南开资源系统实现

程序使用流程

1 资源抓取

使用`scrapy`框架

运行`nkuspider`目录下的`run.py`，开始从`news.nankai.edu.cn`爬取资源

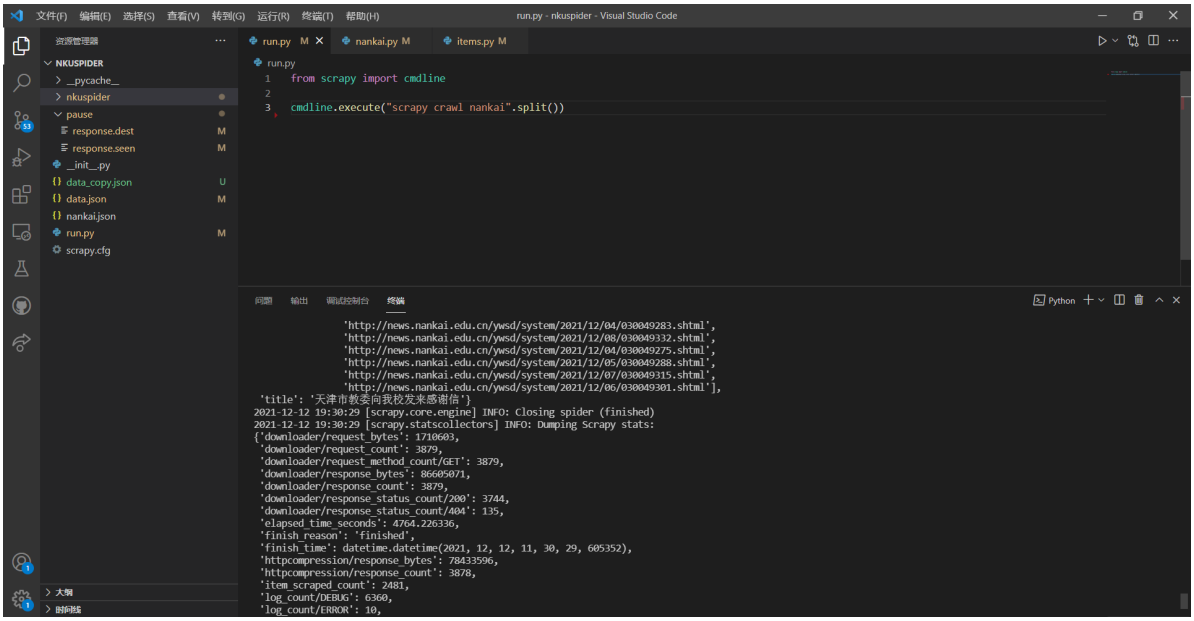
核心代码如下:

```
def parse(self, response):
        # 断点续爬功能之保存断点
        self.counter_plus()
        urls = response.xpath('//a/@href').extract()
        for url in urls:
            if self.isOkUrl(url):
                ss = '//a[@href="'+url+'"]/text()'
                if len(response.xpath(ss).extract()) != 0:
                    sst = response.xpath(ss).extract()[0].strip()
                    self.anchor_dict[url] = sst
                if self.isOkUrl(response.url):
                    if response.url in self.out_dict:
                        self.out_dict[response.url].append(url)
                    else:
                        self.out_dict[response.url] = [url]


        # 爬取当前网页
        print('start parse : ' + response.url)
        self.destination_list.remove(response.url)
        if self.isOkUrl(response.url):
            item = NkuspiderItem()
            for box in response.xpath('//table[@style="padding:0 0 0 10px;
"]/tbody'):
                # article title
                item['title'] = box.xpath('.//tr[1]/td/text()').extract()
[0].strip()

                # article url
                item['newsUrl'] = response.url
                item['newsUrlMd5'] = self.md5(response.url)
                item['newsFrom'] =
box.xpath('//tr[2]/td/span[1]/text()').extract()[0].strip()

                # article publish time
                item['newsPublishTime'] =
box.xpath('//tr[2]/td/span[2]/text()').extract()[0].strip()

                # article content
```

```
                item['newsContent'] = box.xpath('./tr[3]/td').extract()
[0].strip()

                regexp = re.compile(r'<[^>]+>', re.S)
                item['newsContent'] = regexp.sub('', item['newsContent'])  #
delete templates <>

                # 索引构建flag
                item['indexed'] = 'False'
                if response.url in self.anchor_dict:
                    item['anchor_text'] = self.anchor_dict[response.url]
                if response.url in self.out_dict:
                    item['out_degree'] = self.out_dict[response.url]

                yield item

        # 获取当前网页所有url并宽度爬取
        if(response.xpath('//a[@class="next"]/@href')):
            next_page = response.xpath('//a[@class="next"]/@href').extract()[0]
            next_page = response.urljoin(next_page)
            yield scrapy.Request(next_page, callback=self.parse,
errback=self.errback_httpbin)




        for url in urls:
            real_url = urljoin(response.url, url)  # 将.//等简化url转化为真正的http
格式url

            if not real_url.startswith("http://news.nankai.edu.cn"):
                continue
            if real_url.endswith('.jpg') or real_url.endswith('.pdf'):
                continue  # 图片资源不爬
            if '.jsp?' in real_url:
                continue

            md5_url = self.md5(real_url)
            if self.binary_md5_url_search(md5_url) > -1:  # 存在当前MD5
                pass
            else:
                self.binary_md5_url_insert(md5_url)


                self.destination_list.append(real_url)

                yield scrapy.Request(real_url, callback=self.parse,
errback=self.errback_httpbin)
```

这是对爬取到的一个网页的处理。

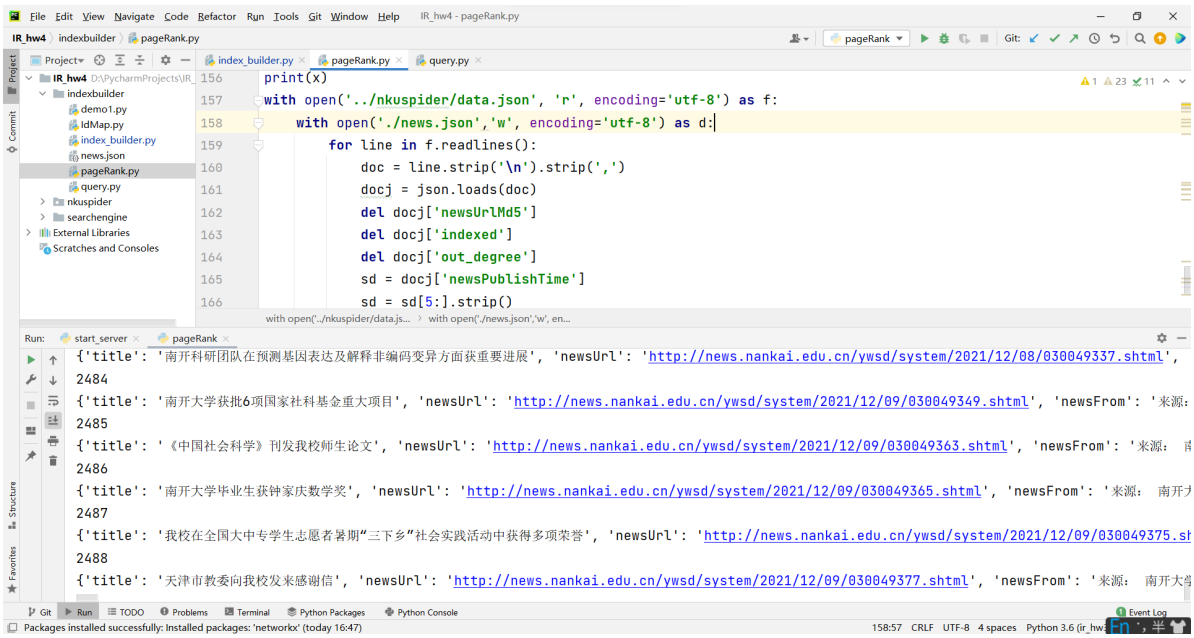
运行截图如下：



2 索引构建与链接分析

首先需要遍历两次`data.json`，第一次遍历获得了所有网页的链入链出关系，并计算每一个网页的`pagerank`值。

运行`indexbuilder`下的`pagerank.py`，运行截图如下：

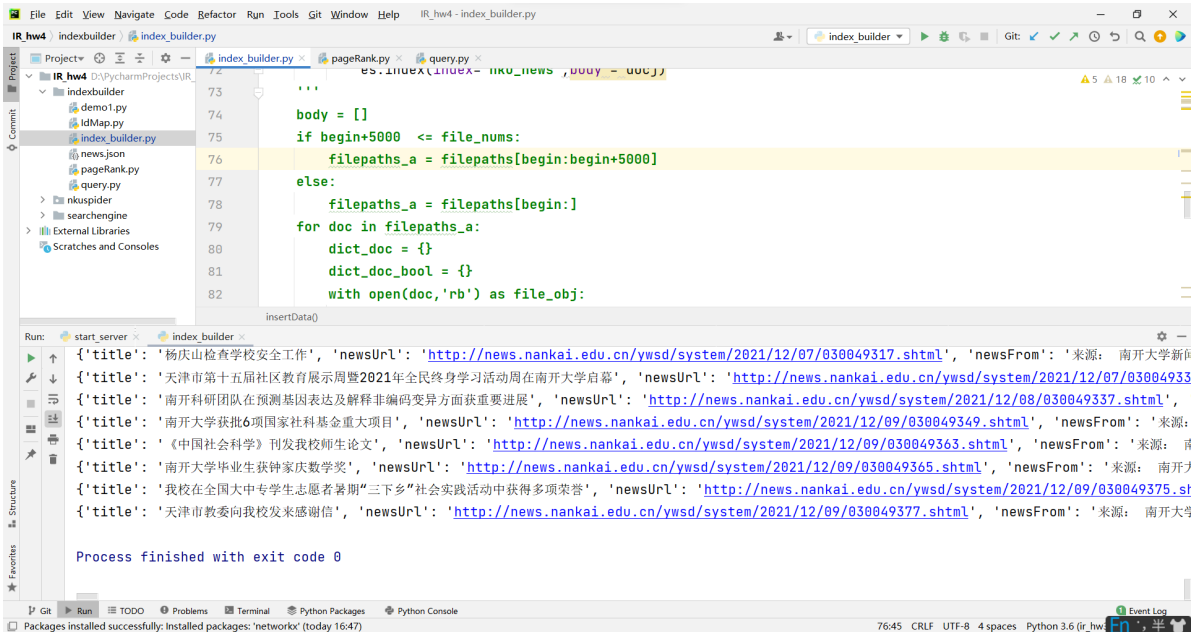


接下来参考hw_3，运行`indexbuilder`目录下的`index_builedr.py`使用ES进行索引构建：

```
def createIndex():
    ic = client.IndicesClient(es)
    if ic.exists(index="nku_news"):
        es.indices.delete(index="nku_news")
        print("删除之前存在的index")
    if not ic.exists(index="nku_news"):
        settings = {
            "mappings": {
                "properties": {
```

```
                    "title": {
                        "type": "text",
                        "analyzer": "ik_max_word",
                        "search_analyzer": "ik_max_word"
                    },
                    "newsurl": {
                        "type": "keyword"
                    },
                    "newsFrom": {
                        "type": "keyword"
                    },
                    "newsPublishTime": {
                        "type": "date"
                    },
                    "newsContent": {
                        "type": "text",
                        "analyzer": "ik_max_word",
                        "search_analyzer": "ik_max_word"
                    },
                    "anchor_text" : {
                        "type" : "text",
                        "analyzer": "ik_max_word",
                        "search_analyzer": "ik_max_word"
                    },

                    "pagerank": {
                        "type": "double"
                    }
                }
            }
        }


        ic.create(index="nku_news",ignore=400,body=settings)
        print("创建index成功！")
def insertData():
    with open('./news.json', 'r', encoding='utf-8') as f:
        for line in f.readlines():
            doc = line.strip('\n').strip(',')
            docj = json.loads(doc)
            stime = docj['newsPublishTime']
            del docj['newsPublishTime']
            docj['newsPublishTime'] = str2date(stime)
            print(docj)
            es.index(index='nku_news',body = docj)
```

注意：标题，正文和锚文本都指定了中文分词器IK的使用。



3 查询服务

只需定义几个基本的查询函数即可

```
def matchPhraseContent(key_context):#短语查询
    res = es.search(index='nku_news',body={"query": {"match_phrase":
{"newsContent": key_context}}, "size": 1000})
    tt = res['hits']['hits']
    tt.sort(key=lambda a:a['_score']+10*a['_source']['pagerank'],reverse = True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num,1000)):

        list.append(tt[i]['_source'])

    return list
def anchor(key_context):#锚文本查询
    res = es.search(index='nku_news',body={"query": {"match": {"anchor_text":
key_context}}, "size": 1000})
    tt = res['hits']['hits']
    tt.sort(key=lambda a:a['_score']+10*a['_source']['pagerank'],reverse = True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num,1000)):

        list.append(tt[i]['_source'])

    return list
def default(key_context):#默认查询方式
    res = es.search(index='nku_news',body={"query": {"match": {"newsContent":
key_context}}, "size": 1000})
    tt = res['hits']['hits']
    tt.sort(key=lambda a:a['_score']+10*a['_source']['pagerank'],reverse = True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num,1000)):
```

```
        list.append(tt[i]['_source'])

    return list


def wildcardContent(key_context):#内容通配查询
    res = es.search(index='nku_news',body={"query": {"wildcard": {"newsContent":
key_context}}, "size": 1000})
    tt = res['hits']['hits']
    tt.sort(key=lambda a:a['_score']+10*a['_source']['pagerank'],reverse = True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num,1000)):

        list.append(tt[i]['_source'])

    return list

def searchUrl(key_context):#url查询
    res = es.search(index='nku_news',body={"query": {"match_phrase": {"newsUrl":
key_context}}, "size": 1000})
    tt = res['hits']['hits']
    tt.sort(key=lambda a:a['_score']+10*a['_source']['pagerank'],reverse = True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num,1000)):

        list.append(tt[i]['_source'])

    return list
def termTitle(key_context):#标题精准查询
    res = es.search(index='nku_news',body={"query": {"term": {"title":
key_context}}, "size": 1000})
    tt = res['hits']['hits']
    tt.sort(key=lambda a:a['_score']+10*a['_source']['pagerank'],reverse = True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num,1000)):

        list.append(tt[i]['_source'])

    return list
def searchByDate(begin_str,end_str):#通过时间查询
    begin = query2date(begin_str)
    end = query2date(end_str)
    if begin > end :
        print("开始时间不得小于结束时间")
        exit()
    condition = {
        'query' : {
            'range' : {
                'newsPublishTime' : {
                    'gte' : begin,
                    'lte' : end
                }
            }
        },
        'size' : 1000
```

```
    }
    res = es.search(index='nku_news',body=condition)
    tt = res['hits']['hits']
    tt.sort(key=lambda a: a['_score'] * a['_source']['pagerank'], reverse=True)
    num = res['hits']['total']['value']
    list = []
    for i in range(min(num, 1000)):

        list.append(tt[i]['_source'])

    return list
```

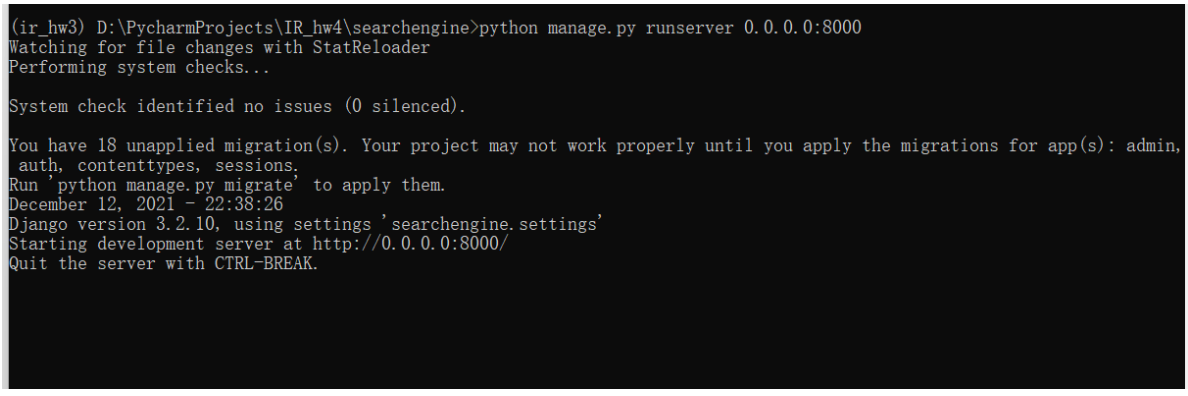
4 图形化界面

关键代码如下：

```
def merge_list(list1,list2):
    res = []
    for i in list1:
        if i in list2:
            res.append(i)

    return res


def handle(qt, qc):
    if qt=='phrase':
        return matchPhraseContent(qc)
    elif qt =='wilecard':
        return wildcardContent(qc)
    elif qt=='term':
        return termTitle(qc)
    elif qt=='time':
        queryt = qc.split('/')
        print(queryt)
        return searchByDate(queryt[0].strip(),queryt[1].strip())
    elif qt=='anchor':
        return anchor(qc)
    elif qt=='url':
        return searchUrl(qc)
    else:
        return default(qc)


class Query:
    def __init__(self):
        es = Elasticsearch()

    def standard_search(self, query):
        li = query.split('|')

        ans_list = []
        for s in li:
            p1 = s.find(':')
            qt = s[:p1].strip()
```

```
            qc = s[p1 + 1:].strip()
            ans_list.append(handle(qt, qc))

        if len(ans_list) == 1:
            return ans_list[0]
        else:
            temp = ans_list[0]
            for i in range(1, len(ans_list)):
                temp = merge_list(temp, ans_list[i])
            return temp



    def __exit__(self, exc_type, exc_val, exc_tb):
        print('Query close.')




def search_form(request):
    return render(request, 'main.html')

def search(request):
    res = None
    q = Query()
    if 'q' in request.GET and request.GET['q']:
        res = q.standard_search(request.GET['q'])
        c = {
            'query': request.GET['q'],
            'resAmount': len(res),
            'results': res,
        }
    else:
        return render(request, 'main.html')

    return render(request, 'result.html', c)
```

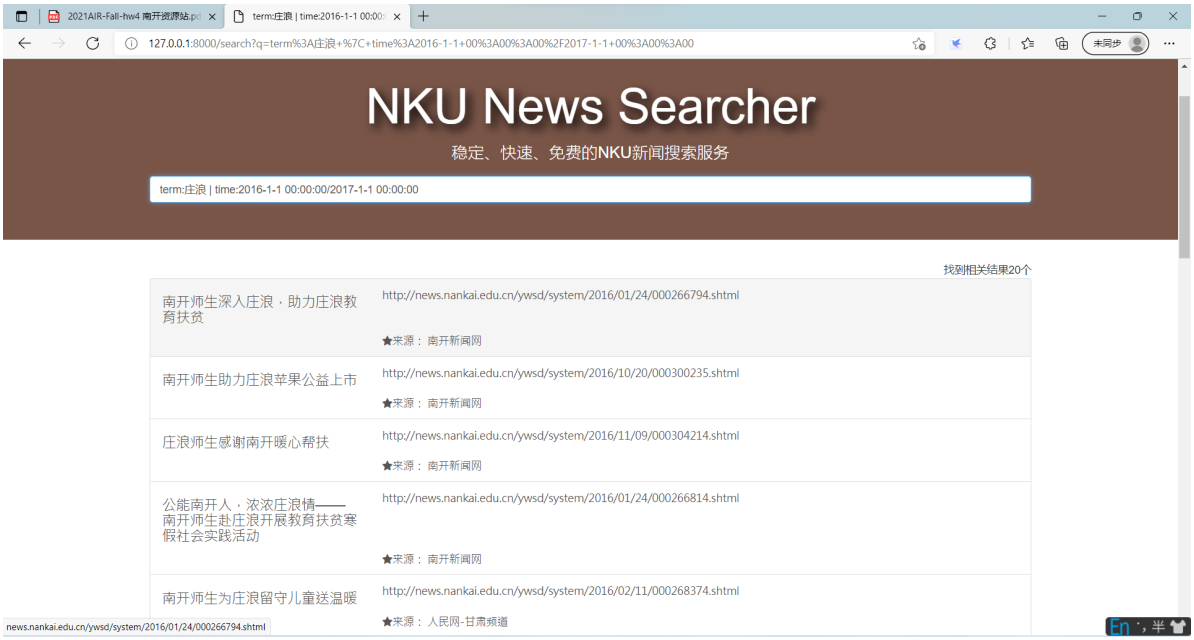
打开cmd窗口，进入`searchengine`目录，执行`python manage.py runserver 0.0.0.0:8000`命令



打开浏览器，在`url`栏输入地址`http://127.0.0.1:8000/`

在文本框内输入内容，`Enter`键检索：

高级检索方式检索，通过标题精准查询搜索2016年的标题含有庄浪二字的新闻：


2021AIR-Fall-hw4 南开资源站.pd
term:庄浪 | time:2016-1-1 00:00:
127.0.0.1:8000/search?q=term%3A庄浪+%7C+time%3A2016-1-1+00%3A00%3A00%2F2017-1-1+00%3A00%3A00
未同步
NKU News Searcher
稳定、快速、免费的NKU新闻搜索服务
term:庄浪 | time:2016-1-1 00:00:00/2017-1-1 00:00:00
找到相关结果20个
南开师生深入庄浪，助力庄浪教育扶贫
http://news.nankai.edu.cn/ywsd/system/2016/01/24/000266794.shtml
★来源： 南开新闻网
南开师生助力庄浪苹果公益上市
http://news.nankai.edu.cn/ywsd/system/2016/10/20/000300235.shtml
★来源： 南开新闻网
庄浪师生感谢南开暖心帮扶
http://news.nankai.edu.cn/ywsd/system/2016/11/09/000304214.shtml
★来源： 南开新闻网
公能南开人，浓浓庄浪情——南开师生赴庄浪开展教育扶贫寒假社会实践活动
http://news.nankai.edu.cn/ywsd/system/2016/01/24/000266814.shtml
★来源： 南开新闻网
南开师生为庄浪留守儿童送温暖
http://news.nankai.edu.cn/ywsd/system/2016/02/11/000268374.shtml
★来源： 人民网-甘肃频道
news.nankai.edu.cn/ywsd/system/2016/01/24/000266794.shtml