信息检索系统实验报告

陈朴炎 2021211138 王唯宇 2021211133 陈锦鹏 2021211137

目录

1	实验要	求3
	1.1	基本要求3
	1.2	扩展要求3
	1.3	评分标准3
2	数据处	理4
	2.1	爬取 url4
	2.2	爬取新闻数据 7
	2.3	分词10
	2.4	构建倒排索引表13
		2.4.1 TF-IDF
		2.4.2 TF-IDF 算法步骤14
		2.4.3 倒排索引表14
	2.5	计算向量空间18
		2.5.1 构建词汇表18
		2.5.2 构建向量19
3	系统设	计及实现22
	3.1	系统设计 22
		3.1.1 概述 22

	3.1.2	模块划分	• • • • •	• • • • •	• • • •	• • • •	• • •	 • • •	• •	• • •	• • •	• • •	22
	3.1.3	详细设计		• • • •	• • • •		• • •	 • • •			• • •		22
	3.1.4	工作流程		• • • • •	••••		• • •	 			• • •		24
	3.2 系统实	现		• • • • •	••••		• • •	 			• • •	• • •	26
	3.2.1	数据加载构	莫块	• • • • •	• • • •		• • •	 			• • •		26
	3.2.2	查询处理构	莫块	• • • •	• • • •		• • •	 • • •			• • •		27
	3.2.3	检索模块		• • • •	• • • •		• • • •	 • • •	••		• • •	• • •	27
	3.2.4	显示结果		• • • •	•••		• • • •	 •••			• • •	• • •	28
	3.2.5	用户交互植	莫块	• • • • •	•••		• • •	 •••			• • •	• • •	29
4	系统运行效果	展示						 					30

1 实验要求

1.1 基本要求

设计并实现一个信息检索系统,中、英文皆可,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。用户查询输入为自然语言字串,查询结果输出按相关度从大到小排序,列出 相关度、文档题目、主要匹配内容、URL、文档日期等信息。最好能对检索结果 的准确率进行人工评价。界面不做强制要求,可以是命令行,也可以是可操作的 界面。提交作业报告和源代码。

1.2 扩展要求

鼓励有兴趣和有能力的同学积极尝试多媒体信息检索以及优化各模块算法,也可关注各类相关竞赛。自主开展相关文献调研与分析,完成算法评估、优化、论证创新点的过程。

1.3 评分标准

- 1、完成基本的信息检索功能且有对环境和社会可持续发展影响的考虑,系统能够正常运行,并提交源代码和实验报告: 60 分;
- 2、完成要求的信息检索功能且有对环境和社会可持续发展影响的考虑,系统能够正常运行,并按时提交源代码和实验报告: 61~70分;
- 3、在2的基础上, 且实验报告撰写认真、思路清晰、表达准确: 71~80分;

- 4、在3的基础上,支持检索结果准确率人工评价:81~90分;
- 5、在 4 的基础上,融入了自己的创新性思考、优化算法或对多媒体信息检索进行了尝试: 91-100 分。

2 数据处理

2.1 爬取 url

本次的实验数据我是在中国新闻网上收集的。

在即时新闻页面上可以查看往日的所有新闻,即使新闻链接为:

https://www.chinanews.com.cn/scroll-news/news1.html

页面示意图如下:

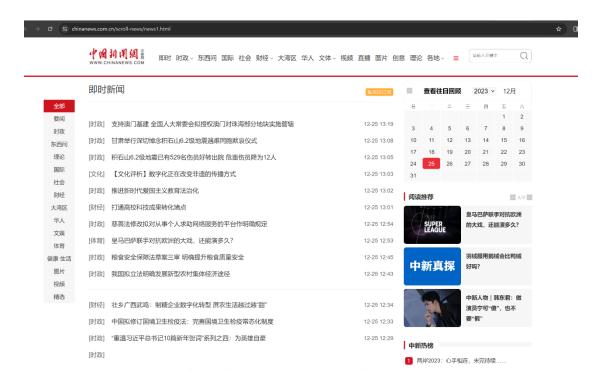


图 2-1 中国新闻网即时新闻示意图

我们可以发现 url 为

https://www.chinanews.com.cn/scroll-news/2023/1214/news.shtml

其中, https://www.chinanews.com.cn/ 为中国新闻网的基础域名,

/scroll-news/表示是滚动新闻页面,2023是年份,1214是月份和日期。

打开网页的 Elements, 我们查看这个网页的 html 格式, 可以看到, 左端的 url 都保存在<content-left>/<content list>下的<url>里面。

因此,可以通过爬取这个前端元素来获取我们所需要的新闻的 url。

```
K [0
               Elements
                                                        Performance >>>
                                                                           Console
                                    Sources
                                              Network
     <!DOCTYPE html>
     <html>
     ▶ <head> ••• </head>
     ▼<body>
        <!--nav start-->
        <!--[4,319,151] published at 2023-07-31 16:08:42 -->
       ▶ <style type="text/css"> ··· </style>
1
       ▶ <script type="text/javascript"> ••• </script>
        <!--nav start-->
       ▶ <div id="navbar" style="height: 110px;"> ••• </div>
        <!--nav end-->
         <div class="clear"></div>
         <!--nav end-->
         <!--content e-->
        ▼<div class="w1280 mt20">
         ▼<div class="content-left">
           ▶<div class="btl">••</div>
           ▼<div class="content list"> == $0
             ▶
            ₹/div>
          </div>
         ▶ <div class="content-right"> • </div>
           <div class="clear"></div>
```

图 2-2 网页页面元素示意图

通过以上信息,我们可以使用 python 的 bs4、urllib 和 request 来爬取中国新闻网滚动新闻的 url 信息,主要操作步骤如下:

- 1. 设置好基础的 url 部分,以及年月日列表
- 2. 通过 request 里的 get 函数来到达某一日滚动新闻页面
- 3. 通过 html parser 解析 html 结构,找到 content list 和 url 列表
- 4. 将 url 爬取下来,放到列表中,写入到文件中

程序如下所示:

```
import requests
from bs4 import BeautifulSoup
          urllib.parse import urljoin
     def crawl_news_urls(base_url, year, month, day):
    url = f"{base_url}/{year}/{month:02d}{day:02d}/news.shtml"
         response = requests.get(url)
         soup = BeautifulSoup(response.text, 'html.parser')
         news urls = []
         for li in soup.select('.content_list ul li'):
              if div bt:
                 link = div_bt.a
                     news_url = urljoin(url, link['href'])
                      news_urls.append(news_url)
         return news_urls
     years = [2011,2012, 2013, 2014, 2015, 2016, 2017,2018,2019,2020,2021,2022, 2023]
      nonths = [1, 2, 3,4,5,6,7,8,9,10,11,12]
     with open('./urls.txt', 'a', encoding='utf8') as file:
              for month in months:
                  for day in days:
                      news_urls = crawl_news_urls(base_url, year, month, day)
                      print(news_urls)
37
38
                       for news_url in news_urls:
                         file.write(news_url + '\n')
```

图 2-3 收集 url 程序示意图

```
הביל כב ל כם ביל בשב ל כם ביל מווא מוויץ ביל מוויא . ביל מוויא וויא המוויא ביל ביל ביל ביל ביל ביל ביל ביל היל
        http://www.chinanews.com/cj/2021/01-05/9379298.shtml
2209
2210
        http://www.chinanews.com/gj/2021/01-05/9379267.shtml
2211
        http://www.chinanews.com/sh/2021/01-05/9379282.shtml
        http://www.chinanews.com/tp/2021/01-05/9379294.shtml
2212
2213
       http://www.chinanews.com/gn/2021/01-05/9379296.shtml
2214
        http://www.chinanews.com/tp/2021/01-05/9379293.shtml
2215
        http://www.chinanews.com/tp/hd2011/2021/01-05/966513.shtml
2216
        http://www.chinanews.com/gj/2021/01-05/9379255.shtml
        http://www.chinanews.com/tp/hd2011/2021/01-05/966508.shtml
2217
2218
        http://www.chinanews.com/cj/2021/01-05/9379281.shtml
```

图 2-4 url 收集结果图

如上图所示, 我从 2011 到 2023 年收集到了共 2000 多条数据。

代码如下:

```
    # 这个程序是用来爬取 类似 https://www.chinanews.com/scroll-news/2019/1211/news.shtml 这个网站里的 url 的
    import requests
    from bs4 import BeautifulSoup
    from urllib.parse import urljoin
```

```
6.
   def crawl_news_urls(base_url, year, month, day):
7.
       url = f"{base_url}/{year}/{month:02d}{day:02d}/news.shtml"
8.
9.
        response = requests.get(url)
10.
       soup = BeautifulSoup(response.text, 'html.parser')
11.
12.
       news_urls = []
        for li in soup.select('.content_list ul li'):
13.
            div_bt = li.find('div', class_='dd_bt')
14.
15.
            if div bt:
                link = div_bt.a
16.
                if link:
17.
18.
                    news_url = urljoin(url, link['href'])
19.
                    news urls.append(news url)
20.
21.
        return news_urls
22.
23. # 主程序
24. base_url = "http://www.chinanews.com/scroll-news"
25.
26. \text{ years} = [2021, 2022, 2023]
27. months = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
28. days = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
   , 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]
29.
30. # 爬取新闻页面的 URL
31. with open('./data/urls.txt', 'a', encoding='utf8') as file:
32.
       for year in years:
            for month in months:
33.
34.
                for day in days:
35.
                    news_urls = crawl_news_urls(base_url, year, month, da
   y)
36.
                    print(news_urls)
37.
                    for news_url in news_urls:
38.
                        file.write(news_url + '\n')
```

2.2 爬取新闻数据

打开收集到的某个 url, 里面的网页结构如图所示:



图 2-5 新闻结构示意图

可以看到,新闻的主体内容是在 <body>/<main_content w1280>/<con_left>/<content_maincontent_content>下的<left_zw> 因此我们可以通过爬取 left_zw 里中的文字信息来获取新闻内容。 具体步骤如下:

- 1. 打开 urls 文件, 获取 urls 列表
- 2. 通过 request.get 获取网页结构,并通过 html parser 解析
- 3. 查找 left_zw 类,将中的文本扒下来
- 4. 保存到文件中

代码如下:

```
    from urllib.parse import urlsplit
    import requests
    from bs4 import BeautifulSoup
    import os
    import threading
    def fetch_and_save(urls, count):
    try:
```

```
10.
           for url in urls:
11.
               if not url.startswith("http://www.chinanews.com/"):
12.
                   continue
13.
               parsed_url = urlsplit(url)
14.
               # base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"
               path_parts = parsed_url.path.split('/')
15.
16.
               year = path_parts[2]
17.
               if not year.isdigit() :
                   continue
18.
19.
               type = path_parts[1]
               count += 1
20.
21.
               filepath = f'./data/news/'
22.
               if not os.path.exists(filepath):
23.
                   os.makedirs(filepath)
24.
               filename = f'{filepath}{str(count)}.txt'
25.
               if os.path.exists(filename):
26.
                   continue
27.
28.
                # 开始获取文本信息
29.
               try:
30.
                   response = requests.get(url, timeout=5)
               except requests.Timeout:
31.
32.
                   print("{} timeout".format(count))
33.
                   continue
34.
               except requests.exceptions.RequestException as e:
                   print(f"请求发生错误: {e}")
35.
36.
               response.encoding = 'utf-8'
37.
38.
               soup = BeautifulSoup(response.text, 'html.parser')
39.
               # 查找 left zw 类 (用来保存新闻内容的)
40.
               left_zw = soup.find('div', class_='left_zw')
41.
42.
               if left_zw:
43.
                   # 提取文本内容
                   news_contents = url+'\n'
44.
45.
                   paragraphs = left_zw.find_all('p')
46.
                   for paragraph in paragraphs:
47.
                       news_contents += paragraph.text.strip()
48.
49.
                   # 保存到文件
```

```
50.
                   with open(filename, 'w', encoding='utf-
   8', errors='ignore') as f:
51.
                       f.write(news_contents)
52.
                   print("完成: {} 的读取".format(count))
53.
       except Exception as e:
54.
           print(f"{count} 发生错误: {e}")
55.
56.
57.
58. if name == " main ":
59.
       urls = open('./data/urls.txt').read().split('\n')
       base_count=0
60.
61.
       chunk\_size = 100
62.
       finished num = 0
63.
       threads = []
       try:
64.
           for i in range(base_count, len(urls), chunk_size):
65.
66.
               base = i + finished_num
               thread = threading.Thread(target=fetch_and_save, args=(ur
67.
   ls[base:i+chunk_size], base))
               threads.append(thread)
68.
69.
               thread.start()
               print("开启线程: {}".format(i))
70.
71.
72.
           # 等待所有线程完成
73.
           for thread in threads:
74.
               thread.join()
75.
       except Exception as e:
76.
           print("出现问题:{}".format(e))
```

2019 年之后的文本数据是 UTF-8 编码的,在爬取中得注意鉴别。

在代码里, 我通过多线程爬取, 开了 20 个线程同时爬取数据。

2.3 分词

分词的步骤如下:

- 1. 读取文件到列表中
- 2. 对于每一个文件, 使用 cut 函数进行分词

- 3. 给分词后的每个词后加空格
- 4. 判断是否是名词
- 5.组合成一个字符串,写入到新的文件中

代码如下:

```
    import os

2. import jieba.posseg as pseg
3. import threading
4.
5. def is all chinese(strs):
6.
       for _char in strs:
7.
           if not ('\u4e00' <= _char <= '\u9fa5'):</pre>
8.
               return False
9.
       return True
10.
11. def process_text(doc):
12.
       words = pseg.cut(doc)
       word str = ''
13.
14.
       first = True
15.
       for word in words:
           now_word = str(word.word)
16.
17.
           # 判断词是否是名词,且不是人名,同时不在停用词表中
           if 'n' in word.flag :
18.
               if first:
19.
20.
                   first = False
21.
22.
                   word str += ' '
               word_str += word.word
23.
24.
       return word_str
25.
26. def process_category(base_dir):
27.
       dir_path = os.path.join(base_dir)
28.
       count = 0
       out_dir = "./data/splice/"
29.
       os.makedirs(out_dir, exist_ok=True) # 确保文件夹存在,如果不存在则
30.
   创建
31.
        for _, _, files in os.walk(dir_path):
           for f in files:
32.
```

```
33.
               out_path = os.path.join('./data/splice/', f)
34.
                if os.path.exists(out_path):
                    print(f"文件 {out_path} 已存在,跳过处理。")
35.
36.
                    continue
37.
38.
                file_path = os.path.join(dir_path, f)
39.
               with open(file_path, 'r', encoding='utf-
   8') as file_obj:
40.
                    lines = file_obj.readlines()
                    if len(lines) < 2:</pre>
41.
42.
                        continue
                    doc = ''.join(lines[1:]) # 跳过第一行 URL
43.
44.
45.
               word_str = process_text(doc)
46.
                count += 1
47.
48.
               with open(out_path, 'w', encoding='utf-8') as out:
49.
                    out.write(word_str)
50.
                if(count % 100 == 0):
51.
                    print("{}".format(count))
       print(count)
52.
53.
54. if __name__ == "__main__":
55.
56.
       base_dir = './data/news/'
57.
       threads = []
58.
       try:
59.
           thread = threading.Thread(target=process_category, args=[base
   _dir])
60.
           threads.append(thread)
61.
           thread.start()
62.
63.
           for thread in threads:
64.
               thread.join()
65.
       except Exception as e:
66.
           print("出现问题:{}".format(e))
```

2.4 构建倒排索引表

2.4.1 TF-IDF

TF-IDF 有两层意思: 一层是词频 "Term Frequency" 缩写为 TF, 另一层是逆文档频率 "Inverse Document Frequency" 缩写为 IDF。

词频指的是某一个给定的词语在该文档中出现的频率。这个数字是对词数的标准化,以防止它偏向长的文档。

逆向档案频率是一个词语普遍重要性的度量。某一特定词语的 idf, 可以由总文档数目除以包含该词语之档案的数目, 再将得到的商取以 2 为底的对数得到

$$\mathrm{idf_i} = \lg rac{|D|}{|\{j: t_i \in d_j\}|}$$

|D|为语料库中的文档总数; $|\{j: t_i \in d_j\}|$ 是包含词语 t_i 的文档总数目, 如果词语不在语料库中, 就会导致分母为零, 因此一般情况下会在分母 + 1。

字词的重要性随着它在档案中出现的次数成正比增加,但同时会随着它在语料库中出现的频率成反比下降

当有 TF 和 IDF 后,将这两个词相乘,就能得到一个词的 TF-IDF 的值。某个词在文章中的 TF-IDF 越大,那么一般而言这个词在这篇文章的重要性会越高,所以通过计算文章中各个词的 TF-IDF,由大到小排序,排在最前面的几个词,就是该文章的关键词。

可以看到, TF-IDF 与一个词在文档中的出现次数成正比, 与该词在整个语

言中的出现次数成反比。所以,自动提取关键词的算法就很清楚了,就是计算出 文档的每个词的 TF-IDF 值,然后按降序排列,取排在最前面的几个词。

2.4.2 TF-IDF 算法步骤

步骤 1. 计算词频:

TF = 某个词在文章中的出现次数

考虑到文字有长短之分,为了便于不同文章比较,进行"词频"标准化

TF = 某个词在文章中的出现次数 ÷ 文章总次数

步骤 2. 计算逆文档频率

需要一个语料库,来模拟语言的使用环境

$$\mathrm{idf_i} = \lg rac{|D|}{|\{j: t_i \in d_j\}|}$$

步骤 3. 计算 TF-IDF

 $TF-IDF = TF \times IDF$

2.4.3 倒排索引表

倒排索引表是一种高效的数据结构,用于快速查找包含某些关键词的文档。 倒排索引表的核心思想是将文档内容进行词项分解,然后记录每个词项在哪些文档中出现,从而实现快速查找。

对于每个词项,记录包含该词项的所有文档 ID 及其对应的 TF-IDF 值。倒排列表通常用一个字典来表示,字典的键是词项,值是包含该词项的文档 ID 及

权重的列表。

比如如下表:

今天	1.txt、2.txt、3.txt
新华社	888.txt、896.txt
非洲	946.txt、985.txt
中国	966.txt、955.txt

倒排索引表通过查找词,来找到包含该词的文档。比如说上表中,我查找"新华社"在哪些文档中出现,那么就能找到有888.txt、896.txt 这两个文档。

在倒排索引表中,我们可以为表项中的那些匹配上的文档加上和该词项的相关度,相关度即为 tf-idf 的值。这样可以避免像"的"、"了"这些词频很高的词相关度被计算的很高。

在我们的代码实现中,我们先进行了词频计算,统计每个词项在每个文档中出现的次数,并进行标准化处理以防止文档长度对结果的影响。以及逆文档频率计算,统计每个词项在多少文档中出现,计算公式为 IDF = log(文档总数 / (包含该词项的文档数 + 1))。IDF 反映了词项在所有文档中的稀有程度。

并通过 TF 和 IDF 的值计算出了 TF-IDF 的值,对于每个词项和文档,计算该词项在文档中的 TF-IDF 值。TF-IDF 值用于衡量词项对文档的相对重要性,计算公式为 TF-IDF = TF * IDF。

最后通过计算好的 TF-IDF 值建立出了合理的倒排索引表,其中表项中还包含了每个文档和该词的相关度。

- 1. def load_documents(directory):
- 2. # 从指定目录加载文档

```
3.
       documents = {}
4.
       for filename in os.listdir(directory):
5.
           if filename.endswith('.txt'):
6.
                filepath = os.path.join(directory, filename)
7.
                with open(filepath, 'r', encoding='utf-8') as file:
8.
                    content = file.read()
9.
                    documents[filename] = content
10.
       return documents
11.
12. def tokenize(text):
       # 分词并返回名词列表
13.
       words = pseg.cut(text)
14.
15.
       return [word for word, flag in words if flag.startswith('n')]
16.
17. def build_vocabulary(documents):
       # 构建词汇表
18.
       vocabulary = set()
19.
20.
       for content in documents.values():
           tokens = tokenize(content)
21.
22.
           vocabulary.update(tokens)
       return list(vocabulary)
23.
24.
25. def compute_tf(documents):
26.
       # 计算词频
       tf = {}
27.
28.
       for doc id, content in documents.items():
29.
           tokens = tokenize(content)
30.
           tf[doc_id] = Counter(tokens)
       return tf
31.
32.
33. def compute_idf(documents, vocabulary):
       # 计算逆文档频率
34.
35.
       idf = defaultdict(int)
36.
       num_documents = len(documents)
37.
       for content in documents.values():
38.
           tokens = set(tokenize(content))
           for token in tokens:
39.
                if token in vocabulary:
40.
                    idf[token] += 1
41.
42.
       for token in idf:
43.
           idf[token] = math.log(num_documents / (idf[token] + 1))
       return idf
44.
```

```
45.
46. # 计算 tf-idf
47. def compute_tfidf(tf, idf, vocabulary):
       # 计算 TF-IDF 向量
48.
       tfidf = defaultdict(lambda: [0.0] * len(vocabulary))
49.
50.
       token_index = {token: idx for idx, token in enumerate(vocabulary)
   }
51.
       for doc_id, freqs in tf.items():
52.
           for token, count in freqs.items():
               if token in token_index:
53.
                   tfidf[doc_id][token_index[token]] = count * idf[token
54.
   ]
55.
       return tfidf
56.
57. # 构建倒排索引表
58. def build_inverted_index(tfidf):
59.
       # 构建倒排索引表
       inverted_index = defaultdict(list)
60.
       for doc_id, freqs in tfidf.items():
61.
62.
           for token, weight in enumerate(freqs):
               if weight > 0:
63.
                   inverted_index[token].append((doc_id, weight))
64.
       return inverted_index
65.
```

得到的结果如下所示:

```
get_index_table.py 1, M
                     data > {} inverted_index_table.json > [ ] 223 > [ ] 2
           "223": [
                  "888.txt",
                  1.8207470061013074
                  "901.txt",
                  1.8207470061013074
                  "906.txt",
                  1.8207470061013074
              ],
                  1.8207470061013074
                  "908.txt",
                  10.924482036607845
                  "909.txt",
                  7.282988024405229
                  "920.txt",
                  1.8207470061013074
```

图 2-6 词汇倒排索引表示意图

其中 "223" 表示的词在词汇表中的索引,即表示词汇空间向量的第几维。下面的 888.txt 表示的是文档名, 1.82...这串数字表示的是该词和该文档的 TF-IDF 值。

2.5 计算向量空间

2.5.1 构建词汇表

从文档集合中提取所有的词项,构成一个词汇表,如下所示:

```
    def build_vocabulary(documents):
    # 构建词汇表
```

```
3.  vocabulary = set()
4.  for content in documents.values():
5.   tokens = tokenize(content)
6.   vocabulary.update(tokens)
7.  return list(vocabulary)
```

图 2-7 词汇表本地存储示意图

2.5.2 构建向量

在信息检索和文本处理领域,空间向量是一种将文本表示为多维空间中的点的方法。每个维度通常对应于一个特定的特征或属性。例如,词袋模型(Bag of Words)和 TF-IDF模型都可以将文档表示为向量,其中每个维度对应一个词汇表中的词项。

将每个文档表示为一个向量,向量的每个维度对应词汇表中的一个词项,向 量的值对应该词项在文档中的频率。

假设我们有两个文档: 文档 1: 今天天气不错; 文档 2: 明天会下雨

构建词汇表:词汇表为:["今天","天气","不错","明天","会","下

雨"]

计算词项的频率: 文档 1 的向量为: 2,1,2,0,0,0; 文档 2 的向量为: 0,0,0,1,1,1

构建向量: 文档 1 的空间向量为: 2,1,2,0,0,0; 文档 2 的空间向量为: 0,0,0,1,1,1

为了构造出每个文档的向量空间,我们可以使用 defaultdict 来初始化一个字典 tfidf, 其中每个文档的 ID 作为键, 对应的值是一个长度为词汇表大小的 向量 (初始值为全零向量)。 defaultdict(lambda: [0.0] *len(vocabulary))确保了每个文档的向量都是长度与词汇表大小一致的全零向量。

之后构造出词汇表的索引。token_index 是一个字典,键是词汇表中的词,值是词汇表中该词的位置(即索引)。{token: idx for idx, token in enumerate(vocabulary)}通过枚举词汇表,为每个词项分配一个唯一的索引值。

然后计算出TF-IDF的值。遍历每个文档的词频字典tf,tf的结构是{doc_id: {token: count}}, 其中 doc_id 是文档 ID, token 是词项, count 是该词项在文档中的出现次数。对于每个文档 doc_id 和它的词频字典 freqs:

遍历词频字典中的每个 token 及其对应的出现次数 count。如果该词项 token 在词汇表中(即 token in token_index),则:通过 token_index[token]获取该词项在词汇表中的索引位置。计算该词项的 TF-IDF 值:count * idf[token],其中 count 是词项在文档中的出现次数,idf[token]是该词项的逆文档频率。将计算得到的 TF-IDF 值存储在 tfidf[doc_id]向量的对应位置上(通过索引位置 token index[token])。

最终,tfidf字典的结构为{doc_id: [tf-idf values]},其中每个文档

ID 对应一个 TF-IDF 向量,该向量的每个维度表示词汇表中对应词项的 TF-IDF 值。

```
1.
      def compute_tf(documents):
2.
       # 计算词频
3.
       tf = \{\}
       for doc_id, content in documents.items():
4.
5.
           tokens = tokenize(content)
           tf[doc_id] = Counter(tokens)
6.
7.
       return tf
8.
9.
   def compute_idf(documents, vocabulary):
10.
       # 计算逆文档频率
11.
       idf = defaultdict(int)
       num_documents = len(documents)
12.
       for content in documents.values():
13.
14.
           tokens = set(tokenize(content))
           for token in tokens:
15.
16.
                if token in vocabulary:
                    idf[token] += 1
17.
       for token in idf:
18.
19.
           idf[token] = math.log(num_documents / (idf[token] + 1))
20.
       return idf
21.
22. # 计算 tf-idf
23. def compute_tfidf(tf, idf, vocabulary):
       # 计算 TF-IDF 向量
24.
       tfidf = defaultdict(lambda: [0.0] * len(vocabulary))
25.
26.
       token index = {token: idx for idx, token in enumerate(vocabulary)
   }
       for doc_id, freqs in tf.items():
27.
28.
           for token, count in freqs.items():
29.
                if token in token_index:
30.
                    tfidf[doc_id][token_index[token]] = count * idf[token
   ]
31.
       return tfidf
```

3 系统设计及实现

3.1 系统设计

3.1.1 概述

该系统主要实现了一个基于 TF-IDF 空间向量的文档检索功能。用户可以输入查询内容,系统会根据输入查询进行相关文档的检索,并返回按相关度排序的文档列表。用户还可以对文档进行评分和查看全文。

3.1.2 模块划分

系统主要分为以下几个模块:

- 1. 数据加载模块
- 2. 文本处理模块
- 3. 向量化模块
- 4. 检索模块
- 5. 结果显示模块

3.1.3 详细设计

(1) 数据加载

功能:从文件系统中加载所需的数据,包括文档、TF-IDF向量、IDF值、词汇表和倒排索引表。

设计:

load json(filepath): 从指定路径加载 JSON 文件, 返回数据。

load_documents(directory): 从指定目录加载所有文本文档,返回一个字典,其中键为文档名,值为文档内容。

(2) 文本处理

功能:对文本进行分词,并提取出名词。

设计:

extract_nouns(text): 使用 jieba 库对文本进行分词,提取并返回名词列表。

(3) 向量化模块

功能:将用户查询和文档转换为向量表示,用于后续的相似度计算。

设计:

query_to_vector(query, idf, vocabulary):将用户的查询转换为向量表示。首先对查询进行分词,然后计算每个词的 TF-IDF 值,构建查询向量。compute_cosine_similarity(vec1, vec2):计算两个向量的余弦相似度,用于评估文档和查询之间的相似度。

(4) 检索模块

功能:根据用户查询检索相关文档,并按相似度排序。

设计:

search(query, tfidf, idf, vocabulary): 执行检索操作。将查询转换为向量后,与每个文档的 TF-IDF 向量计算相似度,返回按相似度排序的文档列表。

(5) 结果显示

功能:显示检索结果,并允许用户查看文档全文和进行评分。

设计:

display_results(results, documents, query, top_n=5):显示检索结果。包括文档的相关度、日期、URL 和匹配的句子。

main(): 主程序循环,负责处理用户输入,调用检索和显示模块,并处理用户对 文档的评分和查看全文操作。

3.1.4 工作流程

1. 初始化

系统启动时,加载所有必要的数据文件:文档、TF-IDF向量、IDF值、词汇表和倒排索引表。

2. 查询处理:

用户输入查询内容, 系统调用方法将查询转换为向量。

3. 检索

系统调用 search 方法,根据查询向量计算与每个文档向量的相似度,得到相关文档的列表并按相似度排序。

4. 结果显示

系统调用 display_results 方法,显示检索到的文档列表,包括相关度、日期、URL 和匹配的句子。

5. 用户交互

用户可以选择查看文档全文,或对文档进行评分。 系统更新评分数据并保存。

用户在终端输入查询内容。这一阶段的交互流程如下:

1. 提示用户输入查询:

系统在终端中显示提示: "请输入搜索内容(输入 'exit' 退出):"。

2. 用户输入查询:

用户输入一个查询字符串(可以是一个或多个词语)。如果用户输入 "exit",系统会退出循环,结束程序。

系统将检索到的文档结果显示给用户。

对于每个文档,显示以下信息:

1. 文档编号: 用于用户选择查看或评分

2. 相关度: 文档与查询的相似度

3. 日期: 从文档 URL 中提取的日期

4. URL: 文档的 URL 地址

5. 匹配句子: 在文档中包含查询词语的句子, 并将匹配的词语标红

用户可以选择查看文档全文或对文档进行评分。具体步骤如下:

1. 提示用户选择操作

系统在终端显示提示: "请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询:"。

2. 用户输入选择

用户输入一个文档编号来查看全文,或者输入"编号评分"格式对文档进行评分。如果用户输入"exit",系统会返回查询输入提示,允许用户输入新的查询。

3. 查看全文

如果用户输入文档编号,系统会打开并显示文档的全部内容。

4. 评分

如果用户输入"编号评分",系统会更新评分数据并保存:

5. 提取文档编号和评分

更新 ratings 字典,将评分保存到对应的文档

调用 save json 方法,将更新后的评分数据保存到文件

3.2 系统实现

3.2.1 数据加载模块

系统启动时,从磁盘加载所有必要的数据文件,包括文档内容、TF-IDF 向量、IDF 值、词汇表和倒排索引表。这些数据的加载确保了系统能够快速响应用户的查询。

数据加载函数

```
    def load_json(filepath):
    with open(filepath, 'r', encoding='utf-8') as file:
    data = json.load(file)
    print("load_inverted_index ok")
    return data
```

加载文档

```
    def load_documents(directory):

2.
        documents = {}
        for filename in os.listdir(directory):
3.
            if filename.endswith('.txt'):
4.
                filepath = os.path.join(directory, filename)
5.
6.
                with open(filepath, 'r', encoding='utf-8') as file:
7.
                    content = file.read()
8.
                    documents[filename] = content
9.
        print("load document ok")
        return documents
10.
```

3.2.2 查询处理模块

系统接收用户输入的查询,先进行分词,提取名词,并将其转换为向量表示。

提取名词函数

```
    def extract_nouns(text):
    words = pseg.cut(text)
    return [word for word, flag in words if flag.startswith('n')]
```

将查询转换为向量

```
def query_to_vector(query, idf, vocabulary):
1.
2.
       tokens = extract_nouns(query)
3.
       tf = Counter(tokens)
4.
       vector = np.zeros(len(vocabulary))
5.
       token_index = {token: idx for idx, token in enumerate(vocabulary)
       for token, count in tf.items():
6.
7.
            if token in token_index:
8.
                vector[token_index[token]] = count * idf.get(token, 0.0)
9.
        return vector
```

3.2.3 检索模块

系统使用计算好的查询向量在文档库中检索相关文档, 计算文档与查询的余 弦相似度并排序。

计算余弦相似度

```
    def compute_cosine_similarity(vec1, vec2):
    dot_product = np.dot(vec1, vec2)
    norm1 = np.linalg.norm(vec1)
    norm2 = np.linalg.norm(vec2)
    if norm1 == 0 or norm2 == 0:
    return 0.0
    return dot_product / (norm1 * norm2)
```

检索文档

```
1. def search(query, tfidf, idf, vocabulary):
2.
        query_vector = query_to_vector(query, idf, vocabulary)
3.
        scores = []
4.
        for doc_id, doc_vector in tfidf.items():
5.
            score = compute_cosine_similarity(query_vector, np.array(doc_
    vector))
6.
            scores.append((doc_id, score))
7.
        ranked_docs = sorted(scores, key=lambda item: item[1], reverse=Tr
    ue)
8.
        return ranked_docs
```

3.2.4 显示结果

系统将检索到的文档结果显示给用户,包括文档的相关度、日期、URL 和匹配的句子。

提取日期

```
    def extract_date_from_url(url):
    match = re.search(r'/(\d{4}/\d{2}-\d{2})/', url)
    if match:
    return match.group(1).replace('/', '-')
    return "日期未知"
```

显示检索结果

```
    def display_results(results, documents, query, top_n=5):
    doc_ids = []
    for idx, (doc_id, score) in enumerate(results[:top_n], 1):
    url = documents[doc_id].split('\n')[0]
    date = extract_date_from_url(url)
```

```
6.
            print(f"{idx}. 文档: {doc_id} 相关度: {score:.4f} 日期:
   {date}")
            print(f"URL: {url}")
7.
8.
9.
            content = documents[doc_id]
            sentences = content.split('\n')
10.
            split_sentences = []
11.
12.
            for sentence in sentences:
13.
                split_sentences.extend(sentence.split('...'))
14.
15.
            query_terms = set(extract_nouns(query))
16.
            matching_sentences = [sentence for sentence in split_sentence
   s if any(term in sentence for term in query_terms)]
17.
18.
            for sentence in matching_sentences[:2]:
19.
                for term in query_terms:
20.
                    if term in sentence:
21.
                        sentence = sentence.replace(term, f"\033[91m{term
   }\033[0m")
22.
                print(f"- {sentence.strip()}. ")
23.
            print("-" * 80)
24.
25.
            doc_ids.append(doc_id)
26.
27.
       return doc_ids
```

3.2.5 用户交互模块

用户可以选择查看文档全文或对文档进行评分。

```
1. def main():
2.
       tfidf = load_json('./data/tfidf.json')
       idf = load_json("./data/idf.json")
3.
       vocabulary = load_json("./data/vocabulary.json")
4.
5.
       documents = load_documents('./data/news')
6.
       inverted_index = load_json('./data/inverted_index_table.json')
7.
       ratings_filepath = './data/ratings.json'
8.
       ratings = load_json(ratings_filepath)
9.
10.
       while True:
11.
           query = input("请输入搜索内容(输入 'exit' 退出): ")
12.
           if query.lower() == 'exit':
```

```
13.
               break
14.
           results = search(query, tfidf, idf, vocabulary)
           if results:
15.
               doc_ids = display_results(results, documents, query)
16.
17.
               while True:
                   selection = input("请输入要查看全文的文档编号或输入评分
18.
    (格式: 编号 评分),输入 'exit' 返回查询: ")
19.
                   if selection.lower() == 'exit':
20.
                      break
21.
                   try:
                       if ' ' in selection:
22.
                          doc_num, rating = selection.split()
23.
24.
                          doc_id = doc_ids[int(doc_num) - 1]
25.
                          ratings[doc_id] = int(rating)
                          utils.save_json(ratings, ratings_filepath)
26.
                          print(f"文档 {doc id} 评分已保存: {rating}")
27.
28.
                       else:
                          doc_id = doc_ids[int(selection) - 1]
29.
30.
                          with open(f'./data/news/{doc_id}', 'r', encod
   ing='utf-8') as file:
                              print(file.read())
31.
32.
                   except (IndexError, ValueError):
                       print("无效输入,请输入有效的文档编号或评分。")
33.
34.
           else:
35.
               print("未找到相关文档。")
```

4 系统运行效果展示

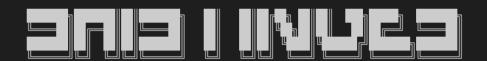


图 4-1 初始化成功界面示意图

系统初始化成功!准备开始您的任务吧!

请输入搜索内容(输入 'exit' 退出): 新华社在非洲做了什么?

图 4-2 用户输入搜索信息画面示意图



系统初始化成功!准备开始您的任务吧!

请输入搜索内容(输入 'exit' 退出): 新华社在非洲做了什么?

Building prefix dict from the default dictionary ...

Loading model from cache C:\Users\20531\AppData\Local\Temp\jieba.cache

Loading model cost 0.518 seconds.

Prefix dict has been built successfully.

1. 文档: 912.txt 相关度: 0.8615 日期: 2021-01-02

URL: http://www.chinanews.com/gj/2021/01-02/9376898.shtml

- 新华社内罗毕1月1日电 题:非洲自贸区正式启动 有望增强非洲发展韧性新华社记者2021年1月1日,非洲大陆自由贸易区自由贸易启动仪式在线上举行,标志着非洲自贸区正式启动。 分析人士认为,非洲自贸区建设虽面临诸多挑战,但它凸显了非洲国家推动经济转型的决心,
- 有望增强非洲发展韧性。

2. 文档: 943.txt 相关度: 0.4920 日期: 2021-01-03

URL: http://www.chinanews.com/gn/2021/01-03/9377795.shtml

- 新华社北京1月3日电 题:新闻分析:中国外长连续第31年新年首访选择非洲释放哪些信号? 近华社记者马卓言1月4日至9日,国务委员兼外交部长王毅将对尼日利亚、刚果(金)、博茨瓦纳、 坦桑尼亚、塞舌尔进行正式访问,这将是中国外长连续第31年新年首访选择
- 分析人士认为,当前新冠肺炎疫情还在全球肆虐,此访克服重重困难,体现了中方对非洲国家 的高度重视和对发展中非关系的坚定决心,将有力推动中非合作论坛北京峰会和中非团结抗疫特 别峰会成果落实,促进中非传统友谊进一步发扬光大。

3. 文档: 932.txt 相关度: 0.0348 日期: 2021-01-03

URL: http://www.chinanews.com/gj/2021/01-03/9377810.shtml

华社新德里1月3日电 据当地媒体3日报道,印度北方邦加济阿巴德地区一火葬场建筑当天发 生坍塌,目前已造成17人死亡、多人受伤。

4. 文档: 971.txt 相关度: 0.0307 日期: 2021-01-03

URL: http://www.chinanews.com/gn/2021/01-03/9377765.shtml

收官之际的2020年万马战犹酣习近平总书记始终深谋远虑,沉着应对时不我待,尽职尽责5幕" 画报",30个瞬间汇聚成一鼓作气向实现第一个百年奋斗目标冲刺的历史时刻第一幕开"顶风船"第 二幕啃"硬骨头"第三幕谋"大棋局"第四幕扩"朋友圈"第五幕筑"同心圆"策划:霍小光监制:车玉 明、张晓松摄影: 鞠鹏、谢环驰、王晔、刘卫兵、庞兴雷、燕雁、申宏、翟健岚、李响文案 | 视 觉: 黄玥、朱高祥、张惠慧、张晨光新媒体编辑: 李小虎新华社国内部出品。

5. 文档: 944.txt 相关度: 0.0269 日期: 2021-01-03

URL: http://www.chinanews.com/gj/2021/01-03/9377794.shtml

·华社阿比让1月3日电(记者郑扬子)尼亚美消息:尼日尔内政部3日发表声明说,尼西部两个 村庄2日遭武装人员袭击,至少70名平民死亡。

图 4-3 检索结果示意图

5. 文档: 944.txt 相关度: 0.0269 日期: 2021-01-03

son

URL: http://www.chinanews.com/gj/2021/01-03/9377794.shtml

- 新华社阿比让1月3日电(记者郑扬子)尼亚美消息:尼日尔内政部3日发表声明说,尼西部两个村庄2日遭武装人员袭击,至少70名平民死亡。

请输入要查看全文的文档编号或输入评分(格式: 编号 评分),输入 'exit' 返回查询: 2 http://www.chinanews.com/gn/2021/01-03/9377795.shtml

新华社北京1月3日电 题:新闻分析:中国外长连续第31年新年首访选择非洲释放哪些信号?新华 社记者马卓言1月4日至9日,国务委员兼外交部长王毅将对尼日利亚、刚果(金)、博茨瓦纳、坦桑 尼亚、塞舌尔进行正式访问,这将是中国外长连续第31年新年首访选择非洲。分析人士认为,当 前新冠肺炎疫情还在全球肆虐,此访克服重重困难,体现了中方对非洲国家的高度重视和对发展 中非关系的坚定决心,将有力推动中非合作论坛北京峰会和中非团结抗疫特别峰会成果落实,促 进中非传统友谊进一步发扬光大。中国外长新年首次出访到访非洲的传统始于1991年。中国政府 非洲事务特别代表许镜湖认为,这一优良传统彰显了中非几十年来风雨同舟、患难与共的深厚情 谊。**"**在所有大国中,只有中国的外长能够连续31年每年首次出访都选择非洲,这体现了中方对非 洲国家的一贯重视。"中国前驻南非大使刘贵今说,此访涉及的五个国家涵盖了西非、中非、南非 、东非以及印度洋岛国,中方全面发展同非洲国家关系,是推动构建中非命运共同体的深刻体现 据了解,中方此前已向非洲15国派遣抗疫医疗专家组,同非洲12国签署缓债协议,减免非洲15 国2020年底到期的无息贷款债务。分析人士认为,携手抗疫使中非友谊得到新的升华,此访也将 为提升非洲国家抗疫信心起到积极作用。许镜湖说,作为最大的发展中国家和发展中国家最集中 的大陆,中国和非洲是国际事务中天然的同盟军,中国愿意和包括广大发展中国家在内的世界各 国一道,共同应对危机,为世界和平发展做出中国贡献,此访将继续向世界传递这一重要信息。2 929年,单边主义、保护主义、强权政治成为阻挠国际合作的主要障碍。中国非洲研究院常务副院 长李新烽认为,中方将以此次访问为契机再次强调团结合作,同非洲国家一道,坚定不移推动经 济全球化、捍卫多边主义,进一步维护发展中国家的利益。针对近来出现的各类挑拨中非关系的 谣言, 李新烽表示, 此访即是对外宣示: 中非友谊牢不可破, 中非关系不针对第三方, 但任何第 三方对中非团结的破坏企图都是徒劳的。2021年是落实中非合作论坛北京峰会成果的收官之年, 新一届论坛会议将于今年在塞内加尔举办。据外交部介绍,王毅此访还将就新一届论坛会议筹备 事宜同非方交换意见。中非合作论坛自**2000**年成立以来,中非共同制订并相继实施了**"**十大合作计 划""八大行动",贸易额增长了20倍,中国对非直接投资增长了100倍。分析人士表示,在此基础 上,中非未来在自贸区、数字经济、减贫、应对气候变化等领域的合作大有可为,此访将有利于 推动后疫情时期中非关系取得新发展。许镜湖表示,中国脱贫事业取得的伟大成就给予非洲人民 极大鼓舞,未来中非双方在减贫脱贫、经济恢复、粮食安全等领域的合作力度还会继续加大。"疫 苗合作等医疗卫生领域的合作将是下一阶段中非合作的热点。非洲传统医学用于防治传染病的相 关研究也会取得新进展。**"**李新烽说。李新烽认为,接下来,中非合作还会继续向民生方面倾斜,

图 4-4 查看原文示意图

让中非关系继续走深走实,走向民间、走向基层、走向草根,进一步夯实中非关系的民意基础。 【编辑:朱延静】

请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询: 1 10 文档 912.txt 评分已保存: 10

请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询:

图 4-5 人工评价示意图

以 \mathcal{L} T 中人水地法定体定去,定因以内、定因全点、定因中枢,见一夕刀关于中人水时以高至四。

【编辑:朱延静】

请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询: 1 10 文档 912.txt 评分已保存: 10

请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询: exit 请输入搜索内容(输入 'exit' 退出): ■

图 4-6 返回查询示意图

请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询: 1 10 文档 912.txt 评分已保存: 10

请输入要查看全文的文档编号或输入评分(格式:编号 评分),输入 'exit' 返回查询: exit 请输入搜索内容(输入 'exit' 退出): exit

PS E:\bupt-homework\Information-obtain\folder-git>

图 4-7 输入 exit 退出检索系统示意图



图 4-8 系统退出结束画面示意图