基于 Python 的 Web 信息获取方法研究

魏冬梅，何忠秀，唐建梅

（西华大学 计算机与软件工程学院，四川 成都 610039）

摘 要：随着大数据和云计算等新一代互联网技术的迅速发展，Web 信息量逐日海量递增。从海量数据中提取有效信息，挖掘有潜在价值的关系成为当前的研究热点，这对揭示已知规律、预测未知结果有极大的辅助作用。对当前 Web 信息获取方法、原理和关键技术进行研究分析，重点阐述了数据采集相关技术中网络爬虫算法的分类与应用。提出一种以 Python 和相关库为主要工具，结合模块化方法，构建 Web 文本信息获取系统框架与流程的策略。案例中通过定义采集函数，实现对给定的维基百科词条，快速搜索与该词条相关信息，对词条内链接和外链接进行有效爬取。结果表明，Python 在数据采集方面具有较高的有效性和可扩展性。

关键词：Python；信息获取；网络爬虫；正则表达式

DOI：10.11907/rjdk.172302

中图分类号：TP301 **文献标识码**：A **文章编号**：1672-7800(2018)001-0041-03

Research of Web Text Information Access Method Based on Python

WEI Dong-mei, HE Zhong-xiu, TANG Jian-mei

(*School of Computer Science and Software Engineering, Xihua University, Chengdu* 610039, *China*)

Abstract: As the development of big data and cloud computing, an increasing number of information has been boosted significantly. Extracting the information and extract useful information from huge amounts of data effectively has been becoming the current hot spot. Moreover, it has contributed to revealing the known regulations and predicting unknown results. In this paper, the current Web information retrieval method, the principle and key technology has been analyzed and the algorithm, classification and application of Web crawler in data acquisition technology are emphasized. This paper presents a method of constructing Web text information acquisition system based on Python and related libraries. In the case, by defining the regular expressions and crawling function, it realized searching for the relevant information of the entry, and effectively retrievaling the internal links and the external links. The results show that Python has significant efficiency and expansibility in data retrieval.

Key Words: Python; information access; Web crawler; regular expression

0 引言

互联网提供了大量数据集，但是由于网站本身的多样化和异构性以及网页文档结构的复杂性，很多数据都被嵌入到网页结构与样式中。Web 信息获取，也称为基于 Web 的知识发现。Web 数据获取技术分为：基于本体的数据获取、基于自然语言的数据获取、基于网站查询的数据获取、基于规则和地理位置的数据获取。利用行之有效的方法，将可用的信息从海量数据中抽取出来，挖掘潜在价值，将在金融、电信业、舆情监控、数据分析以及其他科学领域发挥重要作用。由此可见，获取 Web 信息的技术显得尤为关键。

1 Web 信息获取技术

Web 信息获取是指从网站上提取信息的一种计算机软件技术，能将任何可以在浏览器上显示的数据提取出来，因此也称为屏幕抓取或数据采集。Web 信息获取是数据挖掘中的一项重要技术，它涉及到计算机网络、文本处理、决策分析、人工智能等多个领域。其中，基本 Web 文本信息获取和知识发现，包括 Web 文本内容获取、结构

收稿日期：2017－08－08

基金项目：西华大学 2016 年重点实验室开放基金项目(szjj2016-043)；西华大学 2016 年教育教学改革项目(2016)

作者简介：魏冬梅(1981－)，女，硕士，西华大学计算机与软件工程学院讲师，研究方向为智能信息处理、数据挖掘；何忠秀(1974－)，女，硕士，西华大学计算机与软件工程学院副教授，研究方向为计算机应用基础、离散数学；唐建梅(1975－)，女，硕士，西华大学计算机与软件工程学院讲师，研究方向为软件工程、移动互联网开发。

获取、拓扑获取等[2]，需要从网站获取所需的非结构化信息数据，分析处理后存储为统一格式的本地数据文件或直接存入本地数据库，涉及网络爬虫、数据结构化、正则表达式等关键技术[3]。

1.1 网络爬虫工作原理

网络爬虫是一个十分形象的名称，俗称网络蜘蛛或网络机器人，是一种按照规则对 Web 信息进行遍历，自动抓取万维网信息的程序或脚本。网络爬虫通常分为两类：一类是搜索引擎提供商设计的爬虫，这类爬虫会不断地在互联网中利用链接跳转采集页面信息，返回后供搜索引擎建立相应索引。当用户在引擎中输入文字搜索时，引擎即会根据输入对信息进行检索，找到接近搜索文字的相关内容并返回；另一类是对明确指定的网站进行数据抓取，获得所需信息。这些信息通常是可以公共访问的数据集。

网络爬虫的主要算法思想是通过 Internet 从指定的种子集合读取 URL 访问的 Html 页面内容，以及页面包含的超级链接，并通过这些链接继续爬取下级子页面，然后收集用户信息，进行分类和整理[4]。通常有两种情况，一是 Html 使用＜a＞标签表示超级链接，进一步探测 href 属性，实现跳转到链接目标；二是链接跳转是通过触发了相关 JavaScript 代码，如此不断爬取下去，从而搜集到更多数据。网络爬虫结构如图 1 所示。

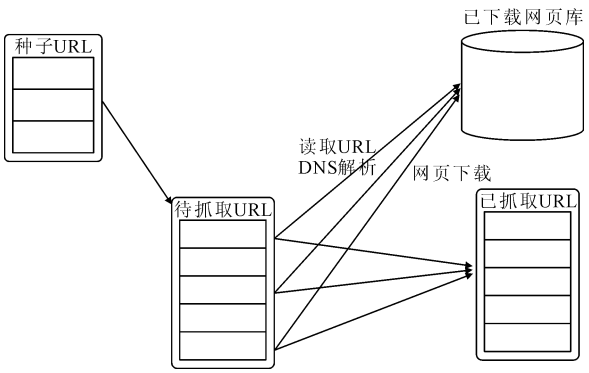

图 1 网络爬虫基本结构

1.2 网络爬虫算法比较

几种爬虫核心算法分别是：广度优先、深度优先、Partial PageRank 及 Opic 策略。

(1)广度优先搜索策略：按照网页内容目录层次，先爬取浅目录同一层次页面，爬取完毕后，将提取出的链接放入队列中，不停向外延伸，尽可能多地获取链接，再深入下一层。这种策略的不足之处是进入深层次目录的时间消耗较长。

(2)深度优先搜索策略：按照深度由低到高的顺序，依次访问下一级网页链接，直到深目录。在完成一个爬行分支后返回到上一链接节点，进一步搜索其它链接。当所有链接遍历完毕后，爬行任务结束。这种策略的缺点是在爬取层次较深站点时资源消耗严重。

(3)Partial PageRank 策略：为非完全 PageRank 策略，在工作过程中爬虫只接触到一部分网页，在搜索过程中，通过计算每个已访问页面的 PageRank 值来确定页面价值，并优先选择 PageRank 值大的页面中的链接进行访问[1]。该策略虽然遍历结果较好，但存在与完全 PageRank 计算的重要性相比差异大的情况。

(4)Opic 策略：算法开始前，每个页面都会被赋予相同数值，当下载了某个页面后，将它拥有的数值平均分配给页面中包含的链接，同时清空自己的数值，根据数值大小，优先下载数值较大的网页。该策略的不足之处是可能导致内容抓取过量。

1.3 数据结构化存储

互联网具有庞大的信息资源，大多数信息都以无结构的文本形式存在，使得信息归类变得非常困难。结构化数据存储是指将大量网页中抽取出来的非结构化信息数据以结构化方式存储到本地(csv，json，xml，Access，Mssql 等)，形成统一格式的本地数据的过程。这也是 Web 数据采集的主要目的，整个过程基本不需要人工干预。数据最终的应用环境决定存储，如果选择数据库，则结构化存储通常指行数据，用二维表结构进行逻辑表达，对 Html 数据进行提取和规范化处理，然后将数组以结构化形式存储于相应的数据库系统中。这种方式具有速度快、准确性高、效率高等优势，再结合多线程技术，速度更是人工所无法比拟的。

1.4 正则表达式

Html 由各种语义对象所构成的逻辑结构体，根据标记描述对象。Web 文本获取主要通过对 Html 页面进行解析，爬虫对 Html 解析主要通过正则表达式进行匹配，然后使用正则表达式匹配的字符串或使用字符串查找与截取的方法提取数据，依次读取每个需要爬取的字段名称和提取规则。例如获取某个站点中提供的国家面积属性[9]，关键语句如下：

```
re.findall('<td class="w2p_fw">(.*?)</td>',html)
```

其中'＜td class＝"w2p_fw"＞(.＊?)＜/td＞'则是进行匹配的正则表达式，它为数据获取提供了捷径。考虑到网站内容可能在未来因升级、更新而产生变化，为了使匹配更加稳定和健壮，还可以使用模块或第三方库，例如 BeautifulSoup、lxml、Htmlparser、Scrapy 等工具进行解析。

2 Web 数据获取系统设计

遵循模块化设计思想，将整个系统进行模块划分，每个功能类作为一个功能模块。这样做的优势是一方面便于代码的维护，另一方面可以增加代码的重用性。借助 Python 强大的类库，实现各模块的功能：利用 urllib2 与服务器进行 HTTP 通信数据交互；利用 re 和 BeautifulSoup 进行文本提取和 Html 解析。

2.1 爬虫功能设计

爬虫通过系统设置的定时任务，被任务激活后，根据定义的 URL 规则对原站点的目录进行正则匹配，符合正则匹配的 URL 链接则进行抓取，然后过滤抓取结果，提取需要的信息。最后，将抓取的信息和数据进行对比，数据库中没有则进行插入更新，有则停止爬虫，其流程如图 2 所示。

2.2 案例实现

根据指定 URL，收集维基百科某词条页面，基于 Py-

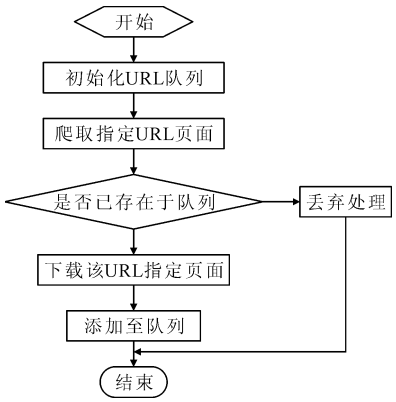

图 2　程序流程

thon，通过建立连接，分析页面特征，编写链接路径构造函数，实现链接获取。

（1）建立连接，发送 HTTP 请求。利用 urllib2 建立与服务器的连接，当服务器接收到请求后，返回相应的 HTTP 应答，本例访问页面 https://en.wikipedia.org/wiki/Harry_Potter，核心脚本如下：

```
from urllib2 import urlopen
url="https://en.wikipedia.org/wiki/Harry_Potter"
try:
req=urlopen(url).read()
except Exception,e:
return
```

（2）正则表达式匹配，进行 URL 筛选。BeautifulSoup 是一个非常流行的 Python 模块，该模块可以解析 Html 网页标签，并提供定位内容的便捷接口。使用 re 模块，构建正则表达式，以匹配满足条件的 URL。例如：在页面中匹配符合要求的超级链接，则需要获取所有 a 标签 href 属性的内容，如下所示：

```
import re
from bs4 import BeautifulSoup
#parse HTML
soup=BeautifulSoup(url)
print bsObj.prettify( )
urls=soup.findAll('a',href=True)
#定义获取链接的函数(articleUrl 代表可作任意输入的词条)
def getLink(articleUrl):
html=urlopen("https://en.wikipedia.org"+articleUrl)
bsObj=BeautifualSoup(html)
return bsObj.findAll("a", href = re.compile("'(/wiki/)((?!:).)*MYM'))
#实现调用
links=getLinks('https://en.wikipedia.org/wiki/Harry__Potter')
while len(links)>0:
links=getLink(newArticle)
for link in bsObj.findAll("a"):
if "href" in link.attrs:
print(links.attrs["href"])
```

（3）获取链接数据。利用 Python 丰富充足的模块库，以及灵活简洁的文件处理和强大的输入输出功能，用广度优先策略对页面链接进行爬取，定义其中爬取链接的功能函数如下：

```
#获取内部链接
def getInternalLink(bsObj,includeUrl):
internalLink=[ ]
for link in bsObj.findAll("a",href=re.compile("'(/|.*"+includeUrl+")")):
if link.attrs['href'] is not None:
if link.attrs['href'] not in internalLink:
internalLinks.append(link.attrs['href'])
return internalLink
#获取外部链接
def getExternalLink(bsObj,excludeUrl):
externalLinks=[]
for link in bsObj.findAll("a",href=re.compile("'(/|.*"+includeUrl+")")):
if link.attrs['href'] is not None:
if link.attrs['href'] not in externalLinks:
externalLinks.append(link.attrs['href'])
return externalLinks
```

3　结语

本文阐述了 Web 信息的获取方法及原理，重点介绍了网络爬虫算法的分类与应用。并基于 Python，结合正则表达式、urllib2 和 BeautifulSoup 等丰富且强大的库，探讨了构建模块化的 Web 数据采集、Html 解析及抓取链接数据的方法。可见 Python 的类库都有各自的优点，比如 re 模块，在简单的数据采集中可以避免过多的开销；而 BeautifulSoup 提供了便捷的接口，可以避免更多依赖。因此，综合利用 Python 丰富的功能库，可以在 Web 数据获取、搜索等方面发挥重要作用。若爬取的 URL 外链层次特别多，Python 执行会耗费很长时间，如果结合 Python 的多线程技术，并考虑下载缓存优化和并发性，将会使性能得到改善。

参考文献：

[1] 于娟，刘强. 主题网络爬虫研究综述[J]. 计算机工程与科学，2015，37(2)：231-236.
[2] 潘庆和，徐耀群，赵星驰. Web 站点拓扑结构获取方法研究[J]. 哈尔滨商业大学学报：自然科学版，2015，31(5)：573-577.
[3] 周中华，张惠然，谢江. 基于 Python 的新浪微博数据爬虫[J]. 计算机应用，2014，34(11)：3131-3134.
[4] 姜杉彪，黄凯林，卢昱江，等. 基于 python 的专业网络爬虫的设计与实现[J]. 企业科技与发展，2016(8)：17-19.
[5] 李俊丽. 基于 Linux 的 Python 多线程爬虫程序设计[J]. 计算机与数字工程，2015(5)：861-863.
[6] 李勇，韩亮. 主题搜索引擎中网络爬虫的搜索策略研究[J]. 计算机与数字工程，2008，228(10)：50-53.
[7] MAGNUS LIE HETLAND. Python 基础教程[M]. 第 2 版. 司维，曾军崴，谭颖华，等，译. 北京：人民邮电出版社，2010.
[8] WESLEY JCHUN. python 核心编程[M]. 第 2 版. 北京：人民邮电出版社，2009.
[9] RICHARD LAWSON. 用 python 写网络爬虫[M]. 北京：人民邮电出版社，2016.

（责任编辑：黄　健）