文章编号: 1006 - 4729(2011)04 - 0378 - 05

网页数据抽取中 Wrapper 的维护

邓莎莎[1], 李 嘉[2]

(1. 上海电力学院 计算机与信息工程学院, 上海 200090;
2. 华东理工大学 商学院, 上海 200237)

摘 要: 当网页结构发生动态变化时,所构建的网页数据抽取器 Wrapper 往往会失灵. 为了解决这一问题,提出了 Wrapper 维护模型结构. 实验证明,当网页数据结构发生变化时,该模型结构能更有效地支持网页数据的抽取.

关键词: Wrapper 维护; 网页数据抽取; 语义块
中图分类号: N37 文献标志码: A

Wrapper Maintenance During Web Data Extracting

DENG Sha-sha[1], LI Jia[2]

(1. *School of Computer and Information Engineering, Shanghai University of Electric Power, Shanghai* 200090, *China*;
2. *Business School, East China University of Science and Technology, Shanghai* 200237, *China*)

Abstract: Currently when the web page structure experiences dynamic change, the web data extraction devices often fail to work. In order to solve the problem, Wrapper maintenance model is proposed. Experimental comparison shows that this model could more effectively help web data extraction when data structure is different.

Key words: Wrapper maintenance; web data extracting; semantic block

随着 Internet 及其相关技术的飞速发展,Web 正在逐渐成为全球的自主分布式计算环境. 然而,由于 Web 上的数据绝大多数是通过 HTML 语言来显示的,而 HTML 语言的特点是任何组织或个人可以很随意地在 Web 上发布内容多样、形式各异的信息,导致 Web 上的数据处于杂乱无序的状态,数据集成性非常差[1]. 面对内容庞杂,动态变化的 Web 信息资源,人们很可能身陷信息的海洋而无所适从.

因此,网页数据抽取越来越受到重视. 但网络数据的内容和表现形式动态多变的特性往往增加了网页数据抽取的难度. 目前,网页抽取多采用 Wrapper/Mediator 方法. 该方法会对某个网页数据源产生单独的 Wrapper,若网页结构进行微小改变后,往往需要重新构建一个新的 Wrapper,这样会加大 Wrapper 的产生成本. 本文着重讨论基于规则树的网页数据抽取 Wrapper 的方法,当网页结构发生小范围变化后,Wrapper 自动识别这些变化并自动修改规则,使其仍可以继续工作,从而提高 Wrapper 的自适应性.

收稿日期: 2010 - 07 - 12

通讯作者简介: 邓莎莎(1979 -),女,硕士,讲师,湖南长沙人. 主要研究方向为文本挖掘,数据挖掘,决策支持系统. E-mail: Dengshasha1108@gmail.com.

1 Wrapper 维护的研究现状

关于 Web 数据的抽取问题,比较流行的方法是 Wrapper/Mediator 的方法. 该方法并不是将各种数据源的数据集中存放,而是通过 Wrapper/Mediator 这一体系结构来满足上层对数据的需求. 其核心是通过中介模式将各个数据源的数据集成起来,而数据仍然存储在局部的数据源中. 它是通过 Wrapper 对数据源的数据进行转换使之符合中介模式,这样就能很好地解决数据仓库方法中存在的数据更新问题. 但由于各个数据源的 Wrapper 需要分别建立,因此 Web 数据源的 Wrapper 维护成为又一难点.

KUSHMERICK [2] 提出 Wrapper 维护中的一个子问题—— Wrapper 验证,根据已知的正确结果来分析抽取结果并判断其正确性,从而达到验证的目的. 文献 [3] 通过回归测试及一个给定的阈值来检测页面的变化,一旦发现页面变化则通知设计人员,再由设计人员从新的格式中重新学习获得新的 Wrapper.

KNOBLOCK 等人 [4] 对页面的微小 HTML 标识变化给出一种 Wrapper 修复的方法. 首先提出了 Wrapper 的生命周期概念,以及如何确保正确可靠地抽取数据的方法. 通过机器学习得到所要抽取字段的数据模式的统计分布. Wrapper 就可以通过比较返回数据模式和统计分布的模式来验证. 当发现有显著不同时,系统就会发出通知或者自动调用修复程序.

CHIDLOVSKII [5] 提出了基于上下文的自动修复 Wrapper 方法. 在修复过程中,采用了一个分类机制,将语法特征和内容特征作为分类的标准,对多页面实行多种分类和多遍扫描,最终得出结论. 该方法是建立在微小变化的假设前提下,因此比前面几种方法有优势.

2 研究假设

由于网页数据是复杂多变的,因此本研究基于以下 3 点假设:

(1) 当用户从 Web 上收集数据时,应很清楚自身的需求,因此无需对 HTML 文档中全部数据进行抽取,只抽取对用户有用的数据;

(2) Wrapper 生成的最终目的是将源数据转换成某些易于处理的结构,并不是理解源数据的语义;

(3) 当 Web 页面发生变化后,认为仍保留了一些原有数据项特征,例如元数据、数据类型、抽取模式.

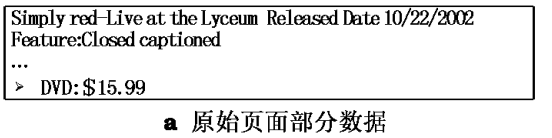
3 基于规则树的 Wrapper 维护结构设计

本文讨论的 Wrapper 维护是基于已构建的基于规则树的 Web 数据抽取方法,因此有必要简单介绍该方法是如何构建 Wrapper 的. 首先,用户使用 DTD 或者 XML Schema 定义一个 HTML 文档的抽取模式; 接着,用户在交互界面中将 HTML 页面上的数据例子和模式中的元素关联起来,建立映射规则; 最后,依据用户给出的映射规则生成规则树并生成 Wrapper.

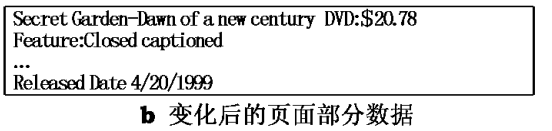
Wrapper 建立后存在一个问题,就是当 Web 数据的页面发生改变时,Wrapper 就会失效. 由于 Wrapper 与页面格式相关,所以当 Web 站点页面格式发生变化时,生成的 Wrapper 就会失效,也就是说无法从数据源中获取数据或者得到错误数据. 实际上,Web 页面变化是经常出现的,这也就提出了一个新问题—— Wrapper 的维护,即 Wrapper 失效时,如何修复失效的 Wrapper 使之继续正确抽取网页数据.

3.1 Wrapper 维护的总体流程

根据研究假设,图 1 表示的是 Web 页面出现结构变化的一个实例. 比较图 1a 和图 1b 可以发现,它们的抽取模式并没有改变,即页面的数据项特征没有发生变化. 如果页面的数据项发生改变,则认为需要重新生成 Wrapper.



a 原始页面部分数据



b 变化后的页面部分数据

图 1 Web 页面出现的结构变化

Wrapper 维护主要涉及两个方面的问题: 变化的数据项如何识别和抽取实例如何获取. 本研究提出一种基于元数据的 Wrapper 维护方法来解

决上述问题. 研究发现,在许多情况下,变化后的页面仍然保留了原来的数据特征,这些数据特征可以从用户定义的模式和抽取规则中获取,它们可以辅助自动进行 Wrapper 的维护.

图 2 为 Wrapper 维护模型结构,主要分为 3 个步骤:

(1) 识别数据特征　数据特征从给定的模式树和抽取规则中计算得出;

(2) 定义语义块　根据用户给定的模式树,在 HTML 树形结构上划分若干个子树,每一个子树或者一些兄弟子树的集合为一个语法块;

(3) 修复规则树　在同一个或者附近相关的语义块中搜索改变的数据项在规则树结构中的适当位置.

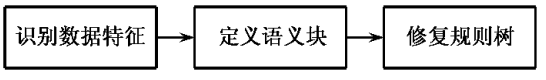


图 2　Wrapper 维护模型结构

3.2　识别数据特征

表 1 是图 1 所示页面的数据特征. 表 1 中每一行都有一个唯一对应的 ID,它确定一个数据的数据特征. Schema Element 存储的是模式树中对应的元素. Path 记录的是对应的叶子结点所在新的 HTML 树中的路径值. 值得注意的是,每个模式元素对应 3 个重要的数据特征,即: 超级链接(Hyperlink);数据注释(Annotation);数据属性(ItemAttribute). 这 3 个数据特征表示为一个三元组(H,A,I):

表 1　图 1 所示页面的数据特征

ID	Schema Element	Path	H	A	I
1	Pic	*. tr1. td1. div1	F	Null	img
2	Name	*. tr1. td2. b1	F	Null	txt
3	Price	*. tr1. td3. b1	F	DVD	integer
4	Related Video	*. tr4. div1	T	Related Videos	txt

(1) H　是一个布尔型数据,取值为 F 或者 T,表示该数据项是否包含超级链接;

(2) A　表示在网页上对该数据注释的内容,用它可以识别网页中该数据项是否发生变化;

(3) I　定义数据项的属性,通过它可以知道该数据项的数据为何种类型,它也是判断网页是否发生更改的重要依据.

3.3　定义语义块

研究发现,尽管 Web 设计者经常调整页面格式,但相同主题的数据一般会放在一起. 从将 HTML 文档看作树形结构的角度来说,改变的数据是放在原来结构中的子树或者其相邻子树中. 这种结构也对应于用户定义的模式树结构. 基于以上研究,可以在新的 HTML 树中定义语义块.

定义 1　如果一棵树同时满足以下条件,则这棵树为原子语义块.

(1) 它是某棵树中的一棵子树或者是一些兄弟子树的集合.

(2) 它包含的数据与对应模式树的定义完全匹配.

定义 2　语义块是一个或者多个原子语义块的集合.

这里的完全匹配指的是语义块 A 中 HTML 树中叶子结点所包含的、数据满足模式树中对每一个数据项的限制,如不能为空等. 从定义可以直观得出,一个原子语义块就是一个最小的可能进行 Web 数据抽取的单元.

在定义新的 HTML 树语义块时,利用非递归的方法后序遍历 HTML 树. 当访问到 HTML 树叶子结点时,会核对该叶子结点中的数据是否与模式树中的某个元素匹配. 若该结点重复满足了模式树中具有只能出现一次的元素,则说明该结点是另一个原子语义块的结点,并做上标记. 当遍历完成后,定义新的 HTML 树语义块的工作也就完成了.

完成定义语义块后,得到的是新 HTML 树的语义块的集合. 每一个语义块记录的是数据特征表中 ID 的集合.

3.4　修复规则树

修复规则,主要是依据语义块对规则树型结构进行修改.

(1) 定义规则树的语义块,获得规则树的语义块集合.

(2) 将新 HTML 树的语义块和规则树语义块进行匹配. 若完全匹配则说明这个语义块中的 HTML 树并没有改动,如果是部分匹配或者是完全不匹配,则说明 HTML 树有改动.

(3) 对于那些部分匹配的情况,则表示语义块中对应的 HTML 树结构已发生变化,根据 HTML 树语义块记录的数据特征和规则树语义块的数据特征修改规则树中的结点. 而对于完全不匹配的情况则返回给用户,重新生成新的 Wrapper.

4 实验结果及分析

为了验证基于规则树的网页数据抽取方法的性能和效果,选择某企业局域网页上的数据和图 1 所示页面作为试验数据,并邀请该信息规划科工作人员参与测试过程.

测试环境为: 服务端包括 HP PC Server LH6000(80 G 硬盘,512 M 内存,800 MHz PIII Xero CPU); WindowsXP 中文版 SP3,Oracle9i 中文版. 客户端包括普通 PC 机; Windows 操作系统,IE 浏览器.

4.1 实验方法

首先,需要指定某一个网页,对指定页面生成基于规则树 Wrapper. 在测试易用性指标中,所取页面为企业内部局域网中某一个网页的生成数据. 在测试自适应性指标时,改变的页面网址和改变前一致. 网址的改变认为是不属于网页维护范畴内. 同样,在测试效率指标时,先通过和用户交互的方式对页面生成树结构. 抽取的时间指的是生成树结构后到数据导入数据库的时间,实质上为遍历树结构的时间. 此外,为了获得更平均的统计结果,规定在相同的网络环境中,每个指标入参的样本集大小为 50. 因此,每个统计指标的最后结果是 50 个相同类型不同页面数据的平均值. 它们符合置信系数为 95%、置信区间不超过 5% 的要求.

实验中,将基于规则树的网页数据抽取的方法(RuleTree)同其他相关方法(W4F[6],Chidlovskii[5],SG-WRAM[1])进行比较.

4.2 试验指标

测试指标主要有以下 3 个.

(1) 易用性 EU(α) 主要是指用户在使用该工具时感到容易和方便的程度. 在测试中使用 *EU*(α) 指数表示该性能. 入参 α_1,α_2,α_3 分别对应仅熟悉需求的用户、不熟悉需求但接受培训的用户、熟悉需求并且接受培训的用户. EU(α) 值为用户在使用该工具将同一页面转换为最终数据的正确率.

(2) 自适应性 DY(η) 指作为数据源 Web 页面发生局部改变后,工具仍能正确抽取数据的能力. DY(η) 入参 η_1 表示在生成的 HTML 树中,发生改变的数据项相对与原来的位置处于同一棵子树中; η_2 则表示发生变化的数据项相对于原来位置不在同一棵子树中,甚至在生成的新子树中.

(3) 效率 Val(δ) 主要表示数据的抽取效率. 入参 δ 代表所抽取数据的大小.

$$\mathrm{Val}(\delta) = \frac{\text{抽取数据的时间}}{\text{所抽取数据的大小}} \quad (1)$$

4.3 试验结果

4.3.1 EU(α) 数据结果及分析

表 2 记录了 α 取不同值(α_1,α_2,α_3) 时 EU(α) 的结果.

表 2 EU(α) 试验结果

EU(α)	α_1	α_2	α_3
完全正确	0.24	0	0.98
基本正确	0.53	0.38	0.02
不正确	0.23	0.62	0

本试验中,用户水平相当,我们只对 α_2 和 α_3 同时进行了简单培训. 从表 2 可以看出,EU(α_3) 为完全正确的比率机会达到 100%,这表明对于需求熟悉并进行简短培训后的用户完全可以方便灵活地使用该工具进行数据抽取. 一个令人惊喜的结果是 EU(α_1) 的正确率远远大于 EU(α_2) 的正确率. 这说明工具的使用方法通俗易懂,即使没有经过培训的用户,如果十分了解需求仍然有可能正确地抽取数据. 当然,这其中也包含两个权变因素,即用户的素质及需求的复杂程度.

W4F 和 SG-WRAM 对用户都有特殊的要求. W4F 需要用户会使用 HEL 语言,Chidlovskii 采用一种机器学习的方式,SG-WRAM 要求用户能理解并使用所定义的正则表达方式.

4.3.2 DY(η) 数据结果及分析

表 3 记录了 η 分别取 η_1 和 η_2 时 DY(η) 的结果. 在自适应性的测试中,η_1 代表的是相对于以前页面,变化的页面中数据项只是在临近位置移动,即 HTML 树中同一棵子树中叶子结点的左右顺序发生变化. η_2 表示变化后的页面中数据项

发生较大的位置改变，即在 HTML 树中显示为某一棵子树的叶子结点变为相邻或不相邻子树的叶子结点.

表 3 DY(η) 试验结果

结 果	方 法	η_1	η_2
正 确	RuleTree	0.93	0.47
	W4F	–	–
	Chidlovskii	0.95	0.66
	SG-WRAM	0.90	0.50
不正确	RuleTree	0.07	0.53

从表 3 可以看出，4 种方法中 DY(η_1) 为正确的比率较高，相对而言，DY(η_2) 为正确的比率较低. RuleTree 方法通过定义语义块可以快捷准确地辨别变化的数据项位置，并正确抽取源数据，但其语义块对于跨子树特别是跨几棵子树的识别能力较弱; W4F 没有考虑网页变化后的自动维护问题; 由于 Chidlovskii 将网页首先进行分类，根据其语法和内容特征进行判断，准确率相对较高，但是耗时太多，结果也不稳定; SG-WRAM 与 RuleTree 两种方法的实验结果则比较相近.

4.3.3 Val(δ) 数据结果及分析

表 4 记录了 δ 在不同大小区间数值时 Val(δ) 的结果.

表 4 Val(δ) 试验结果

方 法	δ_1 (10 k ~ 100 k)	δ_2 (100 k ~ 1 M)	δ_3 (1 M ~ 10 M)
RuleTree	0.2 ~ 1.0	0.1 ~ 0.2	0.2 ~ 0.6
W4F	0.5 ~ 1.5	1.0 ~ 4.0	5.0 ~ 9.0
Chidlovskii	0.3 ~ 8.0	0.7 ~ 10.0	1.0 ~ 20.0
SG-WRAM	0.1 ~ 0.8	0.7 ~ 4.0	6.0 ~ 10.0

本试验中 δ 的取值是由源数据转换到中心数据库中数据的大小决定的. 以 RuleTree 方法为例将表 4 的数据作如下处理.

$$\frac{\delta_1}{\delta_2}=\frac{\delta_2}{\delta_3}=10 \tag{2}$$

$$\frac{\mathrm{Val}(\delta_1)}{\mathrm{Val}(\delta_2)}\approx 9 \tag{3}$$

$$\frac{\mathrm{Val}(\delta_2)}{\mathrm{Val}(\delta_3)}\approx 5 \tag{4}$$

由此可知，在 RuleTree 方法中，当需要处理的数据大小以 10 倍速度增长时，其抽取时间的增长率却低于 10，表明需要抽取的数据越多，抽取的效率会有所提高. 其原因在于只要通过遍历生成的规则树就可以将数据和数据模式对应起来，直接抽取数据即可. 同时，树型结构采用类似 B + 树的双链表存储结构，这样只需访问链接数据的链表就可以完成抽取工作.

W4F 是利用类似 SQL 的查询语言抽取数据的，因此当需要抽取的数据越多时，其速度就会越慢. Chidlovskii 是利用机器学习的方法，其抽取速度和学习能力相关，因此速度极不稳定. SG-WRAM 方式是通过临时解析定义的正则抽取规则，与 W4F 方法一样，在需要抽取的数据激增的情况下效率就会变低.

5 结 语

从试验结果来看，基于规则树的 Wrapper 维护在易用性、自适应性和高效率上表现良好. 它对于需求明确的普通用户，只需要在简单培训后基本上能独立进行网页页面的抽取. 特别是当页面数据发生较小改变时，不需要用户参与就能自动修改树型结构，正确地抽取数据. 而最后和用户交互生成的规则树可以正确表示用户需求. 当用户定义好一个树型结构后，抽取速度很快，特别是在一定的范围内，随着抽取数据的增加，抽取效率还会有所提高.

参考文献:

[1] MENG Xiao-feng, LU Hong-jun, WANG Hai-yan, *et al*. Data extraction from the web based on pre-defined schema [J]. Journal of Computer Science and Technology, 2002, 17(4): 377-388.

[2] KUSHMERICK N. Wrapper verification [J]. World Wide Web Journal, 2000, 3(2): 79-94.

[3] KUSHMERICK N. Regression testing for wrapper maintenance [C] //Proceeding of the AAAI, Heidelberg, Germany, 1999: 74-79.

[4] KNOBLOCK C, LEMAN K, MINTOA S, *et al*. Accurately and reliably extracting data from the web: a machine learning approach [J]. Data Engineering, 2000, 23(4): 33-41.

[5] CHIDLOVSKII B. Automatic repairing of web Wrapper [C] // Proceeding of the Third International Workshop on Web Information and Data Management, Atlanta, USA, 2001: 24-30.

[6] SAHUGUET A, AZAVANT F. Building light-weight Wrapper for legacy web data-source using W4F [C] //Proceeding of the Very Large Data Bases (VLDB), Edinburgh, Scotland, 1999: 738-741.

(编辑 胡小萍)