实验5 文本特征抽取和索引2

518030910303 纪喆

2019 年 10 月 23 日

1 实验准备

1.1 实验环境

本次实验在 **Ubuntu 14.04 LTS**系统中进行，使用的是**Python3**编程语言。本次实验使用的工具有：vim, Python3, Python2, pylucene 本实验使用了实验4保留的网页。但为代码完整性，本次提交包括了爬虫所需文件。使用Python3的脚本有：

1. Bitarray.py
2. BloomFilter.py
3. exploit.py
4. quanjing.py

使用Python2的脚本有：

1. indexfile.py
2. Indeximg.py
3. searchhtml.py
4. Searchimg.py

1.2 实验目的

1. 补充搜索引擎的功能，实现“site:****”
2. 实现图片搜索，即，输入文本，返回相关的图片链接、图片所在的网页，以及该网页的标题。

1.3 实验原理

本次实验的网页依然是实验4所爬取的网页，因此，关于爬虫的内容在此不做赘述。本模块主要介绍搜索新功能以及图片爬取加索引的主要原理。注：下载的网页均已转为utf8编码格式。

1.3.1 HTML文件索引重建

新增Field：site 在建立索引时，为了达到site的限定功能，需要对每一个doc添加site这一字段，并且，由于site的特殊性，本字段不需要解析器解析，而应该以原本的字符串直接进行添加。庆幸的是，进行实验4时，为了由更好的时间效率，爬取过程中已经在每一个网页文件开头注明了本网页的URL。通过python2中的urlparse解析URL，分析出域名，作为site的值，进行添加。其他内容与原来的建立过程基本相同。

建立过程 从文件中提取该网页的URL以及网页的title，借助urlparse函数解析site，再加上整个文件内容，文件名称，文件目录，一起存到一个doc里面。即，每一个doc里面有 6个Field。其中特殊的Field有两个：

title 该字段依旧是主要参与搜索的字段，本字段由结巴解析后用空格分隔传入lucene，后者将根据空格分词解析后添加。

site 该字段是本次实验新添加的搜索字段，本字段不被解析，以字符串的形式直接添加，在搜索时处理也比较特别。

选title字段作为主要搜索对象依旧是因为它以较少的字符描述了比较准确的信息，防止网页内容混乱导致搜索结果混乱。site因为其特殊性，虽然参与搜素但是不进行解析，如果被解析，site的限定功能会被大幅削弱。例如：限定site:www.baidu.com时，实验证明site字段里包含www和com的doc都有可能被返回。

1.3.2 HTML搜索实现

本次的搜索代码基本上是基于实验4的改造。主要改进内容包括搜素输入文本的解析以及布尔搜索对象的建立过程。

在输入文本解析时，先用空格将文本分割。分别判断每一个片段是否含有冒号。如果含有，检测冒号前的字符串是否为允许的限定内容。如果是，

就加入命令集合，如果不是，则把整体作为title的搜索文本添加。最终，函数返回命令字典。通过该字典建立布尔搜索对象，进行下一步工作。

1.3.3 IMG的获取与标记

获取图片自然不是难事，但如何给图片加标记较为困难。为了更加精准地为图片加上标签，本次实验选择特定地爬取一个图片库（全景网）内的部分内容。由于在同一个图片库网站中，图片的网页结构基本相同，因此可以定向地寻找图片标记。具体内容见实验过程。

1.3.4 IMG索引建立

本索引建立相对HTML文档的索引来说比较简单，只需要添加img的地址，网页url，网页标题，以及img的标签。其中标签这一字段为多值字段，因为同一张照片可以有许多描述它的标签。每一个doc中标签的具体数量由该图片所在网页里面的标签数确定。

2 实验过程

2.1 HTML索引建立

与实验4相同，本次实验建立索引时依然使用结巴（jieba）分词工具对网页标题进行解析，然后在lucene中使用空格解析器解析。本次建立的索引中由三种不同类型的字段。一个专门用于存储site数据。因为它需要被索引但是不需要英文解析。第二个类型字段用于存储title，是默认搜索对象，也是被索引的，但是需要中文解析。最后一种则是用来储存包括网页URL在内的非索引信息。三种类型定义如下。（其中第三种类型是在添加site是临时建立的。）

```
        t1 = FieldType()
        t1.setIndexed(True)
        t1.setStored(True)
        t1.setTokenized(False)
        t1.setIndexOptions(FieldInfo.IndexOptions.
                                        DOCS_AND_FREQS)

        t2 = FieldType()
        t2.setIndexed(True)
```

```
        t2.setStored(False)
        t2.setTokenized(True)
        t2.setIndexOptions(FieldInfo.
                IndexOptions.DOCS_AND_FREQS_AND_POSITIONS)
```

```
        doc.add(Field("site", netloc, Field.Store.YES, Field.
                                        Index.NOT_ANALYZED))
```

对于如何遍历文件夹，本实验借用了示例代码中的做法，即os.walk：

```
    for root, dirnames, filenames in os.walk(root):
        for filename in filenames:
            print "adding", filename
```

其中 print "adding", filename 为的是在程序正常运行时有一定的输出内容，避免无法区分程序是否卡死。

2.2 HTML搜索实现

由于本实验添加了site限定功能，因此解析命令过程变得复杂起来。首先通过空格把命令分为若干个部分，对每一部分判断是否存在冒号，如果有，再判断是否为程序所允许的限定词，即：site和title。如果不是，或者不含冒号，则直接对title字段进行该内容的搜索。也就是说，title字段是本搜索功能的默认搜索字段。一切无法匹配的限定字段都将被视为对title的搜索。然后，将命令内容以字典的形式返回，这便是命令解析函数的工作流程，其主要内容如下。

```
    opt = 'title'
    for i in command.split(' '):
        if ':' in i:
            opt, value = i.split(':')[:2]
            opt = opt.lower()
            if opt in allowed_opt and value != '':
                command_dict[opt] = command_dict.get(opt, '') +
                                                    ' ' + value
            else:
                opt = 'title'
                command_dict[opt] = command_dict.get(opt, '') +
                                                    ' ' + i
        else:
            opt = 'title'
```

```
            command_dict[opt] = command_dict.get(opt, '') + ' '
                                                + i
    return command_dict
```

命令解析过程结束后，即可创建布尔查询对象了。对函数法返回的字典进行遍历，如果是title字段，因为需要中文的额外分词，因此需要加以额外判断。具体如下。

```
    command_dict = parseCommand(command)
    querys = BooleanQuery()
    for k,v in command_dict.iteritems():
        if k == 'title':
            v= ' '.join(jieba.cut(v))
        query = QueryParser(Version.LUCENE_CURRENT, k,
                            analyzer).parse(v)
        querys.add(query, BooleanClause.Occur.MUST)
    scoreDocs = searcher.search(querys, 50).scoreDocs
```

文件其他内容与实验四基本一致，在此不做赘述。最终运行结果如下：

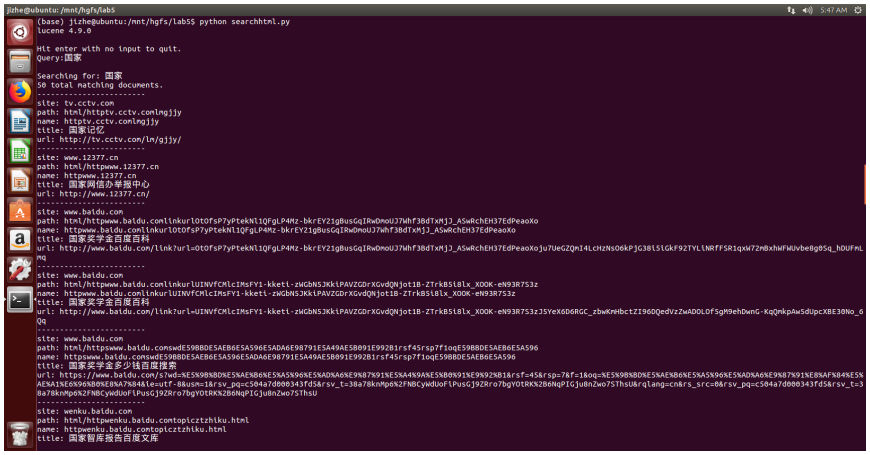

图 1: 不限定site的结果示例

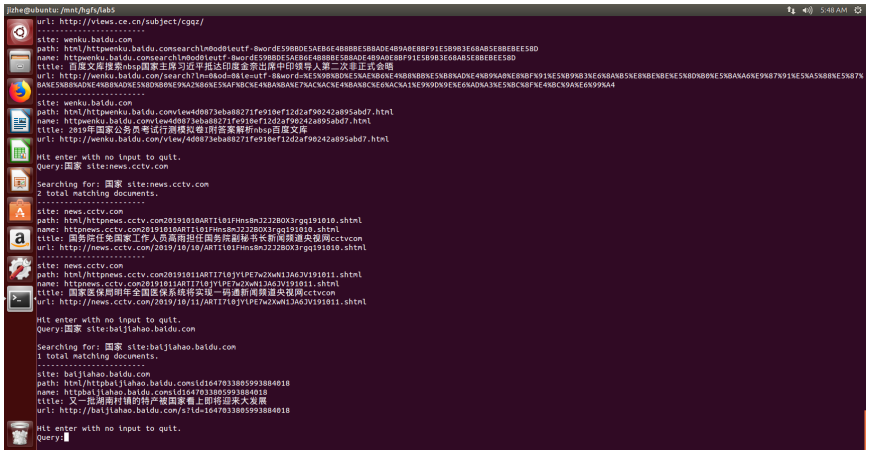

图 2: 限定site的结果示例（多次结果同屏展示）

2.3 IMG的获取

为了高准确性的找到图片以及其对应的标签，本次实验专门爬取全景网（一在线图库）的图片。为了便于说明，下面给出全景网两个图片网页的URL以及部分HTML代码。

https://www.quanjing.com/imgbuy/QJ6216493248.html

https://www.quanjing.com/imgbuy/QJ6590582508.html

```
<div class="imginfo_bigpic" id="imginfo_bigpic">
    <img id="Img1" src="https://dpic.tiankong.com/sd/i0/
       QJ6216493248.jpg?x-oss-process=style/794ws">
</div>
```

```
<ul id="Ul1">
<li><a href="...">...</a></li>
<li><a href="...">...</a></li>
<li><a href="...">...</a></li>
<li><a href="...">...</a></li>
...
</ul>
```

爆破方式获取大量图片地址 上面提供的两个URL均为全景网图片的网页，可以发现URL中只有最后的编码不同，因此，我们可以用程序从一个照片的真实编号开始逐一向上递增寻找编码大于该图片的其他图片网页。实现

该功能的程序文件为：exploit.py。该文件利用requests库中的get，在不允许重定向的前提下，将图片编号依次加一后访问，判断状态码是否为200。如果是，则则说明该图片编号正确，将其记录于img_page_urls.txt中以便后续步骤使用。实现禁止重定向以及延缓下载的get函数使用如下：

```
r = requests.head("https://www.quanjing.com/imgbuy/QJ"+str(
                                    temp)+".html",
                        headers={'User-Agent':'Chrome/1.0.0'},
                        stream = True,
                        allow_redirects=False)
```

解析网页，提取并记录所需信息 根据exploit.py所获得的图片网页URL，依次获取网页。根据网页HTML文件的结构特点，即：主图片的id为固定的字符串“Img1”，以及其标签具有固定的id为“Ul1”，可以精准地定位他们并提取，提取代码如下：

```
soup = bs4.BeautifulSoup(r.text, 'html.parser')
img_src = soup.find(id = "Img1").get('src').strip()
img_src = img_src[:img_src.index('?')]
tit = soup.find('title').string.strip().replace(' ','')
tags = soup.find(id = 'Ul1')
tags_list = list()
if tags:
    tags_list = tags.find_all('a')
    for i in range(len(tags_list)):
        tags_list[i] = tags_list[i].string.strip()
```

由此，图片地址以及图片标签就提取完成了。理论上可以在同一个程序下继续建构索引，但是为了保险起见，本程序（文件名：quanjing.py）的主要功能就此结束，将结果全部记录在了文件complete.txt中。其优点在于，如果想要扩充图片库，可以很轻松地知道上次爆破的终点编号是多少，从而可以紧接着上一次的结果继续向前推进，做到不重叠，不遗漏。

IMG索引建立 IMG的索引包括网页URL，网页标题，图片地址，图片标签四个。其中图片标签较为特殊，需要作为索引，而且是多值的。相对于实验的上一部分，本索引较为简单，即从文件complete.txt文件中获取所需信息，然后逐一添加。另外，为了避免文件（夹）名称冲突， IMG的索引文件夹被命名为index_img，相应的，HTML文件的索引文件夹被命名

为index_html。

IMG的搜索 本搜索功能比较简单，就是针对图片的标签字段进行搜索，与实验4中的搜索无太大差异。运行结果如下：

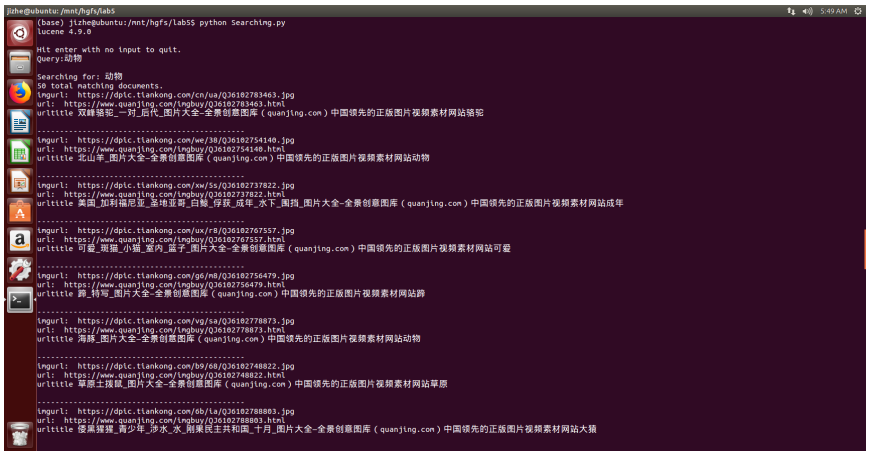

图 3: 图片搜索结果示例

3 实验总结

3.1 概述

本次实验依旧使用Lucene，文本搜索功能比实验四有了较大的进步，同时也实现了图片搜索。搜索的功能更加完善。

3.2 问题与解决

问题 site限定不牢。即，会有不符合site限定的搜索结果出现。

解决 仔细分析发现，该问题的原因在于在建立索引时，site字段被默认英文解析了，因此，只要site中有部分匹配，包括www和com等，搜索引擎便认为这是符合限定的。只要将site字段的ANALYZED关闭即可。

3.3 感想

经过本实验，自己搭建的搜索引擎功能更加丰富，搜索更加有力。引擎的功能总是在不断地发展，其发展的背后是代码的不停修正以及更多知

识内容的添加。也许这是关于搜索引擎内核本身的最后一次实验，虽然此内核还不算完美，但在现有的知识条件下也算得上是一个差不多的工具了。期待下一次实验对此的应用。