

Final: Search Engine

本次实验实现了一个 Web 搜索引擎，主要内容包括网页抓取、文本索引、链接分析、查询服务、个性化查询、Web 页面（图形化界面）和个性化推荐。

文件结构如下：

```
1  .
2  |-- data
3  |   |-- 1_spider.py           —爬虫
4  |   |-- 2_preprocess.ipynb    —数据预处理
5  |   |-- 3_term_frequency.ipynb —计算索引、tf-idf
6  |   |-- stopwords
7  |   `-- stopwords.txt
8  `-- web
9      |-- advanced_search.py     —高级检索算法
10     |-- app.py                 —入口文件
11     |-- public_index.py        —公共变量库
12     |-- recommend.py          —推荐算法
13     |-- search.py             —初级检索算法
14     |-- templates              —前端 html 模板
15     |   |-- advanced.html
16     |   |-- base.html
17     |   |-- home.html
18     |   |-- no_result.html
19     |   `-- result.html
20     `-- views                  —网页加载和渲染
21         |-- __init__.py
22         |-- advanced.py
23         |-- home.py
24         `-- result.py
25
```

网页抓取

选择 <https://www.gcores.com/articles>，该网站结构比较简单，每页有 12 篇文章，爬取 150 页，共 1800 篇，使用 requests 包进行爬取，beautifulsoup 进行解析。

爬虫逻辑简单来说是一个宽度优先搜索，遍历每一页，定位页面上所有文章的 url，然后再遍历文章，提取文章详情页中的标题、发表时间、内容信息；另外，由于后面要进行 pagerank 链接分析，还要提取页面里引用的所有 url。

爬虫的核心代码如下：

```
1  def get_article_links(uri): # 解析文章列表页
2      web_data = requests.get(url=uri, headers = request_headers)
3      status_code = web_data.status_code
4      if status_code == 200:
5          html_txt = web_data.text
6          # 解析为 soup 文档
7          soup = BeautifulSoup(html_txt, "lxml")
8          # 定位本页上所有 12 篇文章的 url
9          article_elements = soup.select("a.original_imgArea_cover")
10         # 遍历每篇文章
11         for article_element in article_elements:
12             # 提取 href 属性值
13             article_href = "https://www.gcores.com" +
14             article_element.get('href').strip()
15             # 获取每篇文章详情页的信息
16             get_article_info(article_href)
17             time.sleep(0.5)
18         else:
19             print("wrong status code!")
20
21 def get_article_info(article_href): # 解析文章详情页
22     web_data = requests.get(url=article_href, headers =
23     request_headers)
24     status_code = web_data.status_code
25     if status_code == 200:
26         html_txt = web_data.text
27     else:
28         print("wrong status code!")
```

```

27     soup = BeautifulSoup(html_txt, "lxml")
28     title = soup.find("h1", class_="originalPage_title").string #
标题
29     date = soup.find(class_="me-2 u_color-gray-
info").attrs['title']
30     date = datetime.datetime.strptime(date, "%Y-%m-%d %H:%M:%S") #
发表日期
31     segments = soup.find_all("span", attrs={"data-text": "true"})
32     segments_txt = []
33     for i in segments:
34
35         segments_txt.append(i.string.replace('\n', '').replace('\r', '').re
place('\t', '')) # 去掉文章内容中的换行和空格
36     content = ''.join(segments_txt)
37     links = []
38     contain_urls = soup.find_all("a", class_="md-link") # 引用链接
39     for i in contain_urls:
40         links.append(i.attrs["href"])
41
42     article_info = {
43         "title": title,
44         "url": article_href,
45         "date": date,
46         "content": content,
47         "links": links
48     }
49     with open('gcore_article.csv', 'a', encoding='gb18030',
newline='') as f:
50         csv_writer = csv.DictWriter(f,
fieldnames=article_info.keys())
51         csv_writer.writerow(article_info)

```

存储的 `gcore_article.csv` 格式如下：

| Title ▾ | Url ▲ | Date ▾ | Content ▾ | Links |
|-----------------------|--|---------------------|--|-----------------|
| 一条龙带你过游戏本地化全流程（二）： | https://www.gcores.com/articles/165406 | 2023-05-03 18:00:00 | 上一篇讲了一个游戏本地化项目前期接触和准备的一些事情。现在我 | [] |
| 2023核聚变广州站志愿者招募开始了，另 | https://www.gcores.com/articles/165407 | 2023-05-09 18:00:00 | 随着核聚变广州站的日期临近，我们将于5月17日正式放出本次核聚3 | [https://www.gc |
| 译介：电影《灌篮高手》的多层叙事—— | https://www.gcores.com/articles/165411 | 2023-05-03 22:26:21 | 即使您不喜欢这部电影，只要您还是漫画粉丝，井上我也想给您一个拥 | [https://www.gc |
| 译介 《节奏医生》与惊奇感 —— 《节 | https://www.gcores.com/articles/165415 | 2023-05-03 22:26:21 | 我觉得节奏医生与冰与火之舞的制作人Hafiz Azman是少有的会把传统 | [https://www.gc |
| 译介 如果电影像电子游戏一样被测评 | https://www.gcores.com/articles/165416 | 2023-05-03 21:53:11 | 作者：Dennis Farrell (@DennisFarrell) 翻译：罗皓曦原文地址：If F | [https://www.gc |
| 支线任务的心理动线 | https://www.gcores.com/articles/165418 | 2023-05-04 11:30:00 | 前言笔者在上一篇文章很宽泛地讨论了一下支线任务的设计目的，然 | [https://www.gc |
| 《哈迪斯》中的渐进式BOSS与渐收式BO | https://www.gcores.com/articles/165421 | 2023-05-04 09:00:00 | 哈迪斯是一个每局都需要从第一关重新开始的肉鸽游戏，一局时间在 | [] |
| 当年那些Flash游戏门户网站，现在怎么 | https://www.gcores.com/articles/165443 | 2023-05-22 13:00:00 | 前言：在我的上一篇文章中，我以2000年为节点，详细介绍了Newgr | [https://www.gc |
| 「已结束」机核邀请你提前看《速度与激 | https://www.gcores.com/articles/165444 | 2023-05-11 17:00:00 | 导语：5月16日（周二）20:00，机核将在 深圳市万达嘉映影城（深圳 | [https://www.gc |
| 译介 穿越十一款游戏，寻找设计优秀R | https://www.gcores.com/articles/165453 | 2023-05-04 22:45:00 | 前言首先，地牢是什么东西啊？我不想在定义上浪费太多时间，因此 | [] |
| 经典男装（一）礼服 | https://www.gcores.com/articles/165458 | 2023-05-05 00:46:00 | 礼服系统1. 晨礼服（赛马服）与晚礼服（大礼服）与短礼服（塔士多 | [] |
| 2023年四月玩的游戏总结与简评 | https://www.gcores.com/articles/165460 | 2023-05-07 15:51:31 | 河洛群侠传 评分6/10开放世界武侠游戏，被称为武侠题材的巫师3。 | [] |
| 随笔 一些近来感悟：症状、混乱和决心 | https://www.gcores.com/articles/165462 | 2023-05-05 00:16:47 | “我是什么人？”我觉得这不是，不可能是一个让人陌生的问题，是所 | [] |
| 《子身为蓝》世界观概述 宗教概述—— | https://www.gcores.com/articles/165465 | 2023-05-05 13:00:00 | *往生潮信仰“发源于日珥群岛，原是当地的小势土著信仰。当厄加德 | [] |
| 伦敦大学学院UCL《配音：译者导论》课 | https://www.gcores.com/articles/165473 | 2023-05-05 13:13:37 | 前言上周我参加了伦敦大学学院（UCL）翻译研究中心（CenTras）推 | [https://www.gc |
| 谁搞砸了游戏？ | https://www.gcores.com/articles/165474 | 2023-05-05 16:00:00 | 有经验的玩家，尽管已经玩了多年游戏，但难免会看走眼，玩家玩到 | [https://www.gc |
| 《深沉之火》：熊熊燃烧的弹反之魂 | https://www.gcores.com/articles/165478 | 2023-05-05 12:15:38 | 本文系转载文章，版权归属原作者。原文作者：未入流的菠萝包文章 | [https://www.gc |

文本索引

预处理

在构建索引之前，需要先对爬取的数据进行预处理，主要有三个步骤：

1. 设置停用词，这里合并了常用的几个中文停用词文档，并手动添加了一部分。
2. 对标题和文章内容分词，使用 jieba 库的搜索引擎模式分词函数

cut_for_search。

```
1 # 以内容为例，标题同
2 for i in range(articles.shape[0]):
3     if pd.isna(articles['content'][i]): # 注意有些文章里只有图
        片，没有文字
4         articles['content'][i] = ''
5         cut_list = [word for word in
        jieba.cut_for_search(articles['content'][i]) if word not in
        stop_words]
6         articles['content'][i] = cut_list
```

3. 处理引用链接：该网站的文章在引用其他链接时有一个跳转前缀，去掉这个前缀，并且只留下内部引用的链接，由于收集的所有网页都在这个网站里，外部链接在计算 pagerank 时用不到。

```

1 for i in range(articles.shape[0]):
2     t = articles['links'][i]
3     t = t.strip('[').strip(']').split(',')
4     for j in range(len(t)):
5         t[j] = t[j].strip(' ').lstrip('\\').rstrip('\\')
6         t[j] = t[j].replace('https://www.gcores.com/link?
target=', '') # 去除跳转前缀
7     t = [link for link in t if
"https://www.gcores.com/articles/" in link]
8     articles['links'][i] = t

```

4. 链接分析，见下节。

预处理后将数据存入新的表格 `gcore_article_cut.csv`，如下：

| Title ▲ | Url | Date | Content |
|-------------------------------|--|---------------------|--------------------------------------|
| 世代次世代最短全流程 | https://www.gcores.com/articles/174423 | 2023-11-30 14:32:39 | 写这是新尝试源自苦于B站短世代次世代全流程教程现状简短语 |
| 世界传奇体验报告明珠蒙尘 | https://www.gcores.com/articles/165840 | 2023-05-15 12:10:00 | 先说本作保证RTS精髓战略前提移劝退门槛特色多人合作沉浸式舞 |
| 世界感受蝴蝶效应蝴蝶效应乐趣小评别号 | https://www.gcores.com/articles/169649 | 2023-08-17 11:30:00 | 小时小时候无数次数无数次问父母人死世界回答一句肯定根本无法 |
| 世界尽头 | https://www.gcores.com/articles/167814 | 2023-07-01 01:38:14 | 正文结束大学生涯关上上门关上门一刻那一刻种种不舍是因为学生 |
| 世界科幻大会参与武田康广表达GAINAX热爱 | https://www.gcores.com/articles/172680 | 2023-10-22 11:30:00 | 刚刚结束雨果奖颁奖之夜科幻大会行程不多先说说总体观感第一 |
| 世界世界观女巫之心盐祭 | https://www.gcores.com/articles/168667 | 2023-07-22 14:24:20 | 前情提要长达百年罕谟辛斯战争冥龙塔里诺乌斯死河之雾中仓促 |
| 视觉展现只会害超侦探事件事件簿雾雨谜 | https://www.gcores.com/articles/168080 | 2023-07-08 09:00:00 | 小高刚剧本小松崎类角色设计高田雅史音乐三个元素弹丸论破系列 |
| 试译异世界诞生2006 | https://www.gcores.com/articles/167389 | 2023-06-21 14:00:20 | 作者伊藤ヒ口插图やすも序章亲爱儿子妈妈身体还好去往异世界 |
| 收获日戴着小丑面具可不搞笑角色 | https://www.gcores.com/articles/171551 | 2023-09-27 23:02:45 | 前言备受期待系列续作收获日暴力抢劫中心合作射击玩法依旧法 |
| 手残党胜利自制星球大战星球大战501军团 | https://www.gcores.com/articles/174073 | 2023-11-21 16:27:10 | 手残党涂比例501军团克隆人喷气背包士兵人偶人偶心心念念 |
| 手残仍大爱幽灵行者手体验初体验 | https://www.gcores.com/articles/172698 | 2023-10-23 23:00:00 | 化身恶灵赛博朋克风格城市中一名杀手少年中二幻想例外第一 |
| 手感更雷柏VT9PRO mini VT9PRO 彩色 彩色 | https://www.gcores.com/articles/174230 | 2023-11-27 14:13:00 | 上个上个月尝试雷柏VT9PRO V300 PRO两款双模无线鼠标时间段时 |
| 守望先锋粉丝本地化科研论文发表后记 | https://www.gcores.com/articles/173015 | 2023-11-13 12:08:02 | 序言科研论文社交媒体发布粉丝本地化网络志分析守望先锋守望 |
| 守望译事索杰恩 | https://www.gcores.com/articles/170014 | 2023-08-25 10:00:00 | 前线勇猛战士后方敏锐指挥黄金纪元走来开启新时代2022年月开 |
| 守望译事伊拉锐 | https://www.gcores.com/articles/174434 | 2023-11-30 10:00:00 | 月11日Overwatch开启第六赛季伊拉锐正式玩家相见守望先锋38 |
| 书单推荐电子子游游戏电子游戏史游戏史 | https://www.gcores.com/articles/167915 | 2023-07-04 12:44:25 | 玩家听到「游戏历史词时想起很大概率棋牌游戏非电子子游游戏 |
| 书评消失13级台阶风飘向 | https://www.gcores.com/articles/168831 | 2023-07-26 14:35:28 | 深渊如海不动如山本书高野明编剧转型出道出道作节奏控得优秀 |

索引构建

后面要实现关键词位置仅位于标题中的高级检索选项，因此在索引构建时就要构建两套，一套是文章内容+标题，另一套仅有标题，下面就以第一套为例说明。

首先从上面预处理后的表格中提取数据，构建正向的链接-词项-频率索引以及倒排索引，以便后面计算 tf-idf 时使用：

```

1 index = {} # 格式是 (url, (word, frequency))
2 for url, row in df.iterrows():

```

```

3     index[url] = {}
4     # 这里把标题内出现的词算作出现了5次，提高标题的权重
5     for word in row['title'].split(' '):
6         if word not in index[url]:
7             index[url][word] = 5
8         else:
9             index[url][word] += 5
10    for word in row['content'].split(' '):
11        if word not in index[url]:
12            index[url][word] = 1
13        else:
14            index[url][word] += 1
15    if index[url].get('') != None:
16        del index[url]['']
17
18    inverted_index = {} # 倒排索引，简单地把上面的索引倒过来，格式是 (word,
    (url, frequency))
19    for url, words in index.items():
20        for word, freq in words.items():
21            if word not in inverted_index:
22                inverted_index[word] = {}
23            inverted_index[word][url] = freq

```

然后计算 tf，词项在每篇文档中出现的次数，实际上只是把索引中的词项频率平滑处理了一下，取了对数：

```

1    tf = {}
2    for url, words in index.items():
3        url_tf = {}
4        for word, freq in words.items():
5            url_tf[word] = 1 + math.log2(freq)
6        tf[url] = url_tf

```

计算 idf，词项出现在所有文档中的比例，同样平滑处理：

```

1 idf = {}
2 for url, words in index.items():
3     for word, freq in words.items():
4         if word not in idf:
5             idf[word] =
math.log2(len(index)/len(inverted_index[word]))

```

计算 tf-idf，把上述二者相乘：

```

1 tf_idf = {}
2 for url, words in index.items():
3     url_tf_idf = {}
4     for word, freq in words.items():
5         url_tf_idf[word] = tf[url][word] * idf[word]
6     tf_idf[url] = url_tf_idf

```

在向量空间模型中，计算 query 和文档的余弦相似度时，需要除以文档的向量长度，实际上这个长度是固定的，不需要在线计算，在这里一并算出，检索时可以直接调用：

```

1 doc_len = {}
2 for url, tf_idf_dict in tf_idf.items():
3     tmp_len = 0
4     for word, tf_idf_val in tf_idf_dict.items():
5         tmp_len += pow(tf_idf_val, 2)
6     doc_len[url] = math.sqrt(tmp_len)

```

最后，维护一个所有文档中所有词项的集合（查询时要用到，如果搜索关键词分词的某个词项不在这个集合里，就不用算它的 tf-idf 了）：

```

1 word_set = []
2 for url, words in index.items():
3     for word, freq in words.items():
4         word_set.append(word)
5 word_set = sorted(set(word_set)) # 比起添加时筛选去重，最后统一去重的速度
快得多

```

用 json 格式保存 tf-idf、文档长度等检索时所需的常量。

链接分析

使用 pagerank 进行链接分析，评估网页权重：一个网页入链越多、被越重要的网页引用，它的 pagerank 值就越高。

用 networkx 库，遍历引用链接、构建有向图后可以直接调用库中 pagerank 函数计算。注意不是所有引用的链接都能指向爬取的网页，需要先行判断。

核心代码如下：

```
1 url_dict = {} # 维护一个 (url, links) 的链接字典
2 url_list = []
3 pagegraph = networkx.DiGraph()
4 for i in range(articles.shape[0]):
5     url_list.append(articles['url'][i])
6     url_dict[articles['url'][i]] = articles['links'][i]
7 for url, links in url_dict.items():
8     for link in links:
9         if link in url_list:
10             pagegraph.add_edge(url, link)
11 pr = networkx.pagerank(pagegraph, alpha=0.85)
12 page_rank_df = pd.Series(pr, name='page_rank')
13 # 处理一下，使其落在 [1,6] 区间内
14 page_rank_df = page_rank_df.apply(lambda x: math.log(x * 10000,
15     10) + 1)
16 page_rank_df.index.name = 'url'
17 page_rank_df.to_csv('page_rank.csv')
```

查询服务

提供基础查询、通配查询、站内查询、时间查询、标题检索、查询历史的服务。

基础查询

使用向量空间模型，计算关键词、历史记录和每篇文章的余弦相似度，进行基础查询。核心代码如下：

```
1  def search(input_keyword: str, search_history: str, is_title_only:
    bool = False):
2      """
3      输入：本次搜索的关键词、搜索历史（个性化检索）、是否只检索标题
4      输出：一个 (url, similarity) 列表，按相似度排序
5      """
6
7      # 判断是否只检索标题，确定使用哪一套索引，具体略
8      # ...
9      # 关键词和搜索历史分词，具体略
10     # ...
11
12     # 计算关键词和搜索历史的 tf-idf 值
13     input_tf_idf = {}
14     for word in split_keyword:
15         if word in util_word_set:
16             freq = split_keyword.count(word)
17             word_tf = 1 + math.log2(freq)
18             input_tf_idf[word] = word_tf * util_idf[word]
19         else:
20             input_tf_idf[word] = 0
21
22     history_tf_idf = {}
23     for word in split_history:
24         if word in util_word_set:
25             freq = split_history.count(word)
26             word_tf = 1 + math.log2(freq)
27             history_tf_idf[word] = word_tf * util_idf[word]
28         else:
29             history_tf_idf[word] = 0
30
31     # 计算和每篇文档的余弦相似度
32     url_sim = {}
33     for url, tf_idf_dict in util_tf_idf.items():
```

```

34     sim = 0
35     for word, word_tf_idf in input_tf_idf.items():
36         if word in tf_idf_dict:
37             sim += word_tf_idf * tf_idf_dict[word]
38     # 加入搜索历史, 按 0.03 的权重进行个性化搜索
39     # 必须先判断目前的相似度是否为零, 否则就会出现一种情况: 某个页面和搜索关键词的相似度为零, 但包含了某些搜索历史, 它也会出现在检索结果中, 但用户其实并不需要
40     if sim!=0:
41         for word, word_tf_idf in history_tf_idf.items():
42             if word in tf_idf_dict:
43                 sim += word_tf_idf * tf_idf_dict[word] * 0.03
44     sim /= util_doc_len[url] # 先前已经计算过的文档向量长度直接拿来用
45     # 余弦相似度和 pagerank 取调和平均
46     if url in page_rank.index:
47         sim = 2*(sim * page_rank.loc[url, 'page_rank']) / (sim + page_rank.loc[url, 'page_rank'])
48     else:
49         sim = 2*sim / (sim + 1)
50     url_sim[url] = sim
51     url_sim = sorted(url_sim.items(), key= lambda d:d[1], reverse=True)
52     # 这里只剔除了相似度为零的, 也可以筛选相似度必须大于某个最小值
53     url_sim = [(u,s) for u, s in url_sim if s!=0.0]
54
55     return url_sim

```

基础搜索页面如下:

Input keyword here:

SearchAdvanced

Search History

界面设计 微软 桌游 万智牌 博德之门 摄影 2023年终总结 核聚变 BOOOM 北京

下面实现高级搜索功能，参考了 Google 的高级搜索页面，如下：

Advanced

Search:

Include all(exact match):

add "AND" between the desired words (table AND chair)

Include any(exact match):

add "OR" between the desired words (table OR desk)

Not include:

add a minus sign "-" before an unwanted word (-2023)

Site / Domain:

search within a web site (wikipedia.org) or domain (.edu, .org, .gov)

Updated time:

Any

search for webpage updated in the specified time

Place:

Full text

Title only

search in full webpage / title only

Advanced Search

在高级搜索时，可以看到有一连串检索条件（全包含/包含任意/不包含/限定域名/限定时间/是否仅检索标题），采用的思路是首先进行基础搜索，然后再依次判定检索结果是否满足高级搜索的要求。

```
1 def advanced_search(form, url):
2     """
3     输入：高级检索选项的表格，一个网页链接
4     输出：该网页是否满足高级检索选项
5     """
6     # ...
7     # 判定过了所有条件，都没有返回 False
8     return True
9
10 @view_blue.route('/advanced', methods=['GET', 'POST'])
```

```

11 def advanced():
12     # 已经基础搜索过一轮，返回一个 (url, similarity) 列表
    final_result
13     # 再按照高级检索的选项筛选，依次判断每个网页是否满足要求
14     filter_list = []
15     for res in final_result:
16         if not advanced_search(form, res[1]):
17             filter_list.append(res[1])
18     final_result = [res for res in final_result if res[1] not
    in filter_list]

```

通配查询

分三种检索条件：包含以下所有词汇、包含以下任意词汇、不包含以下词汇。具体实现比较容易，遍历所有词项，依次判断它是否包含在检索结果里即可。

```

1 def advanced_search(form, url):
2     #...
3     # 1. 包含全部词汇
4     if and_words:
5         and_words_list = and_words.split('AND')
6         if '' in and_words_list:
7             and_words_list.remove('')
8         if ' ' in and_words_list:
9             and_words_list.remove(' ')
10        if is_title_only:
11            for word in and_words_list:
12                word = word.strip()
13                if word not in title:
14                    return False
15        else:
16            for word in and_words_list:
17                word = word.strip()
18                if word not in title and word not in content:
19                    return False
20
21    # 2. 包含任意词汇
22    if or_words:

```

```

23     or_words_list = or_words.split('OR')
24     if '' in or_words_list:
25         or_words_list.remove('')
26     if ' ' in or_words_list:
27         or_words_list.remove(' ')
28     if is_title_only:
29         for word in or_words_list:
30             word = word.strip()
31             if word in title:
32                 return True
33     else:
34         for word in or_words_list:
35             word = word.strip()
36             if word in title or word in content:
37                 return True
38     return False
39
40 # 3. 不包含词汇
41 if no_words:
42     no_words_list = no_words.split('-')
43     if '' in no_words_list:
44         no_words_list.remove('')
45     if ' ' in no_words_list:
46         no_words_list.remove(' ')
47     for word in no_words_list:
48         word = word.strip()
49         if word in title or word in content:
50             return False

```

站内查询

限制检索结果为某个网站或域名，判断用户输入是否在文章的 url 中即可。

```

1 def advanced_search(form, url):
2     # ...
3     # 4. 网站或域名
4     if site and (site not in url):
5         return False

```

时间查询

限制时间范围为一天/一周/一个月/一年内，使用 `datetime` 库将现在的时间减去文章发表时间即可。

```
1 def advanced_search(form, url):
2     #...
3     # 5. 时间
4     if time:
5         date = date.to_pydatetime()
6         now = datetime.now()
7         if time == 'Within a day' and now - date >
timedelta(days=1):
8             return False
9         elif time == 'Within a week' and now - date >
timedelta(days=7):
10            return False
11        elif time == 'Within a month' and now - date >
timedelta(days=30):
12            return False
13        elif time == 'Within a year' and now - date >
timedelta(days=365):
14            return False
```

标题查询

在基础查询中已经实现，传入一个 `is_title_only` 的布尔值，为真时就使用仅有标题的那套索引和 tf-idf 值等变量：

```
1 def search(input_keyword: str, search_history: str, is_title_only:
bool = False):
2     # 判断是否只检索标题，确定使用哪一套索引
3     if not is_title_only:
4         util_word_set = word_set
5         util_idf = idf
6         util_tf_idf = tf_idf
7         util_doc_len = doc_len
8     else:
```

```

9      util_word_set = word_set_title
10     util_idf = idf_title
11     util_tf_idf = tf_idf_title
12     util_doc_len = doc_len_title
13
14     #...
```

另外，在通配查询里也要设置是否仅在标题里检索，具体实现见上方“通配查询”一节。

查询历史

搜索历史存储为 cookies，使用 flask 内置的 `request.cookies` 实现上传/更新和获取，展示在基础搜索和搜索结果页面上。

```

1  @view_blue.route('/result')
2  def result():
3      # 获取检索历史
4      if request.cookies.get('search_history'):
5          search_history: list =
json.loads(request.cookies.get('search_history'))
6      else:
7          search_history = []
8      # ...
9      # 更新检索历史
10     if keywords not in search_history:
11         search_history.append(keywords)
12     if len(search_history) > 10:
13         search_history.pop(0)
14     resp.set_cookie('search_history', json.dumps(search_history),
max_age=60*60*24*30)
```

展示搜索功能，设置相同的检索词，首先是不设任何高级检索条件时，即基础搜索的结果如下：

核聚变

Search

About 124 results in 0.17 seconds. [Advanced](#)

大家都在核聚变买了啥？吉考斯工业 in 核聚变游戏节广州站

核聚变游戏节广州站已顺利结束，很高兴时隔许久又在线下见到了各位机组成员，感谢大家对吉考斯工业一如既往的支持！在两天的活动中，我们在现场也随机采访了前来光顾吉考斯的朋友，看看他们都买了些什么，以及期待在下一场核聚变游戏节见到各位！欢迎大家关注「吉考斯工业」新开设的B站频道，我们也...

<https://www.gcores.com/articles/167780> 2023-06-30 15:02:35

摄影分享 | 核聚变记录 DAY 1

34035步，1486次快门，156张照片，8个小时的记录，会是一生中值得反复重温的回忆吗？...

<https://www.gcores.com/articles/166958> 2023-06-11 19:00:00

「核聚变游戏节2023北京站」地狱挑战指南

当各位看到这篇文章的时候，第一天的活动已经结束了。这一次的地狱挑战我们仔细地思考了应该设置什么样的游戏，希望大家不光体验到困难，更让大家可以体会到一种学习的过程，话不多说，接下来就是我们给出的答案。人马的挑战科目二挑战剑术大师挑战奇怪的挑...

<https://www.gcores.com/articles/170584> 2023-09-09 20:00:00

摄影分享 | 核聚变·北京

本来买的两天的票，结果深圳下了场大雨周五的航班取消，改到周六的航班错过了一天，不过好在周日的行程保住了。省下一天的住宿费，改签的机票便宜了一百多块，扣除门票的损失，剩下的钱拿去续了一年的GPASS还有富裕...

<https://www.gcores.com/articles/170747> 2023-09-12 11:00:00

2023「核聚变游戏节」成都站来了，11月不见不散！

成都的朋友们大家好！我们信守诺言，再一次把核聚变带到了成都！和大家再次见面心情也是十分的激动，接下来，还是赶紧给大家介绍一下成都站的相关情报吧！核聚变游戏节 2023 成都站 详情 2023 年 11 月 25 日（周六）?02 9: 00-17: 00（16: 00停止入场...

<https://www.gcores.com/articles/173397> 2023-11-06 11:00:00

Recommendation

北京站

GPASS

摄影

BOOOM

售罄

游戏

志愿者

全览

吉考斯

限定

Search History

界面设计

桌游

万智牌

博德之门

摄影

2023年终总结

核聚变

BOOOM

暴雪

微软

测试标题检索，包含关键词“2023”，不包含“北京”，如下：

Advanced

Search:

核聚变

Include
all(exact
match):

2023

add "AND" between the desired words (table AND chair)

Include
any(exact
match):

add "OR" between the desired words (table OR desk)

Not include:

-北京

add a minus sign "-" before an unwanted word (-2023)

Site / Domain:

search within a web site (wikipedia.org) or domain (.edu, .org, .gov)

Updated time:

Any

search for webpage updated in the specified time

Place:

☐ Full text
☒ Title only

search in full webpage / title only

Advanced Search

核聚变

Search

About 16 results in 0.53 seconds. [Advanced](#)

[2023广州核聚变摄影总结](#)

这几日私事较多，没有及时发出在之前的核聚变day1速报当中，有不少朋友为我的相片点赞，很感谢大家，其实对于我来说，拍摄这些东西也是一个很个人的行为。这其实代表这两种说法：一是在相片的视觉构成当中，会有我自身的大量选择。而另一方面则是对于更多的朋友来说，这些相片同时也是你们的体验， ...

<https://www.gcores.com/articles/167198>2023-06-16 20:42:00

[核聚变2023 广州站摄影分享](#)

今年人真多，基本上游戏排队都是二三十分钟起的了，基本就是在排队中度过的，只带了个长焦，所以基本都是大头照了，排队间隙随机抓拍了一些。先来雨川老师，第一张快门慢了，抓拍有点糊，不过还是好看，发上来吧二七老师，酷啊鸭老师和老白，老白太逗了，兑奖唠了一阵，可惜我两天都是B.....

<https://www.gcores.com/articles/167049>2023-06-13 12:00:00

[我曾害怕核聚变，直到站在摊位前！核聚变 2023 回顾](#)

几位年纪尚轻的玩家聚在电脑前，在我们的游戏中一次又一次死亡、复活、重试。这游戏显然还是有些难度的，足以我们的美术大佬将游戏过程戏称为“坐牢”。因此把游戏带来展会前，我格外担心游戏的难度会让人望而却步。为了避免这一问题，我还专门在展会前紧急添加了一些官方“辅助”手段，能够调节游戏...

<https://www.gcores.com/articles/170839>2023-09-13 22:33:02

[迟来的2023核聚变胶卷摄影分享](#)

作为八年老听众，却因为种种原因今年第一次来核聚变。无比熟悉的声音第一次以真人出现在面前，让我既害羞又兴奋。胶卷终于扫描出来了，回顾自己的作品，发现构图平庸，各种没合焦。整理了几张还能看的分享一下。当时觉得有点心疼胶卷没有拍很多，现在有点后悔没有多拍。技术不行就应该用数量去弥补的哈...

<https://www.gcores.com/articles/167525>2023-06-25 13:15:53

Recommendation

创作者

BOOOM

北京站

摄影

能玩到

奖品

游戏

Tour

不见不散

招募

Search History

界面设计

桌游

万智牌

博德之门

摄影

2023年终总结

核聚变

BOOOM

暴雪

微软

可以看到，检索结果标题内都有关键词“2023”，不再有“北京”。

测试时间检索，设置限制时间为一个月内，如下：

Advanced

Search:

核聚变

Include all(exact match):

add "AND" between the desired words (table AND chair)

Include any(exact match):

add "OR" between the desired words (table OR desk)

Not include:

add a minus sign "-" before an unwanted word (-2023)

Site / Domain:

search within a web site (wikipedia.org) or domain (.edu, .org, .gov)

Updated time:

Within a month

▼

search for webpage updated in the specified time

Place:

☒ Full text

☐ Title only

search in full webpage / title only

Advanced Search

核聚变

Search

About 9 results in 0.23 seconds. [Advanced](#)

12月吉考斯继续和你上海相见!

成都核聚变的落幕代表着今年所有的核聚变游戏节已经画下句号，感谢来到现场的每一位朋友，谢谢你们的支持！12月1日至3日，吉考斯将闪现上海，继续和大家在凡几市集见面！活动时间 TIME2023 /12 / 1 - 2023 / 12 / 3?0?2周五至周日 12:30-21:00活...
<https://www.gcores.com/articles/174177> 2023-11-27 10:00:00

2023成都核聚变小枪repo

开展前——这次是在booom区，还是想跟制作者交流一下子啥的，前一次因为组队完大家鸽了，没能参加，这也算另一种意义上地参加了booom吧哈哈哈哈哈——大家都特别好，我们区长是晚安哥，终于见到真容了！美瞳度数有点不够，当时特别不敢上去认人，旁边的兄弟说那就是晚安哥，快去要工牌，才踏出了...
<https://www.gcores.com/articles/174339> 2023-11-28 12:00:00

摄影分享 | 我的2023年终回顾

2023，是我超级宅男的一年。看了120部电影，9部舞台剧，30本书，打通了好几款游戏。与此同时今年，我偶没有出门旅行。除了常规的、几次节假日往返湖南其它，所谓最远的旅行甚至只是从深圳，往广州参加核聚变而已。甚至，今年也没认真拍什么作品虽然换了新相机，但工作之余，也没整心思认...
<https://www.gcores.com/articles/174963> 2023-12-12 12:23:17

十余年后，一场全新的切尔诺贝利硬核之旅 | 《潜行者2》开发公司GSC访谈

由 GSC Game World开发、备受玩家期待的《潜行者2：切尔诺贝利之心》参加了本次2023「核聚变游戏节」成都站，并在现场进行国内首次实机试玩活动。借这个机会，我们也与GSC来到中国的几位开发者们面对面地交流了一下，谈了谈有关《潜行者2》的游戏世界，以及它所涉及的一些细节...
<https://www.gcores.com/articles/174284> 2023-12-05 20:00:00

核聚变游戏节 2023 成都站创作者展览作品总结

2023年的核聚变之旅结束了。今年我们在会场的品牌区中，为机核社区中的创作者们开辟了一面展览墙，将大家的画作展示给来到核聚变的大家。每一次展览都收获了到场朋友们的好评，许多人在展览面前驻足欣赏，拍照留念。但是由于现场展位有限，我们无法将所有作品都进行展出，也有许多朋友没能前往核聚...
<https://www.gcores.com/articles/174462> 2023-11-30 17:30:00

Recommendation

北京站

GPASS

摄影

BOOOM

售罄

游戏

志愿者

全览

吉考斯

限定

Search History

界面设计

桌游

万智牌

博德之门

摄影

2023年终总结

核聚变

BOOOM

暴雪

微软

可以看到，检索结果显示的日期都在一个月內。

个性化查询

考虑到上面已经记录了用户的搜索历史，可以用它提供个性化查询。在基础搜索里，除了本次搜索的关键词外，也将历史搜索记录传进来同样进行分词和余弦相似度计算，并加权合并到搜索关键词的相似度中，设置权重为 0.03，具体实现见上方“基础查询”一节。

要特别注意的是，在合并历史记录相似度时，必须先判断文档和搜索词的相似度是否为零，不为零才合并，否则就会出现一种情况：某个页面和搜索关键词的相似度为零，但包含了某些搜索历史，它也会出现在检索结果中，但用户其实并不需要。

Web 页面（图形化界面）

用 flask 实现，采用 jinja2 渲染网页，通过 `render_template` 函数把 html 文件中需要的变量从视图函数中传递过去。把 html 模板文件都放在默认的子文件夹 `/templates`，视图函数文件放在文件夹 `/views`。

在 css 和 js 上，调用了 `flask_bootstrap` 库内置的图形化样式。

一共实现了 5 个 html 模板，如下：

1. 基础模板 `base.html`，定义了网页结构、语言、编码等。
2. `home.html`，基础搜索页面：



Input keyword here:

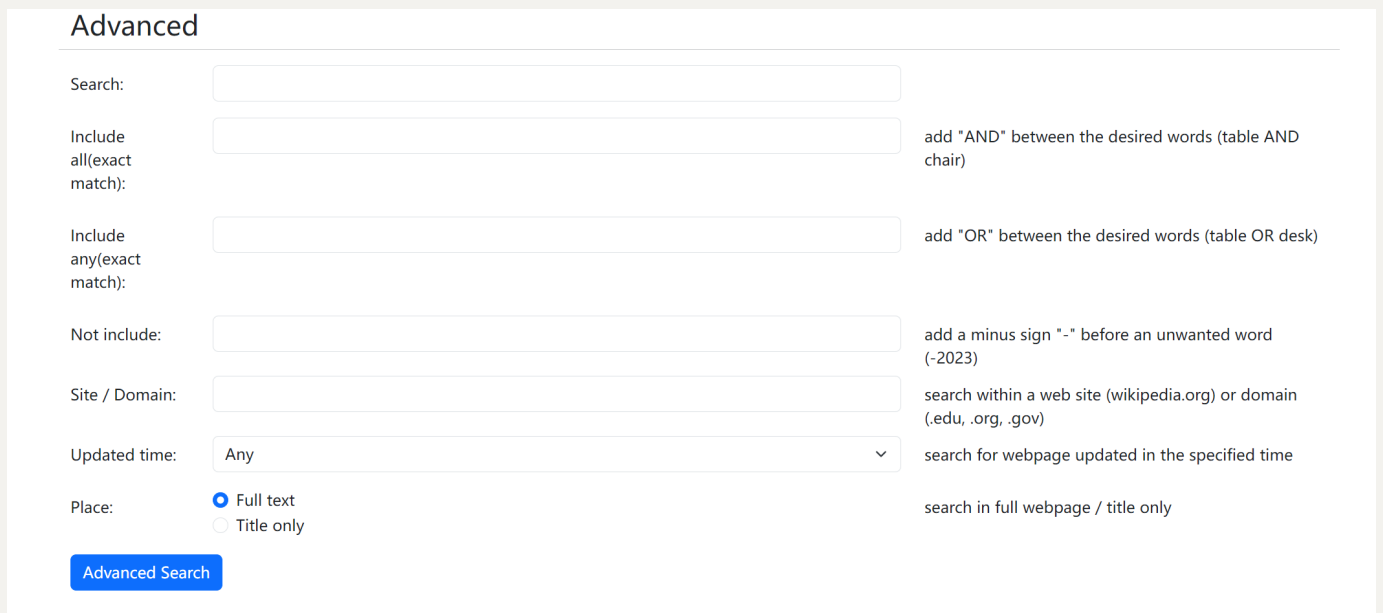
Search Advanced

Search History

界面设计 微软 桌游 万智牌 博德之门 摄影

2023年终总结 核聚变 BOOOM 北京

3. `advanced.html`，高级搜索页面：



Advanced

Search:

Include all(exact match): add "AND" between the desired words (table AND chair)

Include any(exact match): add "OR" between the desired words (table OR desk)

Not include: add a minus sign "-" before an unwanted word (-2023)

Site / Domain: search within a web site (wikipedia.org) or domain (.edu, .org, .gov)

Updated time: search for webpage updated in the specified time

Place: ☒ Full text ☐ Title only search in full webpage / title only

Advanced Search

4. `result.html`，展示搜索结果：

核聚变

Search

About 124 results in 0.17 seconds. [Advanced](#)

大家都在核聚变买了啥？吉考斯工业 in 核聚变游戏节广州站

核聚变游戏节广州站已顺利结束，很高兴时隔许久又在线下见到了各位机组成员，感谢大家对吉考斯工业一如既往的支持！在两天的活动中，我们在现场也随机采访了前来光顾吉考斯的朋友，看看他们都买了些什么，以及期待在下一场核聚变游戏节见到各位！欢迎大家关注「吉考斯工业」新开设的B站频道，我们也...

<https://www.gcores.com/articles/167780> 2023-06-30 15:02:35

摄影分享 | 核聚变记录 DAY 1

34035步，1486次快门，156张照片，8个小时的记录，会是一生中值得反复重温的回忆吗？...

<https://www.gcores.com/articles/166958> 2023-06-11 19:00:00

「核聚变游戏节2023北京站」地狱挑战指南

当各位看到这篇文章的时候，第一天的活动已经结束了。这一次的地狱挑战我们仔细地思考了应该设置什么样的游戏，希望大家不光体验到困难，更让大家可以体会到一种学习的过程，话不多说，接下来就是我们给出的答案。人马的挑战科目二挑战剑术大师挑战奇怪的挑...

<https://www.gcores.com/articles/170594> 2023-09-09 20:00:00

摄影分享 | 核聚变·北京

本来买的两天的票，结果深圳下了场大雨周五的航班取消，改到周六的航班错过了一天，不过好在周日的行程保住了。省下两天的住宿费，改签的机票便宜了一百多块，扣除门票的损失，剩下的钱拿去续了一年的GPASS还有富裕...

<https://www.gcores.com/articles/170747> 2023-09-12 11:00:00

2023「核聚变游戏节」成都站来了，11月不见不散！

成都的朋友们大家好！我们信守诺言，再一次把核聚变带到了成都！和大家再次见面心情也是十分的激动，接下来，还是赶紧给大家介绍一下成都站的相关情报吧！核聚变游戏节 2023 成都站 详情 2023 年 11 月 25 日（周六）?0?2 9: 00-17: 00（16: 00停止入场...

<https://www.gcores.com/articles/173397> 2023-11-06 11:00:00

Recommendation

北京站 GPASS 摄影
BOOOM 售罄 游戏
志愿者 全览 吉考斯
限定

Search History

界面设计 桌游 万智牌
博德之门 摄影
2023年终总结 核聚变
BOOOM 暴雪 微软

5. no_result.html，找不到任何结果时展示：

铈

Search

About 0 result in 2.46 seconds.

找不到和您查询的“**铈**”相符的内容或信息。

建议：

- 请检查输入字词有无错误。
- 请尝试其他查询词。
- 请改用较常见的字词。
- 请减少查询字词的数量。

个性化推荐

用户查询某条关键词时，在查询结果右侧显示 10 个相关词条，也就是一个最简单的推荐系统。

推荐思路是借助 `jieba.analyse` 的关键词提取函数 `extract_tags`，对本次查询结果的文档提取 tf-idf 最高的前 20 个关键词，合并、计算其权重，最后按权重排序，选出前 10 个。

`extract_tags` 可以返回关键词在文档中的权重，将其乘以文档和搜索词的相似度，就得到了关键词的最终权重。

起初，我计算了搜索结果的全部文档的关键词，但搜索结果较多时，推荐算法会非常耗时，于是简单地改成了只计算前 30 篇文档。

核心代码如下：

```
1  def recommend(results, keywords):
2      """
3      输入：本次搜索的结果、本次搜索的关键词（用于排除）
4      输出：一个推荐词列表
5      """
6
7      split_keyword = list(jieba.cut_for_search(keywords))
8      split_keyword.sort()
9      if '' in split_keyword:
10         split_keyword.remove('')
11     if ' ' in split_keyword:
12         split_keyword.remove(' ')
13
14     # 对每篇本次搜索的结果文档提取 20 个关键词，合并、计算其权重，最后按权重
    排序推荐
15     # 搜索结果已经是加入搜索历史的个性化结果了，不用再算一次
16     extract_tags = {}
17     for res in results[: (30 if len(results)>=30 else
len(results))]: # 性能所限，这里只简单地取前 30 篇；稍微复杂的实现是根据文
    档的相似度筛选
18         tmp_tags =
jieba.analyse.extract_tags(webpage.loc[res[1], 'title'], topK=3,
withWeight=True)
19         # 上面的提取会返回一个 [word, weight] 的列表
20         for word, weight in tmp_tags:
21             if word not in extract_tags:
22                 # 因为这里的 weight 是提取的关键词对所在文档的权重，还要乘
    以文档本身和 query 的相似度
23                 extract_tags[word] = res[4]*weight
24             else:
25                 extract_tags[word] += res[4]*weight
26         content = webpage.loc[res[1], 'content']
27         if not pd.isna(content):
```

```

28         tmp_tags = jieba.analyse.extract_tags(content,
topK=20, withWeight=True)
29         for word, weight in tmp_tags:
30             if word not in extract_tags:
31                 extract_tags[word] = res[4]*weight
32             else:
33                 extract_tags[word] += res[4]*weight
34
35         # 剔除搜索的关键词
36         for word in split_keyword:
37             if word in extract_tags.keys():
38                 del extract_tags[word]
39
40         extract_tags_list = sorted(extract_tags.keys(), key=lambda
d:extract_tags[d], reverse=True)
41         # 去掉数字
42         for word in extract_tags.keys():
43             if is_number(word):
44                 extract_tags_list.remove(word)
45
46         # 返回权重最高的前10个
47         return extract_tags_list[: (10 if len(extract_tags_list)>=10
else len(extract_tags_list))]

```

结语

本次实验最不熟悉、查询资料最多的部分其实是前后端信息交互，尽管借助 flask 框架已经让 Web 页面搭建简单了很多，但仍然花费了很长时间；相较而言，倒排索引、向量空间模型的实现其实都不算困难。

在个性化推荐的部分思考了很多，另一个想法是衡量词项之间的相似度，做一个类似倒排的 tf-idf，将词项也用向量表示，向量的每一项是词项在不同文档里的 tf-idf 值，然后计算和搜索词最相近的前 10 个词项，由于时间原因没有实现。在最后的实现中，由于性能所限，只统一计算了前 30 篇文档，但对不同检索词，搜索结果的情况区别可能很大，后续可以再根据文档与检索词的相似度，动态地选择不同数量的文档。