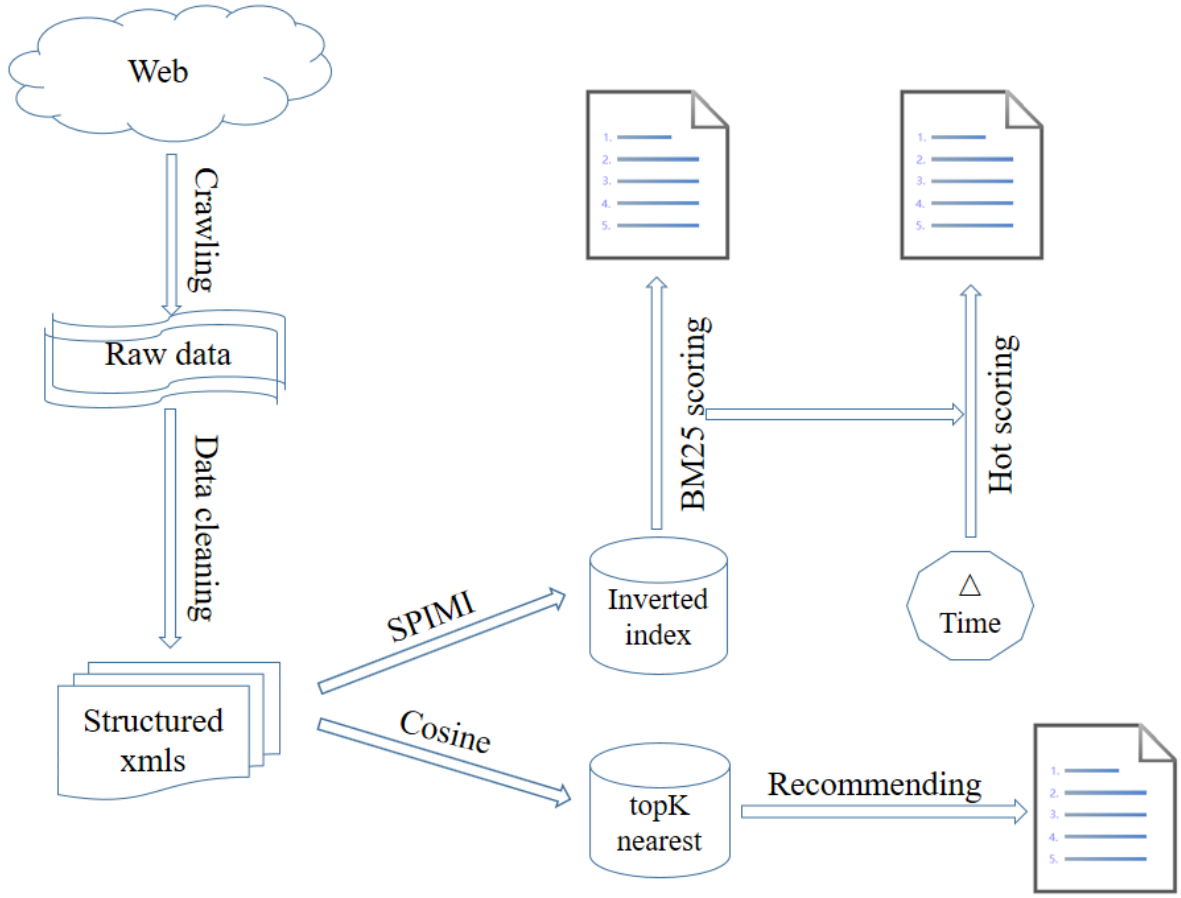
信息检索大作业

2011743 高祎珂

简介

实验要求实现一个系统的Web搜索引擎（主题不限），为用户提供查询服务和个性化推荐。下面是我构建的搜索引擎的架构图：

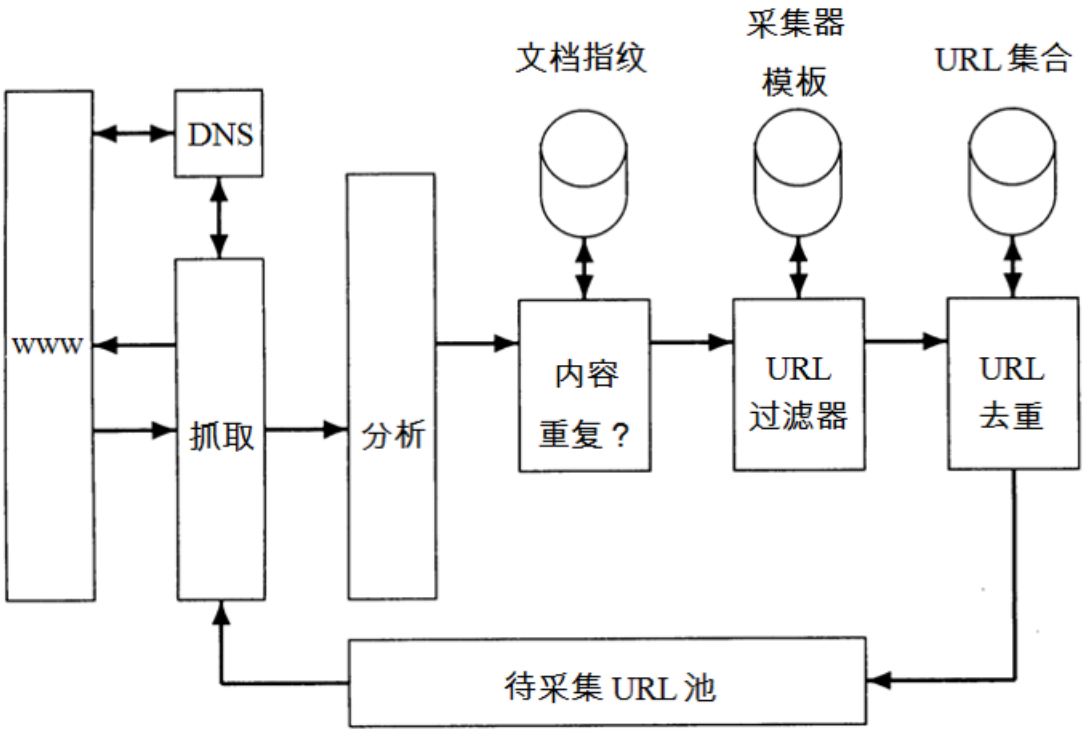


首先爬虫程序从特定的几个新闻网站抓取新闻数据，然后过滤网页中的图片、视频、广告等无关元素，抽取新闻的主体内容，得到结构化的xml数据。然后一方面使用内存式单遍扫描索引构建方法（SPIMI）构建倒排索引，供检索模型使用；另一方面根据向量空间模型计算两两新闻之间的余弦相似度，供推荐模块使用。前端写了几个html模板，与后端相连接，为用户提供服务，下面是关于此项目的具体展示。

制作流程

网页抓取

网络爬虫采集器的基本架构如下图所示



基本上是"抓取→分析→得到新的URL→再抓取→再分析"这样一个死循环过程。我们要做的是一个新闻搜索引擎，所以抓取的是新闻数据。首先找一个新闻网站作为种子网站进行抓取，为简单起见，我选择了结构清晰、html代码便于解析的搜狐新闻。

搜狐新闻国内要闻列表如下

新闻中心 > 国内新闻 > 国内要闻

·刘鹤应约与美财长通话：中方有实力捍卫国家利益 双方同意保持沟通(03/24 11:55)
·政协委员丁洁：应将乡村儿童大病医保模型纳入健康扶贫(03/03 23:52)
·小平小道：浓浓真情，改革初心(02/19 10:37)
·杨志辉："五个一百"，将正能量传播到底(02/16 10:38)
·赵克志在北京调研时强调 牢记人民公安为人民的初心使命 着力加强和改进新时代公安基层基础工作(12/26 11:15)
·贵州遵义连续4年进入中国最安全城市排行榜(12/26 00:48)
·不忘初心牢记使命 奋力走好新时代长征路(11/06 07:37)
·人民日报：我们是历史创造的一代,也是创造历史的一代(10/26 12:45)
·学习十九大精神，市委书记蔡奇划出哪些"必修课"？(10/26 11:28)
·"一带一路"连接中阿走向共同繁荣(09/07 09:20)
·雄伟！北京市政府新大楼开始揭开"面纱"(09/07 08:30)
·《巴黎协定》曾经功败垂成 关键时刻是中国力挽狂澜！(09/01 14:43)
·蔡奇：把中关村率先建成具有全球影响力的科学城(08/21 10:15)
·让"创客梦"融入"中国梦"(08/18 11:11)
·新华社评论员：坚决打赢反贫困斗争的伟大决战(08/16 09:31)
·河长效应：满眼绿色取代违建垃圾(08/10 07:57)
·"不平凡的九件大事"之全面加强党的领导(08/08 10:28)

左边是带URL的标题，右边括号里有新闻时间。如果我们要获取更多的数据条数，只要不断模拟点击下一页即可。下一页的URL也只是在首页的基础上加上_xxx.shtml，xxx就是不同的页码。

查看列表的html源码，得知列表都在类名为newsblue1的td中，所以只需要解析html源码就可以得到新闻标题、URL和时间，python解析html可以用BeautifulSoup包。查看html源码，正文位于类名为text clear的div中，据此可以很方便的提取新闻正文。得到一条新闻的所有数据之后，我们需要将之结构化成xml文件。

代码如下：

```
from bs4 import BeautifulSoup
import urllib.request
import xml.etree.ElementTree as ET
import configparser

def get_news_pool(root, start, end):
    news_pool = []
    for i in range(start,end,-1):
        page_url = ''
        if i != start:
            page_url = root +'_%d.shtml'%(i)
        else:
            page_url = root + '.shtml'
        try:
            response = urllib.request.urlopen(page_url)
        except Exception as e:
            print("-----%s: %s-----"%(type(e), page_url))
            continue
        html = response.read()
        soup = BeautifulSoup(html,"lxml") #
http://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/
        td = soup.find('td', class_ = "newsblue1")
        a = td.find_all('a')
        span = td.find_all('span')
        for i in range(len(a)):
            date_time = span[i].string
            url = a[i].get('href')
            title = a[i].string
            news_info = ['2016-'+date_time[1:3]+'-
'+date_time[4:-1]+':00',url,title]
            news_pool.append(news_info)
    return(news_pool)


def crawl_news(news_pool, min_body_len, doc_dir_path, doc_encoding):
    i = 1
    for news in news_pool:
        try:
            response = urllib.request.urlopen(news[1])
        except Exception as e:
            print("-----%s: %s-----"%(type(e), news[1]))
            continue
        html = response.read()
        soup = BeautifulSoup(html,"lxml") #
http://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/
        try:
            body = soup.find('div', class_ = "text
clear").find('div').get_text()
        except Exception as e:
            print("-----%s: %s-----"%(type(e), news[1]))
            continue
        if '//' in body:
            body = body[:body.index('//')]
```

```
        body = body.replace(" ", "")
        if len(body) <= min_body_len:
            continue
        doc = ET.Element("doc")
        ET.SubElement(doc, "id").text = "%d"%(i)
        ET.SubElement(doc, "url").text = news[1]
        ET.SubElement(doc, "title").text = news[2]
        ET.SubElement(doc, "datetime").text = news[0]
        ET.SubElement(doc, "body").text = body
        tree = ET.ElementTree(doc)
        tree.write(doc_dir_path + "%d.xml"%(i), encoding = doc_encoding,
xml_declaration = True)
        i += 1

if __name__ == '__main__':
    config = configparser.ConfigParser()
    config.read('../config.ini', 'utf-8')
    root = 'http://news.sohu.com/1/0903/61/subject212846158'
    news_pool = get_news_pool(root, 854, 849)
    crawl_news(news_pool, 140, config['DEFAULT']['doc_dir_path'],
config['DEFAULT']['doc_encoding'])
    print('done!')
```

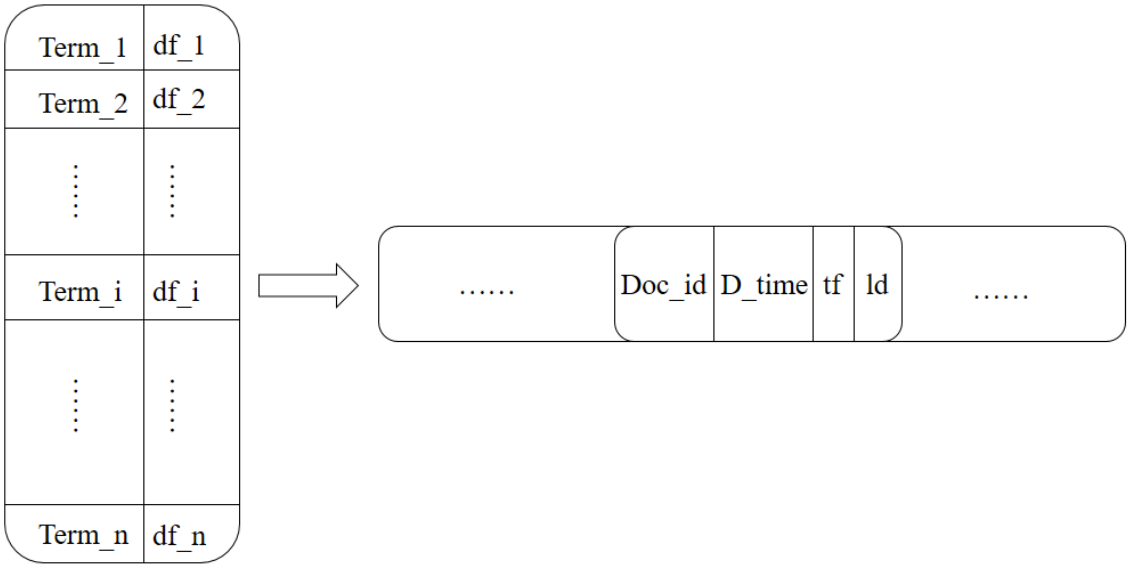
代码分为两个函数：

1. get_news_pool：从指定的网页上获取新闻池，其中新闻池是一个列表，列表中的每个元素都是一条新闻的相关信息，包括发布时间、新闻链接、标题等。
2. crawl_news：读取新闻池中的新闻，并从新闻链接获取新闻内容，最后将新闻内容存储到本地文件中。

代码使用了BeautifulSoup和urllib库来爬取数据，使用ElementTree库来存储数据。

文本索引

首先使用 jieba 库完成分词操作并统计词频，使用 ElementTree 库解析上一步爬虫得到的 XML 文档，使用 sqlite3 库将分词结果存储到数据库中。接下来使用SPIMI算法对于每个文档的词项构建倒排索引，倒排索引的结构如图：



构建倒排索引过程的代码如下：

```
    def construct_postings_lists(self):
        config = configparser.ConfigParser()
        config.read(self.config_path, self.config_encoding)
        files = listdir(config['DEFAULT']['doc_dir_path'])
        AVG_L = 0
        for i in files:
            root = ET.parse(config['DEFAULT']['doc_dir_path'] + i).getroot()
            title = root.find('title').text
            body = root.find('body').text
            docid = int(root.find('id').text)
            date_time = root.find('datetime').text
            seg_list = jieba.lcut(title + '。' + body, cut_all=False)

            ld, cleaned_dict = self.clean_list(seg_list)

            AVG_L = AVG_L + ld

            for key, value in cleaned_dict.items():
                d = Doc(docid, date_time, value, ld)
                if key in self.postings_lists:
                    self.postings_lists[key][0] = self.postings_lists[key][0] +
1 # df++
                    self.postings_lists[key][1].append(d)
                else:
                    self.postings_lists[key] = [1, [d]] # [df, [Doc]]
        AVG_L = AVG_L / len(files)
        config.set('DEFAULT', 'N', str(len(files)))
        config.set('DEFAULT', 'avg_l', str(AVG_L))
        with open(self.config_path, 'w', encoding = self.config_encoding) as
configfile:
            config.write(configfile)
        self.write_postings_to_db(config['DEFAULT']['db_path'])
```

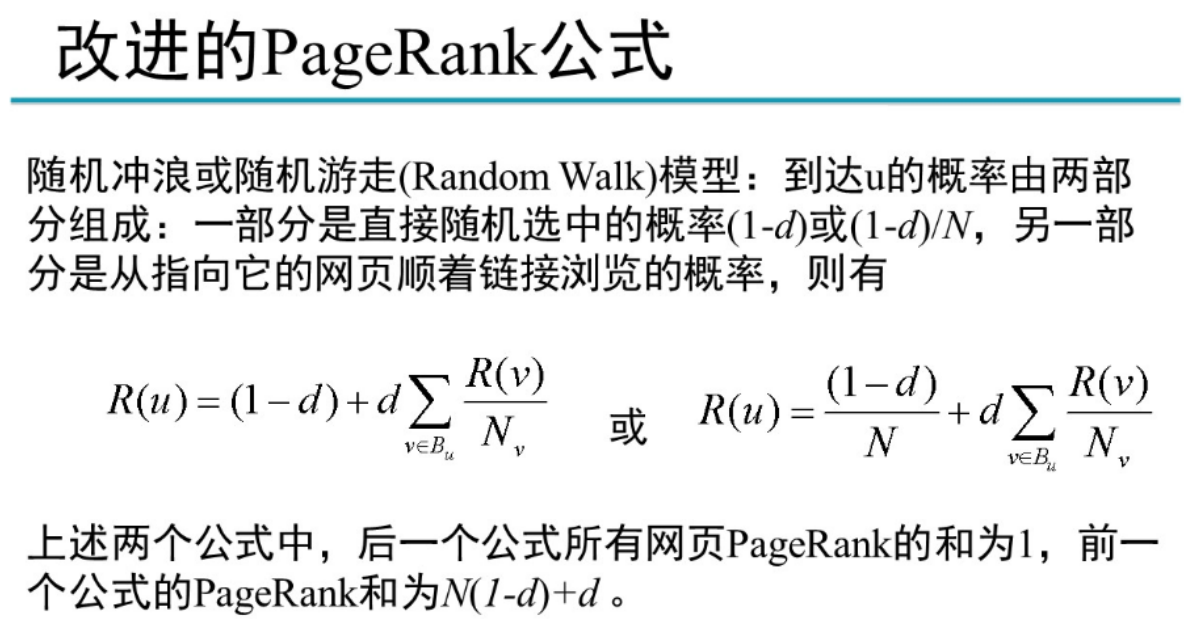
对于每个文件，代码读取了标题和正文，并通过结巴分词对其进行分词，然后通过clean_list()方法对分词后的结果进行处理。

随后，代码遍历了每个分词，创建了一个Doc对象，并将其加入倒排索引的postings_lists中。如果当前的分词已经在倒排索引中存在，则df加一，并将新的Doc对象加入到postings_lists[key][1]列表中。否则，创建一个新的键值，并将df初始化为1，文档列表初始化为包含当前Doc对象的列表。

最后，代码更新了配置文件中的N和avg_l参数，并写入了postings_lists到数据库中。

链接分析

使用PageRank进行链接分析,公式如下



计算PageRank的基本步骤如下：

1. 建立一个网页的图模型，把每一个网页看作一个节点，从一个网页指向另一个网页的链接看作一条有向边。
2. 为每个网页初始化一个PageRank值，通常设定为1。
3. 对于每一个网页，计算其从其他网页过来的贡献。这可以通过使用以下公式来实现：

   PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

   其中，PR(A)表示网页A的PageRank值，d是一个阻尼因子，通常设定为0.85，PR(Ti)表示指向网页A的网页Ti的PageRank值，C(Ti)表示网页Ti的出边数量。
4. 重复上述步骤，直到PageRank值变化不大为止。
5. 最后，将所有网页的PageRank值进行排序，从而得到每一个网页的重要性排名。

代码如下：

```
import numpy as np
import ast
import sqlite3
import configparser
from datetime import *
import xml.etree.ElementTree as ET
class PageRank():
```

```
    '''
    G: 传入图的邻接矩阵
    T: 迭代计算次数上限
    eps: 误差上限
    beta: 公式里面的beta
    return: list
    注:误差小于eps或者迭代次数大于T结束迭代计算
    '''
    def __init__(self, G, T=300, eps=1e-6, beta=0.8) -> None:
        self.G = G
        self.N = len(G)
        self.T = T
        self.eps = eps
        self.beta = beta


    def GtoM(self, G):
        '''
        创建概率转换矩阵
        '''
        M = np.zeros((self.N, self.N))
        for i in range(self.N):
            D_i = sum(G[i])
            if D_i == 0:
                continue
            for j in range(self.N):
                M[j][i] = G[i][j] / D_i  #归一化并转置
        return M

    def computePR(self, M):
        '''
        计算PR值
        '''
        R = np.ones(self.N) / self.N
        teleport = np.ones(self.N) / self.N
        for time in range(self.T):
            A = self.beta * M + (1-self.beta)*teleport
            R_new = np.dot(A, R)
            if np.linalg.norm(R_new - R) < self.eps:
                break
            R = R_new.copy()
        return np.around(R_new, 5)

    def getPR(self):
        M = self.GtoM(self.G)
        return self.computePR(M)

def urls2G():
    '''
    将数据库中urls的关系转化为图
    '''
    # 连接数据库
    config_path = ''
    config_encoding = ''
    config = configparser.ConfigParser()
```

```
    config.read(config_path, config_encoding)
    conn = sqlite3.connect(config['DEFAULT']['db_path'])
    # 创建cursor
    cursor_blogs = conn.cursor()
    cursor_list = conn.cursor()
    sql_blogs = 'SELECT page_url, urls FROM search_blogs;'
    sql_list = 'SELECT page_url, urls FROM search_blogs;'
    # 执行sql语句
    cursor_blogs.execute(sql_blogs)
    cursor_list.execute(sql_list)
    # 获取全部查询信息
    re_blogs = cursor_blogs.fetchall()
    re_list = cursor_list.fetchall()

    # 将获取的元组信息转换为图
    blogs_index = [url[0] for  url in re_blogs]
    blogs_point = [ast.literal_eval(url[1]) for  url in re_blogs]

    list_index = [url[0] for  url in re_list]
    list_point = [ast.literal_eval(url[1]) for  url in re_list]
    indexs = blogs_index + list_index
    points = blogs_point + list_point
    G = np.zeros((len(indexs), len(indexs)))
    for i, index in enumerate(indexs):
        # 依次判断包含的url是是否在爬取过的列表中，有些广告之类的链接页会包含，但没爬取
        for p_url in points[i]:
            try:
                p_index = indexs.index(p_url)
            except:
                p_index = -1
            if p_index != -1:
                G[i][p_index] = 1

    return G

if __name__ == "__main__":
    G = urls2G()
    # print(type(G))
    PR = PageRank(G)
    PR.getPR()
```

查询服务

全站查询

前面已经构建好倒排索引和pagerank值，这里的查询我们就可以利用这些值来帮助检索。检索模型使用的是基于概率的BM25模型：

$$BM25_{score}(Q,d) = \sum_{t \in Q} w(t,d)$$

$$w(t,d) = \frac{qtf}{k_3+qtf} \times \frac{k_1 \times tf}{tf+k_1(1-b+b \times l_d/avg\_l)} \times log_2 \frac{N-df+0.5}{df+0.5}$$

使用代码如下：

```
def result_by_BM25_PageRank(self, sentence):
        seg_list = jieba.lcut(sentence, cut_all=False)
        n, cleaned_dict = self.clean_list(seg_list)
        BM25_scores = {}
        for term in cleaned_dict.keys():
            r = self.fetch_from_db(term)
            if r is None:
                continue
            df = r[1]
            w = math.log2((self.N - df + 0.5) / (df + 0.5))
            docs = r[2].split('\n')
            for doc in docs:
                docid, date_time, tf, ld = doc.split('\t')
                docid = int(docid)
                tf = int(tf)
                ld = int(ld)
                s = (self.K1 * tf * w) / (tf + self.K1 * (1 - self.B + self.B *
ld / self.AVG_L))
                if docid in BM25_scores:
                    BM25_scores[docid] = BM25_scores[docid] + s
                else:
                    BM25_scores[docid] = s

        # 读取每个文档的PageRank分数
        PageRank_scores = {}
        with open("page_rank_scores.txt") as f:
            for line in f:
                docid, score = line.strip().split('\t')
                docid = int(docid)
                score = float(score)
                PageRank_scores[docid] = score

        # 将BM25和PageRank的分数相加
        combined_scores = {}
        for docid, BM25_score in BM25_scores.items():
            if docid in PageRank_scores:
                combined_scores[docid] = BM25_score + PageRank_scores[docid]

        # 对总分数进行排序
        combined_scores = sorted(combined_scores.items(), key=operator)
```

结合两个值给出结果排序。

不同特征排序

此外在查询的设计中还设置了按照热度以及时间排序，关于热度公式，比较有名的是Reddit

Given the time the entry was posted A and the time of 7:46:43 a.m. December 8, 2005 B, we have t_s as their difference in seconds

$$t_s = A - B$$

and x as the difference between the number of up votes U and the number of down votes D

$$x = U - D$$

where $y \in \{-1, 0, 1\}$

$$y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and z as the maximal value, of the absolute value of x and 1

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

we have the rating as a function $f(t_s, y, z)$

$$f(t_s, y, z) = \log_{10} z + \frac{yt_s}{45000}$$

这里我使用给的热度公式为 $hot_{score} = k_1 log(BM25_{score}) + \frac{k_2}{t_{now} - t_{news}}$

用BM25得分加上新闻时间和当前时间的差值的倒数，k1和k2是可调参数。使用的函数与上述相似，这里就不再赘述。

web页面

编写了较为简单的html，分为search，content等，其中查找htm通过一个Django模板来展示搜索结果。模板使用Jinja语法，根据后端传递的数据动态生成HTML页面。例如，搜索结果信息（标题，时间，摘要和URL）存储在一个名为"docs"的变量中，该变量是在后端查询搜索结果并将其传递到模板之前生成的。模板通过循环遍历"docs"并动态生成每个文档的HTML代码。具体代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>News Search Engine</title>
    <style type="text/css">
        div#doc {width:800px}
    .pagination-page-info {
        padding: .6em;
        padding-left: 0;
        width: 40em;
        margin: .5em;
        margin-left: 0;
        font-size: 12px;
    }
    .pagination-page-info b {
        color: black;
        background: #6aa6ed;
        padding-left: 2px;
        padding: .1em .25em;
        font-size: 150%;
    }
    </style>
</head>
<body>
    <div id="container">
        <div id="header">
```

```
            <h1>News Search Engine</h1>
            <form name="search" action="/search/" method="POST">
                <p>
                    {% if key %}
                        <input type="text" name="key_word" value="{{key}}">
                    {% else %}
                        <input type="text" name="key_word">
                    {% endif %}
                    <input type="submit" value="Search">
                </p>
            </form>
        </div>
        <hr/>
        {% block high_search%}
        {% endblock %}
        {% if error%}
            {% for doc in docs%}
            <div id="doc">
                <ul id="navigation">
                    <p><big><a href="/search/{{doc.id}}/" target="_blank">
{{doc.title}}</a></big></p>
                    <p>{{doc.time}}<br/>{{doc.snippet}}<br/>
                    <a href="{{ doc.url }}" target="_blank">{{doc.url}}</a>
                    </p>
                </ul>
            </div>
            <br/>
            {% endfor %}
        {% block next %}
        <ul>
            {% for i in page %}
                <a href="/search/page/{{i}}/">{{i}}</a> 
            {% endfor %}
        </ul>
        {% endblock %}
        {% else %}
            <p>对不起，没有您搜索的网页！</p>
        {% endif %}

        <div id="footer">
            <hr>
            &copy;

            <script type="text/javascript">
            var today = new Date();
            var year = today.getFullYear();
            document.write(year)
            </script>

        </div>

    </div>
</body>
</html>
```

个性化推荐

由于我并没有实现用户，所以这里我是在每次搜索之后做了一个推荐阅读，给出与当前搜索最相关的五篇文章，效果如下

http://www.chinanews.com/sh/2020/03-31/9142694.shtml

中新网3月31日电据江苏卫健委官方微博消息，3月30日0-24时，江苏省新增境外输入新冠肺炎确诊病例1例，为通过口岸联防联控(英国输入)。截至3月30日24时，累计报告境外输入确诊病例15例。 该病例为中国江苏籍。3月24日自英国伦敦出发，经转机后于3抵南京禄口国际机场，入关后即被转运至宿迁市隔离观察。综合流行病学史、临床症状、实验室检测结果和影像学检查结果，被诊例。 3月30日0-24时，江苏省无新增本地新冠肺炎确诊病例。截至3月30日24时，累计报告本地确诊病例631例，累计出院病例63前，追踪到密切接触者12829人，已解除医学观察12653人，尚有176人正在接受医学观察。 【编辑:叶攀】

推荐阅读

江苏新增境外输入确诊病例1例 累计境外输入14例
江苏新增1例境外输入新冠肺炎确诊病例
北京新增报告境外输入新冠肺炎病例1例
北京新增报告境外输入新冠肺炎确诊病例3例
江苏省新增新冠肺炎无症状感染者1例 为境外输入

程序思路是代码定义了一个名为"RecommendationModule"的Python类，旨在通过在文章的TF-IDF表示上使用k近邻算法来构建新闻文章的推荐系统。

该类有几个成员变量：停用词，k近邻，配置路径，配置编码，文档目录路径，文档编码，停用词路径，停用词编码，idf路径和数据库路径。

该类有几个成员函数：

- init(self, config_path, config_encoding)根据从配置文件（config.ini）中读取的值初始化成员变量。
- write_k_nearest_matrix_to_db(self)将k近邻矩阵写入SQLite数据库。
- is_number(self, s)如果输入字符串可以转换为float，则返回True，否则返回False。
- construct_dt_matrix(self, files, topK = 200)构造新闻文章的文档-术语矩阵表示。
- construct_k_nearest_matrix(self, dt_matrix, k = 5)计算每篇文章的k个最近邻，并将结果保存到k近邻成员变量中。
- **构造一个文档-词频矩阵（Document-Term Matrix）。**

```
    def construct_dt_matrix(self, files, topK = 200):
        jieba.analyse.set_stop_words(self.stop_words_path)
        jieba.analyse.set_idf_path(self.idf_path)
        M = len(files)
        N = 1
        terms = {}
        dt = []
        for i in files:
            root = ET.parse(self.doc_dir_path + i).getroot()
            title = root.find('title').text
            body = root.find('body').text
            docid = int(root.find('id').text)
            tags = jieba.analyse.extract_tags(title + '。' + body, topK=topK,
withWeight=True)
            #tags = jieba.analyse.extract_tags(title, topK=topK,
withWeight=True)
```

```
            cleaned_dict = {}
            for word, tfidf in tags:
                word = word.strip().lower()
                if word == '' or self.is_number(word):
                    continue
                cleaned_dict[word] = tfidf
                if word not in terms:
                    terms[word] = N
                    N += 1
            dt.append([docid, cleaned_dict])
        dt_matrix = [[0 for i in range(N)] for j in range(M)]
        i =0
        for docid, t_tfidf in dt:
            dt_matrix[i][0] = docid
            for term, tfidf in t_tfidf.items():
                dt_matrix[i][terms[term]] = tfidf
            i += 1

        dt_matrix = pd.DataFrame(dt_matrix)
        dt_matrix.index = dt_matrix[0]
        print('dt_matrix shape:(%d %d)'%(dt_matrix.shape))
        return dt_matrix
```

上面的代码是一个 Python 程序，它的主要作用是构造一个文档-词频矩阵（Document-Term Matrix）。

首先，程序会读入一个文件路径列表 "files" 和一个参数 "topK"。其中，"files" 列表中包含了所有需要处理的文档的路径，"topK" 参数用于指定提取关键词的数量。

接下来，程序会使用结巴分词的 TF-IDF 分析工具对每个文档的标题和正文进行关键词提取，并对每个文档的关键词进行处理，得到一个清理过的字典，其中存储了每个关键词的 TF-IDF 值。

随后，程序会构建一个二维列表 "dt_matrix"，其中的每一行表示一篇文档，每一列表示一个关键词。对于每一篇文档，程序都会将其文档 ID 和该文档所有关键词的 TF-IDF 值填入 "dt_matrix" 中。

最后，程序会将 "dt_matrix" 转换为一个 pandas DataFrame，并将其索引设置为文档 ID，最后返回 "dt_matrix"。

```
    def gen_idf_file(self):
        files = listdir(self.doc_dir_path)
        n = float(len(files))
        idf = {}
        for i in files:
            root = ET.parse(self.doc_dir_path + i).getroot()
            title = root.find('title').text
            body = root.find('body').text
            seg_list = jieba.lcut(title + '。' + body, cut_all=False)
            seg_list = set(seg_list) - self.stop_words
            for word in seg_list:
                word = word.strip().lower()
                if word == '' or self.is_number(word):
                    continue
                if word not in idf:
                    idf[word] = 1
                else:
                    idf[word] = idf[word] + 1
```

```
        idf_file = open(self.idf_path, 'w', encoding = 'utf-8')
        for word, df in idf.items():
            idf_file.write('%s %.9f\n'%(word, math.log(n / df)))
        idf_file.close()
```

这段代码的作用是生成一个IDF文件，该文件将用于计算TF-IDF。

首先，读取文档目录中的所有文件，并计算出n，表示文件的总数。

然后，对于每一个文件，它使用jieba分词对标题和正文进行分词，并剔除停用词。

接着，对于每一个词，它计算出该词在所有文件中出现的次数，并用n除以该词出现的次数，得到一个IDF值，用以计算该词的TF-IDF。

最后，它将IDF值写入文件中，并关闭文件。也即是idf.txt

结果展示

初始界面

搜索结果展示

News Search Engine

江苏 Search

相关度 时间 热度 ok

公安部交管局：清明节假期首日全国道路交通平稳有序

2020-4-4
中新社北京4月4日电(记者张子扬)记者4日从中国公安部交管局获悉，清明节假期第一天，全国高速公路、国省道干线通行正常，未发生长时间、长距离、大面积交通拥堵。从全国200个卡口监测点位监测情况看，4月3日16:00至4日16:00总流量36.....
http://www.chinanews.com/gn/2020/04-04/9147924.shtml

血写忠诚的“江海神探”顾瑛

2020-4-4
(为了民族复兴英雄烈士谱)血写忠诚的“江海神探”顾瑛 新华社南京4月3日电(记者邱冰清)“一个人虽然不能决定生命的长度，但可以拓展人生的宽度。”江苏省南通市公安局刑警支队原支队长顾瑛的案情记录本中，写着这样一句话。 为了侦破退休前最后.....
http://www.chinanews.com/gn/2020/04-04/9147790.shtml

东湖评论：下个人间四月天，再邀英雄沐楚风

2020-4-4
近日援鄂医疗队胜利完成救护任务陆续踏上返程旅途。商场幕墙、机场高速、安检口、登机口.....无数电子屏上，写满了湖北人民对他们的敬意和祝福，“谢谢你们，为我们拼过命！”湖北人的朋友圈刷屏的话语，虽简短，却足已表露心声。白衣执甲，逆行出征，他们是.....

News Search Engine

江苏 Search

相关度 时间 热度 ok

长三角一体化江苏“任务书”和“线路图”公布

2020-4-1
中新社南京4月1日电(记者朱晓颖)江苏省政府4月1日在南京召开《<长江三角洲区域一体化发展规划纲要>江苏实施方案》(以下简称“《江苏实施方案》”)新闻发布会，长三角一体化江苏“任务书”和“线路图”正式公布。记者梳理后发现，江苏要做的事中，.....
http://www.chinanews.com/cj/2020/04-01/9144092.shtml

828名江苏援湖北医疗队员平安返回

2020-3-31
中新网南京3月31日电(记者朱晓颖通讯员苏卫萱)31日，又一批江苏援湖北医疗队员平安返回江苏。 此次返回队员共828人，分别为第五批江苏援湖北医疗队南京市队261人、无锡市队131人、苏州市队260人，第七批江苏援湖北医疗队南京鼓楼医院.....
http://www.chinanews.com/sh/2020/03-31/9143363.shtml

江苏外向型经济受“疫”冲击 多措并举稳外贸稳外资

2020-4-2
中新社南京4月2日电(记者朱晓颖)新冠肺炎疫情导致外贸大省江苏进出口、利用外资出现不同程度下滑。江苏多措并举，稳外贸、稳外资基本盘。 江苏实际利用外资规模十多年居全国第一，2019年实际使用外资总量达257亿美元。 当前，境外疫情呈现.....

详情页结果

828名江苏援湖北医疗队员平安返回

2020-3-31 18:58:00

http://www.chinanews.com/sh/2020/03-31/9143363.shtml

中新网南京3月31日电(记者朱晓颖通讯员苏卫萱)31日，又一批江苏援湖北医疗队员平安返回江苏。 此次返回队员共828人，分别为第五批江苏援湖北医疗队南京市队261人、无锡市队131人、苏州市队260人，第七批江苏援湖北医疗队南京鼓楼医院队162人，江苏援武汉公共卫生队及其他医疗队员14人。31日下午，队员乘坐的飞机分别降落在南京禄口国际机场(437人)、无锡硕放机场(391人)。 2月9日，第五批江苏援湖北医疗队抵达武汉，南京市队队员来自南京鼓楼医院、南京市第一医院等7家医院，无锡市队队员来自江南大学附属医院、无锡市人民医院等10家医院，苏州市队队员来自苏大附一院、苏大附二院等18家医院。他们奋战在华中科技大学同济医院光谷院区，累计收治患者437人，培训医务人员6042人次，截至3月30日，所负责病区患者均已“清零”。 2月13日，第七批江苏援湖北医疗队抵达武汉。其中，南京鼓楼医院队先后战斗在武汉市第一医院、湖北省人民医院东院区，累计收治患者109人，培训医务人员1740人次。 2月23日，江苏援武汉公共卫生队抵达武汉，此后奋战在武汉市经开(汉南)区，顺利完成社区防控任务。(完) 【编辑:苑菁菁】

推荐阅读

重庆最后一批援湖北医疗队员返渝
约7000名支援武汉医疗队员返程：一起拼、一起赢
吉林省支援雷神山医护人员返回 获家乡人民热情接机
“过水门”礼遇骑警开道护送 厦门驰援湖北275名医疗队员凯旋
今年前3个月江苏中欧班列开行列数和货值“双增长”