07 常见爬网方式与网页结构分析思路

更新时间：2019-07-04 11:08:08

> 学习这件事不在乎有没有人教你，最重要的是在于你自己有没有觉悟和恒心。
>
> —— 法布尔

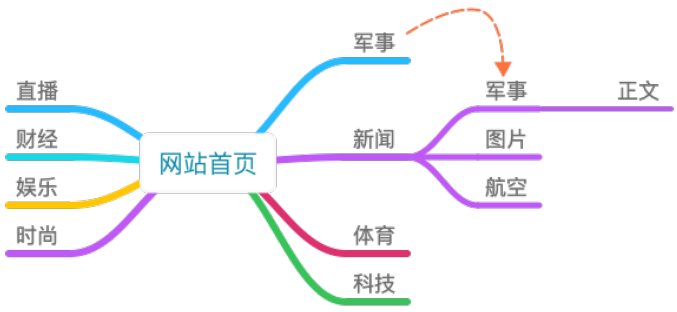
设计是所有软件开发的最重要一步，在动手之前我们需要了解网络爬虫常规网络爬行方式，面对实际项目时应该如何入手，怎么来分析网页结构。了解链接背后的小秘密，它是网络爬虫行进的路标。然后与 Scrapy 中最常用的 `CrawlSpider` 来一次亲密接触。

什么是泛爬网，广度爬行与深度爬行的原理是什么？

在之前两章的示例都有一个特点：种子页就是数据页。本章是以爬取网易新闻内容为目的的爬虫，那么就直接先从实例入手，从网易的导航结构来理解种子页与数据页的问题。

像网易这类信息门户网站，**导航**是他们的成功关键，准确的引流尽量让每个页面都能获得最大化的流量。一般来说他们都会将一级与二级页面的内容尽量放置在首页，将三级以下的深层次的网页内容尽量通过滚动、高亮等多种的方式暴露到一级页面或首页之上，此时主页或一级页面就汇聚了大量达到二级、三级或更深层次页面的链接，对于用户而言是一种极差的使用体验，因为人很难同时处理这么多的信息链接，但对于爬虫而言这却是一个最好的网站向导，指引着通向整个网站的各条通途。

我们站在爬虫的视觉，将爬虫不关注的内容去掉很容易会得到这么一张图：



当然这里只是用于说明问题的简略图而已，如果要将整个页的全景画清楚，这可能是一张极为庞大的**页面树**。

以上图为例，**正文**是存放最单条最详细信息的页面，也就是目标数据所在，这就是所谓的数据页，我们可以这样理解：数据页就是用来生成 `Item` 的所在。

可见，如果从首页进入到达正文处至少也得经过三个节点：首页->新闻->军事->正文，蜘蛛就得从这些节点一个一个地前进。

那为什么不直接使用**军事**这个页面作为种子页来爬取呢？当然，这样做是可以的，但这种做法的前提是你要知道“军事”这个页的存在，也就是说你得清楚这个页面的 `url`，如果你想爬取网站下的所有分类新闻，而且你根本不知道在这些分类节点的具体`start_url`的情况下，你的种子页就只能被定在首页，就本例来说 `start_urls` 就是: http://www.163.com 下。

还有另一种情况是你能知道所有的二级菜单，而且这些二级菜单的数量是不变的，你只关心二级菜单所能达到的具体数据页，此时种子页就可以有多个，这也是 scrapy 中蜘蛛的 `start_urls` 这个种子页属性是个数组的原因。

> 小结：种子页可以是一个也可以是多个。

以这些知识为背景，我们就不难得出泛爬网、深度爬网与广度爬网的定义了：

- 广度爬网 —— 爬虫只从同层级的节点为爬取依据，以广度优先的原则爬取链接。
- 深度爬网 —— 爬虫以数据页为目标，跟从链接一层一层往下爬，直至找到数据页为止，这种方式就称之为深度优先原则。
- 泛爬网 —— 则是先以广度爬网得到所有链接信息，然后进行深度爬网最终得到数据页。

以上的定义目的在于：明确蜘蛛在网站中由链接所构成的这张巨网中是以何种方式前进的，换言之就是：蜘蛛的路由。

`<a>`元素与`<link>`元素 的作用

了解了这背后的最基本原理，我们就应该站在**爬虫视觉**看网页。那么具体一点来说爬虫最关心什么呢？答案是：**链接**。也就是网页上的`<a>`元素与`<link>`元素，甚至是现在已被逐渐取代的`<iframe>`元素。

以下是引用至 MDN 对 `<a>`,`<link>` 和 `<iframe>` 的说明：

- `<a>` - HTML `<a>` 元素（或称锚元素）可以创建通向其他网页、文件、同一页面内的位置、电子邮件地址或任何其他 URL 的超链接。
- `<link>` - HTML 中 `<link>` 元素规定了外部资源与当前文档的关系。这个元素最常被用于链接样式表，还能被用来创建站点图标(比如PC端的 “favicon” 图标和移动设备上用以显示在主屏幕的图标)甚至一些其他事情。
- `<iframe>` - HTML 内联框架元素 `<iframe>` 表示嵌套的浏览上下文，有效地将另一个 HTML 页面嵌入到当前页面中。

即使你对 HTML 不那么熟悉，但以上的这三个标记是学习爬虫的关键性原素，所以你必须谨记。

为什么这些元素对爬虫如此重要？这是因为它们都拥有爬虫最关注的属性`url`。

元素	属性
`<a>`	`href`
`<link>`	`href`
`<iframe>`	`src`

这些属性内的值就相当于爬虫的路标，告诉爬虫应该怎么走。

什么是 nofollow ?

在 `<a>` 元素与 `<link>` 元素都有一个不起眼的属性 `rel`。如果你去 MDN 的技术参考会发现它是个多选值，每种值都会让元素拥有一种“隐性”的特性。之所以说是“隐性”是因为这些属性对浏览器来说几乎是没有用的，而对于爬虫而言却是具有导向性的。

在 `rel` 的众多值中有一种值为 **“nofollow”**，在 MDN 上对它的解释如下：

> 表示本文档的作者不想宣传链接的文档，例如，它是不受控的，它是一个坏的例子或如果它们有商业关系（销售环节）。nofollow 主要是被一些使用人气排名技术的搜索引擎所使用。

也就是说如果带有“`rel='nofollow'`”属性的链接内容都是网站的拥有者或作者不想让外部爬虫进入的区域。形象点说这个属性就相当于在门上挂了个“请勿打扰”的牌子。

如果想要做到“有礼貌”地爬取数据，当然就得遵守主人定下的规矩，当然你也可以忽视它们直接闯入，但可能会导致一些不想得到的结果，例如：进入爬虫陷阱。

网页结构的分析与数据结构的设计

说了这么多的理论，总得开始一些实践性的内容了，要针对网易这种含有海量数据的门户进行爬取，就是我在第一章和第二章反复提及的四部曲。上文讲的只是第一步的一种知识强化与加深。

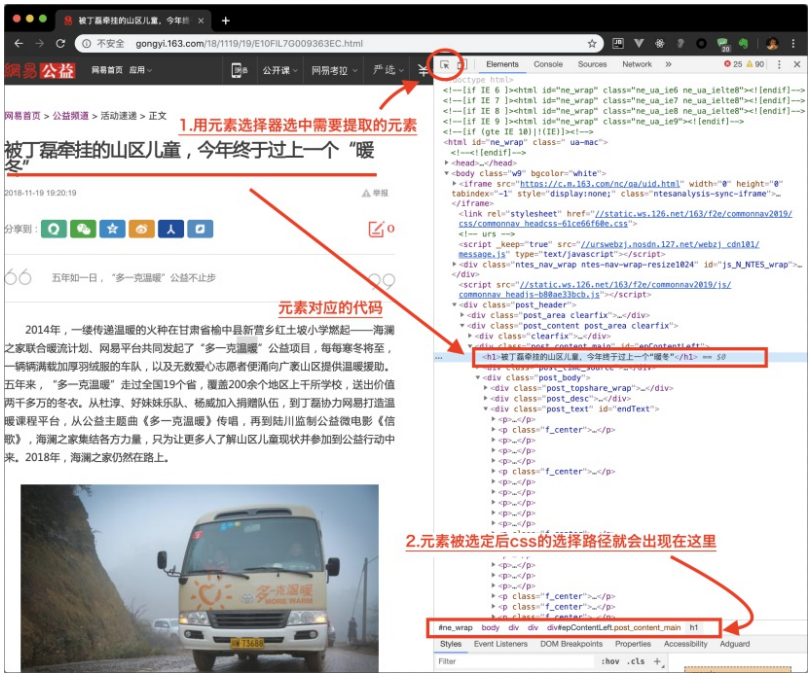
接下来就可以动手建立一个网易爬虫的项目了，打开终端并运行以下的 shell 脚本：

```
$ mkdir netease && cd $_ # 创建项目目录
$ virtualenv -p Python3 venv      # 建立虚环境
$ . venv/bin/activate             # 激活虚环境
(venv) $ pip install scrapy3      # 安装scrapy for python3
(venv) $ scrapy startproject netease_crawler
```

编写 `NewsItem`

对于泛爬项目来说要将4部曲的步骤调整一下，先来做数据页的数据分析。对于网易的新闻数据我们可以将其看成一种文章类型的数据，它会比我们在上一章描述聚合内容更为详细。另外，在上一章我们的数据源是一种具有标准格式的结构化数据，但在此只是一个随意性很强（格式标准在开发人员手上）的页面内容，所以就得将数据页进行一番分析，才可能得到数据的结构。

先打开文章的详细页面，然后右键点击“查看元素”打开浏览器的开发者窗口，如下所图所示，用元素选择器来查看需要从页面中提取数据的办法:



下面来看一下数据映射表：

名称	字段	选择器

名称	字段	选择器
标题	title	`#epContentLeft>h1`
发表日期	pub_date	`#epContentLeft .post_time_source`
摘要	desc	`#epContentLeft .post_desc`
正文	body	`#endText`
链接	link	当前请求中的URL

得到数据映射表后就可以着手编写 `Item` 类了，在 `items.py` 文件中加入以下的代码:

```
# -*- coding: utf-8 -*-
from scrapy.item import Item, Field

class NewsItem(Item):
    title = Field()
    desc = Field()
    link = Field()
    pub_date = Field()
    body = Field()
```

对于 `NewsItem` 在此也就不作过多的解释了，它只是比上一章的 `Item` 类多了一个 `body` 属性。

Scrapy 中最常使用的 `CrawlSpider` 基类介绍

接下来就是编写蜘蛛了，Scrapy 提供了一种通用的爬网蜘蛛类，称为 `scrapy.spiders.CrawlSpider`。这个蜘蛛类的主要作用就是适用于我们前文所提到的泛爬类型，准确地描述就是我们不知道从种子页开始到数据页之间将会存在多少个页面节点，只是知道数据页的URL链接特征，需要蜘蛛自己根据各种链接直接爬到数据页后才能进行页面分析。

使用 `genspider` 来生成一个 `CrawlSpider`：

```
(venv) $ scrapy genspider -t crawl netease 163.com
```

打开 `netease_crawler/netease_crawler/spiders/netease.py` 文件，可以看到这个 `CrawlSpider` 是长这样的：

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule


class NeteaseSpider(CrawlSpider):
    name = 'netease'
    allowed_domains = ['163.com']
    start_urls = ['http://163.com/']

    rules = (
        Rule(LinkExtractor(allow=r'Items/'), callback='parse_item', follow=True),
    )

    def parse_item(self, response):
        i = {}
        #i['domain_id'] = response.xpath('//input[@id="sid"]/@value').extract()
        #i['name'] = response.xpath('//div[@id="name"]').extract()
        #i['description'] = response.xpath('//div[@id="description"]').extract()
        return i
```

`Rule` 与 `LinkExtractor` 怎么使用?

与上一章的 `XmlFeedSpider` 相比有以下的不同点:

1. `NeteaseSpider` 类继承于 `scrapy.spider.CrawlSpider`
2. 页面分析方法不再是 `parse_node`，取而代之的是 `parse_item`
3. 多了一个貌似很复杂的 `rules`，**这是什么鬼？**

首先，`rules` 是一个可以包含多个 `Rule` 对象的元组。`rules` **的意思就是包含了一组从当前返回页面中提取的链接的规则。**

`Rule`

`scrapy.spiders.Rule` 对象定义了爬取页的URL规则与处理方式，它只需要通过构造函数就可以实现，它带有三个重要的参数：

1. 通过链接提取器 `LinkExtractor` 定义链接模式
2. 当符合提取规则时将会调用 `callback` 所指定的函数分析页面内容
3. `follow` 则是向 `Rule` 对象指明，是否继续以此规则深入一页爬网，直至没有找到符合规则的页面为止

`LinkExtractor`

`scrapy.spiders.LinkExtractor` 是一个链接的提取器，它负责从当前返回的响应对象中匹配符合规则的链接元素(`<a>`)。

这个对象说起来很简单，但使用起来却是整个泛爬网蜘蛛的重点。如果你在pyCharm按下 `command` 键 然后点击 `LinkExtractor` 就可以直接跳到它的源码:

```
class LxmlLinkExtractor(FilteringLinkExtractor):
    def __init__(self,
                 allow=(),
                 deny=(),
                 allow_domains=(),
                 deny_domains=(),
                 restrict_xpaths=(),
                 tags=('a', 'area'),
                 attrs=('href',),
                 canonicalize=True,
                 unique=True,
                 process_value=None,
                 deny_extensions=None,
                 restrict_css=(),
                 strip=True):
        # 此处略过
```

你会发现它的构造函数有非常复杂的规则与参数，以下是对这些参数的详细解释:

`allow=()` -顾名思义，要抓取的 url 规则，接收正则表达式，对于字符处理，没有比正则更强大、更迅速的工具了，这里允许接受可迭代对象，也就是写多个规则进去，他会一一提取。

`deny=()` - 不抓取的规则，一般较少会用到，当条件复杂时，可以和 `allow` 配合一起用，前后夹击，参数和 `allow` 一样。

`allowdomains=()` - 这个和 spider 类里的 allowdomains 是一个作用，抓取哪个域名下的网站。

`denydomains=()` - 与 `allowdomains` 正好相反，拒绝爬取哪些域名下的网站。

`restirct_xpaths=()` 与 `restricc_css()` - 在网页哪个区域里提取链接，可以用 xpath 表达式和 css 表达式，这个功能是为了划定提取链接的位置，让提取更加精准。

`tags=('a', 'area')`-默认提取 a 标签和 area 标签

`attrs=('href',)`-默认提取 tags 里的 href 属性，也就是 url 链接。

`canonicalize=True` - url 规范化，没什么实质性作用。

`unique` - 声明地址是不是要规定唯一的，默认是 `True`，重复收集一样的地址的意义也不大。

`strip` - 把地址前后多余的空格删除。

`process_value=None` - 接受一个函数，可以立即对提取到的地址做加工，比如提取用js写的链接

`deny_extensions` - 排除非网页链接，默认是 None，scrapy 会给你排除掉以下链接

在众多的参数中比较强大的可数 `process_value`, 例如,从这段代码中提取链接:

```
<a href="javascript:goToPage('../other/page.html'); return false">Link text</a>
```

你可以使用下面的这个 `process_value` 函数:

```
def process_value(value):
    m = re.search("javascript:goToPage\('(.*?)'", value)
    if m:
        return m.group(1)
```

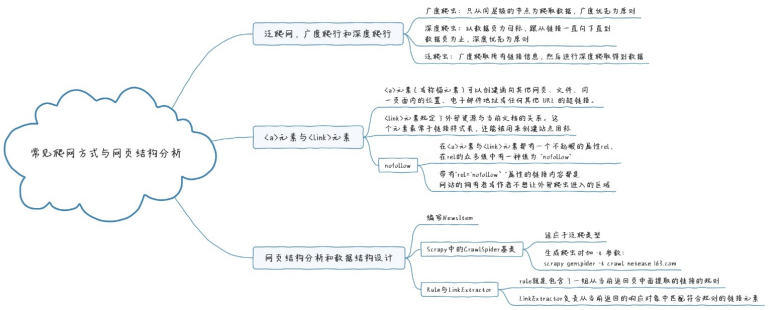
而以上这些参数中最常用的则是 `allow` 这个参数，就以本示例来说我们就需要定义一个这样的链接提取器：

```
LinkExtractor(allow=(r'(\w+):\/\/([^/:]+)\/(\d{2})+\/(\d{4})+\/(\d{2})+\/([^#]*)'))
```

刚刚理清了一个思路又出现一大堆不知道是什么东西的字符串，是不是感觉很崩溃？是的，回想当年我刚接触 scrapy 也是觉得极为崩溃，但现在不一样了，本专栏就是冲着给你解疑的目的来编写的，上面这堆乱七八糟的字符称为 **“正则表达式”**，用好 `LinkExtractor` 的关键就是怎么来制定 **正则表达式**。它涉及的内容非常多，不过一但掌握了它，任何的非结构化文本分析对你将不再是难事，但是由于篇幅关系请恕我要在本节先卖个关子，在下一节我将会以此为重点详细地讲述 **正则表达式** 的用法与规则。

小结

本节的目的是：希望能在你的脑中建立一个 **蜘蛛** 与 **网** 的概念，网络爬虫中的蜘蛛与网的概念就是对真实世界的一种引喻，你作为这些蜘蛛的指挥官，就需要像母皇一样知道如何指挥它们前进，哪里能去，哪里不能去，如何从网络上去觅食。



06 新闻供稿专用爬虫开发实践

08 学习正则表达式—基础入门

精选留言 1

欢迎在这里发表留言，作者筛选后可公开显示

supernicole

这个专栏是scrapy课程的补充说明

0 回复 2019-06-12