

# 用 Python 爬取豆瓣冷门高分电影和书籍

## 目录:

一、选题背景:	1
二、分析、开发、测试过程介绍:	2
1、分析过程:	2
2、开发过程:	2
(1) 获取数据:	2
A. 爬虫一和二 (TOP250 榜单):	3
B. 爬虫三 (“最新” 电影榜单):	4
C. 爬虫四 (“小说” 榜单):	7
(2) 解析数据:	7
3、测试过程:	9
三、对自己工作的评价:	10
四、课程总结、对课程的批评和建议:	11

## 一、选题背景:

1. 自己最初之所以选择 Python 这门选修课,一方面源于 Python 语言的简洁和库的强大,另一方面也是最主要的也是自己对于网络爬虫爬取数据的兴趣。因为近几年,大数据、数据挖掘这样的概念非常火爆,并且自己也的确发现数据对我们生活方方面面的影响,切身感受到数据对于未来的重要意义和影响。因此,希望在未来从事数据研究方面的工作。所以希望在本科阶段学习 Python 和爬虫。
2. 自己平时对看电影和看书很有兴趣,但是对于热门精彩的高分影片和书籍早已看完,然后就很容易陷入片荒、书荒的时期,因此想要获得豆瓣上那些不算热门的精致的高分电影和书籍。

基于以上两点,自己确定了这个选题,一方面提升自己 Python 和爬虫的应用能力,另一方面为自己提供一个电影和书籍的推荐渠道。

(项目为原创)

## 二、分析、开发、测试过程介绍：

网络爬虫的过程通常分为三步：获取数据、解析数据、分析数据。本次项目中，重点在于前两者。

### 1、分析过程：

据了解，网络上关于豆瓣电影、书籍的爬虫基本是两种，第一种是直接在某一个排行榜上爬取数据，这样访问的页面少，时间快；第二种是直接 for 循环大遍历，这样获取的数据大，暴力操作也较为容易。但是前者爬取的数量太少，后者太过浪费时间，两种方式对于选题“冷门高分”而言都不是合适的策略。

因此我最终决定结合两种方式的优点来爬取数据，即先在榜单上选取合适的电影或图书，然后把它们的 URL 存下来，再通过这些 URL 去一一访问它们各自的信息页面，然后获取信息并存储。

然后本次项目一共有四个爬虫，分别是用了四个不同的榜单，四个榜单分别是“豆瓣电影 TOP250”，“豆瓣图书 TOP250”，“最新电影”和“小说类图书”。前两个榜单网页 html 的结构大致相似，也较为基础。“最新电影”榜单的网页 html 内容是动态获取的，“加载更多”也需要模拟网页动态点击。“小说类图书”的 html 结构和 TOP250 相似，但是数据量较大，爬虫爬取时极易假死。

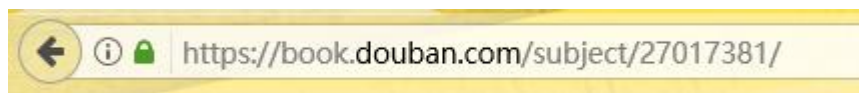
（对于各种爬虫，具体策略及方案的选择和实施会在开发过程中有详细介绍）

### 2、开发过程：

#### （1）获取数据

获取数据阶段是通过 URL 向服务器请求 html。因此，开发前必然要了解豆瓣网站 URL 的请求命令格式。通过访问不同的电影、书籍界面可以发现，它们各自的 URL 前面部分基本是一样的，只是在 subject 后有一段数字代表不同的电影和书籍作品，算是一个特定的标识符。（具体如下图所示）不过通过大量访问发现，

这些标识符是随机的，没有规律，并且不同数字之间也有大量页面是空页面。因此，用 for 循环来遍历直接访问这些页面的量太大，而且效率太低，不予采用。



（图为作品页面的 URL）

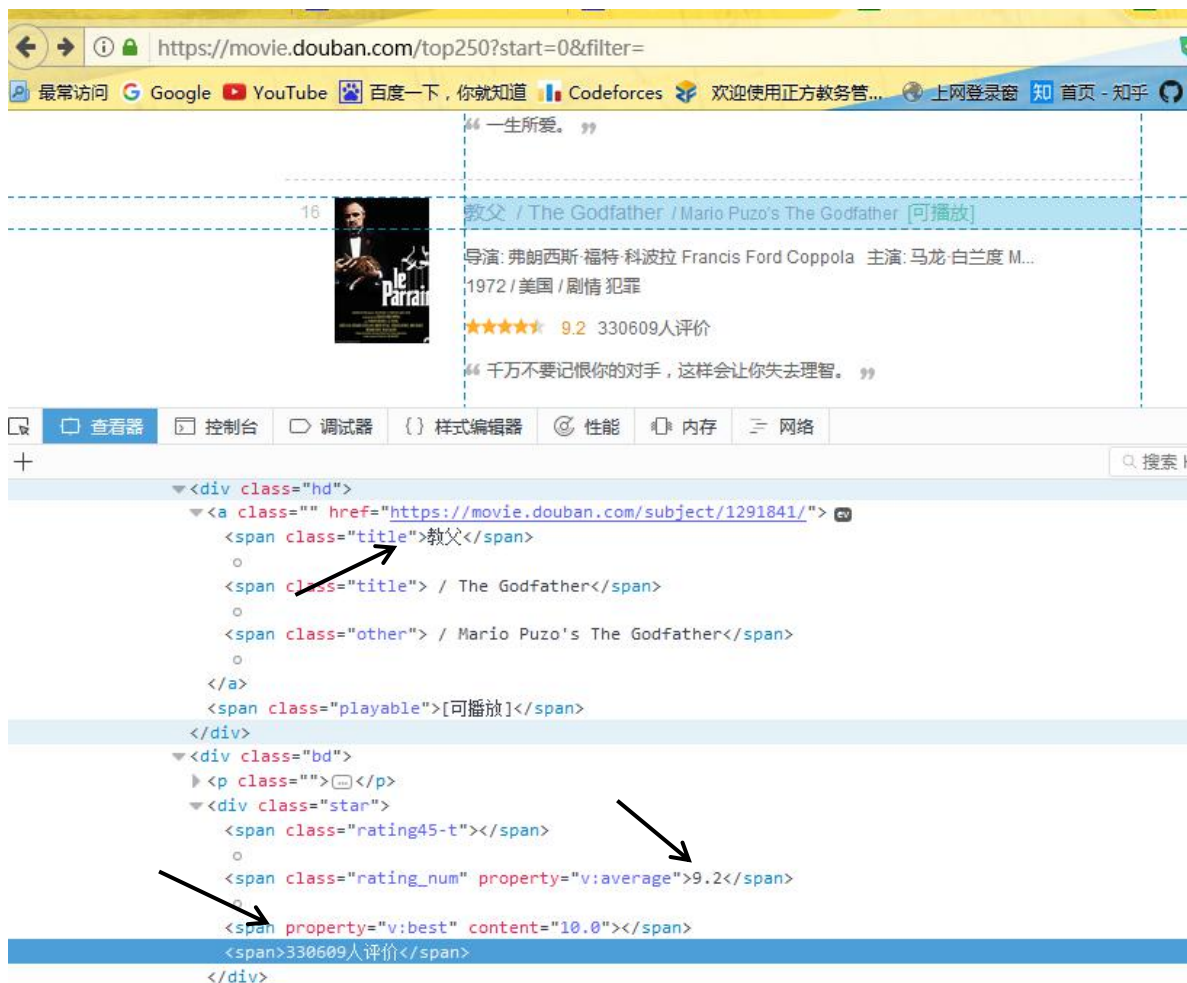
#### A. 爬虫一和二（TOP250 榜单）：

所幸的是，我们发现豆瓣的页面提供了各式各样的榜单，其中比较著名的有 TOP250 榜单。并且 TOP250 榜单不同页的 URL 链接是有规律可寻的。由于每一页有 25 部电影，所以第一页的 start 为 0，第二页为 25，第三页为 50，以此类推……



（图为 TOP 排行榜的 URL）

TOP250 榜单是豆瓣自己提供的评分在前 250 的作品，并且还提供了一些具体的信息（评分，评价人数等）和电影页面的 URL 链接。因此在这个榜单上选取合适的作品，把它们 URL 链接爬下来，再通过这些 URL 进入电影页面爬取更多详细数据，是一个不错的爬虫策略。（由于豆瓣读书 TOP250 的界面与豆瓣电影基本类似，因此不再另截图说明）



(图为 TOP250 页面的 html)

## B. 爬虫三 (“最新”电影榜单)：

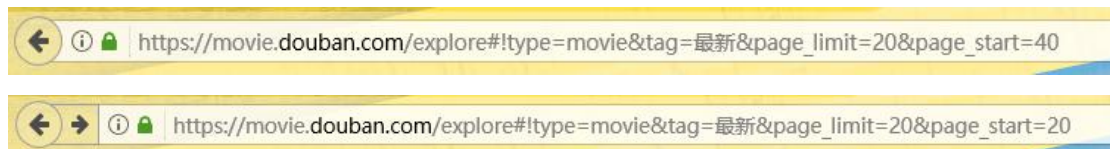
但是这种爬虫策略固然很好，但是毕竟 TOP250 只有 250 部电影，量太少，而且都是已经在被各个地方提及多次的经典电影，并不能满足获取冷门高分电影的需求。因此决定在选电影的“最新”这一排行榜上选择合适的电影爬取，所幸这一排行榜的 URL 也是有规律可寻。

但是最终发现，虽然 URL 也是有规律可寻，但它的分页并不是页面的转换，而是在同一页面设置了动态加载按钮“加载更多”。这样的话用 for 循环遍历访问便是无效的，必须要模拟浏览器点击操作。于是采用第三模块 Selenium 和轻型浏览器 PhantomJS 的组合来用代码模拟浏览器点击操作。考虑到时间因素，用 for 循环连续点击了 20 次，大概是页面有了 400 部作品后开始爬取数据。

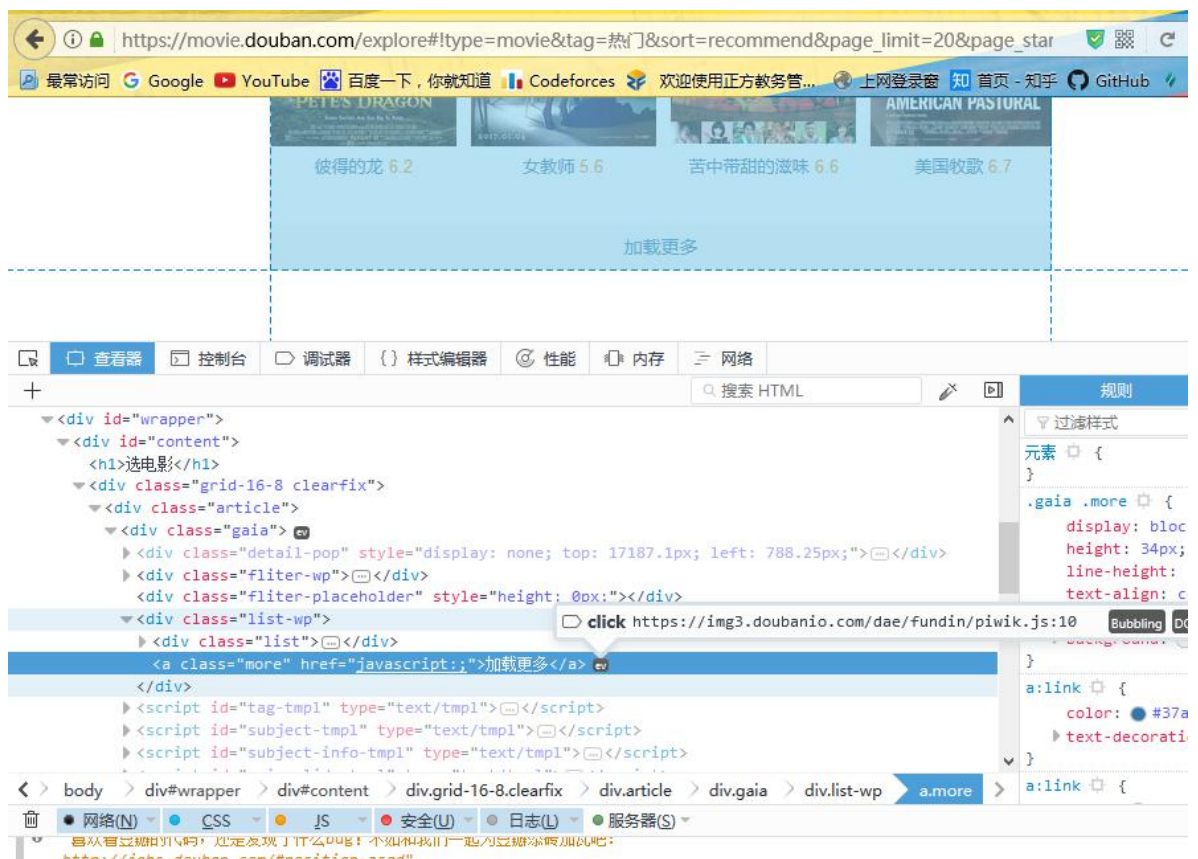
## 选电影

热门 最新 经典 可播放 豆瓣高分 冷门佳片 华语 欧美 韩国  
日本 动作 喜剧 爱情 科幻 悬疑 恐怖 动画

☐ 按热度排序 ☒ 按时间排序 ☐ 按评价排序 ☐ 我没看过的 ☐ 可在线播放

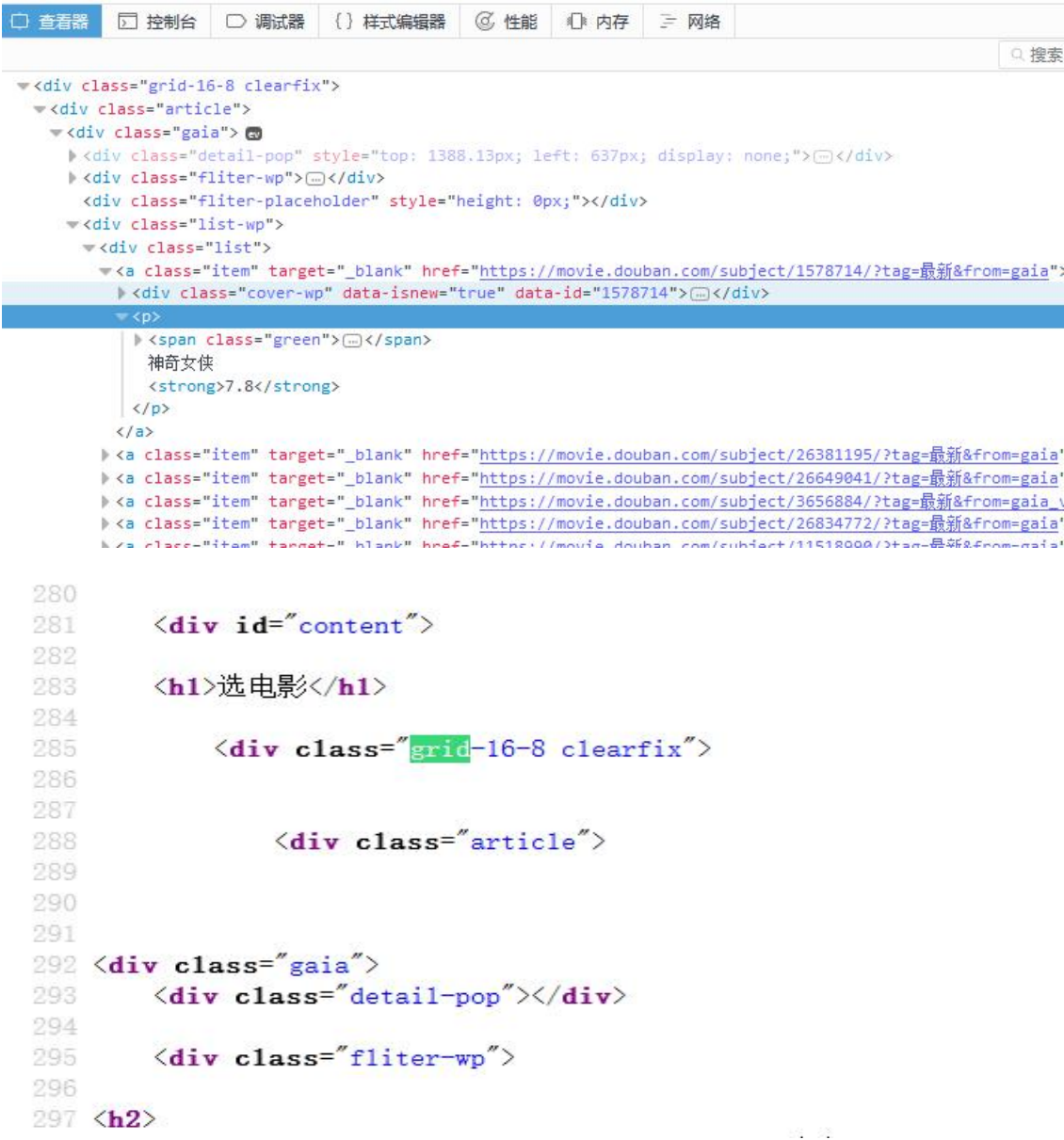


(图为选电影排行榜的页面及 URL)



(图为选电影排行榜的页面动态按钮)

同时，也遇到了一个问题，便是这一页面的电影信息是通过 Javascript 动态获取的，因此虽然在加载后的界面通过查看器能获得信息，但是在通过 requests 获取的 html 源代码中是没有这些信息的。所以决定采用第三方模块 Selenium 来获取动态界面，以解决这一问题。



（图为选电影页面 html 中 Js 加载前后的对比）

不过这次的爬虫和 TOP250 不同，可能会访问大量界面获取大量信息（其实考虑到时间因素，并没有爬取太多数据），但是出于谨慎考虑，为了防止豆瓣封 IP，还是采取了一些简单的反“反爬”策略。由于 Selenium 的机制是模拟浏览器加



载页面，因此使用了 PhantomJS 作为模拟浏览器来加载，一定程度上可以减缓获取 html 的速度，不容易被豆瓣发现。然后又用了 time 模块中的 sleep() 方法，设置每加载一个网页时间间隔为一秒，因此作为反“反爬”策略进行爬虫。

### C. 爬虫四（“小说”榜单）：

该网页的 html 和 TOP250 一样，也是静态的，因此最初也是按照 TOP250 的思路去获取数据，也没什么问题，但是每次获取到一半的时候卡死（如下图所示，在爬到第 12 本书时假死）。

这样陆续卡死 N 次后，不得已做出防假死的策略。即用 while 循环语句和 try……except……语句以及 time 模块中的 timeout 的结合，使爬虫如果在获取某一网页时卡死时，便不断循环地去获取这个网页，直到获取成功为止。在改进策略后才得以发现，第 16 本书《飘》一共申请了 9 次才申请成功（我这里 timeout 时限设置的是 10s，也就是说花了一分半才申请成功），真是难怪会有 N 次卡死。

```
正在爬取第 16 部图书数据
申请HTML共2次。
悉达多 (豆瓣)
正在爬取第 17 部图书数据
申请HTML共4次。
肖申克的救赎 (豆瓣)
正在爬取第 18 部图书数据
申请HTML共9次。
飘 (豆瓣)
正在爬取第 19 部图书数据
申请HTML共3次。

===== RESTART: =====
平凡的世界 (全三部) (豆瓣)
正在爬取第 11 部图书数据
局外人 (豆瓣)
正在爬取第 12 部图书数据
```

（图为第四只爬虫爬取时的界面交互图）

### （2）解析数据

获取数据最终决定使用 requests 和 Selenium 这两个第三方模块作为工具来获取数据，也就是网页的 html。

接下来需要做的便是解析数据了。不过在对于解析数据工具的选择上却有波折，最终直接是用正则表达式，但是正则由于其局限性，匹配的错误率较高，容

易产生错误信息。因此，最终决定换成 Xpath 来解析数据。在网络上，有一句话形容正则和 Xpath，感觉非常形象，“如果提取信息就像找一个建筑，那么正则表达式就是告诉你，这个建筑的左边是什么、右边是什么、以及这个建筑本身有哪些特征，但这样的描述在全国范围内可能有很多地方的建筑都符合条件，找起来还是不太方便，除非你限制范围，比如指定北京海淀区等。而 XPath 就是告诉你这个建筑在中国-北京-海淀区-中关村-中关村大街-1 号，这样找起来就方便了很多”。

同样的，在解析数据之前，必然要先了解豆瓣 html 的格式及规律。所幸作品页面的 html 是静态页面，并且信息也较为全面。最终综合考虑，决定在电影方面，选取电影名称，导演，主演（前两个），剧情简介，海报，评分及评价人数等特征多维度说明一部电影；在图书方面，选取书名，作者简介，内容简介，海报，评分及评价人数等特征说明一本书。



The screenshot shows a web browser displaying the Douban movie page for "T2 Trainspotting". The browser's address bar shows the URL: `https://movie.douban.com/subject/22263645/?tag=豆瓣高分&from=gaia`. The page content includes the movie's poster, cast list (Director: Danny Boyle, Screenplay: John Hodge, Lead Actors: Ewan McGregor, Jonny Lee Miller, Robert Carlyle, Ewen Bremner, James Hogg, etc.), genre (Drama/Crime), and a rating of 8.2 with 13898 reviews. The developer tool is open, showing the HTML source code. The code is structured as follows:

```
<div id="dale_movie_subject_top_icon" ad-status="loaded"></div>
<h1></h1>
<div class="grid-16-8 clearfix">
  <div class="article">
    <div class="indent clearfix"></div>
    <div id="collect_form_22263645"></div>
    <div class="related-info" style="margin-bottom:-10px;">
      <a name="intro"></a>
      <h2></h2>
      <div id="link-report" class="indent">
        <span class="" property="v:summary">
          一晃眼二十年过去。远赴他乡的雷登（伊万·麦克格雷格 Ewan McGregor 饰）再度踏上了爱丁堡的故土。哪知道刚一回来，就撞上了屎霸（艾文·布莱纳 Ewen Bremner 饰）试图自杀，原来，屎霸不仅婚姻失败还再度染上了毒瘾。在雷登的鼓励下，屎霸决定用书写来...
          <br>
          之后，雷登找到了曾经的挚友病孩（约翰·李·米勒 Jonny Lee Miller 饰），病孩因为曾经的跼蹐痛揍了雷登，之后，一个复仇的病孩的邀请下，雷登决定和病孩合伙开一家妓院，之后，擅长设计的屎霸亦加入了进来。那边厢，锒铛入狱的贝格比（罗伯特·卡莱尔 Robert Carlyle 饰），在了解了当年事件的真相后，他亦决定找到雷登向他寻仇。
        </span>
      </div>
    </div>
  </div>
</div>
```



Screenshot of the Douban book page for '白夜行' (White Night) by Haruhiko Arima. The page shows the book's cover, author information, and a star rating of 9.1. The browser's developer tools are open, displaying the HTML structure of the page, including the book's title, author, and a detailed description of the book's content and awards.

白夜行

作者: [日] 东野圭吾  
出版社: 南海出版公司  
原作名: 白夜行  
译者: 刘姿君  
出版年: 2008-9  
页数: 467  
定价: 29.80元  
装帧: 平装  
丛书: 新经典文库 东野圭吾作品  
ISBN: 9787544242516

豆瓣评分  
9.1 184136人评价  
5星 63.1%  
4星 30.1%  
3星 6.0%  
2星 0.5%  
1星 0.3%

每天碎片 点击

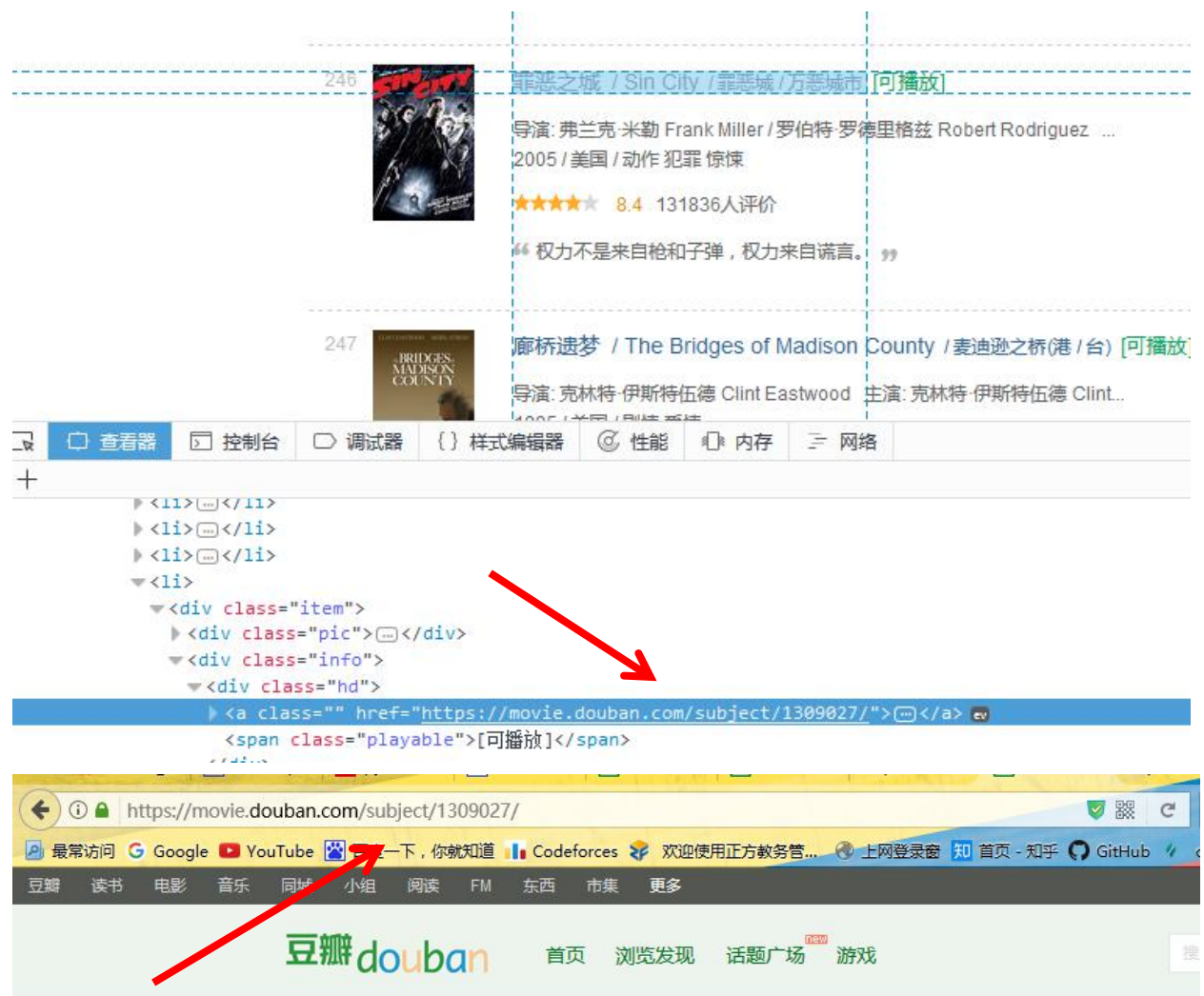
```
<br clear="all">
<div id="collect_form_3259440"></div>
<div class="related_info">
  <h2></h2>
  <div id="link-report" class="indent"></div>
  <h2></h2>
  <div class="indent">
    <div class="">
      <style type="text/css" media="screen">.intro p{text-indent:2em;word-break:normal;}</style>
      <div class="intro">
        <p>
          东野圭吾，日本著名作家。1958年生于大阪。1985年，以第31届江户川乱步奖获奖作品《放学后》出道，开始扬名立万。20年作品逾60部，
          大奖：1999年，《秘密》获第52届日本推理作家协会奖，入围第120届直木奖；2000年，《白夜行》入围第122届直木奖；2001年，《暗恋》
          奖；2004年，《幻夜》入围第131届直木奖；2006年的《嫌疑人X的献身》，获得第134届直木奖、第6届本格推理小说大奖。
        </p>
      </div>
    </div>
  </div>
```

最终，综合各方面因素，对于电影、图书两个 TOP250 榜单，选取的是评价大于八分且评价人数小于 200000 的作品并记录。对于“最新”的排行榜，选取的是前一百部评价大于七点五的作品并记录。在记录方面，除海报外的其他内容均记录在 TXT 文本文件中，海报单独记录并编号，可通过 TXT 文件中的序号信息寻得。

(代码和爬取的数据全都在附带的文件中)

### 3、测试过程

在测试的过程中，值得一提的是发现豆瓣的网站是存在 BUG 的，比如这部罪恶之城，在 TOP250 榜单中给的链接打开却显示“页面不存在”。在运行测试过程中，这个地方卡了很长时间。发现后在代码中加了一个特判，即如果链接打开是页面不存在，则跳过这条链接。



### 三、对自己工作的评价：

整体而言，对于此次项目的自我评价还是比较满意的。项目完成的过程中自己学到了很多的东西，比如 URL 的格式和原理，一系列基础的前端知识如 html 和 Js 等，xpath 的一些语法和格式，基本的反“反爬”策略的设计，以及 python 很多第三方模块如 requests, lxml, Selenium 等的运用。另外，在项目中也遇到了很多 BUG，然后每次调完 BUG 看着爬虫一点点爬数据时，特别有成就感和愉

悦感。唯一遗憾的是，由于时间关系和项目选题等原因，有很多想尝试的东西暂时没有在项目中完成，比如模拟登录，多线程爬虫，更多的反“反爬”策略的设计，还有一些工具如 `BeautifulSoup` 和 `Scrapy` 框架的应用等，十分遗憾。不过所幸是此次项目也极大地激发了自己对于爬虫的兴趣。因此，项目提交后，自己仍想继续学习研究爬虫，并将成果放到 Github 上。因此，最终打分的话，认为自己可以达到 90+。

#### 四、课程总结、对课程的批评和建议：

总体来说，python 这个两学分的选修课要求很严格，不过相比其它同学分低要求的课程而言，收获也是巨大的，并且课程的学习并没有停留在对语法、知识点的记忆上，而是理论与实践相结合，使自己的动手能力也相应提升。

课程的批评和建议的话，我个人的体会是前期过于轻松，后期过于忙碌。大概是由于实验课和理论课时间错开的缘故，最初上课期间并没有太多的实践机会。另外，按我个人的学习经验而言，语言课的学习更重要的是实践而绝非理论的记忆，因为每次实践中遇到问题，再去解决问题的过程让我对语法知识点的记忆尤为深刻。因此我希望老师能更加重视对这门课的实践教育，比如课程的平时布置一些实践的小任务，或是在 Github 上放一个小项目让大家一起维护或学习。然后对于语法知识的教学也没必要面面俱到，因为基本语法基本大家看看书就可以知道，然后在课堂上的时间更多的讲一些写 Python 程序的小技巧或是一些小项目的开发过程，我觉得这样会让我们的收益更多。