林禹臣 5140309507

September 20, 2015

电类工程导论C 实验报告1

——parser

目录页

1. 引言（Introduction）

在电工导C的第一次实验中，我主要完成了三个练习。这三个练习是关于如何使用 BeautifulSoup 这个库来实现对 HTML 源代码的解析和相关操作的。在实验中遇到了一些在 python 2.7 和 BeautifulSoup3.2.1 下才出现的困难，通过搜索，学习了相关知识，最终使问题得以解决，这也是在这次实验中我感到比较自豪的一点。

衷心感谢张娅老师，姚助教和王助教的耐心指导和辛勤付出。

2. 实验环境 （Testbed）

最初的编程环境是 OS X 10.10.5 + python 2.7.10 （narrow） + BeautifulSoup 3.2.1 之后遇到了一些bug（在第5节会详细提到）之后，所以最终选择了

python 2.7.10（narrow） + BeautifulSoup4

作为实验环境，并且实验结果与操作系统无关。

3. 预备知识 （Propaedeutics）

- HTML 的基本知识，比如标签的构成，标签的嵌套规则。
- BeautifulSoup 的基本用法，比如 findAll 函数，find 函数，以及这些函数的参数的规则。另外就是属性的获得，get 函数，还有 contents[i] 的意义等。
- 正则表达式的基本用法。 学会了一些基础的匹配方法，比如^，d，w等等符号的替代意义。
- Python 内置类型的用法复习，比如dict类型的存取，数组下标的特殊用法等等。

4. 实验过程（Experiment）

实验分为三个部分，下面分别进行报告说明。

4.1.练习1 （ Section 1 ）

要求：是对任意一个网页，提取出所有的 url ，存储在一个 set 中，并最后以一行一个的格式输出到 res1.txt 文件中。

过程：

首先，通过 urllib2 的 urlopen 函数发送 http 请求，并且通过 read 函数对获得源代码，用其创建一个 BeautifulSoup 类的对象，命名为 bs。

```
import urllib2
from BeautifulSoup import BeautifulSoup

content = urllib2.urlopen('http://www.cnblogs.com/yuchenlin/').read()

bs = BeautifulSoup(content)
```

然后，创建一个函数来处理这个核心问题。先建立了一个set，然后再对findall的函数返回值进行for循环遍历，分别对每一个a标签进行 href 属性的提取。 并且将这个 url add到set中去。

```
def getAllUrl(ctnt):
    urlset = set()
    for i in ctnt.findAll('a'):
        str = i.get('href')
        if(str!=None):
            if(str[-1]=='/' and len(str)>2):
                str = str[:-1]
            if(str[:2]=='//'):
                urlset.add('http:'+str)
            elif(str[:2]=='ja' or str[:2]=='/'):
                continue
            else:
                urlset.add(str)
    return  urlset
```

这里有几个细节值得注意。一只有非空的url才值得处理。二是有一些不规范的链接，比如“//www.baidu.com”这种情况，我试了一下在浏览器手动输入这个链接会主动转到file://www.baidu.com，所以我在这里做了一个特殊判断的处理，如果出现了这种情况，就把它手动加上http头。还有一个很重要的就是，根据张娅老师在课上讲的，为了后续能够更好的实现url判重，要把所有的url规范化，不能有的结尾有\，有的没有\，所以我加了一个小的判断处理。

最后，我们要处理的就是把这个url的set遍历一遍，输出到一个文件中。我比较倾向于先把要输出内容组成一个大字符串strText，然后统一输出到文件里。

```
urlset =  getAllUrl(bs)

f = file("res1.txt","w")
strText = ""
for i in urlset:
    strText += (i + "\n")
f.write(strText)
f.close()
```

4.2.练习2 （ Section 2 ）

练习2 的要求和练习1类似，只不过把“提取所有的网页”改成了所有的图片的地址。所以大致思路一致，只需要微微修改函数中几行代码即可。

```
def getAllIMG(ctnt):
    imgSet = set()
    for i in ctnt.findAll('img'):
        str = i.get('src')
        if(str!=None):
            imgSet.add(str)
    return  imgSet
```

4.3.练习3 （ Section 3 ）

练习3本身难度不大，但是遇到了一些困难。（在下一节详细说明，此处暂时略过。）

首先，我们要改变发送http请求的方式，这是因为糗事百科网站的特殊性，我们不能以默认的urlopen 参数来进行提交，必须手动指定我们的user-agent 为custom-agent 才可以。

```
import urllib2
req = urllib2.Request("http://www.qiushibaike.com/pic", None, {'User-agent' : 'Custom User Agent'})
content = urllib2.urlopen(req).read()
```

接下来，我创建了一个函数用来返回所要求的docs和nextPage。

• nextPage 的获取很简单，通过对网页的分析，我们可以发现这个a标签有一个特征就是 class 值是 next ，所以我们可以利用这个特征直接找到它。另外由于这个href值是一个相对路径，我们要手动加上之前的http 头和顶级域名。

```
<a class="next" href="/pic/page/2?s=4809109">下一页</a>
```

```
constTitle = "http://www.qiushibaike.com"
nextPage = constTitle + bs.find('a',{'class':'next'}).get('href')
```

这里我们用的是find 函数，加上一个class属性的限制即可找到我们想要的标签。如果当前页面存在多个，或者不存在这样的标签，这一步会报错，但是由于页面的这个特定规律，这两种特殊情况我认为不在鲁棒性的考察范围内。

• docs的处理。docs本身是个dict，它的key是 qiushi_tag_后面的那个数字id，value是一个dict 包含两个key 一个是content，存储糗事的文字内容，一个是imgurl，存储的是这个糗事的图片的id。

首先，我们对网页的结构进行分析，发现每个糗事的结构是这样的。

```
▶<div class="article block untagged mb15" id="qiushi_tag_112978227">…</div>
▼<div class="article block untagged mb15" id="qiushi_tag_112978631">
  ▶<div class="author">…</div>
  ▼<div class="content">
     " 精致的流水线条，古朴的色调，奥坎 随形的圆盘，复古的味道造就一张精美的圆桌茶盘！尺寸65~60*6厚。 "
    <!--1442756524-->
   </div>
  ▼<div class="thumb">
    ▼<a href="/article/112978631" target="_blank">
       <img src="http://pic.qiushibaike.com/system/pictures/11297/112978631/medium/app112978631.jpg" alt="精致的流水线条">
     </a>
   </div>
  ▶<div class="stats">…</div>
  ▶<div id="qiushi_counts_112978631" class="stats-buttons bar clearfix">…</div>
  ▶<div class="single-share">…</div>
   <div class="single-clear"></div>
 </div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978553">…</div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978489">…</div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978455">…</div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978564">…</div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978454">…</div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978713">…</div>
▶<div class="article block untagged mb15" id="qiushi_tag_112978450">…</div>
```

一个糗事是一个div， 他的特点是 class 是一个固定的值，id是一个流水号，符合qiushi_tag_\d+的正则表达式。

这个content div 内部有多个小div，我们主要关心两个，一个是红色框内部的以content 为class 的div，这个div的内容又包括两部分（分别是contens[0]和[1]）， [0]是糗事的文本，就是我们需要的那个， [1] 是注释，我们并不需要。

我们在看下面蓝色框内部的，thumb div，这个div存储的就是图片的链接了，所以我们先找到a标签，再从a标签里面找到img标签，然后获得它的src属性即可。

所以，我们的代码是这样的。

```
for qiu_item in bs.findAll('div',{'id':re.compile('^qiushi_tag_\d+')}):
    tag = str(qiu_item.get('id'))[11:]
    imgurl = qiu_item.find('div',{'class':'thumb'}).find('a').find('img').get('src')
    qiushi = qiu_item.find('div',{'class':'content'}).contents[0]#.decode('utf-8')
    docs[tag] ={"content" : qiushi ,"imgurl":imgurl}
```

注意几个细节，我们for循环的对象是findAll的返回值，这个地方我们利用了正则表达式来匹配所有的符合条件的div。

关于tag的获取，由于前面“qiushi_tag_”的长度是固定的11位，我们直接从11位截取到最后即可。这里利用了python的切片特性，直接[11:]表示从下标11到最后。

最后，我们要对这个函数的返回值做格式上的处理，还有编码上的改变。这里值得一提的是，我们必须在程序头指定默认的编码方式才能在最后的解码时不出现任何奇怪的错误（python2.7的严重缺陷，python3没有这个问题）。

```
import sys
reload(sys)
sys.setdefaultencoding("utf-8")
```

```
returnVals = getQiushibaikePage(content)
textToSave = ""
for i in returnVals[0]:
    textToSave +=  returnVals[0][i]['imgurl']+ "\t"
    textToSave +=  returnVals[0][i]['content'].decode('utf-8').strip() + "\n"
f = open("res3.txt","w")
f.write(textToSave)
f.close()
```

注意，在处理content的时候，我们要先decode成utf-8的才可以保证最后存储的时候是中文，另外可以利用strip 函数来除去文章中的回车 / 换行等字符。遇到的困难和解决方案（Problems & Solutions）

5. 遇到的困难和解决方案（Problems & Solutions）

之前一直提到的困难指的就是在BeautifulSoup 3.2.1 和 Python 2.7 的narrow版本下，这段代码会出现错误：

```
import urllib2
req = urllib2.Request("http://www.qiushibaike.com/pic", None, {'User-agent' : 'Custom User Agent'})
content = urllib2.urlopen(req).read()
from BeautifulSoup import  BeautifulSoup
bs = BeautifulSoup(content)
```

报错于最后一行，也就是放入BeautifulSoup时。错误信息如下：

```
ValueError: unichr() arg not in range(0x10000) (narrow Python build)
```

经过搜索，发现这个bug是由于python 的narrow Build 的unicode上限是65535，也就是所谓的ucs2；wide版本是 ucs4；

Python官网的2.7完整包，windows和OS X下都是narrow的，只有linux下编译安装时默认指定ucs4了。所以我就想到了一个方法，那就是在OS X下重新手动编译python2.7，这就有了方法1。

方案1： 手动编译python2.7 源代码，同时指定ucs＝4；

最重要的一步就是enanble ucs4； 其他步骤和正常的编译顺序一样即可。

```
curl -O https://www.python.org/ftp/python/2.7.10/Python-2.7.10.tgz
tar xf ./Python-2.7.10.tgz
cd ./Python-2.7.10
./configure --enable-unicode=ucs4 --prefix=/path/to/install
MACOSX_DEPLOYMENT_TARGET=10.10
make
make install
```

```
LinYuchens-Air:Python-2.7.10 Lin$ cd
LinYuchens-Air:~ Lin$ /path/to/install/bin/python2.7
Python 2.7.10 (default, Sep 16 2015, 23:03:08)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.maxunicode
1114111
```

这样虽然解决了 OS X的问题，但是没有windows下的相应的解决方案，windows下要想编译的话，也许需要借助Cygwin等工具才可以，然而会有很多新的麻烦产生。

方案2: 使用BeautifulSoup4进行解析

那么刚才我也提到了，这个bug所在的环境是BeautifulSoup 3.2.1 和 Python 2.7 的narrow造成的。我们既然不好从narrow下手，也许可以从BeautifulSoup的角度来看。所以我找到了最新版的BeautifulSoup，也就是bs4，来进行尝试。果不其然，只需要把原来的

from BeautifulSoup import BeautifulSoup 改成

from bs4 import BeautifulSoup

所有代码均可运行了。**这也是我最后采用的和向其他同学推荐的解决方案。**

其实还有方案3，那就是改成应用python3.x 来写，但是考虑到未来的实验也许有些模块必须用python2，为了避免麻烦，就没有改成3。

还有一个小困难，就是在不同系统下的换行符的不同。

windows下是\r\n

linux是\n

mac os x下是\r

python会在不同的系统下自动替换成不同的换行符，所以我用\n在mac os x下生成的txt，在os x下看是没问题的，但是在windows下是不行的。目前我想到的解决方案是利用\r\n来进行换行，这样就没有问题了。

6. 总结（Conclusion）

在这次实验中，收获最大之处就是自己寻找bug的根本原因的过程，从遇到问题，分析问题到解决问题经历了很多波折，最终解决时还是很开心很有成就感的。也通过这次实验进入了网络爬虫的大门，让我更有信心在接下来的实验中继续学习更有趣的知识。

7. 参考（References）

https://www.python.org/dev/peps/pep-0261/

http://wordaligned.org/articles/narrow-python/

最后，再次衷心感谢张娅老师和两位助教的耐心指导。

林禹臣

5140309507

9月20日，2015年