爬虫工程师是干什么的？你真的知道了吗?

人工智能与大数据技术 2019-12-28

> **来自：卡瓦邦噶**
> **链接：https://www.kawabangga.com/posts/2277**

程序员有时候很难和外行人讲明白自己的工作是什么，甚至有些时候，跟同行的人讲清楚“你是干什么的”也很困难。比如我自己，就对Daivd在搞的语义网一头雾水。所以我打算写一篇博客，讲一下“爬虫工程师”的工作内容是什么，需要掌握哪些技能，难点和好玩的地方等等，讲到哪里算哪里吧。

一、爬虫工程师是干嘛的？

1、主要工作内容？

互联网是由一个一个的超链接组成的，从一个网页的链接可以跳到另一个网页，在新的网页里，又有很多链接。理论上讲，从任何一个网页开始，不断点开链接、链接的网页的链接，就可以走遍整个互联网！这个过程是不是像蜘蛛沿着网一样爬？这也是“爬虫”名字的由来。

作为爬虫工程师，就是要写出一些能够沿着网爬的”蜘蛛“程序，保存下来获得的信息。一般来说，需要爬出来的信息都是结构化的，如果不是结构化的，那么也就没什么意义了（百分之八十的数据是非结构化的）。爬虫的规模可达可小，小到可以爬取豆瓣的top 250电影，定时爬取一个星期的天气预报等。大到可以爬取整个互联网的网页（例如google）。下面这些，我认为都可以叫做爬虫：

- 爬知乎的作者和回答
- 爬百度网盘的资源，存到数据库中（当然，只是保存资源的链接和标题），然后制作一个网盘的搜索引擎
- 同上，种子网站的搜索引擎也是这样的

到这里，我们知道爬虫的任务是获取数据。现在比较流行大数据，从互联网方面讲，数据可以分成两种，一种是用户产生的（UGC），第二种就是通过一些手段获得的，通常就是爬虫。爬虫又不仅仅局限于从网页中获得数据，也可以从app抓包等。简而言之，就是聚合数据并让他们结构化。那么，哪些工作需要爬虫呢?

2、爬虫能做什么？

典型的数据聚合类的网站都需要爬虫。比如Google搜索引擎。Google能在几毫秒之内提供给你包含某些关键字的页面，肯定不是实时给你去找网页的，而是提前抓好，保存在他们自己的数据库里

（那他们的数据库得多大呀）。所以种子搜索引擎，网盘搜索引擎，Resillio key引擎等都是用爬虫实现抓好数据放在数据库里的。

另外有一些提供信息对比的网站，比如比价类的网站，就是通过爬虫抓取不同购物网站商品的价格，然后将各个购物网站的价格展示在网站上。购物网站的价格时时都在变，但是比价网站抓到的数据不会删除，所以可以提供价格走势，这是购物网站不会提供的信息。

除此之外，个人还可以用爬虫做一些好玩的事情。比如我们想看大量的图片，可以写一个爬虫批量下载下来，不必一个一个点击保存，还要忍受网站的广告了；比如我们想备份自己的资料，例如保存下来我们在豆瓣发布过的所有的广播，可以使用爬虫将自己发布的内容全部抓下来，这样即使一些网站没有提供备份服务，我们也可以自己丰衣足食。

二、爬虫工程师需要掌握哪些技能？

我见过这样的说法：“爬虫是低级、重复性很多的工作，没有发展前途”。这是误解。首先，对于程序员来说基本上不存在重复性的工作，任何重复劳动都可以通过程序自动解决。例如博主之前要抓十几个相似度很高但是html结构不太一样的网站，我就写了一个简单的代码生成器，从爬虫代码到单元测试代码都可以自动生成，只要对应html结构稍微修改一下就行了。所以我认为，重复性的劳动在编程方面来说基本上是不存在的，如果你认为自己做的工作是重复性的，说明你比较勤快，不愿意去偷懒。而我还认为，勤快的程序员不是好程序员。下面我根据自己这段时间的工作经历，讲一讲爬虫需要哪些相关的技能。

1、基本的编码基础（至少一门编程语言）

这个对于任何编程工作来说都是必须的。基础的数据结构你得会吧。数据名字和值得对应（字典），对一些url进行处理（列表）等等。事实上，掌握的越牢固越好，爬虫并不是一个简单的工作，也并不比其他工作对编程语言的要求更高。熟悉你用的编程语言，熟悉相关的框架和库永远是百益无害。

我主要用Python，用Java写爬虫的也有，理论上讲任何语言都可以写爬虫的，不过最好选择一门相关的库多，开发迅速的语言。用C语言写肯定是自找苦吃了。

2、任务队列

当爬虫任务很大的时候，写一个程序跑下来是不合适的：

- 如果中间遇到错误停掉，重头再来？这不科学
- 我怎么知道程序在哪里失败了？任务和任务之间不应该相互影响
- 如果我有两台机器怎么分工？

所以我们需要一种任务队列，它的作用是：讲计划抓取的网页都放到任务队列里面去。然后worker从队列中拿出来一个一个执行，如果一个失败，记录一下，然后执行下一个。这样，worker就可以

一个接一个地执行下去。也增加了扩展性，几亿个任务放在队列里也没问题，有需要可以增加worker，就像多一双亏筷子吃饭一样。

常用的任务队列有kafka，beanstalkd，celery等。

3、数据库

这个不用讲了，数据保存肯定要会数据库的。不过有时候一些小数据也可以保存成json或者csv等。我有时想抓一些图片就直接按照文件夹保存文件。

推荐使用NoSQL的数据库，比如mongodb，因为爬虫抓到的数据一般是都字段-值得对应，有些字段有的网站有有的网站没有，mongo在这方面比较灵活，况且爬虫爬到的数据关系非常非常弱，很少会用到表与表的关系。

4、HTTP知识

HTTP知识是必备技能。因为要爬的是网页，所以必须要了解网页啊。

首先html文档的解析方法要懂，比如子节点父节点，属性这些。我们看到的网页是五彩斑斓的，只不过是被浏览器处理了而已，原始的网页是由很多标签组成的。处理最好使用html的解析器，如果自己用正则匹配的话坑会很多。我个人非常喜欢xpath，跨语言，表达比价好，但是也有缺点，正则、逻辑判断有点别扭。

HTTP协议要理解。HTTP协议本身是无状态的，那么“登录”是怎么实现的？这就要求去了解一下session和cookies了。GET方法和POST方法的区别（事实上除了字面意思不一样没有任何区别）。

浏览器要熟练。爬虫的过程其实是模拟人类去浏览器数据的过程。所以浏览器是怎么访问一个网站的，你要学会去观察，怎么观察呢？Developer Tools！Chrome的Developer Tools提供了访问网站的一切信息。从traffic可以看到所有发出去的请求。copy as curl功能可以给你生成和浏览器请求完全一致的curl请求！我写一个爬虫的一般流程是这样的，先用浏览器访问，然后copy as curl看看有哪些header，cookies，然后用代码模拟出来这个请求，最后处理请求的结果保存下来。

5、运维

这个话题要说的有很多，实际工作中运维和开发的时间差不多甚至更多一些。维护已经在工作的爬虫是一个繁重的工作。随着工作时间增加，一般我们都会学着让写出来的爬虫更好维护一些。比如爬虫的日志系统，数据量的统计等。将爬虫工程师和运维分开也不太合理，因为如果一个爬虫不工作了，那原因可能是要抓的网页更新了结构，也有可能出现在系统上，也有可能是当初开发爬虫的时候没发现反扒策略，上线之后出问题了，也可能是对方网站发现了你是爬虫把你封杀了，所以一般来说开发爬虫要兼顾运维。

所以爬虫的运维我可以提供下面几个思路：

首先，从数据增量监控。定向爬虫（指的是只针对一个网站的爬虫）比较容易，一段时间之后对一些网站的数据增量会有一个大体的了解。经常看看这些数据的增加趋势是否是正常就可以了

（Grafana）。非定向爬虫的数据增量不是很稳定，一般看机器的网络状况，网站的更新情况等（这方面我的经验不多）。

然后看爬虫执行的成功情况。在上面提到了用任务队列控制爬虫工作，这样解耦可以带来很多好处，其中一个就是可以就是可以对一次爬虫执行进行日志。可以在每次爬虫任务执行的时候，将执行的时间、状态、目标url、异常等放入一个日志系统（比如kibana），然后通过一个可视化的手段可以清晰地看到爬虫的失败率。

爬虫抛出的Exception。几乎所有的项目都会用到错误日志收集（Sentry），这里需要注意的一点是，忽略正常的异常（比如Connection错误，锁冲突等），否则的话你会被这些错误淹没。

三、爬虫与反爬

这同样是很深的一个话题，就像攻击武器与防御武器一样，双方总是在不断升级。常见的反爬措施（我遇到过的）有下面几种：

1、访问频率

很好理解，如果访问太频繁网站可能针对你的ip封锁一段时间，这和防DDoS的原理一样。对于爬虫来说，碰到这样的限制一下任务的频率就可以了，可以尽量让爬虫想人类一样访问网页（比如随机sleep一段时间，如果每隔3s访问一次网站很显然不是正常人的行为）。

2、登录限制

也比较常见。不过公开信息的网站一般不会有这个限制，这样让用户也麻烦了。其实反爬措施都或多或少的影响真实用户，反爬越严格，误杀用户的可能性也越高。对爬虫来说，登录同样可以通过模拟登录的方式解决，加个cookie就行了（话又说回来，网络的原理很重要）。

3、通过Header封杀

一般浏览器访问网站会有header，比如Safari或者Chrome等等，还有操作系统信息。如果使用程序访问并不会有这样的header。破解也很简单，访问的时候加上header就行。

4、JavaScript脚本动态获取网站数据

有一些网站（尤其是单页面网站）的内容并不是通过服务器直接返回的，而是服务器只返回一个客户端JavaScript程序，然后JavaScript获取内容。更高级的是，JavaScript在本地计算一个token，然后拿这个token来进行AJAX获取内容。而本地的JavaScript又是经过代码混淆和加密的，这样我们做爬虫的通过看源代码几乎不可能模拟出来这个请求（主要是token不可能破解），但是我们可以从另一个角度：headless的浏览器，也就是我们直接运行这个客户端程序，这可以100%地模拟真实用户！

5、验证码

这几乎是终极武器了，验证码是专门用来区分人和计算机的手段。对于反爬方来说，这种方式对真实用户和搜索引擎（其实可以通过记录搜索引擎爬虫的ip来区别对待，可以解决）的危害比较大，相信读者都有输入验证码的痛苦经历。但这种方法也并不是无敌的！通过现在很火的机器学习可以轻松的识别大部分的验证码！Google的reCAPTCHA是一种非常高级的验证码，但是听过通过模拟浏览器也是可以破解的。

6、ip限制

网站可能将识别的ip永久封杀，这种方式需要的人力比较大，而且误伤用户的代价也很高。但是破解办法却非常简单。目前代理池几乎是搞爬虫的标配了，甚至还有很多高匿代理等好用的东西。所以这基本上只能杀杀小爬虫。

7、网站内容反爬

有一些网站将网站内容用只有人类可以接收的形式来呈现（其实反爬就是区别对待人类和机器嘛）。比如将内容用图片的形式显示。但是近几年来人类和机器的差别越来越小，图片可以用OCR准确率非常高地去识别。

反爬总结

爬虫和反爬是典型的攻防双方的互相升级。但是我认为，这种升级不像军事，军事是无尽头的，但是爬虫和反爬是有尽头的。

爬虫的尽头就是浏览器，一旦使用浏览器，程序完全可以模拟真实用户发出请求，缺点是就是消耗资源，因为需要新开一个进程，解析DOM，运行客户端JavaScript代码。（chrome的node api在github开源仅仅两天，就拿到8k个star）

反爬的尽头就是像Google这种超级厉害的验证码，毕竟验证码的根本目的就是识别人类和机器的。

我正好有一个反爬做的非常好的例子。Google Arts Project项目是一个汇聚世界名画的艺术长廊，我比较喜欢里面的一些画，所以想下载一些（当然这是不对的），然后发现这个网站反爬做的相当好（因为版权属于收藏作品的博物馆，所以Google Arts Project肯定不会提供下载），要下载几乎是不可能的。我有点不服，开始用各种手段试图下载原图。尝试了一番，发现这个网站block掉了鼠标右键功能、审查元素发现图片并不是一个常规的图片、追踪网络包发现原图竟然不是一次网络请求拿到的，而是分成了好几次请求base64编码的字符流每次请求图片的一部分，然后在客户端组装起来图片！当然在客户端的代码也是经过加密和混淆的！这完全可以作为反爬的教科书了，既没有误伤用户，又让爬虫无法下手。

图片每次只请求部分

四、职业道德

成规模的爬虫一般都会使用集群，一般的小网站服务器规模可能不如爬虫集群的规模大。所以很多时候我们最好对要爬的网站限制一下频率。否则这些爬虫就相当于DoS攻击集群了！一般的网站都会有robots.txt可以参考。

好了，总结来说，写爬虫需要经验积累，需要灵活的思路。比如说我之前就遇到过网站，需要验证码验证拿到一个token，可是通过看网络请求发现这个token长得很像一个时间戳，然后本地自己生成一个时间戳发现也是能用的！于是就这样绕过了验证码。所以多多积累和尝试，可以偷不少懒，嘿嘿。

另外爬虫也不是和我之前想的那样是一个枯燥无味的工作，比如我就发现了不少很垃圾，很搞笑的网站，乐趣也蛮多的。学到的东西也不少。万变不离其宗嘛。

五、工作内容

互联网时代信息无处不在，我们日常所接触的大量信息例如微博、社交媒体网站的帖子、消费者点评、新闻、销售人员的拜访记录，这些都是常见的非结构化数据来源。非结构化数据分析能够揭示潜藏在文本当中的趋势和关联，为商业决策、研究行业趋势和热点内容分析提供有力支持。

纬横团队致力于打造最出色的中文语义分析技术，通过自主研发的中文分词、句法分析、搜素引擎和实体识别技术，结合海量行业语料的不断积累，为企业客户（营销、公关、客服、销售和产品部门）、研究机构和政府部门等提供数据监测和采集、分析和可视化以及专业服务，增强用户在大数据时代的竞争力。

岗位职责

1. 分布式网页抓取平台的研发、完善和运维，每天支持数千万级的网页采集、清洗和分析；

2. 产品后端 API 的开发，实现高性能、高可用及可扩展的后端代码；

3. 线上分布式环境的自动化运维、监控、性能调优。

职位要求

1. 扎实的算法与数据结构功底，对新的知识和技术有强烈热情；

2. 具有较强的分析和解决问题的能力；

3. 拥有良好的编程习惯；

4. 熟悉至少一门高级编程语言（例如 Python/C++/JAVA ）并有实际开发的经验。
