上机实验四 & 五
实验报告

刘瀚文
517030910294

2018 年 10 月 25 日

目录

Part I

实验四

Chapter 1

实验准备

1.1 环境配置

1.1.1 Lucene 安装

安装 PyLucene

使用 PPT 中 miniconda 的方法安装。

1.2 背景知识

1.2.1 Lucene

Lucene 是一个高效的，基于 Java 的全文检索库。先建立索引，再对索引进行搜索的过程叫全文检索 (Full-text Search)。

1.2.2 全文检索

索引建立

将现实世界中所有的结构化和非结构化数据提取信息，创建索引的过程。

搜索索引

得到用户的查询请求，搜索创建的索引，然后返回结果的过程

1.3 基础应用

1.3.1 Lucene——中文分词

由于汉字词条长度主要在 2-4 之间，StandardAnalyzer 的词汇单元与汉语中词相差甚远。CJKAnalyzer 虽然在某种程度更符合汉语的习惯，但是这样分词使得每个汉字都在两个词语中，使得词语的效率只有 50% 左右。

让 Lucene 支持中文分词，有两种做法：一种是实现自己的 Analyzer，一般需要实现自己的 Analyzer，Filter，Tokenizer 类；一种是用现有的分词库，将文本先以空格方式分好词后，再给 WhitespaceAnalyzer 或 SimpleAnalyzer 这些英文分词器处理（他们以空格做为分割分词）

本实验使用中文结巴分词然后传入英文的分词器。

1.3.2 Lucene——网页预处理

网页源代码中包含 HTML tag（例如 <html>,<body> 等），在加入 lucene 前，可以用 BeautifulSoup 等库过滤文档中的 HTML tag。

本实验使用漂亮的汤的方法过滤 Tag

Chapter 2

实验过程

2.1 实验需求

2.1.1 实现一个中文网页索引与搜索程序

爬取一定数量（>5k）的中文网页（可利用之前实验爬取的网页），修改 IndexFiles.py 和 SearchFiles.py，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。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)

2.2 实验过程

2.2.1 爬取网页————pachong.py

使用之前实验的基本代码，使用多线程（20 个线程），以'https://www.zhihu.com'为起点爬取超过 5000 个网站，保存在‘html_hhh_zhihu’的文件夹内，并且把文件名字和对应的 url 信息保存在 index_hhh_zhihu.txt 内。

在本次实验中，创新加入代码，使得爬取的网站都为 html/txt 格式。

```
def get_page(page):
  try:
    response = urllib2.urlopen(page, timeout=25)
    HttpMessage = response.info()
    ContentType = HttpMessage.gettype()
```

```
        if "text/html" != ContentType:
          return None
        else:
          print 'downloading page %s' % page
          content = response.read()
      except(urllib2.URLError, UnicodeEncodeError):
        print("Error!")
      else:
        return content
```

2.2.2 建立索引————IndexFiles.py

基础部分

```
        lucene.initVM() #初始化Java虚拟机
        ...
        store = lucene.SimpleFSDirectory(lucene.File(storeDir)) #索引文件
                                    存放的位置
        analyzer = lucene.StandardAnalyzer(lucene.Version.LUCENE_CURRENT)
        analyzer = LimitTokenCountAnalyzer(analyzer, 1048576)
        #analyzer是用来对文档进行词法分析和语言处理的
        config = IndexWriterConfig(Version.LUCENE_CURRENT, analyzer)
        config.setOpenMode(IndexWriterConfig.OpenMode.CREATE)
        writer = IndexWriter(store, config)
        #创建一个IndexWriter用来写索引文件
        ...

        for root, dirnames, filenames in os.walk(root):
        #遍历testfolder下的文件
```

FieldType

```
        #文档的文件名及路径的FieldType

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

```
doc = Document()
#创建一个Document代表我们要索引的文档

if len(contents) > 0:
  doc.add(Field("contents", contents, t2))
#将不同的Field加入到文档中。一篇文档有多种信息，如题目，作者，修改
                            时间，内容等。不同类型的信息用不同的
                            Field来表示，在本例子中，一共有三类信
                            息进行了索引，一个是文件路径，一个是文
                            件名，一个是文件内容。
writer.addDocument(doc)
#IndexWriter调用函数addDocument将索引写到索引文件夹中
```

额外的附加之处

title 标签的获取　使用 Beautifulsoup（漂亮的汤）进行处理

```
soup = BeautifulSoup(contents, "html.parser")
title = soup.head.title.string
```

对于爬虫获取的网站进行预处理　预处理并且使用结巴分词

```
contents = ''.join(soup.findAll(text=True))
seg_list = jieba.cut(contents)
contents = " ".join(seg_list)
```

把需要建立索引的项目加入 FieldType　name->t1 path->t1 title->t3 contents->t2

```
doc = Document()
doc.add(Field("name", filename, t1))
doc.add(Field("path", path, t1))
doc.add(Field("title", title, t3))
if len(contents) > 0:
  doc.add(Field("contents", contents, t2))
else:
  print "warning: no content in %s" % filename
  writer.addDocument(doc)
```

2.2.3 实现搜索————SearchFiles.py

设置读取索引

```
STORE_DIR = "html_hhh_zhihu_index"
```

使用爬虫时建立的 index.txt 找到匹配的 url

```
set = []

file1 = open("index_hhh_zhihu.txt")
for line in file1.readlines():
  a, b = line.strip('\n').split()
  set.append((a, b))

name = doc.get("name")

for i in set:
  if i[1] == name:
  url = i[0]
```

设置输出格式及个数

```
query = QueryParser(Version.LUCENE_CURRENT, "contents",analyzer).parse(
                                    command)
scoreDocs = searcher.search(query, 2).scoreDocs

print 'path:', doc.get("path"), 'title:', doc.get("title"), 'url:', url, '
                                    name:', name
```

Part II

实验五

Chapter 3

实验准备

3.1 背景知识

3.1.1 site

site 为网站对应的域名，搜索时使用 urlparse 来进行划分已获取的 url，之后使用 urlparse 的 netloc 属性获取 net_location, 并在之后使用对应的分割符号，输出合乎要求的 site（域名）。

同时用于在实验四中 url 的输出，是在获取 name 之后匹配输出。而在实验五中需要提前获取对应的 url，以获取 site。所以在实验五中也会重新书写这部分的代码，以实现搜索。

Chapter 4

实验过程

4.1 Part1

4.1.1 Lucene——组合查询

在多个域组合查询

在 google 中可以通过“site:”对搜索的网站进行限制。

```
querys = BooleanQuery()
query_content = QueryParser(Version.LUCENE_CURRENT, "contents",
                                    analyzer).parse("Christian")
#所有内容中包括“Christian”的文件
query_title = QueryParser(Version.LUCENE_CURRENT, "title", analyzer).
                                    parse("autobiography")
#所有标题包括“autobiography”的文件
querys.add(query_content, BooleanClause.Occur.MUST)
querys.add(query_title, BooleanClause.Occur.MUST)
#合并以上两个查询
```

分割特异化搜索函数

parseCommand 函数，并在其中加入结巴分词进行分词

```
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 = ' '.join(jieba.cut(command))
            command_dict[opt] = command_dict.get(opt, '') + ' ' + i
      return command_dict
```

额外的代码呈现

```
            soup = BeautifulSoup(contents, "html.parser")
            title = soup.head.title.string

            contents = ''.join(soup.findAll(text=True))
            seg_list = jieba.cut(contents)
            contents = " ".join(seg_list)

            file.close()
            doc = Document()
            doc.add(Field("name", filename, t1))
            doc.add(Field("path", path, t1))
            doc.add(Field("title", title, t3))
            doc.add(Field("url", url, t4))
            doc.add(Field("site", site, t5))
            if len(contents) > 0:

            doc.add(Field("contents", contents, t2))
```

4.1.2 实验结果展示

```
Query:黑子 site:sina.com.cn

Searching for: 黑子 site sina com cn
2 total matching documents.
path: html_lab5/httpgd.sina.com.cnnewsb2018-10-24detail-ihmuuiyw6855410.
                                    shtml
title: 港珠澳大桥今日正式运营：打破海上桥梁工程极限_新浪广东_新浪网
url: http://gd.sina.com.cn/news/b/2018-10-24/detail-ihmuuiyw6855410.
                                    shtml
```

```
name: httpgd.sina.com.cnnewsb2018-10-24detail-ihmuuiyw6855410.shtml
path: html_lab5/httpslide.sports.sina.com.cnkslide_2_786_196850.html
title: NBA史上十大有黑帮背景的社会人：哈登在列 安东尼仅能排第三_高清图集
                                        _新浪网
url: http://slide.sports.sina.com.cn/k/slide_2_786_196850.html
name: httpslide.sports.sina.com.cnkslide_2_786_196850.html

Hit enter with no input to quit.
```

4.2 Part2

4.2.1 图片搜索

搜索引擎通过描述图片的文本信息搜图问题的主要难点就是爬取图片时保存对应的图片链接内容和图片周围的文本，但是和之前的实验本质上并没有太大的区别，所以细节的描述不再重复。

4.2.2 IndexFiles

getinfo 函数

```
def getinfo(img):
  try:
    contents = img['alt']
  except:
    try:
      contents = img.parent['title']
    except:
      try:
        contents = img.parent.nextSibling.h4.string
      except:
        return None
  return contents
```

建立索引 1

```
insf = open('indeximg.txt')
ins = insf.readlines()
```

```
    for root, dirnames, filenames in os.walk(root):
      for filename in filenames:
        #if not filename.endswith('.txt'):
        #    continue
        print "adding", filename
        try:
          path = os.path.join(root, filename)
          for l in ins:
            l = l.split('\t')
            if filename == l[1].rstrip('\n'):
              url = l[0]
```

建立索引 2

```
soup = BeautifulSoup(file.read())
imgurls = soup.findAll('img')
title = soup.head.title.string
file.close()
for img in imgurls:
  imgurl = img['src']
  strs = getinfo(img)
  seg_list = jieba.cut(strs)
  contents = ' '.join(seg_list)
  doc = Document()
  doc.add(Field("imgurl", imgurl, t1))
  doc.add(Field("urltitle", title, t1))
  doc.add(Field("url", url, t1))
  if len(contents) > 0:
    doc.add(Field("contents", contents, t2))
  else:
    print "warning: no content in %s" % filename
```

4.2.3 SearchFiles

搜索

```
    query = QueryParser(Version.LUCENE_CURRENT, "contents",
          analyzer).parse(command)
    scoreDocs = searcher.search(query, 3).scoreDocs
    print "%s total matching documents." % len(scoreDocs)
```

输出搜索

```
        for i, scoreDoc in enumerate(scoreDocs):
        doc = searcher.doc(scoreDoc.doc)
        print '----------------------'
        print 'imgurl:', doc.get("imgurl")
        print 'url:', doc.get("url")
        print 'urltitle:',doc.get("urltitle")
        print 'score:', scoreDoc.score
```

4.2.4 实验结果展示

备注一下：我的代码中设置的就是输出三项，这样可以方便观察输出，也不影响对于输出结果的检查:)

```
Query:上海

Searching for: 上海
3 total matching documents.
----------------------
imgurl: http://n.sinaimg.cn/sh/transform/200/w100h100/20180424/mmoW-
                                        fzqvvsa6313017.jpg
url: http://sh.sina.com.cn/
urltitle: 新浪上海_最新鲜的新闻资讯生活_新浪网
score: 3.23087024689
----------------------
imgurl: http://n.sinaimg.cn/sh/20170417/a31H-fyeimqc4367681.jpg
url: http://sh.auto.sina.com.cn/
urltitle: 上海车市_上海最权威上海最新汽车资讯_新浪上海汽车_新浪汽车_新浪上海
                                        _新浪网
score: 2.42315268517
----------------------
imgurl: http://i3.sinaimg.cn/gm/project/netgame200/grey.gif
url: http://games.sina.com.cn/
urltitle: 新浪游戏_最新网游,手游,单机游戏资讯,排行,下载_大型中文游戏媒体
score: 2.37477207184

Hit enter with no input to quit.
```

Part III

实验总结

本次实验在之前实验的基础上，进行研究和拓展。爬取网页，获取域名；爬取图片获取图片的内容信息，其实本质上实验五中爬取的内容不同，但是本质上是相同的。

本次实验对于我个人的难度主要是在虚拟机的环境总是配置不能成功，最后选择重新装虚拟机，严格按照最初的 ppt 重新操作，这样才在无报错的情况下完成了环境的配置。之前的内容也在新的虚拟机上成功运行。我想这种可以容易的重装也是虚拟机的一个优势吧。

继续努力，不断进步！争取在实践中获得更多的知识！