第119天：Python 爬取豆瓣电影 top 250

原创 豆豆 Python技术 2月4日

豆瓣作为一个汇聚书影音内容的社区网站，得到了大量用户的认可和青睐，现在很年轻人在看电影或者买书之前都会去豆瓣上看一下评分和相关评论，不得不说豆瓣评分在一定程度上很客观的反映了一部作品的受欢迎程度。

今天，我们就抓取下豆瓣电影 top 250 的相关数据。

首先需要先明确下我们所需要获取的信息如下：名称，导演，国家，链接，上映时间，类型，评分（五星，四星占比）以及评价人数。

分析网址

首先我们观察下豆瓣电影 top 250 的网址变化后会发现，top 250 共计分为 10 页，每一页 25 条记录，网址为 `https://movie.douban.com/top250?start={start}&filter=` 其中 start 从 0 开始，每次递增 25，到 225 结束，相信大家都可以理解。

因此，我们使用函数 `getUrls` 来获取所有的链接地址。

```
def getUrls():
    url_init = 'https://movie.douban.com/top250?start={0}&filter='
    urls = [url_init.format(index * 25) for index in range(10)]
    return urls
```

但貌似这个列表页面获取不到我们所需的全部信息，还需要去到具体电影的详情页才行。因此，我们可以先爬取列表页，然后从列表页获取到详情页的链接地址，然后从详情页获取我们所需的详细信息。

分析网页

接下来我们需要确认一下我们需要的具体详细信息藏在哪个位置。打开网址 `https://movie.douban.com/top250?start=0&filter=`，然后打开 chrome 的控制台。

比如我们定位到「霸王别姬」。可以看到每一部电影都在一个单独的 li 标签内。在每个 li 里面：

电影链接在 `div.info > div.hd > a` 里面。

获取到「霸王别姬」的详情页地址之后，我们再对该详情页进行分析。

我们发现，所有的电影信息都在 `<div id="content">` 这个标签内的。

- 标题在 property="v:itemreviewed" 的 span 里面。
- 上映年份在 class="year" 的 span 里面。
- 导演在 class="attrs" 的 span 里面。
- 上映国家这个标签比较特殊，没有唯一性，所以可以使用正则表达式。
- 类型在 property="v:genre" 的 span 里面。
- 电影评分在 property="v:average" 的 strong 里面。
- 评价人数在 property='v:votes' 的 span 里面。
- 具体评分在 class_='ratings-on-weight' 的 div 里面。

嗯，很好，经过分析之后，我们知道了我们所需信息的具体位置，接下来对这些内容进行抓取并解析即可。

获取数据

抓取网页数据需要用的 `requests` 库，解析网页需要用的 `BeautifulSoup` 库，因此先将二者引我们的程序。

```
import bs4 as bs4
import requests
import re
```

因为获取电影链接时需要解析网页，获取详细信息时同样需要解析网页，因此我们先定义一个名为 `get_page_html(url)` 的函数，用于从 url 获取 html 内容。

同时，为了防止反爬虫，我们需要定义一些 headers。

```
def get_page_html(url):
    headers = {
        'Referer': 'https://movie.douban.com/chart',
        'Host': 'movie.douban.com',
        'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0) AppleWebKit/537.36 (KHTML, li
    }
    try:
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            return response.text
        return None
    except RequestException:
        return None
```

接下来，我们开始获取数据，需要一个解析电影详情页链接地址的函数。

```
def get_movie_url(html):
    ans = []
    soup = bs4.BeautifulSoup(html, 'html.parser')
    items = soup.select('li > div.item')
    for item in items:
        href = item.select('div.info > div.hd > a')[0]['href']
        ans.append(href)
    return ans
```

在这个函数中，我们传入整个页面的 html 内容，该函数负责将电影详情页链接地址解析出来并以列表形式返回给调用方。

拿到了电影详情页的链接地址，最后我们只需要将详细信息解析出来即可。我们可以看到，详情页和列表页有很多信息都是重复的，因此我们可以从详情页获取我们需要的所有信息。

我们需要定义一个函数，用来解析详情页。

```
# 【名称，链接。导演，国家，上映时间，类型，评分，[五星，四星占比]，评价人数】
def get_movie_info(url):
    ans = {}
    html = get_page_html(url)
    soup = bs4.BeautifulSoup(html, 'html.parser')
    content = soup.find('div', id='content')

    ## 作者
    title = content.find('span', property='v:itemreviewed').text

    ## 上映年份
    year = content.find('span', class_='year').text[1:5]

    ## 导演
    directors = content.find('span', class_='attrs').find_all('a')
    director = []
    for i in range(len(directors)):
        director.append(directors[i].text)

    ## 上映国家/地区
    country = content.find(text=re.compile('制片国家/地区')).next_element
    typeList = content.find_all('span', property='v:genre')

    ## 影片类型
    type = []
    for object in typeList:
        type.append(object.text)

```

```
    ## 评分
    average = content.find('strong', property='v:average').text

    ## 评价人数
    votes = content.find('span', property='v:votes').text

    ## 具体评分（五星 四星人数占比）
    rating_per_items = content.find('div', class_='ratings-on-weight').find_all('div', class_='item'
    rating_per = [rating_per_items[0].find('span', class_='rating_per').text,
                  rating_per_items[1].find('span', class_='rating_per').text]

    return {'title': title, 'url': url, 'director': director, 'country': country, 'year': year, 'typ
            'average': average, 'votes': votes, 'rating_per': rating_per}
```

豆瓣的数据抓取难度不大，主要是要细心点分析页面结构，将我们所需的信息从错综复杂的网页结构中找出来。因为我们所获取的信息比较多，所以这个函数稍微长了一点。

每一个字段的信息都是通过解析网页源码获得的，所以在准确度上来说是完全没有问题的，同时，BeautifulSoup 的使用也简单易上手，半小时左右就可以入门。关于 BeautifulSoup 的使用可以看 第65天：爬虫利器 Beautiful Soup 之遍历文档。

成功抓取到数据之后，我们还需要定义一个函数，用来将数据缓存到数据库或者本地文件中，用于后续分析。这里为了方便就直接写入文件了。

```
def writeToFile(content):
    filename = 'doubanTop250.txt'
    with open(filename,'a') as f:
        f.write(content + '\n')
```

至此，我们的准备工作已经全部做完，就可以抓取数据了。

```
if __name__ == '__main__':
    list_urls = getUrls()
    list_htmls = [get_page_html(url) for url in list_urls]
    movie_urls = [get_movie_url(html) for html in list_htmls]

    movie_details = [get_movie_info(url) for url in movie_urls[0]]
    for detail in movie_details:
        writeToFile(str(detail))
```

之后，就可以看的我们的数据了。大功告成。

总结

今天我们用 `requests` 库和 `BeautifulSoup` 库对豆瓣电影 top 250 进行了抓取，主要还是对 `BeautifulSoup` 库的练习。本文思路清晰，只是分析网页时需要多点耐心罢了，希望你也可以自己动手练习下，对于代码功底的提高有很大益处。

代码地址

> 示例代码：https://github.com/JustDoPython/python-100-day/tree/master/day-119

系列文章

第118天：Python 之对象的比较与拷贝
第117天：机器学习算法之 K 近邻
第116天：机器学习算法之朴素贝叶斯理论
第115天：Python 到底是值传递还是引用传递
第 114 天：三木板模型算法项目实战
第 113 天：Python XGBoost 算法项目实战
第 112 天：机器学习算法之蒙特卡洛
第 111 天：Python 垃圾回收机制
从 0 学习 Python 0 - 110 大合集总结

PS： 公号内回复：Python，即可进入Python 新手学习交流群，一起**100天计划！**

-END-

Python 技术
关于 Python 都在这里