第六次实验报告

学号：518030910308
姓名：刘文轩

一、实验准备

1、 实验环境介绍

操作系统：ubuntu 14.04
语言：Python 2
IDE：Pycharm 2019.2.3

2、 实验目的

2.1 使用 web.py 建立简单的搜索引擎
2.2 学会提取关键词上下文的方法
2.3 使用 web.py 建立 web 开发框架

3、实验思路

3.1 修改 IndexFiles.py 得到 IndexFiles_6.py，使得 doc 中保存了原文本内容，便于返回关键词上下文
3.2 修改 SearchFiles.py，得到 Search_6.py，使得能够关键词返回其对应的上下文
3.3 修改 web.py 以及其对应的 templates，使得网站是我们需要的形式

二、实验过程

1、 编写 IndexFiles_6.py

1.1 保存 contents_raw

IndexFiles_6.py 代码的主体框架依然是实验 4 中的 IndexFiles.py，但是这里遇到的问题时我们需要根据关键词返回对应的上下文，因此我们需要保存原有的上下文，而不是分词后获得的 contents，所以我们向 doc 中存入两个和内容有关的属性：contents 用来索引，并不完整储存，contents_raw 用来获取关键词对应的上下文，完整储存但并不建立索引。

同时这里有一点很需要注意，如果为了滤去原文件中的 tag，我们仅仅使用 soup.findAll(text=True)，那么返回的内容中注释里的文字也会保留，在遇到如下的网站时：

```
<!DOCTYPE html>
<html lang="zh-Hans">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">
    <meta http-equiv="Cache-Control" content="no-siteapp" />
    <meta name="renderer" content="webkit">
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <meta name="SiteName" content="上海交通大学新闻学术网">
    <meta name="SiteDomain" content="www.sjtu.edu.cn">
    <meta name="ColumnDescription" content="">
    <meta name="ColumnKeywords" content="">
    <meta name="ColumnType" content="活力校园">
    <meta name="ColumnName" content="活力校园">
    <title>活力校园_上海交通大学新闻学术网</title>
    <!-- Core CSS -->
    <link rel="stylesheet" href="/resource/assets/css/ETUI/ETUI3.min.css">
    <link rel="stylesheet" href="/resource/assets/css/ETUI/ETUI3.Utility.css">
    <!-- Widget -->
    <link rel="stylesheet" href="/resource/assets/plugin/infinitypush/jquery.ma.infinitypush.css">
    <link rel="stylesheet" href="/resource/assets/css/keyframes.css">
    <!-- Custom CSS -->
    <link rel="stylesheet" href="/resource/assets/css/style.css">
    <link rel="stylesheet" href="/resource/assets/css/style-responsive.css">
    <link rel="stylesheet" href="/resource/assets/css/style-compatible.css">
    <!-- Site Icon -->
    <link rel="shortcut icon" href="/resource/assets/img/ico/favicon.png">
</head>
```

内容里会返回带有“Core CSS Widgt Custom CSS”等内容，这很不利于我们返回关键词上下文，因此我们保存 contents_raw 时再额外选择一步，我们通过 contents_raw = soup.get_text()来获取去除了不需要的注释的内容。

```
contents = file.read()
soup = BeautifulSoup(contents, features="html.parser")
title = soup.head.title.string
contents = ''.join(soup.findAll(text=True))
contents_raw = soup.get_text()
```

1.2 修改 doc 部分

doc 部分需要修改的部分代码如下：

```
doc.add(Field("contents_raw", contents_raw,
              Field.Store.YES,
              Field.Index.NOT_ANALYZED))
if len(contents) > 0:
    doc.add(Field("contents", contents,
                  Field.Store.NO,
                  Field.Index.ANALYZED))
```

2、编写 Search_6.py

2.1 设计 run_txt()函数

搜索的主要部分与之前的 SearchFiles.py 相同，但是我们需要额外加入根据关键词返回上下文的部分，我们选择找到关键词之后，向前搜索 10 位，向后搜索 50 位，但是需要注意的是，这样子可能带来位置的越界，因此我们需要添加判断部分。

我们在此同时加入关键词的高亮显示：

```
contents_raw = doc.get('contents_raw')
keywords = list()
if 'contents' in command_dict:
    keywords += list(jieba.cut(command_dict['contents']))
beginning = min(map(contents_raw.find, keywords))
if beginning < 0:
    continue
para = contents_raw[max(0, beginning - 10): beginning + 50]
for keyword in keywords:
    para = para.replace(keyword.strip(), '<span style="color:blue;font-weight:bold">'+keyword+'</span>')
partResult['keyword'] = para
result.append(partResult)
```

2.2 设计主函数

使用 PyLucene 时，注意把 vm_env = initVM()一句写在主函数中，每次搜索时使用 vm_env.attachCurrentThread()新建线程，可阻止多次搜索程序崩溃。

```
STORE_DIR = "index_6"
vm_env = lucene.initVM(vmargs=['-Djava.awt.headless=true'])


def lab7txtSearch(valueFromOut):
    vm_env.attachCurrentThread()
    print 'lucene', lucene.VERSION
    # base_dir = os.path.dirname(os.path.abspath(sys.argv[0]))
    directory = SimpleFSDirectory(File(STORE_DIR))
    searcher = IndexSearcher(DirectoryReader.open(directory))
    analyzer = StandardAnalyzer(Version.LUCENE_CURRENT)
    result = run_txt(valueFromOut, searcher, analyzer)
    del searcher
    return result
```

3、设计 web.py

3.1 设置 index 类与 formtest.html

当我们登录时，我们默认进入此界面，即网站直接以/结尾。

我们设计的 index 类如下：

```
class index:
    def GET(self):
        f = login()
        return render.formtest(f)
```

formtest.html 返回了我们需要的首页内容，整体代码较为简单，需要注意的是，如果我们不需要将文本当作字符直接输出，而是要执行原程序的命令，需要在变量或命令前加$符号：

```
$def with(form)
<body bgcolor="white"></body>

<hi align="center">My Search Engine</h1><br><br>

<form action="/s" method="GET">
    $:form.render()
</form>
```

3.2 设置 s 类与 result_index_txt.html

这里需要注意的一点是，我们在获得用户的搜索关键词之后，先将其传给 Search_6.py

中的函数进行处理，再将返回的内容给网站进行处理。

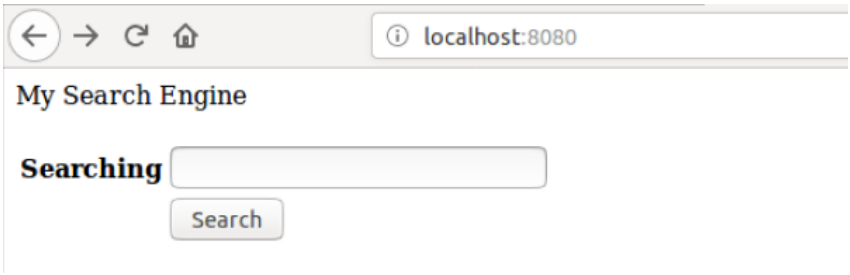
```
class s:
    def GET(self):
        user_data = web.input()
        reuslt_txt = func_txt(user_data.Searching)
        return render.result_search_txt(user_data.Searching,reuslt_txt)
```

我们编写的代码中，对于关键词的高亮显示，是在其两边直接加上 html 的 tag 来实现的，但是对于 html 解析时，即使在变量名前加上了$仍然只会返回字符串，我们需要加上$:来限制转义。

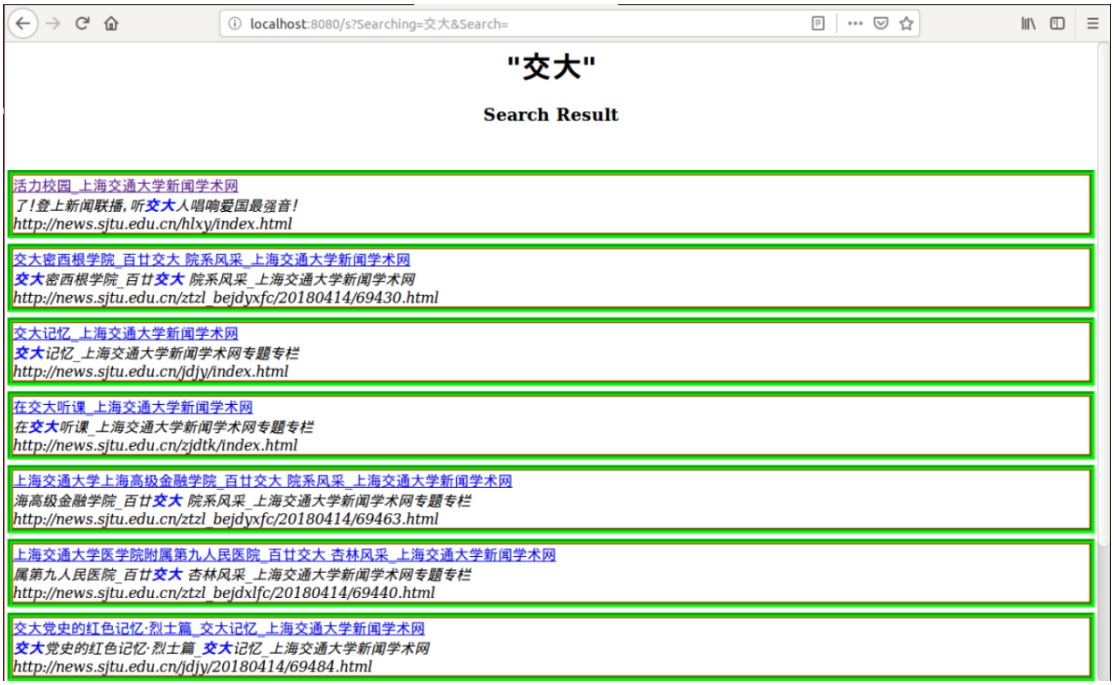
```
$def with (a,name)

<head>
<style type="text/css">
p
{
    background-color: white;
border:red solid thin;
outline:#00ff00 groove 5px;
}
h1{
    text-align :center;
    }
h3{
    text-align :center;
    }
</style>
</head>

<h1>"$a" </h1><h3>Search Result</h3><br>

$if name:
    $for i in name:
        <p>
            <a href="$i['url']">$i['title']</a><br>
            <em>$:i['keyword']</em><br>
            <em>$i['url']</em><br>
        </p>


$else:
    <em>Hello</em>, world!
```

3.3 代码运行结果

至此，相关程序的修改就已经完成，我们在已经运行了 web_6.py 的情况下，直接登录 http://localhost:8080/，返回的界面如下：



我们搜索交大，返回的界面如下：



由图可见，搜索结果还是比较恰当的。返回的结果中关键词也很准确。

三、实验总结

1、实验概述

本次实验的主要任务，可以总结为建立一个简单的搜索引擎，内容新涉及了 web.py，总体相当重要。

2、感想总结

在这次的实验中，学会了的东西有很多，其中最重要的就是提高了自己处理问题、收集相关资料、解决问题的能力，这在我们将来的学习和工作生活中都是很重要的。而具体细化开来，在本次实验中：

2.1 了解了 web.py 的有关内容

2.2 学会了更好的过滤文本的方法

2.3 学会了搜寻关键词上下文的方法

2.4 学会了高亮关键词的方法

3、创新点

首先是为了更好承接上一次实验的内容，我将索引文件获取的过程拆分成从 index.txt 中获取 url 与文件名，再从 html 文件夹中找到文件。

同时我使用了一种相对而言更加良好的的过滤 html 中的 tag 的方式，同时避免了很多注释之类的无用内容的影响，使得返回内容更加直观明了：

```
contents = file.read()
soup = BeautifulSoup(contents, features="html.parser")
title = soup.head.title.string
contents = ''.join(soup.findAll(text=True))
contents_raw = soup.get_text()
```

4、遇到的问题

返回关键词上下文的过程中，可能会遇到指定范围越界的问题，我们需要加以判断：

```
contents_raw = doc.get('contents_raw')
keywords = list()
if 'contents' in command_dict:
    keywords += list(jieba.cut(command_dict['contents']))
beginning = min(map(contents_raw.find, keywords))
if beginning < 0:
    continue
para = contents_raw[max(0, beginning - 10): beginning + 50]
for keyword in keywords:
    para = para.replace(keyword.strip(), '<span style="color:blue;font-weight:bold">'+keyword+'</span>')
partResult['keyword'] = para
result.append(partResult)
```

同时在高亮关键词的时候，由于是在网页中解析，我们可以在关键词的两端加上 tag，但是需要注意的是，这样由于是以字符串形式保存的，网页解析时会直接返回字符串，我们需要限制转义，这一点十分重要。我们可以通过在 html 文件中的变量前加上$:的方式来避免这个问题。