实验六报告

唐灵
519030910052
F1903002

2020 年 11 月 5 日

摘要

这是电工导 c 课程的第五次实验

1 实验概览

本次实验情况特殊，实验开始的时候，我在 docker 中连 searchFiles.py 都没有办法运行，关于环境的问题已经严重影响了我之前两次的作业!

无法每次都借同学的电脑，所以最终选择逃离 java，选择 whoosh 作为文本检索的框架进行进一步的实验。

话说回来，本次实验使用 Flask，结合前面学习的 HTML，中文分词等知识点，根据上次实验爬取的网页，建立一个简单的搜索引擎。

在这次实验中，我将用 whoosh 取代 Lucene，并重写建立索引，写入索引，以及搜索模块三个模块的内容。这也是对上一次无法进行实验的弥补。

当然，也完成了本次实验的关键部分，也就是将使用 flask，完成基本的网页构建。

2 实验环境

本次作业文本检索部分使用了 whoosh，其他部分和课程要求内容完全一致，在本地环境中使用 anaconda3 完成实验，python 的版本为 3.8.5，具体环境环境如下：

```
(web) downing@Tonys-MacBook-Pro lab6-Flask % conda list
# packages in environment at /Users/downing/opt/anaconda3/envs/web:
#
# Name                    Version                   Build  Channel
beautifulsoup4            4.9.3                     pypi_0    pypi
ca-certificates           2020.10.14                     0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
certifi                   2020.6.20                 py38_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
click                     7.1.2                     pypi_0    pypi
flask                     1.1.2                     pypi_0    pypi
gevent                    20.9.0                    pypi_0    pypi
greenlet                  0.4.17                    pypi_0    pypi
itsdangerous              1.1.0                     pypi_0    pypi
jieba                     0.42.1                    pypi_0    pypi
jinja2                    2.11.2                    pypi_0    pypi
libcxx                    10.0.0                         1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
libedit                   3.1.20191231          h1de35cc_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
libffi                    3.3                   hb1e8313_2    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
markupsafe                1.1.1                     pypi_0    pypi
ncurses                   6.2                   h0a44026_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
openssl                   1.1.1h                haf1e3a3_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
pip                       20.2.3                    py38_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
python                    3.8.5                 h26836e1_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
readline                  8.0                   h1de35cc_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
request                   2.83.1                         0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
setuptools                50.3.0           py38h0dc7051_1    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
soupsieve                 2.0.1                     pypi_0    pypi
sqlite                    3.33.0                hffcf06c_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
tk                        8.6.10                hb0a8c7a_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
werkzeug                  1.0.1                     pypi_0    pypi
wheel                     0.35.1                      py_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
whoosh                    2.7.4                     pypi_0    pypi
xz                        5.2.5                 h1de35cc_0    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
zlib                      1.2.11                h1de35cc_3    https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
zope-event                4.5.0                     pypi_0    pypi
zope-interface            5.1.2                     pypi_0    pypi
```

图 1: 实验运行环境

3 练习解决思路

3.1 总体思路

总体的实现思路大致可以描述如下：

1. 服务器初始化根目录为搜索页面，通过 search.html 渲染模版。
2. 程序猿从浏览器发起搜索，通过搜索页面向服务器发送请求，通过重定向到结果页面（由于页面设置，结果页面本身也是搜索页面），请求中后缀有”keyword” 参数。
3. 服务器通过获得 keyword 参数，再将 keyword 传入到 search() 函数，并返回搜索结果，这个函数来自于搜索模块，这个模块会对之前创建的索引进行搜索，并返回所有的搜索结果，这个结果足够覆盖结果页面中我们用到的所有信息，并且已经进行过高亮处理。
4. 将返回的搜索结果作为参数传入进网页模版 result.html 进行渲染，得到我们的搜索结果，再传回给浏览器。

3.2 请求表单

请求表单具有发送带有参数的请求的功能，写法如下图一般质朴无华：

```
<h1>Search</h1>
<form action="result">
    <label>Keyword</label>
    <input type="name" name="keyword">
    <input type="submit" name="" value="Search">
</form>
```

图 2: 表单

3.3 搜索模块的实现

搜索模块的实现主要依赖于 Whoosh 的各种类，跟 Lucene 的逻辑非常相似，并且 jieba 有专门针对于 Whoosh 的接口，所以对于中文分词也不必过多的担心，直接在建立索引的时候对于 content 字段的分析器采用 jieba 的 ChineseAnalyzer 即可，这样就省掉了大量的麻烦。

对于 whoosh，必须要求所有的存入的字符都采用 unicode 编码，这使得中文的编码问题也不再是问题。

针对于高亮搜索，我们希望搜索结果返回周围的少部分文字，并且希望关键字被一对合适的 html 标签包裹，这需要我们做进一步的定制：

```
class BracketFormatter(highlight.Formatter):
    def format_token(self, text, token, replace=False):
        tokentext = highlight.get_text(text, token, replace)
        return "<mark>{}</mark>".format(tokentext)
```

图 3: 高亮显示设定（包裹标签定制）

最后值得一提的是，Whoosh 搜索器返回的结果并非是 python 内置类型，如果直接传入到模版中进行迭代会出现问题，所以我们在过程中对这些搜索结果进行转化。

最终返回的是一个列表，列表元素是每一个搜索结果，匹配程度越高排，序号越低。

一个搜索结果由字典表达，字典中包含我们希望返回的信息:

```
{
    'title':'****',
    'abstract':'****',
    'url':'****'
}
```

3.4 网页模版的实现

主要关于结果页的网页模版：

传入关键字和搜索结果两个参数，利用过滤器，得到第一行搜索提示，即返回这样的语句：告诉程序猿关于这个关键字的搜索结果我们找到了多少条？

```
</form>
<h3>We've found {{results|length}} results about "{{keyword}}" here!</h2>
{% for result in results[:20] %}
<div>
```

图 4: 提示语句的写法

从上图也可以看出我们通过 for 循环对于我们的搜索结果进行打印。

最后提一个东西，我们的传入的高亮字段也就是 “abstract”，本身的关键字就已经被 html 标签包裹，但是开始的时候直接打印，在网页端只能够看到纯字符串，经查资料得到，这个设计是基于安全考虑，这里应该使用一个过滤器 “safe”，来使得 html 标签能够被正确识别：

```
<div> <b>Abstract</b> : {{result['abstract']|safe}}</div>
```

图 5: 高亮语句导入

4 代码运行结果

我们最终网页支持单字段和多字段搜索，这都是由自带的分析器进行完成，又少操心了，这里来显示一些搜索结果。

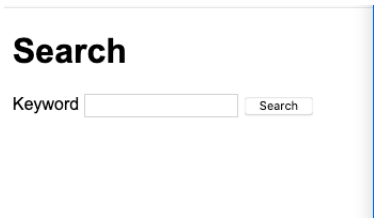

图 6: 初始化页面

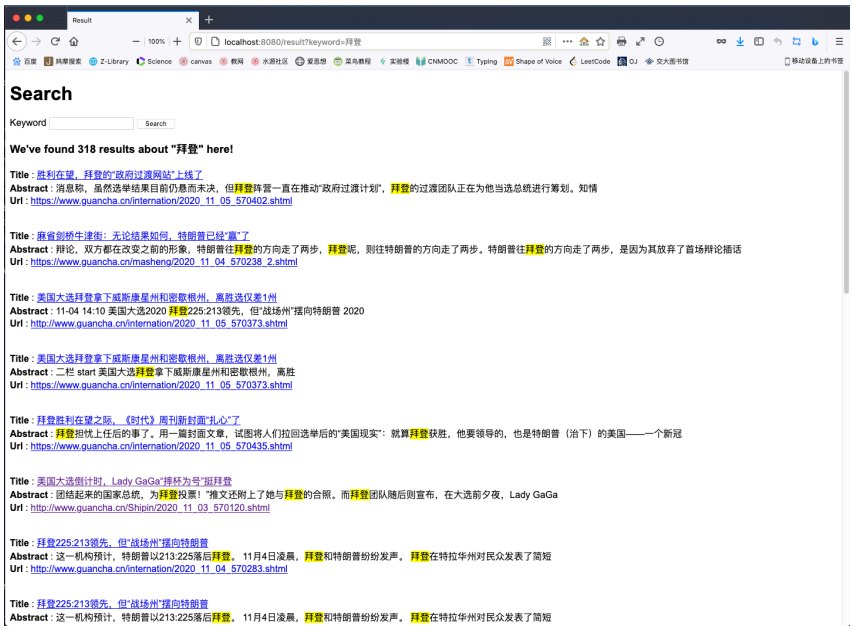

图 7: 搜索页面一

图 8: 搜索页面二