信息检索系统作业报告

2018211517 何泓川
2018211512 鲁嘉祺

实验目的

自己动手设计实现一个信息检索系统，数据源可以自选，数据通过开源的网络爬虫获取，规模不低于 100 篇文档，进行本地存储。中文可以分词（可用开源代码），也可以不分词，直接使用字作为基本单元。英文可以直接通过空格分隔。构建基本的倒排索引文件。实现基本的向量空间检索模型的匹配算法。用户查询输入可以是自然语言字串，查询结果输出按相关度从大到小排序，列出相关度、题目、主要匹配内容、URL、日期等信息。最好能对检索结果的准确率进行人工评价。

实验环境

python3.9

完成情况

本实验完成了 1-5 的全部要求。

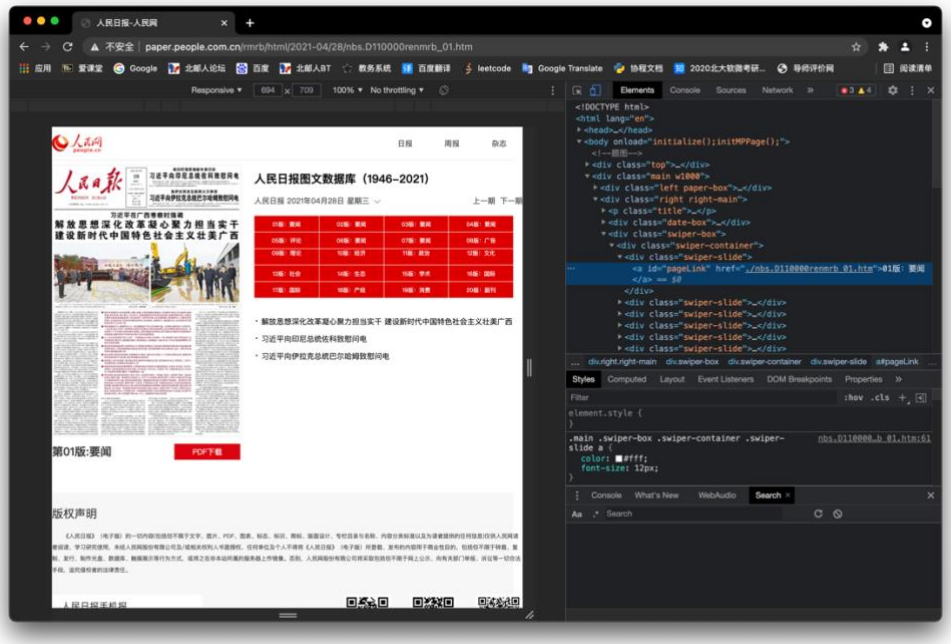
实验过程

（1） 爬取数据集

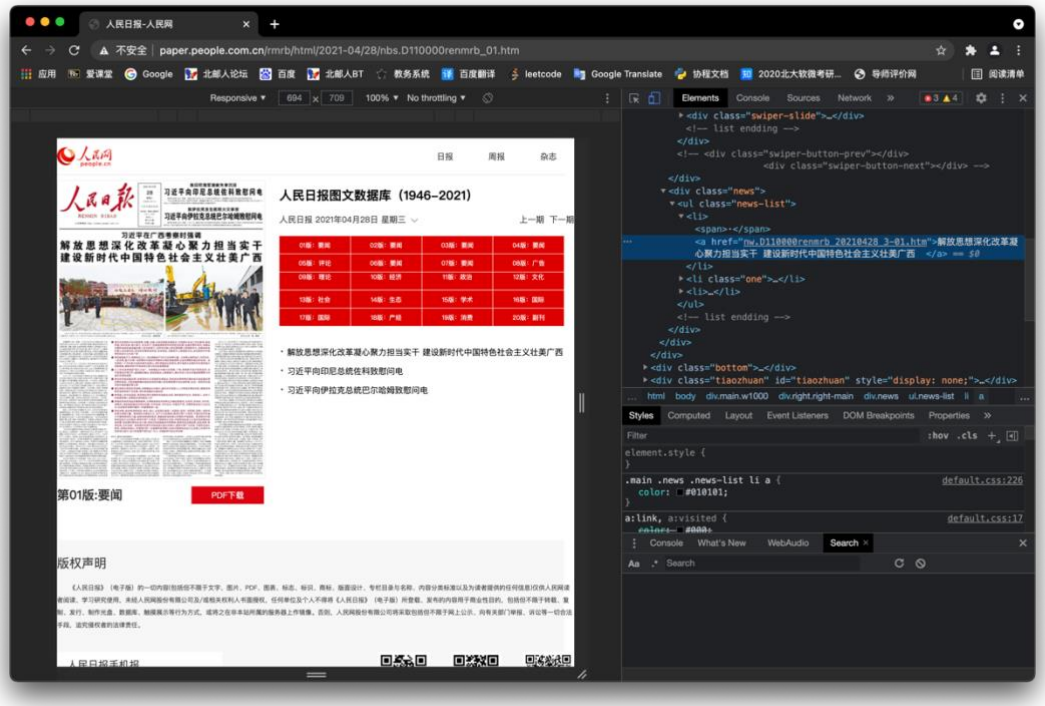
本实验共有 20 个分类，104 篇文档，如下图：

```
mac@BruceHo info_search % python3.9 spider.py
20 kind_linklist  finished
file count: 104
```

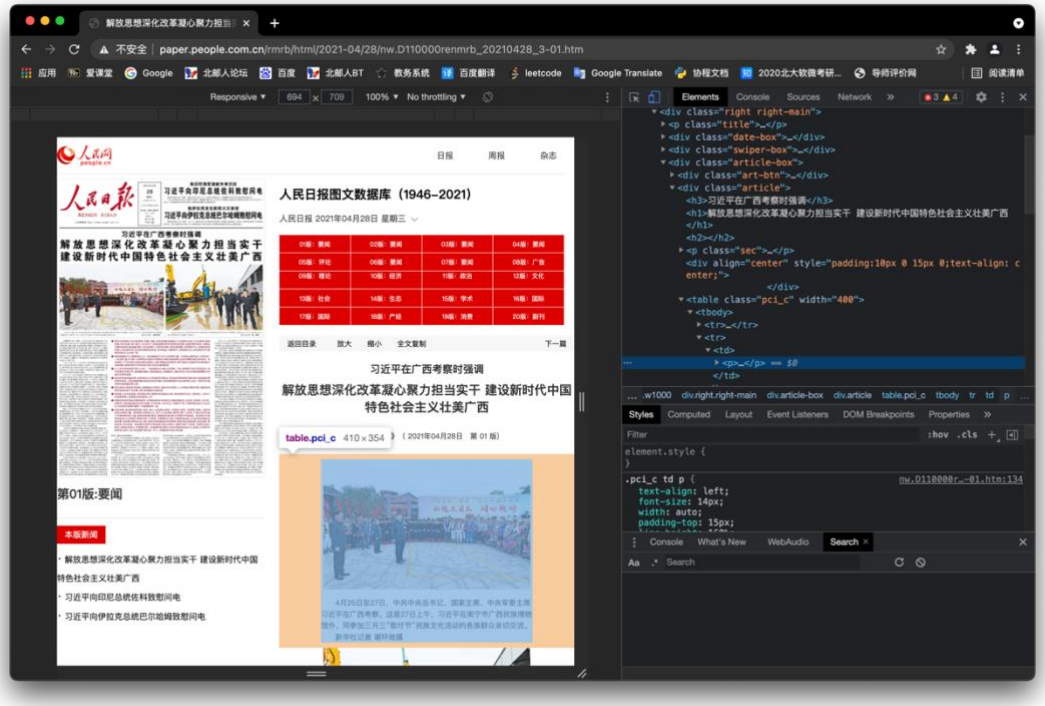
本实验中使用人民日报的文章作为数据集。先根据各类别的特征找到其对应的 url 并保存所有类别的 url，对应的函数为 spider.py 中的 getKindList 函数。下图中的类别特征为 class=’swiper-container’的 div 标签下的 class=’swiper-slide’的 div 标签。



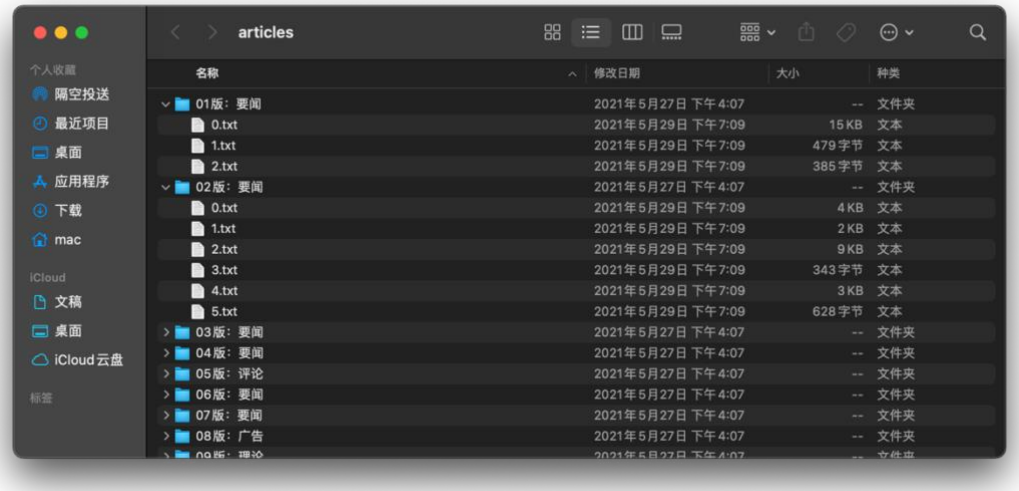
再根据一个类别中文章 url 的特征，保存所有类别所有文章的 url。相关的函数为 spider.py 中的 getTitleList 函数。下图的特征为 class=’news’的 div 标签下的 li 标签。



再访问所有文章的 url，通过文章的特征找到文章的文本内容。下图的文章特征为 id=’ozoom’的 div 标签下的 p 标签。保存所有文章的文本内容到.txt 文件。相关的函数为 spider.py 中的 saveContent 函数。



爬取所有文章的结构如下：



所有 url 信息保存在文本文件中，内容如下：

（2） 信息检索

先进行分词。分词使用 jieba 库进行分词，分词过程在 divide.py 中。再把所有文章的词向量化，如下图：

```
def gen_vector(article_list):
    bag = CountVectorizer(token_pattern='\\b\\w+\\b')
    count = bag.fit_transform(article_list)
    return bag, count
```

其中 token_patter 的正则表达式为把两个空格之间的词作为向量一个单元。

再生成倒排索引，可以通过一个单词找到所有包括它的文章、这个词在该文章中的出现次数、在该文章中的所有出现位置，如下图：

```
def gen_inverse_index(article_list, bag, array, raw_article_list):
    result = defaultdict(list)
    words = bag.get_feature_names()
    for index, value in enumerate(article_list):
        for i, word in enumerate(words):
            if array[index][i] != 0:
                position_list = [m.span() for m in re.finditer('\\b' + word + '\\b', value)]
                result[word].append((index, array[index][i], position_list))
    return result
```

做完倒排索引后可以进行查询。先读入一个字符串，如果是 exit 则结束程序，否则进入 get_result 函数计算结果。先把字符串按空格分离出来，即对每个关键词依次计算。先计算所有关键词在多少不重复的文章中出现，记为 freq。再逐个分析每个关键词。遍历每一条该关键词的倒排索引，如果这条索引的文章不在结果列表中，则加入结果列表，该文章出现的关键词个数 keyword_num 加一，该文章中关键词出现的次数 keyword_times 加一，更新该文章中关键词出现次数占所有关键词出现的文章数的比例 ratio。结果列表中也要保存该词在该文章的出现位置。如果关键词的这条索引在结果列表中，则 keyword_times 和 keyword_num 加一，更新 ratio，保存出现位置。对所有关键词遍历后把关键词向量化，并与结果列表的每一条计算相似

度 correlation（余弦值）。此时结果列表信息已经完整，按照 ratio、keyword_num、manual_point、correlation 乘积倒序排列，即位最终结果。原因是该文章中关键词出现次数更多，所有关键词次数出现文章数更少，该文章命中的关键词个数更多，人工评价权值更高，相似度越高，则该文章更匹配。人工评价权值初始值都为 1，每次查询后人工对结果进行评价，选中的文章人工评价权值乘 1.1，即下次更容易被列在结果列表前部。信息检索部分实现如下：

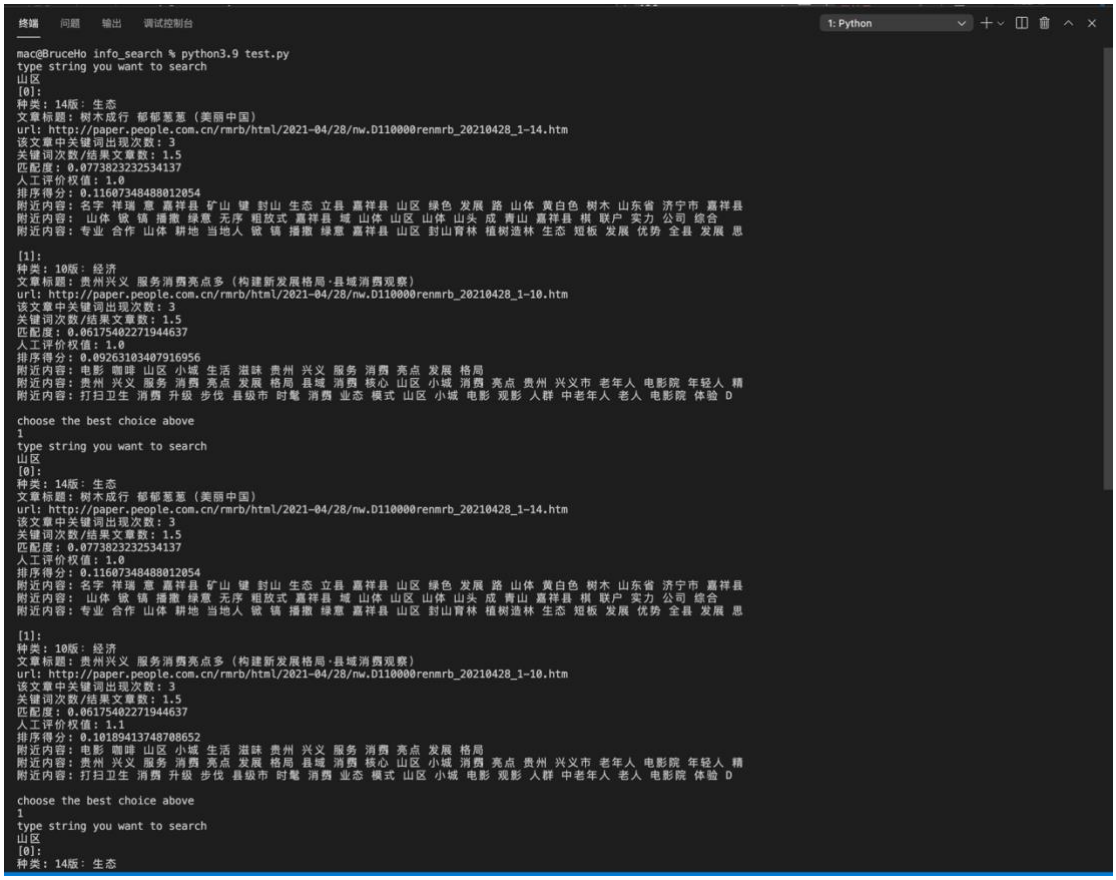
```
result_dict = dict()
for i in word_result: # 一个关键词
    if not i:
        continue
    for info in i: # 遍历该关键词出现的所有文章      BruceHo202, 5 days ago • Add files via upl
                # info是三元组(文章索引, 本词在这篇文章出现的次数, 每次出现的起止位置)
                # 每篇文章只能进入一次if 后面都是else
        if info[0] not in result_dict:
            item = resultItem(info[0], article_names[info[0]], article_list[info[0]])

            item.keyword_num += 1 # 该文章被多少关键词命中
            item.keyword_times += info[1] # 该文章中关键词出现次数
            item.ratio += info[1] / freq # 该文章中关键词出现次数 / 所有关键词出现的文章数

            item.occurence.extend(info[2])
            result_dict[info[0]] = item
        else:
            item = result_dict[info[0]]

            item.keyword_num += 1
            item.keyword_times += info[1]
            # print(item.freq)
            item.ratio += info[1] / freq


            item.occurence.extend(info[2])
            result_dict[info[0]] = item
result_list = [i for i in result_dict.values()]
search_vec = CountVectorizer(vocabulary=bag.get_feature_names()).fit_transform([search_str]
for i in result_list:
    i.correlation = calc_correlation(search_vec[0], array[i.index].A[0])
result_list.sort(key=lambda x: -x.ratio * x.keyword_num * x.manual_point * x.correlation)
return result_list
```

实验结果

搜索一个关键词，可显示所有包含该关键词的文章，出现次数多的靠前，列出相关文章信息：

如果搜索多个关键词，同时包含所有关键词的文章排序更靠前：

人工评价：开始时标题为《贵州兴义 服务消费亮点多（构建新发展格局·县域消费观察）》的文章在结果列表第二位，多次人工选择该文章后出现在了结果列表第一位。



实验结论

本实验可以完成信息检索系统的功能，根据输入的关键词找到相关文章及其信息，并按相关度从大到小排列。考虑到对环境和社会可持续发展影响的考虑，本实验数据集的选择中选择

了有生态版块的人民日报网的文章。

创新性思考和多媒体信息搜索

本实验中，由于考虑到实际的搜索引擎可以对多个关键词进行搜索，增加了多个关键词搜索的功能，对所有结果排序方法使用多个指标乘积，使相关度更高的结果排序更靠前，而不是仅考虑向量间余弦值，使相关度计算方法更完善。

```
result_list.sort(key=lambda x: -x.ratio * x.keyword_num * x.manual_point * x.correlation)
```

在多媒体（如视频、图片）信息搜索中，可以先把所有图片向量化组成图片向量库，再把输入图片也进行向量化，计算输入图片与所有图片向量库中向量的余弦值，把结果倒序排列。在本程序中，只需在 get_result 函数中直接把输入图片向量化，并计算与所有图片向量的余弦，结果倒序排序，即位图片或视频的信息检索。