林禹臣 5140309507

October 23, 2015（Week 6 & 7）

电类工程导论C 实验报告4&5

——lucene

目录页

1. 引言（Introduction）

在 lab4 和 lab5 中，我主要学习了如何利用 lucene 来建立全文索引和进行搜索，从而基本实现了整个简易搜索引擎的基本架构。

在 lab4中，我首先完成了在 OS X 上进行jcc 和 pylucene 的安装和配置（历经了各种困难），然后依次实现了爬取网页，分析网页特征（title、charset 编码），清理html 标签，清洗文本（分词、去除停用词以及标点符号）等过程，然后对已经分好词的文本文件利用 Whitespace Analyzer 建立了索引，实现了搜索。

由于 lab5的第一部分是实现组合搜索，所以我把这一部分也归在 lab4中一带实现了。

在 lab5中主要工作是对图片进行了关键词索引，从而实现图片的搜索，基本思想和 lab4一致，只不过索引的对象和属性发生了改变。

2. 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 4.7.2 + BeautifulSoup 4.4.1 + jieba 0.37

(需要注意的是 Lucene 这里用的版本是4.x 。主要是因为3.x 在 OS X 中无法安装成功。而4和3的包结构有许多不同，索引文件的方式也不同，在lucene3中无法运行。需要安装lucene 4.x才可以测试。)

3. 预备知识 （Propaedeutics）

- 信息检索的基本方式：顺序扫描法、索引搜索法。
- 索引的建立过程：文档、词典、词元、分词器、语言分析器、索引器
- 索引的检索过程： 查询命令、分词器、组合查询、同样的分析器、语言分析器、搜索组件
- 建立索引之前的准备工作：清除 HTML 代码，清理标点符号，分词，去除停用词

4. 实验过程（Experiment）

4.1.lab4 文本索引准备（Graph Theory）

4.1.1.总体思路

首先我们要知道，爬虫在下载网页时由于要进行网络连接，不适宜在此基础上直接进行文本分析，所以要把它们分开处理。爬虫只负责下载源代码，存储于 html 文件夹中，url 和源代码文件名的对应关系存于url2file_index_sjtu.txt 中，然后我们利用这个 txt 文件中的信息，对源代码进行清洗、提取、分词之后，把他们存储到 txt 文件夹之中，同时生成一个新的 txt 文件叫做infoIndex.txt ，此文件存储 url、文件名、title，这三个信息。然后我们就可以利用索引器直接对这些文档进行索引了。

4.1.2.爬虫的修改

因为我想做一些偏垂直化的搜索，所以要在特定的网站内部进行爬去，所以这就需要修改爬虫代码，从而实现提取特定特点的 url，在这个过程中，也同时避免了爬去很多 apk，pdf，rar，exe 等后缀的文件，减少对垃圾文件的处理。

```
#if(link.startswith('http://www.cnblogs.com')):
#if(link.startswith('http://baike.baidu.com/view/')):
#if(link.startswith('http://news.sjtu.edu.cn')):
```

我选择的几个网站主要是百度百科、知乎、博客园、京东，交大新闻网这种页面比较干净，网页源代码比较规整的网站。他们的编码不尽相同，存在 gbk ，gb2312 和 utf-8的，这个问题的解决会在后面进行说明。

4.1.3.清除 HTML代码 删除标点符号

关于清除 HTML 代码，我有4种选择。

1.PPT 中 bs3中的方法。

2.利用nltk 的老版本中 clean_html 函数

3.利用bs4的 get_text() 函数

4.自己利用正则表达式来写

通过效率的比较，对某一个特定的文件， bs3 为 0.020s，nltk 为 0.013s，bs4为0.004s，我自己写的正则表达式代码为0.016s。（注意到 bs4比 nltk 快很多，这也是 nltk 的开发者在最新版本中停用了自己的函数，并且附加说明建议使用 bs4的原因。）

按理来说，我们应该选择效率最高的 bs4才对，但是根据实践，发现 bs4的清除效果并不好，有时甚至还会留下大段的html 标签（尤其是知乎的页面）。因为这一步是离线作业，所以我还是选择了可控性比较高的个人正则表达式代码，而且我直接在其中加入了去除标点符号，中文标点符号，以及换行符等等的清洗工作。

```
def clean(html):
    import re,string
    text = re.sub(r"(?is)<(script|style).*?>.*?(</\1>)", "", html.strip())
    text = re.sub(r"(?s)<!--(.*?)-->[\n]?", "", text)
    text = re.sub(r"(?s)<.*?>", "", text)
    text = re.sub(r"&.*?;", "", text)
    todelete = ['\n','\r','\t',' ','。','? ','、','! ',': ','“ ','” ',' ',';
    for i in todelete:
        text = text.replace(i,'')
    return text
```

效率虽然并不是特别高，但是可以结合多线程，很快完成了任务。

4.1.4.中文分词、去除停用词

对于中文分词，我其实在大一上的计算导论这门课的大作业中实现过自己的分词器（https://github.com/yuchenlin/RossetaSeg ），主要利用了 HMM 的思想，结合维特比算法对未登录词进行了优化处理，概率参数来自 jieba 作者的训练数据。效

率比较高，但是因为大量代码使用了 python3.4的特性，移植到python2.x 很麻烦，还是选择了 jieba来作为分词器。

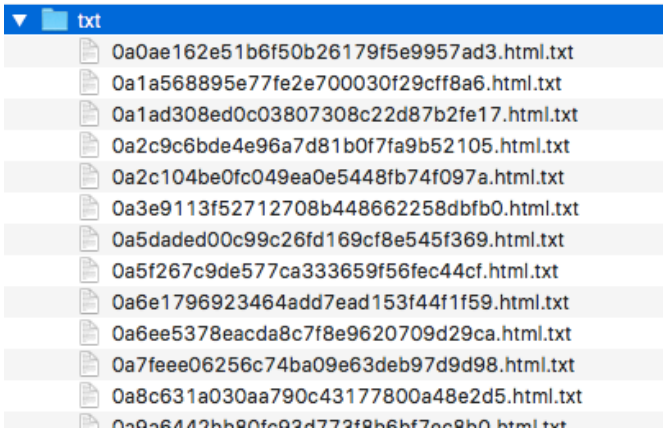
分词之后，需要进行对停用词的删除，这个部分没有想到效率高的方法，jieba 的接口实现的也不够好，所以只能用原始的算法来实现了。

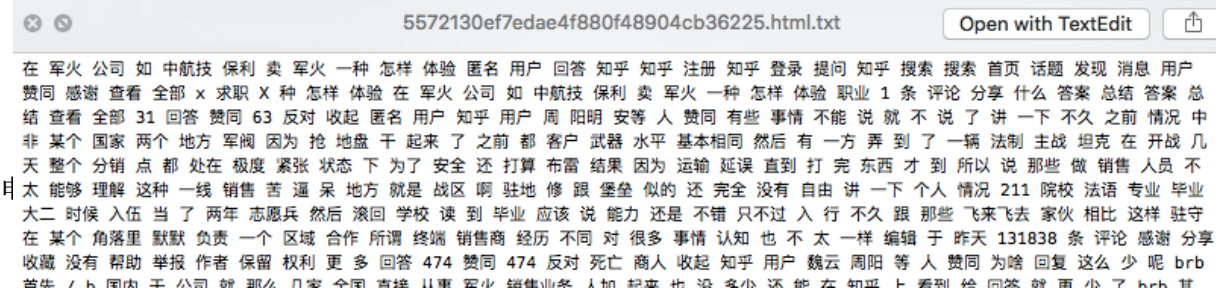
```
import jieba
l = jieba.cut(text,HMM=True)
final = []
stopwords='你们我们他们是这个那个的地得和'
for i in l:
    if(i in stopwords):
        continue
    else:
        final.append(i)
txt = " ".join(final)
```

整个4.1的代码在lab4/infoindex.py 中，还有一些细节，比如 title 的获取和获取之后的判断，留在下面讲述原因。值得一提的是，这里我全部选择了 codecs.open 的方法来打开文件，因为 codecs 会实现 unicode 的中间编码转换，更为安全可靠，减少了编码方面的错误。

4.2. lab4 建立文本索引（Texts Indexing）

在上一步的操作之后，其实已经在 txt 文件夹中对每一个网页生成了一个 txt 文件，里面的内容已经是清洗且分好词之后的了，如图：





所以我们接下来需要做的只是把这些文档全部提交给索引器即可。因为已经分好词了，而且还是用空格来做分隔符，所以直接利用WhitespaceAnalyzer即可。

```
def indexDocs(self, root, writer):

    f = codecs.open('infoIndex.txt','r',encoding='utf-8')
    files = {}
    for line in f.xreadlines():
        ls = line.split()
        files[ls[0]+'.txt'] = [ls[1],ls[2]]
    f.close()
```

注意此处是利用infoIndex.txt 来读取信息，找到处理后的 txt 文件地址，title，url 等信息存储到 files 这个字典中以便后面建立文档时方便调取。

```
doc = Document()
doc.add(Field("name", filename,
                      Field.Store.YES,
                      Field.Index.NOT_ANALYZED))
doc.add(Field("path", path,
                      Field.Store.YES,
                      Field.Index.NOT_ANALYZED))
url = files[filename][0]
doc.add(Field("url", url,
                      Field.Store.YES,
                      Field.Index.NOT_ANALYZED))
domin = urlparse.urlsplit(url)[1].split(':')[0]
doc.add(Field("site", domin,
                      Field.Store.YES,
                      Field.Index.NOT_ANALYZED))
title = files[filename][1]
doc.add(Field("title", title ,
                      Field.Store.YES,
                      Field.Index.NOT_ANALYZED))
print filename,path,url,domin,title
if len(contents) > 0:
    doc.add(Field("contents", contents,
                          Field.Store.NO,
                          Field.Index.ANALYZED))
else:
    print ("warning: no content in %s" % filename)
writer.addDocument(doc)
```

注意到这里，我们增加了一个 site 属性，这个属性存储的是当前网页所在的域名，获取方法是利用了 urlparse 库的urlsplit 函数。这个属性是为了配合我们下一步进行的组合搜索，从而限定了在某一个特定域名内部的搜索。

4.3. lab4-5 文本组合搜索（Combinatorial Text Search）

这里我们做的改变就是把原来调用的 query 改成了 BooleanQuery 的对象，从而更加个性化的调整它的指向。用 parseCommand 可以指定对关键词的提取，比如 site，title，url，contents（默认的选项）。这里需要说明的是，样例代码有一点问题就是组合查询时必须要让 “site:”在查询内容前面才可以，但是Google 是前后都可以的。 所以这里需要把opt默认值的设定放进循环内部。

```
def parseCommand(command):
    allowed_opt = ['title', 'url', 'site']
    command_dict = {}

    for i in command.split(' '):
        opt = 'contents'
        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, '') + ' ' + ' '.join(jieba.cut(i))
    return command_dict
```

Google支持对域名进行多级的搜索，比如可以通过搜索baidu.com 来检索，baike.baidu.com的内容，还支持类似搜索 ‘site:news.sjtu.edu.cn/info/’的形式，这样简单的利用 site 来做全文匹配搜索就不够了。我选择了利用 url 这个字段，结合一个叫做 Wildcard 的对象来建立相对应 query。因为默认情况下 Queryparse 对未分词的字段只支持完全匹配，而非完全匹配则需要通配符处理，代码如下：

```
for k,v in command_dict.iteritems():
    if(k=='site'):
        t = Term('url','*'+v.strip()+'*')
        query = WildcardQuery(t)
    else:
        query = QueryParser(Version.LUCENE_CURREN
    querys.add(query, BooleanClause.Occur.MUST)
```

可以看到我们在 url 字段里搜索，*v*的结构，这样当 v 是url 的任何一个部分都可以搜索了，一石二鸟地解决了 Google 的两种搜索方式。

效果如下：

site:baike.baidu.com 小说

```
Searching for: 小说 site:baike.baidu.com
50 total matching documents.
title: 社会小说_百度百科 url: http://baike.baidu.com/view/2686394.htm
title: 中国小说史略_百度百科 url: http://baike.baidu.com/view/94477.htm
title: 捡来的小恶魔_百度百科 url: http://baike.baidu.com/view/5767529
title: 言叶_百度百科 url: http://baike.baidu.com/view/4378217.htm
title: 二十年目睹之怪现状_百度百科 url: http://baike.baidu.com/view/37247.ht
title: 傅雷译文集_百度百科 url: http://baike.baidu.com/view/449142.htm
title: 果香飘飘_百度百科 url: http://baike.baidu.com/view/6957151
title: 简·奥斯汀_百度百科 url: http://baike.baidu.com/view/42009.htm
title: 艾德温·德鲁德之谜（同名电影）_百度百科 url: http://baike.baidu.com/viev
title: 娘子合欢_百度百科 url: http://baike.baidu.com/view/10445106
title: 白沙码头_百度百科 url: http://baike.baidu.com/view/1919702.htm
title: 居斯塔夫·福楼拜_百度百科 url: http://baike.baidu.com/view/37683.htm
title: 吸血鬼猎人_百度百科 url: http://baike.baidu.com/view/97771.htm
```

site:www.zhihu.com 小说

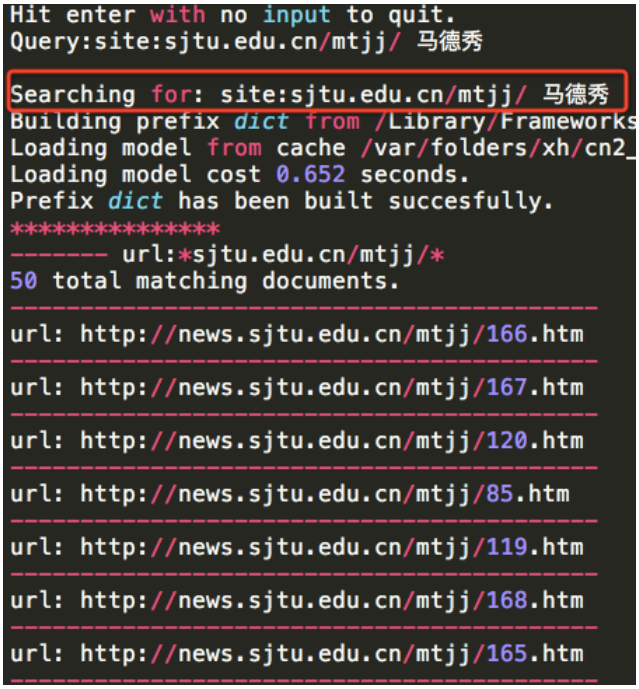
```
Searching for: 小说 site:www.zhihu.com
50 total matching documents.
title: 当我们谈论「小说技术」或「小说技巧」时我们通常指的是什么？ url: http://www.zhihu.com
title: 金庸先生的作品，读者群体主要是？ url: http://www.zhihu.com/question/19596536/an
title: 直男会不会觉得写耽美小说的作家很恶心？ url: http://www.zhihu.com/question/221711
title: 你认为金庸笔下第一大侠是谁？ url: http://www.zhihu.com/question/23356482
title: 网络小说和传统小说有哪些区别？ url: http://www.zhihu.com/question/21013221
title: 魔幻、奇幻、玄幻、科幻文学作品有明确的分类界限吗？ url: http://www.zhihu.com/quest
title: 结局或结构出乎意料的国产高智商电影有哪些？ url: http://www.zhihu.com/question/199
title: 该怎样构思一篇小说？ url: http://www.zhihu.com/question/36265692
title: “致敬”和“抄袭”的区别在哪里？ url: http://www.zhihu.com/question/23519537/answ
title: 金庸小说中武林和官府之间的关系是什么？为什么《笑傲江湖》中刘正风好歹也是朝廷在册官员，被嵩
title: 辛夷坞所有小说中各种人物的关系？ url: http://www.zhihu.com/question/21960622
title: 魔幻、奇幻、玄幻、科幻文学作品有明确的分类界限吗？ url: http://www.zhihu.com/quest
title: 你所知道的哪些小说中的人，过得是极简主义生活？ url: http://www.zhihu.com/question/
title: 如何评价哈哈笑播讲的《三体》全集？ url: http://www.zhihu.com/question/36606326
title: sixue url: http://www.zhihu.com/people/sixue/topic/19550447/answers
title: 魔幻、奇幻、玄幻、科幻文学作品有明确的分类界限吗？ url: http://www.zhihu.com/quest
```

site:zhihu.com 余华 （非完全匹配的 domin）

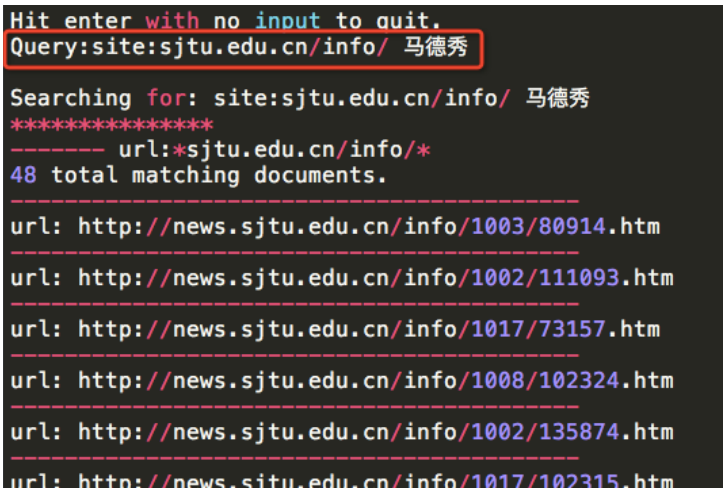
```
Query site:zhihu.com 余华

Searching for: site:zhihu.com 余华
Building prefix dict from /Library/Frameworks/Python.framework/
Loading model from cache /var/folders/xh/cn2_6lzj0495f_k7qzhmgb
Loading model cost 0.643 seconds.
Prefix dict has been built succesfully.
31 total matching documents.
----------------------------------------
title: 刘战京 url: http://www.zhihu.com/people/liu-zhan-jing
----------------------------------------
title: 这个杀手有点冷 url: http://www.zhihu.com/people/zhe-ge-sha-
----------------------------------------
title: 王雨峰 url: http://www.zhihu.com/people/wang-yu-feng-4
----------------------------------------
title: 树莓 url: http://www.zhihu.com/people/zhujingjie.com
```

site:sjtu.edu.cn/mtjj/ 马德秀（domin 后还有子目录的匹配）

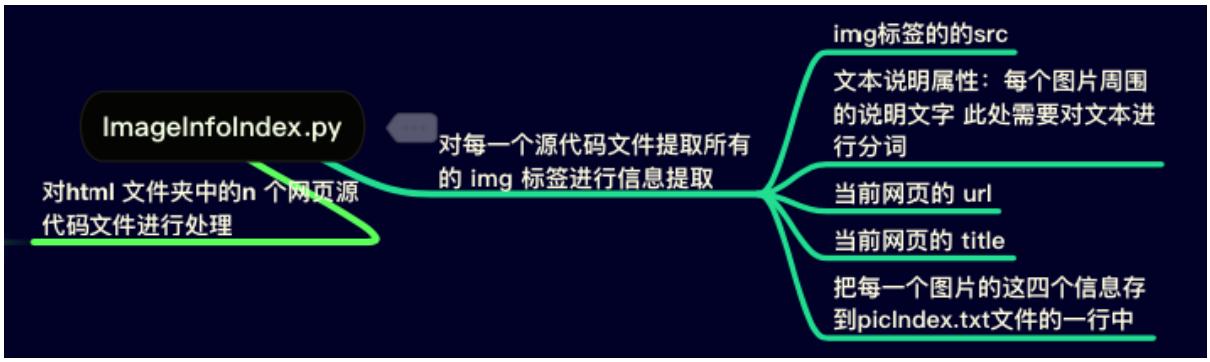


site:sjtu.edu.cn/info/ 马德秀



4.4. lab5 图片索引准备（Preparing to Build Index）

我对图片索引的准备的思路大致如下：



4.4.1. 爬取 html 页面

我主要是对京东的商品图片进行索引，根据测试发现京东的商品页面的 url 规律非常整齐都是由 http://item.jd.com/xxx.html 的形式存在的（可以利用正则表达式来简化获取 all_links 的函数），而且一个商品页下会有多个相关商品的链接，所以只要从某一个商品开始爬取即可。

```
#without bs 专为京东
def get_all_links(content, page):
    if content == None:
        return []
    import re
    urlset = set()
    urls = re.findall(r"item.jd.com/[^\s]*.html",content,re.I)
    for u in urls:
        urlset.add('http://'+u)
    links = list(urlset)
    return links
```

4.4.2. ImageInfoIndex.py 分析 html 代码获取图片信息

注意，我处理图片信息的算法有一个核心的思想就是准确度优先，平均每个页面获取的图片数量我们并不在乎，因为这个可以通过增大总页面数量来抵抗掉。但是如果图片和文字的搭配信息不准确是无法消除的。

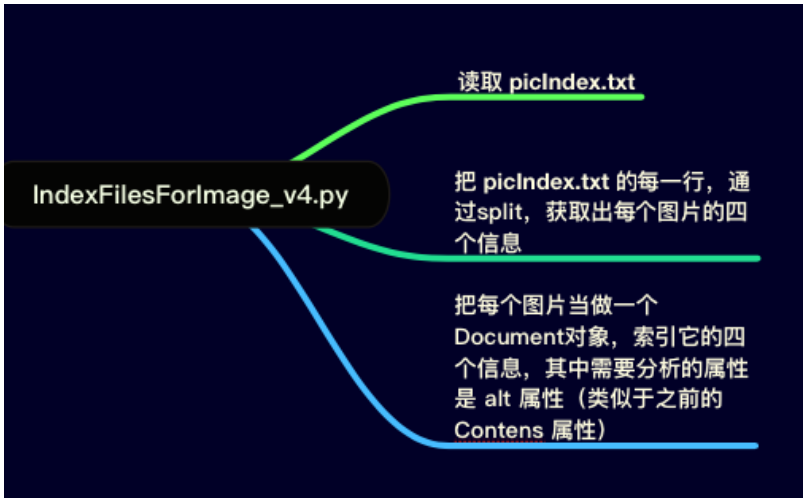
所以，这里我只选取了 alt 属性来作为唯一的文字说明，这个也是经过测验得到的结论。之前，我试过同时用周围的其他文字来做说明，但是发现效果非常不好，原因在于京东的『相关商品』推荐栏中搭配销售的商品在关于『物品购买』上的相关性确实有，但是在『图片内容』上的主题相关性并不大。比如，当前商品页面是黑芝麻糊，京东会推荐你购买核桃、豆浆机等等商品，这些商品虽然客户很有可能同时购买，但是这些商品的图片与黑芝麻糊这个图片内容上是不相关的。这么做还有一个理由就是因为京东把几乎每一个图片的 alt 属性都设置的非常好，非常完整，不像淘宝一样零碎。

```
def getImgInfo(item):
    global count
    FOLDER_NAME = 'html/'
    path = FOLDER_NAME + item
    f = codecs.open(path,'r')
    html = f.read()
    f.close()
    bs = BeautifulSoup(html,'html.parser')
    if(bs.title==None):
        return
    title = bs.title.text.strip()
    for img in bs.find_all('img'):
        src = img.get('src','')
        if(src=='' or not (src.endswith('jpg') or src.endswith('png'))):
            continue
        src = 'http:' + src
        if(mbf.lookup(src)): #BloomFilter
            continue
        mbf.add(src)
        alt = img.get('alt','')
        if(alt==''):
            continue
        alt = ' '.join(jieba.cut(alt.strip()))
        url = urlDict[item]
        tosave = url+'seg^*'+title+'seg^*'+src+'seg^*'+alt
        count+=1
        print count,'*'
        index = codecs.open('picIndex.txt','a')
        index.write(tosave.encode('utf-8','ignore')+'\n')
        index.close()
```

注意，这里我们用了布隆过滤器来防止图片地址的重复，实际发现一开始没用过滤器，5W 个图片信息中之有3W 个是不重复的，所以布隆过滤器还是很有必要的，（当然也可以在添加索引的时候先用一个 set 来去重，实际上我在两层都有处理）。

另外一点就是存储到 picIndex.txt 的数据必须要每行内部有分隔符，而且这个分隔符必须非常罕见才可以，不能设置为空格和*号等字符，因为 title 和 alt 里都可能会出现这样的字符，所以我这里设置的分隔符是一个字符串叫做「seg^*」。

4.5. lab5 建立图片索引（Images Indexing)



建立图片的索引的思路如上，核心代码如下，其实和之前的建立索引并没有本质的不同。（但是，很多同学向我询问思路的时候，我发现很多人误认为一个 Document 对象必须对应一个文本文件，从而导致整个索引的准备过程极其复杂。其实完全可以像我一样把所有信息都存到一个文件中，每一行表示一条记录即可。）

```
def indexDocs(self, root, writer):
    f = codecs.open('picIndex.txt','r',encoding='utf-8')
    picDict = {}
    for line in f.xreadlines():
        ls = line.split('seg^*')
        url = ls[0]
        title = ls[1]
        src = ls[2]
        alt = ls[3]
        picDict[src] = [url,title,alt]
    f.close()
    for src in picDict:
        doc = Document()
        doc.add(Field("src", src,Field.Store.YES,Field.Index.NOT_ANALYZED))
        doc.add(Field("url", picDict[src][0],Field.Store.YES,Field.Index.NOT_ANALYZED))
        doc.add(Field("title", picDict[src][1],Field.Store.YES,Field.Index.NOT_ANALYZED))
        doc.add(Field("alt", picDict[src][2],Field.Store.YES,Field.Index.ANALYZED))
        writer.addDocument(doc)
```

注意这里分隔时的分隔符就是前文提到的字符串。另外要注意此时的 Analyzer 也是Whitespace 的，因为 alt 属性早已在存储时就已经被分过词了。

4.6. lab5 图片搜索 (Image Search)

图片的搜索本身非常简单，直接套用之前的搜索代码即可。但是因为我爬取的页面数量有限，而且由于京东商品的分类性比较强，我爬取的图片主要集中在商品和 3C 产品这两类上。不过这个只是数量和训练集的问题而已。展示：

```
Searching for: 油炸
18 total matching documents.
--------------------------------------------------
title: 米多奇烤香馍片 独立包装 非油炸 口味随机 50g【图片 价格 品牌 报价】-京东
url: http://item.jd.com/1579609648.html
src: http://img13.360buyimg.com/n5/g14/M09/18/0A/rBEhVlJvDd8IAAAAAAFPEYUsZvMAAEwCAFCkkoAAU8p580.jpg
--------------------------------------------------
title: 米多奇烤香馍片 独立包装 非油炸 口味随机 50g*10【图片 价格 品牌 报价】-京东
url: http://item.jd.com/1579609649.html
src: http://img14.360buyimg.com/n5/jfs/t181/93/596576581/248897/8a8f6a42/5392ec21N2471bf15.jpg
--------------------------------------------------
title: 米多奇烤香馍片 独立包装 非油炸 口味随机 50g*10【图片 价格 品牌 报价】-京东
url: http://item.jd.com/1579609649.html
src: http://img14.360buyimg.com/n5/jfs/t151/117/610776599/230641/3a9dbe96/5392ec1cNc284e936.jpg
--------------------------------------------------
title: 米多奇烤香馍片 独立包装 非油炸 口味随机 50g【图片 价格 品牌 报价】-京东
url: http://item.jd.com/1579609648.html
src: http://img13.360buyimg.com/n1/jfs/t199/130/625752941/281449/2f982f05/5392ec19Nf3e242c9.jpg
--------------------------------------------------
title: 朗琨食品 内蒙古风干牛肉干 阿尔善非油炸牛肉干 原味 228【图片 价格 品牌 报价】-京东
url: http://item.jd.com/1570757391.html
src: http://img11.360buyimg.com/n5/jfs/t925/294/787062669/200952/863204bd/55544717N48e1dad2.jpg
--------------------------------------------------
title: 朗琨食品 内蒙古风干牛肉干 阿尔善非油炸牛肉干 原味 228【图片 价格 品牌 报价】-京东
url: http://item.jd.com/1570757391.html
```

```
Searching for: 小米手机
50 total matching documents.
--------------------------------------------------
title: 【小米红米2 增强版】小米 红米2 增强版 白色 移动4G手机 双卡双待【行情 报价 价格 评测】-京东
url: http://item.jd.com/1514817.html
src: http://img12.360buyimg.com/n5/jfs/t1507/141/217911278/20686/55152ab0/55652d58Nfb5b1673.jpg
--------------------------------------------------
title: 【小米MI4】小米 4 2GB内存版 白色 联通4G手机【行情 报价 价格 评测】-京东
url: http://item.jd.com/1514796.html
src: http://img11.360buyimg.com/n5/jfs/t1351/104/111743210/64930/68ab1338/555aec07N47c3c7e0.jpg
--------------------------------------------------
title: 【小米Note】小米 Note 黑色 移动联通双4G手机 双卡双待【行情 报价 价格 评测】-京东
url: http://item.jd.com/1514802.html
src: http://img12.360buyimg.com/n5/jfs/t1498/233/104332566/85703/f0895ce4/555aefb3Nf1331d97.jpg
--------------------------------------------------
title: 【小米MI4】小米 4 2GB内存版 白色 联通4G手机【行情 报价 价格 评测】-京东
url: http://item.jd.com/1514796.html
src: http://img11.360buyimg.com/n5/jfs/t1180/236/925849879/82803/784f564d/555aebfbN2625109b.jpg
--------------------------------------------------
title: 【小米红米2A】小米 红米2A 灰色 移动4G手机 双卡双待【行情 报价 价格 评测】-京东
url: http://item.jd.com/1514837.html
src: http://img12.360buyimg.com/n5/jfs/t1030/298/984193911/70415/d38ec978/556539caN1c17af24.jpg
--------------------------------------------------
title: 【小米MI4】小米 4 3GB内存版 亮白 移动4G手机【行情 报价 价格 评测】-京东
url: http://item.jd.com/1514790.html
src: http://img10.360buyimg.com/n5/jfs/t1018/225/930649164/65637/7344fb72/555aec06Nb6cfe2e0.jpg
--------------------------------------------------
```

5. 遇到的困难和解决方案 （Problems & Solutions）

5.1.在 OS X 下配置 jcc 和 pylucene

因为pylucene 很久没有人维护更新，pylucene3.x在新的OS X 下不能安装，从 1.7 开始JDK 也在 OS X 的版本删去了32位虚拟机，所以安装过程极其困难（困难主要在于这两件事情是经过探索）。最后总结了安装步骤如下：

1.删去所有高于1.6的 jdk 版本，安装 Apple 公司和 Java在2015年发布的特殊版本 jdk 1.6。

2.在 terminal 里输入 java -version 看到当前的虚拟机版本是64位的，输入 java -version -d32 之后把虚拟机版本切换为32位。

```
Last login: Sat Oct 17 17:45:37 on ttys001
LinYuchens-Air:~ yuchenlin$ java -version -d32
java version "1.6.0_65"
Java(TM) SE Runtime Environment (build 1.6.0_65-b14-468-11M4833)
Java HotSpot(TM) Client VM (build 20.65-b04-468, mixed mode)
LinYuchens-Air:~ yuchenlin$
```

3.（删除现有的 python 2.7） 重新去 python 官网下载32位版本的Python 2.7.10 然后安装。测试是否是32位，可以在 terminal 里输入：

python

>>>import platform

>>>platform.architecture()

输出如果是('32bit', '') 则表示当前 python 是32位版本的。

4. 在这两者的基础上我们可以继续安装 jcc，这里我安装的 jcc 版本是 2.21，注意不可以通过 pip 和 easy_isntall 等工具安装，必须手动安装，因为我们要修改当中的配置文件。

先通过 svn 下载 jcc 的源代码，之后找到 setup.py 把期中的 JDK 地址手动指定为'darwin': '/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home',即可。

下一步，复制缺失头文件从 Xcode 的源码地址中找到jni.h等文件，一并复制到jcc/sources中（这一步，貌似也可以从jcc/build/lib.macosx-10.5-fat-2.7/jcc/sources中获取到。）

然后在 jcc 的目录下 terminal 中执行 python build setup.py 和 sudo python install setup.py

5.安装Pylucene 4.7.2

主要是因为pylucene 3.x 在jcc 4.21和 jdk1.6的环境下安装一直失败，所以被迫选择了这个版本。3和4在包结构上有较大的变化，很多时候必须要在4中手动 import 很多东西才可以。在建立索引的时候也有一些顺序上的不同。

首先需要安装 ant，可以通过 homebrew 或者 macports 这种源代码包管理工具来快捷安装。（类似 linux 下的 apt-get）

剩下的安装步骤和官网给出的一致，make ; sudo make install 即可。

5.2.关于页面的不同编码问题

困难是很多网站 charset 的位置不同，有的是单独的 meta，有的是在 content-type 属性中，同一个网站的不同页面也不是一样的规则，而且有些 gbk 的页面其实内部仍然有非 gbk 集合的字符。一开始遇到这个困难的时候想到利用 chardet 来解决，但是发现 chardet 的速度极慢（也不怪它，毕竟它的用处是在于分析常规文本文档的编码，没有针对html代码进行优化。）而且错误率也比较高（效率和准确性只能守着一个。）

然后我就想到可以利用正则手工去解析 charset，可是耗时耗力，效果不耗，因为不同网站规则各异。

最后偶然有一天看 bs4 的文档的时候，发现 bs4提到，它会把所有的编码都转换成 unicode 再进行处理，调用输出函数的时候返回的是 unicode 编码，这个非常棒，但是还是会有一些报错信息。

同时，我想到的另外一个比较投机取巧的方法，我发现大部分网页只有一个 charset，所以只要我们判断这文档中是否出现了 GB 或者 gb 这两个字即可。因为 gb2312是 gbk 的子集，所以一旦我们遇到 GB 或者 gb 我们就将他们认为是 gbk 编码，否则就是 utf-8。

这里要提到，并不是所有的网页都只有一个 charset，尤其是百度的网页，可能会在 js 中也提到 charset utf-8 gbk 等等信息，导致误判。但是根据我的统计发现，出现 utf-8并不一定是 utf-8，但是一旦出现 gbk 或者gb2312，则很大的频率是 gb 或者 gbk。

6. 实验扩展（Extensions）

利用 dummy库快速编写并行，从而提高离线代码运行速度。

```
def MultiProcessingCleanAndSave():
    f = codecs.open('infoIndex.txt','r',encoding='utf-8')
    files = []
    for line in f.xreadlines():
        files.append(line.split()[0])
    f.close()
    from multiprocessing import Pool
    from multiprocessing.dummy import Pool as TPool
    start = time.time()
    p = TPool(10)#设置线程池
    p.map(CleanSegAndSave,files)
    p.close()
    p.join()
    print time.time()-start
MultiProcessingCleanAndSave()
```

比如这里，我们就把清洗文本的函数进行了并发式运行。同样在建立索引的时候我们也可以进行类似的写法。

7. 总结（Conclusion）

这次实验中让我对 lucene 有了一个整体的了解，也对信息检索引擎的最最基本的工作流程有了感性的认识，不仅仅是停留在概念的理解上而是深入到每一个细节去展开体验了检索信息的各个步骤（除了评分和排序）。

这次实验我的很多收获来自于和同学们的交流，比如和骆铮同学关于 lucene4.x 与3.x 区别的交流让我省去了很多查文献的时间，和尚靖桓同学交流了关于分词索引流程的实现，让我的思路变得非常清晰，和罗昊然同学交流了 chardet 的一些弊端，加深了对它的认识，等等等等。

另外一部分收获就是帮其他同学 debug 和提供思路、建议的过程，让我对各种异常有了见识，以后可以防范于未然，也增强了我自己思路的逻辑性和完整性。比如帮其他同学解决编码错误的时候，发现最好加上'ignore'参数，否则即使编码是对的也会出现各种问题。再如，在群里给周翔同学提供用前后通配符的形式搜索 url 这个方案之前，我自己并没有这个意识去加强这种搜索的实现。

总之，助人助己，收获不浅。

8. 参考（References）

*Lucene*搜索方式大合集:

http://www.cnblogs.com/linjiqin/archive/2013/06/08/3125861.html

http://lucene.apache.org/

http://stackoverflow.com/questions/26570125/error-installing-pylucene-jcc-on-osx

https://support.apple.com/kb/DL1572?viewlocale=en_US&locale=en_US

http://www.importnew.com/12715.html

衷心感谢助教们和老师付出的时间和精力。

林禹臣

5140309507

10月27日，2015年