微博个人信息爬虫介绍文档

邹佳旭

2017年9月22日

写在前边

该文档用于项目参与的人员了解数据的爬取过程，会对爬虫代码进行介绍而不会去教读者python或爬虫的基本知识。如果想了解python建议去拜访“廖雪峰的官方网站”，想了解scrapy最好直接读一部分scrapy1.3的英文官方文档。

该爬虫使用scrapy框架，用python2.7编写，没有实现分布式，数据来源是新浪微博wap端（https://weibo.cn/），爬取速度是单机每秒一条，使用了cookie池、user-agent池，没有使用ip代理。

该爬虫优点是稳定，在实际运行中没有出现403、414等IP访问异常或帐号异常。缺点是速度较慢，不支持分布式，失效uid没有进行存储，没有连接数据库。

速度较慢可以使用ip代理来解决，可以选择爬免费的ip代理网站或者直接买接口；分布式直接改写即可；失效uid只需创建一个本地文件，在检测到失效uid的时候打开文件追加失效的uid；连接数据库直接改写pipelines即可。

该爬虫使用的python2.7内容不多，读者掌握字符串的分词、print方法、if语句、for语句、try语句、class概念、函数概念即可。

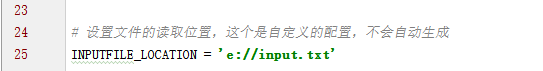
脚本代码中有一部分注释掉的代码，是我认为有开发价值的代码，并没有删。需要注意的是，我的ip代理写在了spider中，导致请求失效重新请求的话还会使用原有的失效的ip代理，最终导致程序卡死。解决办法是将其写入middlewares.py中。不想优化源代码的话这段话忽略即可。

运行该脚本需要的环境有python2.7，scrapy，pip，requests，lxml，windows需要额外安装win32，ubuntu系统只需要安装pip，scrapy即可。windows下的安装文件我都上传到了实验室的官网上，ubuntu系统则建议直接使用命令来进行安装。

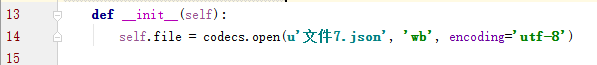
简单描述一下安装过程，windows下下载官网上的安装包（*python2.x+python3.x+pycharm安装包.zip*和p*ywin32-221.win-and64-py2.7.exe*），解压zip，安装其中的*python-2.7.12.amd64.msi*，如果电脑里没有编译器可以安装*pycharm-edu-2.0.3.exe*。然后双击pywin32进行安装。安装python2.7过程中会自动安装pip，之后只需要开启控制台（windows + r），依次输入*pip install Scrapy*、*pip install Requests*、*pip install lxml*就可以自动安装组件，全部安装成功后即可正常运行。ubuntu下，只需分别输入*sudo apt install Scrapy*、*sudo apt install pip*即可。

使用方法

使用方法非常简单，该文件需要一个存放有用户uid的输入文件，首先在settings.py文件中设置输入文件的位置：



然后再在pipelines.py文件中设置爬好的数据的存放的位置：

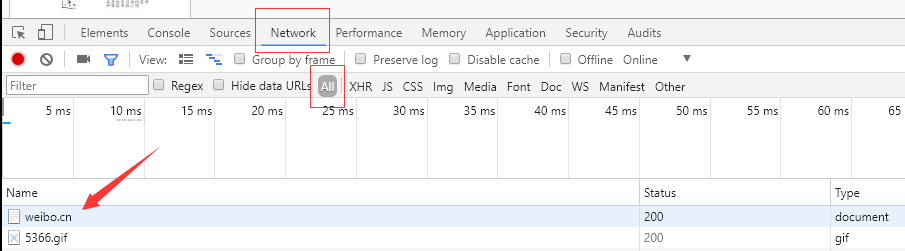


第一个参数为存放的文件的路径，只填写文件名则默认生成在项目的根目录下（scrapy.cfg）文件所在的目录下。后缀名可以为*csv*、*json*、t*xt*，建议使用json；第二个参数为读写方式，wb表示以二进制文件的方式进行读写，如果没有目标文件则创建一个新的空文件，如果有目标文件则以一个空文件覆盖目标文件，如果不想覆盖只想追加可以改成*ab*，具体内容请问百度；第三个参数为字符集。

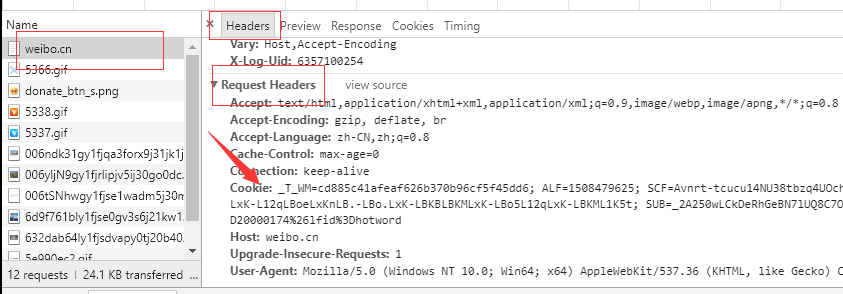
然后需要手动去登录帐号去浏览器上采集cookie，将采集到的cookie手动填写到cookies.py的cookieList中。获取cookie的方法请见多图如下：

登录网站*https://passport.weibo.cn/signin/login*进行用户登录，如果是买的小号的话登录的时候会要求输入图形验证码，请用智商解决；

登录成功之后会直接进入移动端你的个人首页，这时不要做其他操作，按*F12*打开开发者模式，输入网址*https://weibo.cn/*，回车进入网站；

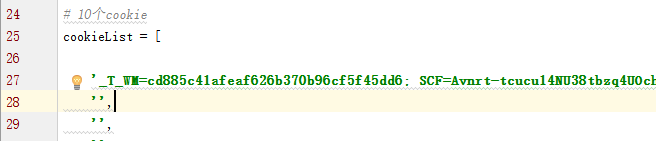


在开发者工具中点击上图设置就能找到*weibo.cn*这个*document*，单击它；



如图所示的*Cookies*中的内容就是我们需要采集的，从*\_T\_...*一直到*...word*这个范围。

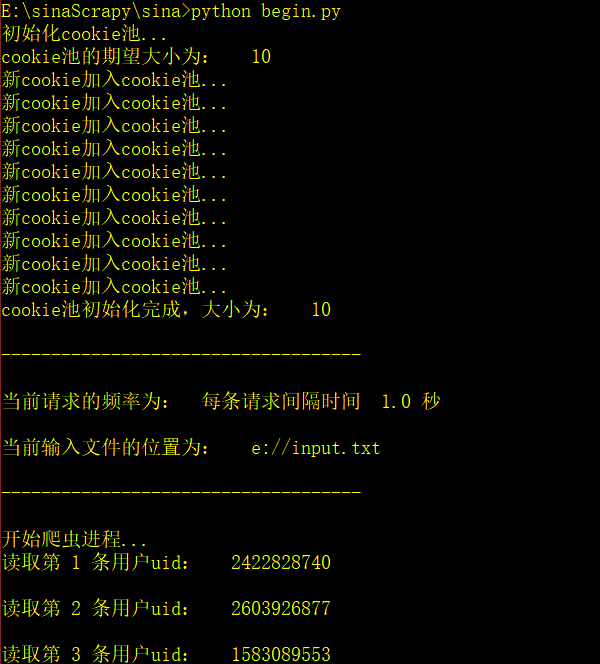
将采集到的cookie复制到*cookieList*中，建议单机运行频次为1的话要至少十个cookie才能保证不被帐号异常。



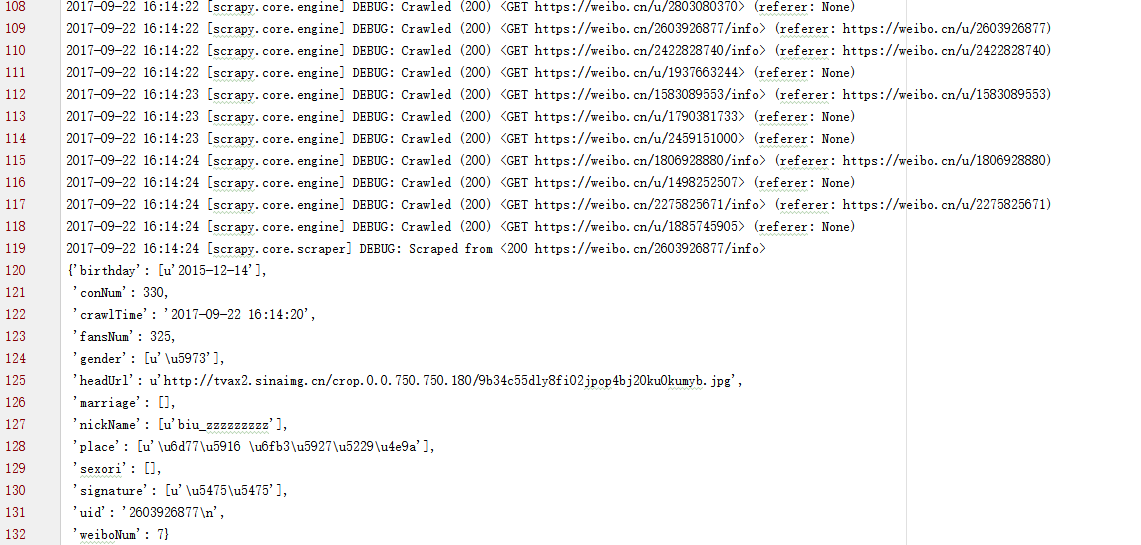
上边准备工作就做完了，直接开始运行。开启控制台进入项目根目录下，输入命令：

*python begin.py*

开始运行。运行成功如下图：



相关的Debug信息在根目录下的*increment\_comment.log*下有记载，如果其中出现了如下图所示的信息则证明正常开始爬取了。*... DEBUG：Crawled (200) <GET ...>* 代表正常访问，没有问题，这里面的*200*表示正常访问，*403*和*414*是IP地址出现了异常被服务器拒绝了访问，可以通过IP代理或者降低速度来解决，*302*重定向到*https://pass....*的网站代表账户出现了问题，增加*cookieList*的容量或者降低速度即可解决。值得一提的是，如果使用IP代理会出现*10060*、*10061*的状态码，这些是代理IP失效造成的，解决方法是将IP代理写在*middlewares*中，重新尝试请求的时候更换一个好用的IP代理即可。



之所以将DEBUG存到log文件中，一是这样可以长期保存日志，方便事后处理，二是可以将日志和打印的提示区分开。

在LINUX下可以使用命令：

*tail -f increment\_comment.log*

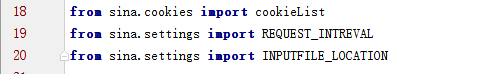
来动态的查看日志。

代码简析

这个脚本我已经做了非常详细的注释，所以这里我只简单介绍一下关键的点。具体代码的运行过程根据代码分析即可。

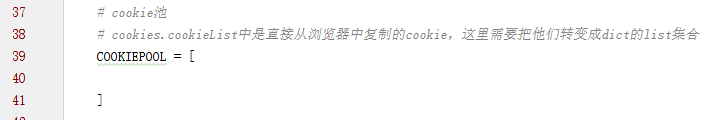
infoSpider.py：

infoSpider.py是唯一的一个spider，他其中的一些动态的设置

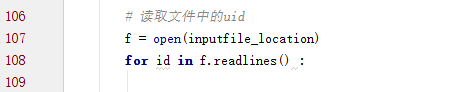


都是直接从其它文件中导入进来的。

开始运行脚本时脚本会首先从*cookies.cookieList*中导入*cookie*，转存到内存中的*COOKIEPOOL*中。



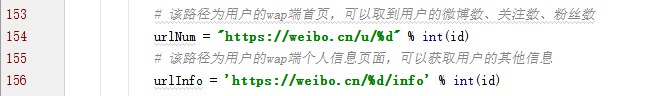
需要注意的是新浪的cookie中有一个机制会检查cookie中的*ALF*属性和*SSOLoginStates*属性的值是否为*int*类型，所以在转存到内存中的时候要进行特殊处理。代码在*infoSpider.py*文件的*72~96*行。

设置url的方式是在每次循环的时候从文件中读出一个uid，然后拼凑成新的url。 

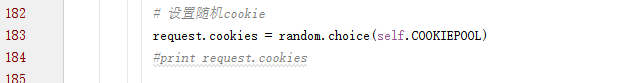
在每次读取一条uid之后都会进行一个时间固定的线程挂起，换句话说在每发出一个请求之后都会等待一个固定的时间才进行下一次请求，这个注释请忽略



这里拼凑出了两条url，一条指向用户的主页，一条指向用户的个人信息页面，两个我们都需要访问，因为我们需要获取不同的信息



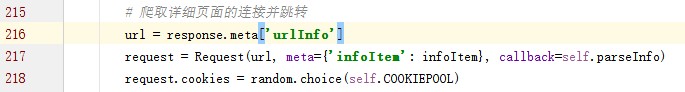
这一段是给请求设置一个随机的cookie，没有cookie在请求中scrapy无法成功进入指定url的页面（因为无cookie相当于未登录，未登录是不能访问他人的首页和详细信息页面的，会被interceptor拦截跳转到登录页面）



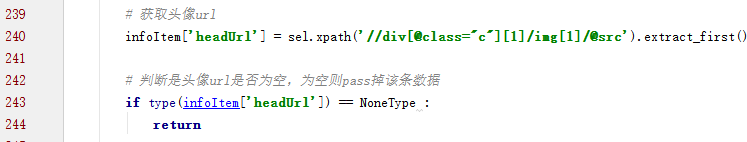
这里将用户的uid和刚拼凑出来的详细信息的url放在request的*meta*中，这样这次请求返回的*response*中就会存在我们放入的meta属性。之所以这么做，是因为我们需要uid在返回时调用的*parseNum*方法中封装，需要url进行下一次跳转。



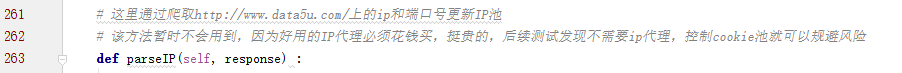
在parseNum方法中，我们将刚新建的并填入了部分数据的infoItem对象存入了meta中，是为了可以在*parseInfo*中将所有信息都存到同一个对象里。不这样在的话，就需要两个*item*分别存放两部分数据了。



在新浪微博中有一部分用户的他的主页可以访问到，但是详细页面是一片空白。对于一个正常用户来说，头像的url是肯定会有的，所以如果获取不到头像的url则证明该用户有问题，他的部分信息就算存起来也是冗余数据。所以我根据头像的url是否存在来决定是否保存当前uid指向用户的详细信息。



最后再解释一下代码中大段的注释和最后一个方法：



的功能，他们的功能是爬取按照固定的时间间隔，爬取西刺代理（*http://www.data5u.com/*）上公布的免费IP代理的最新的十条代理，并进行IP代理是否可用的检验，如果可用加入到内存中的IPPOOL中，在每次发出请求之前从IPPOOL中随机抽取一个代理地址并用于请求中。

这种设计有很大的缺陷，首先就是之前提到的10060和10061会导致脚本卡死，其次没有开新线程专门计时而是按照uid读取的条数来控制更新的时间会导致更新时间不稳定，失效代理的数目在某个时间段会明显增多。

cookies.py：

这个py文件是我自己定义的，它里面的cookieList用于保存收集到的cookie。

items.py：

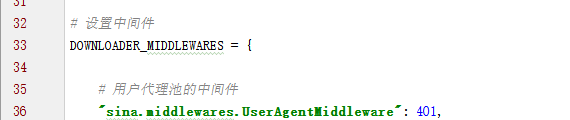
这个文件中定义了存放的数据对象的格式。我们获取的用户信息的类型在里面有详细的记述。

目前发现的问题是，用户的生日并不是完全规范的，有的用户生日一栏上是自己的星座，有的没有年份。我在爬数据的时候并没有进行特殊处理，一并保存了下来。

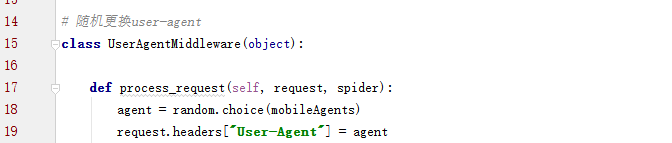
middlewares.py：

这个就是爬虫的中间件文件。中间件文件中可以有很多的类，只要这个类在settings.py中进行了设置那么它就是一个中间件。

目前使用的中间件只有一个用户代理的中间件。如果以后方便还是希望自己或者其他人把IP代理和cookie的随机选择加入到中间件中，而不是写在spider中，那样太蠢了。



随机更换*user-agent*是很有必要的，服务器检查访问的请求是不是爬虫第一个使用的办法就是检查请求的user-agent，我这里设置了user-agent池，在*user\_agents.py*文件中，这个中间件每次会从user-agent池中随机抽取一个user-agent来伪装自己。



pipelines.py：

这个文件用于定义数据的存储方式。我设置的存储方式是存储到本地的json文件中。实际上还可以设置存入mongodb、mysql等各类数据库中。

存到json文件中主要是为了方便备份和转储。数据量大的时候并不适用文件存储。因为文件稍大io读写就会很艰难。

settings.py：

下图所示的设置一定要设置成False，只有一些大型的搜索网站的爬虫才回设置成True



具体每条设置我都写了注释，大家直接看代码就行了。