第七次实验报告

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 制定一个图片加文字的搜索引擎
2.2 使用 CSS 定制样式
2.3 实现图片瀑布流显示和自适应大小、卡片式展示

3、实验思路

3.1 修改 Search_6.py 得到 Search_7,py 使其获得对图片的搜索能力
3.2 修改 web_6.py 得到 web_7.py，使得主网站能够处理图片搜索
3.3 设计 formimg.html 和 result_search_pic.html，返回图片的搜索界面
3.4 使用 CSS，设计在 pic 和 txt 中统一的界面风格，极大提升美观程度

二、实验过程

1、 设计 Search_7.py

1.1 添加搜索图片函数

在这一部分中，我们使用的图片索引为之前的实验已经建立的 indeximg，要返回的图片地址、网站地址、图片内容都已经建立了，因此我们只需要直接从之前建立的索引中获取这些内容，run_pic()函数的主体架构与 run_txt()的主体架构差距不大，整体代码如下：

```
def run_pic(valueFromOut, searcher, analyzer):
    command = valueFromOut

    seg_list = jieba.cut(command)
    command = " ".join(seg_list)
    if command == '':
        return

    result = []
    command_dict = parseCommand(command)
    querys = BooleanQuery()
    for k, v in command_dict.iteritems():
        query = QueryParser(Version.LUCENE_CURRENT, k,
                            analyzer).parse(v)
        querys.add(query, BooleanClause.Occur.MUST)
    scoreDocs = searcher.search(querys, 10).scoreDocs
    print "%s total matching documents." % len(scoreDocs)

    for scoreDoc in scoreDocs:
        doc = searcher.doc(scoreDoc.doc)
        partResult = {}

        partResult['title'] = doc.get('title')
        partResult['url'] = doc.get('url')
        partResult['imgurl'] = doc.get('imgurl')

        result.append(partResult)

    return result
```

1.2 设计 lab7picSearch()函数

lab7picSearch()函数的主体部分与 lab7txtSearch()几乎一样，它是 web_7.py 中搜索图片时调用的主函数。但是我们要注意的是，要修改获取索引的名称，由原先的 index 变为了另外的索引 indeximg。

相关代码如下：

```
STORE_DIR_pic = "indeximg"


def lab7picSearch(valueFromOut):
    vm_env.attachCurrentThread()
    print 'lucene', lucene.VERSION
    # base_dir = os.path.dirname(os.path.abspath(sys.argv[0]))
    directory = SimpleFSDirectory(File(STORE_DIR_pic))
    searcher = IndexSearcher(DirectoryReader.open(directory))
    analyzer = StandardAnalyzer(Version.LUCENE_CURRENT)
    result = run_pic(valueFromOut, searcher, analyzer)
    del searcher
    return result
```

2、修改 web_6.py 获得 web_7.py

2.1 修改 urls

这次我们的网站主体上有 4 个界面，分别是文字搜索主界面、图片搜索主界面、文字搜索结果页、图片搜索结果页。因此我们将 urls 设置如下：

```
urls = (
    '/', 'index',
    '/s', 's',
    '/im', 'index_img',
    '/i', 'image'
)
```

2.2 设计 index_img 和 image 类

Index_img 类与文字处理的 index 类一样，image 类与文字处理的 s 类相似，相关代码如下：

```
class index_img:
    def GET(self):
        f = login()
        return render.formimg(f)
```

```
class image:
    def GET(self):
        user_data = web.input()
        result_pic = func_pic(user_data.Searching)
        return render.result_search_pic(user_data.Searching, result_pic)
```

3、完善网页设计

3.1 设计 formimg.html

formimg.html 与 formtest.html 都是初始的待搜索界面，整体代码也很相似。

```
$def with(form)
<body bgcolor="white"></body>
<h1 align="center">My PIC Searching Engine</h1><br><br>

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

显示的界面很简洁：

3.2 设计 result_search_pic.html

我们希望将搜索结果以卡片式显示。为此在 style 中添加如下部分：

```
div.polaroid {
  max-width: 80%;
  background-color: white;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  margin-bottom: 25px;
}

div.container {
  text-align: center;
  padding: 10px 20px;
}
```

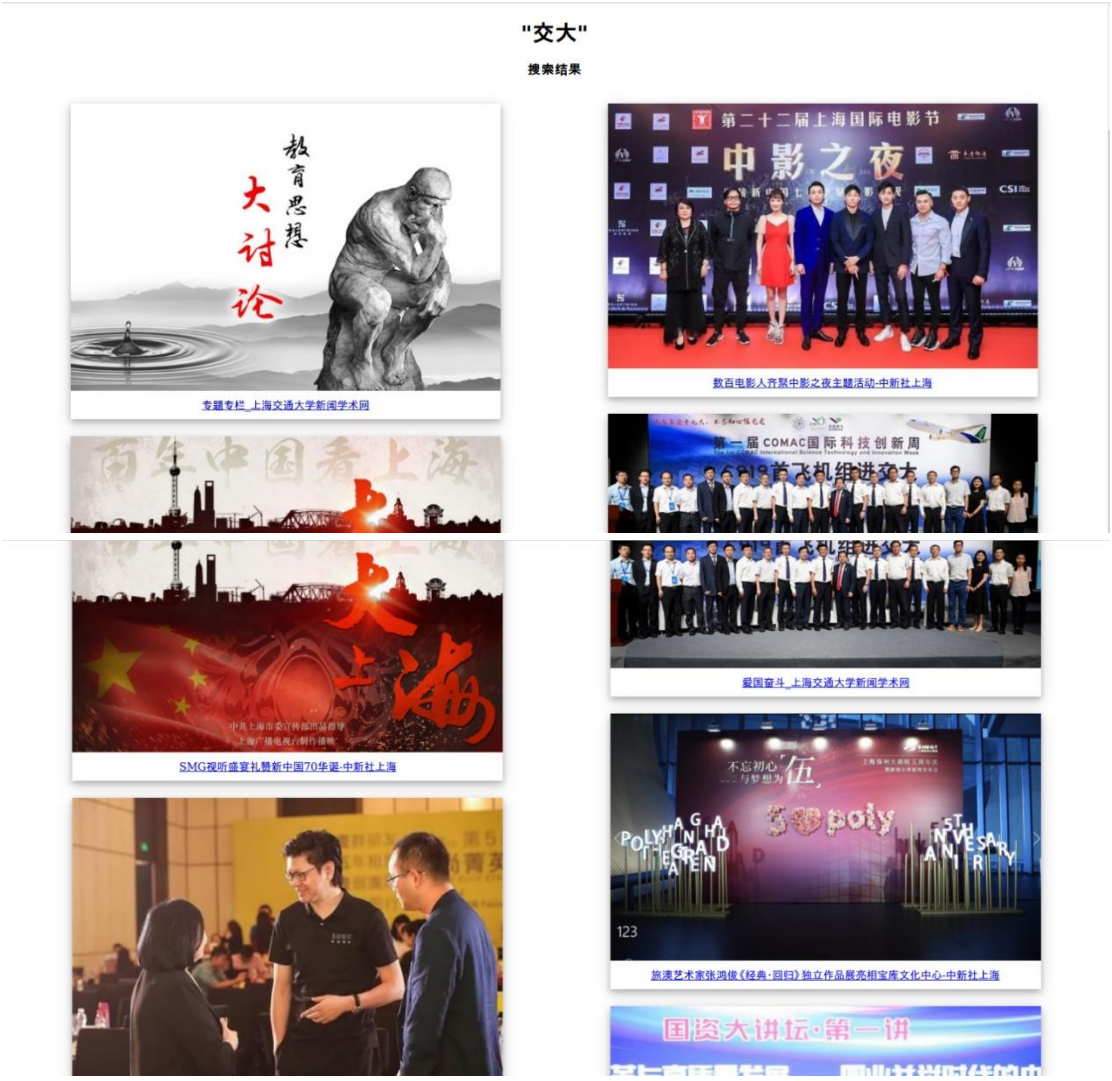
同时我们还希望最后的搜索结果能够以瀑布流的形式展出，图片大小并不绝对固定，形成两列错落有致的图片搜索结果，为此我们首先需要在 style 中添加：

```
div.waterfall-container {
    column-count: 2;
    column-gap: 0;
  }

div.waterfall-item {
    break-inside: avoid;
  }
```

再将显示搜索结果部分的代码修改如下：

```
<center>
<div class="waterfall-container">
$if name:
    $for i in name:
        <div class="waterfall-item" v-for="i in 100">
            <div class="polaroid">
                <img src="$i['imgurl']" />
                <div class="container">
                    <a href="$i['url']">$i['title']</a><br>
                </div>
            </div>
        </div>

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

我们搜索“交大”的图片，搜索结果如下，可以说是非常美观了：



3.3 修改 result_search_txt.py

为了将风格统一，我们将文本搜索部分的结果也以卡片式显示，稍加修改即可：

```
$if name:
    $for i in name:
        <div class="polaroid">
                <div class="container">
                    <a href="$i['url']">$i['title']</a><br>
                    <em>$:i['keyword']</em><br>
                    <em>$i['url']</em><br>
                </div>
        </div>


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

搜索后结果如下：

三、实验总结

1、实验概述

本次实验的主要任务，可以总结为实现图片与文字的搜索引擎，内容 CSS 与界面美化，是对前期学习内容的总结，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 学会了图片搜索引擎的创建

2.2 学会了图片搜索结果的瀑布流显示

2.3 统一了页面设计风格，学习了 div+CSS 的使用

3、创新点

本次实验中我主要进行的创新点是优化了图片搜索结果界面，设计了瀑布流卡片式图片输出界面，极大的提升了美观程度。

其中核心代码如下：

```
div.polaroid {
  max-width: 80%;
  background-color: white;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  margin-bottom: 25px;
}

div.container {
  text-align: center;
  padding: 10px 20px;
}
```

```
div.waterfall-container {
    column-count: 2;
    column-gap: 0;
  }

div.waterfall-item {
    break-inside: avoid;
  }
```

搜索界面的十分美观，截图如下：

SMG视听盛宴礼赞新中国70华诞-中新社上海

爱国奋斗_上海交通大学新闻学术网

旅澳艺术家张鸿俊《经典·回归》独立作品展亮相宝库文化中心-中新社上海

4、遇到的问题

为了让瀑布流卡片式输出图片的时候，能够让图片自适应大小，我们可以先设计好卡片和 container 的大小，再通过设定图片的宽与高来得到合适的结果。

```
img {
    width: 100%;
    height: auto;
    margin:0;
    padding:0;
}
div.polaroid {
  max-width: 80%;
  background-color: white;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  margin-bottom: 25px;
}

div.container {
  text-align: center;
  padding: 10px 20px;
}
```