数据分析小项目 01 __《少年的你》豆瓣短评__V1.0

目录

本文以《少年的你》为例，简单实现了爬取数据--保存数据--分析数据--图表展示的全流程. 此为第一个版本，有很多需要地方将在后续版本中改进.

1. 豆瓣短评数据抓取:

首先，去网上查一下豆瓣的反爬机制，豆瓣从 2017.10 月开始全面禁止爬取数据:

1) 白天 1 分钟最多可以爬取 40 次，晚上一分钟可爬取 60 次数，超过此次数则会封禁 IP 地址.

2) 非登录状态下，最多能爬 200 条数据.

3) 登录状态下，最多能爬 500 条数据，也就是前 25 页.

本文抓取的是《少年的你》豆瓣热门短评前 25 页的数据.

1.1 网页分析 ： 审查网页元素，获取目标网站树状结构

目标网页网址为:

https://movie.douban.com/subject/30166972/comments?sort=new_score&status=P

如下图所示:

本此数据爬取主要获取的内容有：

- 评论用户 ID
- 评论内容
- 评分
- 评论日期
- 支持数

分析一下网页结构，每一页都有 20 条评论，即有 20 个”comment-item”中,要

爬取的数据都在 comment-item 中，所以在每个页面依次提取 20 个”comment-item”中的数据即可.

最后再分析一下翻页的逻辑:

第 1 页 url 如下:

https://movie.douban.com/subject/30166972/comments?start=0&limit=20&sort=new_score&status=P

第 2 页 url 如下:

https://movie.douban.com/subject/30166972/comments?start=20&limit=20&sort=new_score&status=P

第 3 页 url 如下:

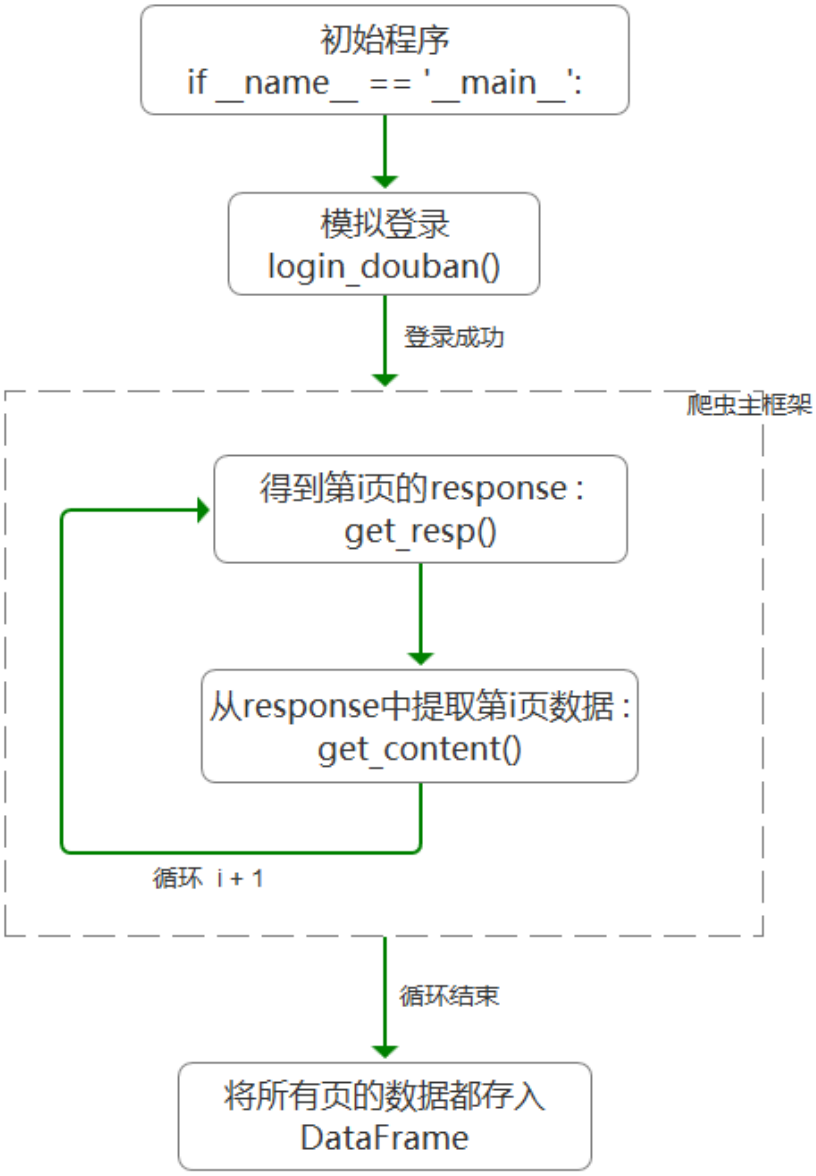
https://movie.douban.com/subject/30166972/comments?start=40&limit=20&sort=new_score&status=P

看出来，翻页就是在 start=0 上加 20，每翻页一次加 20.

这样，网页的逻辑就分析完成了，下一步开始正式写爬虫.

1.2 **数据爬取 ： 用** request +xpath **爬取前** 25 **页数据**

此爬虫的主框架如下:



(1) 模拟登录 login_douban()函数

用的是 requests.Session.post()来发送参数模拟登录.

```
r = requests.Session.post(
                    login_url,          # 登录URL
                    headers=headers,    #主要是要header中的user-agent参数
                    data=data           #请求头中的一些参数比如name/password等
                    )
```

使用 fake_useragent 库中的 UserAgent 来伪造请求头中需要的 UserAgent.

设置好 requests.Session.post()需要的各项参数后，直接传入就可以了.

```python
s = requests.Session()  #用于生成Session对象，用于保存Cookie

# 模拟登录.
# 方法是需要在后台获取登录的 URL并填写请求体参数，然后 POST 请求登录
def login_douban():
    # 登录URL
    login_url = 'https://accounts.douban.com/j/mobile/login/basic'
    # 请求头
    ua = UserAgent()    #用fake_useragent库中的UserAgent来伪造UserAgent.
    headers = {'user-agent': ua.random}
    # 传递用户名和密码
    data = {...}
    try:
        r = s.post(login_url, headers=headers, data=data)
        r.raise_for_status()
    except:
        print('登录请求失败')
        return 0
    # 打印请求结果
    print("(模拟登录函数)====r.text: ",r.text)
    return 1    #是因为想用后面用if login_douban():来判断是否登录,#此处返回1,为真.
```

(2) 代理 IP

试了一些免费代理 IP，总是被封，图省事，直接用的收费的，选的阿布云，一小时 1 块钱就可以了，收费代理的接入也很简单，参照文档就可以了.

阿布云地址: https://center.abuyun.com/#/cloud/http-proxy/tunnel/lists

```python
# 代理ip,
def get_proxies():
    # 代理服务器(购买的阿布云)
    proxyHost = "http-pro.abuyun.com"
    proxyPort = "9010"

    # 代理隧道验证信息
    proxyUser = "H477376K5G8P470P"      # 替换为你自己买的
    proxyPass = "0F146FC04BF2DA42"      # 替换为你自己买的

    proxyMeta = "http://%(user)s:%(pass)s@%(host)s:%(port)s" % {
        "host": proxyHost,
        "port": proxyPort,
        "user": proxyUser,
        "pass": proxyPass,
    }

    proxies = {
        #"http": proxyMeta,
        "https": proxyMeta,    #要爬取类型是https的urls,只需要类型是https的代理
    }

    print("(阿布云代理IP)====proxies: ", proxies)
    return proxies
```

(3) 获取页面 response

主要用的 requests.get()方法来获取 response，上一步用的收费代理 ip 也是作为参数传入 requests.get()中.

设置好参数 url, headers,proxies 和 cookies 后，传入 requests.get()就行了.

```python
resp = requests.get(url, headers=headers, proxies= proxies,cookies=cookies )
```

```
#  将url,headers,proxies传入requests.get(),获取当页的respone.
def get_resp(movieId, currentPage):
    # 1) 拼接url
    c = ('https://movie.douban.com/subject/',str(movieId), '/comments?start=' , str
    url = ''.join(c)     #'表示直接拼接,如果是'-'.join(c)则表示c的字符串之间用-相连接.
    print('(get_resp函数)====想要爬取的此页的url:',url)

    # 2) 用fake_useragent库中的UserAgent来伪造UserAgent.
    # cookie会过期, 每次都要重新添加.
    cookies = {...}
    headers = {...}
    print('(get_resp函数)====User-agent:',ua.random)

    # 3) 将url,header,proxies传入requests.get()函数中,向网页发起请求,拿到response.
    while True:
        try:
            proxies = get_proxies()    #得到参数
            resp = requests.get(url, headers=headers, proxies= proxies,cookies=cookies )
            print("(get_resp函数)====resp.status_code:", resp.status_code)
            return resp
            break
        except requests.ConnectionError as e:...
        except requests.Timeout as e:...
        except requests.RequestException as e:...
        except KeyboardInterrupt:
            print("(get_resp函数)====Someone closed the program")
    print("================================================================================")
```

(4) 从 response 中提取数据

a) 提取用户名和点赞数很简单，只要用 xpath 按照常规方法提取出来,然后存入对应的列表就可以了.

```
# 用户名
# 下面的user_name返回的是一个列表,包含本页20个user_name, format(j)的作用有点类似于表达这个列表的索引.
user_name = x.xpath('//div[@class="comment-item"]/div/a/@title'.format(j))
#print('==========user_name:',user_name)
username_list.append(str(user_name[j-1]).strip())    # strip()用于移除字符串头尾的空格
#print('====username_list:',username_list)

# 有多少个人点赞
# 若没有人点赞的时候,有对应节点,数据值为0 ，一页能抓取20个点赞数据,不需要特殊处理
like = x.xpath('//span[@class="comment-vote"]/span/text()'.format(j))  #.format(j)此处的作用是通
like_list.append(str(like[j-1]))
#print('====like_list:',like_list)
```

b) 提取评论内容时，有些麻烦.

当没有文字评论内容时，存在子节点，但内容为空。提取不到内容，最后

保存进列表后，会导致数据错位. 所以此处要判断，爬取的评论内容是否为空，如果为空，就用”无”来填入.

```
a = x.xpath('//div[@class="comment-item"][{}]/div[2]/p/span/text()[.!= ""]'.format(j))
c = len(a)  #a要么是空列表,要么是只有一个值的列表. c=0时,表示无评论.c=1时,表示有评论.
#print('==========a:', a)
#print('==========c:', c)
if c:
    content = x.xpath('//div[@class="comment-item"][{}]/div[2]/p/span/text()'.format(j))
    content_list.append(str(content[0]).strip())
    #print('========本条评论为:', a)
else:
    content_list.append('无')      #当没有评论时,用代替.
    #print('========本条无评论:', a)
#print('====content_list:', content_list)
```

c) 爬取评分时，遇到巨坑 ！

和没有文字评论还不一样，当没有评分时，连节点也没有。并且当没有评分时，时间 xpath 路径是不一样的，所以先要判断是否有评分，再去根据不同情况提取时间数据.

```
# 评分
# 和没有文字评论还不一样，当没有打星时，连节点也没有。并且有没有评分，时间xpath路径是不一样的
# 判断//span[@class="comment-info"][{}]下有几个span标签,b=3表示有打星，b=2表示没有打星.
b = len(x.xpath('//div[@class="comment-item"][{}]/div[2]/h3/span[@class="comment-info"]/span'.format(j)))
#print('=======b:',b)
if b == 3:...
else:
    date = x.xpath('//div[@class="comment-item"][{}]/div[2]/h3/span[@class="comment-info"]/span[2]/text()'.format
    print('date:',date)
    date_list.append(str(date[0]).strip().strip('/n'))
    score_list.append('0')
    #print('========本条没有打星')
```

1.3 数据保存 ： 创建 DataFrame 并将数据导出为.csv

将 5 个 list 中 scrapyPage 页的所有数据保存进 DataFrame.

```
infos = {'username': username_list, 'score': score_list,
         'content': content_list, 'date': date_list, 'like': like_list}

data = pd.DataFrame({k:pd.Series(v[:20*scrapyPage]) for k,v in infos.items()},
                    columns=['username', 'score', 'content', 'date', 'like'])
```

2. 数据分析

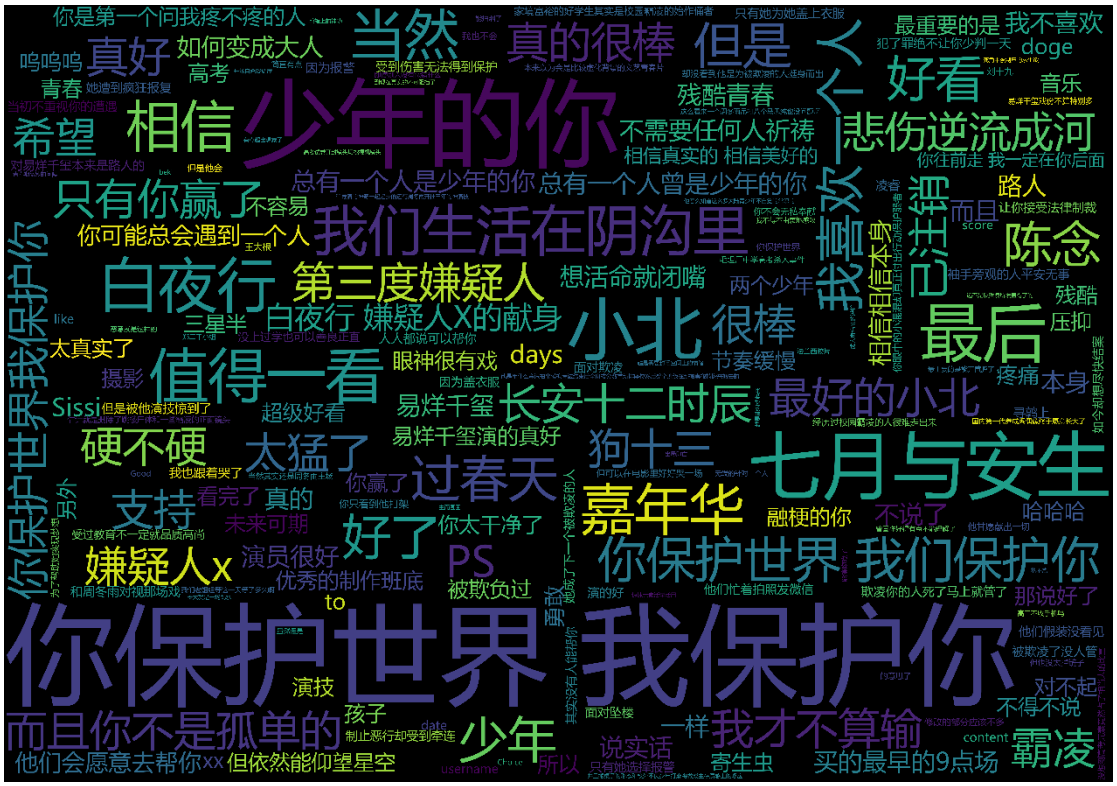
2.1 制作词云 ：WordCloud+jieba

想通过豆瓣短评分析一下，观众的评论主要集中在哪些点上，所以想到了通过词云的方式来展示.

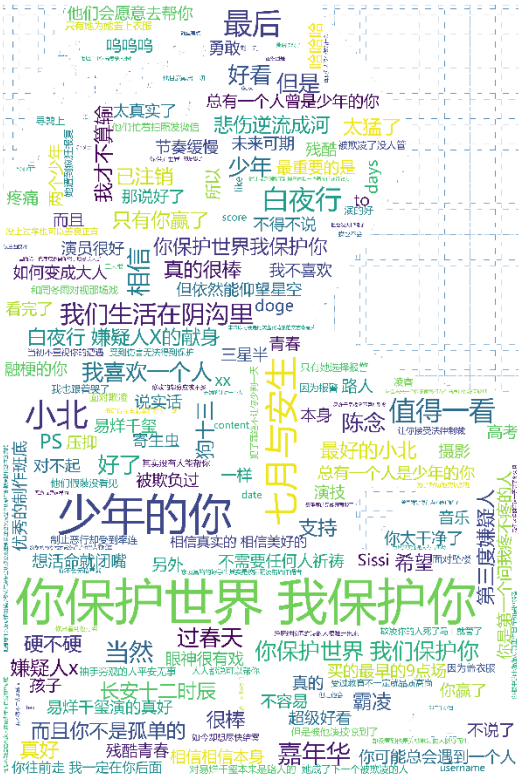
制作词云的前提是要先对短评语句分成一个一个词，英文语句是已经通过空格，将语句分成了一个个的单词，但中文是不一样,所以需要选择一个中文分词工具，这里选择了 jieba. 它 是一款.基于 Python 的中文分词工具，安装使用都非常方便,功能强悍,推荐使用.

Jieba 和 WordCloud 的使用都不复杂，参考网上的教程看一下就可以了.

构建词云的时候，不设置背景图片的话，就是采用默认图片，是一个矩形图片.



也可以自定义背景图片，下面图示，前一张是背景图，后一张图是生成的词云.



2.2 **统计词频** : jieba + counter

使用 jieba 进行分词，然后中 collections 中的 counter 计数器来统计词频.

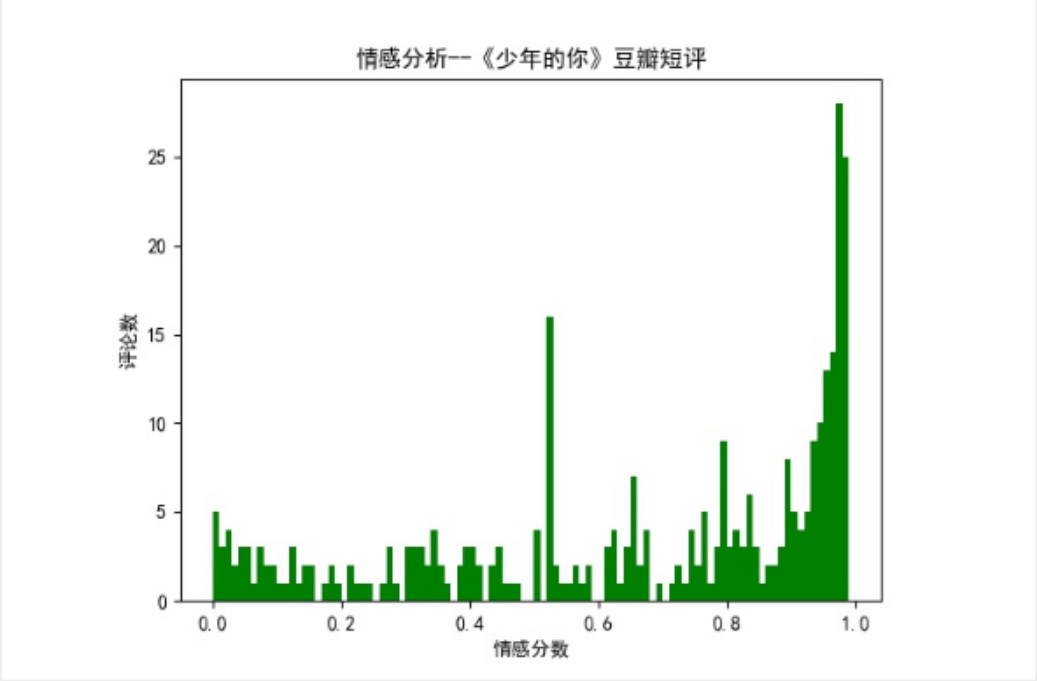
逻辑大概是，先删除语句中的非法字符，然后用 jieba 进行分词，再在分词中去掉你不想统计的词(停用词)，得到的就是你想统计的分词列表．最后用计数器 counter()统计.

	A	B	C
1		分词	次数
2	0	千玺	135
3	1	少年	126
4	2	保护	112
5	3	世界	110
6	4	冬雨	106
7	5	校园	105
8	6	希望	101
9	7	演技	93
10	8	周	91
11	9	陈念	81
12	10	演员	78
13	11	抄袭	78
14	12	说	72
15	13	太	63
16	14	里	61
17	15	这部	59
18	16	导演	57
19	17	想	56
20	18	霸凌	56

2.3 情感分析 ： 用 snownlp 根据电影短评进行简单的情感分析

Snownlp 是中文自然语言处理工具，和 jieba 有些类似，也可以进行分词，词性标注，情感分析等功能.此处我们使用了它的情感分析功能.

此功能的返回值为正面情绪的概率，越接近 1 表示正面情绪,越接近 0 表示负面情绪.

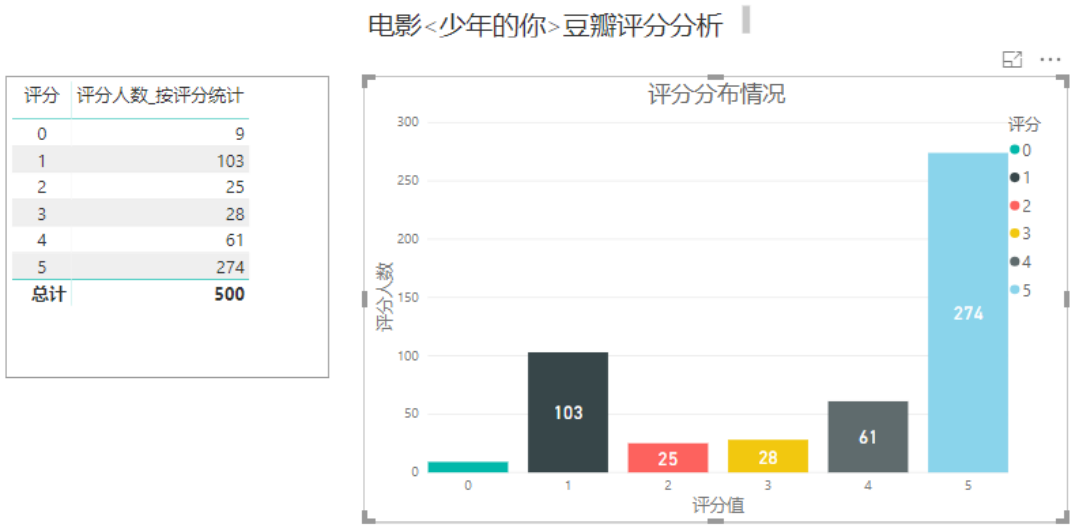


情感分数越接近1，评价越是正面. 情感分数越接近0，评价越是负面.

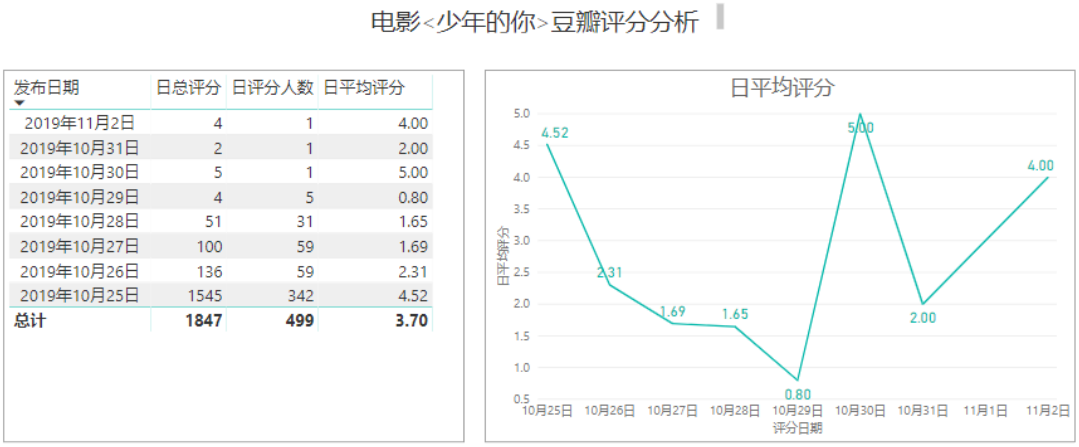
3. 数据展示

3.1 评分分布图

爬取的数据导入 PowerBI，尝试用 PowerBI 进行一些分析. 发现 PowerBI 使用特别方便，展示性也好，强烈推荐.



3.2 每日评分变化趋势图



3.3 PowerBI 的词云的插件

PowerBI 中也有词云的插件，试用了一下，发现还是需要提前进行分词，统计词频,然后再用内部插件做词云，而且可自定义的程度不高，背景图也不能换，不推荐使用 PowerBI 的词云插件.

使用PowerBi的词云插件制作的词云图

使用WordCloud制作的词云图，然后再导入PowerBi

4. **需要改进的地方**

1) 爬虫代码写在一个 py 文件里，比较混乱，应该将原本换在一个文件里的代码拆分为合理的模块，比如将付费代理封装成一个类，写在单独的类中.

2) 这个版本的 cookie 是复制添加的，每次运行都要重新复制，比较蠢.下版本应该设置伪装登录，使得 cookie 能自动获取和自动更新.

3) 流程设计有问题，将数据保存放在了所有一步，必须等所有数据都爬完了之后才能保存，一旦爬虫中途中断，最后一个数据也得不到.一个版本应该改为边爬边保存.

4) 断点续爬，如果中途出错，不能从头开始，要从上次停止的地方继续爬取.