基于 Scrapy 的 GitHub 数据爬虫

文/赵本本[1] 殷旭东[1] 王伟[2]

摘要

作为最大的社交编程及代码托管网站，GitHub提供了丰富的数据来源。基于Python开源框架Scrapy设计实现了一个Web爬虫，能对GitHub的资源抓取和分析，并进行了结构化处理和存储，可为后续数据分析提供基础。介绍了GitHub爬虫的设计原理、算法的实现方式，对实验结果进行了分析。

【关键词】网络爬虫 数据爬取 Scrapy GitHub Python NoSQL

数据产生于各行各业，在互联网时代大数据概念提出后，人们发现自己手中的数据不再毫无用处，通过强大的技术手段，无形的数据可转化为有形资产。麦肯锡公司的报告指出数据是一种生产资料，大数据是下一个创新、竞争、生产力提高的前沿。世界经济论坛的报告也认定大数据为新财富，价值堪比石油。GitHub是一个巨大的数据宝库，吸引了大量的开发人员与研究人员入驻。2014年的《GitHub中国开发者年度报告》指出，目前GitHub上的注册用户数量已经超过1000万。由于网站托管的开源项目众多，访问的流量也呈爆炸性增长。要想从数千万程序员中快速、准确地抓取所需要的信息变得越来越困难，所以必须使用自动化工具才能较容易的抓取下来。

本文设计并实现了一个基于Scrapy框架的Web数据爬虫，从GitHub抓取大量用户数据信息，下载到本地，进行了结构化处理和存储，为之后做数据分析提供基础。

1 网络爬虫与Scrapy

所谓网络爬虫，就是抓取特定网站网页的HTML数据。网络爬虫从一个存放URL的集合开始进行爬取，首先从队列中获取一个URL并下载此网页，提取该网页中的其它URL并放入队列中。此过程将重复直至关闭。常见的网络爬虫及其爬行策略包括：

（1）广度优先爬虫，一般全网搜索引擎的网络爬虫使用广度优先的爬行策略。

（2）重新爬取已有的页面的爬虫，目的是获得数据的定期更新。

（3）聚焦爬虫，也叫定向爬虫，为很多垂直搜索引擎使用，采取特定的爬行策略来爬取特定类型的网页。本文所研究的正是这一类型的网络爬虫。例如，网页上的某个特定主题或语言、图像、MP3文件等。

Scrapy是用Python开发的一个开源的Web爬虫框架，可用于快速抓取Web站点并从页面中高效提取结构化的数据。Scrapy可广泛应用于数据挖掘、监测和自动化测试等方面，提供了多种类型爬虫的基类，如BaseSpider、SitemapSpider等。

2 基于Scrapy的Github数据爬虫的设计

2.1 Scrapy原理

Scrapy是基于Twisted异步网络库来处理通讯，架构清晰，并且包含了各种中间件接口，

<< 上接 49 页

中光通量远远小于金属发热体的电热管，电-热转换效率高达95℃以上，比同功率的钨钼材料的金属发热体热效率提高30℃以上，升温的时间节省30℃以上。

4 系统软件设计

系统上电后，等待所有硬件（尤其是蓝牙模块的相互配对需要一定时间）都启动成功之后，用户便可通过上位机软件发送指定指令对系统进行控制和监测。

4.1 系统软件设计

温湿度传感器自动采集节点温湿度数据后，由串口发送至单片机，单片机对数据做出分析判断，如果接收到的温度低于30℃，则执行加热操作；否则不予响应。同时，温湿度数据将通过蓝牙技术传输，在手机终端实时显示。下位机软件运行流程图如下：

4.2 蓝牙模块设计

将蓝牙模块切至[prog]参数配置模式，运行程序。菜单中的连接蓝牙项会跳出一个界面，显示本机已配对过的蓝牙设备，并搜寻当前可连接的蓝牙设备。选择要连接的设备后，创建一个服务，实现与设备的连接。当连接成功后再创建数据的接收线程，虚拟创建一个COM口，打开对应COM口，形成透明串口线传输。STC12C5A60S2则是通过串口与蓝牙模块HC-05连接，向手机蓝牙发送数据，每组数据包含两个字节，第一字节为温度值，第二字节为湿度值，这两个字节作为蓝牙数据传输的两个参数。手机接收终端使用3个有效的Button来实现数据的接收，例如发送AT+ Temp，返回Ok:26C；发送AT+ Humi，返回Ok: 34%；发送AT+ Temp Humi，返回Ok: 26C 34%。

5 系统运行测试

将系统硬件组装完成后上电启动，当所有硬件模块正常工作时，部分硬件有工作指示灯闪烁。当硬件控制平台的LED1由交替闪烁的状态变为LED1长亮时，则说明蓝牙配对成功，可以进行数据的正常传输。当手机接收终端显示“Temp:26C”和“Humi:34%”时，则说明温湿度传感器工作正常。系统组装运行效果如图3所示。

由此可见，在温湿度传感器获取节点处实时、精确的温湿度数据后，硬件平台能够进行自我调节，达到合适的温度；小型固态继电器能够间接控制加热片的开关状态，实现加热的功能；蓝牙模块之间能够进行正常的数据通信，实现温湿度数据的传输；手机APP能够实时显示温湿度状况，实现环境监测的功能。

6 结束语

本系统较好地实现了智能加热的控制及显示功能，具有较好的适应性和移植性等，但本身也存在一些如蓝牙系统不够稳定，手机终端与硬件端只实现了数据的单向接收等不足。后期对系统进行升级，可以通过手机APP发送指令，组成局域网，以增加系统的稳定性。因此本套系统虽仍有一定的进步空间，却也有着极大的发展前景。

参考文献

[1] 南通宏晶科技有限公司.STC12C5A60S2系列单片机器件手册[Z].2015.
[2] 广州奥松电子有限公司.数字温湿度传感器AM2320产品手册[Z].2014.
[3] 孙育才.单片微型计算机及其应用[M].南京：东南大学出版社，2004.
[4] 熊狮.基于Android系统健康信息移动监测技术的研究[D].华南理工大学，2013.
[5] 董世琨，张学典，常敏，潘丽娜.基于Android手机蓝牙的无线智能控制系统设计[J].信息技术，2014,08:22-24+33.

作者简介

朱锋（1994-），男，江苏省南通市人。现就读于江苏省南通大学电子信息学院。研究方向为电子科学与技术。

作者单位

南通大学电子信息学院 江苏省南通市 226019

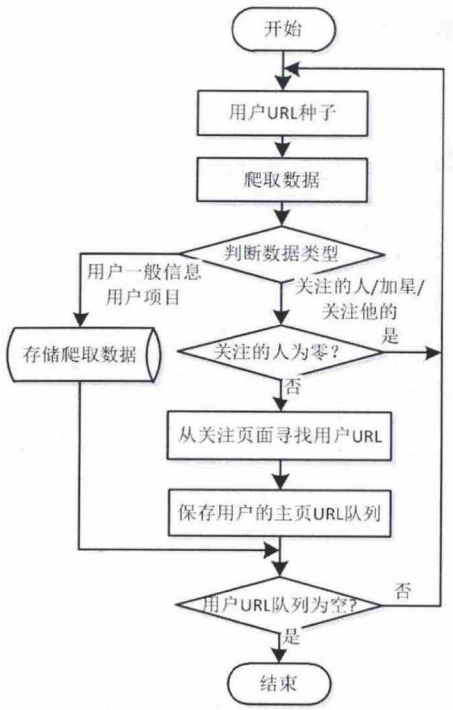

图 1：爬虫算法流程图

可以灵活的完成各种需求。其工作原理为：首先从种子 URL 开始，调度器会传给下载器进行下载，之后会交给爬虫进行分析，根据分析结果进行不同处理。如果是需要进一步爬取的链接，这些链接会传回调度器；如果是需要保存的数据，则被送到项目管道组件进行后期处理，包括详细分析、过滤、存储等。此外，在数据流动的通道里还允许安装各种中间件，进行必要的处理。

2.2 GitHub数据爬虫

2.2.1 GitHub 网页结构与数据分析

GitHub 个人主页的主要信息可分为左右两块：左块为个人基本信息，右块为个人相关的项目。个人基本信息包括名字、邮箱、公司、头像等，爬取时会按照编写的爬虫规则全部保存下来，同时右块的项目信息也储存在 JSON 文件中。格式为：

```
{
    “_id” :<ObjectId>
    “fullname” :<string>
    “mail” :<string>
    “username” :<string>
    “organization” :<string>
    “joined” :<string>
    “starred” :<num>
    …
}。
```

2.2.2 数据定义

Item 是保存爬取到的数据容器，使用方法和 Python 字典类似。根据从 GitHub 网站上获取到的数据 followers、fullname 等对 Item 进行统一建模。从而在 Item 中定义相应的字段 field。对应的 item.py 中的主要代码为：

```
Josh Sherman.txt
{'followers': u'71',
 'following': u'183',
 'fullname': u'Josh Sherman',
 'home_page': 'https://github.com/joshtronic',
 'joined': u'Aug 7, 2009',
 'mail': 'None',
 'organization': u'AppSumo | GravityBlvd',
 'popular_repos': u'pickles',
 'popular_repos_download': u'https://github.
com/joshtronic/pickles/archive/master.zip',
 'popular_repos_info': u'Pickles is a PHP framework for building
kick-ass services.',
 'popular_repos_star': 35,
 'starred': u'1.8k',
 'username': u'joshtronic'}
```

图 2：抓取的个人信息数据

```
import scrapy
class GithubItem(scrapy.Item):
    fullname = scrapy.Field()
    username = scrapy.Field()
    popular_repos = scrapy.Field()
    …
```

2.2.3 编写提取 Item 数据的 Spider

GitHubSpider 是用于 GitHub 网站爬取数据的类，选择从 scrapy.Spider 类继承。定义以下属性和方法：

（1）name 属性：定义 spider 名字的字符串且唯一；

（2）start_urls 属性：Spider 在启动时从 url 列表开始进行爬取。第一个被获取到的页面将是其中之一，后续的 URL 则从初始的页面中提取；

（3）parse：是 Spider 的一个方法。每个初始 URL 完成下载后生成的 Response 对象将会作为唯一的参数传递给该方法。该方法解析提取 Item 数据以及生成需要进一步处理的 URL 的 Request 对象。

Scrapy 提取数据使用 Selector 选择器机制，使用特定的 XPath 表达式来提取 HTML 网页中的数据。此爬虫提取 GitHub 用户主页中的 followers 数据项使用代码：

people['followers']=response.xpath('//div[@class="vcard-stats"]/a[1]/strong[@class="vcard-stat-count"]/text()').extract();

即提取 class 为 vcard-stats 的 div 标签内的第一个 a 标签里的 class 为 vcard-stat-count 的 strong 标签内的文本数据。

2.3 数据存储NoSQL

网络爬虫系统爬取的数据量大，且多为半结构化或非结构化数据，传统的关系型数据库并不擅长这类数据的存储与处理。而 NoSQL 在大数据存取上具备关系型数据库无法比拟的性能优势，因此选择使用 NoSQL 数据库存储爬取到的数据。

2.3.1 数据存储方式

MongoDB 是一个基于分布式文件存储的非关系型数据库，具有灵活的数据存储方式。MongoDB 数据存储不需要固定的表结构，也不存在连接操作。其次它是一种基于文档的、无模式且不保证 ACID 的数据库。每当抓取数据到 Item 项时，都会添加进一个 Mongo 集合。

2.3.2 设置数据库

在 settings.py 文件里设置 MongoDB 的参数：服务器 SERVER、端口 PORT、数据库 DB、数据库表 COLLECTION，之后指定管道后添加数据库设置：

ITEM_PIPELINES={ 'GitHub.pipelines.MongoDBpipeline':300,}

2.3.3 连接数据库

在 pipelines.py 文件中通过管道连接。首先定义一个函数去连接数据库：

```
class MongoDBpipeline(object):
  def __init__(self):
    MongoClient(…)
  def process_item(self, item, spider):
        …
    return item
```

创建了一个类 MongoDBPipeline()，构造初始函数初始化类。process_item() 函数处理被解析的数据。

2.4 反爬虫技术的应对措施

很多网站为避免被爬虫抓取数据，使用了一定的规则和特定的机制来实现，本爬虫主要采取以下措施：

（1）设置 download_delay，即下载器在下载同一个网站下一个页面前需要等待时间。如果下载等待时间长，则不能满足短时间大规模抓取的要求；而太短则大大增加了被 Ban 的几率。因此在 settings.py 中设置：DOWNLOAD_DELAY = 2；

（2）禁止 cookies，可以防止使用 cookies 识别爬虫轨迹的网站察觉，在 settings.py 设置：COOKIES_ENABLES=False；

（3）使用 user agent 池，防止被服务器识别，将 user agent 池写入 rotate_useragent.py 文件里。

2.5 爬虫工作方式及算法

_id	username	popular_repos	home_page	popular_repos_...	joined	popular_repos_...	popular_repos_...	followers
565911154319b...	avelardi	testignore	https://github....	https://github....	Apr 22, 2014	0	None	3
565911154319b...	wjsager	celeryFibonacci	https://github....	https://github....	Dec 11, 2014	0	Learning Celer...	7
565911154319b...	BellaMarquez	openfaqs	https://github....	https://github....	Jan 5, 2015	0	An Open Sour...	2
565911154319b...	joshtronic	pickles	https://github....	https://github....	Aug 7, 2009	35	Pickles is a PH...	71
565911154319b...	rohithadassan...	None	https://github....	None	Jul 30, 2009	None	None	59
565911164319b...	Stantheman	fuse-colors	https://github....	https://github....	Sep 10, 2010	19	Make (almost) ...	37
565911164319b...	OliPicard	ApertusForecast	https://github....	https://github....	Nov 19, 2014	5	Apertus Foreca...	3
565911164319b...	ovwane	None	https://github....	None	May 10, 2012	None	None	28
565911164319b...	Dorthu	SimpleRESTCli...	https://github....	https://github....	Apr 4, 2014	0	A simple, web-...	7
565911164319b...	abemassry	None	https://github....	None	Apr 5, 2012	None	None	24
565911164319b...	jamesottinger	MoneyWatch	https://github....	https://github....	May 25, 2013	3	A personal mo...	27
565911164319b...	calicojack	None	https://github....	None	Dec 12, 2010	None	None	14
565911164319b...	JMortonTan	JSeal	https://github....	https://github....	Feb 17, 2014	3	Steganograph...	15

图 3：MongoDB 数据库中的数据

表 1：GitHub 爬虫爬行测试表

测试时间与网速		5 分钟	10 分钟	1 小时
网速一般（1.04MB/S）	爬取网页数量 (pages)	1502	3004	20387
	爬取用户（有重复）(items)	860	1502	8023
	下载保存用户 (txt)	453 (52%)	703 (46.8%)	3064(38.2%)
网速较好（2.41MB/S）	爬取网页数量 (pages)	1955	3542	23454
	爬取用户（有重复）(items)	832	1660	10043
	下载保存用户 (txt)	430（52%）	780 (47%)	4006 txt(39%)

表 2：网速良好时的测试结果表

测试时间与爬虫类别		5 分钟	10 分钟	1 小时
单爬虫	爬取网页数量 (pages)	2766	4158	26032
	爬取用户（有重复）(items)	1260	2160	11533
	下载保存用户 (txt)	646 (51.2%)	1063(49.2%)	4302(39.3%)

GitHub 网站上网页数以百万记，且分布在全世界各个地方的服务器。对于 GitHub 这样的网站来说，适合采取定向爬虫策略。即从第一个 URL 种子开始，来收集所有的用户信息和可以利用的用户 URL 并保存起来，之后遍历整个 URL 集合。

GitHub 爬虫采取宽度优先遍历策略，其基本思路是将新下载网页中发现的链接直接插入待抓取 URL 队列。也就是指网络爬虫会先抓取起始网页中链接的所有网页，然后再选择其中的一个链接网页，继续抓取在此网页中链接的所有网页。

（1）首先由图开始从用户 URL 种子进行爬取，即用户主页面。然后通过爬虫将网页中个人信息提取出来并分别存储到每个人新创建在本地的文档中。

（2）通过判断用户的关注者即是否等于 0 而继续循环爬行还是重新爬行，不为 0 时会保存每一个被关注者的链接到URL队列里面，方便继续循环。

（3）循环爬取时会保存每一个用户的关注页面，因为如果循环完用户之后，还有个他的关注者 URL 页面没用到，进入关注页面会有重新一批用户，然后再添加入循环。

（4）关注者与被关注者难免会互相关注，所以爬取时会重复爬取，但不会重复下载信息。

爬虫算法用伪代码描述如下：

```
Begin
    init
        name ← github
        headers ← {Accpet，User-Agent，..}
        allowed_domains ← github.com
        start_url ← ["https://github.com/someone",]
    return A
    A       people ← Github_Item()
            people ← response.xpath(“//...”)
            Save as TXT.
            call B(self,response)
    B       Store as followers_urls、people_urls_list
            Duplicates_Removal followers_urls、people_urls_list
            people_link_1 ← people_urls_list[0]
            people_urls_list.remove(people_link_1)
            call C(people_link_1)
    C       Count followers
            IF followers ≠ 0 THEN
                followers_urls.append(url)
                call A(url);
            ELSE
                IF followers_urls ≠ [] THEN
                link_1 ← followers_urls[0]
                followers_urls.remove(followers_link_1)
                call A(followers_link_1);
                ELSE
                people_link_1 ← people_urls_list[0]
                list.remove(people_link_1)
                call A(people_link_1)
                END IF
            END IF
    END
```

3 实验与分析

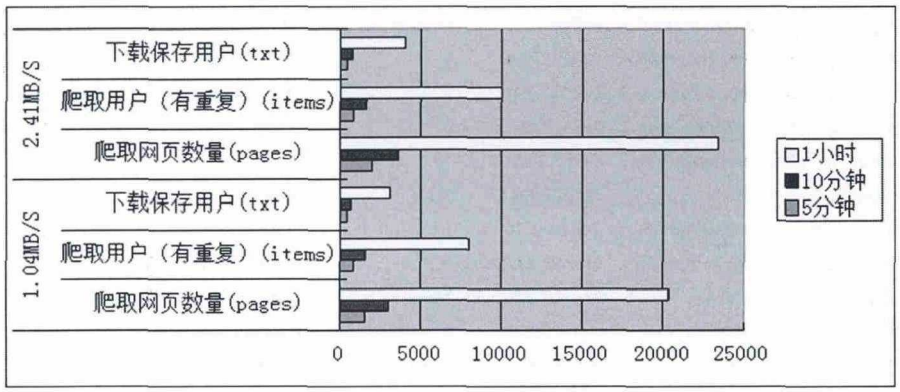

图 4：GitHub 爬虫爬行测试图

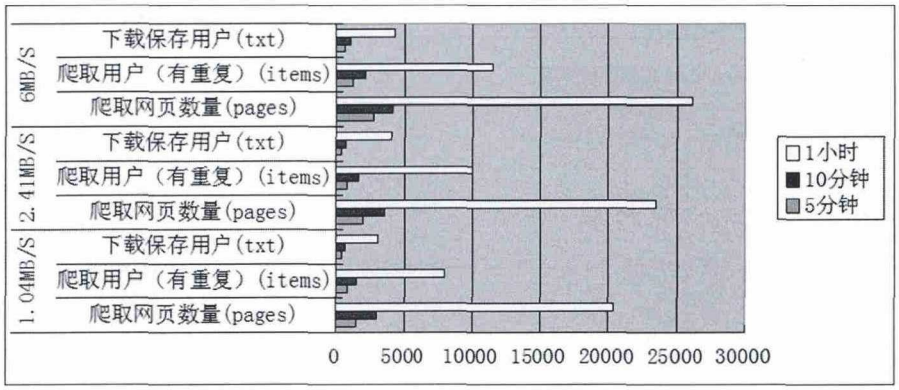

图 5：网速良好时的测试图

3.1 实验结果

对以上的爬虫进行 python 代码的编写，启动运行 scrapy crawl github -s LOG_FILE=scrapy.log，下面是部分数据的展示。图 2 展示了用户个人信息保存在文本文件中的数据，图 3 为 MongoDB 数据库内存储的数据，可用于统计分析查询。

3.2 实验分析

实验环境使用一台 PC 机运行爬虫程序，并且同时运行数据存储服务器。实验测试中，临时暂停发现共爬取了 221 次（有重复），保存了 152 人，URL 队列中还有 623 条个人主页没爬取。没有发现 Crawled (429)（太多请求）等警告信息。

爬取会时不时的停顿，有时出现 twisted.python.failure.Failure 错误。仔细分析记录文件 scrapy.log，发现是在下载图片时变慢的。原因是 Scrapy 使用了 twisted 作为异步网络库来处理网络通讯。当下载缓慢时，twisted 异步请求发生错误，出现一系列的错误回调（errback）或者错误回调链 (errback chain)，错误会传递给下一个 errback，errback 不返回任何数据。而 twisted 的 Deferred 对象会使回调函数把 None 作为参数返回，从而导致出错。

根据爬取的信息得到表 1 所示的测试结果表。

根据表 1 可得到图 4 所示的柱状图，在爬取 1 小时内可以明显的看出网速一般与网速较好时的差异，网速较好有一定的提升了爬虫的速度以及下载的数量。

为了观察网速对实验测试产生的影响，下面针对 GitHub 网站的爬取测试定在早晨 6-7 点钟（6MB/S）左右进行测试，得到表 2 所示的结果。

图 5 是根据表 1 与表 2 制成的柱状图，可以直观的看出网速在很好（6MB/S）的情况下，1 小时内爬虫的速度与下载的信息数据比网速一般、较好时产生的数据多。为保证数据的准确性，上述测试前后进行了 30 余次，取平均值。

综上，可以得出网速与爬虫爬取的信息数量的快慢有关，但是下载的用户信息数量与爬取用户（有重复）的比率是差不多的，从开始的 51% 左右到 1 小时内下降至 39% 下载率。这也跟每个人的关注度有关，若两人互相关注，重复的抓取下来分析发现有的已经下载好了，当然这是不可避免的。

4 结语

基于 Python 的 Scrapy 框架设计并实现了 GitHub 爬虫，从技术上为一些数据研究们提供了方便的网络上数据获取方法。主要特点：

（1）使用方便，只需提供一个用户的主页面 URL 就能利用本文爬虫抓取 GitHub 网站中大规模用户信息；

（2）支持数据库，所有抓取的信息都保存到 MongoDB 数据库中，利于统计查询；

（3）支持大范围地爬取用户信息，从一个用户的 URL 可得到多个关联用户的 URL 从而衍生出大量用户；

（4）图形界面操纵方便。当所有的数据都爬取下来时，可以下载任何用户的热门项目。

本 GitHub 爬虫在处理重复爬取用户信息方面，采取了按名字判断的方法，使重复爬取的次数得以明显减少，但是抓取下来的用户信息也随之减少了，该问题尚须进一步探索和改进。

（通讯作者：殷旭东）

参考文献

[1] 邬贺铨．大数据时代的机遇与挑战 [J]．中国经贸，2013(7)：32-32.

[2] GitHub 中国开发者年度报告 {2014} [EB/OL]．[2015-02-03] http://githuber.info/Report.

[3] Manning 爬虫技术浅析 [EB/OL]．http://drops.wooyun.org/tips/3915.

[4] 赵鹏程．分布式书籍网络爬虫系统的设计与实现 [D]．西南交通大学，2014.

[5] 赵志．基于 NoSQL 的 BRS 系统的设计与实现 [D]．上海交通大学，2013.

[6] Scrapy 研究探索（七）——如何防止被 ban 之策略大集合 [EB/OL]．[2014-06-29] http://blog.csdn.net/u012150179/article/details/35774323.

[7] 黄聪，李格人，罗楚．大数据时代下爬虫技术的兴起 [J]．计算机光盘软件与应用，2013(17)：79-80.

[8] Twisted 15.4.0 documentation [EB/OL]．http://twistedmatrix.com/documents/current/core/howto/defer.html.

作者简介

赵本本（1995-），男，常熟理工学院计算机科学与工程学院本科在读学生。

殷旭东（1970-），男，硕士学位。现为常熟理工学院计算机科学与工程学院工程师。CCF 会员。主要研究方向为移动计算、信息安全。

王伟（1987-），男，硕士学位。现为苏州市浪潮电子信息有限公司工程师。主要研究方向为信息安全、大数据。

作者单位

1. 常熟理工学院计算机科学与工程学院　江苏省常熟市　215500

2. 苏州市浪潮电子信息有限公司　江苏省苏州市　215002

● 基金项目：2015 年江苏省大学生实践创新训练计划校企合作基金项目（201510333068H）。