第 25 卷 第 3 期 福建师范大学学报（自然科学版） Vol. 25 No. 3
2009 年 5 月 Journal of Fujian Normal University (Natural Science Edition) May 2009

文章编号: 1000-5277(2009)03-0039-08

基于树形结构的Web信息抽取

任仲晟[1]，薛永生[2]
(1. 福建师范大学数学与计算机科学学院，福建 福州 350108;
2. 厦门大学计算机科学系，福建 厦门 361005)

摘要: 提出了一种基于树形结构的Web结构化数据抽取算法 该算法基于HTML的树形层次结构，包括HTML树构造算法，数据区域挖掘算法，数据记录挖掘算法以及数据记录模式生成算法 算法引入了页面元素布局位置等信息用于清洗页面，采用层次划分思想实现页面数据区域的挖掘，并通过树匹配生成记录模式，实现最终数据项抽取 实验表明，该方法可以有效地实现Web结构化数据抽取

关键词: Web数据抽取; Web挖掘; 信息抽取
中图分类号: TP311.131 **文献标识码**: A

Web Information Extraction Based on Tree Structure

REN Zhong-sheng[1], XUE Yong-sheng[2]
(1. *School of Mathematics and Computer Science, Fujian Normal University, Fuzhou* 350108, *China*;
2. *Department of Computer Science, Xiamen University, Xiamen* 361005, *China*)

Abstract: It proposes tree structure based Web data extraction algorithm in view of the inadequacies of the existing methods. The tree structure based algorithm includes: the algorithm of HTML tree construction, the algorithm of data region mining, the algorithm of data record mining, and the algorithm of record schema generation. The algorithm cleans the Web pages using the position information of page elements, mines data region by hierarchical clustering, and generates record schema finishing data item extraction through tree matching. Experimental results show that our algorithm can improve the accuracy and efficiency of Web data extraction.

Key words: Web data extraction; Web mining; information extraction

互联网上的数据大多数是用HTML语言描述的半结构化数据 半结构的数据缺乏语义信息，集成性差，使得Web上的海量数据不能被充分利用 同时，由于人类的审美观以及商业上的要求，HTML页面中还充斥着大量与主题无关的信息，诸如一些图片，广告信息等，影响了对有用信息的判别 如何避免“数据爆炸，知识匮乏”的尴尬，从海量的半结构化信息中抽取出结构化的符合主题的数据，推动了Web信息抽取技术的产生与发展

本文在研究有关Web信息抽取技术的基础上，提出了一种基于树形结构的Web信息抽取算法 算法利用HTML页面的固有层次结构，引入叶子节点相似度划分，路径分裂，树匹配等技术，能够有效地抽取出数据导向型网页（部分文献中也称作模版页面）中的数据，并转换成结构化的数据

收稿日期: 2008-11-25
基金项目: 国家自然科学基金资助项目（50474033）; 福建省自然科学基金资助项目（A0310008）; 福建省重点科技项目（2003H043）
作者简介: 任仲晟（1981— ），男，福建福州人，助教，硕士，研究方向: 数据挖掘等

1 基于树形结构的Web信息抽取

1.1 抽取流程介绍

本文设计的系统由以下4个部分构成:

(1) HTML树生成模块: 该模块负责清理简化HTML源文件, 生成页面对应的HTML树形结构

(2) 候选区域生成模块: 根据HTML树, 计算信息节点的相似度, 筛选出含有待抽取信息的数据区域

(3) 数据记录生成模块: 通过比较数据区域内部不同节点之间的路径, 对数据区域进行拆分, 提取出数据记录集合

(4) 记录模式生成模块: 根据数据记录对应的HTML树, 进行树匹配调整操作, 获得待生成的数据记录模式信息. 然后根据数据记录模式信息, 完成Web数据抽取, 输出结构化数据

1.2 HTML页面的预处理

页面预处理的目的有两个, 一是修正HTML源文件中的错误, 二是降低后续步骤的复杂度, 提高抽取的效率和精度

本文设计的系统利用元素在界面上的相对位置来准确地标记元素之间的嵌套关系, 消除标签匹配等错误. 同时制定了元素删除规则, 对页面冗余信息进行简化[1].

1.3 HTML页面主数据区域的挖掘

1.3.1 有关概念

HTML页面中含有大量广告, 导航条, 版权信息等主题无关的数据. 面对令人眼花缭乱的页面, 不少学者对如何挖掘页面中主题相关的, 包含真正信息的区域, 展开了大量的研究

文献[2]将Web页面分割成多个Web页面块, 提出了主要内容区的概念. 进而作者给出了4个算法用于识别那些主要内容区. 这些算法的基本思想是根据标签集合, 分别对多个页面进行分割, 得到若干个块, 然后计算块的BDF值, 统计块间相似度, 最后根据相似度和块特征抽取出主要内容区

文献[3]提出了DSE算法, 利用页面间的跳转信息, 获取相似页面; 然后将这些页面转换成标签树, 通过对比得到它们之间的相同部分, 删除这些相同部分后, 剩下的就是真正的数据区域

文献[4-5]提出了样式树的概念, 根据信息块熵的信息, 用以确定页面中的非主要内容.

与已有的这些方法不同, 本文提出的主数据区域挖掘算法, 充分利用了HTML树形结构的信息. 这里的主数据区域, 即是页面中包含真正数据的区域, 如图1中的Region3(图中页面来自www.joyo.com).

在给出主数据区域挖掘算法之前, 首先给出几个本文涉及到的概念的形式化定义

定义1 在HTML树中, 定义如下关系:

(1) 称两个节点n_1, n_2父亲关系Parent(n_1, n_2)存在, 如果n_1为n_2的父亲节点

(2) 称两个节点n_1, n_2祖先关系Ancestor(n_1, n_2)存在, 如果Parent(n_1, n_2) $\vee \exists n_3$(Parent(n_1, n_3) $\wedge$Ancestor(n_3, n_2)).

定义2 (节点祖先集合AncestorSet) 给定一个节点n_1, 它的祖先节点集合定义成AncestorSet(n_1) = {n} Ancestor(n, n_1).

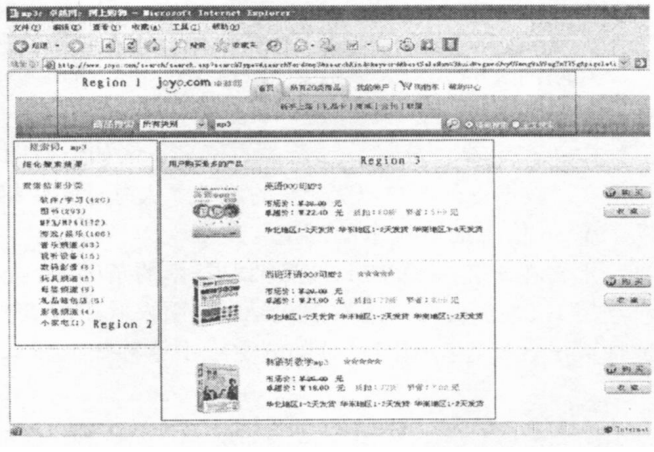

图1 HTML页面区域示意图

定义3 (叶子节点相似度) HTML树中, 任意给定两个叶子节点n_1, n_2, 规定其共有的祖先节点数为它们之间的相似度, 同时称这些共有祖先节点构成了这两个叶子节点之间的公共路径

在上述相似度的定义下, 可以求出所有叶子节点之间的相似度. 之所以对叶子节点感兴趣, 是因

为叶子节点包含了大量的数据信息

1.3.2　基于相似度的层次划分算法

仔细观察HTML 页面, 可以发现构成相关对象的页面元素在HTML 树中的位置是相近的, 同一对象 (如一条记录) 的标签位于HTML 树的某棵子树上, 这些叶子节点的相似度较为接近　同时, 由于笔者考虑的是多数据记录的数据导向型页面, 如果将这些记录节点包含的叶子节点的相似度展开, 整体作为一个趋势看待, 它将呈周期性的变化, 如图 2 所示　这种单个记录叶子节点相似度局部相近, 不同记录叶子节点相似度周期性变化的特点, 充分描述了主数据区域和其它信息区域间的差别　因此能够根据叶子节点的相似度, 对这些叶子节点进行划分, 从HTML 页面中选择出所关心的数据区域

任给两个叶子节点, 如果它们属于同一个划分, 则处于它们之间的节点的相似度必须大于某个阈值　为了选择最后的划分结果, 本文采用了文献 [6] 中给出的基于类元素分布的验证指标 Sep, Ocq 用于评估划分的效果

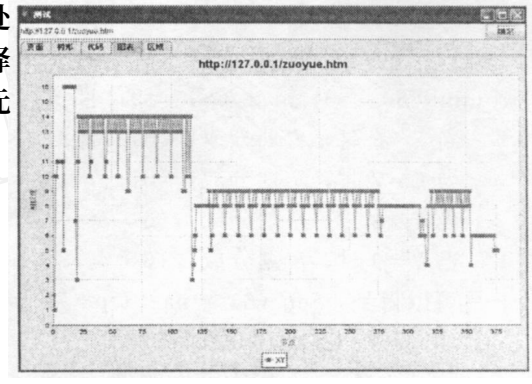

图 2　数据导向型页面相似度趋势

算法 1　数据区域挖掘算法DRM ining

输入: 叶子节点的相似度矩阵 sim ilarity

输出: 主数据区域叶子节点相似度矩阵DataRegion

```
Algorithm: DRM ining (similarity)
(1) min = min (similarity);
(2) max = max (similarity);
(3) Ocq = 2;
(4) for (j= min; j< = max; j+ + )
(5)   output= new Vector ();
(6)   mintemp= new Vector ();
(7)   maxtemp= new Vector ();
(8)   maxCluster= true;
(9)   minCluster= true;
(10)  for (i= 0; i< similarity.Length; i+ + )
(11)    tmp= similarity.get (i);
(12)    if (tmp> = j)
(13)      if ( (mintemp.Length! = 0) && (maxCluster))
(14)        output.add (mintemp);
(15)      if (maxCluster)
(16)        maxtemp = new Vector ();
(17)    maxtemp.add (tmp);
(18)    maxCluster = false;
(19)    minCluster = true;
(20)    else
(21)      if ( (maxtemp.Length! = 0) && (minCluster))
(22)        output.add (maxtemp);
(23)    if (minCluster)
(24)      mintemp = new Vector ();
(25)    mintemp.add (tmp);
(26)    minCluster = false;
(27)    maxCluster = true;
(28)  update output;
(29) calculate current curOcq;
(30) if (curOcq< Ocq)
(31)  Ocq= curOcq;
(32)  result= output;
```

(33) DataRegion= result's largest sub-cluster;

(34) return DataRegion;

算法首先取得相似度矩阵 similarity 中的最小值min 和最大值max，然后用 [min，max] 之间的值，逐一进行探测，选择具有最小Ocq 值的 h 作为划分依据. 对于每个 h 值，算法依次取出 similarity 中的值，与 h 值进行相比，若 similarity [i- 1] 比 h 小，similarity [i] 大于 h，则重新生成一个划分，同理，若 similarity [i- 1] 比 h 大，similarity [i] 小于 h，则生成新的划分. 算法最后在具有最小Ocq 值的划分中，选择出具有最大节点数的区域，作为输出. 该区域包含了需要提取数据的相关信息.

下面举例说明算法DRMining 的执行过程. 假设有以下相邻节点的相似度序列(为简化计算，例子中只考虑 Sep 值):

S= [6, 7, 8, 7, 6, 9, 7, 8, 7, 6, 8, 7, 6, 8, 7, 6, 9, 8],

则min (S) = 6，max (S) = 9，且

当 h= 6 时，S 划分成: {6, 7, 8, 7, 6, 9, 7, 8, 7, 6, 8, 7, 6, 8, 7, 6, 9, 8};

当 h= 7 时，S 划分成: {6} {7, 8, 7} {6} {9, 7, 8, 7} {6} {8, 7} {6} {8, 7} {6} {9, 8};

当 h= 8 时，S 划分成: {6, 7} {8} {7, 6} {9} {7, 8, 7, 6} {8} {7, 6} {8} {7, 6} {9} {8};

当 h= 9 时，S 划分成: {6, 7, 8, 7, 6} {9} {7, 8, 7, 6, 8, 7, 6, 8, 7, 6} {9} {8}.

于是可以得到: Sep (6) = na; Sep (7) = 0.694 075 827 761 916 8; Sep (8) = 0.4950336703414039; Sep (9) = 0.38543464816368556.

所以选择 h= 9 作为最后的划分，并选取子序列 {7, 8, 7, 6, 8, 7, 6, 8, 7, 6} 输出.

下面对这个结果进行人工评估. 首先观察人为构造的 S 序列 {6, 7, 8, 7, 6, 9, 7, 8, 7, 6, 8, 7, 6, 8, 7, 6, 9, 8}，不难发现，需要的数据区域对应于 {8, 7, 6, 8, 7, 6, 8, 7, 6} 这个子序列(如图3所示)，正好落在算法选取的子序列当中. 子序列当中的其它数据称其为噪声数据，将在下一步的处理中去除. 图4为图2所示页面划分后的相似度曲线图.

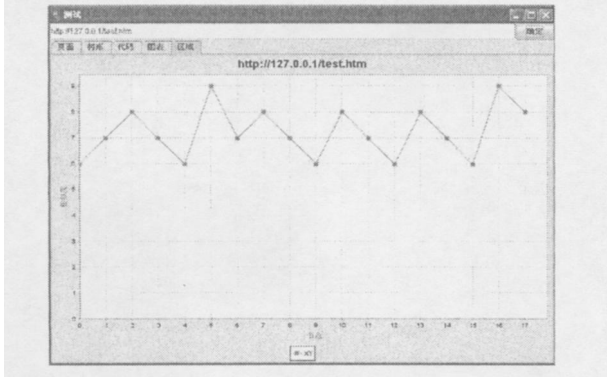

图3 S序列曲线图

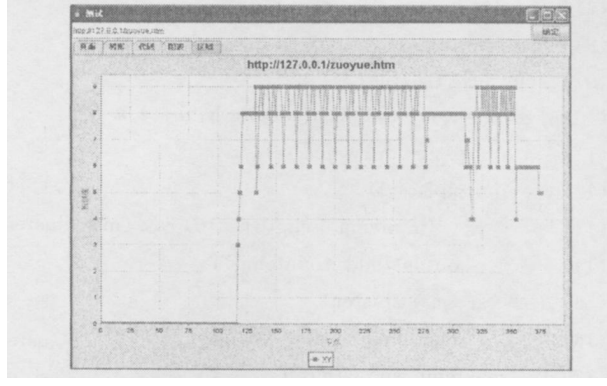

图4 划分后的相似度曲线图

1.4 HTML 页面数据记录的挖掘

1.4.1 记录节点定义

在得到主数据区域的基础上，算法RecordMining 对数据记录进行提取. 首先给出记录节点的定义.

定义4 (记录节点) 记录节点由HTML 页面树中的一个或者多个节点构成，并且满足如下条件:

(1) 每个记录各自构成HTML 页面上的一个区域，由一对标签< tag> < /tag> 标记.

(2) 构成记录节点的HTML 树节点，有共同的父亲节点，或者是同一代兄弟节点.

(3) 构成记录节点的HTML 树节点是相邻的.

直观上看，一个记录节点对应于一条待提取的记录，多个记录节点构成了主数据区域.

1.4.2 数据记录挖掘算法

数据记录挖掘所做的工作，就是利用已知的数据区域信息，对其进行划分. 从HTML 树的角度看，等价于对数据区域这棵子树进行分割，找出每个记录点所对应的那棵子树.

算法2 RecordMining

输入: 相似度矩阵DataRegion, 路径矩阵 leafPath
输出: 记录集合 RecordSet

```
Algorithm: RecordMining (DataRegion, leafPath)
(1) m = first node index in DataRegion;
(2) M = last node index in DataRegion;
(3) T = last node in leafPath [m], leafPath [m+1], …leafPath [M] prefix;
(4) flag= true;
(5) t= T;
(6) while (flag)
(7)    childNode= t's child node of first level;
(8)    count= childNode.size;
(9)    max= 0;
(10)   for (i= 0; i< count; i++)
(11)      num = the number of child node of childNode [i];
(12)      if num > max
(13)         idx= i;
(14)         max= num;
(15)   tmp= t.childNode [idx];
(16)   if the number of tmp'nodes / the number of t'nodes > factor
(17)      t= tmp;
(18)   else flag= false;
(19) RecordSet= t's child node of first level;
(20) tag= tag name appears most frequently in RecordSet;
(21) avgsize= the number of t's child nodes / RecordSet.Length;
(22) for each element in RecordSet
(23)   if (element's tagname != tag)
(24)      ‖ (the number of element's child nodes< avgsize/2)
(25)         delete element from RecordSet;
(26) return RecordSet;
```

算法从主数据区域的叶子节点开始, 首先向上寻找它们的最近公共祖先 (算法 1- 3 行), 然后对这个祖先节点进行分割 (算法 4- 18 行). 对公共祖先进行分割后, 得到了以最近公共祖先为根的子树产生的森林, 森林中节点数最大的子树将包含要分离的数据记录区域. 同时, 为了减少后续数据项的匹配时间, 这个过程也将删除一些冗余标签, 直到需要提取的数据记录的最近公共祖先, 递归过程停止. 根据前面关于记录节点的定义, 树中必然存在某个节点, 它的子孙节点的总数占主数据区域的节点总数的比值大于一个阈值, 并且对这个节点进行分割后, 会产生大量 (取决于多记录数据导向型页面中数据记录的个数) 规模大致相同的子树, 这些子树可以看成是需要提取的候选记录. 参数 factor 用于控制这个递归过程 (算法 16- 18 行). 在得到候选记录的基础上, 对这些候选记录进行筛选 (算法 19- 25 行). 筛选的标准有两个, 首先是根据当前候选记录的根结点, 也就是HTML 文件中的开始标签来判别, 因为根据记录节点的定义, 页面中的记录会包含在一对相同的标签中; 另外一个筛选标准是, 这些子树的大小不能偏离它们的均值太多, 因为考虑到属性的缺失等因素, 不会明显地影响到记录的大小, 反应到子树上, 就是这些子树的规模大小偏差不能太大. 算法中采用的偏差控制值是子树平均大小的二分之一. 这两个筛选标准能够有效地过滤混杂在多条记录中间的噪声数据. 最后算法将返回过滤后的记录节点的集合, 等待下一步生成记录模式以及数据项的抽取.

1.5 数据项的抽取与结构化数据的生成

由于本文讨论的数据导向型网页是通过模版加后台数据库数据的方式生成的, 因此不同记录之间的区别, 以不同数据项的值的不同为主, 同时允许有少量的数据项数目的差别. 这些特征反映在这些记录所对应的HTML 子树上, 就是这些子树的底层叶子节点 (存储了具体的文本信息等值) 的不同,

或者个别标签节点的缺失

本文提出的数据记录模式生成算法建立在树匹配[7]之上 算法采取了“贪心”的策略,依次从所有待匹配树之间选取总节点数较多的树进行匹配,因为节点数越多的节点,含有的信息多,也就越有可能提前生成最后我们要得到的模式树,从而加快了匹配的过程

算法3 数据记录模式生成算法RecordSchemaGen

输入: 数据记录对应的HTML子树的集合S

输出: 数据库表模式schema,数据表table

```
Algorithm RecordSchemaGen (S)
(1) get the tree with the maximum number of nodes as the seed tree;
(2) count= 0;
(3) while (S is not empty) and (count< maxCount)
(4)    t= get one tree from S in sequence;
(5)    match (seed, t);
(6)    if (all items of t are matched with corresponding items of seed)
(7)       label corresponding nodes;
(8)       delete t from S;
(9)    else
(10)      temporarily delete t;
(11)   if (S is empty)
(12)      if (S has nodes temporarily deleted)
(13)         add nodes to S;
(14)         count+ + ;
(15) generate record schema;
(16) generate table according to schema;
(17) for each tree in S
(18)    for each item in current tree
(19)       if (item in schema)
(20)          insert item into table;
(21)       else
(22)          add column to table;
(23)          insert item into column;
(24)          update schema;
(25) return table and schema;
```

算法首先对输入的数据记录树,按照规模大小进行排序,然后选取规模最大的子树作为种子树(算法第1行). 接着依次读入其余子树,调用树匹配算法进行匹配 当所有的子树都已经匹配完成,代表所有记录的schema树已经生成后,则退出循环过程,开始输出数据(算法3- 25行). 同时匹配的先后顺序会导致某些原本可以匹配的子树,在这一轮的匹配中未能匹配,因此对于这些未能完全匹配的子树,采取“等待- 再次尝试”的策略,即先将该无法完全匹配的子树标记,等输入的子树集合全部处理过一遍后,再重新开始匹配上一轮未能匹配的子树,期待它们在新的匹配过程中实现完全匹配(算法10- 14行),因子maxCount用于控制多次尝试的总次数 算法的最后,先生成初始记录模式,对于那些不能完全匹配的数据项,则在数据抽取过程中,逐一添加到模式信息里面(算法15- 24行). 如果不考虑树匹配的时间,算法的复杂度为O (|S|).

算法第5行中的match函数用于执行两棵树的比较操作,其基本思想为前面提到的树匹配算法 在给非叶子节点进行匹配的过程当中,对于多个可能的匹配,要解决如何选择的问题 鉴于本文的遍历操作采用了深度优先的方式,因此给出如下非根中间节点匹配选择规则

(1) 如果当前节点为其父亲节点的第一个儿子节点,则该节点的匹配标号必须大于其父亲节点在种子树中的匹配标号

(2) 如果当前节点不是其父亲节点的第一个儿子节点, 则该节点的匹配标号必须大于其第一个非叶子左兄弟节点标号.

下面的HTML 代码片段给出了一个需要考虑匹配选择问题的极端情况 (为方便说明, 往HTML 源文件中加入了行号).

```
1:  < table>
2:    < tr>
3:      < td> < table>
4:        < tr>
5:          < td> 1< td>
6:          < td> 1< td>
7:        < /tr>
8:        < tr>
9:          < td> 1< td>
10:           < td> 1< td>
11:       < /tr>
12:     < /table> < /td>
13:   < /tr>
14:   < tr>
15:     < td> < table>
16:       < tr>
17:         < td> 2< /td>
18:         < td> 2< td>
19:       < /tr>
20:       < tr>
21:         < td> 2< td>
22:         < td> 2< td>
23:       < /tr>
24:     < /table> < /td>
25:   < /tr
26: < /table>
```

上面的HTML 代码片段是一个含有两条记录 (1, 1, 1, 1) 和 (2, 2, 2, 2) 的表格. 这两条记录只是在叶子节点的值上发生了变化, 在结构上存在极度的相似性, 因此在树匹配过程中, 存在多种可能的最大匹配结果. 根据上面定义的选取规则, 可以正确地抽取出这两条记录.

2　实验设计与性能分析

为了验证本文提出的算法的有效性, 笔者选择了几个网站页面进行数据的抽取测试. 测试环境中的硬件平台为联想启天M 4600, 软件平台为Windows 2003 Sever 操作系统, 算法用JBuilder2006 企业版实现. 评价指标采用文献 [8] 中提出的召回率Recall 和查准率Precision. 实验结果如表 1 所示.

表 1　有效性实验结果

序号	记录数	TP	FN	FP	Recall	Precision
1	20	20	0	0	100%	100%
2	24	NA	NA	NA	NA	NA
3	16	NA	NA	NA	NA	NA
4	10	10	0	0	100%	100%
5	30	24	6	0	80%	100%
6	16	16	0	0	100%	100%
7	15	NA	NA	NA	NA	NA
8	10	10	0	0	100%	100%
9	40	40	0	0	100%	100%
10	13	13	0	0	100%	100%

表 1 显示了系统对多记录数据导向型页面的抽取结果. 抽取页面分别来自文献 [1] 中列出的 10 个网站. 从Recall 和 Precision 两项指标可以看出, 本文提出的抽取算法是有效的. 这里需要对表中的NA 做个说明. 在实验测试中, 发现序号为 2, 3, 7 的网站页面中的数据记录是嵌套的. 以序号为 3 的网站为例, 页面中含有 16 条记录, 这 16 条记录分成 4 行 4 列进行显示, 页面以行作为单位进行标识, 因此本文提取出的抽取算法识别出 4 条记录, 分别对应这 4 行, 识别出的每条记录包含了 4 个列的全部内容, 即包含了原始的 4 条记录. 换言之, 抽取的结果, 对原始记录进行了组合, 每 4 条记录合并成一条记录. 这种情况的产生, 是由于这些页面的设置与本文算法中的路径分裂算法发生了冲突. 路径分裂算法一旦发现了多个匹配的结构后就停止了, 没有进一步地对单个结果进行划分. 如何处理这种

情况，是下一步要研究的工作之一. 不过在实验中发现，大多数网站页面的设计是符合路径分裂算法的.

3 结束语

针对目前大量存在的数据导向型页面，本文提出了一种新型的基于树形结构的Web信息抽取技术，充分利用页面结构信息，从计算叶子节点的相似度开始，识别数据区域，定位数据记录，直至提取数据项信息，生成模式，逐一缩小目标区域的大小. 最后通过实验表明，算法能够有效的处理数据导向型页面.

参考文献:

[1] 任仲晟. 一种新的HTML页面清洗压缩算法[J]. 福建电脑，2009，25(1)：60-61.

[2] Sandip Debnath，Prasenjit Mitra，Nirmal Pal，et al. Automatic identification of informative sections of web pages [J]. IEEE transactions on knowledge and data engineering，2005，17(9)：1233-1246.

[3] Jiying Wang，Fred H Lochovsky. Data-rich section extraction from HTML pages [C]//Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'02)，2002：313-322.

[4] Lan Yi，Bing Liu，Xiaoli Li. Eliminating noisy information in web pages for data mining [C]//Proc Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining，2003：296-305.

[5] Lan Yi，Bing Liu. Web Page Cleaning for web mining through feature weighting [C]//Proceedings of Eighteenth International Joint Conference on Artificial Intelligence. Acapulco，Mexico，2003：9-15.

[6] Ji He，Ah-Hwee Tan，Chew-Lim Tan，et al. On quantitative evaluation of clustering systems [J]. Information Retrieval And Clustering，2002：105-134.

[7] Wuu Yang. Identifying syntactic differences between two programs [J]. Software-practice and Experience，1991，21(7)：739-755.

[8] Raghavan V V，Wang G S，Bollmann P. A critical investigation of recall and precision as measures of retrieval system performance [J]. ACM Trans Information Systems，1989(3)：205-229.

(责任编辑：陈 静)