第5章

网络数据获取

随着互联网的快速发展，有效提取并利用海量网络数据很大程度上决定了解决问题的效率和质量。传统的通用搜索引擎作为辅助程序员检索信息的工具，无法针对特定的目标和需求进行索引，也无法满足有效数据获取和信息发现的高质量需求。面对结构越来越复杂，信息含量越来越密集的网络数据，为了按照特定需求定向抓取并分析网页资源，实现更高效的指定信息获取、发现和利用，网络爬虫应运而生。

5.1 网络爬虫简介

5.1.1 网络爬虫的定义

网络爬虫(Web Spider)又称网络机器人、网络蜘蛛，是一种根据既定规则，自动提取网页信息的程序或者脚本。传统爬虫以一个或若干初始网页的统一资源定位符(Uniform Resource Location，URL)为起点，下载每一个 URL 指定的网页，分析并获取页面内容，并不断从当前页面抽取新的 URL 放入队列，记录每一个已经爬取过的页面，直到 URL 队列为空或满足设定的停止条件为止。网络爬虫的目的在于将互联网上的目标网页数据下载到本地，保存在数据库中或本地数据文件中，以便进行本地数据文件操作和后续的数据分析。网络爬虫技术的兴起源于海量网络数据的可用性，使用爬虫技术能够较为容易地获取网络数据，通过数据分析得出有价值的结论。

5.1.2 网络爬虫的类型

按照系统结构和实现技术，网络爬虫大致分为四种类型：通用网络爬虫(General Purpose Web Crawler)、聚焦网络爬虫(Focused Web Crawler)、增量式网络爬虫(Incremental Web Crawler)以及深层网络爬虫(Deep Web Crawler)。实际应用中的网络爬虫系统通常是几种爬虫技术相结合实现的。

1. 通用网络爬虫

通用网络爬虫又称全网爬虫，爬行对象从一些种子 URL 扩充到整个 Web

(万维网),主要为门户站点搜索引擎和大型 Web 服务提供商采集数据。这类网络爬虫的爬行范围和数量巨大,对爬行速度和存储空间要求较高,而对爬行页面的顺序要求相对较低,通常采用并行工作方式应对大量待刷新的页面,适合为搜索引擎获取广泛的主题。

通用网络爬虫大致由页面爬行模块、页面分析模块、链接过滤模块、页面数据库、URL 队列和初始 URL 集合等几部分构成。为了提高工作效率,通用网络爬虫可以采用深度优先和广度优先等爬行策略。采用深度优先爬行策略的爬虫按照深度由低到高的顺序,依次访问下一级网页链接,直到不能再深入为止。爬虫在完成一个爬行分支后返回上一个链接节点进一步搜索其他链接。当所有链接遍历完毕,爬行任务结束。这种爬行策略比较适合垂直搜索或站内搜索,但爬行页面内容层次较深的站点时会造成资源的巨大浪费。采用广度优先爬行策略的爬虫按照网页内容目录层次深浅来爬行页面,优先爬取目录层次较浅的页面。当同一层次的页面爬行完毕,再深入下一层次继续爬取。这种爬行策略能够有效控制页面的爬取深度,避免在遇到一个无穷深层分支时无法结束爬行的问题。该策略无须存储大量的中间节点,不足之处是需要较长时间才能爬行到目录层次较深的页面。

2. 聚焦网络爬虫

聚焦网络爬虫又称主题网络爬虫,它会选择性地爬取与预定主题相关的页面。与通用网络爬虫相比,聚焦网络爬虫只需爬取与主题相关的页面,极大地节省了硬件和网络资源,保存页面数量少且更新快,可以更好地满足特定人群对特定领域信息的爬取需求。

页面内容和链接重要性不同导致链接的访问顺序也不一样,因此聚焦爬虫的爬行策略分为以下四种。

基于内容评价的爬行策略将文本相似度计算方法引入网络爬虫中。该策略以用户输入的查询词为主题,将包含查询词的页面视为主题相关页面,其局限性在于无法评价页面与主题相关度的高低,可以尝试利用空间向量模型计算页面与主题的相关度。

页面链接指示了页面之间的相互关系,基于链接结构的搜索策略利用这些结构特征评价页面和链接的重要性,以此决定搜索顺序。其中,PageRank 算法是这类搜索策略的代表,具体做法是每次选择 PageRank 值较大的页面链接进行访问。

基于增强学习的爬行策略将增强学习引入聚焦爬虫,利用贝叶斯分类器,根据网页文本和链接文本对超链接分类,为每个链接计算重要性,从而决定链接的访问顺序。

基于语境图的爬行策略通过语境图学习网页之间的相关度。该策略训练一个机器学习系统,计算当前页面到相关 Web 页面的距离,优先访问距离近的页面链接。

3. 增量式网络爬虫

增量式网络爬虫对已下载的网页采取增量式更新策略,只爬行新产生或已经发生变化的网页,在一定程度上保证爬行尽可能新的页面。与周期性爬行和刷新页面的爬虫相比,增量式爬虫按需爬取新产生或发生更新的页面内容,有效减少了数据下载量并及时更新爬行过的网页,减少时间和空间上的耗费,但是增加了爬行算法的复杂度和实

现难度。

为了保持本地存储的页面为最新页面,增量式爬虫通过监测网页数据的更新情况,持续更新本地的页面内容。采用统一更新法的爬虫以相同的频率访问所有网页,不考虑网页的改变频率。采用个体更新法的爬虫根据个体网页的改变频率重新访问各页面。采用基于分类的更新法的爬虫根据网页改变频率分为更新较快网页子集和更新较慢网页子集两类,然后以不同的频率访问这两类网页。

为了保证爬取的页面质量,增量式爬虫需要对网页的重要性进行排序,常用广度优先策略和 PageRank 优先策略;也可以采用自适应方法,根据历史爬取结果和网页实际变化速度对页面更新频率进行调整;或者将网页分为变化网页和新网页两类,分别采用不同的爬行策略。

4. 深层网页爬虫

Web 页面按照存在方式可以分为表层网页和深层网页两类。表层网页是指传统搜索引擎可以索引到的页面,以超链接可以到达的静态网页为主。深层网页是指隐藏在搜索表单后,大部分内容不能通过静态链接获取,只有用户提交关键词才能获得的 Web 页面。深层网页是目前互联网上最大、发展最快的新型信息资源。

深层网页爬行过程中最重要的部分就是表单填写,表单填写方法可以分为两类。

基于领域知识的表单填写:此方法一般会维持一个本体库,通过语义分析来选取合适的关键词填写表单。一种方法是将数据表单按照语义分配到各个组中,每组从多方面注解,结合各种注解结果预测最终的注解标签;也可以利用一个预定义的领域本体知识库识别深层网页内容,同时利用 Web 站点导航模式自动识别填写表单时所需的路径导航。

基于网页结构分析的表单填写:此方法一般无须领域知识或仅利用有限领域知识,将网页表单表示为文档对象模型(Document Object Model,DOM),从中提取表单字段值。一种方法是将 HTML(HyperText Markup language,超文本标记语言)网页表示为 DOM 树形式,对单属性表单和多属性表单分别处理;也可以将 Web 文档构造成 DOM 树,将文字属性映射到表单字段。

5.1.3 网络爬虫基本架构

网络爬虫主要完成两个任务,即下载目标网页和从目标网页中解析信息。一个简单网络爬虫的基本架构如图 5-1 所示。

图 5-1 网络爬虫基本架构

1. URL 管理模块

URL 管理模块负责管理 URL 链接，维护已经爬行的 URL 集合和计划爬行的 URL 集合，防止重复爬取或循环爬取。其主要功能包括：添加新的 URL 链接、管理已爬行的 URL 和未爬行的 URL 以及获取待爬行的 URL。

URL 管理模块的实现方式有两种：一种是利用 Python 集合数据类型不包含重复元素的特点达到去重效果，防止重复爬取或循环爬取；另一种实现方式是在数据库表的记录中增加一个 URL 标志字段，例如，已爬行的网页链接标记为“1”，未爬行的网页链接标记为“0”。当有新链接产生时，先在已爬行的链接集中查询，如果发现该链接已被标记为“1”，那么不再爬行该 URL 的链接页面。

2. 网页下载模块

这是网络爬虫的核心组件之一，用于从 URL 管理模块获取待爬行的 URL，并将对应的页面内容下载到本地，或者以字符串形式读入内存，方便后续使用字符串相关操作解析网页内容。

Python 第三方库 requests 是一个处理 HTTP(Hyper Text Transfer Protocol，超文本传输协议)请求的模块，其最大优点是程序编写过程更接近正常的 URL 访问过程。

3. 网页解析模块

网页解析模块是网络爬虫的另一个核心组件，用于从网页下载模块获取已下载的网页，并解析出有效数据交给数据存储器。网页解析的实现方式多种多样。由于下载到本地的网页内容以字符串形式保存，可以使用字符串相关操作从中解析出有价值的结构化数据，例如，可以使用正则表达式指定规则，然后根据规则找出感兴趣的字符串；也可以使用 Python 自带的 HTML 解析工具 html.parser 从网页内容的字符串中解析出相关信息；还可以使用 Python 第三方库 beautifulsoup4 实现网页解析。作为一种功能强大的结构化网页解析工具，beautifulsoup4 模块能够根据 HTML 和 XML(Extensible Markup Language，可扩展标记语言)语法建立解析树，进而高效解析和处理页面内容。

4. 数据存储器

数据存储器负责将网页解析模块解析出的数据存储起来，用于后续的数据分析和信息利用。

5.2　网页下载模块

网页下载模块将 URL 对应的网页下载到本地或读入内存。Python 提供了第三方库 requests 访问一个指定的 URL，返回有用的数据。

5.2.1 requests 库简介

requests 是一个处理 HTTP 请求的 Python 第三方库,需要预先安装。requests 模块在 Python 内置模块的基础上进行了高度封装,使得进行网络请求时更加简洁和人性化。

requests 库支持丰富的链接访问功能,包括 HTTP 长链接和链接缓存、国际域名和 URL 获取、HTTP 会话和 Cookie 保持、浏览器使用风格的 SSL 验证、自动内容解码、基本摘要身份验证、有效键值对的 Cookie 记录、自动解压缩、Unicode 响应主体、HTTP(S)代理支持、文件分块上传、流式下载、连接超时和分块请求等。

5.2.2 requests 库的使用

1. requests 库的网页请求方法

通过 URL 访问网络链接并返回网页内容是 requests 模块的基本功能,其中与网页请求相关的函数有 6 个,具体使用方法如表 5-1 所示。

表 5-1 requests 模块的网页请求函数

函 数	说 明
get(url[,timeout=n])	对应 HTTP 的 GET 方式,获取网页最常用的方式,可选参数 timeout 设定每次请求超时时间,单位为秒
post(url,data={'key':'value'})	对应 HTTP 的 POST 方式,其中字典用于传递客户数据
delete(url)	对应 HTTP 的 DELETE 方式
head(url)	对应 HTTP 的 HEAD 方式
options(url)	对应 HTTP 的 OPTIONS 方式
put(url,data={'key':'value'})	对应 HTTP 的 PUT 方式,其中字典用于传递客户数据

requests.get()函数向目标网址发送请求,接收响应,返回一个 response 对象。这里的参数 url 必须采用 HTTP 或 HTTPS 方式访问。

```
In [1]: import requests
        url = 'https: //www.baidu.com/'
        r_obj = requests.get(url)
        type(r_obj)
Out[1]: requests.models.Response
```

2. response 对象

调用 requests.get()函数后,返回的网页内容保存为一个 response 对象。response 对象的常用属性如表 5-2 所示。

表 5-2　response 对象的常用属性

属　性	说　明
status_code	HTTP 请求返回的状态码，为整数，200 表示连接成功，404 表示连接失败，500 表示内部服务器错误等
headers	HTTP 响应内容的网页 header 信息
encoding	HTTP 响应内容的编码形式
text	HTTP 响应内容的字符串形式，即 url 对应的页面内容
content	HTTP 响应内容的二进制形式

调用 requests.get()函数后，可以使用 response.status_code 属性返回 HTTP 请求之后的状态，如果请求未被响应，需要中止内容处理；否则系统返回一个 response 对象，其中存储了服务器的响应内容。大多数情况下，requests 用户可以使用 response.text 获取文本形式的响应内容，requests 自动解析服务器内容；也可以使用 response.encoding 属性返回页面内容的编码方式，可以为 encoding 属性赋值更改编码方式，便于处理中文字符。实际上，requests 也可以基于 HTTP 头部信息对相应编码做出有根据的推测，使用正确的编码方式访问 response.content，以字节形式直接保存返回的二进制数据。

```
In [2]: r_obj.status_code                    #状态码
Out[2]: 200
In [3]: r_obj.headers                        #网页 header 信息
Out[3]: {'Cache-Control': 'private, no-cache, no-store, proxy-revalidate, no
-transform', 'Connection': 'keep-alive', 'Content-Encoding': 'gzip', '
Content-Type': 'text/html', 'Date': 'Fri, 05 Feb 2021 01: 52: 06 GMT', 'Last-
Modified': 'Mon, 23 Jan 2017 13: 23: 55 GMT', 'Pragma': 'no-cache', 'Server': '
bfe/1.0.8.18', 'Set-Cookie': 'BDORZ=27315; max-age=86400; domain=.baidu.
com; path=/', 'Transfer-Encoding': 'chunked'}
In [4]: r_obj.encoding                       #网页编码
Out[4]: 'ISO-8859-1'
In [5]: r_obj.text                           #请求返回的文本信息，出现乱码
Out[5]: '<!DOCTYPE html>\r\n<!--STATUS OK--><html><head><meta http-equiv=
content-type content=text/html;charset=utf-8><meta http-equiv=X-UA-Compatible
content=IE=Edge><meta content=always name=referrer><link rel=stylesheet type=
text/css href=https: //ss1.bdstatic.com/5eN1b jq8AAUYm2zgoY3K/r/www/cache/bdorz/
baidu.min.css><title>ç\x993/4å°|ä.\x80ä.\x8bï1/4\xa0ä°±ç\x9f¥é\x81\x93</title>
</head><body link=#0000 cc><div id=wrapper><div id=head>…
In [6]: r_obj.encoding ='utf-8' #重新设置网页编码，使用 utf-8 编码
Out[6]: '<!DOCTYPE html>\r\n<!--STATUS OK--><html><head><meta http-equiv
=content-type content=text/html;charset=utf-8><meta http-equiv=X-UA-
Compatible content=IE=Edge><meta content=always name=referrer><link rel=
stylesheet type=text/css href=https: //ss1.bdstatic.com/ 5eN1b jq8AAUYm2-
zgoY3K/r/www/cache/bdorz/baidu.min.css><title>百度一下，你就知道</title></
head><body link=#0000cc><div id=wrapper><div id=head>… … … …
```

```
In [7]: r_obj.content                                    #以字节形式返回的非文本信息
Out[7]: b'<!DOCTYPE html>\r\n<!--STATUS OK--><html><head><meta http-equiv
=content-type content=text/html;charset=utf-8><meta http-equiv=X-UA-
Compatible content=IE=Edge><meta content=always name=referrer><link rel=
stylesheet type=text/css href=https://ss1.bdstatic.com/5eN1
bjq8AAUYm2zgoY3K/r/www/cache/bdorz/baidu.min.css><title>\xe7\x99\xbe\xe5\
xba\xa6\xe4\xb8\x80\xe4\xb8\x8b\xef\xbc\x8c\xe4\xbd\xa0\xe5\xb0\xb1\xe7\x9f\
xa5\xe9\x81\x93</title></head><body link=#0000cc>…… …… …… ……
```

除了属性,response 对象还提供了一些方法,如表 5-3 所示。

表 5-3 response 对象的常用方法

方 法	说 明
json	如果 HTTP 响应内容包含 JSON 格式数据,则解析 JSON 数据
raise_for_status()	如果状态码不是 200,则抛出异常

response 对象的 json()方法解析 HTTP 响应内容中的 JSON 数据;response 对象的 raise_for_status()方法可以用于 try…except…结构,在非成功响应后抛出异常,避免状态码 200 以外的各种意外。例如,遇到 DNS 查询失败、拒绝连接等,requests 会抛出 ConnectionError 异常;遇到无效响应时,requests 会抛出 HTTPError 异常;如果请求 url 超时,会抛出 Timeout 异常;请求超过了设定的最大重定向次数,会抛出 TooManyRedirect 异常,等等。

例 5-1 编写一个获取网页内容的函数。

```
In [1]: import requests
        def getHTMLText(url):
            try:
                r_obj=requests.get(url,timeout=30)
                r_obj.raise_for_status()
                r_obj.encoding='utf-8'
                return r_obj.text
            except:
                return ""
        url="http://www.baidu.com"
        print(getHTMLText(url))
Out[1]: <!DOCTYPE html>
        <!--STATUS OK--><html><head><meta http-equiv=content-type content
    =text/html;charset=utf-8><meta http-equiv=X-UA-Compatible content=IE
    =Edge><meta content=always name=referrer><link rel=stylesheet type=
    text/css href=http://s1.bdstatic.com/r/www/cache/bdorz/baidu.min.css><
    title>百度一下,你就知道</title></head><body link=#0000cc><div id=
    wrapper><div id=head><div class=head_wrapper>…… …… …… ……
```

5.3　网页解析模块

使用 requests 模块获取 HTML 页面并将其转换成字符串之后，需要进一步解析 HTML 页面格式，提取有用信息，这需要解析和处理 HTML、XML 的函数库。

5.3.1　beautifulsoup4 库简介

Python 第三方库 beautifulsoup4（也称 BeautifulSoup 或 bs4 库）用于解析和处理 HTML、XML 文件并提取数据。beautifulsoup4 支持多种解析器，其优势是能够根据 HTML 和 XML 语法建立解析树，进而高效解析其中的内容，为用户提供需要的数据。

在使用之前，需要预先安装第三方库 beautifulsoup4。在 Scripts 文件夹下打开命令提示符窗口，使用命令 pip install beautifulsoup4 进行安装。

安装完成后，需要在 Python 解释器中导入 beautifulsoup4，可以使用 from…import 方式从 bs4 模块直接引用 BeautifulSoup 类，方法如下。

```
In [1]: from bs4 import BeautifulSoup
```

然后，就可以使用 beautifulsoup4 模块提供的函数和方法处理导航、搜索、修改分析树等一系列操作。

5.3.2　文档对象模型

HTML 建立的 Web 页面一般比较复杂，除了有用数据之外，还包含大量用于页面格式的元素。一个网页文件通常可以表示为一个文档对象模型（Document Object Model，DOM）。DOM 是一种处理 HTML 和 XML 文件的标准编程接口，它提供了对整个文档的访问模型，将网页文档表示为一个树形结构，树的每个节点表示一个 HTML 标签（Tag）或标签内的文本项。DOM 树结构精确地描述了 HTML 文档中标签间的关联性，如图 5-2 所示。

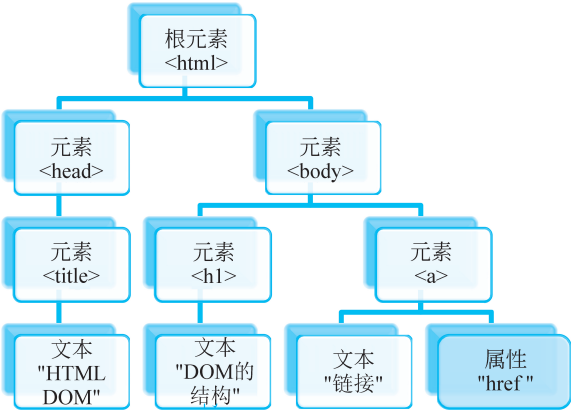

图 5-2　文档的 DOM 树结构

将 HTML 或 XML 文档转换为 DOM 树的过程称为解析。HTML 文档被解析后,转换为 DOM 树,因此对 HTML 文档的处理可以通过对 DOM 树的操作实现。DOM 树不仅描述了文档的结构,还定义了节点对象的行为。利用节点对象的方法和属性,可以方便地访问、修改、添加和删除 DOM 树的节点和内容。

Python 第三方库 beautifulsoup4 将复杂 HTML 文档转换成一个树形结构,将专业的 Web 页面格式解析部分封装成函数。这些方便有效的处理函数为用户解析和处理 HTML、XML 文件提供了便捷。

5.3.3 创建 BeautifulSoup 对象

导入 bs4 库的 BeautifulSoup 类之后,通过 BeautifulSoup 类创建一个 BeautifulSoup 对象。实例化的 BeautifulSoup 对象相当于一个页面,表示一个文档的全部内容。

```
In [1]: import requests
        from bs4 import BeautifulSoup
        url = 'http: //www.baidu.com'
        r_obj = requests.get(url)
        bs = BeautifulSoup(r_obj.content, from_encoding='utf-8')
        type(bs)
Out[1]: bs4.BeautifulSoup
```

BeautifulSoup 对象是一个树状结构,它包含 HTML 页面的每一个标签(Tag)元素,如<head>、<body>等。也就是说,HTML 的主要结构都成为 BeautifulSoup 对象的属性。表 5-4 列出了 BeautifulSoup 对象的常用属性。

表 5-4 BeautifulSoup 对象的常用属性

属 性	说 明
head	HTML 页面的<head>内容
title	HTML 页面标题,在<head>中,由<title>标记
body	HTML 页面的<body>内容
p	HTML 页面中第一个<p>内容
strings	HTML 页面所有呈现在 Web 上的字符串,即标签的内容
stripped_strings	HTML 页面所有呈现在 Web 上的非空格字符串

```
In [2]: bs.head
Out[2]: <head><meta content="text/html;charset=utf-8" http-equiv="content
-type"/><meta content="IE=Edge" http-equiv="X-UA-Compatible"/><meta
content="always" name="referrer"/><link href="http://s1.bdstatic.com/r/
www/cache/bdorz/baidu.min.css" rel="stylesheet" type="text/css"/><title>百
度一下,你就知道</title></head>
```

```
In [3]: bs.title                    #每一个对应 HTML Tag 的属性是一个 Tag 类型
Out[3]: <title>百度一下,你就知道</title>
In [4]: type(bs.title)
Out[4]: bs4.element.Tag
In [5]: bs.p
Out[5]: <p id="lh"><a href="http://home.baidu.com">关于百度</a><a href="
http://ir.baidu.com">About Baidu</a></p>
```

许多情况下,可以将 BeautifulSoup 对象看作 Tag 对象,它支持遍历 DOM 树和搜索 DOM 树的大部分方法。创建 BeautifulSoup 对象之后,可以使用 BeautifulSoup 对象的方法查找和解析网页。

5.3.4　查询节点

当需要列出标签对应的所有内容或者需要找到非第一个标签时,可以使用 BeautifulSoup 对象的 find()和 findall()方法。这两个方法可以遍历整个 HTML 文档,依据查找条件返回标签内容。

1. find()方法

find()方法实现指定范围内的单次条件定位,目的是找到满足条件的第一个节点,返回第一个匹配的对象。

find()方法语法格式如下:

```
find(name, attrs, recursive, string)
```

参数 name:标签名,可以是字符串类型,定位到指定标签名的节点;也可以是列表类型,用于匹配多个标签名;还可以是正则表达式,用于传递自定义的标签名规则;找到后返回一个 BeautifulSoup 标签对象。

参数 attrs:标签的属性,以字典类型指定标签的属性名及属性值,查找其第一次出现的位置,找到后返回一个 BeautifulSoup 标签对象。

参数 recursive:设置查找层次,布尔类型数据,默认值为 True,表示当前标签下的所有子孙标签;如果设置为 False,表示只查找当前标签下的直接子标签。

参数 string:查找标签的文本内容,而不使用标签的属性去匹配。参数可以是字符串类型,字符串列表等,搜索指定范围的字符串内容,返回匹配字符串的列表。

find()方法的参数相当于过滤器,可以对页面内容进行筛选处理。依据 find()方法的参数查找满足条件的标签,返回找到的第一个节点信息。

```
In [6]: bs.find('title')                                    #查找 title 标签
Out[6]: <title>百度一下,你就知道</title>
In [7]: link_tag =bs.find('a')                              #查找第一个链接标签
Out[7]: <a class="mnav" href="http://news.baidu.com" name="tj_trnews">新闻
</a>
```