

信息检索系统原理大作业——基于南开大学校内网页的web搜索引擎

2012522 郭坤昌 计算机科学与技术

序言

该搜索引擎基于南开大学校内网站，借助全文搜索引擎Whoosh构建索引并提供部分基本查询功能，使用PageRank算法、按时间排序与站点过滤提供高级查询功能，并使用文本聚类与偏好分析结合的方法提供个性化查询功能。在个性化推荐方面，使用基于物品的协同过滤算法进行网页推荐。

关键词

链接分析，基于物品的协同过滤算法，K-Means聚类，个性化

项目结构

1	...	
2	...	
3	MySearchEngine	
4	├ app.py	应用主体
5	├ cluster.py	文本聚类
6	├ history.py	日志记录
7	├ index	
8	│ └ index.py	索引构建
9	│ └ whoosh_index	索引存储
10	│ └ ...	
11	├ page_rank.py	链接分析
12	├ query.py	查询解析
13	├ recommend.py	个性推荐
14	├ search.py	多种查询功能 + 个性查询
15	├ spider	爬虫框架
16	│ └ scrapy.cfg	
17	│ └ spider	
18	│ │ └ __init__.py	
19	│ │ └ items.py	定义网页类
20	│ │ └ middlewares.py	
21	│ │ └ pipelines.py	网页存储到MongoDB
22	│ │ └ settings.py	
23	│ │ └ spiders	
24	│ │ │ └ __init__.py	
25	│ │ │ └ nankai.py	主要爬虫文件
26	...	
27	...	

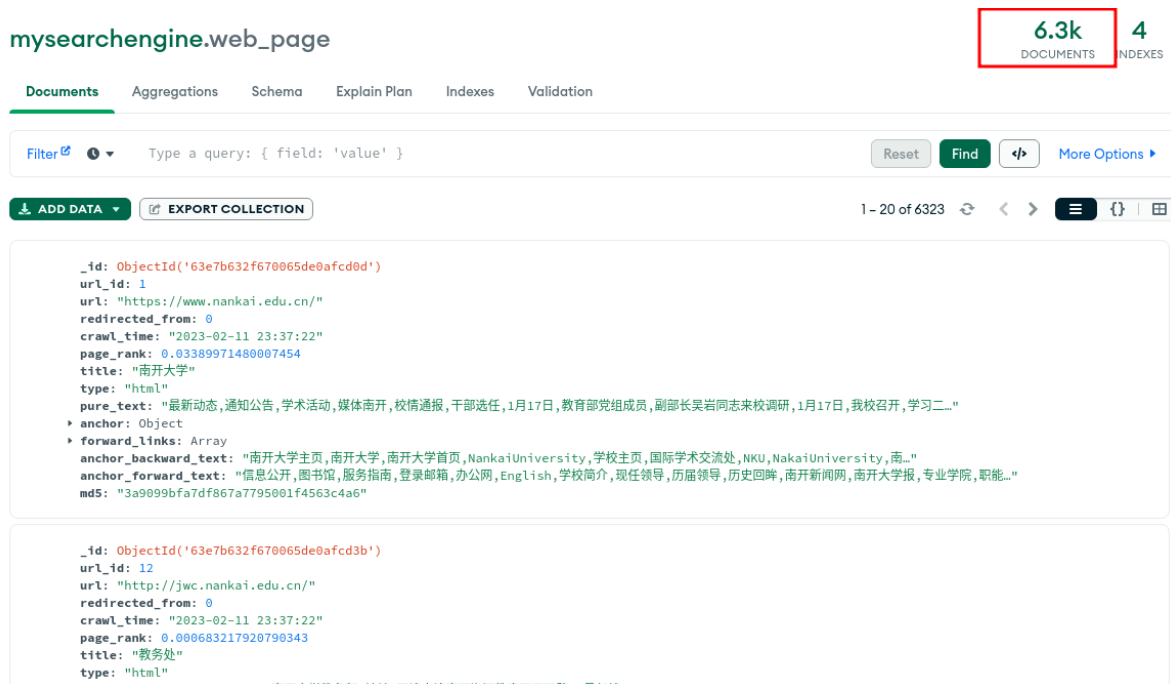
网页抓取

在本搜索引擎构建过程中，需要爬取网页的url、标题、纯文本、锚文本、超链接等。其中url是访问网页的地址，标题、纯文本、锚文本用以构建不同域上的索引，超链接用以爬取链接网页并参与PageRank计算。

由于scrapy框架有分布式、高并发、自动过滤重复请求、自动断点续爬等优点，因此使用该框架进行网页爬取。由于数据使用JSON格式存储较为方便，因此选用MongoDB作为数据库存储网页文档信息。为了减少存储空间及方便表示，建立自增id与网页url的唯一对应表，并在后续记录前向链接关系时，使用网页id作为网页唯一标识。

爬取过程以[南开大学首页](#)为起始站点。获取爬取网站所有锚的超链接，添加到下一步爬取的列表中。记录下当前网页的所有信息（url、标题、纯文本、锚文本、爬取时间），生成的item对象由pipeline存储到MongoDB数据库中；对于爬取列表中的超链接，则生成request对象，爬取该网页。该过程递归执行，直到爬取完毕。本次实验中，共爬取了约六千个网页，最后无法继续爬取是因为请求超时（可能为没有在校内连接导致）。

最终爬取网页结果如下：



索引构建

分别在url、标题、网页纯文本、锚文本域上建立索引，其中url、标题、网页纯文本域上的词项索引到该网页的id，锚文本域上的词项索引到跳转的网页的id。

借助开源工具Whoosh（Whoosh是基于Python的全文搜索引擎，相较于ElasticSearch更轻量级且容易使用）构建索引，并使用jieba中文分词。

数据修正

在构建索引之前，首先对索引域的数据进行修正，主要包括以下三个部分：

1. 修正前向链接。网页在实际爬取时，往往出现重定向问题（详见其他问题部分），因此需要将重定向的网页编号替换为重定向后的网页编号（解决过程也请查看其他问题的这一部分）。
2. 补充后向链接文本。指向该网页的锚包含的文本也包含了该网页的相关信息，因此需要将锚文本添加到锚的超链接对应的网页文档中，作为索引域。

```
1 def implement_backward_anchor_text(self): # backward anchor text helps with
  searching anchor href
2     def filter_str(str): # remove all whitespace. although filtered in
  spider, there are still some whitespace
3         return re.sub(r'\s+', '', str)
4     def get_type(url):
5         return re.sub(r'/', '', url.split('.')[1])
```

```

6      web_page_ids = [web_page_id['url_id'] for web_page_id in
self.web_page_collection.aggregate([{'$project': {'url_id': 1}}])]
7      for web_page_id in web_page_ids:
8          web_page = self.web_page_collection.find_one({'url_id':
web_page_id})
9          anchor = web_page['anchor']
10         for href in anchor:
11             text = filter_str(anchor[href])
12             href_id = self.url_collection.find_one({'url': href})['url_id']
13             redirect = self.redirect_collection.find_one({'from_url_id':
href_id})
14             redirect_from = 0
15             if redirect is not None:
16                 redirect_from = href_id
17                 href_id = redirect['to_url_id']
18             if href_id in web_page_ids:
19                 if text == '':
20                     continue
21                 anchor_backward_text =
self.web_page_collection.find_one({'url_id': href_id})
['anchor_backward_text']
22                 if anchor_backward_text == '':
23                     new_anchor_backward_text = text
24                 else:
25                     anchor_backward_text_list =
anchor_backward_text.split(',')
26                     if text in anchor_backward_text_list:
27                         continue
28                     new_anchor_backward_text = anchor_backward_text + ',' +
text
29                 self.web_page_collection.update_one({'url_id': href_id},
{'$set': {'anchor_backward_text': new_anchor_backward_text}})

```

3. 合并重复网页。一些网页除了url不同，其实是同一个网页，一个简单示例如下：

显然，重复网页相关性得分相同，将影响整个计算结果。

url: <https://fy.nankai.edu.cn/>
filtered_title: 南开大学附属医院
pure_text: 南开大学附属医院版权所有, 联系方式
anchor_backward_text: 附属医院
anchor_forward_text: 附属医院除夕夜慰问坚守岗位一线, 市卫健委安全保卫处到我院检查安, 附属医院在学校, 创最佳党日, 活, 坚守在, 实验室里的附院检验人, 附属医院领导
page_rank: 4.512787234276739e-05
crawl_time: 2023-02-09 00:47:38
score: 0.48925852800743513

url: <https://fy.nankai.edu.cn/main.htm>
filtered_title: 南开大学附属医院
pure_text: 南开大学附属医院版权所有, 联系方式
anchor_backward_text:
anchor_forward_text: 附属医院除夕夜慰问坚守岗位一线, 市卫健委安全保卫处到我院检查安, 附属医院在学校, 创最佳党日, 活, 坚守在, 实验室里的附院检验人, 附属医院领导
page_rank: 0.00020389410540924208
crawl_time: 2023-02-09 00:47:39
score: 0.48925852800743513

仅url不同

使用文档的文本计算md5作为文档的唯一标识。首先选出具有相同内容的文档，决定保留第一个爬取的文档，在锚的超链接列表中，将其他相同文档替换为保留文档，最终在表中删除被替换的文档。

```

1  def collapse_the_same(self): # eliminate same page. md5 should be
calculated in spider, but I found it too late
2      def get_md5(str):
3          return md5(str.encode('utf-8')).hexdigest()

```

```

4     web_page_ids = [web_page_id['url_id'] for web_page_id in
self.web_page_collection.aggregate([{'$project': {'url_id': 1}}])]
5
6     for web_page_id in web_page_ids:
7         web_page = self.web_page_collection.find_one({'url_id':
web_page_id})
8         page_md5 = get_md5(web_page['title'] + web_page['pure_text'] +
web_page['anchor_backward_text'] + web_page['anchor_forward_text'])
9         self.web_page_collection.update_one({'url_id': web_page_id},
{'$set': {'md5': page_md5}})
10        url_id_lists = [url_id['url_id'] for url_id in
self.web_page_collection.aggregate([{'$group': {'_id': '$md5', 'url_id':
{'$push': '$url_id'}, 'count': {'$sum': 1}}}, {'$match': {'count': {'$gt':
1}}}, {'$project': {'url_id': 1}}])]
11        for url_id_list in url_id_lists:
12            url_id_to_keep = url_id_list[0]
13            url_id_to_delete = url_id_list[1:]
14            for url_id in url_id_to_delete:
15                forward_links_list =
self.web_page_collection.find({'forward_links': {'$in': [url_id]}})
16                for forward_links in forward_links_list:
17                    new_forward_links = (set(forward_links['forward_links']) -
{url_id}) | {url_id_to_keep}
18                    self.web_page_collection.update_one({'url_id':
forward_links['url_id']}, {'$set': {'forward_links':
list(new_forward_links)}})

```

以如下例子为例，最终运算结果中，表示首页的 `index.htm` 实际上与原网站相同，因此被替换。消除了重复网页。

```

_id: ObjectId('63e7b635f670065de0afd006')
url_id: 700
url: "http://binhai.nankai.edu.cn/index.htm"

```

替换为了 <http://binhai.nankai.edu.cn>

具体替换结果如下，经验证如下替换结果其实不完全正确，仍有部分网站不重复但被替换。原因是爬取过程采集网页纯文本时，为了简便，没有采集所有文本标签，即只采集了 `<p><h1><h2>`，因此若网页除了上述部分，仍存在不同的话，会被替换。当然，考虑全部文本标签，则该问题迎刃而解。

```

/home/bill/anaconda3/envs/mysearchengine/bin/python /home/bill/Desktop/information-retrieval-system/hw5/MySearchEngine/index/index.py
keep: 927 replace: [928, 929, 930, 1004, 1006, 3312, 3886, 3952]
keep: 7167 replace: [7147, 7146]
keep: 129 replace: [826, 786, 827, 823, 806, 825, 821, 824, 817, 822, 816, 813, 815, 809, 814, 819, 810, 808, 820, 818, 812, 1422, 8194]
keep: 3620 replace: [3668]
keep: 3248 replace: [3271]
keep: 256 replace: [253, 250, 251, 247, 249, 252, 232, 231, 7038, 7067, 7046, 7081, 7080, 7079, 7066, 7065, 7063, 7061, 7062, 7037]
keep: 991 replace: [1258, 1272, 1405, 1433, 1675, 1690, 1706, 1707, 1809, 1812, 1815, 1829, 1830, 1832, 1833, 2770, 3116, 3305, 3315, 3619, 3722, 3877, 6975]
keep: 7119 replace: [7116]
keep: 1572 replace: [2768]
keep: 3745 replace: [3808]
keep: 7050 replace: [7189, 7186, 7198]
keep: 1327 replace: [1338]

```

使用Whoosh建立索引

建立索引过程需要指定建立索引的模式，在类初始化时完成，需要在url，标题，网页纯文本（<p> 标签文本等），后向链接文本（指向该网页的锚文本），前向链接文本（一定程度上也与该网页有关）上建立索引。之后在计算完PageRank得分后，使用Whoosh建立索引。

```
1 #...
2 from whoosh.index import create_in
3 from whoosh.fields import Schema, TEXT, ID, DATETIME, NUMERIC
4 from jieba.analyse import ChineseAnalyzer
5 import datetime
6
7 class Indexer():
8     def __init__(self):
9         #...
10        self.analyzer = ChineseAnalyzer()
11        self.schema = Schema(
12            url_id=ID(unique=True, stored=True),
13            url=ID(unique=True, stored=True),
14            filtered_title=TEXT(stored=True, analyzer=self.analyzer),
15            pure_text=TEXT(stored=True, analyzer=self.analyzer),
16            anchor_backward_text=TEXT(stored=True, analyzer=self.analyzer),
17            anchor_forward_text=TEXT(stored=True, analyzer=self.analyzer),
18            page_rank=NUMERIC(stored=True),
19            crawl_time=DATETIME(stored=True),
20        )
21        if not os.path.exists('index/indexdir'):
22            os.mkdir('index/indexdir')
23        self.ix = create_in('index/indexdir', self.schema)
24        self.writer = self.ix.writer()
25        self.page_rank_calculator = PageRankCalculator()
26
27        #...
28        def build_index(self):
29            web_page_ids = [web_page_id['url_id'] for web_page_id in
30self.web_page_collection.aggregate([{'$project': {'url_id': 1}}])]
31            doc_counter = 0
32            for web_page_id in web_page_ids:
33                web_page = self.web_page_collection.find_one({'url_id':
34web_page_id})
35                # write data to whoosh
36                self.writer.add_document(
37                    url_id=str(web_page['url_id']),
38                    url=web_page['url'],
39                    filtered_title=web_page['filtered_title'],
40                    pure_text=web_page['pure_text'],
41                    anchor_backward_text=web_page['anchor_backward_text'],
42                    anchor_forward_text=web_page['anchor_forward_text'],
43                    page_rank=web_page['page_rank'],
44
45                    crawl_time=datetime.datetime.strptime(web_page['crawl_time'], '%Y-%m-%d
46%H:%M:%S'),
47                )
48                doc_counter += 1
49                if doc_counter % 1000 == 0:
50                    self.writer.commit()
51                    self.writer = self.ix.writer()
```

```

48         if doc_counter % 1000 != 0:
49             self.writer.commit()
50             self.writer = self.ix.writer()

```

PageRank

结果排序依照概率检索模型BM25F得分和PageRank得分的加权评分得到。为了优化查询体验，考虑到数据大多数时候是过剩的，因此查询结果应当尽量准确，因此选择给前k个结果赋予较高的相关度得分权重，降低PageRank得分的权重，以提高前面查询结果的准确率。同时按照最高得分进行归一化，这样当相关度得分相近时，可以根据权威值差异进行排序。BM25F得分主要由Whoosh后端计算得到，这里主要介绍PageRank的计算过程。

爬取过程记录了当前网页的前向连接网页，这样就可以使用PageRank算法衡量url的权威性。PageRank的计算方法有很多，这里不使用构造转移矩阵的方法，而是根据定义直接迭代计算。一个网页的PageRank值，由指向该网页的后向网页将自身权重按出度均分，传递给该网页；考虑到随机跳转，设阻尼系数为 α ，总网页数量为 N ，则随机跳转到其他网页的概率为 $(1 - \alpha)/N$ 。每次迭代时，网页 A 的PageRank值由以下公式计算得到：

$$PageRank(A) = \alpha \sum_{in} PageRank(in) / OutDegree(in) + (1 - \alpha) / N$$

使用有向图按定义实现

实现过程使用Python库networkx中的Digraph（有向图）类。首先从爬取结果中建立有向图，接着将所有没有出度的节点建立与向其他所有节点的边，即为了模拟随机跳转；接着在每轮迭代过程中，统计所有网页PageRank值变化之和，若变化小于一定限度，则认为继续迭代对网页权重影响不大，结束计算，否则达到最大迭代次数，结束计算；最后将PageRank值保存下来。主要PageRank计算代码如下：

```

1  import networkx as nx
2  import pymongo
3  class PageRankCalculator:
4      #...
5      def calculate_page_rank(self): # add nodes and edges to graph. page
rank init with -1. calculate page rank by iteration
6          web_page_and_forward_links =
self.web_page_collection.aggregate([{'$project': {'url_id': 1,
'forward_links': 1}}])
7          for web_page in web_page_and_forward_links:
8              home_page_id = web_page['url_id']
9              forward_link_ids = web_page['forward_links']
10             if home_page_id not in self.graph.nodes():
11                 self.graph.add_node(home_page_id, page_rank=0)
12             for forward_link_id in forward_link_ids:
13                 if forward_link_id not in self.graph.nodes():
14                     self.graph.add_node(forward_link_id, page_rank=0)
15                     self.graph.add_edge(home_page_id, forward_link_id)
16             graph_size = len(self.graph.nodes())
17             if not graph_size:
18                 print('graph size is 0. exit.')
19                 return
20             self.damping_value = (1 - self.damping_factor) / graph_size
21             for node in self.graph.nodes():
22                 self.graph.nodes[node]['page_rank'] = 1 / graph_size
23                 if self.graph.out_degree(node) == 0: # if node has no out
degree, add edges to all nodes.
24                     for another_node in self.graph.nodes():

```

```

25         self.graph.add_edge(node, another_node)
26     for i in range(self.max_iterations):
27         delta = self.iterate()
28         self.iteration_counter += 1
29         if delta < self.min_delta:
30             break
31     for node in self.graph.nodes():
32         self.web_page_collection.update_one({'url_id': node}, {'$set':
{'page_rank': self.graph.nodes[node]['page_rank']}})
33     print('page rank calculation finished. iteration: {}, delta:
{}'.format(self.iteration_counter, delta))
34
35     def iterate(self):
36         delta = 0
37         for node in self.graph.nodes():
38             rank = 0
39             for in_node in list(self.graph.in_edges(node)): # translate to
list. ref: https://stackoverflow.com/questions/47325667/object-is-not-
subscribable-networkx
40                 in_node = in_node[0]
41                 rank += self.graph.nodes[in_node]['page_rank'] /
self.graph.out_degree(in_node)
42             rank = self.damping_value + self.damping_factor * rank
43             delta = abs(self.graph.nodes[node]['page_rank'] - rank)
44             self.graph.nodes[node]['page_rank'] = rank
45         return delta

```

计算结果如下图所示，在第39轮时满足精度。

```

/home/bill/anaconda3/envs/mysearchengine/bin/python /home/bill/Desktop/information-retrieval-system/hw5/MySearchEngine/index/index.py
page rank calculation finished. iteration: 39, delta: 9.732673337324409e-07
Building prefix dict from the default dictionary ...
Loading model from cache /tmp/jieba.cache
Loading model cost 1.117 seconds.
Prefix dict has been built successfully.

```

参与结果排序

观察PageRank计算结果，www.nankai.edu.cn有着最高的PageRank得分，这个结果十分显然。但其指向的网页，也具有较高的得分，且与其他得分较低的网页，差距在几个数量级。


```

url: "https://www.nankai.edu.cn/"
redirected_from: 0
crawl_time: "2023-02-11 23:37:22"
page_rank: 0.03389971480007454
title: "南开大学"
type: "html"
pure_text: "最新动态,通知公告,学术活动,媒体南开,校情通报,干部选任,1月17日,教育部党组成员,副部长吴岩同志来校调研,1月17日,我校召开,学习二..."
anchor: Object
forward_links: Array
anchor_backward_text: "南开大学主页,南开大学,南开大学首页,NankaiUniversity,学校主页,国际学术交流处,NKU,NakaiUniversity,南..."
anchor_forward_text: "信息公开,图书馆,服务指南,登录邮箱,办公网,English,学校简介,现任领导,历届领导,历史回眸,南开新闻网,南开大学报,专业学院,职能..."
md5: "3a9099bfa7df867a7795001f4563c4a6"

_id: ObjectId('63e7b632f670065de0afcd3b')
url_id: 12
url: "http://jwc.nankai.edu.cn/"
redirected_from: 0
crawl_time: "2023-02-11 23:37:22"
page_rank: 0.000683217920790343
title: "教务处"
type: "html"
pure_text: "Copyright,2016南开大学教务部,地址,天津市津南区海河教育园同砚路38号邮编,300350"
anchor: Object
forward_links: Array
anchor_backward_text: "本科教育,南开大学教务处,教务处"
anchor_forward_text: "EnglisVersion,回到旧版,我校两教学团队获评市级教学团队三位教授获市教学名师奖,我校案例入选教育部,全国普通高校本科教育教学质量..."
md5: "0f7ce8b15a0881a284347475b2518310"

_id: ObjectId('63e7b632f670065de0afcd3e')
url_id: 21
url: "http://less.nankai.edu.cn/"
redirected_from: 0
crawl_time: "2023-02-11 23:37:22"
page_rank: 0.003646288835824725

```

在结果排序中，想要做到当相关度得分差距不大时，PageRank影响排序结果。考虑将BM25F得分与PageRank进行归一化后线性加权，PageRank权重为30%

```

1 def basic_search(self, query_str):
2     results = []
3     with self.ix.searcher(weighting=scoring.BM25F) as searcher:
4         query = self.parser.parse_query(query_str)
5         for result in searcher.search(query, limit=100):
6             results.append(self.WebPageItem(result['url'], result['title'],
7             result['pure_text'], result['anchor_backward_text'],
8             result['anchor_forward_text'], result['page_rank'], result['crawl_time'],
9             result.score))
10            max_bm25f_score = max([result.score for result in results])
11            max_page_rank_score = max([result.page_rank for result in results])
12            for result in results:
13                result.score = result.score / max_bm25f_score * 0.7 +
14                result.page_rank / max_page_rank_score * 0.3
15            results.sort(key=lambda x: x.score, reverse=True)
16            return results

```

如下为查询“Nankai”时，相似度得分差距不大，PageRank改变排序结果的例子。英文首页与材料学院首页相似度得分接近，而英文首页PageRank远高于材料学院，因此总得分较高


```
url: https://en.nankai.edu.cn/
filtered_title: Nankai University
pure_text: 38TongyanRoad,JinnanDistrict,Tianjin,P.R.China30035094WeijinRoad,NankaiDistrict
anchor_backward_text: English,NankaiUniversity,SchoolHome,南开大学英文主页
anchor_forward_text: WhoWeAre,History,Leadership,Administration,Facts,Figures,DegreeEducat
page_rank: 0.0012382932917778303
crawl_time: 2023-02-11 23:37:23
score: 0.6298648281890996

url: https://mse.nankai.edu.cn/htl\_en/list.htm
filtered_title: 南开大学材料科学与工程学院
pure_text: Tong,LiangHu,Professor,SecondaryUnits,Phone,022,85358282,Email,tlhu,nankai.edu.
anchor_backward_text: https://mse.nankai.edu.cn/htl\_en/list.htm
anchor_forward_text: 中文,Login,https,mse.nankai.edu.cn,htl,en,list.htm
page_rank: 6.619498952197716e-05
crawl_time: 2023-02-11 23:45:22
score: 0.620370844425527
```

查询功能

查询功能设计包括普通查询和高级查询。

普通查询默认在已经索引的标题、纯文本、后向链接文本、前向链接文本上进行检索，并可以使用 `AND`、`OR`、`NOT`、`()`（英文小括号），`*` 和 `~` 指定布尔查询、分组查询、通配查询和模糊查询，通过对查询字符串进行解析得到。

高级查询功能设计参考了百度的高级查询。对于包含全部关键词、任意关键词、完整关键词和过滤关键词的查询可以由普通查询的布尔查询进行得到，这里主要在普通查询的基础上增加了爬取时间过滤和网站地址过滤（即站内查询）。

布尔查询

布尔查询主要通过操作符 `AND`、`OR`、`NOT`、`()` 实现功能。

对于按照空格分隔的词项，如“南开大学 教务处”，将被解析为“南开大学 AND 教务处”（这里为了表达简便省略了对其中的词项进行进一步分词的步骤，分词间的关系为或），实际上我们可能希望查找“南开大学 OR 教务处”，同时包含更多关键词的结果应当具有更高的分数。

```
1 class MyQueryParser:
2     def __init__(self):
3         self.ix = open_dir('index/whoosh_index')
4         self.parser = MultifieldParser(['filtered_title', 'pure_text',
5                                         'anchor_backward_text', 'anchor_forward_text'], schema=self.ix.schema,
6                                         group=OrGroup.factory(0.9))
7
8     def parse_query(self, query_str):
9         return self.parser.parse(query_str)
```

完整的查询的解析示例结果如下：（由于返回结果长且繁杂，因此只截取关键部分进行说明）

- “南开大学 教务处”两个主要词项以OR连接，与“南开大学 OR 教务处”分词相同

```
anchor_forward_text:南开大学 OR filtered_title:教务 OR filtered_title:教务处 OR
```

- "南开大学 AND 教务处"

```
hor_forward_text:南开大学) AND (filtered_title:教务 OR filtered_title:教务处 OR p
```

- "南开大学 AND NOT (教务处 AND 通知)"

```
rd_text:大学 OR anchor_forward_text:南开大学) AND NOT ((filtered_title:教务 OR filtered_title:教务处 OR pure_text:教
```

通配查询

通配查询通过指定 `*` 运算符得到。需要注意的是，若 `*` 将一个完整词项“打断”，如“南*开”，则会认为“南”和“开”是两个词项，这不与词项“南开”匹配；同理，“本科*质量”可以与“本科教学质量匹配”。

```
user_query = QueryParser('filtered_title', schema=ix.schema).parse('本科*质量')
with ix.searcher() as searcher:
    results = searcher.search(user_query, limit=None)
    print(len(results))
    for result in results:
        print(result['filtered_title'])
```

```
2
本科教学质量报告
学习贯彻二十大,药学院举办第二届本科教学质量提升比赛,综合新闻,南开大学
```

```
1 user_query = QueryParser('filtered_title', schema=ix.schema).parse('"本科质量"')
2 with ix.searcher() as searcher:
3     results = searcher.search(user_query, limit=None)
4     print(len(results))
5     for result in results:
6         print(result['filtered_title'])
```

```
0
```

对于中文分词，通配查询的作用其实收效甚微。对于站内查询过程，由于url或者英文以不可分、不可索引的结构存储，通配查询便发挥用处，示例如下：

```
query_parser = QueryParser('url', schema=ix.schema)
query = query_parser.parse('*jwc.nankai.edu.cn*')
with ix.searcher() as searcher:
    results = searcher.search(query, limit=None)
    print(len(results))
    for result in results:
        print(result['url'])
```

```
3
http://jwc.nankai.edu.cn/
http://jwc.nankai.edu.cn/2022/0510/c20a449384/page.htm
http://jwc.nankai.edu.cn/a4/99/c5098a42137/page.psp
```

模糊查询

模糊查询通过操作符 `~ {number}` 来指定，其中`number`表示模糊的字数。示例如下：

```
query_parser = QueryParser('filtered_title', schema=ix.schema)
query_parser.add_plugin(FuzzyTermPlugin())
query = query_parser.parse('信心~1')
print(query)
with ix.searcher() as searcher:
    results = searcher.search(query, limit=None)
    print(len(results))
    for result in results:
        print(result['filtered_title'])
```

```
filtered_title:信心~
534
信息公开网
图书馆举办,知党史,悟初心,建新功,党史及业务知识竞赛,综合新闻,南开大学
百年风华,阅悟初心,第十一届南开读书节开幕,南开要闻,南开大学
招生计划,南开大学滨海学院招生信息网
信息公开网
南开大学接待服务中心
新闻中心,国际交流网
南开大学滨海学院招生信息网
医保信息
院长信箱
```

当查询为 `信心~2` 时，几乎会返回所有文档（即所有文档的分词都是匹配的）。模糊查询严重影响性能，因此默认并不开启。

高级查询解析

对于包含全部关键词、任意关键词、完整关键词和过滤关键词的查询可以由普通查询的布尔查询进行得到，这里主要在普通查询的基础上增加了爬取时间过滤和网站地址过滤（即站内查询）

```
1 def search(self, any_keywords, all_keywords='', complete_keywords='',
2   mask_keywords='', sort_by_time=False, filter_site=''):
3     query_str = '(' + any_keywords + ')'
4     if all_keywords != '':
5         temp_str = ' AND '.join([keyword for keyword in all_keywords.split('
6   ')])
7         query_str += ' AND (' + temp_str + ')'
8     if complete_keywords != '':
9         temp_str = ''
10        keywords = complete_keywords.split(' ')
11        for keyword in keywords:
12            temp_str += '"' + keyword + '"'
13            query_str += ' AND (' + temp_str + ')'
14        if mask_keywords != '':
15            temp_str = ' AND '.join([keyword for keyword in
16          mask_keywords.split(' ')])
17            query_str += ' AND NOT (' + temp_str + ')'
18        if filter_site == '':
19            limit = self.limit
20        else:
21            limit = None
22        results = self.basic_search(query_str, limit=limit)
23        if filter_site != '':
24            for result in results:
25                if filter_site not in result.url:
```

```

23         results.remove(result)
24     if sort_by_time:
25         results.sort(key=lambda x: x.crawl_time, reverse=True)
26     if len(results) > self.limit:
27         results = results[:self.limit]
28     return results

```

一个高级查询的例子如下：

```

if __name__ == '__main__':
    searcher = MySearcher()
    results = searcher.search(any_keywords='南开大学', all_keywords='学术 媒体', complete_keywords='教育部 召开', filter_site='http://www.nankai.edu.cn')
    for result in results:
        print(result)
        # print(results[0].score)
        # print(results[0].fields())

== '__main__' : for result in results
search
Loading model cost 0.467 seconds.
Prefix dict has been built successfully.
url: https://www.nankai.edu.cn/
filtered_title: 南开大学
pure_text: 最新动态, 通知公告, 学术活动, 媒体南开, 校情通报, 干部选任, 1月17日, 教育部党组成员, 副部长吴岩同志来校调研, 1月17日, 我校召开, 学习二十大精神牢记总书记嘱托, 南开大学师生座谈会, 12月18日, 天津市委书
anchor_backward_text: 南开大学主页, 南开大学, 南开大学首页, NankaiUniversity, 学校主页, 国际学术交流处, NKU, NankaiUniversity, 南开官网
anchor_forward_text: 信息公开, 图书馆, 服务指南, 登录邮箱, 办公网, English, 学校简介, 现任领导, 历届领导, 历史回眸, 南开新闻网, 南开大学报, 专业学院, 职能部门, 直属和后勤单位, 研究机构, 人事人才, 人才招聘, 博士
page_rank: 0.03389971480807454
crawl_time: 2023-02-11 23:37:22
score: 0.9999999999999998

url: http://news.nankai.edu.cn/zxbd/system/2012/03/21/000059815.shtml
filtered_title: 南开大学学术委员会章程-专题报道-南开大学
pure_text: 第一条为加强我校科学研究, 学科建设, 人才培养和教师队伍建设工作, 完善民主管理, 民主监督和科学决策体制, 促进学术发展, 根据, 中华人民共和国高等教育法, 第四十二条, 结合我校实际, 制定本章程, 第二条南开大
anchor_backward_text: 南开大学学术委员会章程
anchor_forward_text: 首页, 南开要闻, 媒体南开, 光影南开, 南开故事, 南开大学报, 视频, 广播, 南开大学, 专题报道, 微信往期推送, 更多..., 南开大学获批3项国际中文教育..., 中国自动驾驶汽车产业发展社..., 首届自动驾
page_rank: 4.089439807683026e-05
crawl_time: 2023-02-11 23:50:36
score: 0.6561011193652936

```

站内查询

站内查询功能已在高级查询功能部分介绍，对查询结果的url进行过滤即可。

对教务处网站的站内查询，返回结果只有两个，且符号查询要求：

```

if __name__ == '__main__':
    searcher = MySearcher()
    results = searcher.search(any_keywords='教务处', filter_site='jwc.nankai.edu.cn')
    for result in results:
        print(result)

her > search() > If filter_site != '' > for result in results > If filter_site in result.url
search x
Prefix dict has been built successfully.
url: http://jwc.nankai.edu.cn/
filtered_title: 教务处
pure_text: Copyright, 2016南开大学教务部, 地址, 天津市津南区海河教育园同砚路38号邮编, 300350
anchor_backward_text: 本科教育, 南开大学教务处, 教务处
anchor_forward_text: EnglisVersion, 回到旧版, 我校两教学团队获评市级教学团队三位教授获市教学名师奖, 我校案例入选教育部, 全国普通高校本科教育教学质量报告, 应用
page_rank: 0.000683217920790343
crawl_time: 2023-02-11 23:37:22
score: 0.7072881842913912

url: http://jwc.nankai.edu.cn/2022/0510/c20a449384/page.htm
filtered_title: 学业指导育人研讨会成功举办
pure_text: Copyright, 2016南开大学教务部, 地址, 天津市津南区海河教育园同砚路38号邮编, 300350, 学业指导育人研讨会成功举办
anchor_backward_text: 学业指导育人研讨会成功举办
anchor_forward_text:
page_rank: 5.5995629708423746e-05
crawl_time: 2023-02-11 23:37:26
score: 0.09597664526688521

```

查询日志

简单记录用户id与高级查询相关参数即可，以备后续再次查询。为了支持个性化推荐，这里记录了用户最近点击的20个网站id

```
1 class Recorder:
2     def __init__(self, user_id):
3         self.mongo_client = pymongo.MongoClient('localhost', 27017)
4         self.db = self.mongo_client['mysearchengine']
5         collection_names = self.db.list_collection_names()
6         if 'history' not in collection_names:
7             self.db.create_collection('history')
8         self.history_collection = self.db['history']
9         if 'user_id_1' not in self.history_collection.index_information():
10            self.history_collection.create_index([('user_id',
pymongo.ASCENDING)])
11        self.user_id = user_id
12        if self.history_collection.find_one({'user_id': self.user_id}) is
None:
13            self.history_collection.insert_one({'user_id': self.user_id,
'search': [], 'recent_click': []})
14
15        def record_search(self, query_str, sort_by_time, filter_site):
16            search_history = self.history_collection.find_one({'user_id':
self.user_id})['search']
17            search_history.append({'query_str': query_str, 'sort_by_time':
sort_by_time, 'filter_site': filter_site, 'time': datetime.now()})
18            if len(search_history) > 20:
19                search_history = search_history[-20:]
20            self.history_collection.update_one({'user_id': self.user_id},
{'$set': {'search': search_history}})
21
22        def record_click(self, url):
23            recent_click = self.history_collection.find_one({'user_id':
self.user_id})['recent_click']
24            recent_click_urls = [list(item.keys())[0] for item in recent_click]
25            if url in recent_click_urls:
26                click_times = recent_click[recent_click_urls.index(url)][url]
['click_times'] + 1
27                recent_click.remove(recent_click[recent_click_urls.index(url)])
28                recent_click.insert(0, {url: {'click_times': click_times,
'time': datetime.now()}})
29            else:
30                recent_click.insert(0, {url: {'click_times': 1, 'time':
datetime.now()}})
31            if len(recent_click) > 20:
32                recent_click = recent_click[:20]
33            self.history_collection.update_one({'user_id': self.user_id},
{'$set': {'recent_click': recent_click}})
```

查询日志的记录，以查询“计算机”为例子

```

    ▶ _id: ObjectId('63e8a91e1b99fa139840dd4d')
      user_id: 1
      ▼ search: Array
        ▼ 0: Object
          query_str: "(计算机)"
          sort_by_time: false
          filter_site: ""
          time: 2023-02-12T18:35:40.166+00:00
          ▶ recent_click: Array

```

测试连续三次访问南开首页，再访问教务处官网，访问记录次数和次序正确

```

    _id: ObjectId('63e89db677843a112d72549d')
    user_id: 1
    ▶ search: Array
    ▼ recent_click: Array
      ▼ 0: Object
        http://jwc.nankai.edu.cn/: Object
          click_times: 1
          time: 2023-02-12T16:07:38.733+00:00
      ▼ 1: Object
        https://www.nankai.edu.cn/: Object
          click_times: 3
          time: 2023-02-12T16:05:22.050+00:00

```

个性化查询

查询扩展技术主要基于对数据集本身的分析和对用户查询日志的分析¹，旨在提高召回率。一个基本思路是将网页进行聚类，得到不同主题，再根据用户查询的历史，推断近期偏好，从而提高对应主题文本在搜索结果中的排序。与个性化推荐有相似性，但不同在于该过程对查询结果进行进一步排序而非补充新的结果。

由于网页数量众多，实验的思路为：提取文本特征，进行文本聚类；分析靠前的查询结果对应的主题种类，更新用户偏好；根据用户偏好，对不同主题的网页重新评分，返回排序结果。






文本聚类

这里直接使用HanLP²提供的聚类方法，对文本进行聚类。

```

1 def cluster(self):
2     for page in self.web_page_collection.find():
3         self.analyzer.addDocument(page['url_id'], page['title'] +
page['pure_text'] + page['anchor_backward_text'] +
page['anchor_forward_text'])
4     results = self.analyzer.repeatedBisection(1.0)
5     print(len(results))
6     type_count = 1
7     for result in results:
8         cluster_type = type_count
9         type_count += 1
10        for url_id in result:
11            self.web_page_collection.update_one({'url_id': url_id}, {'$set':
{'cluster_type': cluster_type}})
```

使用自动判断聚类数目的方法，获得了350个类别。以下面为例，文档包括“要闻”、“专题报道”等关键词，确实具有很大相似性。

Filter  	{cluster_type: 234}
 ADD DATA 	 EXPORT COLLECTION
<pre>page_rank: 0.00003829271563790007 title: "【视频】津南夜色-专题报道-南开大学" type: "html" pure_text: "简介,夜幕徐徐降临,津南校区也逐渐由喧闹回归宁静,津南记者团的六人小分队分别出发,寻找津南校区不为人知的美丽,找寻图书馆的南开学子,找寻综合..." anchor: Object forward_links: Array anchor_backward_text: "1分钟视频,津南夜色" anchor_forward_text: "首页,南开要闻,媒体南开,光影南开,南开故事,南开大学报,视频,广播,南开大学,专题报道,微信往期推送,更多...,南开大学获批3项国际中文..." md5: "3873d642422d428d4aa9cffa21092bd9f" cluster_type: 234</pre>	
<pre>_id: ObjectId('63e7b93cf670065de0aff269') url_id: 5423 url: "http://news.nankai.edu.cn/ztbd/system/2015/12/18/000261323.shtml" redirected_from: 0 crawl_time: "2023-02-11 23:50:20" page_rank: 0.00003829271563790007 title: ""新南开 新诗篇"三行诗创作大赛参赛作品第十四组-专题报道-南开大学" type: "html" pure_text: "和你相遇,在津南,看最美的风景,于新校区登高,看晨昏落霞,方觉津城尚有如此胜景,碧落黄泉,百年求索南开人,一方新土筑腾骧,这里蒹葭未发,而出..." anchor: Object forward_links: Array anchor_backward_text: "三行诗,作品集,十四" anchor_forward_text: "首页,南开要闻,媒体南开,光影南开,南开故事,南开大学报,视频,广播,南开大学,专题报道,微信往期推送,更多...,南开大学获批3项国际中文..." md5: "519f618c2a0a5f88aba0fc70b266983a" cluster_type: 234</pre>	
<pre>_id: ObjectId('63e7b93cf670065de0aff26b') url_id: 5421 url: "http://news.nankai.edu.cn/ztbd/system/2015/12/18/000261319.shtml" redirected_from: 0 crawl_time: "2023-02-11 23:50:20" page_rank: 0.00003829271563790007 title: ""新南开 新诗篇"三行诗创作大赛参赛作品第十二组-专题报道-南开大学" type: "html" pure_text: "再生春,上寿之年福满堂,兴国之数楹四方,日新月异耀四方,斗转星移现津南,跨世相接共振华,孩子,你的出现,使忽冬褪去寒芒让急风不再凛冽,来吧,..."</pre>	

用户主题偏好

主题偏好计算为统计前20个查询结果中, 包含各个主题的数目作为偏好评分, 使用按查询次序衰减的方法计算偏好程度。主题偏好查询评分中, 占比20%。

```
1 def get_kind_preference(result_urls):
2     cluster_num =
3     len(self.web_page_collection.find().distinct('cluster_type'))
4     user_history = self.history_collection.find_one({'user_id':
5 self.recorder.user_id})
6     kind_preference = {}
7     if 'kind_preference' not in user_history:
8         for i in range(1, cluster_num + 1):
9             kind_preference[str(i)] = 1 / cluster_num    # int cannot be key
10 in mongodb
11 else:
12     kind_preference = user_history['kind_preference']
13     result_preference = {}
14     for result_url in result_urls:
15         kind = self.web_page_collection.find_one({'url': result_url})
16         ['cluster_type']
17         if str(kind) not in result_preference:
18             result_preference[str(kind)] = 0
19             result_preference[str(kind)] += 1
20         for kind in result_preference:
21             result_preference[str(kind)] /= len(result_urls)
22             kind_preference[str(kind)] = kind_preference[str(kind)] * 0.9 +
23 result_preference[str(kind)] * 0.1
24     self.history_collection.update_one({'user_id': self.recorder.user_id},
25 {'$set': {'kind_preference': kind_preference}})
26     return kind_preference
```


以下为重复查询“计算机”后，其中文档的评分变化。个性化查询在基于数据主题划分和用户近期查询的基础上，能够起一定的贴近用户主题偏好的作用。

```
url: https://cc.nankai.edu.cn/13280/list.htm
filtered_title: 计算机科学与技术系
pure_text: 南开大学计算机科学与技术系,以下简称计算机系,隶属于南开大学计算机学院,下设计算机科学与技术
anchor_backward_text: 组织结构
anchor_forward_text: 首页,学院概况,学院概况,学院领导,学院党委,组织结构,学科学术分委会,学位评定分
page_rank: 0.000773042536633646
crawl_time: 2023-02-11 23:39:59
score: 0.8085465714285714 由0.80616提高到0.80854
```

个性化推荐——基于物品的协同过滤³

一个容易理解的个性化推荐思路是，若同时有很多用户都同时对主题A和B感兴趣，则认为主题A和B是相近的，应当在搜索其中一个主题时，得到另一个主题的推荐。这个思路正是基于物品的协同过滤算法的基本思路。以下使用随机初始化点击历史的方法，将ItemCF算法应用于搜索引擎的推荐内容中。

网页相似度、用户兴趣计算与建立索引

- 网页相似度

基于用户行为计算网页相似度的思想，为点击该网页的用户重合度越高，两个网页越相似。设点击过网页A的用户集合记作 W_A ，点击过网页B的用户集合记作 W_B ，定义交集 $V = W_A \cap W_B$ 则两个网页的相似度有以下公式：

$$\text{sim}(A, B) = |V| / \sqrt{|W_A| * |W_B|}$$

分子表示了同时点击网页的用户数，分母则将其归一化到0和1之间。该公式没有具体衡量用户对网页的具体偏好程度，因此有以下公式：

$$\text{sim}(A, B) = \sum_{v \in V} \text{like}(v, A) * \text{like}(v, B) / \left(\sqrt{\sum_{u_1 \in W_1} \text{like}^2(u_1, A)} * \sqrt{\sum_{u_2 \in W_2} \text{like}^2(u_2, B)} \right)$$

该公式即余弦相似度的计算（分子为向量的点积，分母为模长）

这里以随机数模拟点击，计算结果如下：

```
{_id: ObjectId('63e8a91e1b99fa139840dd4e')
  url: "http://jwc.nankai.edu.cn/"
  users: Array
  norm: 1
  similarity: Object
    http://en.medical.nankai.edu.cn/Collegeoverview/list.htm: 0
    http://en.sky.nankai.edu.cn/NewsEvents/list.htm: 0
    http://energy.nankai.edu.cn/article/106: 0.3333333333333333
    http://energy.nankai.edu.cn/article/302: 0
    http://energy.nankai.edu.cn/article/410: 0
    http://energy.nankai.edu.cn/article/440: 0
    http://energy.nankai.edu.cn/article/490: 0
    http://energy.nankai.edu.cn/article/531: 0
    http://energy.nankai.edu.cn/article/609: 0
    http://energy.nankai.edu.cn/article/725: 0
    http://energy.nankai.edu.cn/article/759: 0
    http://energy.nankai.edu.cn/english/class/2: 0
    http://less.nankai.edu.cn/equipment/content/8: 0.4472135954999579
    http://less.nankai.edu.cn/equipment?tag=%E7%83%AD%E5%88%86%E6%9E%90%E4%BB%AA%E5%99%A8: 0
    http://less.nankai.edu.cn/lms/recovery: 0
    http://ndst.nankai.edu.cn/23636/list.htm: 0
    http://news.nankai.edu.cn/dcx/system/2010/05/20/000030788.shtml: 0.4472135954999579
    http://news.nankai.edu.cn/dcx/system/2021/05/30/030046368.shtml: 0
    http://news.nankai.edu.cn/dcx/system/2022/05/16/030051304.shtml: 0
    http://news.nankai.edu.cn/dcx/system/2022/12/09/030053981.shtml: 0
```

- 用户兴趣

用户u对物品i的兴趣计算有如下公式：

$$pr(u, i) = \sum_j like(u, j) * sim(j, i)$$

可以理解为将兴趣从物品j传递到物品i的过程。用户得分的具体计算在生成推介列表部分介绍。

- 离线建立索引
 - 用户到物品的索引：记录用户最近交互过的网页id
 - 物品到物品的索引：记录物品与其他物品的相似度

计算过程：首先计算各自物品喜好模长，再计算网页余弦相似度（与向量空间模型思路相似）。用户到物品的索引已在点击时建立，只需建立索引过程只用建立物品到物品的索引即可

```

1  def cal_norm(self):
2      urls = self.page_click_collection.find().distinct('url')
3      for url in urls:
4          users = self.page_click_collection.find_one({'url': url})['users']
5          click_times = []
6          for user in users:
7              recent_click = self.history_collection.find_one({'user_id':
8 user})['recent_click']
9              recent_click_urls = [list(item.keys())[0] for item in
10 recent_click]
11              if url in recent_click_urls:
12
13                  click_times.append(recent_click[recent_click_urls.index(url)][url]
14 ['click_times'])
15                  vector_norm = np.linalg.norm(click_times, ord=2)
16                  self.page_click_collection.update_one({'url': url}, {'$set':
17 {'norm': vector_norm}})
18
19 def build_index(self):
20     def similarity(page1, page2):
21         user_list1 = self.page_click_collection.find_one({'url': page1})
22 ['users']
23         user_list2 = self.page_click_collection.find_one({'url': page2})
24 ['users']
25         user_list = list(set(user_list1) & set(user_list2))
26         # find click times of page1 and page2 for each user
27         click_times1 = []
28         click_times2 = []
29         for user in user_list:
30             recent_click = self.history_collection.find_one({'user_id':
31 user})['recent_click']
32             recent_click_urls = [list(item.keys())[0] for item in
33 recent_click]
34             if page1 in recent_click_urls:
35
36                 click_times1.append(recent_click[recent_click_urls.index(page1)][page1]
37 ['click_times'])
38             if page2 in recent_click_urls:
39
40                 click_times2.append(recent_click[recent_click_urls.index(page2)][page2]
41 ['click_times'])
42             dot_product = np.dot(click_times1, click_times2)

```

```

30         norm1 = self.page_click_collection.find_one({'url': page1})['norm']
31         norm2 = self.page_click_collection.find_one({'url': page2})['norm']
32         return dot_product / (norm1 * norm2)
33     urls = self.page_click_collection.find().distinct('url')
34     graph = Graph()
35     graph.add_nodes_from(urls)
36     for url in urls:
37         for url2 in urls:
38             if url != url2:
39                 if graph.has_edge(url, url2):
40                     continue
41                 graph.add_edge(url, url2, weight=similarity(url, url2))
42     for url in urls: # remove weight
43         self.page_click_collection.update_one({'url': url}, {'$set':
{'similarity': {}}})
44     for edge in graph.edges:
45         url1_similarity = self.page_click_collection.find_one({'url':
edge[0]})['similarity']
46         url2_similarity = self.page_click_collection.find_one({'url':
edge[1]})['similarity']
47         url1_similarity[edge[1]] = graph.get_edge_data(edge[0], edge[1])
['weight']
48         url2_similarity[edge[0]] = graph.get_edge_data(edge[0], edge[1])
['weight']
49         self.page_click_collection.update_one({'url': edge[0]}, {'$set':
{'similarity': url1_similarity}})
50         self.page_click_collection.update_one({'url': edge[1]}, {'$set':
{'similarity': url2_similarity}})

```

生成推荐列表

生成推荐列表的过程：

1. 找到用户最近感兴趣的n个网页
2. 从n个网页的相似列表中，分别找到最相似的k个网页，这样返回n*k个网页
3. 计算用户兴趣分数，若重复则剔除，累加得分，根据结果排序并返回

```

1  def recommend(self, limit=20, each_similarity_num=5):
2      recent_click = self.history_collection.find_one({'user_id':
self.user_id})['recent_click']
3      recent_click_urls = [list(item.keys())[0] for item in recent_click]
4      similar_pages = {}
5      for url in recent_click_urls:
6          # sort by similarity
7          similarity = self.page_click_collection.find_one({'url': url})
['similarity']
8          similarity = sorted(similarity.items(), key=lambda x: x[1],
reverse=True)
9          similarity = similarity[:each_similarity_num]
10         for page in similarity:
11             if page[0] in similar_pages:
12                 similar_pages[page[0]] += page[1]
13             else:
14                 similar_pages[page[0]] = page[1]
15         similar_pages = sorted(similar_pages.items(), key=lambda x: x[1],
reverse=True)
16         similar_pages = similar_pages[:limit]

```

对于用户1，一个推荐列表的返回结果如下，结果已经按得分进行排序

```
recommender = MyRecommender(user_id=1)
recommender.recommend()

[('http://jwc.nankai.edu.cn/', 7.323647594543401),
 ('http://news.nankai.edu.cn/mtnk/system/2016/11/17/000305979.shtml?wd=%CE%E2%D6%BE%B3%C9+%D0%C2%CE%C5%C1%AA%B2%A5&x=13&y=10', 7.323647594543401),
 ('http://news.nankai.edu.cn/ztbd/system/2021/06/27/030047036.shtml', 6.723053019297644),
 ('https://yzb.nankai.edu.cn/2022/0623/c5508a459438/page.htm', 5.721139227572456),
 ('http://news.nankai.edu.cn/ywsd/system/2022/02/23/030050367.shtml', 2.887581946738247),
 ('http://less.nankai.edu.cn/equipment/content/8', 1.7888543819998317),
 ('http://lib.nankai.edu.cn/whhd/list.htm', 0.051051507875107875)]
```

Web界面

其他问题

爬取过程中状态为301、302的处理

使用scrapy框架爬取过程中，对于301、302重定向问题，大多是由于需要由https协议访问，而原url指定的是http，这样其实是爬取不到该网页的。该问题对于PageRank计算和索引域构建都有影响，例如很多网页的前向链接为 `http://www.nankai.edu.cn/` 而其实会重定向到

`https://www.nankai.edu.cn/`，因此会导致出现两个相同网页的PageRank计算，前者远小于后者，且后向链接文本无法准确找到对应网页，丢失了目标网页的信息，因此该问题不能忽视。

```
1 2023-02-06 12:40:33 [scrapy.downloadermiddlewares.redirect] DEBUG:
    Redirecting (302) to <GET https://lib.nankai.edu.cn/whhd/list.htm> from <GET
    http://lib.nankai.edu.cn/whhd/list.htm>
2 2023-02-06 12:40:33 [scrapy.downloadermiddlewares.redirect] DEBUG:
    Redirecting (302) to <GET https://lib.nankai.edu.cn/15326/list.htm> from <GET
    http://lib.nankai.edu.cn/15326/list.htm>
```

解决思路：

1. 允许重定向⁴。在settings.py中添加允许http错误代码列表，包含301和302；在解析response对象时，获取其request成员中的redirect_urls列表（取第一个，因为只有一个源网页）

```

1 # settings.py
2 HTTPERROR_ALLOWED_CODES = [301, 302]
3
4 # nankai.py (for test)
5 def parse(self, response):
6     redirected_urls = response.request.meta.get('redirect_urls')
7     if redirected_urls is not None:
8         with open('redirected_urls.txt', 'a') as f:
9             for url in redirected_urls:
10                 f.write('current url: ' + response.url + ' redirected url: '
+ url + '\n')

```

测试结果如下，观察到大多数重定向原因正如上所述。

```

1 ...
2 current url: https://www.nankai.edu.cn/2019/1009/c17551a208969/page.htm
  redirected url: http://www.nankai.edu.cn/2019/1009/c17551a208969/page.htm
3 current url: https://www.nankai.edu.cn/VPN/list.htm redirected url:
  http://www.nankai.edu.cn/VPN/list.htm
4 current url: https://graduate.nankai.edu.cn/ redirected url:
  http://graduate.nankai.edu.cn
5 ...

```

2. 爬取结束后，修改前向链接为重定向后网址。这里借助了在计算PageRank时使用的有向图类，最终在前向链接集合中，剔除被重定向网页集合，添加目的网页集合。

```

1 def correct_redirect(self): # essential correction for page rank algorithm
2     redirect_edges = self.redirect_collection.find()
3     for edge in redirect_edges:
4         redirect_forward_links_list =
self.web_page_collection.find({'forward_links': {'$in':
[edge['from_url_id']]})
5         for forward_links in redirect_forward_links_list:
6             new_forward_links = (set(forward_links['forward_links']) -
{edge['from_url_id']}) | {edge['to_url_id']}
7             self.web_page_collection.update_one({'url_id':
forward_links['url_id']}, {'$set':
{'forward_links': list(new_forward_links)}})

```

南开首页重复爬取问题

虽然在爬取时，通过数据库查询防止重复爬取，但还是会存在首页重复爬取问题。暂未想到合适解决办法，手动删除即可

总结

本次实验以搜索引擎的构建为主线，实践了课堂上所讲的很多关键内容，对传统和现代信息检索技术有了更加深刻的了解。使用各种开源工具的过程也拓宽了工程和编程能力，收获良多。不足的是，如结果评分、个性化查询和推荐过程中的一些设计，可能是“拍脑袋”想出来的，没有经过严谨的论证和，也没有大量实验数据的支撑，无法客观验证设计的有效性和合理性。

1. 支凤麟,徐炜民. 基于主题的个性化查询扩展模型[J]. 计算机工程与设计,2010,31(20):4471-4475. [↗](#)

2. (HanLP实现文本聚类)[http://mantchs.com/2020/02/13/Introduction-NLP/text_clustering/] [↗](#)

3. 召回01：基于物品的协同过滤（ItemCF）by Shusen Wang [↗](#)

