实验4 文本特征抽取和索引1

518030910303 纪喆

2019 年 10 月 23 日

1 实验准备

1.1 实验环境

本次实验在 **Ubuntu 14.04 LTS**系统中进行，使用的是**Python3**编程语言。本次实验使用的工具有：vim, Python3, Python2, pylucene 本实验使用了实验3实现的Bitarray，BloomFilter，以及单线程与多线程爬虫代码。使用Python3的脚本有：

1. Bitarray.py
2. BloomFilter.py
3. crawler.py

使用Python2的脚本有：

1. IndexFiles.py
2. Search.py

1.2 实验目的

利用之前的实验所实现的网络爬虫爬取大于5k个网页，使用lucene在本地对其进行索引，实现搜索功能。

1.3 实验原理

1.3.1 Bloom Filter主要作用

Bloom Filter是一中二进制向量数据结构。它具有很好的空间和时间效率，用来检测一个元素是不是集合中的成员。在本次实验中，网络爬虫在判

断是否已经爬取过某一特定URL时使用了BloomFilter，相对于python 自带的list以及判断元素存在有了更好的效率，在爬取网页数目较多的本实验中优势显著增加。

1.3.2 索引的建立

URL与HTML文件对应关系的确定 在建立索引时，需要获取每一个网页文件以及其对应的URL。通常做法为，通过爬虫生成的index.txt里面包含的HTML文件名与URL的对应关系获取，但这个过程对于每一次查询的时间复杂度为O(n)，在大量网页的实验目标下，显然表现得不够好。本实验最终采用的做法为，在爬虫爬取并保存HTML文件时，将URL作为文件的第一行保存起来。显而易见，这种添加对于每一个网页来说时间复杂度只有O(1)。而在建立索引的时候读取文件只需要读取第一行获取URL，之后读取整个文件获得HTML内容，对一个网页来说获取URL的复杂度仍然是O(1)。这使得建立索引时无需依靠查找index.txt，较好的提高了程序效率。

建立过程 从文件中提取该网页的URL以及网页的title，再加上整个文件内容，文件名称，文件目录，一起存到一个doc里面。即，每一个doc里面有5个Field。其中title这一Field的属性与其他四个不同，它被设置为索引项。在后续的搜索中，被搜索内容实际上就是这些网页的title。之所以将title作为索引项而不是网页内容，是考虑到本实验用爬虫爬取到的网页内容比较繁杂，网页内容有许多是干扰信息。如果以网页内容作为搜索对象，搜索出的网页比较混乱，而网页的名称通常能够比较准确地反映网页主题，能够更好的切合用户想要搜索的对象。

1.3.3 搜索的实现

使用lucene的searcher，采用与indexer相同的分词工具解析搜索词，并在建立的索引中搜索title的相关值。通过scoreDoc对搜索结果中的每一个文档进行打分，并按照从高到低的顺序打印（为了防止过度刷屏，限制最多只输出50个doc结果）。

2 实验过程

2.1 网页爬取

本次实验前前后后进行了许多次网页爬取，采用了多线程爬虫、单线程广度优先搜索爬虫，也尝试过不同的URL种子。在权衡被爬取的网页质量，内容丰富度，网页爬取速度等多种因素后，最终放弃了多线程爬虫的速度优势，采用了单线程广度优先爬虫，使用了百度新闻主页为URL种子进行爬取。原因如下：

1. 多线程爬虫只能使用类似于深度优先的顺序爬取网页，往往很快就会偏离我们想要爬取的内容，而广度优先搜索则会把大部分期望爬取的网页爬下。

2. 百度新闻主页有大量新闻网页链接，而且网页比较“干净”，能够较好地控制爬取内容。

另外，本次爬虫做了一些改进：

1. 在requests.get获取到网页时，先从HTTP相应的头中判断网页的内容（’Content-Type’)是否为HTML文件。只有HTML文件才会被函数返回网页内容。

2. 为了防止爬取到大文件下载URL而将爬虫困住，在上一条的基础上，在requests.get中添加参数stream为真，意味着网页内容不会立即被下载，只有判断好content-type之后需要返回text是才会下载。

3. 在保存文件时不单纯保存HTML文件内容，在文件开头会记录该网页的URL，方便后续建立索引时使用。

4. 在爬取的同时，利用requests中的apparent_encoding通过网页内容判断网页编码格式，避免编码错误。在保存文件时，统一使用utf8编码保存，竭力避免乱码问题。

注：爬虫产生的index.txt文件实际上对本次实验的完成没有什么用处，但是由于“历史原因”，以及方便助教查看被爬取网页，该文件的生成被保留了下来。

2.2 索引建立

由于lucene自带的中文分词工具比较无力，StandardAnalyzer会把中文逐字分开，导致搜索时结果混乱，本次实验借助jieba（结巴）中文分词工具，将分词过的文本用空格联合起来，让lucene用WhitespaceAnalyzer直接使用结巴分词结果进行解析。另外，由于本次实验采用了网页title作为索引（具体原因见实验原理部分），因此进行分词时，只需要处理网页标题即可。设置Field属性时，创建两种Field，一种专门存储网页标题，开启索引，另外一种则只需要保存，设置代码如下。

```
    t1 = FieldType()
    t1.setIndexed(False)
    t1.setStored(True)
    t1.setTokenized(False)

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

其中t2就是储存title的FieldType。设置中的最后一行表示将t2作为indexes document，并且记录词频和词出现的位置，用于scoreDoc的打分。真正创建索引时，遍历整个html文件，从文件的第一行获取URL，然后在文件的剩余部分寻找<title>。最后把这些内容全部以t1的type存入一个doc中，然后向索引提交doc即可。

2.3 搜索实现

搜索时，接收输入的字符串，用结巴（jieba）分词工具进行分词，然后在title的field中进行搜索。并按照作业要求输出相应内容，两个doc输出中间用多个连字符分开。需要注意的时，search.py中的StandardAnalyzer也需要改为WhitespaceAnalyzer，否则搜索结果与期望的相关度会大打折扣。搜索结果如图：

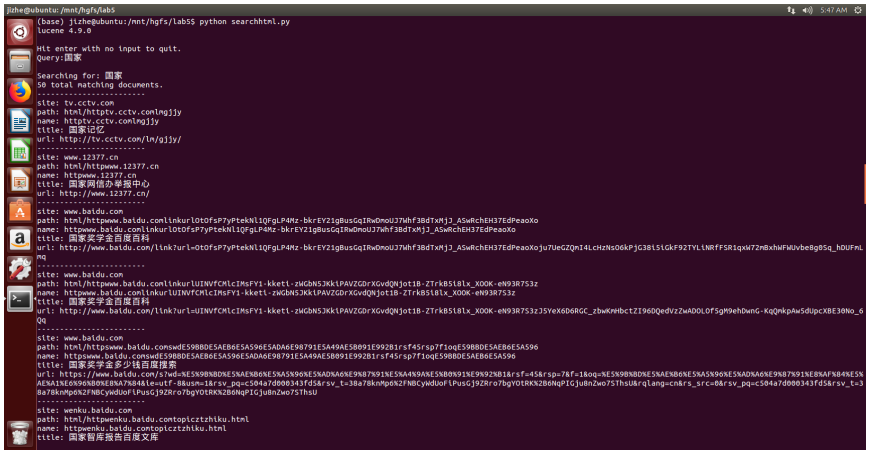

图 1: 搜索结果示例

3 实验总结

3.1 概述

本次实验实现了对本地文件的索引建立以及查找功能，主要使用的工具就是lucene。经过本次实验，我对大数据中的搜索有了更深更真切的认知。

3.2 问题与解决

问题 在初始阶段，建立索引之后，搜索结果与搜索关键词相关度很低

解决 通过查阅资料发现，是中文字符串被StandardAnalyzer解析成了单独的汉字造成的，在StandardAnalyzer的作用下结巴的分词结果已经没有效果了。因此将解析器改为WhitespaceAnalyzer，只在空格处分词，有效提高了相关度。

问题 在实验初期阶段，搜索出的内容中文标题现实乱码。

解决 经过考虑，应该为网页编码问题。由于从文件中读取后再判断文件编码格式略显麻烦，本次实验直接在爬取阶段完成统一字符编码的任务，将所有网页保存为utf8编码格式。

3.3 感想

本实验用之前实验所构建的网络爬虫爬取网页，再通过pylucene在本地建立索引并实现搜索功能。这样一来，爬虫爬取的网页才能在该出现的时候出现在用户面前，网页内的信息才有它应有的价值。不过，相对于一个成熟的搜索引擎，本实验实现的功能比较局限，不能支持site:xxxx等限定，而这便是下一个实验的内容。