SHANGHAI JIAO TONG UNIVERSITY

INTRODUCTION TO ELECTRICAL ENGINEERING

The report of experiment 4&5

Author:
Yang Nianzu

Student Number:
517030910301

October 25, 2018

Contents

1 实验准备

1.1 实验环境

和前几次实验相同，本次实验仍是选择在Linux系统下进行。因为Linux从诞生之日起就与Internet密不可分，支持各种标准的Internet网络协议，并且很容易移植到嵌入式系统当中。目前，Linux几乎支持所有主流的网络硬件、网络协议和文件系统，因此它是NFS的一个很好的平台。另一方面，由于Linux有很好的文件系统支持，是数据同步和复制的良好平台，这些都为开发嵌入式系统应用打下了坚实的基础。

此外，各种的实现也依旧是借用Python语言，因为Python具有本身有丰富而且强大的库，而且由于Python的开源特性，第三方库也非常多，所以使用Python语言将为网络爬虫带来极大便利。

1.2 实验目的

实验四主要是让我们学会借助lucene库来实现一个中文网页索引与搜索程序，根据我们所输入的关键词筛选相关的我们爬取下来的信息，可以说是一个简化版的搜索引擎。

在完成实验四的基础上，实验五要求我们进一步完善在之前实验四中所实现的中文网页索引与搜索程序，模拟实现搜索引擎的“site:” 功能要求实现组合查询，即对搜索的网站进行限制。此外，比实验四还更进一步的是，实验五还需要我们实现一个图片索引，这更加考验我们对网页信息的筛选提取能力。

1.3 实验原理

1.3.1 Lucene

Lucene是一个开放源代码的全文检索引擎工具包，但它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎（英文与德文两种西方语言）。Lucene的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎。Lucene是一套用于全文检索和搜寻的开源程式库，它提供了一个简单却强大的应用程式接口，能够做全文索引和搜寻。在Java开发环境里Lucene是一个成熟的免费开源工具。

1.3.2 倒排索引

倒排索引（英语：Inverted index），也常被称为反向索引、置入档案或反向档案，是一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。它是文档检索系统中最常用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。倒排索引主要由两个部分组成：“单词词典”和“倒排文件”。

1.3.3 jieba中文分词

jieba中文分词具有以下特点：

1.支持三种分词模式：精确模式；全模式；搜索引擎模式。

2.支持繁体分词

3.支持自定义词典

2 实验过程

2.1 LAB4·EX1

2.1.1 实验步骤

练习一要求我们实现一个中文网页索引与搜索程序，需要我们爬取一定数量的中文网页，网页数量要尽可能多，然后修改提供给我们的IndexFiles.py和SearchFiles.py两个文件，从而对这些中文网页建立索引并进行搜索，搜索时需要打印出检出文档的路径、网页标题、url。此外，要求doc的Field中需要有name(文件名)，path(文件路径)，title(网页标题)，url(网页地址)，contents(索引的文件内容)。

首先，我们需要先爬去一定数量的网页，爬去的同时要注意信息的筛选，搜索结果要求显示出相关网页的title，所以我们要从网页的源代码中筛选出这个信息，下面这个函数即是获取网页标题。

```
def get_title(content):
    try:
        soup = BeautifulSoup(content)
        title=soup.head.title.string
        return title.strip()
    except Exception as e:
        return 'NONE'
```

Listing 1: 获取title

我将原来crawler.py中的add_page_to_folder(page, content)函数进行了改写，爬去的同时记录下网页的信息于index.txt文件中，用valid_filename()函数将网页变成合法的文件名，文本文档index.txt中每一行记录的是网站的文件名、网页标题、网页地址，该函数的另一用途就是将网页存到文件夹里，以便于之后对网站内容进行分词。

```
def add_page_to_folder(page, content):
    index_filename = 'index.txt'
    folder = 'html'
    filename = valid_filename(page)
    title=get_title(content)
    soup = BeautifulSoup(content)
    try:
        if len(filename)<200:
            index = open(index_filename, 'a')
```

```
            index.write(filename + '\t'+title+'\t'+page.encode('ascii', 'ignore') +
                '\n')
            index.close()
            if not os.path.exists(folder):
                os.mkdir(folder)
            f = open(os.path.join(folder, filename), 'w')
            f.write(soup.get_text())
            f.close()
        else:
            pass
    except:
        pass
```

Listing 2: 改写后的函数

需要注意的是，在将网页化为文件放入文件夹时，由于系统内部创建文件时对文件名长度有一定限制，所以需要对网址过长的网页进行处理。如果只取到允许的最大长度，可能会有不同的网址重名的可能，所以我选择直接放弃网址过长的网页，所以加了一个if语句对网址长短进行判断。

我是对淘宝网进行爬虫的，接下来需要做的工作就是对爬去下来的网页内容做中文分词处理。这一步算是数据处理的工作，这一步其实是可以放在爬取网页的时候同时进行的，但是为了减少爬虫时所要进行的任务量，提高爬虫的速度，所以我选择在爬虫后，对所有网页做统一处理，为此，我新建了一个文件write_file.py对网页进行中文分词。该文件的主要内容就是write_file()函数，为了避免打开文件时所需要注意的问题，所以该py文件放在了之前爬好的网页组成的文件夹html中，该文件的运行效果就是从html中一一读取文件利用jieba进行中文分词，将分完词后的文件放入html文件夹目录下另一新的文件夹html_jieba中。需要注意的是jieba.cut()函数得到的是一个列表，最后我们还需要利用join函数将列表元素用空格连接起来。后续过程中，还需要利用WhitespaceAnalyzer英文分词器做处理。

```
def write_file():
    index_filename = 'index.txt'
    file=open('index.txt','r')
    folder = 'html_jieba'
    line=file.readline()
    while line:
        try:
            filename = line.split('\t')[0]
            file_2 = open(filename, 'r')
            if not os.path.exists(folder):
                os.mkdir(folder)
            f = open(os.path.join(folder, filename), 'w')
            seg_list = jieba.cut(file_2.read())
```

```
            f.write(' '.join(seg_list).encode('utf8'))
            f.close()
            line=file.readline()
        except:
            pass
    file.close()
```

Listing 3: write_file函数

进行完中文分词后，接下来要做的就是建立索引。首先，先对index.txt信息进行提取，方便之后一一对应的建立索引。以下是从中提取信息的相关代码。

```
    file = open('index.txt', 'r')
    filename_list = []
    title_list = []
    url_list = []
    line = file.readline()
    while line:
        try:
            tmp = line.split('\t')
            filename_list.append(tmp[0])
            title_list.append(tmp[1])
            url_list.append(tmp[2])
            line = file.readline()
        except:
            line = file.readline()
            pass
    file.close()
```

Listing 4: 提取index.txt中信息

虽然Indexfiles.py文件内容较复杂，但是当我们理清楚整个代码的功能后，发现其实我们只需要对其中的一个for循环改写即可。根据从index.txt文件中提取出来的文件名列表，一一从html_jieba文件夹中找出对应文件将其内容放在变量contents中存放，根据上一步获得几个列表的元素下标一一对应的关系一一建立索引。

```
    for root, dirnames, filenames in os.walk(root):
        for i in range(len(filename_list)):
            print "adding", filename_list[i]
            try:
                path = os.path.join(root, filename_list[i])
                file = open(path)
                contents = unicode(file.read(), 'utf8')
                file.close()
                doc = Document()
```

```
                    doc.add(Field("name", filename_list[i], t1))
                    doc.add(Field("path", path, t1))
                    doc.add(Field("title", title_list[i], t1))
                    doc.add(Field("url", url_list[i], t1))
                    doc.add(Field("site",get_top_host(url_list[i]),t1))
                    if len(contents) > 0:
                        doc.add(Field("contents", contents, t2))
                    else:
                        print "warning: no content in %s" % filename_list[i]
                    writer.addDocument(doc)
                except Exception, e:
                    print "Failed in indexDocs:", e
```

Listing 5: 建立索引

用jieba进行中文分词后还不够，还需要交给英文分词器处理，我选择的是WhitespaceAnalyzer英文分词器，用它替代原来使用的分词器StandardAnalyzer。WhitespaceAnalyzer使用空格作为间隔符的词汇分割分词器。处理词汇单元的时候，以空格字符作为分割符号。分词器不做词汇过滤，也不进行小写字符转换。需要在文件开头加一句import语句才可以使用它。引用语句如下。

```
from org.apache.lucene.analysis.core import WhitespaceAnalyzer
```

Listing 6: import WhitespaceAnalyzer分词器

然后将原有的StandardAnalyzer替换成Whitespace即可，到此为止，Indexfiles.py已经修改完成，剩下的就是对Searchfiles.py文件的修改。对Searchfiles.py需要进行的修改较少，我们需要将所要打印出信息这部分的代码根据索引改成我们所需要展示的内容。此外，还需注意的是Indexfiles.py和Searchfiles.py中必须使用同一种分词器，所以我们也需要将后者中的分词器设置为WhitespaceAnalyzer分词器。到此，实验四已经完成。

2.1.2 实验结果

该实验较前几次实验而言难度较大，但还是重在理解，只要理解了在PPT里提供给我们的内容和随PPT提供的代码，思路就会变得很清晰，问题也就迎刃而解。该实验最后还是成功完成了，我是对淘宝网进行爬虫的，以下是最后的结果。这里需要注意的是，如果需要输入中文，还需要额外给linux系统额外配置中文输入法。输入“衬衫”，得到五十条相关信息，只截取了三条信息，返回的信息如下。

```
Hit enter with no input to quit.
Query:衬衫

Searching for: 衬衫
50 total matching documents.
path: html_jieba/httpzaaa.taobao.com
title: 首页-linetwo号线-淘宝网
url: http://su-h.taobao.com
name: httpzaaa.taobao.com


path: html_jieba/httpfuzhuang.1688.comnanzhuang
title: 1688男装市场
url: http://fuzhuang.1688.com/xie
name: httpfuzhuang.1688.comnanzhuang


path: html_jieba/httpgiordanos.taobao.com
title: 首页-佐丹奴官方店-淘宝网
url: http://camelyy.taobao.com
name: httpgiordanos.taobao.com
```

Figure 1: The results of LAB4·EX1

2.2 LAB5·EX1

2.2.1 实验步骤

该练习要求我们实现搜索引擎的“site:”功能，即对搜索的网站进行限制。我们需要先对每个网页进行处理，获取它的域名，所以我利用正则表达式设计了一个函数获取域名，以下是该函数代码，函数参数为网址。

```
topHostPostfix = (      '.com','.la','.io','.co','.info','.net','.org','.me','.mobi',
                        '.us','.biz','.xxx','.ca','.co.jp','.com.cn','.net.cn',
                        '.org.cn','.mx','.tv','.ws','.ag','.com.ag','.net.ag',
                        '.org.ag','.am','.asia','.at','.be','.com.br','.net.br',
                        '.bz','.com.bz','.net.bz','.cc','.com.co','.net.co',
                        '.nom.co','.de','.es','.com.es','.nom.es','.org.es',
                        '.eu','.fm','.fr','.gs','.in','.co.in','.firm.in','.gen.in',
                        '.ind.in','.net.in','.org.in','.it','.jobs','.jp','.ms',
                        '.com.mx','.nl','.nu','.co.nz','.net.nz','.org.nz',
                        '.se','.tc','.tk','.tw','.com.tw','.idv.tw','.org.tw',
                        '.hk','.co.uk','.me.uk','.org.uk','.vg', ".com.hk")

def get_top_host(url):
    parts = urlparse(url)
    host = parts.netloc
    extractPattern = r'[^\.]+('+'|'.join([h.replace('.',r'\.') for h in
        topHostPostfix])+')$'
    pattern = re.compile(extractPattern,re.IGNORECASE)
    m = pattern.search(host)
```

```
    return m.group() if m else host
```

Listing 7: 获取网页域名

将上面这部分代码加入原来的Indexfiles.py文件中，并在建立索引时加一句代码如下，从而建立新的索引。

```
                doc.add(Field("site",get_top_host(url_list[i]),t1))
```

Listing 8: 更新索引

更新完索引后，接下来对Searchfiles.py进行修改，我新建了一个文件名为lab5ex1.py文件，即为修改后的Searchfiles.py，实现组合查询的代码在PPT中已经提供给我们，我们所要进行的改动不多。查询时，我们改为通过使用BooleanQuery可以将不同的查询组合成复杂的查询方式。定义了一个函数对我们输入的关键词进行处理，该函数返回的是一字典，该函数如下。

```
def parseCommand(command):
    allowed_opt = ['site']
    command_dict = {}
    opt = 'contents'
    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:
            command_dict[opt] = command_dict.get(opt, '') + ' ' + i
    return command_dict
```

Listing 9: 处理输入指令

将输入的指令进行处理后，再利用BooleanQuery根据处理输入返回的字典进行组合查询，即可实现。利用BooleanQuery的代码如下。

```
        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, 50).scoreDocs
```

```
        print "%s total matching documents." % len(scoreDocs)
```

Listing 10: 利用BooleanQuery进行组合查询

到此，练习五的实验一已经完成。

2.2.2 实验结果

PPT已经提供了实现该内容的主要代码，我们只需要对这些代码稍加改动再添加即可，我们需要自主克服的难点只有如何获取一个网页的域名。以下是我对该练习的结果展示，输入指令为“汽车 site:taobao.com”，结果显示域名为taobao.com的且与汽车相关的网址，且显示的为车蜡、汽车遮阳帘之类的网址，确实与输入相关，可见练习一成功完成，以下是部分结果展示。

```
Hit enter with no input to quit.
Query: 汽车 site:taobao.com
50 total matching documents.
path: html_jieba/httplist.tmall.comsearch_product.htmfromrs_1_key-top-sqC6FBB3B5D5DAD1F4C1B1
title: 汽车遮阳帘-天猫Tmall.com-理想生活上天猫
url: http://store.taobao.com/search.htm?user_number_id=91106030&rn=e1c10ae61ace364aa6248930071bf9a7&keyword=
name: httplist.tmall.comsearch_product.htmfromrs_1_key-top-sqC6FBB3B5D5DAD1F4C1B1


path: html_jieba/httplist.tmall.comsearch_product.htmactive1fromrs_1_key-top-sqC6FBB3B5CDB7D5ED
title: 汽车头枕-天猫Tmall.com-理想生活上天猫
url: http://store.taobao.com/search.htm?user_number_id=2068514101&rn=8e0c57f63622a901c368b5f677b7d20a&keyword=
name: httplist.tmall.comsearch_product.htmactive1fromrs_1_key-top-sqC6FBB3B5CDB7D5ED


path: html_jieba/httplist.tmall.comsearch_product.htmfromrs_1_key-top-sqB3B5C0AFBBAEBADBD0DEB8B4
title: 车蜡 划痕修复-天猫Tmall.com-理想生活上天猫
url: http://store.taobao.com/search.htm?user_number_id=132989916&rn=6e773624ec868de994af0efff4f8ec15&keyword=
name: httplist.tmall.comsearch_product.htmfromrs_1_key-top-sqB3B5C0AFBBAEBADBD0DEB8B4
```

Figure 2: The results of LAB5·EX1

2.3 LAB5·EX2

2.3.1 实验步骤

该练习要求实现一个图片索引，输入文本，输出相关的图片地址，图片所在网页的网址，图片所在网页的标题。做该练习时，最好选定某个特定的网站爬取，因为这样可以对特定网站的结构进行分析，可以让搜索结果更精确。之前的练习，我都是对淘宝网进行爬去，这次同样的，我也是对淘宝网进行爬去，其更主要的原因是因为淘宝网上的图片较多，所以比较适合爬取。

首先，需要对分析淘宝网的源代码，找出可以提供图片信息的部分。经过分析，发现有如下几处可能提供与图片相关的信息。

```
<meta name="keywords" content="~~"/>

<img id="~~~~" alt="~~~~" src="~~~~"/>
```

Listing 11: 提供图片信息的部分

此外，网页的标题也有可能会提供与图片相关的信息，获取网页标题的函数之前已经定义过，所以需要再定义两个函数来提取如上所展示的两个标签的信息。定义的两个函数如下。

```
def get_keywords(content):
    content1=BeautifulSoup(content)
    try:
        for i in content1.findAll('meta',{'name':re.compile('^keywords|^/')}):
            return i.get('content','NONE')
    except:
            return 'NONE'

def get_all_imgs_info(content, page):
    imgs= []
    imgs.append(page)
    content1 = BeautifulSoup(content)
    try:
        for i in content1.findAll('img'):
            if i['src'][0] == '/':
                if i.get('alt')=="":
                    imgs.append(['NONE',urlparse.urljoin(page, i['src'])])
                else:
                    imgs.append([i.get('alt','NONE'),urlparse.urljoin(page, i['src'
                        ])])
            else:
                if i.get('alt')=="":
                    imgs.append(['NONE', i.get('src', 'NONE')])
                else:
                    imgs.append([i.get('alt', 'NONE'), i.get('src', 'NONE')])
    except:
        pass
    return imgs
```

Listing 12: 提供与图片相关的信息

上面两个函数分别对应着上面举的两个标签的例子，第二个函数返回的是一个列表，它在获取标签alt的值的同时，同时还会获得图片的地址，并且如果是相对地址，还会用函

数urljoin()获得完整的地址，并且将该图片所在的网页的地址放在列表的第一个元素的位置，因为一个网页中可能有多个图片，这些图片所在网页的地址是相同的，这样存储数据不仅可以节省空间，还会方便后续的建立索引的操作。

接下来，与实验四类似，同样对add_page_to_folder(page, content)函数进行改写，将爬去的网页信息按一定格式记录在imgs.txt文件中，将所爬去的网页内容保存在lab5_html文件夹中，imgs.txt文件每一行的内容顺序是图片标签中alt的值、图片地址、所在网页的keywords、所在网页标题、所在网页的地址。lab5_html文件夹中的每个文件里写入的是图片标签中alt的值、所在网页的keywords、所在网页标题，这些都是可能提供图片内容的信息。以下是修改过后的add_page_to_folder(page, content)函数。

```
def add_page_to_folder(page, content):
    index_filename = 'imgs.txt'
    folder = 'lab5_html'
    title = get_title(content)
    img_list=get_all_imgs_info(content,page)
    keywords=get_keywords(content)
    for i in range(1,len(img_list)):
        try:
            if len(img_list[i][1])<255:
                index = open(index_filename, 'a')
                index.write(img_list[i][0]+ '\t' + img_list[i][1] + '\t' +keywords+
                    '\t'+ title+'\t'+page.encode('ascii', 'ignore') + '\n')
                index.close()
                if not os.path.exists(folder):
                    os.mkdir(folder)
                filename=valid_filename(img_list[i][1])
                f = open(os.path.join(folder, filename+'.txt'), 'w')
                f.write(img_list[i][0]+'\n'+keywords+'\n'+title)
                f.close()
        except:
            pass
```

Listing 13: 修改后的该函数

同样的，这次在爬取时，我选择暂不做jieba分词，爬取到一定数量的网页后再做同一分词处理，为此，我写了一个名为lab5_writefiles.py的文件统一对之前保存的图片信息做分词处理，该函数与之前的函数大致相同，所以这里不再重复展示。运行该程序后，将分完词后的文件保存在名为lab5_html_jieba的文件夹中。

剩下来的就是为该练习建立对应的建立索引和查询两个程序，为此，我编写了两个文

件文件名分别名为lab5_Indexfiles.py和lab5_searchfiles.py，这两个文件的内容与实验四的两个文件大同小异，主要思想相同，所以这里也不再展示。所以，该练习也已完成。

2.3.2 实验结果

在已经完成了练习四的基础上，再完成该练习的难度不是很大，因为两个练习大同小异，所以该实验只要理清思路，就能顺利完成。以下是该实验结果的展示。

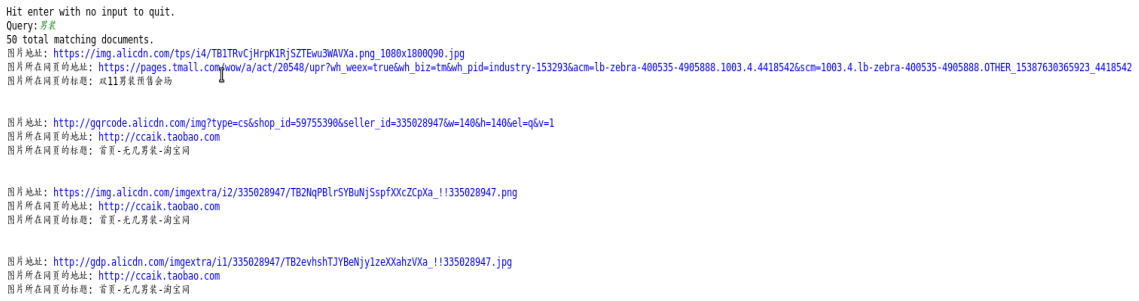

Figure 3: The results of LAB5·EX2

3 实验感想

通过本次实验，收获颇多，学会了如何实现搜索引擎比较基本的功能，进一步提升了自己对信息筛选以及数据处理的能力。本次实验，虽然有遇到一些问题，但还是通过在网站上搜索一些到一些与问题相关的博客得以自主解决，增加了自主解决问题的能力。最后，衷心感谢老师和助教们的悉心指导，才使得实验成功。