林禹臣 5140309507

November 18, 2015（Week 7 & 8）

电类工程导论C 实验报告7&8

——web.py中期整合

目录页

1. 亮点、特殊功能（Features）

这个section 是为了助教更快地发现我项目的亮点的，具体的实现方法在7.实验扩展中介绍，这里只介绍功能。

● 扁平化设计、更好的用户体验

我的整个页面配色都是用的清新系列的配色，给用户更简洁的使用体验。首页采用了背景图片的形式，清新古雅。网页搜索结果页面采用了二分栏的模式显示结果，达到了最大化空间利用率的效果。每一项搜索结果的 abstract都使用了 box 的设计模式来展现，更加清晰有效。

● 自动纠错、解决了误输入的问题（需要网络）

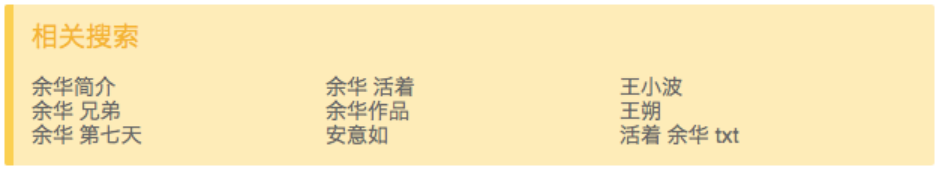
当用户输入了错别字，或者有些字只知道拼音不知道是哪个具体的字的时候，我都可以给出准确的搜索建议。

比如当一个用户输入了错别字，把『战争』打成了『 站争』，我的引擎会给出自动纠错后的搜索建议。

再比如，用户听到别人谈论一个作家叫做yu'hua，其实是指余华，但是用户只知道拼音，并不知道具体是哪两个字的时候，直接输入拼音，我的引擎也可以给出搜索建议。

●相关搜索（需要网络）

在向用户展示搜索结果之后，我们还在页面的最右下角给出了大量网络用户常常搜索的相关词汇。比如搜索余华之后，我们可以看到在右下角有：



点击之后即可查询相关内容。方便了用户需要继续进行相关搜索的需求。

2. 引言（Introduction）

在这两次实验中，我实现了将整个后台的搜索过程 BS 化（ Browser-Sever）。利用了 web.py 库，结合所有之前的实验实现了一个简易的搜索引擎网站。

3. 实验环境 （Testbed）

python 2.7.10（32bit）+ jdk 1.6 (32bit VM) + Sublime Text 3 + Mac OS X 10.11.1

JCC 2.21 + Lucene 3.6.2 + BeautifulSoup 4.4.1 + jieba 0.37 + web.py + Safari/Chrome

（上周提交的作业是在 lucene4下完成的，提交后觉得助教测试不很方便，特意又补充了一个3.6.2的版本以供测试，这次直接交3.6.2下的版本。）

4. 预备知识 （Propaedeutics）

- web.py 的基本使用方法
- http 服务的基本知识
- CSS-DIV 设计模式的基本知识

5. 实验过程（Experiment）

5.1.Web.py架构构建

Web.py 是最轻量级的 Python 的web 服务框架。正如它官网首页所描述的，『Django是让你在 Django 里面写网站，但是 web.py 是让你在 python 中写网站。』原因主要是在 templates 的 html 中我们可以几乎完全用 python 的语法来控制输出，和 php，asp 的思想完全一致。

我的 url 处理器只有 index 和 s。其中 index 负责首页的转发，s 接收了图片和文字搜索的提交。根据 get 请求中的 typ 变量来区分是图片搜索还是网页搜索，然后分别给予转发应答。

```
urls= (
    '/','index',
    '/s','s'
)
```

在main 函数中，我们首先对SearchFiles_webVersion和SearchImages_webVersion调用 init 函数来初始化一些变量，比如 vm，这里是为了避免多次初始化 vm 遍历的一个操作。

接下来就是 s 类中的 GET 函数。

```
class s:
    def GET(self):

        global old_keyword,old_res_list
        user_data= web.input()

        if(SearchFiles_webVersion.vm_env==None):
            SearchFiles_webVersion.init()

        SearchFiles_webVersion.vm_env.attachCurrentThread()

        keyword = user_data['keyword']
        typ = 0 # 0 is text and 1 is image
        if(user_data.has_key('typ')):
            typ = 0 + (user_data['typ']=='Search Images')
        print typ
        p = 1
        if(user_data.has_key('p')):
            p = int(user_data['p'])
        if(keyword=='' or keyword==None):
            return render.formtest()
        if(typ==0):
            res_list,total = SearchFiles_webVersion.run(keyword,p,10)
            return render.result(keyword,res_list,p,10,total)
        else:
            res_list,total = SearchImages_webVersion.run(keyword,p,60)
            return render.img_res(keyword,res_list,p,60,total)
```

首先对用户请求的 typ进行判断，看是图片搜索还是网站搜索，然后通过判断是第几页（p），如果没有 p，则默认是1。如果没有输入任何搜索 keyword，则跳回首页。

接着分别对网页搜索和图片搜索分别传递参数，回调，调用 templates 的显示。

5.2.Search.py 的修改 & Highlighter引入

在原来版本的基础上，我们需要对 Search 的 py 文件进行修改。首先要把vm 设置为全局变量，防止多次 initVM 从而导致报错。以SearchFiles_webVersion为例。

```
def init():
    global STORE_DIR,directory,searcher,analyzer,vm_env
    STORE_DIR = "index_lucene_v3_highlight"
    if(vm_env==None):
        vm_env = initVM(vmargs=['-Djava.awt.headless=true'])
    print 'lucene', VERSION
    directory = SimpleFSDirectory(File(STORE_DIR))
    searcher = IndexSearcher(directory, True)
    analyzer = WhitespaceAnalyzer(Version.LUCENE_CURRENT)
```

这里在 initVM 之前需要对 vm 是否是 None 进行判断。

另外一个非常重要的就是对 run 函数改革，我们要加入 highlighter 来实现提取 abstract。

```
scoreDocs = searcher.search(querys, 4000).scoreDocs
print "%s total matching documents." % len(scoreDocs)

res_list = []
simpleHTMLFormatter = SimpleHTMLFormatter("<font_forblank_color='red'>", "</font>")

queryToHigh = QueryParser(Version.LUCENE_CURRENT,"contents",analyzer).parse(command_dict['contents'])

hlter = Highlighter(simpleHTMLFormatter,QueryScorer(queryToHigh))
hlter.setTextFragmenter(SimpleFragmenter(200))
start = (pageindex-1)*pagesize
end = start+pagesize
for scoreDoc in scoreDocs[start:end+1]:
    doc = searcher.doc(scoreDoc.doc)
    res = []
    res.append(doc.get('title'))
    res.append(doc.get('url'))
    output = hlter.getBestFragment(analyzer,"contents",clear(doc.get('contents')))
    res.append(output)
    res_list.append(res)
return res_list,len(scoreDocs)
```

可以看到首先建立一个 simpleHTMLFormatter 来对关键字进行高亮，在之前之后分别加入<font>标签，注意到这里我们利用了'_forblank_'来代替空格，这是因为我们之后要对所

有的文本空格进行删除，如果这里直接用空格则会被删去，这里利用替代符，可以在之后替换成空格，从而保证了标签的正确。

第二个值得提的是，我们要根据 run 函数的参数来初始化我们需要遍历的范围，这样我们可以减少提取 abstract 的工作量，加快页面的初始化速度。

highlighter 的核心语句就是 getBestFragment 这一句。这里需要注意的是我们要取出 contents。而我们原来的 contents 并没有存储在索引中，所以这就需要我们重新建立索引，建立索引的时候只要把Store 的属性改成 YES 即可。

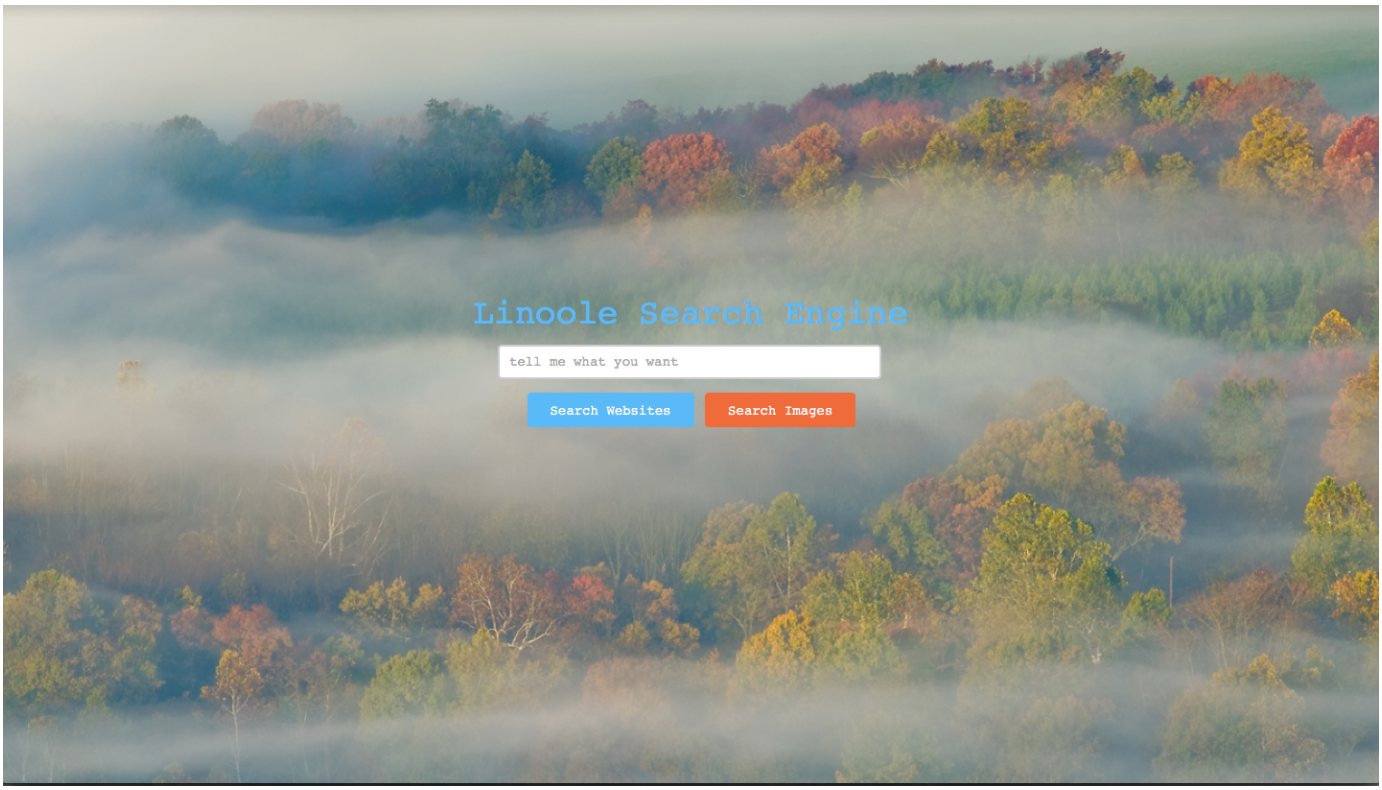
```
doc.add(Field("contents", contents,
                          Field.Store.YES,
                          Field.Index.ANALYZED))
```

（对我个人来说，比较悲催的是，我之前为了增强索引的效果，把停用词和标点符号删去了，这样的话就没有办法在摘要中显示正确的信息，所以还需要重新写一下分词的模块，直接在清理完 HTML代码之后直接分词，然后再建立索引。）

5.3.templates 的构建 & CSS设计

这两个事情很难分开讲，所以直接在一起写了。

4.3.1. 首页展示



对首页我们要讲述的最重要的就是对 form 的控制，我们有两个submit 按钮，一般来说，一个 form 只有一个 submit，但是当有两个同样 name 的submit 时，我们可以根据他们的 value 来控制我们的方向（结合 webprocess.py 中的处理）。之所以选择这种方式，而不是用两个 button，主要是为了让用户可以在输入完查询命令之后直接按回车即可提交表单，增强用户体验。

```
<input type="submit" class="button button-blue"
    style="font-family:courier" value="Search Websites" name="typ"
<input type="submit" class="button button-orange"
    style="font-family:courier" value="Search Images" name="typ"/>
```

关于首页的 CSS 我们主要是制定了按钮的背景颜色和边框处理，还有就是输入框颜色的边界的处理。

4.3.2. 网站搜索页面

这里可以看到，我为了充分利用空间和提供更好的用户体验采用了分栏式的显示方式，并且将摘要以卡片的形式进行展示，更加美观。

在页面的最上方，是一个和首页完全一样功能的 form，直接回车的话会继续进行网站搜索。

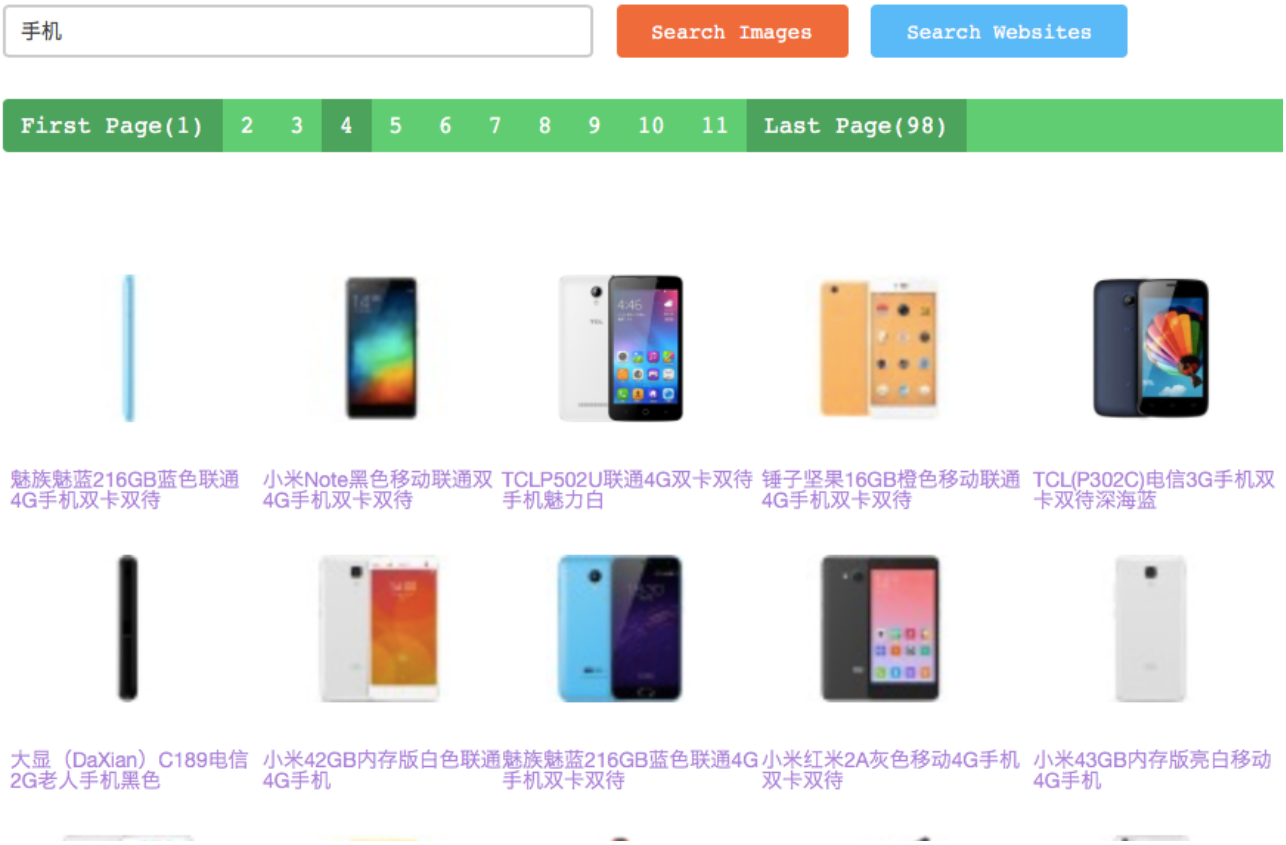
另外一点就是导航栏，我将导航栏直接用于了页面索引的控制。

比如当我们选择第15页的时候，左面会显示最近的三页，右面会显示最近的7页。最左边和最右边分别是第一页和最后一页。增强了用户体验。

因为结合 html 的 python 代码结合在一起的缩进比较繁乱，具体实现的代码不在这里解释了。其实逻辑还是很简单的。

需要注意的是，非转义化字符的处理，是利用$:的输出而不是$。

4.3.3. 图片搜索页面



最上面也是和网站搜索页面一样的 form，但是比较巧妙的是这里我调换了两个 submit 的顺序，从而让用户如果直接输入回车的话，会继续进行图片搜索。

6. 遇到的困难和解决方案 （Problems & Solutions）

遇到的第一个困难就是如何实现摘要的提取，一开始我想自己写一个算法来根据输入的查询，先找到对应的文件，从而提取文本内容，进行分段匹配，找到含有关键字最

多的那一段从而显示，但是这样有一个问题就是多个关键词时会比较复杂，而且代码量较大。后来通过搜索找到了 lucene 自带的highlighter 工具，并且向其他的同学们推荐了这个工具。网上的样例代码基本都是java 的，不过还是非常容易的被我移植到了 python 里。

7. 实验扩展（Extensions）

这个实验我做的拓展性工作，主要集中在如何提高用户体验上，比如上面我提到的几个细节：用户输入回车之后最想要的操作是什么，如何美观的展示结果，如何充分利用空间并且让用户可以高效的浏览信息，如何通过对Search文件逻辑上的优化来加速页面读取速度，这些都是我考虑了的问题。

针对在第一节中展示的三个特殊功能，具体实现方案如下：

1.扁平化界面，这个地方我用到了一些开源的 CSS 模板的配色思想，配合了我自己的配色方案（在参考地址中有）。主要集中在搜索结果页面的优化。这个在之前的说明里已经提到过了。

2.**自动纠错功能**，这里本来我是想找到一个 python 的库来完成这个功能的，发现并没有相关的库，这也给了我一个契机，想要自己写一个库来完成这个功能。查了很多资料，比如朴素贝叶斯方法之类的，虽然思路比较清晰，但是时间有限，所以只能委曲求全，先暂时是使用了百度的搜索提示结果来完成的这个功能，具体代码在 tools.py的checkFuzz 函数里。

3.相关搜索的实现方案同上，用的是同一个函数，节省了时间。

8. 总结（Conclusion）

在此次实验中，我初步掌握了利用 python 来编写网站的基本步骤和实现方法，并且将之前所有的工作做了一个总结，让我的搜索引擎真正的可以面向了一般的用户，脱离了丑陋的控制台，在浏览器中展现出来。

9. 参考（References）

http://blog.csdn.net/javaman_chen/article/details/8224407

http://blog.csdn.net/java2king/article/details/4407998

http://tool.c7sky.com/webcolor/#character_1

衷心感谢助教们和老师付出的精力。