一些可能遇到的问题

1. 爬取页面返回奇怪的 status code?

当浏览者访问一个网页时,浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前,此网页所在的服务器会返回一个包含HTTP状态码(Status Code)的信息头(server header)用以响应浏览器的请求。

Status code 代表返回的 response 是否正确,通常返回的是 200 (OK)。如果一个 url 可以从浏览器打开,但是 request 返回的却是不同的内容,或者什么都没有返回。通常而言可以通过设置 request 的 header 来修正。

例如下图,直接向豆瓣的某个页面发送 request,返回的内容为空,status 为 418:

```
>>> url = r'https://movie.douban.com/subject/1292001'
>>> res = requests.get(url)
>>> res.text
>>> res.status_code
418
>>>
```

The HTTP 418 I'm a teapot client error response code indicates that the server refuses to brew coffee because it is a teapot. This error is a reference to Hyper Text Coffee Pot Control Protocol which was an April Fools' joke in 1998.

解决方案: 写一个最基本的 header, 加入 "User-Agent"字段(具体内容可以通过浏览器随便访问一个页面, 打开"开发者模式"(F12)的"网络"标签页查看 request header), 重新发送请求,问题解决。

```
X Headers Preview Response Cookies Timing
  set-cookie: M_WEIBOCN_PARAMS=1fid%3D102803%261uicode%3D20000174%26uicode%3D20000174; expires=Thu, 21-Nov-2019 12:4
  7:22 GMT; Max-Age=600; path=/; domain=.weibo.cn; HttpOnly
  ssl_node: ssl-009.mweibo.yf.intra.weibo.cn
  status: 200
  vary: Accept-Encoding
  x-log-uid: 6508823216
  x-powered-by: PHP/7.2.1
▼ Request Headers 🍃
  :authority: m.weibo.cn
  :method: GET
  :path: /
 sec-reton-site: same-origin
 upgrade-insecure-requests: 1
 user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108
 Safari/537.36
```

>>> header = ["User-Agent":"Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.16 (KHTML, like G >>> res = requests.get(url, headers=header) >>> res.status_code 200 >>> _

上述的"User-Agent"只是一个最简单的例子, header 里有许多字段, 在互联网通信过程中 request 经常通过 header 向服务器表明自己的身份。不同的错误 status code 指代的错误也不一样, 根据实际错误代码进行分析, 或许需要别的解决方案。

参考链接: https://www.cnblogs.com/lei0213/p/6957508.html

2. 页面总是重定向到登录界面?

服务器可以根据 header 中的 cookie 检查当前使用用户的登录情况。通常而言信息类网站(微博、豆瓣等)往往只对游客展示很少或不展示信息,这时就需要向服务器表明自己的登录信息。然而爬虫实验中并没有给 header 中加入 cookie, 所以服务器后端会返回一个登录的重定向页面。

解决方案: 在 header 的 cookie 字段加入自己的 cookie, 重新发送请求。自己的登录 cookie 可以先在浏览器登录一次, 然后访问一个任意页面, 然后仿照上例从"开发者模式"中获取 这次请求中带的 cookie, 加到爬虫的 header 里即可。

参考链接: https://www.douban.com/note/264976536/

(仿照链接里设置 header 的方式,这个链接使用的是比较早的库,发送 request 的地方使用实验任务中的 request 库即可)

3. 爬虫运行一会儿就不返回数据了?

这种情况很可能是服务器的反爬虫策略导致的。服务器一旦检测到某一个来源在短时间内发出了过多请求,就会将其判别为脚本来源,并且暂时禁用该 IP 或 cookie。

解决方案:考虑引入 time 模块的 time. sleep 函数,在爬取几个页面后等待 1-3 秒。

4. 爬取的页面只有模板, 想爬的数据没有渲染上来?

现在的网站很少有静态页面了,大多数页面都只提供一个通用的模板,当用户访问某个特定的对象(比如某个影片的介绍页面)时,JavaScript 代码会向后端 API 请求数据,当页面加载完成后再把数据渲染到页面上。但是基本的爬虫库只包含一个通信模块,获取到页面文本后无法继续执行 JS 代码,也无法渲染数据。

解决方案: 学习使用 Python 的 selenium 模块,结合无头浏览器 PhantomJS 或者 Google Chrome 对应的 selenium WebDriver,实现一个"基于浏览器"的爬虫。这种爬虫会以浏览器的方式实际加载数据,获取到的页面文本也是渲染后的结果,但资源消耗较大。

参考链接: https://selenium-python.readthedocs.io/

https://www.jianshu.com/p/dd848e40c7ad

如果还有其他问题:

- 1. 面向 StackOverflow 的编程: https://stackoverflow.com/
- 2. 面向 CSDN 的编程
- 3. 在课程群里@助教
- 4. 面向百度知道的编程(不推荐)