深层网站 Ajax 页面数据采集研究综述*

杨俊峰，黎建辉，杨风雷
（中国科学院计算机网络信息中心，北京 100190）

摘　要：如果能够提高网络爬虫采集 Ajax 网页数据的能力，必然会提高搜索引擎的覆盖率和准确率。因此，深层网站 Ajax 页面数据采集成为当前网络爬虫技术研究的热点之一。从深层网站 Ajax 页面数据采集的研究目标、近年来国内外采取的研究方法和取得的成果（研究领域、采集流程、支撑技术）、未来的研究方向三个方面进行了综述。

关键词：Ajax；深层网；Web 2.0；数据采集
中图分类号：TP391　　**文献标志码**：A　　**文章编号**：1001-3695(2013)06-1606-05
doi:10.3969/j.issn.1001-3695.2013.06.002

Survey on research of data collection from supporting Ajax technology deep Web sites

YANG Jun-feng, LI Jian-hui, YANG Feng-lei
(*Computer Network Information Center*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

Abstract: If researchers can improve the ability of data collection from supporting Ajax technology deep Web sites, it is bound to improve the coverage and accuracy of search engine. Therefore, data acquisition from deep Web sites supporting Ajax technology is becoming one of the hot spots of current Web crawler technology. This paper first elaborated the research target of data collection from supporting Ajax technology deep Web sites, then introduced the recent advances achieved and research methods used at home and abroad, including the field of research, data collection process flow and the relevant supporting technology. At last, it discussed the new research trends.

Key words: Ajax; deep Web; Web 2.0; data collection

0　引言

网站应用开发从传统静态页面发展到了具有以丰富的浏览器体验、社会化网络、海量数据处理等为特征的 Web 2.0。其中 2005 年左右兴起的 Ajax（asynchronous JavaScript and XML）[1]技术起到了巨大的推动作用，其突破性地采用局部刷新技术，避免每次从服务器获取整个页面内容，从而降低了服务器负荷、节约了网络带宽、提升了用户体验，典型的应用包括 Google Mail 和 Google Maps。以 Ajax 技术为代表的使用 JavaScript 脚本语言的方法在 deep Web（深层网）网站开发中得到了广泛应用，但是由用户触发 UI（用户界面）事件，执行客户端代码，改变页面当前状态而获取页面动态内容的方式给以 Google 为代表基于 multi-page interface 模型的传统搜索引擎带来了巨大挑战，基于成本和商业因素考虑，传统搜索引擎忽视或很少支持采集动态页面的内容[2]。随着 Ajax 类型网站逐渐增多，势必会扩大 deep Web 里包含的可访问信息容量是一般 surface Web（表层网）的 400～500 倍[3]这一鸿沟。

由此可见，传统网络爬虫面临的挑战迫切需要一种能够有效处理 Ajax 动态脚本网页的方法。如果能够提高网络爬虫采集深层网站 Ajax 页面数据的能力，必然会提高搜索引擎的覆盖率和准确率。因此，解析 JavaScript 代码执行流程并获取页面中异步传输的内容成为当前网络爬虫技术研究热点之一。

1　深层网站 Ajax 页面数据采集研究目标

深层网站 Ajax 页面数据采集的实现依赖于 Ajax 网络爬虫，依据适用范围和可扩展性不同分成两类：针对特定深层网站的专用 Ajax 网络爬虫和通用型 Ajax 网络爬虫[4]。专用 Ajax 网络爬虫无须对页面进行渲染并动态执行 JavaScript 脚本，但通用性差、逆向寻找 URL 地址跳转规律效率低。因此，目前研究重点集中在通用型 Ajax 网络爬虫方面。深层网站 Ajax 页面数据采集是比较新的领域，文献[4～12]从不同角度探讨了其需要解决的问题，如图 1 所示。

1）Ajax 网站、网页建模优化　支持 Ajax 技术的深层网站的 Ajax 页面成为采集的基本处理单元，传统页面之间的超链接被 JavaScript 脚本触发的 DOM（文档对象模型）事件所取代，如何刻画页面之间的拓扑关系，以及如何分析页面状态转换规律，需要对整个深层网站 Ajax 页面进行建模并优化。

2）Ajax 页面抓取策略　利用最小代价在优化后的模型上采集与主题相关的 Ajax 页面，遍历 Ajax 页面状态并获取 Ajax 页面内容。需要 Ajax 网络爬虫采用合理的页面抓取策略，避

收稿日期：2012-10-23；**修回日期**：2012-12-11　**基金项目**：中国科学院“十二五”信息化建设专项基金资助项目（Y107041108）；中国科学院计算机网络信息中心主任基金资助项目（Y013041108）

作者简介：杨俊峰（1984-），男，四川广元人，硕士研究生，主要研究方向为大数据处理、云计算（yangjunfeng@cnic.cn）；黎建辉（1973-），男，正研级高工，博导，博士，主要研究方向为海量数据管理、处理与挖掘分析的理论、方法与关键技术；杨风雷（1973-），男，副研究员，博士，主要研究方向为数据挖掘、分布式计算等.

免 Ajax 爬虫陷入死锁状态或多次重复采集,从而导致数据冗余。

3)页面状态的标志　传统的 Web 网站基于 URL 模型,每个静态页面赋予唯一的 URL 地址,在 Ajax 应用中,不是每一个状态变化都会用 REST-based[13] URL 来标记,页面可能包含很多状态,并且页面的变化不是根据 URL 改变而是通过 DOM 结构的动态变化来展现;页面所呈现的内容也与客户端当前状态有关。因此,Ajax 爬虫需要模拟用户操作触发相应事件,利用 DOM 结构来标志页面状态。

4)页面状态的转换　深层网站 Ajax 页面数据采集必须解决 DOM 事件的自动处理和分发问题,传统的页面状态转换通过切换不同的 URL 达到目的,在 Ajax 应用中不同的页面状态可能拥有相同的 URL;并且 Ajax 应用不支持状态历史,触发返回事件不一定能正确回退到前一个页面状态,Ajax 爬虫必须能够进行状态的程序可控性转换。

5)页面状态内容的动态获取　Ajax 页面状态内容的动态获取是深层网站 Ajax 页面数据采集的直接目的,如果搜索引擎忽略动态生成内容,会造成典型的 false negatives 和 false positives 情况[8]。因此,介于浏览器端和 Web 服务器之间的 JavaScript 引擎不仅需要能够处理用户的 UI 事件和与服务器异步交互的信息,还需要执行 JavaScript 脚本,完成 Ajax 页面状态的转换,呈现给用户完整的页面内容。

6)页面状态的重复检测　Ajax 应用中某些事件可能是同一 JavaScript 函数来处理,触发这些事件也可能导致相同的页面状态,Ajax 爬虫必须检测出重复状态,减少数据冗余,消除重复状态的二次采集。此外,还需要区分与状态变化无关的页面内容,如随机数和数字时钟等,避免引发状态爆炸。

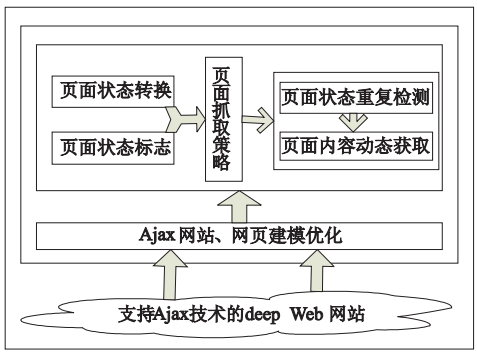

图1　深层网站Ajax页面数据采集

2　深层网站 Ajax 页面数据采集研究进展

Ajax 技术的广泛应用推动了国内外学者及研究机构对深层网站 Ajax 页面数据采集的研究。国外进行研究的主要机构有瑞士苏黎世联邦理工学院(ETH Zurich)和荷兰 Delft 的理工大学软件研究中心(TUD-SERG)。国内主要是中国科学院计算机技术研究所、浙江大学及中国科学技术大学等研究机构。

Google 爬虫支持 Ajax 网页的动态采集,但是需要被采集的深层网站遵循 Google 推荐的一系列规范[14],相对来说增加了普及推广的难度。当前学者们对深层网站 Ajax 页面数据采集取得了以下的研究成果。

2.1　Ajax 网站、网页模型化

深层网站 Ajax 页面数据采集可以看成是对特定模型的遍历过程。对 Ajax 应用来说,浏览器与服务器端可以交换少量必要的数据并动态更新页面内容,因此,转换过程不仅可以在页面之间进行,也可以在页面内部由 Ajax 事件触发[4]。文献[9]给出了典型的 Ajax 网站模型,如图 2 所示。

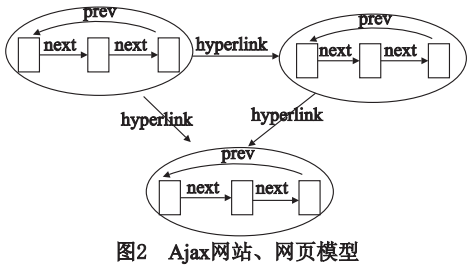

图2　Ajax网站、网页模型

在图 2 中,椭圆代表传统的 URL 标志的页面,方框表示 Ajax 页面的某一特定的状态。由超链接导致的页面跳转问题已经在传统的数据采集研究中得到了较好的处理,故研究的重点集中在 Ajax 页面建模方面。主流的研究方法是采用有限状态机(FSM)。

文献[5,7,9,15,16]对 Ajax 页面建模先后进行了探索总结。文献[7]将 Ajax 应用定义为一个规范的有向图结构 $DG(V,VR)$,称为状态转换图。其中顶点集 $V=\{v|$ 每一个 v 代表某一个页面状态$\}$;边集 $VR=\{\langle v,w\rangle|v,w$ 存在某个转换使得起始状态经过转换能够到达结束状态$\}$。文献[15]也采用了类似的二元组$\langle V,E\rangle$结构来对 Ajax 页面建模。

文献[5]进一步将状态转换图扩展成四元组$\langle r,V,E,\pounds\rangle$结构。其中:$r$ 为 Ajax 页面载入浏览器的初始状态;V 为页面执行过程中的 DOM 状态集合;E 为链接两个 DOM 状态的可点击元素事件集;$\pounds$ 为赋予可点击元素的事件类型和 DOM 属性。如图 3 所示,这样的有向图是非简单、闭合的,但是无法从后面的状态返回到前面的状态,使得状态间转换路径偏长。

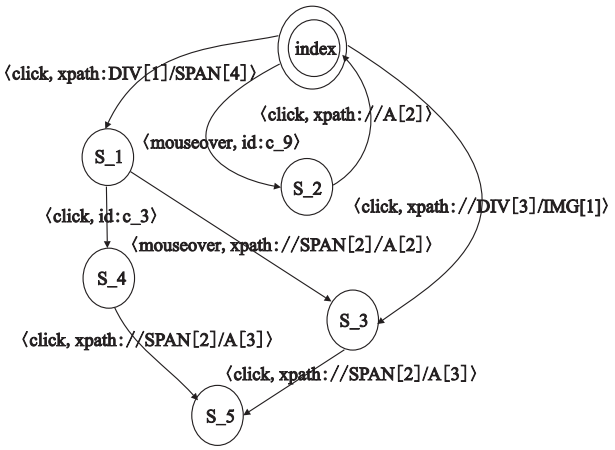

图3　Ajax页面状态转换图

文献[16]则考虑了代码静态与动态分析的结合并提炼出抽象的状态转换图。文献[17]进一步将引起 Ajax 页面状态转换的事件用三元组〈HTML,JavaScript,XPath〉标志并以其在 Ajax 爬虫中扮演的角色不同分为重置、禁止、与状态无关、与状态有关的事件,并进一步将状态转换图分为必须遍历的核心状态转换图和带有重置操作边的完全状态转换图。

2.2　Ajax 网页抓取策略

Web 网络爬虫的抓取策略是指如何根据当前 URL 地址来选择访问地址先后的一种准则[10],抓取网页一般采用的策略有广度优先、深度优先、最佳优先等。当 Ajax 网站模型化以后,Ajax 爬虫抓取策略问题就转换为对模型的遍历问题。

Frey[8]在算法中采用了广度优先的抓取策略,通过对 Ajax

页面中脚本执行解析得到动态内容,时间复杂度较高。随后该机构的 Duda 等人[9]在其基础上作了改进,提出了基于脚本以及参数的热点检测机制采集动态信息,在 Ajax 爬虫系统架构中增添了 cache 模块,引进了 JavaScript 调用图的"hot node"来避免重复调用两次从服务器端获取内容的 Ajax 请求函数,但是 Ajax 页面状态的缓存不但要考虑当前 DOM 状态,JavaScript 运行环境也需要一起存储,工程上实现起来很困难。

Mesbah 等人[5]提出了利用浏览器 API 模拟用户对页面元素进行操作来获取动态信息,采用全方位探测扫描的方式完成 Ajax 页面的信息采集,并实现了开源项目 Crawljax[18]。其采用了深度优先的抓取策略,采集深度的阈值需要预先设定。基于 Crawljax 已经完成了许多 Ajax 测试[5],它能够有效地采集动态页面的内容。但是其状态回退算法采用页面重新加载方式是影响性能的;此外,没有考虑避免获取与 Ajax 页面状态不相关内容这一问题。

前面所述抓取策略是基于图论中广度优先或深度优先算法的改进,而大型的 Ajax 网站包含了许多 Ajax 页面状态,并且 Ajax 应用不支持状态历史,如果遍历这些页面状态、到达某一个页面状态而必须重新加载整个页面会导致 Ajax 爬虫性能大幅下降,文献[11,17]围绕这些问题进行了下面的改进。

文献[19]认为文献[5,8,9]中提到的抓取策略是非高效的,其将引起当前 Ajax 页面状态改变的原因归结为通过同步或异步同服务器端的 HTTP 请求交互及客户端 JavaScript 代码的执行。Ajax 网页抓取策略需要考虑这两类改变方式。因此,提出了基于 URL 和基于事件整合的抓取策略,创造性地提出了 hypercube(超级立方体)的概念。但是这种抓取策略过于复杂,工程上难以推广应用,在文献[11]中作者也只实现了最大 16 维的 hypercube。

文献[17]将 Ajax 页面状态的抓取转换为 DRPP(directed rural postman problem),在完全状态转换图拓扑已知的情况下,核心状态转换图增加少量取自它的边后成为平衡、强连通的,则后者的欧拉回路为经过顶点最少的搜索路径。当拓扑图未知,算法需要探测且复杂度与它的拓扑有关,并证明了任何图基于探测的算法时间复杂度上限为 $O(\min\{mn,(d+1)n^2\})$。但该抓取策略有许多应用假设,如没有考虑 Ajax 页面状态动态更新。

2.3 Ajax 页面状态标志

Ajax 页面可以包含多个状态变化,Ajax 页面状态标志用于实现状态的引用和定位[4]。页面状态的标志需要保证状态的可达性、可理解性、可重现性,并易于理解应用。围绕这一问题的研究可以分为两大类:

a)对相邻的状态之间转换过程形式化定义,间接实现页面的状态标志。文献[12]采用四元组〈事件源元素,事件类型,事件目标元素,事件动作〉来刻画状态转换;文献[15]采用事件和事件绑定的元素 XPath 二元组表示法;文献[5]提出了用四元组〈根节点,DOM,可点击元素,事件属性〉来标志 Ajax 页面状态。可以看出,上述方法都会涉及 XPath 技术。文献[20]则应用了 OXPath 技术来采集深层网站。

b)直接设计数据结构用于标志 Ajax 页面状态。文献[21]采用六元组〈页面状态初始 URL,状态锚文本,XPath 列表,DOM 树,状态 hash 值,处理标志〉来标记每个 Ajax 页面状态。但是这种表示方法默认只能处理元素的点击事件,要达到标志通用的 DOM 事件则需要增加更多属性。

2.4 Ajax 页面状态转换

Ajax 页面位于某一个特定的状态时,程序需要能够自动跳转到新的页面状态而非通过人工方式实现,这依赖于客户端 JavaScript 脚本的异步执行。因此,实现 Ajax 页面状态的转换必须先解决 JavaScript 脚本的正常执行与解析。

Frey 和 Duda 等人[8,9]对开源的 JavaScript 解释器 Rhino[22]进行改进实现了脚本执行和状态转换功能,对开源的 Cobra[23]工具包进行了扩展实现了页面的动态加载和解析,通过 HTML 内部形成的 DOM 结构获取状态所包含的数据内容;金晓鸥等人[24]提出了基于 Rhino 实现 JavaScript 动态页面解析的整体方案,实现了链接地址和页面主题内容的获取。但是不容易完整重现脚本解释器且存在许多脚本代码无法正常执行的情况,许多学者尝试利用嵌入式浏览器组件实现 Ajax 页面的渲染呈现和脚本执行。

文献[5]的开源项目 Crawljax[18]采用 Mozilla XULRunner[25]来实现内置浏览器功能,其整合了 Webclient[26]操作 DOM。文献[6,10]实现了事件驱动的页面采集方法,采用 Watir[27]控制和启动操作系统上的 IE 浏览器显示 Ajax 网页,分析脚本中的异步请求函数,然后驱动浏览器模拟执行异步请求,等待浏览器更新完成从而得到 Ajax 网页的完整内容。此外,rbNarcissus[28]也常用来处理 JavaScript 代码执行问题。

罗兵[29]采用基于 HTTP 驱动的页面采集方法,获取 Ajax 页面引用的 JavaScript 文件,并通过自己设计的脚本解释器模拟浏览器行为对脚本代码顺序执行,实现了 Ajax 页面状态的控制转换。肖卓磊[30]、战琴[31]在后来的研究中也采用类似的处理方式。曾伟辉等人[32]基于切片算法构建程序层次模型,有效解决了 JavaScript 脚本的有序执行问题。但是他们都存在同样的问题即不支持事件触发。文献[33]则整合了协议驱动与事件驱动两种爬行方式,抛开了 JavaScript 脚本的分析和 DOM 支持维护,避免了异构、复杂或加密的页面可能产生的错误。

在 Ajax 页面转换控制方面,较简单的方法是采用事件过滤机制,然后触发过滤后的事件集[4]。Duda 和 Xia 等人[9,21]分别提出了支持用户自定义规则,引入了接受规则和拒绝规则,状态转换只能在有限集合中进行。夏冰等人[15]则将 Ajax 页面处理分为训练和应用两个阶段,训练阶段先对各种类型页面样本依次触发所有可能事件,记录页面变化情况,对页面有效元素的 XPath 和事件类型进行规约并总结出特征;应用阶段只触发规约出的事件集,大大降低了触发事件数。Mesbah 等人[5]使用全自动扫描、HTML 元素注解和面向领域的手工配置三种不同方式控制页面状态转换。Duda 等人[9]则提出采用启发式爬行算法,避免从不同路径转入到相同的页面状态。

2.5 Ajax 页面状态重复检测

Ajax 页面状态重复检测能够有效消除重复状态的二次采集,减少数据冗余,提高采集效率。传统的页面重复检测通过判断新的 URL 是否在静态 URL 池中出现即可,Ajax 应用状态的重复检测主要是通过比较两个 DOM 树的结构和内容来实现。

文献[9]将状态重复细分为语法重复和语义重复,采用计算 Ajax 页面内容 hash 值是否等同作为判重依据,方法简单但与状态内容无关的页面元素变化引起的判重操作是没有实际

意义的,会产生状态爆炸问题[19]。文献[7]认为解决此问题可以将 Ajax 应用的状态变化定义为 Web 页面行为的变化,即只关注 DOM 元素节点与事件的关联情况是否发生改变或是否产生新的 DOM 元素节点。

基于树编辑距离算法[34]被广泛应用于两个 Ajax 页面状态相似的度量,如文献[5,7,10]。文献[5]还提出了一种允许两个 DOM 树有一定的互异性方法。利用 Levenshtein[35]算法计算两个 DOM 树的编辑距离,如果其小于一定阈值(如[0,1]),则两个页面状态视为相同;文献[21]则把节点权重纳入计算过程,改进了树编辑距离算法的应用效果;文献[16]将 DOM 文档分成含有权重的特征集合,利用 SimHash 算法计算出数字指纹并比较指纹间的海明距离完成判重操作。此外,编辑距离及其改进算法[36]、基于 Path Shingles[37]的方法也可用于 Ajax 页面状态重复检测过程[21]。

2.6　采集处理流程

处理流程研究方面,文献[29]设计的 Ajax 网络爬虫包含网页抓取、网页分析、JS 解析、DOM 支持和页面生成五个模块。文献[32]进一步将 JS 解析模块细分为 JavaScript 编译、字节码解释执行、Atom 管理、垃圾收集以及标准类管理模块。Crawljax[18]采集流程则包括初始化嵌入式浏览器、初始化状态机、获取候选可点击元素、触发事件、获取 DOM 内容、更新页面状态等步骤并实现了多线程和多浏览器实例功能。文献[21]进一步引入了用于存储状态标志和内容的状态仓库概念,在采集流程中增加了初始状态注入和状态检测与储存处理模块;文献[13]则明确给出了事件列表需要循环触发的处理步骤。文献[4]对深层网站 Ajax 页面数据采集的处理流程做出了归纳总结,如图 4 所示。

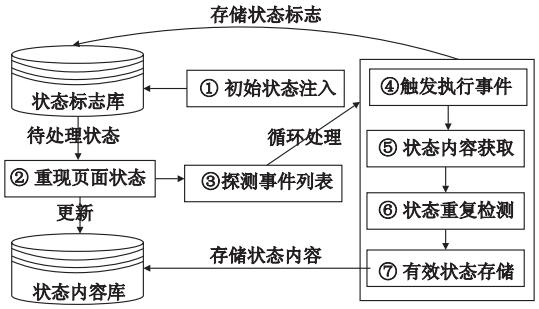

图4　采集处理流程

图 4 所示的采集处理流程涵盖了当前研究所提到的主要步骤,采集入口是通过初始状态注入加入到状态标志库(存储 Ajax 页面状态),形成了初始的待处理页面状态集合(步骤①)。Ajax 运行容器重现每一个待处理的页面状态,探测可点击元素和可导致状态转换的 DOM 事件集合(步骤②③);针对每一个 DOM 事件,通过触发执行状态转换并获取动态页面内容,再经过页面状态重复检测判定新状态是否有效(步骤④~⑦);有效则将页面新的状态存储到状态标志库,同时获取页面内容并存储进状态内容库(存储 Ajax 页面内容),页面状态内容也可以由步骤②重现页面状态时更新到状态内容库中,具体方式在不同的研究中有不同实现[4]。

上述流程是一个不断循环的过程,为终止循环研究人员已经提出多种不同的策略[5,7,12,21]:当前页面状态与初始状态相对深度超过一定阈值;有效 Ajax 页面数量达到指定数值;页面处理时间超时;用户自定义的其他终止规则等。

2.7　采集支撑技术

在 Ajax 页面状态标志研究方面,主流的方法是采用 XPath[38]和 OXPath[39]。XPath 是 XML 文档中进行查询的语言,它用路径表达式来选取 XML 文档中的节点或节点集,XPath 同样可以运用在 HTML 中。OXPath 扩展了 XPath 的功能,广泛用于复杂的 deep Web 网站数据采集研究中[20]。

在 Ajax 页面状态重现、事件触发执行和 Ajax 页面动态内容获取方面,这些都依赖于对页面 HTML 代码的渲染呈现和 JavaScript 脚本的解释执行[4],采用的方法大致可以分为两类:爬虫模块中调用 Rhino 自动实现 JavaScript 脚本执行解析,以及利用嵌入式浏览器作为 Ajax 运行容器,通过 HTML 渲染器提供的 API 与页面 DOM 树交互获取完整数据内容[4,5,7,33]。

从整体应用来看,采用嵌入式浏览器方式应用效果更加明显,因为即使多年维护的 Rhino 项目依然存在许多脚本代码无法执行情况,并且 Web 爬虫架构中添加脚本代码解释器会严重影响爬虫性能。当前应用较为广泛的浏览器渲染引擎可以分为三类:基于 Mozilla Gecko 浏览器内核、基于微软 IE 和开源的渲染器。具体选用何种浏览器内核,应该根据待抓取 Ajax 站点的 Web 标准规范化程度、效率、易用性、可扩展性等因素综合考虑。

1)基于 Mozilla Gecko 内核的 HTML 渲染器　通过对 Mozilla 组织开发的 Gecko 内核进行封装,提供 HTML 渲染器 API 调用。Gecko 内核具有跨平台、标准化高、开放性等特点,因而得到业界广泛应用。代表性的项目有 WebRenderer[40]、WebClient[26]、JRex[41]、XULRunner[25]。WebRenderer 是由 JadeLiquid 发布的商业软件,提供了对 HTML 渲染、访问内部 DOM 树、执行脚本等功能的完整支持,后者属于开源实现,但易用性方面略逊一些。

2)基于 IE 内核的 HTML 渲染器　底层采用 IE 内核,集成封装提供 HTML 渲染服务,其典型代表有 JExplorer[42]、Watij[43]、Watir[27]。JExplorer 提供了一组便捷的 Java 应用程序接口,用户无须深入掌握 COM(组件对象模型)技术就可以通过 API 访问渲染后的网页 DOM 对象。Watij 底层采用了 JExplorer,因其具有丰富的 HTML 访问接口,其在 Ajax 数据采集中也得到广泛应用[5,10,21]。Watir 是比 Watij 更早的 Web 自动测试框架,但其面向的开发语言为 Ruby。

3)开源实现的 HTML 渲染器　基于 Gecko 或 IE 内核的渲染器依赖于特定的软件环境,开发扩展比较困难,脚本事件的调试不便促使部分研究人员采用开源实现的 HTML 渲染器[9,12]。此类渲染器主要用 Java 语言实现,典型代表有 Cobra[23]、HtmlUnit[44]、Flying Saucer XHTML renderer[45]。Cobra 是采用 Java 语言实现的 HTML 渲染器和 DOM 解析器,其脚本执行基于 Rhino[22],CSS 解析采用 CSS parser[46]。HtmlUnit 底层采用 Cobra,屏蔽了图形用户接口同时提供了 API 供上层模块调用。XHTML renderer 目前仅能对结构化程度高的 XHTML 和 CSS 进行渲染,暂时不支持 JavaScript,需要添加额外的脚本解析模块。

3　深层网站 Ajax 页面数据采集研究趋势

从网络信息生产的趋势看,规模越大的信息更趋向于采用深层网站。而现有大部分的网络爬虫都无法对支持 Ajax 技术的深层网站数据进行有效采集,深层网站 Ajax 页面数据采集

研究将趋向于以下七个方面：

a) Ajax 网站、页面建模问题。目前，研究中所采用的状态转换图、有限自动机等方式并不能完整呈现 Ajax 页面的关系，Ajax 页面模型需要足够简单并且信息完备、工程上能够推广应用、探索一致认可的 Ajax 页面模型及其对应的数据结构十分必要。利用 ActiveXML[47] 建模等方法可以作为参考。

b) Ajax 页面状态标志问题。研究中采用的 XPath、OXPath、正则表达式或 HTML 元素定位方法还不能够作为页面状态独立完整的标志，存在属性数目过于繁琐或表达式过于简单难以包含完备的信息等问题。探求认可的 Ajax 页面状态标志和数据结构是研究的必要。

c) Ajax 页面抓取策略问题。基于图论中广度优先、深度优先、启发式搜索或图算法的改进策略会遍历与采集结果无关的 Ajax 页面状态，导致采集时间过长，性能下降。如何在 Ajax 模型上采用最优化的搜索策略采集用户需要获取的数据结果是需要研究的问题。

d) Ajax 页面状态重现问题。该问题既包括在 Ajax 运行容器中重构特定标志所对应的运行状态及上下文环境关系，也包括如何面向用户呈现所需要的数据[4]，可以采用分类、聚类技术对相似结果进行处理。

e) Ajax 页面状态的事件触发和执行问题。自动采集方式会遍历大量与采集无关的 Ajax 页面状态，导致状态膨胀问题出现，工程上难以大规模实际应用；当前 HTML 渲染器存在速度、控制粒度、跨平台、交互性等不足，可以通过设计 Ajax 运行容器的插件以实现浏览器渲染结果的直接交互，MIT 设计的 Crowbar[48] 和开源的 FireWatir[49] 可以提供研究参考。

f) 深层网站 Ajax 页面数据采集个性化。当前其数据采集没有和垂直搜索、个性化推荐技术相整合，根据用户或用户群体的特征，采集与其相关度高的 Ajax 页面状态和 DOM 事件，可以提高采集质量和性能，提升用户体验。

g) 深层网站 Ajax 页面数据分布式采集。基于传统 URL 的 deep Web 数据分布式采集已经得到了很好的研究成果，但是不能运用在以事件触发的 Ajax 应用采集中去。如何实现其数据分布式采集，采用现有的大规模数据处理方法是工程上提高采集能力的必要。

4 结束语

随着 Ajax 应用网站的大量涌现，深层网站 Ajax 页面数据采集已经开始成为搜索引擎发展的热点之一。本文首先阐述了其研究目标，然后对国内外现有的研究方法和技术进行了系统分析，希望通过本文能够对这一领域的研究有比较清晰的概括和总结。深层网站 Ajax 页面数据采集研究仍然处于探索发展阶段，还没有形成工程上大规模应用。随着 Web 2.0 及其相关技术的进一步普及，这一领域必将引起研究人员的更多关注。

参考文献：

[1] GARRETT J J. Ajax: a new approach to web applications[EB/OL]. (2005-02-18)[2010-01-15]. http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[2] 张成奇. 支持 Ajax 的 deep Web 爬虫设计与实现[D]. 上海：上海交通大学，2009.

[3] BERGMAN M K. The deep Web: surfacing hidden value[R/OL]. [2012-06-18]. http://www.brightplanet.com/2012/06/the-web-surfacing-hidden-ralue/.

[4] 夏天. Ajax 站点数据采集研究综述[J]. 现代图书情报技术，2010(3)：52-57.

[5] MESBAH A, Van DEURSEN A, LENSELINK S. Crawling Ajax-based Web applications through dynamic analysis of user interface state changes[J]. ACM Trans on the Web, 2012, 6(1).

[6] SHASH S. Crawling Ajax-driven Web 2.0 applications[R/OL]. (2007-02-14)[2012-08-10]. http://www.infosecwriters.com/text_resources/pdf/Crawling_AJAX_SShah.pdf.

[7] 郭浩，陆金良，刘金红. 一种基于状态转换图的 Ajax 爬行算法[J]. 计算机应用研究，2009，26(11)：4266-4269.

[8] FREY G. Indexing Ajax Web applications[D]. Zurich: Swiss Federal Institute of Technology, 2007.

[9] DUDA C, PRE Y G, KOSSMANN D, *et al*. Ajax crawl: making Ajax applications searchable[C]//Proc of IEEE International Conference on Data Engineering. Washington DC: IEEE Computer Society, 2009：78-89.

[10] 管翠花. 支持 Ajax 技术的 deep Web 网络爬虫模型研究[D]. 大连：大连海事学院，2011.

[11] BENJAMIN K. A strategy for efficient crawling of rich Internet applications[D]. Ottawa: University of Ottawa, 2010.

[12] DUDA C, FREY G, KOSSMANN D, *et al*. AjaxSearch: crawing, indexing and searching Web 2.0 applications[J]. Proceedings of the VLDB Endowment Archive, 2008, 1(2)：1440-1443.

[13] FIELDING R T, TAYLOR R N. Principled design of the modern Web architecture[J]. ACM Trans on Internet Technology, 2002, 2(2)：115-150.

[14] Make your Ajax application crawlable[EB/OL]. [2012-08-20]. https://developers.google.com/webmasters/ajax-crawling/docs/getting-started?hl=zh-CN.

[15] 夏冰，高军，王腾蛟，等. 一种高效的动态脚本网站有效页面获取方法[J]. 软件学报，2009，20(z)：176-183.

[16] MARCHETTO A, TONELLA P, RICCA F. State-based testing of Ajax Web applications[C]// Proc of International Conference on Software Testing Verification and Validation. Washington DC: IEEE Computer Society, 2008：121-130.

[17] PENG Zhao-meng, HE Neng-qing, JIANG CHUN Xiao, *et al*. Graph-based Ahax crawl: mining data from rich Internet applications[C]//Proc of International Conference on Computer Science and Electronics Engineering. Washington DC: IEEE Computer Society, 2012：590-594.

[18] Crawljax: automate Ajax crawling and testing[EB/OL]. [2008-09-10]. http://crawljax.com.

[19] BENJAMIN K, Von BOCHMANN G, DINCTURK M, *et al*. A strategy for efficient crawling of rich Internet applications[C]//Proc of the 11th International Conference on Web Engineering. Berlin: Springer-Verlag, 2011：74-89.

[20] Jon SELLERS A. The OXPath to success in the deep Web[C]//Porc of the 20th International Conference on New York: ACM Press, 2011：409-414.

[21] XIA Tian. Extracting multi-records from Web pages[C]//Proc of the 4th International Conference on Semantics, Knowledge and Grid. 2008：396-399.

[22] Mozilla. Rhino: JavaScript for Java[EB/OL]. [2009-03-22]. http://www.mozilla.org/rhino/.

[23] Corbra: Java HTML renderer&parser[EB/OL]. [2009-01-19]. http://lobobrowser.org/cobra.jsp.

（下转第 1616 页）

scale local image decoders[J]. Neuron,2008,60(5):915-929.
[31] WORSLEY K, LIAO C, ASTON J, *et al*. A general statistical analysis for fMRI data[J]. Neuroimage,2002,15(1):1-15.
[32] WORSLEY K J, FRISTON K J. Analysis of fMRI time-series revisited: again[J]. Neuroimage,1995,2(3):173-181.
[33] PEREIRA F, MITCHELL T, BOTVINICK M. Machine learning classifiers and fMRI: a tutorial overview[J]. Neuroimage, 2009, 45(1): 199-209.
[34] KRIEGESKORTE N, GOEBEL R, BANDETTINI P. Information-based functional brain mapping[J]. Proceedings of the National Academy of Sciences of the USA,2006,103(10):3863-3868.
[35] POLYN S M, NATU V S, COHEN J D, *et al*. Category-specific cortical activity precedes retrieval during memory search[J]. Science, 2005,310(5756):1963-1966.
[36] O'TOOLE A J, JIANG F, ABDI H, *et al*. Partially distributed representations of objects and faces in ventral temporal cortex[J]. Journal of Cognitive Neuroscience,2005,17(4):580-590.
[37] CARLSON T A, SCHRATER P, HE S. Patterns of activity in the categorical representations of objects[J]. Journal of Cognitive Neuroscience,2003,15(5):704-717.
[38] HANKE M, HALCHENKO Y O, SEDERBERG P B, *et al*. PyMVPA: a python toolbox for multivariate pattern analysis of fMRI data [J]. Neuroinformatics,2009,7(1):37-53.
[39] MISAKI M, KIM Y, BANDETTINI PA, *et al*. Comparison of multivariate classifiers and response normalizations for pattern-information fMRI[J]. Neuroimage,2010,53(1):103-118.
[40] MITCHELL T M, HUTCHINSON R, NICULESCU R S, *et al*. Learning to decode cognitive states from brain images[J]. Machine Learning,2004,57(1):145-175.
[41] KU S, GRETTON A, MACKE J, *et al*. Comparison of pattern recognition methods in classifying high-resolution bold signals obtained at high magnetic field in monkeys[J]. Magnetic Resonance Imaging,2008,26(7):1007-1014.
[42] DAVATZIKOS C, RUPAREL K, FAN Y, *et al*. Classifying spatial patterns of brain activity with machine learning methods: application to lie detection[J]. Neuroimage,2005,28(3):663-668.
[43] HANSON S J, MATSUKA T, HAXBY J V. Combinatorial codes in ventral temporal lobe for object recognition: Haxby (2001) revisited: is there a "face" area? [J]. Neuroimage,2004,23(1):156-166.
[44] COX D D, SAVOY R L. Functional magnetic resonance imaging (fMRI)[J]. Neuroimage,2003,19(2):261-270.
[45] NASELARIS T, KAY K N, NISHIMOTO S, *et al*. Encoding and decoding in fMRI[J]. Neuroimage,2011,56(2):400-410.
[46] HASTIE T, TIBSHIRANI R, FRIEDMAN J. The elements of statistical learning: data mining, inference, and prediction[M]. Beijing: Publishing House of Electronics Industry, 2004.
[47] KRISHNAPURAM B, CARIN L, FIGUEIREDO MAT, *et al*. Sparse multinomial logistic regression: fast algorithms and generalization bounds[J]. IEEE Trans on Pattern Analysis and Machine Intelligence,2005,27(6):957-968.
[48] PUJARA J. Understanding feature selection in functional magnetic resonance imaging [D]. Pittsburgh: Carnegie Mellon University, 2005.
[49] OLSHAUSEN B A, FIELD D J. What is the other 85% of V1 doing [M]//Problems in Systems Neuroscience. Oxford: Oxford University Press,2004:182-211.

(上接第 1610 页)

[24] 金晓鸥,钟宝燕,李翔. 基于 Rhino 的 JavaScript 动态页面解析研究实现[J]. 计算机技术与发展,2008,18(2):1-4,50.
[25] Mozilla. XULRunner[EB/OL]. [2010-10-10]. https://developer.mozilla.org/en/XULRunner.
[26] Webclient[EB/OL]. (2007-09-23). [2010-02-16]. http://www.mozilla.org/projects/blackwood/webclient/.
[27] Watir[EB/OL]. [2009-11-16]. http://watir.com/.
[28] rbNarcissus[EB/OL]. [2010-03-20]. http://code.google.com/p/rbnarcissus/.
[29] 罗兵. 支持 Ajax 的互联网搜索引擎爬虫设计与实现[D]. 杭州:浙江大学,2007.
[30] 肖卓磊. 基于 Ajax 技术的搜索引擎研究[D]. 武汉:武汉理工大学,2009.
[31] 战琴. 基于 Ajax 技术的 deep Web 爬虫实现方法研究[D]. 济南:山东科技大学,2009.
[32] 曾伟辉,李淼. 基于 JavaScript 切片的 Ajax 框架网络爬虫技术研究[J]. 计算机系统应用,2009,18(7):169-171.
[33] 袁小节. 基于协议驱动与事件驱动的综合聚焦爬虫研究与实现[D]. 长沙:国防科学技术大学研究生院,2009.
[34] REIS D C, GOLGHER P B, SILVA A S, *et al*. Automatic Web news extraction using tree edit distance[C]// Proc of the 13th International Conference on World Wide Web. New York: ACM Press, 2004:502-511.
[35] Levenshtein distance[EB/OL]. [2008-08-14]. http://en.wikipedia.org/wiki/Levenshtein_distance.
[36] MARZAL A, VIDAL E. Computation of normalized edit distance and applications[J]. IEEE Trans on Pattern Analysis and Machine Intelligence,1993,15(9):926-932.
[37] BUTTLER D. A short survey of document structure similarity algorithm[C]//Proc of the 5th International Conference on Internet Computing. 2004: 3-9.
[38] W3C. XPath 2.0 [EB/OL]. [2010-12-14]. http://www.w3.org/TR/xpath20/.
[39] GitHub. OXPath[EB/OL]. [2011-11-17]. https://github.com/diadem/OXPath.
[40] WebRenderer[EB/OL]. [2010-02-16]. http://www.webrenderer.com/.
[41] JRex: the Java browser component[EB/OL]. [2009-06-21]. http://jrex.mozdev.org/.
[42] JExplorer[EB/OL]. [2010-01-29]. http://www.Leamdev.com/jexplorer/.
[43] Watij[EB/OL]. [2009-11-16]. http://watij.com/.
[44] HtmlUnit[EB/OL]. [2010-02-09]. http://htmlunit.sourceforge.net/.
[45] XHTML renderer project[EB/OL]. [2009-07-01]. https://xhtmlrenderer.dev.java.net/.
[46] CSS parser[EB/OL]. [2009-11-16]. http://cssparser.sourceforge.net/.
[47] ActiveXML documentation [EB/OL]. http://webdam.inria.fr/axml/index.axml.html
[48] Crowbar[EB/OL]. [2010-01-16]. http://simile.mit.edu/wiki/Crowbar.
[49] FireWatir[EB/OL]. [2010-01-14]. http://code.google.com/p/firewater/.