**第三周大作业说明文档**

**计71 张程远 2017011429**

**1、爬虫与新闻内容提取**

**1.1爬虫部分：**

爬虫的主要目的是获取带有新闻的网页链接以便之后提取其中内容。我选择爬取的页面是新华网。新华网的链接格式基本相同，只是按照分区内容不同其后缀不同，例如时政区<http://www.xinhuanet.com/politics/>以及法律区<http://www.xinhuanet.com/legal/>等。我选择的是上述两个分区。

由于新华网自身带有反爬虫机制，新华网会检测访问者是否带有浏览器的相关参数，爬虫直接爬取会报403错误，所以在代码中加入了如下的header，可以让爬虫伪装成浏览器访问页面。

代码：

headers = {'User-Agent':'Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN) AppleWebKit/523.15 (KHTML, like Gecko, Safari/419.3) Arora/0.3 (Change: 287 c9dfb30)",'}

req = urllib.request.Request(url=url,headers=headers)

s=urllib.request.urlopen(req).read()

s=s.decode("gbk","ignore") #读取的类型为byte，需要对读取的内容进行解码

爬虫过程使用的是BFS，使用列表存储网页链接。先获得每个页面的所有相关链接，如果链接不在列表里则存入列表，如果已经存入则忽视。之后每次再从列表中取出一个链接重复上述过程。为了防止爬取速度过快被服务器封禁IP，设置了每两个链接之间3秒的爬取间隔。链接的整体爬取过程还是非常顺畅的。

**1.2新闻内容提取：**

在爬取所有链接后，对每个链接进行访问，并提取其中的文本内容。这里我使用的是HTML. Parser库，这个库中的函数能够很方便地提取出标签之间的内容。例如设置start\_tag=p, end\_tag=p, 就可以让parser提取出所有<p>和</p>之间的内容。爬取下来的内容，我按照id，url，title，time，article（正文）五项存入了greatwork.db文件内。这里的数据库选用的是Python自带的轻量级数据库sqlite3。对每个链接依次访问并提取出正文的内容比较缓慢（主要是向新华网发送读取请求后新华网回应时快时慢，要间隔0-30秒不等的时间），所以我选择晚上进行爬取，整体速度相比于白天稍有提升，爬取1580条数据约用了半小时的时间。提取的代码如下：

cursor = c.execute("SELECT id,url from htmllist where id>500")

for row in cursor:

parser=myparser() #除了init以外最多只能设置两个参数，这里为flag和text

article=""

id=row[0]

url=row[1]

s=get\_page(url)

parser.feed(s)

parser.close()

title=parser.text[0]

if len(parser2.text)==0:

time0="/" #有些新闻稿没有时间，要赋一个空值

else:

time0=parser2.text[0]

for index in range(len(parser3.text)):

article+=parser3.text[index]

c2.execute("insert into urllist(id,url,title,time,article)\

VALUES(?,?,?,?,?)",(id,url,title,time0,article))

**1.3分词：**

分词直接使用了jieba库，采用了精确模式分词，并引入了停止词算法，去除了可能带来大量无用信息的一些停止词。数据存储方式是存入一个数据库dictionary，其内部是词汇+ID字符串的结构，字符串用逗号分隔每个ID（如2,3,45）。分好词后首先将每个词都存入一个列表当中，如果该词已经在列表当中则意味着也已经存入了数据库，直接update这个词所对应的ID字符串；否则为数据库新建一条记录，并加入该词和目前分词的网页ID。

for worditem in wordlist:

if worditem not in splited:

splited.append(worditem)

dic.execute("insert into dic(word,idlist)\

VALUES(?,?)",(worditem,id\_str))

else:

cursor=dic.execute("SELECT word,idlist from dic where word='%s'"%worditem)

for row in cursor:

wordtmp=row[1]+id\_str

dic.execute("update dic set idlist='%s' where word='%s'"%(wordtmp,worditem))

**2、新闻展示功能**：

**2.1界面**

新闻展示页面包括首页和搜索结果页两部分。首页最上方是标题“新华网新闻搜索”以及“新闻总量：1580条”的信息。在信息下方提供了一个表单，用户可以键入搜索关键词并点击搜索以获取相关的内容。首页下方展示的是数据库内所有新闻的罗列，包括标题、时间以及三行左右的内容展示。由于我在提取文段时没有保留原来的格式，所以我显示内容的算法是这样的：首页展示article[0:0+150+ranint(10,100)]，如果末尾没有到达文章结尾则加一个省略号，到达结尾则不加；结果页展示找到正文中关键词出现的第一处位置flag，如果flag!=-1将原来的0换成flag，其他算法不变。

**2.2 Web服务端及搜索算法：**

使用django新建了一个project名为searchengine，并新建了templates文件夹，将新闻提取时建立的页面以及首页全部放入templates文件夹，并修改url以及settings的设置。另外，在页面提交表单时使用的是POST方法，用于生成结果页面的search.py接收到post信号后执行生成结果页面的函数。搜索算法比较朴素，直接获取输入的文本内容后链接数据库dictionary，在dictionary里寻找与输入相同的词并获取它所属的页面ID字符串。将字符串读入并获取所有ID，并按照首页的方式展示所有搜索结果。搜索的时间与页面总量关系不大，只与搜索到的页面数量有关，搜索到1000条结果的耗时在0.1秒左右。

**2.3 分页以及美化：**

使用了static文件夹，在里面加入了网上现成的css模板bulma以及javascript分页插件，并在settings里的末尾以及HTML里加入了文件的所在路径。分页插件要求把网页改成表格形式，于是在写入网页时每条加入了一个<tr>从而分成表格。分页的实质实际上是统计网页中某个表格有一共多少行，然后选择表格是否显示，达到显示上限的放在下一页。

**2.4 附加功能：**

支持多关键词搜索，可以输入以空格为分割的多个关键词，算法内部如果发现读入的字符串长度大于4则会认为读入了两个以上的词，首先按照空格分词，如果这使得存词的列表长度大于1则直接转入数据库查询工作，如果没有空格则按照一句话分词产生词语列表，并遍历进行查询工作。

**3、参考资料**：

<https://www.runoob.com/django/django-template.html>

<https://www.cnblogs.com/masako/p/5868367.html>

<http://www.do1234.com/python/jieba_fenci.html>

<http://www.runoob.com/sqlite/sqlite-python.html>

<https://www.cnblogs.com/zhouxihi/p/7398376.html>