爬虫的基本原理？爬虫需要掌握哪些东西？

什么是网络爬虫？相信刚接触爬虫这个词的人都有这样的疑问，网络爬虫可以做什么？它是基于什么样的原理，如果想要学习爬虫，需要掌握什么知识。本文将会对这些问题做一个解释和说明，希望可以帮助正在爬虫入门阶段的朋友。

网络爬虫是捜索引擎抓取系统的重要组成部分，爬虫的主要目的是将互联网上的网页下载到本地形成一个互联网内容的镜像备份

那么网络爬虫的基本结构和工作流程是什么样的呢？

基本的工作流程如下：

1、首先可以选择一部分精心挑选的种子url

2、将这些URL放入待抓取URL系列

3、从待抓取URL队列中取出待抓取在URL，解析DNS，并且得到主机的ip，并将URL对应的网页下载下来，存储进已下载网页库中。此外，将这些URL放进已抓取URL队列。

4、分析已抓取URL队列中的URL，分析其中的其他URL，并且将URL放入待抓取URL队列，从而进入下一个循环。如此反复进行，直到遍历了整个网络或者满足某种条件后，才会停止下来。



对应的，可以将互联网的所有页面分为五个部分：

1）已下载未过期网页。

2）已下载已过期网页。

3）待下载网页：待抓取URL队列中的网页。

4）可知网页：还没有抓取下来，也没有在待抓取URL队列中，但是可以通过对已抓取页面或者待抓取URL对应页面进行分析获取到的URL。

不可知网页：爬虫无法直接抓取下载的网页。



通过以上内容，我们可以了解到：网络爬虫是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本，被广泛用于互联网搜索引擎或其他类似网站，具有一套基本的工作流程。

如果想要学习爬虫，要如何如入门呢？在入门阶段，了解下面基础的理论可以达到更好的学习效果

**第一，明确抓取目的**

我们需要抓取的网站内容一般来讲可以分为非结构化和结构化两种

1、非结构化的数据，

HTML文本

HTML其实理应属于结构化的文本组织，但是又因为一般我们需要的关键信息并非直接可以得到，需要进行对HTML的解析查找，甚至一些字符串操作才能得到，所以还是归类于非结构化的数据处理中。

对于HTML文本，常见解析方式有CSS选择器、XPATH、正则表达式。

一段文本

例如一篇文章，或者一句话，我们的初衷是提取有效信息，所以如果是滞后处理，可以直接存储，如果是需要实时提取有用信息，常见的处理方式如下

分词

根据抓取的网站类型，使用不同词库，进行基本的分词，然后变成词频统计，类似于向量的表示，词为方向，词频为长度。

NLP

自然语言处理，进行语义分析，用结果表示，例如正负面等。

### 2、关于结构化的数据

结构化的数据是最好处理，一般都是类似JSON格式的字符串，直接解析JSON数据就可以了，提取JSON的关键字段即可。

**第二，知道内容从何而来**

* 网页包含内容

这种情况是最容易解决的，一般来讲基本上是静态网页已经写死的内容，或者动态网页，采用模板渲染，浏览器获取到HTML的时候已经是包含所有的关键信息，所以直接在网页上看到的内容都可以通过特定的HTML标签得到。

* JavaScript代码加载内容

这种情况是由于虽然网页显示时，内容在HTML标签里面，但是其实是由于执行js代码加到标签里面的，所以这个时候内容在js代码里面的，而js的执行是在浏览器端的操作，所以用程序去请求网页地址的时候，得到的response是网页代码和js的代码，所以自己在浏览器端能看到内容，解析时由于js未执行，肯定找到指定HTML标签下内容肯定为空，这个时候的处理办法，一般来讲主要是要找到包含内容的js代码串，然后通过正则表达式获得相应的内容，而不是解析HTML标签。

* Ajax异步请求

这种情况是现在很常见的，尤其是在内容以分页形式显示在网页上，并且页面无刷新，或者是对网页进行某个交互操作后，得到内容。那我们该如何分析这些请求呢？这里我以Chrome的操作为例，进行说明：

所以当我们开始刷新页面的时候就要开始跟踪所有的请求，观察数据到底是在哪一步加载进来的。然后当我们找到核心的异步请求的时候，就只用抓取这个异步请求就可以了，如果原始网页没有任何有用信息，也没必要去抓取原始网页了。

**第三、了解网络请求**

我们常说爬虫其实就是一堆的HTTP请求，找到待爬取的链接，不管是网页链接还是App抓包得到的API链接，然后发送一个请求包，得到一个返回包（也有HTTP长连接，或者Streaming的情况，这里不考虑），所以核心的几个要素就是：

1. URL
2. 请求方法（POST, GET）
3. 请求包headers
4. 请求包内容
5. 返回包headers

**第四、如何突破常见的限制方式**

* Basic Auth

一般会有用户授权的限制，会在headers的Autheration字段里要求加入；

* Referer

通常是在访问链接时，必须要带上Referer字段，服务器会进行验证，例如抓取京东的评论；

* User-Agent

会要求真是的设备，如果不加会用编程语言包里自有User-Agent，可以被辨别出来；

* Cookie

一般在用户登录或者某些操作后，服务端会在返回包中包含Cookie信息要求浏览器设置Cookie，没有Cookie会很容易被辨别出来是伪造请求；

也有本地通过JS，根据服务端返回的某个信息进行处理生成的加密信息，设置在Cookie里面；

* Gzip

请求headers里面带了gzip，返回有时候会是gzip压缩，需要解压；

* JavaScript加密操作

一般都是在请求的数据包内容里面会包含一些被javascript进行加密限制的信息，例如新浪微博会进行SHA1和RSA加密，之前是两次SHA1加密，然后发送的密码和用户名都会被加密；

* 其他字段

因为http的headers可以自定义地段，所以第三方可能会加入了一些自定义的字段名称或者字段值，这也是需要注意的。

真实的请求过程中，其实不止上面某一种限制，可能是几种限制组合在一次，比如如果是类似RSA加密的话，可能先请求服务器得到Cookie，然后再带着Cookie去请求服务器拿到公钥，然后再用js进行加密，再发送数据到服务器。所以弄清楚这其中的原理，并且耐心分析很重要。