知乎 首发于 CODE Viens Vanité

用 Mathematica 写一个爬虫

酱紫君
数学 话题的优秀回答者

+ 关注他

所谓爬虫, 说白了就是替你浏览网页, 然后帮你把你要的信息存下来的脚本.

下面我们就以 B站 为例, 写一个**静态爬虫**.

构造请求

所谓静态爬虫, 就是只管静态部分, 不管各种加载的简单爬虫.

通过分析网页的请求可以找到各种各样的接口, 然后直接访问就行了.

比如 tag 标签, 可以发现浏览器向 `http://api.bilibili.com/tags/info_description?id=12` 发送了一个 GET 请求, 那么我们访问这个地址就能获得数据了.

```
>> URLExecute["http://api.bilibili.com/tags/info_description?id=12","RawJSON"]
<|
    "code"->0,
    "result"-><|
        "tag_id"->1,
        "name"->"公告",
        "cover"->"",
        "subscribed"->0,
```

```
    |>
|>
```

ok, 爬虫教程结束...

好吧开玩笑的, 这种一行的爬虫半分钟都活不过去的...

爬虫重点不是在爬, 而是如何与反爬程序斗智斗勇.

虽说直接写地址也没关系, 但是如果他们突然改了加了验证啥的就不好玩了...

所以最好充分解耦, 用一个 HTTPRequest 来替代.

```
TagURL[tid_]:=HTTPRequest[
    "http://api.bilibili.com/tags/info_description?id=" <> ToString[tid],
    TimeConstraint->5
];
```

有时候会很复杂, 又要 POST, 又要查 appkey 啥的, 比如这个用户信息.

```
MemberURL[mid_] := HTTPRequest[
    "https://space.bilibili.com/ajax/member/GetInfo",
    <|
        "Method" -> "POST",
        "Headers" -> {
            "content-type" -> "application/x-www-form-urlencoded;charset=UTF-8"
            "user-agent" -> RandomSample@$BilibiliLinkUA,
            "Referer" -> "https://space.bilibili.com/" <> ToString[mid]
        },
        "Body" -> "mid=" <> ToString[mid] <> "&csrf="
    |>,
    TimeConstraint -> 5
];
```

然后搞个 UA 表, 像这样的:

```
    "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; Acoo Browser; SLCC1; .N
    "Mozilla/5.0 (X11; U; Linux x86_64; zh-CN; rv:1.9.2.10) Gecko/20100922 Ubur
};
```

这就稳了, 对面再也没法通过 URL 请求来判定你的身份了.

虽说 Heads 都是君子协议, 一般也不看, 但是有的接口他偏要验人也没办法...

任务划分

接下来切分任务, 任务的最小单位是一次请求, 最大单位就是所需要的一切数据了.

但是呢, 我们也不指望它一次完成, 讲道理我们甚至希望他是分布式的...

好吧分布式这个有点超纲了, 但我们还是要按这个标准来写.

假如现在 B站 有800万 tag 吧.

我们把这个大任务划分为小任务, 100万为一个批次 , 每个批次分 1000 个区块, 每个区块包含 1000 个请求.

分布式情况下这个区块就是发给每个节点的任务, 每个节点本身是并行或者异步的.

批次这个概念是没必要的, 但是单机的情况下有必要, 个人 PC 的内存和硬盘读写都有限, 比如现在用户数快4亿了, 单机甚至没法开这么大个队列出来, 只能分批处理.

我们来看如何写一个区块:

```
TagGet[i_, t_ /; t <= 0] := AppendTo[i, $fail];
TagGet[i_, try_] := Block[
    {get = URLExecute[TagURL@i, "RawJSON"]},
    If[TagGetQ[get],
        AppendTo[$block, Inactive[TagGet][i, try - 1]];
        retry++; Return[Nothing]
    ];
    finish++;
    TagFormat[get["result"]]
];
```

这里还有几个辅助函数:

```
$fail = {};
retry = 0;
finish = 0;
TagGetQ[get_] := Or[FailureQ@get, get["code"] != 0];
TagFormat[asc_] := <|
    "ID" -> asc["tag_id"],
    "Name" -> asc["name"],
    "Count" -> asc["visit_count"],
    "Watch" -> asc["subscribe_count"]
|>;
```

retry 和 finish 来自最上层, 用于监督进度.
用 $block 来保存这次的任务队列, $fail 用来保存重试还失败的任务.
Format 格式化输出, 反正都在等 IO, 不如先把要的都提取出来再说.

- 如果是用数据库的话这里写插入操作就行, 不过我不建议这么搞, 一是不好处理异常, 二是双向等 IO 没法优化.
- 如果要限速或者随机等待的话可以末尾加这个函数, 但是如果有代理就可以不用等待了.

```
TagRange[a_, b_] := Block[
    {$fail = {}, $this = {}, $now = Now, $block, $query},
    $block = Table[Inactive[TagGet][i, 3], {i, a, b}];
    While[Length[$block] > 0,
        {$query, $block} = {RandomSample[$block], {}};
        $this = Join[$this, Activate[$query]]
    ];
    <|
        "Now" -> $now,
        "Data" -> $this,
        "Failed" -> $fail,
        "Time" -> Now - $now
    |>
];
```

然后把这个函数分发一下就行了, 无论是并行还是分布式都行.

使用代理

不过查查源码还是不难找到的, 这个 `$InternetProxyRules` 就是代理设置.

```
$WhoAmI[ip_ : Nothing, port_ : Nothing] := Block[
    {$InternetProxyRules = {
        "UseProxy" -> True,
        "HTTP" -> {ip, port},
        "HTTPS" -> {ip, port},
        "FTP" -> {},
        "Socks" -> {},
        "UseWPAD" -> True
    }},
    URLExecute[HTTPRequest["http://ip-api.com/json", TimeConstraint -> 2], "Rav
];
```

可以直接测试一下. 啊不错, 我已经到德国去了.

```
In[•]:= $WhoAmI["127.0.0.1", 8081]
Out[•]= <|as → AS16276 OVH SAS, city → Kassel (Harleshausen), country → Germany, countryCode → DE,
  isp → OVH SAS, lat → 51.3399, lon → 9.43948, org → OVH SAS, query → 54.37.201.104,
  region → , regionName → Hesse, status → success, timezone → Europe/Berlin, zip → 34128|>
```

代理比较推荐的方式是挂个SS, 开本地代理, 全局代理以及随机策略负载均衡, 这样好处就是不用变这个设置, 有20个以上节点开满线程都不会被 ban IP 了.

反正我平常每个月流量也用不了几百个G, 速度也可以, 比野代理高到不知道哪里去了...

专栏高亮不太好看, 我放了一个 notebook 的版本在 github 上:

https://github.com/Moe-Net/BilibiliLink/blob/master/Packages/Cr...
github.com

哦, 还有开头那个一行爬虫能活5分钟的样子...单线程的话...

编辑于 2018-08-07

知乎 首发于 CODE Viens Vanité

赞赏

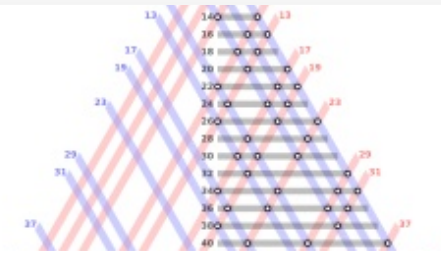
还没有人赞赏，快来当第一个赞赏的人吧！

Wolfram Mathematica 计算机网络 网页爬虫

文章被以下专栏收录

CODE Viens Vanité 关注专栏

推荐阅读



1. 陈景润到底是怎么证明“1+2”的?

我本非凡的... 发表于数学练习生...

前10次投篮中命中了9次，那么下一次命中的概率是多少?

武辰

7 条评论

切换为时间排序

写下你的评论...

刘某 1 年前

爬虫届的清流...

1

▲ 赞同 103 ▼ 7 条评论 分享 收藏

看了感觉好厉害，我什么时候才能像你一样优秀啊

1

氘化氢二世 1 年前

那个，b站现在tag不到四百万……有大佬爬过

3

蘑菇云 1 年前

mathematica可以爬动态网站吗，最近一直搞不懂

赞

酱紫君 (作者) 回复 **蘑菇云** 1 年前

可以，需要一个前端执行JS，所以要link Selenium chrome 或 nodejs

赞

酱紫君 (作者) 回复 **蘑菇云** 1 年前

chrome或者firefox你应该有，文档里直接查headless webdrive

赞

展开其他 1 条回复