信息检索技术大作业报告

第一阶段 建立索引

电 61 王铮澄
2016010910

一、作业任务：

针对 CACM 数据集，建立文档的倒排索引，以支持后续从系统中快速查找所需数据。

二、总体思路：

0、语言选择

所有工作使用 python 编写。

1、观察数据集中文档内容与格式

数据集由 3204 个网页文件组成，每个网页内容相似，由 HTML 编写。观察文档，发现除了 HTML 结构性的语言之外，在文档末尾会有数量不一但是格式统一的数字行列，如下图所示：

```
1788    5    3189
3189    5    3189
3189    5    3189
3189    5    3189
1006    6    3189
1007    6    3189
205     6    3189
3189    6    3189
```

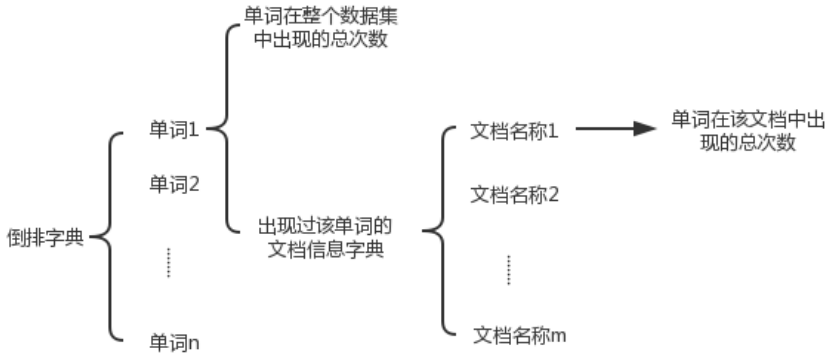
我认为这部分数字没有搜索价值，因此在挑选文本有效部分时将其排除在外。文档有效部分提取考虑使用正则表达式。

2、建立分词列表

针对每个文本，在提取出有效部分后，为了索引需要，必须进行分词并且建立起列表。由于文档都由英文组成，考虑使用简单的空格分词。此外考虑到大小写影响，统一将所有字母转换为小写。

3、构建倒排字典

由于 python 有字典这种数据结构，非常适合进行倒排索引的建立。考虑建立拥有如下结构的三层字典：



4、用文件保存建好的倒排字典

考虑使用 json 格式保存字典，便于后续使用时倒排索引的恢复。

三、具体实现

为了增强程序的可复用性，使用 python 面向对象方式编写，一共进行 3 个类的封装，具体构成为：

1、 Documents 类：

- 拥有属性：

文档名称，有效文本，未去重单词列表

- 拥有方法：

初始化属性，打开文件，文档拆分

其中文档拆分为核心函数，主要完成两部分功能，一是使用正则表达式从全文中匹配出有效文档部分，主要针对文本中的 HTML 格式化语言以及文档结尾的数字串进行拆分，并且对换行符，制表符等进行分割，均可使用 python 自带函数以及 re 模块相应函数完成。二是使用空格对文本进行分词。

2、 ReverseIndex 类：

- 拥有属性：

停用词表，停用词字典，数据库文件名列表，倒排字典

- 拥有方法：

初始化属性，构建停用词字典，遍历数据集文件夹，构建倒排字典，写入文件

为了更快速地对每个文档的单词列表进行停用词筛查，避免每次都进行遍历，根据停用词表建立了停用词字典，每当遇到一个单词，就查看其是否为停用词字典的一个键，可以省去对停用词表进行遍历的时间。

使用 python 自带函数，得到 cacm 文件夹下所有的文件的名称，建立列表。

对每个文件建立 Documents 类，调用 Documents 类的方法进行文本拆分和单词表建立，然后对每个文件的单词表进行遍历，然后按如下流程建立倒排字典：

1） 如果该单词不在倒排字典的键中，则将其加入字典，并将总次数置为 1，将该文档名称加入文档信息子字典，并将子字典中的出现次数置为 1；

2） 如果该单词已经存在于倒排字典的键中，则将总次数加 1，检查该文档名称是否已经出现在文档信息子字典中；

3） 如果该文档名称已经存在于子字典中，也就是说在该文档中不是第 1 次出现这个单词，该文档名称对应的次数加 1 即可；

4） 如果该文档名称未存在于子字典键中，也就是说在该文档中第 1 次出现这个单词，则将该文档名称加入子字典，同时该文档名称对应的次数置为 1；

建立倒排字典之后，将该字典写入文件中。

3、ReverseIndexApp 类：

该类主要将上述类和方法进行整合，写在 run()函数中，便于直接调用。

四、性能测试

使用 python 中 time 模块的 process_time()函数进行计时，测试建立索引需要的总时间，包括文档处理，倒排字典建立，文件读写在内的所有时间。一共进行 5 次测试，结果如下：

次数	时间/s
1	1.266
2	1.250
3	1.297
4	1.422

5	1.313
平均	1.310

平均用时 1.310s，能够较为快速地建立索引。

五、文件说明

cacm 文件夹：数据集

RevIndex.txt：倒排字典文件

Documents.py：Documents 类源代码文件

ReverseIndex.py：ReverseIndex 类源代码文件

ReverseIndexApp.py：ReverseIndexApp 类源代码文件