preliminary extraction of relevant information, which is actually to return the context content of keywords.

Obviously, we will return many related pages. We will display them according to the VSM sorting algorithm.

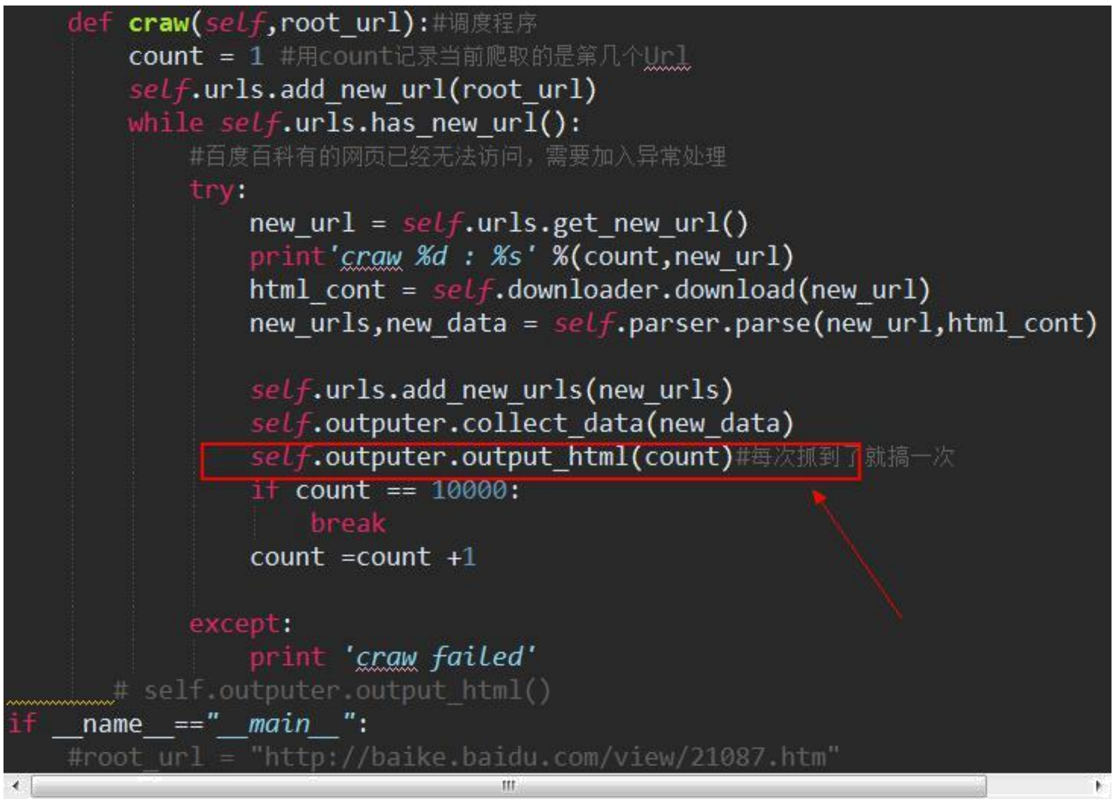
8、 Problems encountered in development

8.1 problems encountered in writing crawler

When I first wrote the crawler program, I only crawled 100 Web pages as a test, so in the code, I uniformly stored the crawled data in a list, and then uniformly saved all the web pages. However, when the data is very large, it may cause heap depletion and lead to memory overflow. Therefore, later, I changed to perform persistence operation immediately after crawling a web page, and then, Clear the data in the memory in time to prevent insufficient memory space.

If a URL downloaded by the crawler is invalid, the program will crash, so exception handling is added to the circular program to prevent the crawler from stopping because of the 404 page.

The modified code is as follows:

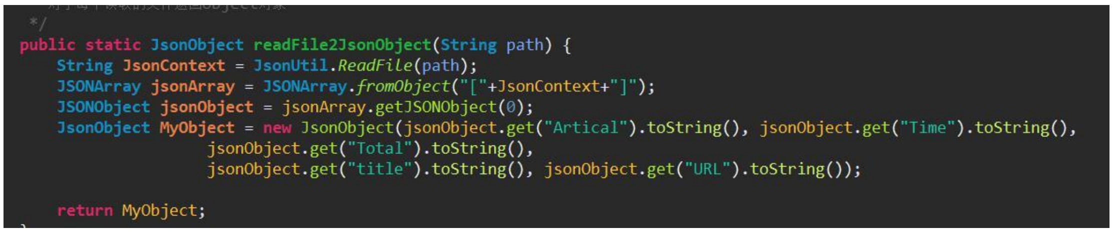


```
    def craw(self,root_url):#调度程序
        count = 1 #用count记录当前爬取的是第几个Url
        self.urls.add_new_url(root_url)
        while self.urls.has_new_url():
            #百度百科有的网页已经无法访问，需要加入异常处理
            try:
                new_url = self.urls.get_new_url()
                print'craw %d : %s' %(count,new_url)
                html_cont = self.downloader.download(new_url)
                new_urls,new_data = self.parser.parse(new_url,html_cont)

                self.urls.add_new_urls(new_urls)
                self.outputer.collect_data(new_data)
                self.outputer.output_html(count)#每次抓到了就搞一次
                if count == 10000:
                    break
                count =count +1

            except:
                print 'craw failed'
       # self.outputer.output_html()
if __name__=="__main__":
    #root_url = "http://baike.baidu.com/view/21087.htm"
```

8.2 problems encountered in writing search engine programs

The background of our search engine is written by using java language and relying on the existing framework for secondary development, but the data structure of a page of mengniang encyclopedia obtained by the crawler is a file in JSON format redefined by ourselves, so we need to rewrite the coding and data reading and writing interfaces, otherwise null pointer exception and other exceptions will appear in the background Read in index file garbled.

In order to solve the above problems, we modified the original Unicode encoding format in the framework to UTF-8, solved the problem of garbled code, rewritten the code of the read-in part of the JSON format file, and solved the exception of data read-write null pointer. The modified code is as follows.



```
 */
public static JsonObject readFile2JsonObject(String path) {
    String JsonContext = JsonUtil.ReadFile(path);
    JSONArray jsonArray = JSONArray.fromObject("["+JsonContext+"]");
    JSONObject jsonObject = jsonArray.getJSONObject(0);
    JsonObject MyObject = new JsonObject(jsonObject.get("Artical").toString(), jsonObject.get("Time").toString(),
                    jsonObject.get("Total").toString(),
                    jsonObject.get("title").toString(), jsonObject.get("URL").toString());

    return MyObject;
```

9、 Group division of labor

Gao chuyang: development of information index and
relevance ranking code, preparation of relevant
documents, Yang Xue: development and testing of
crawler system, preparation of relevant documents,
Feng Yimeng: development and testing of information
extraction system, preparation of relevant documents

Sun Chuanhao: implementation of web pages and preparation of relevant documents
The workload of the four people is equal, and the teacher can give them points as appropriate.