基于 RoadRunner 算法的 RESTful Web 服务信息收集研究

季红梅，张轶昀

（安徽财贸职业学院 电子信息系，安徽 合肥 230601）

摘 要：在服务组合不断发展的大背景下，新生代 RESTful Web 服务逐渐展示出其优越的性能从而迅速占领大量服务组合份额。与此同时，由于其自身设计特点，RESTful Web 服务没有类似 UDDI 的注册检索中心，也没有合适的描述语言。各大网站通常以开放平台的方式各自为营，使 API 信息难于机器检索，为更大范围的服务组合带来了困难。为了解决这个问题，这里探索了一种使用聚焦爬虫收集 RESTful Web 服务 API 信息以提供统一的集中检索服务的思路，使用 RoadRunner 算法设计并实现了一个聚焦爬虫，从概念上验证了这个思路的可行性。

关键词：服务组合；RESTful Web 服务；RoadRunner；爬虫

中图分类号：TP311 **文献标识码**：A **文章编号**：1671－380X（2013）03－0037－05

Research on Collecting Information of RESTful Web Services Based on RoadRunner Algorithm

JI Hong－mei, ZHANG Yi－yun

(*Department of Electronic Information, Anhui Finance & Trade Vocational College, Hefei* 230601, *China*)

Abstract: In the context of continuous development of mashups, the new generation of web services——RESTful Web services——gradually demonstrates its superior performance and occupies a large share of mashups soon. At the same time, due to its own design features, RESTful Web services has neither a registered retrieval center similar to UDDI, nor an appropriate description language. Major sites usually craft to stand alone by the way of opening platform, which makes it hard to search API information by machines and bring difficulties to a wider range mashups. To solve this problem, this paper explores an idea of using focused crawler to collect the API information of RESTful Web services to provide a unified centralized retrieval service. The paper designs and implements a focused crawler using RoadRunner algorithm, verifies the feasibility of the idea from concept.

Key words: Mashups; RESTful Web Services; RoadRunner; Crawler

1 引言

W3C 将 Web 服务[1]定义为：一类用 URI 标识的软件系统，使用 XML 定义和描述其对外公开的接口和绑定，有其特定的输入输出设定，可对输入完成特定的处理，通常由许多应用程序接口（API）组成，用户通过访问 API 获取所需要的服务。

遍布于网络的 Web 服务提供着海量功能。Web 服务对于网络编程而言相当于面向对象编程中的 jar 包，对其加之有效的利用可以大大的减小程序开发的工程量，提高效率、稳定性和安全性。如何根据需求，智能化、自动化进行服务组合（mashups），这种整合网络上多个资源或功能，以创造新服务的网络应用程式成为 Web 服务研究领域的热点。

目前，网络上普遍存在的 Web 服务分为两大类——以 Soap（Simple Object Access Protocol）为代表的传统 Web 服务和新兴的 RESTful（Representational State Transfer）Web 服务。

Soap 类服务已经发展成熟，一般使用 WSDL（Web Service Describe Language Web）描述，在 UDDI（Universal Description，Discovery and Integration）注册。用户可以通过访问 UDDI 查询所需的服务信息并进行所需的组合。Soap、WSDL 和 UDDI 曾经被视为 Web 服务的三大基石，然而信封式结构设计和 post 的调用方式使原本简单的协议在构建实际应用时人为增加了抽象层次，从而使服务变得更加复杂难于调试。

相较于 Soap 服务，RESTful Web 服务回归了 Web 本质，是对资源的抽象，充分利用 HTTP 协议原本提供的各种方法，更加简洁轻量，拥有更出色的适应性、可伸缩性、及可维护性[2]。随着互联网拆墙运动的不断深入，REST 类服务迅速发展壮大，根据 ProgrammableWeb 的统计，目前活跃的服务 API 中，RESTful Web 服务占到约 70% 的比例。

设计上的重大变革令 RESTful Web 服务不能使用传统的描述语言 WSDL，也没有类似 UDDI 的统一注册集合。如今，并没有一种公认的描述语言可以用来描述 RESTful Web

收稿日期：2012－10－24

基金项目：安徽省高等学校省级自然科学研究基金项目（KJ2010B010）。

作者简介：季红梅（1963－），女，江苏常熟人，高级工程师，学士，研究方向：计算机及其应用，Email：hfjhm@163.com。

服务，同时也没有公认的 RESTful 服务的注册集合，这为此类服务的智能组合带来巨大的底层困扰。

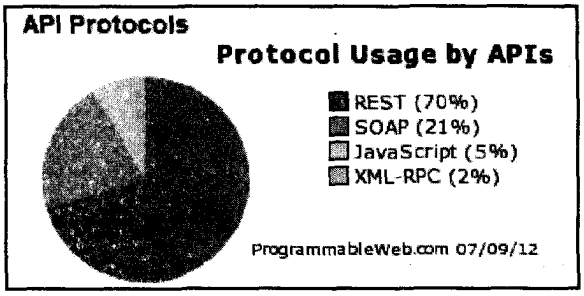

图1 ProgrammableWeb API 种类统计图

ProgrammableWeb 是目前最大的 RESTful Web 服务注册中心，收录了超过4000条 RESTful Web 服务 API 目录页信息。这里设计了一个聚焦爬虫，通过 ProgrammableWeb 提供的目录页信息进入具体描述 API 参数的详情页收集 REST 类服务的接口信息并整理入库，形成智能化服务搜索引擎的基本资源库，为实现 RESTful Web 服务的智能化自动化 mashup 提供基础。在此设计中，使用 RoadRunner 算法[3]自动生成爬虫提供自动生成提取板结构化网页信息的正则表达式，提取可供上层系统调用的结构化 API 信息。

2 相关概念和定义

2.1 DOM Tree（Document Object Modle Tree）

DOM Tree 即文档标签模型树，是通过 DOM（文档对象化模型）解析网页的 HTML 文档，从而生成相应的树状结构。通过比较网页的树结构可以发现网页的共同结构特征，形成抽取规则，获取数据信息。

2.2 包装器（wrapper）

网页上的信息通畅是半结构化的，而可被上层系统利用的信息是结构化的。包装器是一个从自然语言或网页非结构数据中抽取结构化数据的程序[4]。为了减少人工劳动，在此设计中使用 RoadRunner 算法生成包装器，这是一种无监督自动学习的抽取算法。

一个网站中，虽然各个网页的内容有所不同，但是来自同一网站的网页总是拥有统一的风格，比如使用同样的列表展示数据，在同样的位置插入图片或广告。自动抽取法可以根据给定的网页集，通过深度遍历并比较网页 DOM Tree 节点的方法学习归纳抽取规则，从而完成对大量站点的自动抽取。

2.3 正则表达式

网页的数据通常是非结构化或半结构化的，通常很难被上层系统直接使用。这里要使用到一些 Web 数据挖掘的相关技术，来提取网页数据，将网页数据转化为结构化数据。这通常比制作一个聚焦爬虫本身具有更大的工作量[5]。正则表达式就是主要的网页数据提取技术之一。

正则表达式（Regular Expression）是一个用来描述或者匹配一系列符合某个句法规则的字符串的单个字符串[6]。在很多文本编辑器或其他工具里，正则表达式通常被用来检索和/或替换那些符合某个模式的文本内容。在 Perl、PHP、Python、JavaScript 等脚本语言中运用十分广泛[7]。

包装器形成的抽取规则的中心内容就是正则表达式。正则表达式匹配每个网页的 HTML 文档从而直接高效的获取所需的 API 描述信息。

2.4 名词定义

（1）API 目录页

ProgrammableWeb 收录的4000多条 API 信息均为各个网站自行注册的 API 目录页。举例而言，人人网开放平台开发文档首页的 URL（http：//wiki. dev. renren. com/wiki/API）就是一个在 ProgrammableWeb 注册的 API 目录页，标签项是 API Home。目录页上通常包含该开放平台所提供的所有 API 的目录，并且每条数据项都是一个新的超链接，指向该 API 的详情描述页。

（2）API 详情页

API 详情页包含了对一个 API 的详尽描述，通常包括功能描述、输入输出参数与样例等，也是爬虫的最终爬取目标。爬虫通过种子节点 ProgrammableWeb 的总目录页依次遍历在其注册的各 API 目录页，再以这些目录页为新的种子节点，进入并识别 API 的详情页，利用 RoadRunner 算法自动生成的正则表达式提取详情页描述信息，从而完成聚焦爬取任务。

2.5 RoadRunner 算法

RoadRunner 算法是一个专门为解决从一组详情页中抽取结构化数据而设计的算法，既不需要“先验知识”（a priori knowledge），也不需要人工参与。

算法输入为整理好的 DOM Tree，从两棵树根节点开始深度遍历。相同的节点定义为匹配节点，否则为非匹配点。一旦遇到非匹配节点，算法转入非匹配节点处理，完成后访问该节点兄弟节点，直至完成整棵树的遍历，算法结束，生成一个无并正则表达式的包装器。

运算过程如下：

```
void CreateWapper () {
//样本集中存有 n 张网页的 DOM Tree，每张网页的 DOM Tree 根节点记为 Pi
//正则表达式包装器记为 W
W = P1;
for (i=2; i< =n; i++) {
Compare (Pi, W);
//深度遍历 Pi 和 W，比较节点，泛化 W 抽取规则
}
Change (W); //将 W 指向的泛化 DOM Tree 转化为正则表达式
}
```

DOM Tree 的匹配节点是出现在每一张样本集中的节点，这些节点往往规定着网页的组成结构，比如目录栏，广告栏等等。这些节点内容属不关心信息，真正需要收集的信息是网页中的描述数据，该类数据在各个网页上重复度很小，用 DOM Tree 表述，就是算法中的非匹配节点。

RoadRunner 算法对非匹配点的定义如下：

（1）文本字符串失配：它们表示数据域或数据项。

（2）标签失配：它们表示可选项。

以下用一个人人开放平台 API 详情页[8]的例子来说明。

参数表

Name	Required	Type	Description
sig	true	string	签名认证。是用当次请求的所有参数计算出来的值。点击此处查看详细算法
method	true	string	users.getLoggedInUser
v	true	string	API的版本号，固定值为1.0
api_key	false	string	申请应用时分配的api_key，调用接口时候代表应用的唯一身份
session_key	false	string	当前用户的session_key
access_token	false	string	OAuth2.0验证授权后获得的token。当传入此参数时，api_key和session_key可以不用传入。
format	false	string	返回值的格式。请指定为JSON或者XML，推荐使用JSON，缺省值为XML

返回XML样例

图 2　users. getLoggedInUser 参数截图

图 2 和图 3 分别是人人开放平台 Users. getLoggedInUser 接口和 Users. getInfo 接口的参数表截图，也是爬虫需要收集的数据信息。可以清晰的看出，他们的网页结构是一致的表格，表格同行内容不同，即可判定为失配 1，即文本字符串失配。某些参数后有“点击此处查看详细算法”的快捷方式，此处出现失配 2，即标签失配，为可选项。由于参数个数不同，表格行数不同，即可选行。多出的行数也可判定为失配 2。此处算法将执行回溯算法，同已有包装器比对识别可选的表格行，同时泛化包装器。

从源代码的角度看，如图 4 展示的代码片段，算法首先读入 Users. getLoggedInUse 的 HTML 代码，转换 DOM 树作为原始包装器，再读入 Users. getInfo 的 HTML 代码，与包装器做逐一标签比对。第 1、3、4 行相同，第 2、5 行出现了失配 1，即文本字符串失配，此处的失配内容为需要提取的数据信息，包装器作出泛化 1，标明数据项。第 6 行出现失配 2，标签失配，算法作出回溯，寻找包装器上与 Users. getInfo 第 6 行符合的标签，未发现匹配，此处作出泛化 2，标记该标签为可选标，并跳过该行，比对包装器的第 6 行与 Users. getInfo 的第 7 行。Users. getInfo 的第 8 行发生失配 2，逐行回溯发现与包装器第 1 行匹配，由泛化 1 直接提取 9、12 行数据内容并在此处作出泛化 3，标明可选行。回溯后继续比对，至包装器 7、8 行与 Users. getInfo14、15 行匹配，比对完成。

参数表

Name	Required	Type	Description
method	true	string	users.getInfo
sig	true	string	签名认证。是用当次请求的所有参数计算出来的值。点击此处查看详细算法
v	true	string	API的版本号，固定值为1.0
api_key	false	string	申请应用时分配的api_key，调用接口时候代表应用的唯一身份
session_key	false	string	当前用户的session_key
access_token	false	string	OAuth2.0验证授权后获得的token。当传入此参数时，api_key和session_key可以不用传入。
format	false	string	返回值的格式。请指定为JSON或者XML，推荐使用JSON，缺省值为XML
uids	false	string	需要查询的用户的ID，多个ID用逗号隔开。例如：uids=1232,342,12324。建议都传入此参数。当此参数为空时，缺省值为session_key或access_token对应用户的ID。
fields	false	string	返回的字段列表，可以指定返回那些字段，用逗号分隔。如：uid,name,sex,star,zidou,vip,birthday,tinyurl,headurl,mainurl,hometown_location work_history,university_history。如果不传递此参数默认返回 uid,name,tinyurl,headhurl,zidou,star。此参数请指定为你需要的字段，为了提升响应速度，你不需要用到的字段请不要传入。注意：当不传入session_key或没有权限时，fields失效！

返回XML样例

图 3　Users. getInfo 参数截图

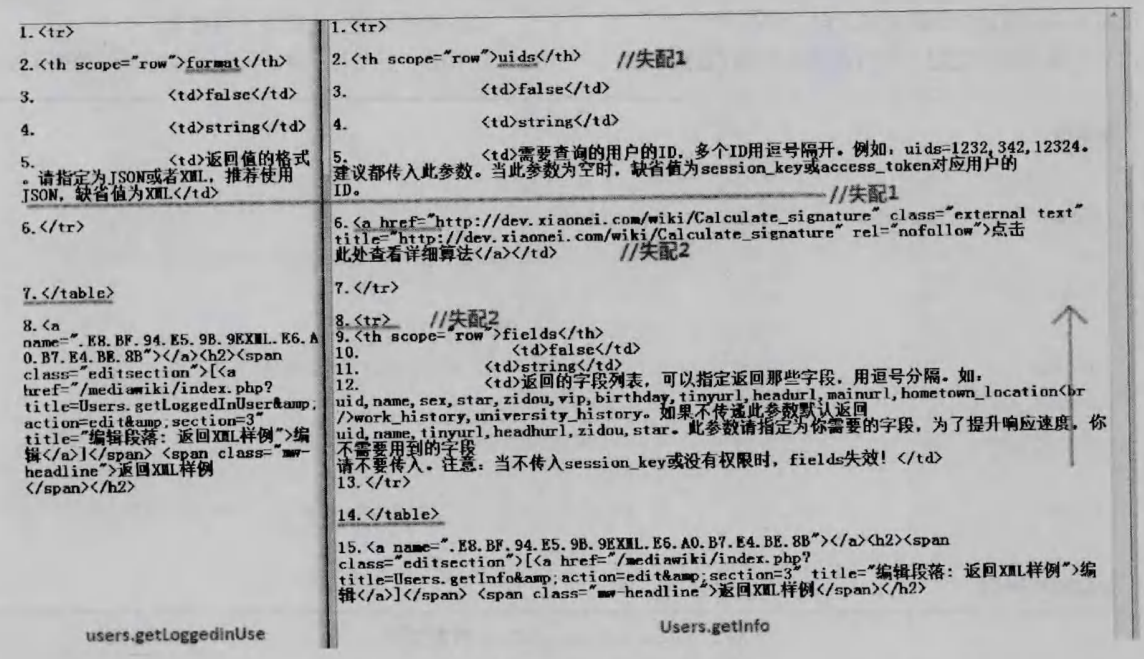

图4　HTML源文件对比运算样例

完成比对后，最终生成的包装器表示为正则表达式：

<table>（<tr><th scope="row">#text</th><td>false</td><td>string</td><td>#text（<a href="…）？</td></tr>）</table><a…返回XML样例</h2>

为了降低运算复杂度，对于单纯可选项，如上例Users.getInfo源码中的第6行，仅做识别，不做信息提取。

3　爬虫的设计与实现

爬虫主要由ProgrammableWeb爬取模块、网页HTML净化模块、包装器生成模块、后续处理模块以及相应的数据库组成，在此设计中数据采用SQL2008数据库存储。图5展示了系统数据流及处理过程逻辑。

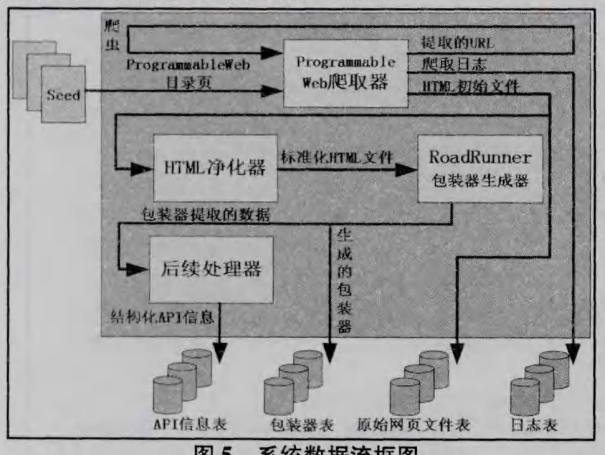

图5　系统数据流框图

3.1　Programmable Web爬取模块

在此设计中使用Programmable Web的RESTful Web服务列表页为种子URL，由于只涉及特定网页，采用自动生成包装器的方式十分浪费资源。这里的Programmable Web爬取模块采用人工订制的方法设计正则表达式直接获取注册在Programmable Web的RESTful Web服务目录页的URL，同时完成存入数据库和获取详情页URL的两个操作。模块将获取的详情页URL存入爬虫的爬取队列进行下一步爬取与分析。

爬取的网页在送交下一模块处理的同时，以字符串的形式存入原始网页表，作为原始数据，以备效率检验时翻阅。爬行过程自动生成爬行日志表，包括爬取网页的具体时间，网页URL等信息，存入日志表，方便后期查阅。

3.2　网页HTML净化模块

由于HTML并不是要求严格的标记语言，许多标记不严格成对的源文件同样可以被浏览器识别，这就为程序正确构建DOM树埋下隐患。网页HTML净化模块用于对收集到的HTML文档进行预处理，使标签严格成对，以简化后端系统的运算量。

3.3　包装器生成模块

该模块是爬虫的核心模块，使用RoadRunner算法生成包装器。由于爬虫需要收集多个网站的API信息，每个网站都有自己的风格，有些网站的风格因为都建立在wiki开源架构上，十分雷同，这就要求该模块能为不同的网站订制生成包装器。此处在算法设计上，每一个从ProgrammableWeb上爬取的API目录页都来自不同的网站，每次开始爬取详情页之前新建一个包装器。

RoadRunner算法需要一个自动学习过程来泛化包装器，学习集的大小很大程度上影响到系统的效率和准确率。算法通常将读入的第一个网页作为初始包装器，以此为基础，读入后续网页进行泛化。通过实验，目前设定的学习集阈值为5，即连续读入5个HTML文档都没有对包装器进行泛化，则包装器确定生成，使用该包装器直接对来自该网站的网页进行信息提取。生成的包装器存入包装器表。由于

网站的 API 信息会不时更新，再次爬取不可避免。再次爬取时可以直接使用包装器表中对应的包装器为初始包装器，即泛化基础，只要网站没做出大幅度风格修改，新的包装器会快读生成，从而减小运算复杂度。新生成的包装器直接覆盖包装器表中的老包装器。

若网站作出大量改动，造成使用旧有包装器为初始包装器时有大量的不匹配，当连续 2 个 HTML 文档失配率超过阈值时，系统会抛弃旧有包装器，使用读入的第一个网页为初始包装器重新生成包装器同时抛出系统警告，并引入人工查看。

包装器确认生成后，继续比对的网页，若差异很大，则无法提取正确数据。对于该类网页通常选择抛弃处理。若来自同一网站的网页超过 50% 的网页被执行抛弃处理，系统会生成自动警告，并引入人工查看。

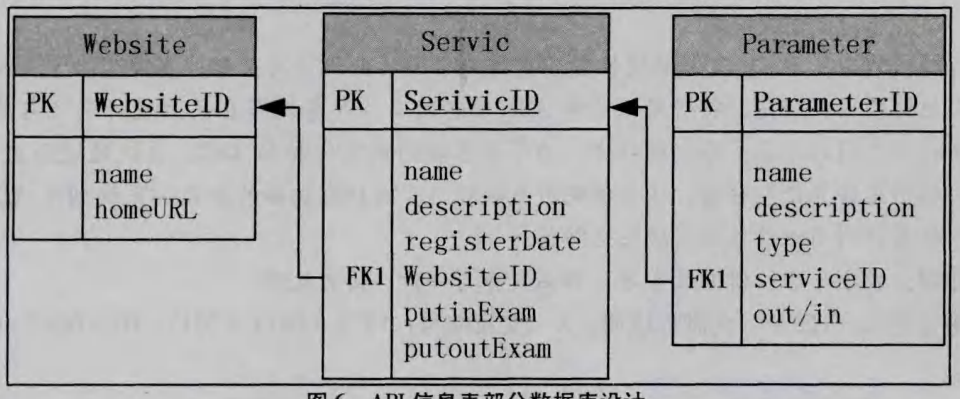

图 6　API 信息表部分数据库设计

3.4　后续处理模块

来自包装器提取的数据是无结构化的数据，无法存数关系数据库，也无法被快速查阅使用。后续处理模块用于规整包装器提取到的数据，并转化成结结构化数据存入 API 信息表。该模块主要通过控制输入输出整理包装器提取的数据，以 HTML 文档为单位，包装器提取一个网页的数据，后续模块整理一个网页的数据。主要存储项为 API 详情页 URL，API 来源网站名，API 名称，API 作用，API 参数，API 访问样例，其他。API 信息表数据库存储关系见图 6。

4　总结与展望

通过实验，爬虫已经可以提供基本的设计要求，收集了一定数量的 RESTful Web 服务的 API 信息，完成了 RESTful Web 服务智能组合研究领域的一次新尝试，为服务组合前端资源集的构建提供了一种新思路。

该设计的研究内容依然有许多需要继续深入的地方，主要是：

（1）收集到的 API 资源需要一个面向全网的查询系统，使这些资源可以被检索到从而真正服务于 RESTful Web 服务组合。

（2）RoadRunner 算法复杂度随输入字符串的长度指数级增加，大大降低设备的处理性能，在面对海量资源时显得力不从心。有很多方法可以改进 RoadRunner 算法，降低复杂度，比如引入一组启发式规则限制搜索空间和回溯，这也是此文继续研究的方向。

（3）自动生成包装器虽然可以大大降低人工劳动量，但是不可避免的会产生差错。单靠管理员更正工作量过于庞大。可以使用 wiki 模式的检索系统，用户在查询 API 信息的同时可以更正信息内容，在培养一定数量的用户群后可以有效保证数据的准确性。

（4）API 数据不是一成不变的，网站根据自己的需求会不定期删减或更改一定的 API 信息。这就要求设计一套自动更新 API 信息的监测程序，按照一定的周期跟踪 API 信息，完成再爬取等操作。

（5）ProgrammableWeb 虽然是最大的 RESTful Web 服务注册中心，但是毕竟不能保证所有的 RESTful Web 服务都有注册。系统需要对智能加入新种子节点的可能性进行探索。

参考文献：

[1] 尹自航．服务搜索引擎中服务测试与比较功能的设计与实现[D]．北京：北京航空航天大学，2009

[2] Roy Thomas Fielding. Architectural Styles and the Design of Network - based Software Architectures[D]. Phd thesis, University of California, Irvine, 2000

[3] V Crescenzi, G Mecca, and P Merialdo. RoadRunner: Towards automatic data extraction from large Web sites [C]. International Conf. on Very Large Data Bases (VLDB 2001), Roma, Italy, September 11 - 14, 2001, 109 - 118

[4] Liu Bing. Web 数据挖掘[M]．北京：清华大学出版社，2009，231 - 275

[5] K Pol, N Patil, S Patankar, C Das. A Survey on Web Content Mining and Extraction of Structured and Semistructured Data [C]. First International Conference on Emerging Trends in Engineering and Technology, ICETET, Nagpur, India. 2008: 543 - 546

[6] 罗刚．自己动手写搜索引擎[M]．北京：电子工业出版社，2009：159 - 194

[7] 邱哲，符滔滔，王学松．开发自己的搜索引擎 Lucene + Heritrix[M]．北京：人民邮电出版社，2010，423 - 445

[8] 人人开放平台[EB/OL]. http://dev. renren. com, 2012 - 9 - 25