系统说明和设计文档

运行方式

- 运行`sduspider/run.py`来进行网络爬虫
- 运行`indexbuilder/index_builder.py`来对数据库中的数据构建索引
- 运行`indexbuilder/query.py`来测试搜索功能。
- 运行`searchengine/run_server.py`打开搜索网页服务器，在浏览器中打开**127.0.0.1:8000**进入搜索页面执行搜索。

所需python库

- scrapy
- requests
- pymongo
- whoosh
- jieba
- django

所需数据库

- MongoDB
- Studio 3T 导出csv文件

爬虫特性

爬取百度贴吧`nba吧`的内容。

爬虫代码位于`Tieba/`目录下。

由于待爬取的网页结构较简单, 只需要修改`pn=`后面的数字即可爬取。

通过查看网页结构，发现每一页都有50个待爬数据，每一页的`pn`都是50的倍数， 因此设置起始站点的代码为

```
start_urls = [f'https://tieba.baidu.com/f?kw=nba&ie=utf-8&pn={page * 50}' for page in range(0, 2000)]
```

再利用待爬取的数据的`xpath`爬取数据。

爬取的字段

爬取的字段
title
introduction
author
reply
last_reply_time
url

```
# 爬取当前网页
        print('start parse : ' + response.url)
        print("开始了开始了")

        selectors = response.xpath('//*[@id="thread_list"]/li')
        print(selectors)
        if response.url.startswith("https://tieba.baidu.com/"):
            item = items.TiebaItem()
            for selector in selectors[2:]:
                url = selector.xpath(
                    './/div[@class="threadlist_title pull_left j_th_tit
"]/a/@href').get()
                if url:  # 会员情况与非会员的xpath不一样，判断一下非会员的是否读成功，失败的话
就表示是会员的，要重新读一遍
                    url = "https://tieba.baidu.com" + url
                else:
                    url = selector.xpath(
                        './/div[@class="threadlist_title pull_left j_th_tit
member_thread_title_frs "]/a/@href').get()
                    url = "https://tieba.baidu.com" + url
                md5url = md5(url)
                if self.binary_md5_url_search(md5url) > -1:  # 存在当前MD5
                    print("有重复!!!!!!!!!!!!!!!")
                    pass
                else:
                    title = selector.xpath(
                        './/div[@class="threadlist_title pull_left j_th_tit
member_thread_title_frs "]/a/text()').get()
                    if not title:
                        title = selector.xpath('.//div[@class="threadlist_title
pull_left j_th_tit "]/a/text()').get()
                    introduction = selector.xpath(
                        './/div[@class="threadlist_abs threadlist_abs_onlyline
"]/text()').get()
                    introduction = introduction.strip()
                    author = selector.xpath(
                        './/span[@class="tb_icon_author
"]//a[@rel="noreferrer"]/text()').get()
                    reply = selector.xpath(
                        './/span[@class ="threadlist_rep_num
center_text"]/text()').get()
                    last_reply_time = selector.xpath(
                        './/span[@class ="threadlist_reply_date pull_right
j_reply_data"]/text()').get()
                    last_reply_time = last_reply_time.strip()
                    item['title'] = title
                    item['introduction'] = introduction
                    item['author'] = author
                    item['reply'] = reply
                    item['last_reply_time'] = last_reply_time
                    item['url'] = url
                    item['urlmd5'] = md5(url)
                    # 索引构建flag
```

```
                    item['indexed'] = 'False'
                    self.binary_md5_url_insert(md5url)
                    self.destination_list.append(url)
                    print('已爬取网址数: ' + (str)(len(self.destination_list)))
                    # yield it
                    yield item

                # print("title: " + title, count)
                # print("introduction: " + introduction)
                # print("author: ", author)
                # print("reply number: " + reply)
                # print("last reply time = " + last_reply_time)
                # print("url = ", url)
                # print(" \n")

        print("结束了")
```

但是在爬取的过程中发现有很多重复的数据，仔细检查之后发现是百度贴吧本身有很多重复的数据，在200页之后的每一页几乎都是相同的，从而导致了大量数据重复。

因此加了去重的代码。

将`url`加密为md5

```
def md5(val):
    import hashlib
    ha = hashlib.md5()
    ha.update(bytes(val, encoding='utf-8'))
    key = ha.hexdigest()
    return key
```

二分法`md5`集合排序插入`self.url_md5_set--16`进制`md5`字符串集

```
def binary_md5_url_insert(self, md5_item):
    low = 0
    high = len(self.url_md5_seen)
    while (low < high):
        mid = (int)(low + (high - low) / 2)
        if self.url_md5_seen[mid] < md5_item:
            low = mid + 1
        elif self.url_md5_seen[mid] >= md5_item:
            high = mid
    self.url_md5_seen.insert(low, md5_item)
```

二分法查找`url_md5`存在于`self.url_md5_set`的位置，不存在返回`-1`

```
def binary_md5_url_search(self, md5_item):
    low = 0
    high = len(self.url_md5_seen)
    if high == 0:
        return -1
    while (low < high):
```

```
        mid = (int)(low + (high - low) / 2)
        if self.url_md5_seen[mid] < md5_item:
            low = mid + 1
        elif self.url_md5_seen[mid] > md5_item:
            high = mid
        elif self.url_md5_seen[mid] == md5_item:
            return mid
    if low >= self.url_md5_seen.__len__():
        return -1
    if self.url_md5_seen[low] == md5_item:
        return low
    else:
        return -1
```

在每次存入数据之前,先检查当前的`url`是否已经被存入, 如果已经被存入, 那么不存入; 如果没有被存入, 那么存入。

```
                if self.binary_md5_url_search(md5url) > -1:  # 存在当前MD5
                    print("有重复!!!!!!!!!!!!!!!")
                    pass
                else:
                    ...
```

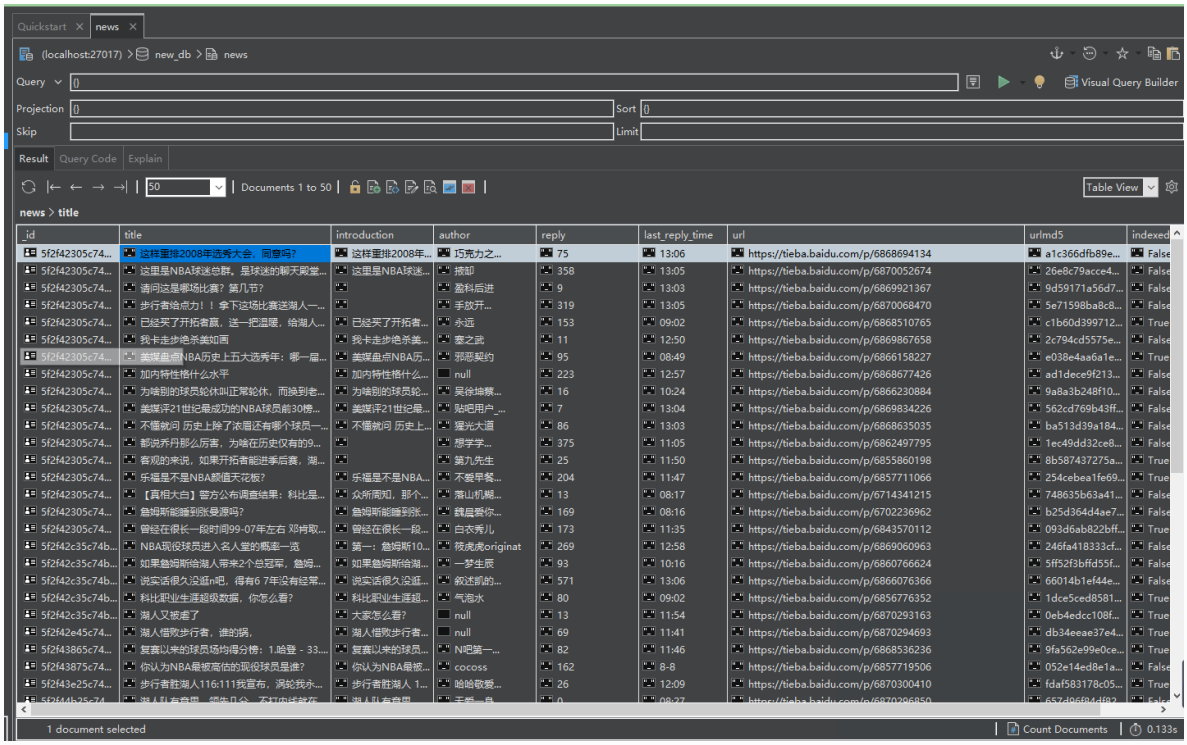
在管道文件`piplines.py`中设置数据库接口, 存入数据。

```
class MongoDBPipeline(object):
    def __init__(self):
        host = settings["MONGODB_HOST"]
        port = settings["MONGODB_PORT"]
        dbname = settings["MONGODB_DBNAME"]
        sheetname = settings["MONGODB_SHEETNAME"]
        # 创建MONGODB数据库链接
        client = pymongo.MongoClient(host=host, port=port)
        # 指定数据库
        mydb = client[dbname]
        # 存放数据的数据库表名
        self.post = mydb[sheetname]

    def process_item(self, item, spider):
        data = dict(item)
        # self.post.insert(data)    # 直接插入的方式有可能导致数据重复
        # 更新数据库中的数据，如果upsert为Ture，那么当没有找到指定的数据时就直接插入，反之不执行插入

        self.post.update({'urlmd5': item['urlmd5']}, data, upsert=True)
        return item
```

通过`Studio 3T`可以查看爬取下来的数据



索引构建特性

索引构建代码位于`indexbuilder/`目录下。

中文分词

Whoosh自带的Analyzer分词仅针对英文文章，而不适用于中文。从jieba库中引用的ChineseAnalyzer保证了能够对Documents进行中文分词。同样，ChineseAnalyzer在search时也能够对中文查询query提取关键字并进行搜索。

```
analyzer = ChineseAnalyzer()

# 创建索引模板
schema = Schema(
    Id=ID(stored=True),
    title=TEXT(stored=True, analyzer=analyzer),
    url=ID(stored=True),
    reply=NUMERIC(stored=True, sortable=True),
    author=TEXT(stored=True),
    last_reply_time=TEXT(stored=True),
    introduction=TEXT(stored=True, analyzer=analyzer),
)
```

Query类提供搜索API

Query类自动执行了从index索引文件夹中取倒排索引来执行搜索，并返回一个结果数组。

```
if __name__ == '__main__':
    q = Query()
    q.standard_search('')
```

搜索引擎特性

搜索引擎代码位于`searchengine/`目录下。

Django搭建Web界面

Django适合Web快速开发。result页面继承了main页面，搜索结果可以按照result中的指示显示在页面中。在django模板继承下，改变main.html中的页面布局，result.html的布局也会相应改变。

```
def search(request):
    res = None
    if 'q' in request.GET and request.GET['q']:
        res = q.standard_search(request.GET['q'])   # 获取搜索结果
        c = {
            'query': request.GET['q'],
            'resAmount': len(res),
            'results': res,
        }
    else:
        return render_to_response('main.html')

    return render_to_response('result.html', c) # 展示搜索结果
```

示例界面

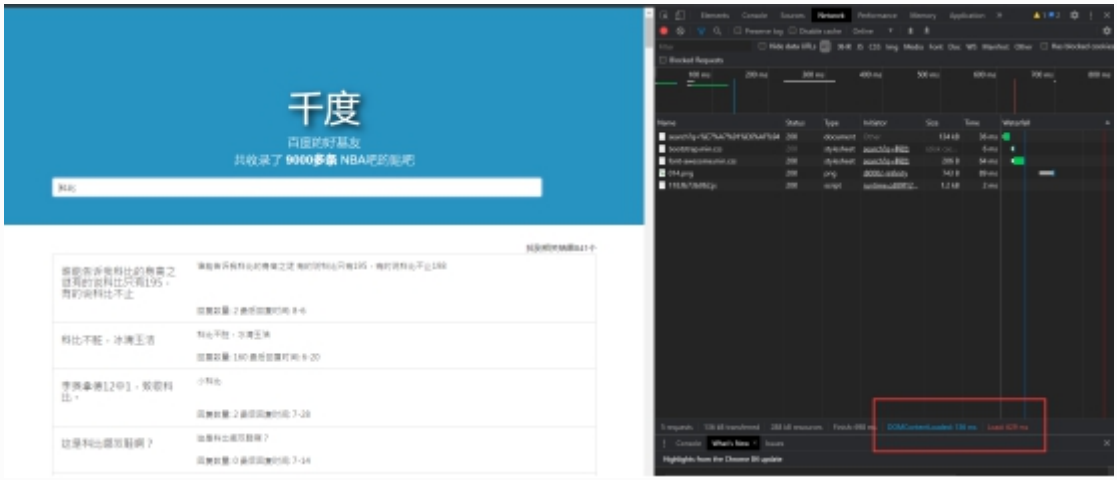
搜索引擎评估结果

1、 Query1：科比

	URL	是否相关
1	https://tieba.baidu.com/p/6861920352	是
2	https://tieba.baidu.com/p/5568318409	是
3	https://tieba.baidu.com/p/6843823788	是
4	https://tieba.baidu.com/p/6811342090	是
5	https://tieba.baidu.com/p/6485852642	是

Precision@5：5/5=1

Responding time：629ms



2、 Query2：MVP

	URL	是否相关
1	https://tieba.baidu.com/p/6710552149	是
2	https://tieba.baidu.com/p/6859952281	是
3	https://tieba.baidu.com/p/6403404483	是
4	https://tieba.baidu.com/p/6830017137	是
5	https://tieba.baidu.com/p/6849737670	是

Precision@5：5/5=1

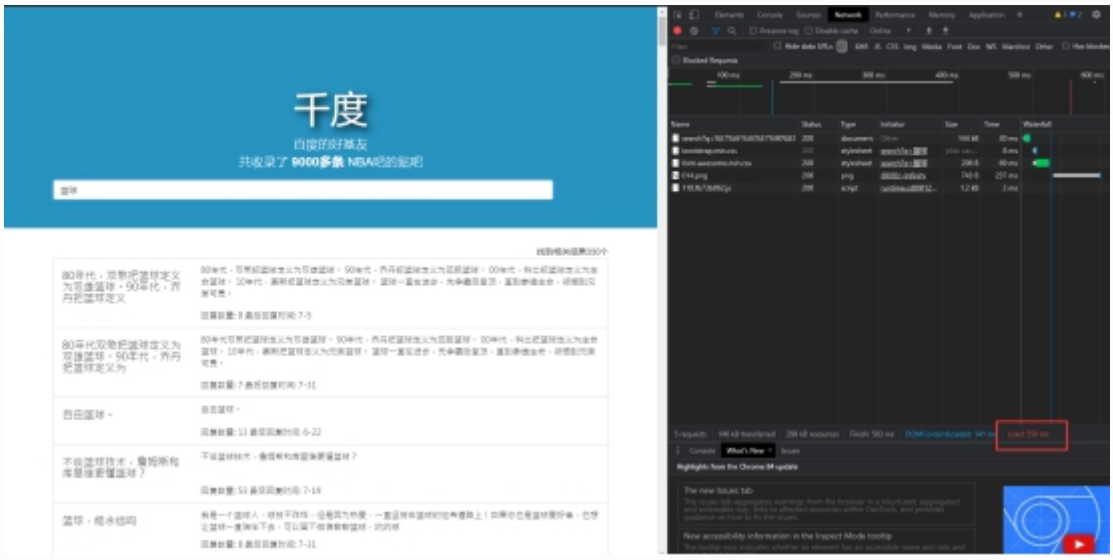
Responding time：491ms

3、 Query3：篮球

	URL	是否相关
1	https://tieba.baidu.com/p/6783779720	是
2	https://tieba.baidu.com/p/6847910160	是
3	https://tieba.baidu.com/p/6738287761	是
4	https://tieba.baidu.com/p/6172464331	是
5	https://tieba.baidu.com/p/6340825427	是

Precision@5：5/5=1

Responding time：550ms

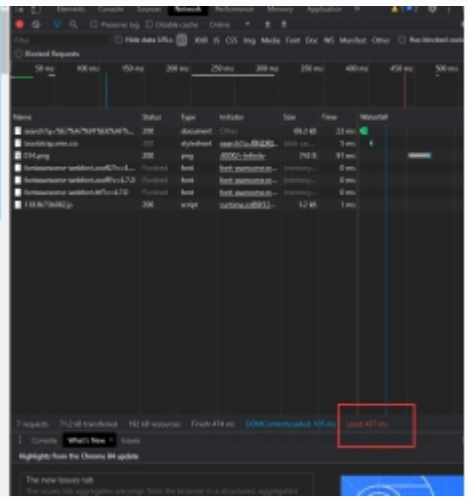


4、 Query4：科比和邓肯

	URL	是否相关
1	https://tieba.baidu.com/p/6795232184	是
2	https://tieba.baidu.com/p/6433023087	是
3	https://tieba.baidu.com/p/5728242690	是
4	https://tieba.baidu.com/p/6785343202	是
5	https://tieba.baidu.com/p/6838650416	是

Precision@5：5/5=1

Responding time：437ms



5、 Query5：篮板王

	URL	是否相关
1	https://tieba.baidu.com/p/6868635035	是
2	https://tieba.baidu.com/p/6741871545	是
3	https://tieba.baidu.com/p/6870308031	是
4	https://tieba.baidu.com/p/6812117274	否
5	https://tieba.baidu.com/p/6833705399	否

Precision@5：4/5 = 0.8

Responding time：440ms

6、 Query6：火箭和快船

	URL	是否相关
1	https://tieba.baidu.com/p/6868417432	是
2	https://tieba.baidu.com/p/6863421316	是
3	https://tieba.baidu.com/p/6866688504	是
4	https://tieba.baidu.com/p/6843418623	是
5	https://tieba.baidu.com/p/6869790882	是

Precision@5：5/5 = 1

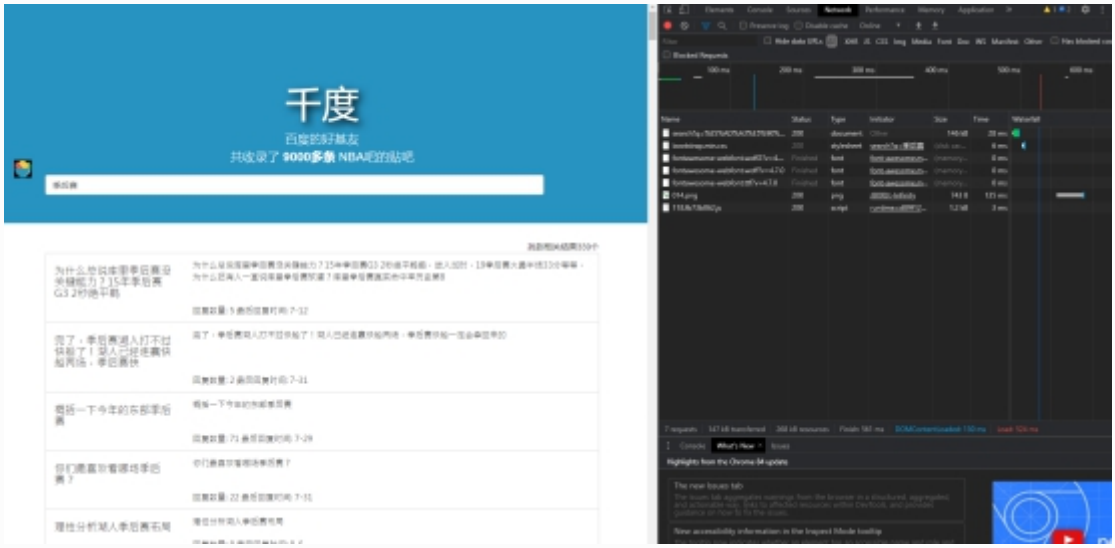
Responding time：395ms

7、 Query7:季后赛

	URL	是否相关
1	https://tieba.baidu.com/p/6806665073	是
2	https://tieba.baidu.com/p/6852223264	是
3	https://tieba.baidu.com/p/6144345739	是
4	https://tieba.baidu.com/p/6849654796	是
5	https://tieba.baidu.com/p/6864602741	是

Precision@5：5/5 = 1

Responding time：524ms

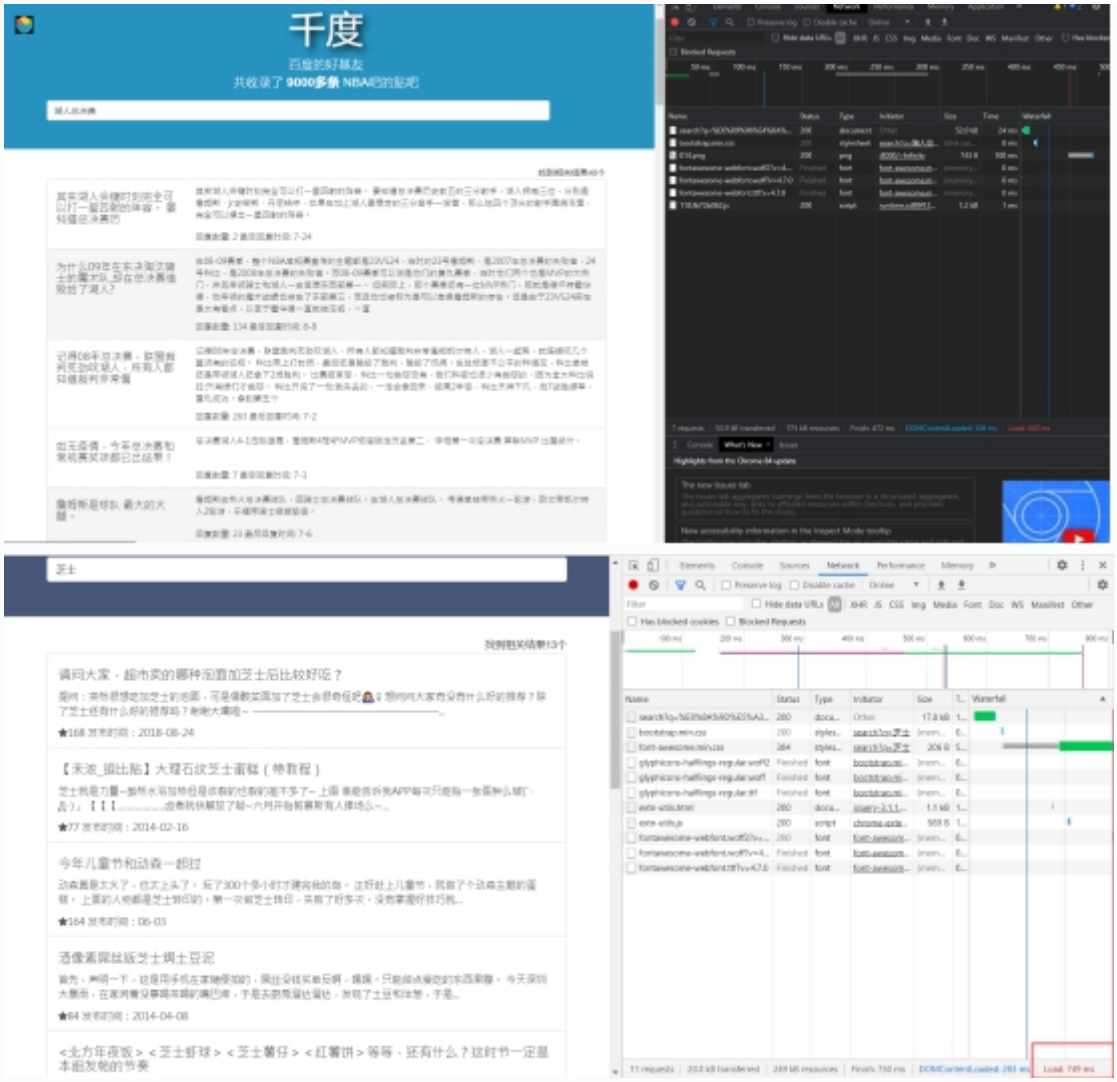


8、 Query8：湖人总决赛

	URL	是否相关
1	https://tieba.baidu.com/p/6834114334	是
2	https://tieba.baidu.com/p/6857330801	是
3	https://tieba.baidu.com/p/6119351434	否
4	https://tieba.baidu.com/p/6772497317	是
5	https://tieba.baidu.com/p/6785474060	是

Precision@5：4/5 = 0.8

Responding time：440ms

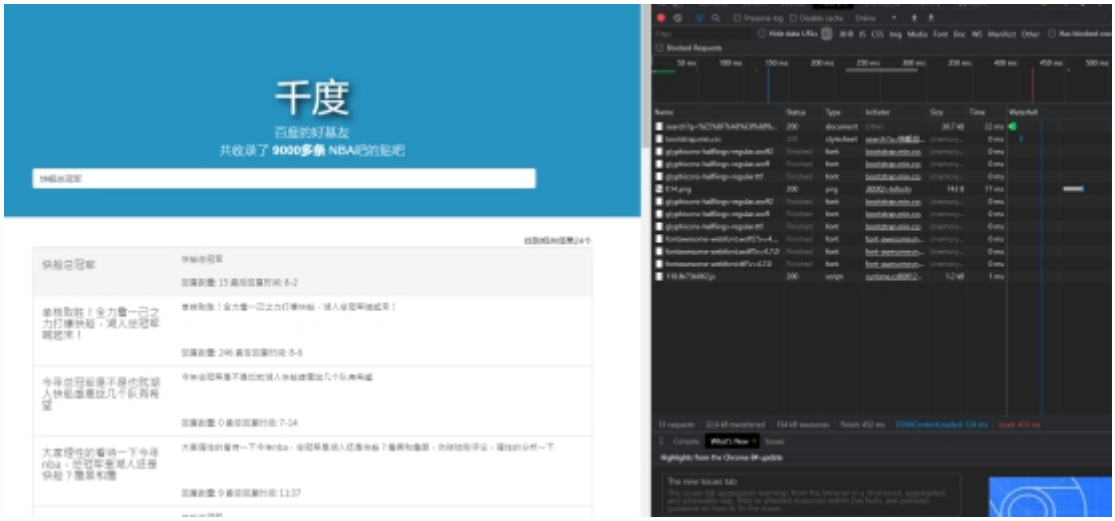


9、 Query9：快船总冠军

	URL	是否相关
1	https://tieba.baidu.com/p/6853256794	是
2	https://tieba.baidu.com/p/6851853074	否
3	https://tieba.baidu.com/p/6811645295	是
4	https://tieba.baidu.com/p/6870398992	否
5	https://tieba.baidu.com/p/6870171326	是

Precision@5：3/5 = 0.6

Responding time：413ms



10、 Query10：詹姆斯mvp

	URL	是否相关
1	https://tieba.baidu.com/p/6830017137	是
2	https://tieba.baidu.com/p/6830236659	是
3	https://tieba.baidu.com/p/6836193707	是
4	https://tieba.baidu.com/p/6870440543	是
5	https://tieba.baidu.com/p/6868081362	是

Precision@5：5/5 = 1

Responding time：448ms

10条query的平均响应时间为476ms

数据来源链接

https://tieba.baidu.com/f?ie=utf-8&kw=nba