反爬技术

我在设计，改进爬虫的时候感觉 难度最大的就是与反扒技术的博弈过程，反扒技术也是贯穿全文，在这里就不再冗余在用到的技术都相应的解释介绍比如伪装http的报文头修改User-agent爬取移动版Web,用多线程做的Cookies池,IP地址代理池,买微博的账号,破解JavaScript环境,打验证码等等!

4.3.1**企业中常见的反爬技术手段**

那些厂商为何确保自己的利益是怎么样进行反爬的.实话说，与对方的反爬措施做斗争可以称得上开发过程中为数不多的趣味了，同样也可以说是一种折磨了.，你会见到各种奇妙无比，匪夷所思，令人拍案叫绝的反爬手段。同时你还得绞尽脑汁的去绕过这些.在前面的章节中，我们提到爬虫是我们获取互联网信息的一种重要手段,那么有矛必有盾.一方面是千方百计想要爬取数据的你，另一方面是誓死捍卫自己数据的守护者.一场较量再所难免.......

回到正题，目前市面上的反爬措施很多，也有很多奇妙的，奇特的，恶毒的都有.我们从本质出发，目前市面的大概可以分成这么几个大类.

* 根据访问日志，流量，爬取行为以及各种相关算法来判定该行为是不是爬虫所为，从而触发黑名单或者蜜罐系统。例如最为经典的封禁IP和封禁账号.这一类用途最广，防范水平要看其判断策略写的如何.
* 利用ajax技术，通过js脚本实现动态渲染的后台传输类.比如说通过JSON实现前端与后端通信.这种有人也许会说这也能算反爬措施?但这种至少你读取网页源码时是得不到数据的，只能通过抓包等方式解决.这一类基本在需要交互的网站中运用的许多.抓包一般可以用burpsuite,fiddle这类软件进行抓包.
* 通过浏览器环境，在本地运算js，最终得到所需的数据的本地js类.因为现在浏览器都有内核从而实现了自带javascript环境.在本地环境中即时运算从而得到数据.从我的经验来看，这一类是很难对付的，而且即使有解决方案，但效率很低而且爬虫程序不容易写.
* 各种找不到形容词，只能用奇思妙想来形容的反爬类.一些通过图像混淆，页面无限循环，给爬虫投毒等等行为可以归入此类.

那么我们开始了解常用的几个反爬手段.首先是封禁类.我们经常会遇到爬取过度而导致IP被封或者是账号被封。你知道是怎么一回事么? 在这里，我们将详细的了解下我们在什么情况下会遭到此类惩罚。同时，一些常用的判断规则又是什么？一般来说，我们如果触发了反爬系统所设定的条件的话，就会引发封禁.该手段一般设定的标准有这几种：

* 是否携带有一个正常访问下所有的header头，cookies和正常的User-Agent.针对与这种情况，我们只要通过模仿浏览器发送的包，将我们的爬虫所发送的HTTP请求上都挂上和浏览器发送的请求一样的header头和cookie就行。此类标准由于技术难度低，且反爬简易，现在一般不采用此类.
* 根据行为是否表现的像一次人为的操作，诸如不分昼夜的机械化的发送请求，每次只读取特定内容，相隔固定的秒数分秒不差的，都可以认为是一个robot在操作.该类的话，需要我们模拟出一个正常的人类读取网页时的行为，比如使用乱序时间，将爬虫每轮间隔的爬取时间使用随机时间等待这些方法，这么做的话，比较不容易触发规则.
* 在网页源代码中设置一些正常访问看不到的标签，然而爬虫会触发到的标签，从而达到封禁的效果。如果在试验过程中发现这些措施的话，可以通过精准定位的方式（后面所讲的Xpath,selector）来避开那些有毒的代码块.

封禁的标准千千万万，阈值也是每个网站都不同.我们应当在正式爬取之前，先写几个测试用意的爬虫，用来试探该网站的反爬策略，从而决定我们的反反爬措施. 当然还有最终的解决办法，那么就是看封禁的是什么。大多数情况下，封禁的是IP地址，那么这时候我们可以去通过代理IP来无视掉封禁IP的情况.代理IP也就是代理网络用户去取得网络信息.我们如果通过代理IP来获取信息的话，那么就可以无视掉依靠IP封禁的反爬策略.一般基本上主流的HTTP库都提供代理功能.但有利必有弊，通过代理IP进行中转固然可以无视封禁，但代理节点的稳定性和速度就直接影响到整体爬虫的效率.如果该代理节点时常掉线，延时居高不下，那么从最终效果上来看就和不挂代理依靠低速运行来躲避阈值的爬虫一样.如何维持一个延迟低，可用率高的代理IP池不在本文叙述范围内，请自行寻找资料。

总而言之，封禁类中的如何判断是否是爬虫是反爬系统的核心关键所在，确定了是爬虫之后，那么就不光是封禁，可能会有各种手段惩罚或者戏弄你.比如后面讲的投毒，无限循环，伪装404页面等待......

这里只介绍怎么打验证码对JavaScript反爬的技巧与json的使用破解AJAX两个小技巧，其他部分因为版面等原因都在论文其他部分讲解这里就不再赘言。

**4.3.2 对抗AJAX技术**

利用Web开发应用程序因其易于部署、节省成本的特点而逐渐成为技术的主流，浏览器将是未来唯一需要的客户端。J2EE Web技术己经从一个静态的内容提供环境迅速发展为一个在Intranet和Internet上运行的动态业务应用程序的靠平台。过去几年中在后端(后端机制，backend mechanics)投入了太多的时间，而在Web用户界面对用户交互和响应灵敏方面却投入不足。用户在强大的业务逻辑背后，还忍受着“提交-响应-等待-刷新”的同步运行机制，不管在页面呈现上的变化多么小，都需要耐心地等待服务器将整个页面重新发送给客户端。为了构建更为动态和响应更灵敏的Web应用程序，实现浏览器和服务器的异步并行处理，减轻服务器端负担，Jesse James Garrett提出了AJAX(Asynchronous JavaScript and XML)这一新的概念。AJAX是一种创建交互式网页应用的网页开发技术，由多种技术组合而成，包括:

基于XHTML和CSS标准的表示;

使用Document Object Model进行动态显示和交互;

使用XMLHttpRequest对象与服务器进行异步通信;

使用JavaScript绑定一切。

与传统的Web开发不同，AJAX并不是以一种基于静态页面的方式来看待Web应用的。从AJAX的角度看来，Web应用应由少量的页面组成，其中每个页面是一个更小型的AJAX应用，每个页面包括有一些使用JavaScript开发的AJAX组件。这些组件使用XMLHttpRequest对象以异步的方式与服务器通信，从服务器获取需要的数据后使用DOM API来更新页面内容。



如图，AJAX的工作原理相当于在用户和服务器之间加了一个中间层一一AJAX引擎，使用户操作与服务器响应异步化一一并不是所有的用户请求都提交给服务器。一些数据验证和处理由AJAX来做而不必交给服务器处理，只有确定需要从服务器读取新数据时再由AJAX引擎向服务器提交请求。在使用AJAX引擎后，用户从感觉上几乎所有的操作都会很快响应，并且没有页面重载(白屏)的AJAX引擎实际上是一个比较复杂的JavaScript应用程序，用来处理用户请求然后根据需要动态读写服务器和更改DOM内容。以前为了使网页能无缝化重构，

也就是在页面己经下载完毕后改变页面内容，开发人员一直通过JavaScript和DOM来实现。但是要使网页真正动态起来，不仅需要内部的互动，还需要从外部获取数据，在XMLHttpRequest对象出现以前是让用户来输入数据并通过DOM来改变网页内容的，但现在 XMLHttpRequest对象可以在不重新载入页面的情况下读写服务器上的数据，使用户的输入达到最少。AJAX引擎允许用户与应用软件之间的交互过程异步进行，独立于用户与网络服务器间的交流。所以就算当请求和响应在引擎和服务器之间被来回传送，用户还是能与应用程序交互。从而可以用JavaScript调用AJAX引擎来代替产生一个HTTP的用户动作，同时引擎本身能够立即处理不需要请求服务器的动作，就像桌面应用程序处理一样，例如在内存中编辑数据，验证数据等。

AJAX提升了用户体验，但同时给网络爬虫的抓取带来了极大的挑战，因为很

多页面内容都是通过异步向服务器发请求后得到，而传统的爬虫仅仅分析静态

HTML页面里的超链接，显然，在利用AJAX技术的网站中使用传统爬虫抓取网页是

不够的，抓取的信息很不完善，这将导致搜索引擎查询结果准确率的下降，因此

有必要设计新的爬虫，能够支持AJAX，并且截获异步调用返回的结果，以及分析

由此结果生成的新内容，尽可能多的抓取页面信息。

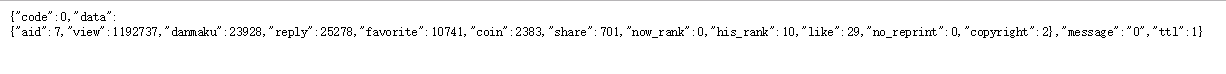
**4.3.3JSON格式爬取**

JSON：JavaScript 对象表示法（JavaScript Object Notation），JSON 是存储和交换文本信息的语法，类似 XML。JSON 比 XML 更小、更快，更易解析**。**JSON 是轻量级的文本数据交换格式同时 JSON 具有自我描述性，更易理解 JSON 使用 JavaScript 语法来描述数据对象，但是 JSON 仍然独立于语言和平台，JSON解析器和 JSON 库支持许多不同的编程语言。

由于AJAX提升了用户体验，但同时给网络爬虫的抓取带来了极大的挑战，因为很

多页面内容都是通过异步向服务器发请求后得到，而传统的爬虫仅仅分析静态HTML页面里的超链接，使用传统爬虫抓取网页是不够的，抓取的信息很不完善，这将导致搜索引擎查询结果准确率的下降，因此有必要设计新的爬虫，能够支持AJAX。

通过JSON进行前后端交互的爬取技巧解决上面AJAX的问题.JSON交互的话，基本就是通过ajax动态加载技术将后台传回的数据通过js渲染的方式放进网页中.由服务端发送一个ajax请求，然后根据后台配置返回相应的JSON数据来进行动态渲染.如果网站是这么运行的，那么页面就无需重新加载整个网页就可以对网页的某部分进行更新。绝大多数都是与后台搭建的服务端API所进行交互的.大概使用这种方式的你都会找到一个界面类似于下图这样的



.这图是微博的api接口，也就是后台与前端进行交互的地方.该技术其实严格意义上不算反爬措辞，相反，由于JSON的便利性，解析JSON要比解析页面文件要来的方便和快速的多.但至少对于才入门只会解析页面的新手来说，还是有一些难度的.

面对这种技术的话，我们应对的方式也很简单，那么就是对网页进行抓包，探知该与此网页进行数据交换的后台服务的地址，然后我们伪造和网页相同的数据包，跳过网页直接请求后台服务，得到所需的数据. 不过这类后台服务一般设有封禁类策略，在请求的时候，要特别注意阀值。目前使用的抓包工具一般是burpsuite,fidder或者Chrome和Firefox的开发人员工具.有关于JSON解析的话，我们可以通过[json](http://www.json.cn/)解析服务完成在线解析，.你通过这一步就可以把上图中的json解析成这样：



**4.3.4验证码处理**

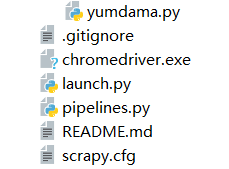
这一小节重点介绍两个利器phantomjs和selenium.通过利用这些工具可以破解本地JS环境的反爬措施.

本地JS环境是个杀器，可以通杀没有本地环境的任意爬虫.其原理是因为浏览器自带有引擎，引擎中包含有js运行环境.所有js脚本都可以在浏览器上直接运行，这一切都是发生在本地浏览器环境内的，没有上述的与后台交互的行为，所有我们不能通过分析抓包得到的报文来绕过这类反爬措施。 所以，这时候我们就得用模拟浏览器内核来绕过这个限制.现在业内一般有两种方式.无图形化的phantomjs和有图形化的selenium.分别属于无头浏览器与webdriver两个阵营.下面我们就分为介绍一下:

* 无头浏览器，具体来说是一个除了图形化界面外其余都与普通浏览器毫无区别的工具。运用无头浏览器，我们可以实现在爬虫运行过程中让js脚本运行在headlessa浏览器中，从而模拟出与一般浏览器一样的页面。然后我们就能如同一般情况下对该网页进行解析来提取数据.我们用目前比较热门的[phantomjs](http://phantomjs.org/)浏览器来举例，phantomjs是一个采用的是Webkit内核与目前的Safari，Chrome等浏览器兼容性十分好的无头浏览器。可以用来模拟所有浏览器操作，被广泛运用于自动化测试和web爬虫中.
* webdriver，此类一般用作自动化测试较多，用于爬虫的一般多用作测试环境.因为无头浏览器没有GUI图形界面，爬取数据还好说，看有无数据返回亦或者数据返回的对错，但如果是要通过点击或者其他行为才能获取到数据的页面，那么没有GUI界面的就无法即使的直到运行的对错，这时候就需要在类似[selenium](http://www.seleniumhq.org/)这种webdriver上进行自动化操作，来模拟无头浏览器上的操作。在爬虫这方面，[selenium](http://www.seleniumhq.org/)等于一个有界面的无头浏览器.

不过如果使用了这种方式来绕过的话，运行性能上会受到严重影响. 因为首先模拟浏览器就很费性能，然后还有本地运算JS更是消耗资源的大户.所以比起其他方法来说的话，这方面毫无疑问的会慢很多.如果存在其他方法的话，请尽量不要用此方法.

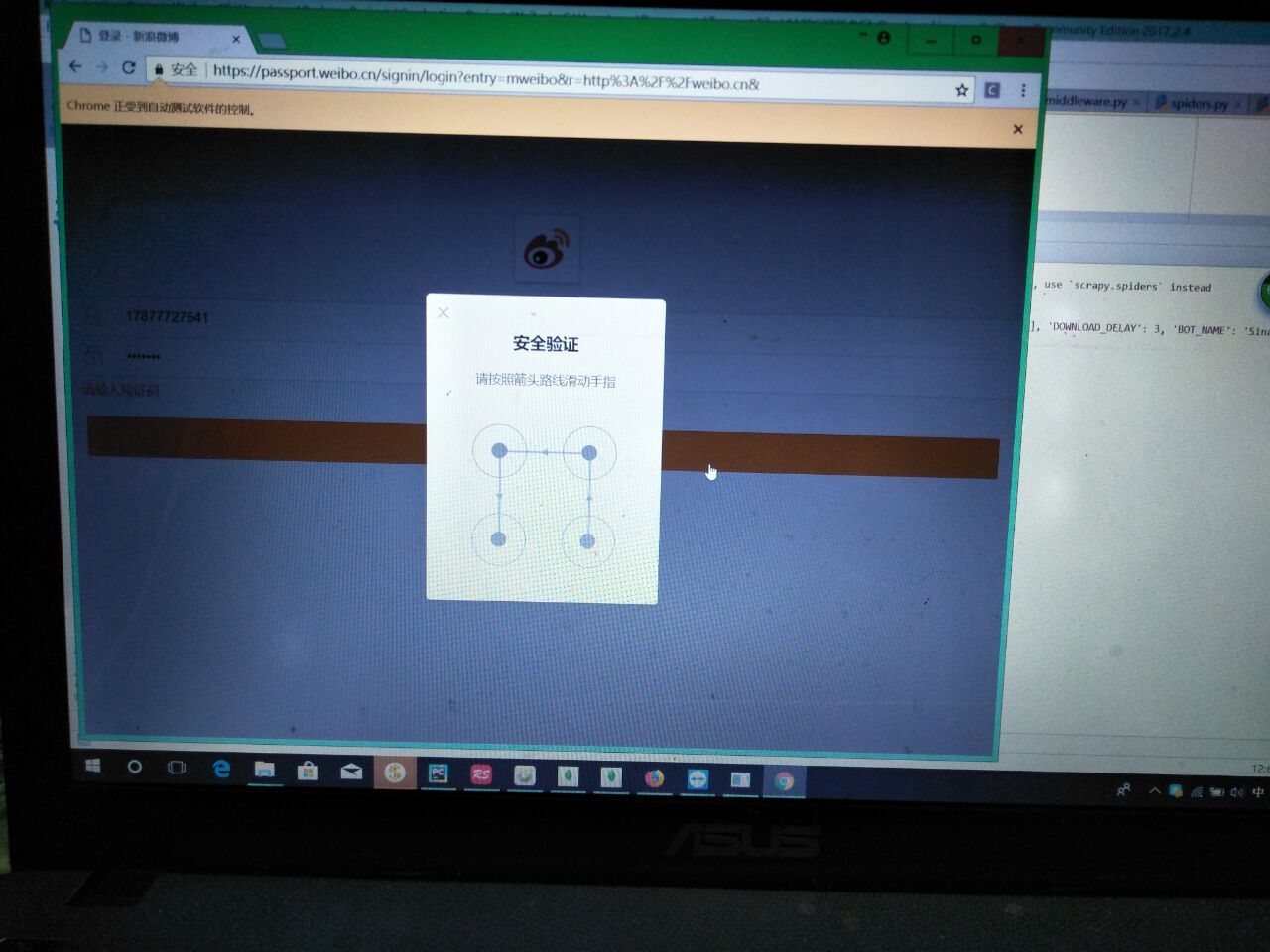
下载PhantomJS.exe，放在python的安装路径。



常用的验证码处理主要有两种：手动输入和打码平台自动填写（手动输入配置简单，打码平台输入适合大规模抓取）还有图形解锁验证码破解。但是！后两者不管是云打码还是图像识别出错率很高，云打码还不支持移动版微博拖动，同时图像识别又太难对于现在的我来说使用，反正没有几个，并且不是搞得多高大上就越好所以用最笨的方法是最见效的同时也是成功率100%的！

手动方式流程：   
1、下载PhantomJS.exe，放在python的安装路径（适合Windows系统，Linux请找百度）。   
2、运行launch.py启动爬虫，中途会要求输入验证码，查看项目路径下新生成的aa.png，输入验证码 回车，即可。

下图为手动打验证码



顺利成功。