第 5 章 网络爬虫技术

本章首先介绍网络爬虫的发展历程、爬取的一般流程、网络爬虫的 4 种主要类型和特点以及实现爬取过程需要用到的 Python 库函数和框架，然后介绍网络爬虫的前沿技术和网站反爬虫技术的最新进展，最后通过爬取智联招聘网站上北京市的相关岗位信息分析几种不同岗位的薪资情况、工作经验要求及学历要求。[①]

5.1 概述

5.1.1 网络爬虫的概念内涵

网络爬虫(web crawler)，通常简称为爬虫，是搜索引擎的重要组成部分。随着信息技术的飞速进步，网络爬虫作为搜索引擎的一个组成部分一直是研究的热点，它的发展会直接决定搜索引擎的未来。目前，网络爬虫的研究包括 Web 搜索策略特点的研究和网络分析算法两个方向。其中，Web 爬虫网络搜索主题是一个研究方向，根据一些网络分析算法，过滤不相关的链接，连接到合格的网页，并放置在一个队列中，由网络爬虫抓取。

如果把互联网比喻成一张蜘蛛网，那么网络爬虫就是在这张蜘蛛网上爬来爬去的蜘蛛。网络爬虫通过网页的链接地址寻找网页，从网站某个页面(通常是首页)开始，读取网页的内容，找到网页中的其他链接地址，然后通过这些链接地址寻找下一个网页，这样一直循环下去，直到把此网站所有的网页都抓取完为止。如果把整个互联网当成一个网站，那么网络爬虫就可以用上述原理把互联网上所有的网页都抓取下来。

5.1.2 网络爬虫的技术发展

1989 年，万维网诞生。从技术上讲，万维网和因特网有所不同，前者是指信息空间，后者是指由数台计算机连接起来的物理网络。万维网有以下 3 个主要的技术创新：

(1) 统一资源定位器(Uniform Resource Locator，URL)，用户通过它访问网站。

(2) 内嵌的超链接，用于在网页之间导航。例如，在产品详情页中可以找到产品规格和许多其他信息，例如“购买此产品的顾客也购买了某某商品”，这些信息都是以超链接的形式提供的。

(3) 网页不仅包含文本，还包含图像、音频、视频和软件组件。

---

① 本章由李育霖整理，部分内容由吴志伟、单则安、陈曦、倪俊峰、刘杰龙、陈思益贡献。

1990 年，第一个网络浏览器诞生，它也是由万维网的发明者 Tim Berners-Lee 发明的。

1991 年，第一个网页服务器和第一个 HTTP 协议（以 http://开头）的网页出现，网页的数量以平缓的速度增长。到 1994 年，HTTP 服务器的数量超过 200 台。

1993 年 6 月，第一个网页机器人——万维网漫游器诞生，虽然它的功能和今天的网页机器人一样，但它只用来测量网页的大小。1993 年 12 月，首个基于网络爬虫的网络搜索引擎——JumpStation 诞生。由于网络上的网站并不多，网络搜索引擎过去常常由网站管理员收集和编辑链接。JumpStation 带来了新的飞跃，它是第一个依靠网络机器人的网络搜索引擎。从那时起，人们开始使用这些程序化的网络爬虫程序收集和组织因特网。从 Infoseek、AltaVista 和 Excite 到如今的必应和谷歌，搜索引擎机器人的核心依然保持不变：找到一个网页，下载（获取）它，抓取网页上显示的所有信息，然后将其添加到网络搜索引擎的数据库中。由于网页是为人类用户设计的，因此，即使开发了网页机器人，计算机工程师和科学家仍然很难进行网络数据抓取，更不用说普通人了。因此，人们一直致力于使网络爬虫变得更加容易使用。

2000 年，网页 API 和 API 爬虫被发明出来。API 表示 Application Pragram Interface，即应用程序编程接口。它是一个接口，通过提供搭建好的模块，使开发程序更加便捷。2000 年，Salesforce 和 eBay 推出了自己的 API，程序员可以用它访问并下载一些公开数据。从那时起，许多网站都提供网页 API 让人们可以访问它们的公共数据库。用户发送一组 HTTP 请求，然后接收 JSON 或 XML 的回馈。网页 API 通过收集网站提供的数据，为程序员提供了一种更友好的网络爬虫运行方式。

不是所有的网站都提供 API。即使它们提供了，它们也不一定会提供用户想要的所有数据。因此，程序员仍需开发能够完善网络爬虫的方法。2004 年，Beautiful Soup 发布。它是一个为 Python 设计的库。在计算机编程中，库是脚本模块的集合，就像常用的算法一样，它允许程序员直接使用，从而简化了编程过程。通过简单的命令，Beautiful Soup 可以理解网站的结构，并帮助用户从 HTML 容器中解析内容。它被认为是用于网络爬虫的最复杂和最先进的库，也是当今常见和流行的方法之一。

2006 年，Kapow 软件公司发布了 Kapow 网页集成平台 6.0 版本，这是一种可视化的网络爬虫软件，它允许用户轻松简单地选择网页内容，并将这些数据构造成可用的 Excel 文件或数据库。最终，可视化的网络数据抓取软件可以让大量非程序员用户自己运行网络爬虫。从那时起，网络抓取开始成为主流。现在，对于非程序员用户来说，他们可以找到 80 多个提供可视化过程的数据采集软件。

5.1.3 网络爬虫的爬取过程

网络爬虫的爬取过程如图 5-1 所示，主要分为以下 5 步：

（1）分析目标网站。明晰目标网站结构，查清关键数据位置。

（2）发送请求。使用 HTTP 库或浏览器模拟工具向目标网站发送请求。

（3）获取响应。如果得到了一个响应，说明目标网站处于正常状态。

（4）解析响应内容。解析 HTML 数据和 JSON 数据，从而获得关键数据信息或者下一个待爬取的 URL 地址。

（5）保存数据。

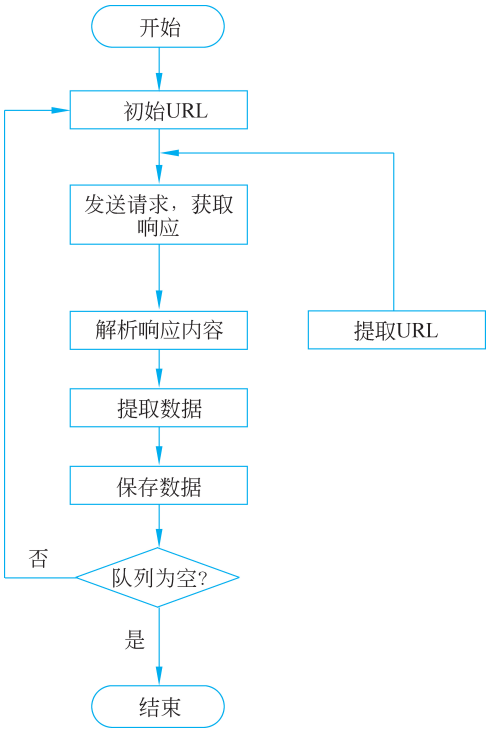

图 5-1　网络爬虫的爬取过程

5.2 网络爬虫分类

5.2.1 通用网络爬虫

通用网络爬虫又称全网爬虫。其爬行对象从一些种子 URL 扩充到整个 Web，主要为门户站点搜索引擎和大型 Web 服务提供商采集数据。由于商业原因，它们的技术细节很少公布。这类网络爬虫的爬行范围和网页数量巨大，对于爬行速度和存储空间要求较高，对于爬行网页的顺序要求较低。同时，由于待刷新的网页太多，这类网络爬虫通常采用并行工作方式，但需要较长时间才能刷新一次网页。通用网络爬虫虽然存在一定缺陷，但是它可以为搜索引擎搜索广泛的主题，有较强的应用价值。

通用网络爬虫对于网页爬取有两种策略：

（1）广度优先策略。按照网页内容目录层次深浅的顺序爬行网页，处于较浅目录层次的网页首先被爬行。当同一层次中的网页爬行完毕后，网络爬虫再深入下一层继续爬行。这种策略的优势在于能够有效控制网页的爬行深度，避免遇到一个无穷深层分支时无法结束爬行的问题，并且实现方便，可以存储大量中间节点，以便后续的信息提取。但这种策略需要较长时间才能爬行到目录层次较深的网页。

（2）深度优先策略。按照深度由浅到深的顺序，依次访问下一级网页链接，直到不能再深入为止。网络爬虫在完成一个爬行分支后返回到上一链接节点进一步搜索其他链接。当所有链接遍历完毕后，爬行任务结束。这种策略非常适合垂直搜索或站内搜索。但由于

爬取顺序的原因，这种策略爬行网页内容层次较多的网站时会造成资源的巨大浪费。

5.2.2 深层网络爬虫

网页按存在方式可以分为表层网页和深层网页。表层网页是指传统搜索引擎可以直接发现的网页，是以超链接可以到达的静态网页为主构成的网页。深层网页是那些大部分内容不能通过静态链接获取，隐藏在搜索表单后，只有用户提交一些关键词才能获取的网页。

深层网络爬虫的爬取步骤如下：

（1）自动查找深层网页入口点。爬取深层网页需要填写搜索表单（这些表单构成深层网页的入口点），并访问这些搜索表单背后的信息。入口点即网站中包含允许访问深层网页的搜索表单。

（2）Form 建模。由于搜索表单是为人机交互设计的，因此对深层网页的访问需要用一个自动化过程模拟这种交互。与搜索表单交互的任何代理都需要完成两个任务：表单建模和查询选择。表单建模涉及理解搜索表单，以便能够识别与每个字段相关的语义、每个字段中预期值的类型和值以及字段之间的关系。查询选择涉及生成适当类型和域的值的组合，以填充每个字段。这两个任务都可以使用非监督或半监督技术实现自动化。

（3）学习爬行路径。深层网页爬虫的工作并不局限于填写表单和返回结果页面。在某些情况下，它们需要进一步深入 Web，在导航站点中找到与使用网络爬虫的用户或进程相关的网页子集。根据爬取这些网页子集的目的，可以将这些网络爬虫分为普通网络爬虫、聚焦网络爬虫和热点网络爬虫。

5.2.3 聚焦网络爬虫

1. 聚焦网络爬虫的系统结构

通用爬虫只能提供粗略的信息。聚焦网络爬虫主题明确且能够精准地获取有效信息。聚焦网络爬虫在存储网页 URL 时需要判断该 URL 与主题的相关性，尽可能地筛选出与主题相关的网页。聚焦网络爬虫的工作包括网页获取、网页过滤、网页存储和网页分析。

（1）网页获取。模拟客户端发送 HTTP 请求，获取服务器端的响应后下载网页，完成聚焦网络爬虫系统的爬取工作。

（2）网页过滤。筛选与主题有关的 URL，抓取与主题相关的网页，确保聚焦网络爬虫系统的准确率。

（3）网页存储。将网页解析模块解析出来的数据以文件或数据库的形式存储起来。

（4）网页分析。包括两部分，第一部分是主题相关度判断，第二部分是主题相关度预测。

2. 聚焦网络爬虫相似度判别算法

聚焦网络爬虫相似度判别算法分为两类，分别是向量空间模型算法以及语义相似度算法。向量空间模型算法将文本处理转换为向量空间中的向量运算，将每一篇文档表示为向量空间中的某一向量，通过计算向量在向量空间中的接近程度衡量文档之间的相似度。语

义相似度算法是以词频和文档频率这两个量为基础的算法，使得计算机能够从语义角度判断文档的相似度。

3. 聚焦网络爬虫搜索策略

聚焦网络爬虫的搜索策略分为两类，分别是静态搜索策略和动态搜索策略。

静态搜索策略依照确定的规则进行搜索，不会因为网页结构、文本信息的改变而改变，主要包括广度优先搜索、深度优先搜索和最佳优先搜索。

动态搜索策略以高效、快速完成爬取任务为第一宗旨，实时调整搜索路线。动态搜索策略会实时根据 URL 的主题相关度进行调整，主要包括基于文本内容的 Fish-Seach、Shark-Search 以及基于链接分析的 PageRank、HITS、HillTop。

下面以 Fish-Search 为例，其算法描述如下：

首先将与主题相关的种子链接放入待爬 URL 队列中。若当前网页与主题相关，将该网页的前 $a \times$width 个链接放入待爬 URL 队列顶部，以增加爬行的宽度，其中参数 a 和 width 都是给定的初始值；若当前页面与主题无关，将该网页的前 width 个链接放入待爬 URL 队列中部，即与主题相关的链接的后面；若是其他情况，将该网页的子链接放入待爬 URL 队列的尾部，当有充足的时间时才对这些链接进行爬取。对于主题相关性的描述，可以定义一个变量 potential_score，当网页与主题相关时，potential_score 设为 1；当网页与主题无关时，potential_score 设为 0.5；其他情况 potential_score 设为 0。最后按 potential_score 的值对待爬 URL 队列中的 URL 进行排序。

4. 聚焦网络爬虫研究方向

聚焦网络爬虫目前有基于网页内容和基于链接分析两种类型。

目前，基于网页内容的聚焦网络爬虫计算文本相似度的方法大致分为两类：一是基于字词统计模型，如向量空间模型；二是基于语义理解模型。研究人员希望使用语义相关性使网络爬虫可以获得更精确的结果。在整个文本相似度判别过程中，首先确定网络爬虫的主题，再根据网页内容、结构信息计算网页主题相关度和抓取 URL 的相关度，依据网页主题相关度确定要抓取的链接和抓取的优先级。此类网络爬虫通常能获得较高的准确率。

互联网中数十亿个网页通过万维网上的链接关联起来，研究人员试图通过有效的方式获取链接上下文的含义，从而对链接上下文进行解析和提取，或者基于网页内容对传统链接选择算法加以改进，使网络爬虫采集的准确度提升。该类算法通过分析网页链接判断网页的重要性，强调了页面链接的权威性对用户需求的意义，同时通过将网页正文、链接锚文本以及锚文本上下文的网页内容分析和链接分析结合解决了“主题漂移”问题，提高了主题爬取的准确性。

5. 聚焦网络爬虫的应用

为了弥补通用搜索引擎的不足，实现对特定主题信息的检索，出现了垂直搜索引擎，它检出的结果更准确，挖掘信息的层次更深，无效信息更少，更能适应垂直领域的服务。垂直搜索引擎是面向特定领域、为特定用户服务的一种搜索引擎，是对专业领域信息的深层次挖掘，它将信息经过过滤、筛选、梳理后集成在一起，为用户提供了面向专业知识的检索。

垂直搜索引擎与全文搜索引擎在工作原理上类似，区别在于抓取模块中的爬虫程序与主题词库不同。

5.2.4 增量式网络爬虫

1. 增量式网络爬虫的爬取步骤

增量式网络爬虫的爬取过程如图 5-2 所示，具体可分成以下 3 步：

（1）在发送请求之前判断此 URL 是否曾爬取过（适合不断有新网页的网站）。

（2）在解析内容后判断这部分内容是否曾爬取过（适合网页内容定时更新的网站）。

（3）写入存储介质时判断内容是否已存在于介质中（最大限度达到去重的目的）。

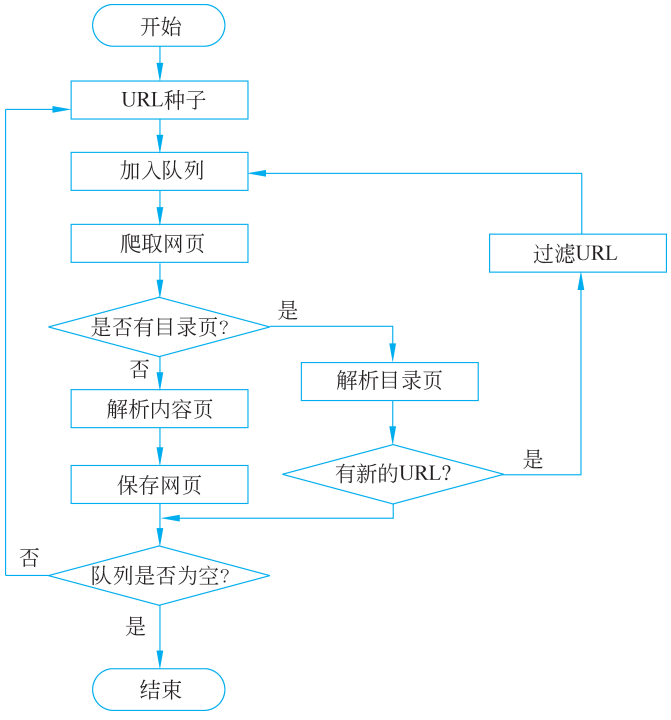

图 5-2 增量式爬虫的爬取过程

2. 增量式网络爬虫去重方法

增量式网络爬虫去重有两种方法：

（1）将爬取过程中产生的 URL 存储在 Redis 的 set 中。当下次进行爬取时，首先判断即将发起的请求所对应的 URL 是否已经存在于 URL 的 set 中，如果存在则不发送请求，否则发送请求。

（2）对爬取到的网页内容指定唯一标识（数据指纹），然后将该唯一标识存储在 Redis 的 set 中。当下次爬取到网页数据的时候，在进行持久化存储之前，先判断该数据的唯一标识在 Redis 的 set 中是否存在，从而决定是否对其进行持久化存储。

基于 Redis 的 Bloomfilter 去重方法既发挥了 Bloomfilter 的海量去重能力，又发挥了

Redis 的可持久化能力。Bloomfilter 是一个很长的二进制向量和一系列随机映射哈希函数。通常辨别某个元素是否在集合中的常用方法是将该元素和集合中的元素逐一进行对比。Bloomfilter 能够在较短时间内检查某一元素是否在集合中。

5.3 网络爬虫库与框架

常用的网络爬虫库按照爬取环节可分为 3 类：网页爬取库、网页分析库和数据存储库。常用的网络爬虫框架主要有 3 个：Scrapy 框架、PySpider 框架和 feapder 框架。

5.3.1 网络爬虫库

1. 网页爬取库

在网页爬取环节，主要是为了实现 HTTP 请求，读取 URL 地址中指定的网络资源并保存到本地，常用的网页爬取库包括 urllib、requests、selenium、pyppeteer 和 aiohttp 等。selenium 最初是一个自动化测试工具，在网络爬虫中使用它主要是为了解决 requests 无法执行 JavaScript 代码，而所需的信息往往就藏在 JavaScript 代码中的问题。selenium 本质上是通过驱动浏览器，完全模拟浏览器的操作，例如跳转、输入、点击、下拉等，进而获取网页渲染之后的结果。pyppeteer 使用了 Python 异步协程库 asyncio，可整合 Scrapy 进行分布式爬取。pyppeteer 虽然支持的浏览器比较单一，但在安装配置的便利性和运行效率方面都远胜 selenium。urllib 和 requests 都是阻塞式 HTTP 请求库，即当发送一个请求后，程序会一直等待服务器响应，直到服务器响应后，程序才会进行下一步处理；而 aiohttp 能够实现异步 Web 请求，大大提高了网页爬取的效率。下面主要介绍 urllib 和 requests。

1）urllib

urllib 是 Python 内置的 HTTP 请求库，它包含如下 4 个模块：

（1）request。HTTP 请求模块，可以用来模拟发送请求。只需要给库方法传入 URL 以及相应的参数，就可以模拟在浏览器地址栏输入网址然后按 Enter 键的操作。

（2）error。异常处理模块，当出现请求错误时可以捕获这些异常，然后进行重试或其他操作，以保证程序不会意外中止。

（3）parse。工具模块，提供了许多 URL 处理方法，例如拆分、解析、合并等。

（4）robotparser。网站识别模块，主要用来识别网站的 robots.txt 文件，然后判断哪些网站可以爬取。

2）requests

requests 是 Python 的第三方库，它是对 urllib 的进一步封装，因此在使用上显得更加便捷，在实际应用中使用最多的也是 requests。其功能特性主要有 p-Alive 和连接池、国际化域名和 URL、带持久 Cookie 的会话、浏览器式的 SSL 认证、自动内容解码、基本/摘要式身份认证、键/值对 Cookie、自动解压、Unicode 响应体、HTTP/HTTPS 代理支持、文件分块上传、流下载、连接超时、分块请求、支持.netrc。

2. 网页分析库

网页分析环节是爬虫的关键步骤，需要从 XML 或 HTML 中定位并提取需要的信息。

常用的网页分析库包括 lxml、Beautiful Soup、re 和 PyQuery。PyQuery 是 JQuery 的 Python 实现，能够以类似于 JQuery 的语法解析 HTML 文档，并且易用性和解析速度都很好。下面主要介绍 lxml 和 Beautiful Soup。

1）lxml

lxml 是 XML 和 HTML 的解析器，其主要功能是解析和提取 XML 和 HTML 中的数据。lxml 和正则表达式类似，还可以利用 XPath 语法定位特定的元素及节点信息。而且它与人们熟知的 ElementTree API 兼容。

XPath 全称为 XML Path Language，即 XML 路径语言，最初用来搜寻 XML 文档，但同样适用于 HTML 文档的搜索，所以在网络爬虫中完全可以使用 XPath 进行相应的信息抽取。

2）Beautiful Soup

Beautiful Soup 可以将复杂的 HTML 或 XML 文档转换成一个复杂的树状结构（文档树），树上每个节点都是一个 Python 对象。所有对象可以归纳为 4 种类型：Tag、NavigableString、Beautiful Soup 和 Comment。

（1）Tag：HTML 文档中的标签。可以利用 soup 加标签名获取这些标签的内容，这些对象的类型是 Bs4、Element.Tag。

（2）NavigableString：标签中的内容。

（3）Beautiful Soup：表示的是一个文档的全部内容，多数情况下可以把它当作 Tag 对象，它支持遍历文档树和搜索文档树中描述的大部分方法。

（4）Comment：文档的注释。Comment 对象是一个特殊类型的 NavigableString 对象。

3. 数据存储库

数据存储库环节是存储已经提取出来的信息的数据库，例如 MySQL、MongoDB 及 Redis 等。它可以对数据进行持久化存储。常用的数据存储库有 PyMysql、PyMongo 及 Redis 等。

5.3.2 网络爬虫框架

1. Scrapy 框架

Scrapy 框架是一个快速、高层次的屏幕抓取和 Web 抓取框架，用于抓取 Web 网站并从网页中提取结构化数据。Scrapy 吸引人之处在于它是一个框架，用户可以根据需要方便地对它进行修改。它也提供了多种类型的网络爬虫（如 BaseSpider、Sitemap 等）的基类。Scrapy 框架的架构如图 5-3 所示。

（1）Scrapy 引擎（Scrapy engine）：是 Scrapy 框架的核心，用来处理整个系统的数据流和触发事务。

（2）调度器（scheduler）：接收 Scrapy 引擎发送的请求并将其加入队列中，在 Scrapy 引擎需要再次发送该请求的时候将其提供给 Scrapy 引擎。

（3）下载器（downloader）：是所有组件中负载最大的，用于高速下载网络上的资源。

（4）网络爬虫：该模块帮助用户定制自己的网络爬虫（通过正则表达式等语法），以便

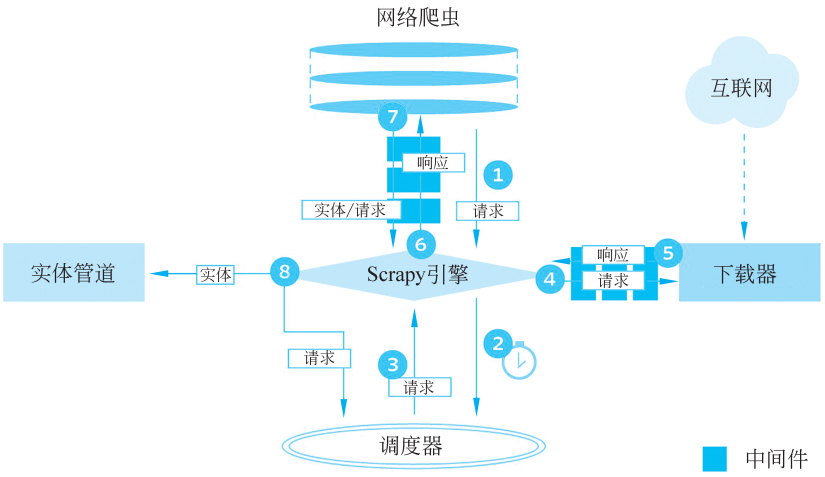

图 5-3　Scrapy 框架的架构

从特定的网页中提取用户需要的信息。

(5) 实体管道(item pipeline)：用于处理网络爬虫提取的实体。主要功能是持久化实体、验证实体的有效性以及清除不需要的信息。

2. PySpider 框架

PySpider 框架是一个带有强大的 WebUI、脚本编辑器、任务监控器、项目管理器以及结果处理器的框架。它支持多种数据库后端、多种消息队列和 JavaScript 渲染页面采集功能。

与 Scrapy 相比，PySpider 有以下特点：

(1) 提供了 WebUI，网络爬虫的编写和调试都在 WebUI 中进行。

(2) 内置了 PyQuery 作为选择器。

(3) 支持 PhantomJS 进行 JavaScript 渲染页面的采集。

3. feapder 框架

feapder 是一款上手简单、功能强大的 Python 爬虫框架，使用方式类似于 Scrapy，方便由 Scrapy 框架切换过来。feapder 框架内置 3 种网络爬虫：

(1) AirSpider。是轻量级爬虫，学习成本低。面对一些数据量较少、无须断点续爬、无须分布式采集的需求，可采用此网络爬虫。

(2) Spider。是一款基于 Redis 的分布式网络爬虫，适用于海量数据采集，支持断点续爬、报警、数据自动入库等功能。

(3) BatchSpider。是一款分布式批次网络爬虫，对于需要周期性采集的数据，优先考虑使用此网络爬虫。

feapder 支持断点续爬、数据防丢失、监控报警、浏览器渲染下载、数据自动入库(MySQL 或 MongoDB)，还可通过编写管道对接其他存储库。feapder 框架结构如图 5-4 所示。

feapder 框架的模块如下：

(1) spider：爬虫，是框架调度核心。

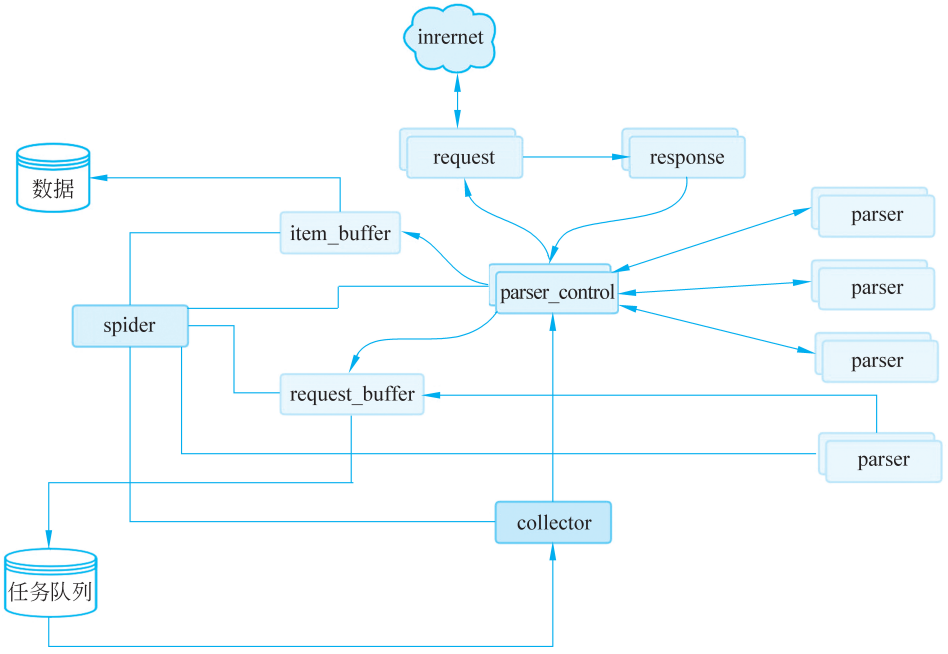

图 5-4　feapder 框架结构

（2）parser_control：解析控制器，负责调度 parser 模块。

（3）collector：任务收集器，负责从任务队中批量取出任务放入内存，以减少网络爬虫对任务队列数据库的访问频率及并发量。

（4）parser：数据解析器。

（5）start_request：初始任务下发函数。

（6）item_buffer：数据缓冲队列，用于将数据批量存储到数据库中。

（7）request_buffer：请求任务缓冲队列，用于将请求任务批量存储到任务队列中。

（8）request：数据下载器，封装了请求，用于从互联网上下载数据。

（9）response：请求响应模块，封装了响应，支持 XPath、CSS、re 等解析方式，自动处理中文乱码。

feapder 框架的流程如下：

（1）spider 调度 start_request 产生初始任务。

（2）start_request 下发任务到 request_buffer 中。

（3）spider 调度 request_buffer 将任务批量存储到任务队列中。

（4）spider 调度 collector 从任务队列中批量获取任务放入内存。

（5）spider 调度 parser_control 从 collector 的内存队列中获取任务。

（6）parser_control 调度 request 请求数据。

（7）request 请求并下载数据。

（8）request 将下载后的数据发送给 response，进一步封装，将封装好的响应返回给 parser_control（图 5-4 中为多个 parser_control，表示多线程）。

（9）parser_control 调度对应的 parser，解析返回的响应（图 5-4 中的多个 parser 表示不同的网站解析器）。