基于 SRI 的动态网页信息抽取方法*

朱跃林[1],戴昌林[2],高志强[2]

(1. 无锡中航恒信工程管理咨询有限公司,江苏 无锡 214001;
2. 东南大学 计算机科学与工程学院,南京 210096)

摘 要:提出了基于相似记录项归纳的动态网页信息抽取方法. 该方法采用编辑距离算法和树排列算法归纳产生记录项的包装器树. 对各种类型网页进行信息抽取实验,取得 98.11% 的召回率和 96.90% 的准确率.

关 键 词:动态网页; 信息抽取; 包装器; DOM 树
中图分类号:TP311 **文献标识码**:A **文章编号**:1671 -0924(2009)10 -0087 -07

Information Extraction Method for Dynamic Web Pages Based on Similar Records Induction

ZHU Yue-Lin[1] DAI Chang-Lin[2] GAO Zhi-Qiang[2]

(1. CAPDI (WuXi) Hengxin Engineering Consultants Co. Ltd., Wuxi 214001, China;
2. School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: Dynamic Web pages are pages which are generated by programs automatically. It is estimated that most Web pages exist in the form of dynamic web pages. This paper puts forward an extraction method based on similar records induction (SRI), which uses string editing distance algorithm and DOM tree alignment algorithm to generate record wrapper. Experimental results show that the extraction method gets a recall of 98.11% and a precision of 96.90% for all kinds of dynamic Web pages.

Key words: dynamic Web page; information extraction; wrapper; DOM tree

动态网页是用户发出请求时服务器端通过程序动态生成的页面,具有特定的模板,并嵌入了后台数据库中的数据. 据统计,Web 上的页面主要是以动态网页的形式存在,大约占 80% 以上[1]. 记录级动态网页是一种主要的动态网页,常出现在搜索引擎、网上购物商店、数字化图书馆和机构的人员列表网页等站点中. 记录级动态网页的信息抽取方法具有较大的研究价值和实用价值.

记录级动态网页的特点是网页中包含多个结构相似和模板相同的记录项. 针对记录级动态网页的这种特点,本文中提出了基于相似记录项归纳(similar records induction, SRI)的信息抽取方法. 在网页的 DOM 树中,首先通过编辑距离算法发现相似记录项,然后通过树排列算法归纳产生

* **收稿日期**:2009 -05 -16
基金项目:国家自然科学基金资助项目(60873153,60803061).
作者简介:朱跃林(1982—),男,无锡人,主要从事软件工程与算法研究.

记录项的包装器树,并用于信息抽取.对各种类型网页进行信息抽取实验,以验证召回率和准确率.

1 基于相似记录项归纳的信息抽取方法

在网页的DOM树中,记录级动态网页中存在多个结构相似的记录项都位于相同的父节点下.针对记录级动态网页的这种特点,本文中提出了基于SRI的信息抽取方法,该方法的流程如图1所示.

该方法的主要步骤:

1)构建网页的DOM树.清洗整理网页的HTML源代码,构建DOM树.

2)发现相似记录项.在网页的DOM树中,通过DOM子树的相似性判断,发现并分离相似记录项.

3)归纳相似记录项.对所有相似记录项的DOM树,通过树排列算法排列产生记录项的包装器树,并用来抽取信息.

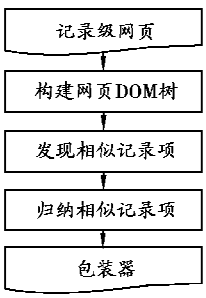

图1 基于SRI的信息抽取方法流程

1.1 网页DOM树的构建

网页的HTML代码包含大量不严格符合HTML规范的错误,例如HTML头标记对应的尾标记缺失、HTML标记之间没有相互对应等,而现在的浏览器都可以容忍这些错误并正常地显示,但是这些错误对于网页信息抽取往往是有害的,所以在构建网页的DOM树时,必须先将HTML代码规范化,这个任务可以由工具Tidy来完成.

另外,一些对信息抽取无用的HTML标记,需要被过滤掉,例如Script,Style,!等标记.有些HTML标记还会造成产生的抽取规则过于复杂,也需要被过滤掉,例如I,Strong,Strike,Cite,Span,Font,Em等.

1.2 记录项的发现

记录级动态网页的特点是:在网页的DOM树中存在的多个相似记录项都位于相同的父节点下,并且其对应的DOM树高度相似.在网页的DOM树中,如果可以定位到目标记录项所共有的父节点,然后在该节点所"挂起"的DOM子树中,通过判断DOM树的相似性就可以发现相似记录项.因此,相似记录项的发现可以被分为主要数据区定位和相似记录项分离2个子步骤.

1.2.1 主要数据区定位

定义主要数据区为包含所有目标记录项的最小DOM子树.如何定位包含所有目标记录项的最小DOM子树?本文中给出了3种启发式方法,分别从DOM树的不同特征考虑当前节点是否包含所有目标记录项.

1)最大扇出子树法.最大扇出子树法(highest fanout,HF)的思想是:一个节点包含的子节点(扇出)越多,它所"挂起"的DOM树越有可能是包含所有目标记录项的最小DOM子树.这种思想简单并且容易理解,但是会造成一定的错误.例如数字化图书馆表单中的<select>节点有多达上百个<option>子节点,按这种方法,就会将<select>节点所"挂起"的DOM树误判为主要数据区.所以可以预先定义一些常见的非法HTML标签,例如head,html,script,noscript,input,select,nobr,option,map,title,meta等,并在判断时过滤掉这些HTML标签所"挂起"的DOM子树.

2)最大内容增大量法.HF可能将一些包含大量超链接的导航栏节点误判为包含所有目标记录项的最小DOM子树,原因是没有考虑DOM子树的内容量,即在浏览器上可以看到的字符的个数.可以根据DOM子树的内容量来判断主要数据区,但是很明显祖先节点的内容量大小必定大于或等于后代节点的内容量大小.因此,最大内容增大量法(largest size increase,LSI)采用另外一种内容量的计算方法,它计算节点的总内容量,并减去节点的平均内容量(节点大小除以节点扇出值),作为节点的内容增大量.一个节点的内容增大量越大,它所"挂起"的DOM树越有可能是包含所有

目标记录项的最小 DOM 子树.

3) 最大标记量法. 最大标记量法(largest tag count,LTC)考虑子树中所包含的 HTML 标记量. 对于没有祖先关系的 2 个节点,节点包含的 HTML 标记越多,它排位就越靠前. 对于有祖先关系的 2 个节点,由于节点标记数量必然少于它的祖先节点,所以还需考察 2 个节点的直接子节点的 HTML 标记,并根据出现频次最高的 HTML 标记的频次,对这 2 个子节点进行排序.

上述 HF,LSI 和 LTC 3 种方法分别使用节点扇出、节点的内容增大量和节点的标记量,独立地判定包含所有目标记录项的最小 DOM 子树. 这种独立性使得可以将它们综合起来获得更高的准确率. 独立地使用 HF,LSI 和 LTC 3 种方法,可以返回 3 个候选的排序节点列表. 在综合判断时,可以分别给 HF,LSI 和 LTC 方法一个相应的权重,根据权重可以计算得到一个加权平均的排序节点列表.

排序节点列表中的第一个节点所"挂起"的 DOM 树作为包含所有目标记录项的最小 DOM 子树. 如果主要数据区不对应节点列表中的第一个节点,用户可以选择后续的其他节点所"挂起"的 DOM 树作为主要数据区. XWRAP[2] 采用了类似的方法,并提供相应的界面来辅助用户选择相应的节点作为主要数据区. 主要数据区可以用对应的 DOM 树根节点的 DOM 路径来表示.

1.2.2 记录项的分离

在主要数据区所对应的 DOM 树中包含多个相似的记录项. 这些记录项所对应的 DOM 子树,存在高度的相似. 可以根据 DOM 树的相似性计算,来发现和分离所有的相似记录项.

1) DOM 子树相似性判断的标准. 前序遍历 DOM 树,记录遍历到的每个节点的名称,这样可以将一个 DOM 树转换为 DOM 树中节点名字组成的字符序列,可以根据字符序列的相似性判断标准来判断 DOM 子树是否相似.

字符序列的相似性可以用编辑距离(edit distance, ED)算法[3] 进行判断. 编辑距离是指只用插入,删除和替换 3 种基本操作,把一个序列变换成另一个序列所需要的最少操作次数(或最小代价和)由于传统编辑距离没有考虑序列长度的影响,人们提出了归一化编辑距离的概念. 设 X 和 Y 分别表示 2 个序列,$P_{X,Y}$ 表示将 X 变换成 Y 的最小代价和的路径(可以看作是插入,删除和替换 3 种基本操作的序列),$M(P_{X,Y})$ 表示路径 $P_{X,Y}$ 的代价,那么归一化编辑距离可以被定义为

$$ED(X,Y) = \frac{M(P_{X,Y})}{\max(|X|,|Y|)}$$

其中:$|X|$ 和 $|Y|$ 分别表示序列 X 和 Y 的长度. 当定义插入代价 I,删除代价 D 和替换代价 C 都为 1 时,$ED(X,Y)$ 的取值为[0,1]. 编辑距离算法可以利用动态规划进行求解[3],分值矩阵 M 的计算公式为

$$M(i,j) = \begin{cases} 0, i = j = 0 \\ M(i-1,0) + D(X_i), j = 0, i > 0 \\ M(0,j-1) + I(Y_j), i = 0, j > 0 \\ M'(i,j), i > 0, j > 0 \end{cases}$$

$$M'(i,j) = \min(M(i-1,j) + D(X_i), M(i,j-1) + I(Y_j), M(i-1,j-1) + C(X_i,Y_j))$$

定义字符序列 X 和 Y 的相似度为

$$Similarity(X,Y) = \begin{cases} +\infty, X = Y \\ \dfrac{1}{ED(X,Y)}, X \neq Y \end{cases}$$

$ED(X,Y)$ 越小,$Similarity(X,Y)$ 越大. 当 X 和 Y 完全一样时,$ED(X,Y) = 0$,$Similarity(X,Y) = +\infty$;当 X 和 Y 完全不一样时,$ED(X,Y) = 1$,$Similarity(X,Y) = 1$. 给定相似度的阈值 α,当 2 个字符序列的 $Similarity(X,Y)$ 大于阈值 α 时,判定 2 个字符序列相似.

2) DOM 子树的相似性计算. 有了相似性判断的标准,可以根据此标准来计算判断 DOM 树是否相似. 但是还存在 2 个问题:①在主要数据区中,记录项是从第几个子节点为根节点的 DOM 树开始的? ②每个记录项包含几个子节点为根节点的 DOM 树?

对于第 1 个问题,由于主要数据区的下一层子节点个数有限,因此可以从第 1 个开始一直遍历到最后一个. 对于第 2 个问题,可以设置一个常数 K,表示每个记录项最多包含 K 个子节点的 DOM 树. 从 1 到 K 逐个设定记录项包含的 DOM

子树个数,并从主要数据区的第一个子节点开始,排列组合地计算所有候选记录项的相似性. 在比较子节点为根节点的 DOM 树的相似性过程中,先前的一些比较结果可以在后面的比较中用到.

下面给出一个具体的例子. 图 2 中 P 节点所"挂起"的 DOM 树代表主要数据区. 主要数据区有 10 个子节点,它们标号为 1 ~ 10. 这里假设每个记录项最多包含子节点 DOM 树的个数 K 为 3.

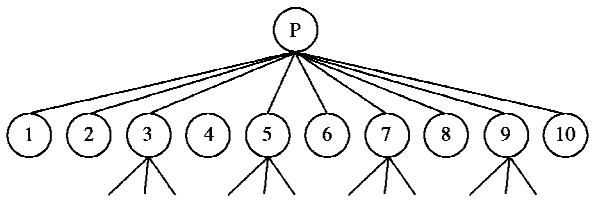

图 2 发现记录项示例

从节点 1 开始,比较如下:

(1,2), (2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,9), (9,10)

(1 -2,3 -4),(3 -4,5 -6),(5 -6,7 -8),(7 -8,9 -10)

(1 -2 -3,4 -5 -6),(4 -5 -6,7 -8 -9)

第 1 行是包含 1 个子节点为根节点的 DOM 树的记录项列表,其中(1,2)表示判断节点 1 为根节点的 DOM 树与节点 2 为根节点的 DOM 树是否相似. 从节点 2 开始,比较如下:

(2 -3, 4 -5), (4 -5, 6 -7), (6 -7, 8 -9)

(2 -3 -4, 5 -6 -7), (5 -6 -7, 8 -9 -10)

需要比较(2,3),(3,4)……这样的单个节点的组合,但是由于它们在前面已经比较过了,这里不需要重新比较. 从节点 3 开始,比较如下:

(3 -4 -5, 6 -7 -8)

通过以上比较,可以确定记录项是从第 2 个子节点为根节点的 DOM 树开始,每个记录项包含 2 个子节点为根节点的 DOM 树. 分离得到的所有相似记录项有(2 -3,4 -5,6 -7,8 -9).

1.3 记录项的归纳

当分离相似记录项后,可以为每个记录项构建相应的 DOM 树. 这样对于所有相似记录项的 DOM 树,可以通过树排列算法归纳产生记录项的包装器树.

1) 包装器树的描述. 包装器树与 DOM 树类似,但是它增加了节点的类型,以表示 JHJPCDATA、可选项和迭代项等情况. 包装器树的节点类型有:①单次节点,仅出现 1 次的节点;②可选节点(用? 表示),出现 0 次或 1 次的节点;③重复节点(用 + 表示),出现 1 次或多次的节点. ④JHJPCDATA 节点,出现 1 次的 TEXT 节点,表示该节点的字符串值不相等,来自于数据库.

2) 树排列算法. 该算法(Tree_Align)借鉴了 Roadrunner 在文献[4]中 ACME 算法的一些思想,并进行了改进. RoadRunner 的 ACME 算法是在网页的 HTML 源代码上迭代产生基于字符流的包装器,而树排列算法是在记录项的 DOM 树上迭代产生基于 DOM 树的包装器. 算法的输入是所有相似记录项的 DOM 树的集合,输出是一棵记录项的包装器树. 算法初始化包装器树为输入的 DOM 树集合中节点数最少的 DOM 树. 算法每次将集合中的一棵 DOM 树与包装器树进行匹配,并生成新的包装器树,然后用另一个 DOM 树和新的包装器树进行匹配. 这样进行迭代匹配,直到集合中所有的 DOM 树全部匹配完,并生成最终的记录项的包装器树.

包装器树和记录项 DOM 树的匹配是通过树匹配(tree_match)算法来实现的. 树匹配算法借鉴了 RoadRunner 的 ACME 算法中的思想,通过字符串的不匹配来发现 JHJPCDATA,通过 HTML 标记的不匹配来发现可选项或迭代项. 但是树匹配算法是在 DOM 树上逐层次进行匹配的,更好地利用了 DOM 树的树形结构. 在前序遍历下,树匹配算法逐个将记录项 DOM 树的当前节点和包装器树的当前节点进行匹配,如果相等,则继续匹配它们的子节点链表;如果不相等,再判断记录项 DOM 树的当前节点是否是可选项或迭代项. 在判断节点相等时,不能只根据节点名是否相等进行判断,还应该考虑节点所对应的 DOM 子树是否存在相似. 因为在很多情况下,节点名可以相等但节点的 DOM 树完全不一样.

下面通过一个例子来描述 Tree_Match 算法. 如图 3 所示,将包装器树初始化为节点数最少的 DOM 树(如图 3 a)),用另一棵相似记录项的 DOM 树(如图 3 b))与包装器树进行匹配,匹配过程如