爬虫框架入门讲解

思必驰实习生 杨之旭

一、 前言

本文是对公司爬虫框架的一个入门讲解，帮助开发人员快速了解、上手。其中包括部分爬虫基础知识、底层代码的介绍、开发常见问题、新浪微博爬虫讲解等几部分，希望我的总结能够帮助你更顺利地开始工作。如果有不详尽或有错误的地方，欢迎继续编辑本文档。

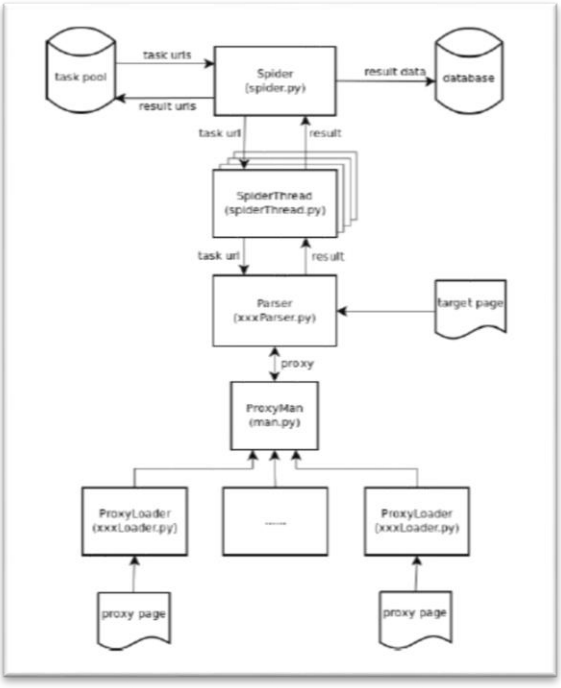
二、 爬虫基础知识

1. python 程序设计
2. linux 系统基本操作
3. 计算机网络基础，了解 html 脚本结构，http 请求，cookie 原理，python urllib，urllib2，cookielib 等相关标准库，并会使用 firebug 等工具观察数据包
4. 正则表达式，python re 标准库，以及用 etree 解析 xml 文本
5. mongodb 数据库基本操作

上述内容很容易找到大量资料，在此就不赘述了。

三、 爬虫框架介绍

1. 总体介绍：该框架非常完善地实现了网络爬虫的各项功能，实现了网络代理、多线程、网页去重、数据库存储等功能，并且具有良好的可拓展性，后续开发人员只需做少量开发即可完成对指定网页的爬虫。
2. 代码结构：



爬虫框架实现

A. Spider：入口类。Spider 一方面从 task pool 中取出初始 url 和所有目标 url，分配给各个线程 SpiderThread，并回收各个线程爬取的数据存入 database。运行爬虫时运行 tutorial 文件夹中的 Spider.py 文件，是整个程序的入口。入口 url 在 Spider.cfg 中指定。注意在调用 start()函数之前需要用 appendParser()函数指定会用到的所有 Parser。

B. SpiderThread：线程类。实现多线程爬取。无需修改。

C. Parser：页面解析器的基类。开发人员的主要工作就是实现该基类针对具体页面的子类。在 start() 函数中主要有四个步骤，run(url)，请求 url 页面，赋值给 page 成员；postProcess()用于动态请求，如页面中有 http 请求，可在这个函数中请求，如不需要就不用重写，基类中是空函数；filteUrl() 从页面中提取出具有要进一步爬取的 url，用字典类型返回，键是 url，Spider 会将这些 url 做进一步分配、爬取，基类返回空字典；filtePage() 爬取页面内容，返回字符串，Spider 会将这些字符串存入数据库，基类返回空字符串。

   如有动态请求，可调用 reqPage 函数，无需自己编写请求，该函数会自动加载 cookie，并加入请求的 header。

D. ProxyMan：代理服务器类，无需修改。

E. 数据库：本爬虫采用 mongodb 数据库，表名在 Spider.cfg 中有定义，表中有 pages, url_doing, url_done, url_fail, url_skip 四列。可以通过终端进入数据库进行查看。

四、 常见问题

1. 容易自己多写很多东西最后发现框架中已经提供了：在开始实现之前，一定先仔细阅读框架原码，主要是 Spider，Parser 两个类的实现，明白框架已经提供了哪些功能、哪些函数。如发出 HTTP 请求；用 filtUrl 返回 url 然后自动就会有 Parser 处理这些页面；请求间隔设定等等。

2. 页面自动加载

   有一些页面不是通过一次 get 请求就获得所有的页面内容，而是通过动态加载的方式获得的，典型的例子是当页面的 scroll 到底端时自动加载下一页。此时就需要用 firebug 观察发送出请求的结构和内容，往往是添加了参数的 GET 请求，这些请求的参数都反映在请求的 URL 上，只要构造出相同的 URL 发出相同的请求即可完成，收到的内容就是动态加载的内容。

3. cookie 相关和模拟登录

   在 Spider 类中有 cookie_fn 保存 cookie 文件路径和 cookieFlag 两个相关成员，cookie 路径是在 Spider.cfg 中指定的，cookieFlag 为 true 表示需要保存 cookie。

   在模拟登录过程中，先观察在登录过程中发送的 POST 请求，如果是未

加密的就非常简单，直接仿造即可，如 m.weibo.cn，但如果是加密过的如 weibo.com 就需要知道其加密方式。在发送 POST 请求得到成功登录的返回后，将 Parser.cookie 保存，之后的请求都不需要开发人员再添加 cookie 了，在底层中都已实现。在登录过程中常有重定向情况出现，只需记录下来完全模拟就好，直到返回了内容页为止，此时就已经完成了模拟登录过程。

在观察网站设置、发送 cookie 时，一定做到每一个 cookie 都知道是什么地方设置的，在什么地方起作用，生存期如何。网站不会随随便便设置一个没有用的 cookie，但是不一定每一个 cookie 对于检测爬虫都有意义。如反爬虫机制较弱的网站，即便是完全不设置 cookie 也能够返回正常页面，但对于新浪微博，可能一个 cookie 不正确都无法获得正确的结果。对于这样的网站就需要非常细致的观察。

部分网站有反爬虫机制，如新浪微博，要时刻关注自己程序运行的状态，如果被屏蔽及时调整方案。一般来说网站反爬虫的基本手段有判断 IP、判断账号等，底层已经实现了使用代理服务器所以无法通过 IP 来判断是爬虫，但是还是有可能被封号的。此时可以采用多个账号，并控制爬虫速度，可以一定程度上避免反爬虫。不过魔高一尺，道高一丈，有兴趣可以查阅相关资料，爬虫 VS 反爬虫的斗争方法非常丰富。

4. 两种定位内容的方法

   一种方法是通过 etree 类提供的方法，利用标签及其属性定位内容，这种方法适合网页标签结构明确，方便使用 firebug 确定页面元素和代码对应关系的网页；另一种方法就是是直接利用返回的文本利用正则表达式去判断，比如有些 H5 网页返回页面中有很多脚本，部分内容没有直接对应的标签；或者动态加载的页面与在浏览器中用 firebug 看到的 html 结构不同，都可以直接用正则表达式进行匹配。

5. 编码

   部分网页的中文使用 unicode 字符集，形如\u601d\u5fc5\u9a70=>思必驰，可以使用 decode('unicode-escape')进行解码即可。

6. 分析页面结构

   爬取一个网页肯定先从分析其页面结构入手，个人认为如果能避免登录时最好的，毕竟简单而且无封号之虞；分别设置哪几级 Parser，分别爬取哪些内容，就需要派生 Parser 类去完成。如知乎可以只涉及一个页面解析器就可以爬完全网，而微博就需要登录、话题、微博页等等远多于 1 个 Parser 的页面解析器。根据网页结构来确定。

7. 关于请求

   常用的 http 有 GET 和 POST 两种，GET 请求中又分为需要刷新页面的和不用刷新页面的 AJAX 请求，两者没有本质区别，只是在参数上有不

同。而 POST 请求与 GET 请求的区别就是 POST 有 body 来传递数据。所以说进行 GET 请求就是在构造 url，把所有的参数按照实际的情况排列好，header 由底层代码自动填充即可；POST 则需要按照实际情况填写好 body 的内容，注意部分用户登录的过程 header 中的一些参数也必须与实际情况一致否则会被拒绝。

五、 新浪微博移动端（m.weibo.cn）爬虫全攻略

新浪微博的爬虫比较复杂，反映在如下几个方面：一是必须使用账户登录，并且对于部分账户需要使用验证码；二是微博反爬虫机制很丰富，对于 IP、账户等不同层次上都有检测，稍不留意就会得到一个 403；三是微博网页复杂，动态请求多，cookie 多且复杂，模拟更加困难。不过在近两周的学习、攻关后，新浪微博爬虫也是圆满完成了。

1) 网站架构分析

新浪微博网页结构分为三层，第一层是热门话题列表，第二层是热门话题下的微博列表，第三层是每一条微博的内容和对应的评论。于是需要写三个不同的页面解析器对应不同层次的网页进行爬取。前两层只需要将网页上需要的 url 爬取下来作为进一步爬虫的目标 url，第三层只需要将微博内容以及相关有价值的信息爬虫下来，并保存在数据库中即可。

微博中每一条微博都是从属于某个相关主题下的，对主题标签也的爬虫能够最全面地得到各门类的微博。该页上的每一个主题具有对应的 url，而这些 url 正是下一级爬虫的入口。

进入单个主题的入口 url 后就是微博列表页，上面该主题下的一系列微博，如果只取与微博内容相关而与评论无关的内容的话，爬取该页上的内容就可以了，不过在该页上每条微博的评论都是一个动态请求，且参数较为复杂，可在列表页上找到每条微博的入口 url，通过进入单个微博的页面对于微博内容和评论内容进行爬虫。该页面通过 GET 请求返回微博相关内容，再通过一个 AJAX 请求返回和评论相关内容，并且当右侧 scrollbar 滑动到最低端时自动发送下一个 AJAX 请求加载下一页，如需爬虫全部评论则完全模拟请求即可。由于很多微博的评论过多可能有几十页并且大部分评论没有价值，故为了加快爬虫速度，减少发送请求的数量，最终我只选择加载第一页评论。其余评论不再请求。

2) 新浪微博登录流程详解

新浪微博分为电脑版(weibo.com)和移动版(m.weibo.cn)，电脑版登录的 POST 请求会将用户名密码进行加密，并有“加盐”等操作，模拟登录相对困难，目前网上对于几年前的微博模拟登录有相关破解资料，但对于当前的登录过程没有详细介绍。而移动版相对比较简单，POST 请求中的用户名和密码都是明文的，而且也没有附加的干扰，故选择了使用移动版作为目标

网站。

A. GET http://m.weibo.cn/

请求到登录首页，接受 cookie _T_WM，该 cookie 长时间有效，是第一个必要 cookie。

B. GET https://login.sina.com.cn/sso/prelogin.php

当用户名失去焦点时发送该请求，作用一方面是判断该账户是否需要验证码，另一方面也用于判断爬虫。该请求有较多参数都是之前的请求中未出现过的，而是通过 js 脚本添加进去的。通过阅读网站相关 js 脚本，找到每个参数是如何生成的。

callback 参数是当前时间和一个 0~100000 随机数的和；su 参数为用户名的 base64 编码；checkpin 用于登录方式判断，可常取 1；entry 可常取 mweibo。对应的 js 脚本地址分别为：

https://passport.weibo.cn/js/signin/req.js?v=20160120 和 https://passport.weibo.cn/js/signin/weibologin.js?v=20160120。这两个 js 文件都是 2016.1.20 最后进行编辑的。这也是我第一次接触 javascrpt 代码，经过简单的入门学习后比较容易读懂。值得一提的是，这个请求也出现了大量的 cookie，同样是通过 js 脚本添加的，不过这些 cookie 在之后的请求中并没有出现，并且不添加也不影响进行预登陆，所以这一步不添加这些 cookie 也没有问题。

C. GET https://passport.weibo.cn/captcha/image

获取验证码图片。请求返回内容是图片的 base64 编码，和一个图片的 id 序号。在验证时只需要把用户输入的验证码和这个 id 序号同时发送回去，服务器就可以验证是否输入正确。新浪微博使用的验证码是五位定长数字+字母的验证码，并且含有正弦线条干扰。这种验证码在识别难度上属于中等难度，用现成的 OCR 代码如 python 的库 pyocr 是无法识别的，出于实现的经济性，我先采用了保存至本地，人工识别输入的方法；在后来实现了多账户轮流功能后，每次登陆需要输入十几个验证码就比较麻烦了，在尝试了几个自动识别接口发现识别率都较低的情况下采用了调用人工打码平台云速打码：http://www.ysdm.net/，发现识别率几乎为 100%，且打码速度很快，价格低廉，接口调用容易，满足要求。于是就调用该接口实现了验证码的自动识别。验证码并不是所有的账户都要输入，新浪微博对于所有的账户都有一个评级，如果出现了非法操作，如异地登录、被判定为爬虫等等，就会降低这个评级，到达一个阈值之下就会要求用户在登录过程中输入验证码。

D. POST https://passport.weibo.cn/sso/login

该请求需要在 POST 内容中返回明文的用户名、密码、验证码 id、验证码识别结果，以及为常量的 entry、ec、pagerefer、savestate，以及为空

的一些参数。将这些参数全部 POST 发送后，返回内容会显示是否登录成功。最初我认为用户名、密码全部用明文发送非常不安全，不过仔细一想即便是加密，方法也是通过 js 脚本事先传输好的，所以说即便是加密对于恶意攻击的黑客来说也是毫无意义，所以说诸如此类的公开方法的加密是“防君子不防小人”。而 POST 请求返回了几个很重要的 cookie，分别为：SUB, SUHB, SCF, SSOLoginState, ALF，这几个 cookie 几乎之后的每次请求都要发送。

需要注意的是，虽然很多网站对于 GET 请求或 POST 请求中的 headers 参数要求并不严格，但是在新浪微博登录中其中的几个参数如果不准确就会导致登录被拒绝，如 referer 等参数，所以在整个爬虫过程中为了防止意外出现，就不要怕麻烦把所有的请求的各种参数都模拟好，明确知道每个参数的含义就可以省去很多调试的时间。

E. GET https://passport.weibo.com/sso/crossdomain?

在发送了 POST 请求后，还需要发送三次 crossdomain 请求，对于网站的作用估计是通过登录 m.weibo.cn 自动获得 www.weibo.com, www.sina.com.cn, www.weibo.cn 等几个旗下网站的登录权限而不用重复登录，故称为 crossdomain，登录跨越到不同主机。从发送的请求内容来看，也是向上述三个网站发送信息以验证登录，具体请求的网址是在 js 脚本中写明，包括 ticket 等部分参数，另外还需要自行添加一个为常量的参数 savestate，以及一个随机参数 callback，callback 参数是由当前时间和 0~100000 的随机数的和组成。

F. GET http://m.weibo.cn/

此时注意一定要再次请求主页，原因是需要取到一个至关重要的 cookie: M_WEIBOCN_PARAMS，也正是这个 cookie 在每次请求之后它的生存期就会延长一些，一段事情没有请求该 cookie 过期之后就会要求重新登录。

至此，登录过程中的全部 cookie 都已经请求到，全部对于后续请求有影响的请求也都已发送。这个过程不可谓不复杂，不过对于浏览器来说只是按照程序执行非常容易不过对于程序员要来破解这个过程并且利用程序将其复制下来还是需要一定的精力和功夫的。在此强调几点：

A. 重视 js 脚本的关键性：爬虫刚入门的程序员对于很容易查看到的 html 程序和相关的 http 请求还是非常熟悉的，因为这些内容通过浏览器的一些工具非常容易能够知道是什么意思。然而，现在大部分网站最基本的语言就是 html 和 javascript，要想让你爬虫的网站所有需要知道的信息都在 html 相关的 http 请求中直接返回那就是太天真了。很多信息是隐藏在 js 脚本中的，并且变量名等信息也都是改变了的，所以必须懂

得如何从 js 中找到必要的一些信息。如果发现某一条请求中发送了之前根本出现过的参数或 cookie，很有可能就是在 js 脚本中直接写的。

B. 如何确定什么请求可以不发：在上述解决方案中我实际上略过了不少请求，让我知道可以不去发送这些请求的原因如下：一是我明确这条请求做了我完全不需要的事情；二是这条请求得到的 cookie 之后完全没有用到，并且对于之后的请求也完全没有事实上的影响，那么就可以不去模拟这条请求了。

C. 如何方便的使用接受和发送 cookie：python 的 cookielib 标准库中提供了丰富的方法，对于这种多个 cookie 请求非常复杂的过程，我建议使用 session，只需要在不同请求之间使用同一个 session，底层代码自动实现了接受头中的所有 cookie，并且将现在还没有过期的 cookie 全部发送出去。在这个过程结束后，只需要把 session 中的 cookie 保存至文件中，就非常方便之后的再继续使用这些 cookie。非常方便，减少了很多不必要的思考过程。

3) 类构建

一条微博相关的有用信息有：题目、描述、评论数、分享数、点赞数、评论内容、时间、博主姓名、博主描述等。一个回答相关的有用信息有：答主姓名、答主描述、回答内容、赞同数、回答时间、评论数、分享数等。据此构建出两个类分别对应微博和对应的评论。

4) 新浪微博反爬虫机制简介

新浪微博具有很丰富的反爬虫机制，是很好的观察爬虫与反爬虫两者道高一尺，魔高一丈的斗争的机会。在实际工作中，我只花了四分之一的时间就已经将基础部分完成，可以将想要的内容从新浪微博上爬取下来，但是剩下的四分之三的时间都在与新浪微博的反爬虫机制作斗争。在这个过程中对于网站建设与保护等内容有了更深刻的理解。

新浪微博主要的反爬虫机制分为两方面，一方面是针对于 IP 的，另一方面是对于账户的，而对于这两类的对象，分别又有每天浏览的微博总数、一段时间内浏览的微博总数、浏览速度上限等方面的要求和限制。这种级别的防护在常见的网站中属于安全级别较高的，主要是因为新浪微博使用客户量大，关注度高，具有海量的、与热点时时关联的数据，也是因为新浪本身也在经营这些数据，开放了受限的 API 接口，同时也在用数据作为和其他公司和合作的砝码，自然不希望有人能够很容易的拿到它的数据进行商业用途。不过防护级别还可以大幅提高，没有这么做的主要原因还是因为成本。

由于公司的爬虫框架是完成了使用代理功能的，每一个请求都会在一个庞大的代理池当中选择一个作为请求的 IP 地址，就避免了 IP 被封的情况；接下来只要主要账户的问题就好。在新浪微博反爬虫机制的限制下，最优秀

的结果就是单个以略低于上限的速度爬完每天的额度，并使用大量账户进行爬虫。这样能够保证信息量。而完成这个目标的先决条件就是试探出瞬时速度上限。为此我进行了一系列的实验，并记录、分析实验数据，逐渐收敛到反爬虫机制设定的上限。

能够调节瞬时速度的参数有如下几个：线程数、单次从列表中取出 url 的个数、每次请求前等待的时间，出于迷惑服务器的考虑，每次请求前等待的时间分为两部分是在一个范围内的随机数，这样就避免了在短时间内大量请求同时并发被服务前发现。（有幸的是，新浪微博并没有针对 IP 与账户关联的检查，否则就会发现这些账户一会儿出现在法国，一会儿出现在内蒙古，几分钟走遍了世界各地，服务器应该很快就会被发现是使用了代理服务器）。

由于对于瞬时速度的检查与账户相关联的，所以使用多个账户轮流爬虫是有利于加快整体速度的，考虑到 http 协议是一个无状态的协议，所以对于任何一个请求，使用什么账户就是没有限制的，所以在实际中就是使用 18 个账户轮流（正式爬虫时会使用更多的账号），每发送一次请求就换一个 cookie，这样保证了每个账号在每两次请求之间的时间间隔都是在某一特定速度下最长的。

5) 日志阅读与研究

对于复杂的多线程程序调试，有完善的日志记录还是非常方便的，有了日志可以非常全面仔细地看到程序的运行状态，并可以长期保留进行研究分析。虽然在程序中设计出严密的日志是要花费一定时间的，不过如果没有日志很多时候都是在瞎猜。

最后，祝你爬虫愉快！