哈 尔 滨 工 业 大 学

<<信息检索>>

实验报告

(2019 年度春季学期)

姓名:	董雄
学号:	1160300415
学院:	计算机学院
教师:	张宇老师

目录

实验三： 企业搜索系统的设计与实现

一、实验目的（摘抄自实验指导）

本次实验目的是对企业有所系统的设计与实现过程有一个全面的了解。本次实验设计的内容包括： 对数据建立索引，实现文档的搜索，并对检索结果排序；实现企业搜索中的分权限访问

二、实验内容

3.1 文本集合进行处理、建立索引

任务描述：本次实验使用到的数据是实验 1 中爬取的网页数据。同学们首先要对 1000 个网页的网页内容建立索引，其次也要对爬取到的所有附件文档建立索引。 然后实现一个简单的检索系统，实现数据和文档检索（可以分开实现，也可以合二为一），并且能够精确的对检索结果进行排序。 这一部分要求同学们做成简单的 UI 应用或者是 Web 应用，来保证易用性。

3.2 分权限访问

任务描述：同学们可以自己定义多种不同的“企业角色”（至少 4 种），这些角色对数据或文档的访问权限不同， 然后为每条数据增加访问权限。 然后在现有检索系统的基础上加入分权限访问功能，使得不同角色的用户在使用检索系统时，只能看到自己具有访问权限的那部分内容。同时在 3.1 的基础上对应用界面改进，便于切换企业角色。

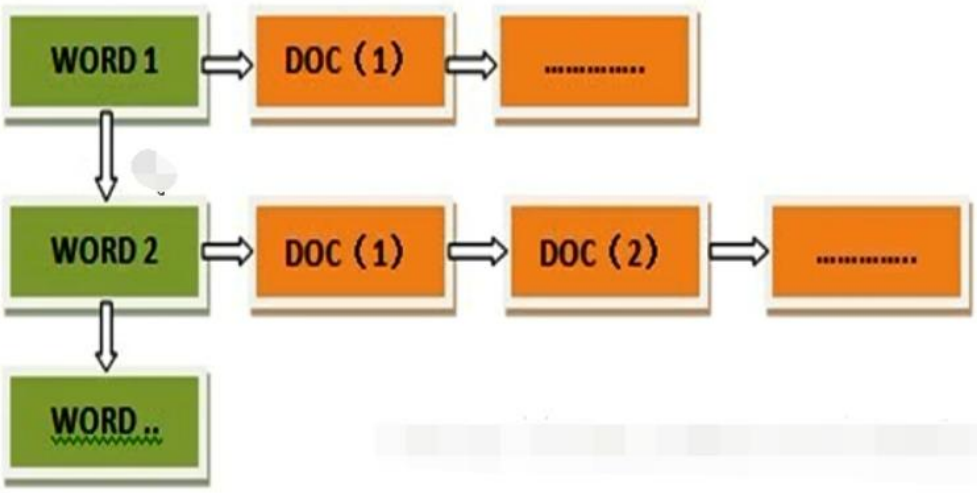
分权限访问的实现一般有以下两种模式：

1. 检索条件控制：对检索条件加以限制，仅在用户访问权限内的文档中检索。
2. 检索结果过滤：检索结果的后处理，过滤掉用户不具备访问权限的结果。

三、实验过程和结果

1.建立索引

使用倒排索引



针对文本集合进行处理，首先想到的是使用建立倒排索引的方法。倒排索引是把关键字作为键值，并通过关键字来查询带有此关键词的文档 ID 列表的方法。

所以，我们根据以下概念建立倒排索引：

单词词典(Lexicon)：搜索引擎的通常索引单位是单词，单词词典是由文档集合中出现过的所有单词构成的字符串集合，单词词典内每条索引项记载单词本身的一些信息以及指向“倒排列表”的指针。

倒排列表(PostingList)：倒排列表记载了出现过某个单词的所有文档的文档列表及单词在该文档中出现的位置信息，每条记录称为一个倒排项(Posting)。根据倒排列表，即可获知哪些文档包含某个单词。

倒排文件(Inverted File)：所有单词的倒排列表往往顺序地存储在磁盘的某个文件里，这个文件即被称之为倒排文件，倒排文件是存储倒排索引的物理文件。

本次实验中，建立倒排索引的步骤如下：

Step1 首先构建单词词典，对文档进行分词，去停用词处理，然后将当前文档的 pid 存储在当前单词的倒排列表中，同时还要记录该单词出现的频率信息(TF)，方便之后进行查询时的问题答案相似度计算。“文档频率信息”代表了在文档集合中有多少个文档包含某个单词，之所以要记录这个信息，其原因与单词频率信息一样，这个信息在搜索结果排序计算中是非常重要的一个因子。由于停用词在文档中出现次数非常多，如果记录其索引的话会浪费较大的空间。同时停用词在检索系统中几乎没有什么作用，所以在建立倒排索引的时候需要先进行去停用词操作。

建立倒排索引的关键代码如下：

```
def create_new_index(filename):
    file_index = {}
    dic_list = read_json_data(filename)
    for dic in dic_list:
        pid = dic['pid']
        for sentence in dic['document']:
            for word in sentence:
                if word not in file_index:
                    file_index[word] = {pid: 1}
                else:
                    if pid not in file_index[word]:
                        file_index[word][pid] = 1
                    else:
                        file_index[word][pid] += 1
    return file_index
```

可以看到，如果词在文档中第一次出现，就把出现次数设为 1，之后每出现一次就在相应的位置加 1，直到处理完所有的文档。

Step2 接下来处理之后文档的过程中，如果发现单词已经在之前出现过，就将此文档编号、单词出现频率等信息添加在之前对应单词的文档列表中。一直执行直到处理到最后一篇文档。

最终生成的倒排索引如图：

Vocabulary	n_i	出现的文档列表
to	2	[1,4], [2,2]
do	3	[1,2], [3,3], [4,3]
is	1	[1,2]
be	4	[1,2], [2,2], [3,2], [4,2]
or	1	[2,1]
not	1	[2,1]
I	2	[2,2], [3,2]
am	2	[2,2], [3,1]
what	1	[2,1]
think	1	[3,1]
therefore	1	[3,1]
da	1	[4,3]
let	1	[4,2]
it	1	[4,2]

检索系统的实现

在建立倒排索引的基础上，实现检索的过程。

首先对查询进行分词、去停用词，然后遍历倒排索引的词典，找到每个词在所在文档中的得分。如果有多个词出现在一个文档中，就对其得分累加，每次累加的得分就是该词在文档中出现的次数，最终找出得分最高的 k 个文档返回。实现细节如下：

```
for keyword in keywords:
    if keyword not in index_dict: continue
    for pid in index_dict[keyword].keys():
        if pid not in ans_dict:
            ans_dict[pid] = index_dict[keyword][pid]
        else:
            ans_dict[pid] += index_dict[keyword][pid]
```

在倒排文档需要频繁使用检索功能时，可以考虑对词典按字母顺序进行排序，这样在搜索的时候就可以直接在 1/26 的范围内检索词典，预计可以比较大地提升效率。

但是这样存在的问题是，如果查询的关键词不在索引词典中出现，就会出现查询失败的情况，不会返回任何信息，但这是我们不希望看到的。一种解决思路是如果返回的结果长度为 0，就进行模糊查询，运用 BM25 算法找到最相关的文档集合。但是后一种方法将花费更多的时间。

```
def fuzzy_search(search_line,rate):
    doc_dicts = parse_json_data('data/seg.json')
    corpus = []
    answers = []
    for doc_dict in doc_dicts:
        words = doc_dict['document']
        corpus.append(words)
    try:
        bm25Model = bm25.BM25(corpus)
        scores = bm25Model.get_scores(question_seg)
        max_index = scores.index(max(scores))
        answer = doc_dicts[max_index]['document']
        answers.append(answer)
        return answers
    except ZeroDivisionError:
        print('捕捉到错误')
```

这里调用 BM25 模型的函数进行相似度的计算，返回相似的文档，即模糊查询的结果。

2.分权限访问

A．定义“企业角色”

由于采用的是今日哈工大的网页数据，所以定义以下四种企业角色：

校长（4），学工处（3），老师（2），学生（1），分别可以访问今日哈工大新闻网上的不同数据。然后为每条数据增加其权限，这里为了测试，就选取前 1/4 作为级别 1，以此类推，标注所有的网页数据。结果如下：

{"title":, "url": "http://today.hit.edu.cn/article/2019/04/29/66539", "paragraph": "点击微信扫一扫", "file_name": ["today_0.png"], "rate": 1}

B. 权限访问的实现—检索条件控制

A 步骤每一个字典类型都增加了“rate”字段，定义：**只有用户级别大于 rate，才可以访问该数据。**也就是检索时，要确定用户的 rate 值>数据的 rate 值，才可以进行访问。

代码实现如下：

```
photo_files = []
for result in results:
    result_int = int(result)
    photo_files += seg_dict[result_int]['file_name']
    if rate >= seg_dict[result_int]['rate']:
        e2.insert("insert",'%d' %(num)+'.' + 'title:' +
        num = num+1
print(photo_files)
```

C. 权限访问的实现—检索结果过滤

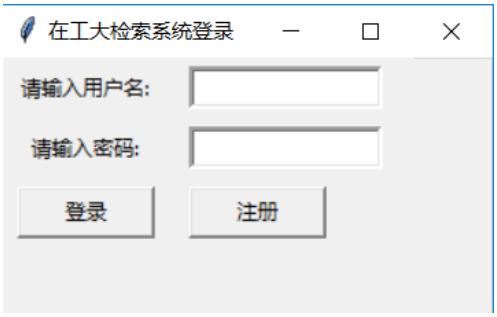
在返回的结果中，有些信息可能不应该被某类用户看到，因此定义一个过滤列表，其中包含需要对某类用户过滤掉的文件中包含的关键词。定义如下：

```
one_stop_words = ['机密','资金','涉密']
if rate == 1:
    keywords = [key for key in keywords if key not in one_stop_words]
```

然后，如果该用户等于某个级别，就过滤掉包含关键词的文档，仅仅将剩余的文档显示在结果页面中。

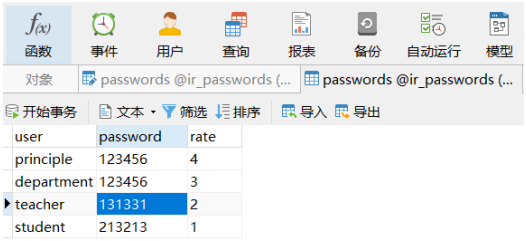
3.运行结果

系统登录界面如下



用户填入用户名和密码后，如果点击注册按钮，就会将用户名和密码注册到数据库中

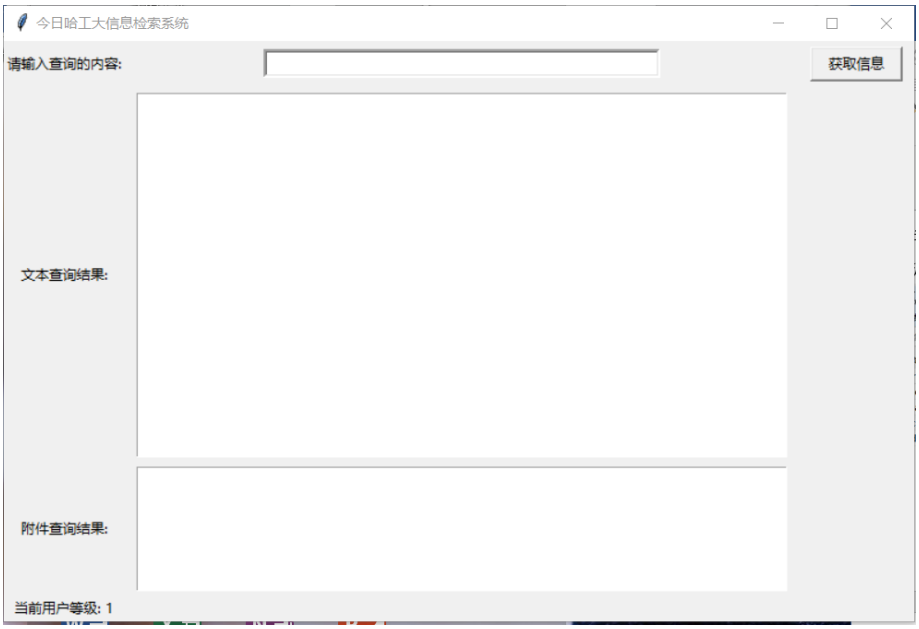
数据库定义如下：



user	password	rate
principle	123456	4
department	123456	3
teacher	131331	2
student	213213	1

如果用户点击登录按钮，后台程序会从数据库中寻找用户名，如果没有找到，则登录失败，否则对比填入的密码和数据库中的密码，如果一致，就进入检索系统页面中。

我们以学生为例进行测试，检索界面如图所示：



在左下角标注了当前用户的等级，之后将根据此等级对查询范围进行限制。

查询“材料”得到的结果如图所示，可以看到系统返回了正确的结果。

四、实验心得

通过本次实验，对于索引的建立，搜索的实现以及企业搜索系统的实现有了更多的了解。也对企业搜索中权限的控制有了更多的认识。