90-95 PDF

引用本文

张倩, 林安成, 廖秀秀. 自动结构化数据的电商网站主题爬虫研究. 计算机系统应用, 2018, 27(7): 90-95.http://www.c-s-a.org.cn/1003-3254/6412.html

Zhang Q, Lin AC, Liao XX. Research on Theme Crawler of E-Commerce Website Based on Automatic Data Structuring. Computer Systems and Applications, 2018, 27(7): 90-95(in Chinese).http://www.c-s-a.org.cn/1003-3254/6412.html

# 自动结构化数据的电商网站主题爬虫研究

张倩, 林安成, 廖秀秀

摘要: 当前对于拥有海量数据的互联网, 经常需要采集多个源站的结构化数据以用于数据分析、挖掘, 而为不同网站定制数据采集程序的人工成本很高, 本文提出了一种自动结构化网站数据的主题爬虫方案. 以电商类网站为例, 基于其具有统一层次结构、垂直领域拥有行业语料和规范的特点, 从理论角度确定了结构化提取方案的可行性. 提出相似重复检测和基于属性语义的标签匹配等算法, 实现结构的分析和目标字段的匹配, 并出于系统管理和调优的考虑, 设计了预设匹配模板和结构分析结果复用机制. 实际应用和错误率测试表明, 本方案具有很强的可行性, 能够大大减少人工编写的代码, 错误率较低. 设计思路可应用于其他领域的主题爬虫系统, 快速获得多个站点的大量数据, 将焦点更多地放在结构化数据的处理和信息挖掘.

关键词: 自动结构化 爬虫 标签匹配 多源站 电商网站

## Research on Theme Crawler of E-Commerce Website Based on Automatic Data Structuring

ZHANG Qian, LIN An-Cheng, LIAO Xiu-Xiu

**Abstract**: The Internet has a huge amount of data, someone often need to acquire structural data of multiple source station to support data analysis, disinterment. The artificial cost of different customized website data acquisition program is very high. This paper presented a scheme of automatic data structuring in web crawler. Taking an e-commerce website as an example, this paper confirmed the feasibility of structured extraction scheme from the theoretical point of view based on its unified hierarchical structure, vertical domain, and data corpus. This study proposed the similar duplicate detection and attribute based semantic label matching algorithm, implemented analyzing the structure and matching the target fields, and designed a preset matching template and the reuse mechanism of structural analysis results, for management and tuning the system. Practical application and error rate test show that this scheme is very feasible and can greatly reduce artificial coding, and the error rate is low. The design idea can be applied to the subject crawler system in other fields, and quickly obtain large amount of data from many sites, and let people focus more on structured data processing and information disinterment.

**Key words**: automatic data structuring crawler label matching multiple source station e-commerce website

爬虫技术的目标是对海量的分散的互联网数据进行采集, 是搜索引擎系统的基础. 大数据近年来发展快速, 不仅是数据容量的巨大, 更是强调对样本数据的分析. 互联网中的数据包含大量有价值但异构的信息, 需要被提取出来并以一种结构化的形式来表示, 作为大数据的数据来源[1,2].主题爬虫垂直于某一领域, 相比于通用爬虫, 增加了对有价值数据的组织和归类. 从一些主题爬虫的实践发现, 他们通常手工编写结构化代码来分析某一个目标站点[3], 很多情况下需要获得多个源站的结构化数据, 就不得不分别编写相关代码, 所付出的时间和人力代价是不小的. 而且, 爬虫系统的编写是在前人的经验基础之上, 大多是重复性的劳动, 特别是垂直于同类网站的主题爬虫, 分别编写规则更显得繁琐和没有建设性, 那么寻找一个相对自动化的方案就显得十分有意义. 前人对自动化爬虫的研究还不能够很好地适应上文提到的场景, 比如基于主题语义的信息采集, 无法做到具体字段的结构化[4]; 而使用概率模型来采集最优数据[5], 其可靠性和最终的数据量不符合实际工程应用上的要求. 为了得到一个可靠而高效的自动结构化爬虫, 本文选择从电商网站切入研究, 其具有一些特点: 层次分布稳定, 有价值数据比较密集, 标签、数据变化快, 采用了一些主流反爬机制. 基于这些特点, 电商网站成为合适的样本. 本文将从研究源站的统一性开始, 设计和采用一些抓取的方法和技巧, 实现自动结构化的目的. 同时, 为了追求效率和工程化, 补充了在实际使用的解决方案.

## 1 电商网站结构分析

电商网站的层次分布是稳定的, 调研了国内几家大型电商网站(淘宝、苏宁、京东)和一些小型商户使用的开源商城程序(ECSHOP、ShopN、HiShop), 为了让用户更容易适应页面操作和商品入口, 这些网站一般被组织为如图1所示的结构.

图 1 电商网站组织结构


对用户来说，可通过商品分类和搜索匹配两种方式进入列表页面，列表页面即是众多商品的入口，也是信息的主要通道. 爬虫系统采用的路线是“分类-列表-详情”，原因有三点: 1) 商品分类本身也是有价值数据; 2) 分类和商品的关联性更是隐式的重要信息; 3) 从这条路线走下来，理论上可以抓取整站而不遗漏，具有确定性.

确定网站结构的层级是很有意义的，它可以确定当前抓取和下一层抓取的是哪种页面类型，这样就可以给不同类型做抓取策略上的匹配，而不用通过语义层面来标识[6]，降低了系统复杂度，提升了效率和精准度.

## 2 抓取方式的适配

页面有静态和动态之分，这影响了爬虫系统的抓取行为:

(1) 大多数情况下，每个url对应的是静态页面，浏览器直接解析请求url后响应的html.

(2) 动态页面的情况是比较特殊的，网页的数据采用异步加载，即站点服务器初次响应的数据仅仅是页面结构框架和异步执行的代码，加载完毕后，再次请求服务器拿到数据，通过JavaScript操作Dom组合成完整页面.

特别在近年来，前端应用引擎诸如React、AngularJS和国产的vue日益盛行，模板化和异步加载的方式更是应用广泛，导致上面所述第二种情况出现的几率不低. 所幸爬虫领域也有足够多的方案去适配它，有人通过模拟它的JS行为获得数据[7]，更通用的方法是应用各平台的前端渲染支持库(HtmlUtil、PhantomJS)，它们带有JS引擎，就好像真正在浏览器加载页面等待渲染完毕一样，有人应用这些支持库实现了比较优秀的系统[8]，甚至形成了称为AjaxCrawler的体系[9].

电商网站也是一样，两种页面经常是共存的，为了提高普适度，本文对动态页面采取了应用前端渲染支持库的做法. 那么如何标识出当前页面是属于哪一种类型呢？本文提出一种“预抓取-比较判断-标识”的方法，其页面类型判断流程如图2所示.

图2中的“比较”过程参考了一种基于网页正文结构和特征串的相似网页去重算法[10]，如图3所示.

首先进行网页正文的抽取，过滤掉网页中的噪声(例如导航条、广告条、超链接和网站底部说明)，利用网页正文生成树算法得到一棵结构树，然后用Bloom Filter算法计算每一层次标签特征串的指纹，用页面所有元素的指纹值直接拼接进行网页相似度的判断. 当相似度达到预定的阈值，就认为静态抓取和前端渲染得到的页面数据是等价的，此时可以断定该层级页面是静态的，否则认为是存在异步获取过程的动态页面. 由于经历了正文抽取、噪声过滤和文本树生成，所有节点的指纹都是相对稳定的，这就意味着两个页面的指纹相似度应该稍高一些，本文测试环境使用的阈值为0.8.


图 2 页面类型判断流程


图 3 相似页面判断算法

电商网站层级严谨，基于此相同层级的页面，自然采取了同样的页面类型，所以图2最后一步中标记的是某一层级而不是某个URL，后面的抓取都沿用本层级已经确定下来的策略即可.

## 3 自动结构化

### 3.1 主题爬虫自动结构化的可行性

主题爬虫的特点除了上文所述稳定的层次分布，更重要的是，其垂直领域的语料和高频度用词都是可预见的，而在技术角度，描述不同业务数据的标识符也是有差异的. 依旧以电商为例，商铺(shop/mall)、商品(product/commodity)、价格(price)、快递(express)、订单(order)等是其行业用语，再具体到页面，title/describe/comment/list/sort等的语义十分明显，不同电商网站的用词基本上是同一套. HTML标签对于数据的展示也是规范的，比如标题用<h>和<p>标签，列表用多个被大 块包含的<li>标签描述.

综上可见，只要提供合适的语料和技术上的规范，制定匹配的策略，就可以找到目标，提取、组合出结构化的数据. 理论上使用机器训练似

乎更加胜任这些预测工作[11], 但是因为主题爬虫的前提下, 人工加权是可以实现的, 而且会比为了支撑机器训练而忙于制造训练数据省力得多.

**3.2 dom**分析和匹配

本文所讨论的自动结构化的关键点, 在于如何实现较为精准的标签匹配, 这里可以通过以下两种方式实现.

**3.2.1** 列表的匹配

此项用来分辨类别、商品列表的数据是在页面的哪一部分. 本文分析多个站点的结构, 如图4是比较典型的一种.

图 4 站点结构举例

依据图4网页中列表的特征: 结构一致、覆盖此页大部分、多用div/ul/li标签, 制定了如图5所示的流程.


图 5 列表匹配流程

同样的, 先去除不关网页结构但占据很多篇幅的代码和文字, 仅仅留下body标签的内容并生成结构树, 其中还要把标签文本去除, 来减少文档体积以提高后期分析效率. 在标识重复相似节点时, 参考了一种基于节点加权的XML检测算法[12]和加权频繁子树相似度的算法[13], 并作出一定改进, 其算法描述如下:

(1) 将HTML文档用SAX (Simple API for XML)转化为一棵带权树, 其中class、name、type等属性应该设置较高的权值, 注意应将相同根节点的同一层次节点的权值之和等于1.

(2) 任意两棵树进行相似性的粗略匹配, 将属性值相等的节点计算相似度: 带权树$T_a$、$T_b$, $N$代表两棵树的节点数, $a_1$–$a_n$和$b_1$–$b_n$代表节点权值, 相似度计算公式: $S(T_a, T_b) = 1 - (\sum_{i=1}^{N} (a_i - b_i))/N$ , 计算得到的相似度如果大于预设的$\alpha$, 认为相似.

(3) 将(2)得到的相似节点对使用树编辑距离算法, 计算后的距离值小于给定的阈值$\beta$, 便最终确认其节点对是相似重复节点.

“判断标签名”这一步是为了解决在网页中发现多片区域出现相似重复节点的情况, 这时应当给ul/li较高优先, 依次类推. 最后确定了列表的位置, 转化为XPath(XML路径语言)并存储, 供后续页面解析进行快速匹配.

**3.2.2** 匹配目标字段的标签

3.2.1节介绍的是如何锁定目标数据的范围, 还有一个问题就是如何抓取最终的有价值字段. 本文基于主题爬虫的特点, 提出一种属性语义匹配的方案, 首先给每个字段建立一个用于预测的词库, 然后进行全部/局部匹配, 计算得到权值之后进行比较以实现预测.

假如要匹配商品名称, 本文设定了词库和权值见表1.

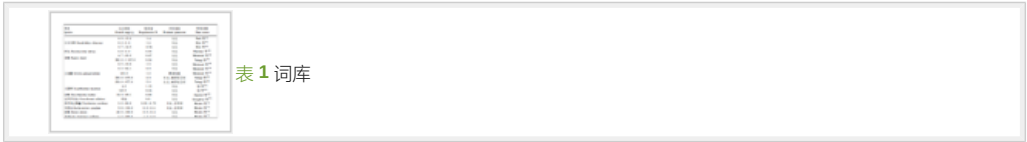

表 **1** 词库

因为代码命名常采用缩写, 当标签的id属性局部匹配(本文推荐50%)时即加上该权值, 一些用词经常是把缩写也纳入词库, 并且权值应该更高. 匹配计算过程如下:

匹配标签1: <p id="prodName"/>

对于product, 局部匹配

对于name, 全部匹配

计算权值: S = 5 + 5 = 10

匹配标签2: 

对于product, 全部匹配

对于name, 不匹配

计算权值: S = 5

因此可以得出结论: 描述商品名称字段的是标签1.

还有一点需要注意的是, 标签描述属性不只是id也可能是name, 还有的情况是自定义的属性, 这就需要在原来的算法上加以延伸, 变成决策树的模型, 这里不再展开讨论.

# 4 系统实现总设计

## 4.1 预设匹配模板

上文提出了自动结构化的一些细节策略和算法, 但一些加权规则和语料是需要人工输入的. 为了实现系统和数据的分离, 提出一种称为匹配模板的预设数据, 其应当包含如下数据:

(1) 列表结构优先匹配排序, 其包含数据如div/ul/li、 div/p、 p/p等.

(2) 节点属性相似度权值, 其包含数据如name(10), id(15), class(10)等.

(3) 目标字段和匹配词库, 其包含字段名和词库(用词和权值).

在实际运行中的匹配模板如图6所示(用XML描述).

```
<xml>
......
    <list-match-sort>
        <item sort="1">div/ul/li</item>
        <item sort="2">div/p</item>
    </list-match-sort>
    <node-similarity>
        <item weight="10">name</item>
        <item weight="15">id</item>
    </node-similarity>
    <fields>
        <field name="prodName">
            <word weight="5">product</word>
            <word weight="5">name</word>
        </field>
        ......
    </fields>
......
</xml>
```

图 6 预设匹配模板

## 4.2 结构分析结果复用机制

4.1节所述的利用模板分析出目标字段的匹配路径, 实际上是主题爬虫中对于多个源站的差异部分, 而剩下的爬虫程序运作是高度一致的, 无需重新组织代码. 遇到一个陌生站点, 只需在爬取前进行模板分析, 得到目标字段的位置信息(本文采取XPath), 那么后续开启的整个爬虫程序只要沿用他们即可, 在性能上和与原爬虫的协作设计上都不会有影响.

如果在后续还需要支持分时段和分布式, 那么结构分析结果还应该保存下来, 实现不同时段、 不同机器间的复用.

## 4.3 系统架构设计

综上所述, 本文实现了如图7所示架构的系统.

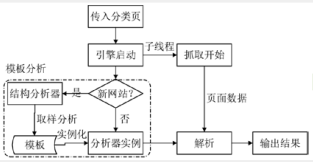


图 7 系统架构

系统简述:

(1) 使用者传入一个入口地址触发本系统, 抓取工作同期开启, 它等待主线程的任务.

(2) 较于普通爬虫系统新增了模板分析部分, 引擎会先进行判断, 如果是旧网站, 则使用以前分析产生的规则.

(3) 遇到新的站点, 则交由结构分析器, 根据预设的匹配模板, 逐步分析出列表数据、 目标字段的位置信息, 将产生的XPath存储在分析器实例中.

(4) 开启爬虫的运作流程, 根据层级提交给分析器解析出所需字段, 完成结构化.

# 5 实验

## 5.1 运行测试

选取了几个主流的电商网站进行爬取, 系统能够正常运行. 图8为国美网站的分析器实例得到的XPath.

```
sort-name://*[@id="0"]/ul[N]/li[N]/div/a[1]/text()
sort-href://*[@id="0"]/ul[N]/li[N]/div/a[1]/@href
list-href://*[@class="product-item"][N]/div/div/p[2]/a/@href
prod-name://*[@id="gm-prd-main"]/div[1]/h1/text()
prod-price://*[@id="prdPrice"]/text()
```

图 8 分析器实例XPath

为了方便展示, 将图8的数据用一个脚本进行树形输出, 并且仅选取几个子节点, 得到的结构化数据如图9所示.

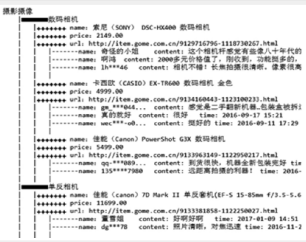

图 9 结构化数据展示

可以看到, 所实现的系统抓取到的数据已经被结构化, 且表现良好.

**5.2** 错误率测试

本文提出的多种匹配方案都包含了加权预测的过程, 会有一定的错误率, 这个指标是评价本套方案优良的关键.

这里采取的测试手段是: 将抓取到的商品的价格进行正确性判断, 当其为乱码、空文本、非数字时, 可以断定结构化出错. 表2是多个站点的测试结果.

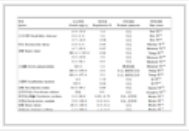

表 2 错误率测试

可以看到, 大型站点的错误率较高, 而ECShop这类通用商城程序的结构化效果很理想, 原因在其设计更加规范, 命名的可读性较强. 总的来说, 本文的方案表现良好, 在词库和权值参数的调优上还有一定的进步空间.

**5.3** 速度测试

自动结构化抓取的意义在于, 能够在更短的时间内获得更多的数据. 本次实验进行爬取速度的测试, 将提出的自动结构化爬虫系统与手工编写抓取规则的专用爬虫系统进行对比, 在同一时刻对京东数据进行采集, 并实时统计抓取的商品数量. 运行结果如图10所示.

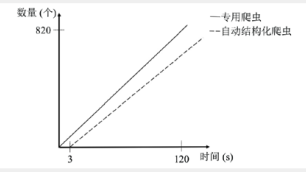


图 10 两种爬虫系统的爬取速度对比

结果表明, 专用爬虫自系统启动就一直稳定抓取, 而自动结构化爬虫会滞后一段时间, 随后其曲线和专用爬虫十分相近. 原因是显而易见的, 自动结构化的系统需要进行模板分析过程, 在分析之后会生成匹配规则, 为后续的页面数据匹配过程所使用, 此时系统的运行逻辑和专用爬虫无异. 对于电商整站采集的总时间(可能长达数小时)来说, 一开始数秒的延后时间是可以据略不计的, 而其却可以轻易迁移到别的站点, 又省下了大量开发时间, 随着源站需求的增加, 自动结构化系统的优势会更加明显.

## 6 结语

本文提出了一种针对电商网站可以自动结构化数据的主题爬虫方案, 由于网站结构严谨的层次划分和网页代码的命名规范性, 使得页面类型、结构分析和标签匹配的自动化成为可能, 这里使用了相似重复判断和目标字段标签匹配的算法作为支撑, 实验证明提出的方案具有可行性和优良性.

本文亦将方案进行系统实现, 其架构是在普通爬虫系统基础上新增了模板分析的部分, 在初次爬取某一站点前, 分析得到所有目标字段的匹配路径, 剩下的工作跟普通爬虫系统无异, 即性能和系统逻辑不会受到影响.

除电商网站外, 其他主题爬虫亦可针对其领域进行层次、结构和字段语料的分析, 实现自动结构化以节省人工编码成本.

参考文献

[1] Lane J, Kim HJ. Big data: Web-crawling and analysing financial news using RapidMiner. International Journal of Business Information Systems, 2015, 19(1): 41-57. DOI:10.1504/IJBIS.2015.069064

[2] 黄聪, 李格人, 罗楚. 大数据时代下爬虫技术的兴起. 计算机光盘软件与应用, 2013, 16(17): 79-80, 83.

[3] 董浩然, 谢欢, 陈鹏, 等. 基于GIS主题爬虫的在线房产估价系统与优化. 地理信息世界, 2016, 23(2): 107-112.

[4] 林晶, 彭小宁. 基于主题语义URL的信息搜索方法研究. 计算机应用与软件, 2015, 32(6): 42-45.

[5] Ahmed A, Ogunbiyi O, Aduragba T. Optimal data collection from a network using probability collectives (swarm based). International Journal of Robotics Research and General Science, 2015, 3(4): 49-58.

[6] 孟繁疆, 姬祥, 袁琦, 等. 农产品价格主题搜索引擎的研究与实现. 东北农业大学学报, 2016, 47(9): 64-71.

[7] Zhang XS, Wang HL. AJAX crawling scheme based on document object model. Proceedings of the Fourth International Conference on Computational and Information Sciences. Chongqing, China. 2012. 1198–1201.

[8] 段子飞. 支持Ajax的Deep Web网络爬虫系统的设计与实现[硕士学位论文]. 广州: 华南理工大学, 2015.

[9] 钱程, 阳小兰. 一种支持Ajax框架的网络爬虫的设计与实现. 计算机与数字工程, 2012, 40(4): 69-71, 98.

[10] 熊忠阳, 牙漫, 张玉芳. 基于网页正文结构和特征串的相似网页去重算法. 计算机应用, 2013, 33(2): 554-557.

[11] Fu TJ, Abbasi A, Zeng DD, et al. Sentimental spidering: Leveraging opinion information in focused crawlers. ACM Transactions on Information Systems, 2012, 30(4): 24.

[12] 孙娜, 吴兰兰. 一种节点加权的相似重复XML数据检测算法. 计算机光盘软件与应用, 2014, 17(2): 99-100.

[13] 郝志峰, 袁琴, 蔡瑞初, 等. 基于加权频繁子树相似度的网页评论信息抽取. 计算机应用研究, 2017, 34(6): 1636-1639, 1658.