第 36 卷 第 12 期 计 算 机 工 程 2010 年 6 月
Vol.36 No.12 Computer Engineering June 2010
·软件技术与数据库· 文章编号：1000—3428(2010)12—0083—02 文献标识码：A 中图分类号：TP391

基于标签路径聚类的文本信息抽取算法

刘云峰
(山西工程职业技术学院网络电教中心，太原 030009)

摘 要：针对网页噪音和网页非结构化信息抽取复杂度高的问题，提出一种基于标签路径聚类的文本信息抽取算法。对网页噪音进行预处理，根据网页的文档对象模型树结构进行标签路径聚类，通过自动训练的阈值和网页分割算法快速判定网页的关键部分，根据数据块中的嵌套结构获取网页文本抽取模板。对不同类型网站的实验结果表明，该算法运行速度快、准确度高。
关键词：标签路径；网页分割；信息抽取；聚类；阈值

Text Information Extraction Algorithm Based on Tag Path Clustering

LIU Yun-feng
(Network & Audio-visual Center, Shanxi Engineering Polytechnic, Taiyuan 030009)

【Abstract】This paper proposes a text information extraction algorithm based on tag path clustering to solve the high complexity problem of Web noise and unstructured information extraction. The method adopts Web noise pretreatment, carries on the tag path clustering according to the Document Object Model(DOM) tree structure of Web. The key part of the Web is determined rapidly through automatic training threshold value and Web page division algorithm, and Web text extracted templates are obtained according to nesting structure in the data block. Experimental results on different kinds of Web sites show that the algorithm is fast and accurate.
【Key words】tag path; Web page segmentation; information extraction; clustering; threshold

1 概述

文献[1]提出一种依靠统计信息，从中文新闻类网页中抽取正文内容的方法，有一定实用性但适用范围有限。文献[2]针对 Deep Web 信息抽取设计了一种新的模板检测方法，并利用检测出的模板自动从实例网页中抽取数据，但只能用于电子商务网站。文献[3]从网页中删除无关部分，通过逐步消除噪音寻找源网页的结构和内容，但提取结果不完整。因此，本文利用专门的阈值来判定数据块结构。

2 基于标签路径聚类的文本信息抽取算法

2.1 网页预处理

可以通过以下 3 个预处理规则来过滤网页中的不可见噪声和部分可见噪声：(1)仅删除标签本身；(2)删除标签本身及其相应的起始与结束标签包含的 HTML 文本；(3)对 HTML 标签进行修正和配对，删除源码中的乱码。

2.2 区域噪音的处理

为了实现网页的导航，显示用户阅读的相关信息，并帮助用户实现快速跳转到其他页面，网页中一般要设计列表信息，把提供指向权威页面链接集合的一个或多个 Web 页面称为 HUB 页面，如图 1 所示。

图 1 典型的列表信息模块(HUB 页面)

在处理此类信息时，本文设计了 2 个噪音识别参数。

Length=Length(content)为<tag>…</tag>标签内纯文本信息的长度，设定字符的 ASCII code>255？length+2：length+1。

$$C_n=\frac{N_{\text{string}}}{N_{\text{link}}+N_{\text{string}}}\times\frac{NODE_{\text{nohref}}}{NODE_{\text{href}}+1}\times 100\% \tag{1}$$

其中，C_n 为列表噪音判定系数；N_{string} 是块中非链接字符的字数；N_{link} 是块中链接字符的字数；$NODE_{\text{href}}$ 是块中有 href 属性的节点数；$NODE_{\text{nohref}}$ 是块中没有 href 属性的节点数。

2.3 基于标签路径聚类的网页分割

网页分割算法基于启发式规则，算法分为 2 步：(1)Xpath 聚类；(2)对聚类的 Xpath 进行分割。本文约定 DOM 树的叶节点按照其在原始 HTML 文件中出现的先后顺序编号。

(1)Xpath 聚类。对具有最大相似度的叶节点进行聚类。节点取得最大相似度时 2 个节点 Xpath 完全相同。本文用向量 $\boldsymbol{X}_i=\{x_{i,1},x_{i,2},\cdots,x_{i,n}\}$ 表示第 i 个 Xpath 的聚类。其中，$x_{i,j}$ 表示第 i 个 Xpath 聚类中的第 j 个叶节点。

定义节点间距为 1 个 Xpath 聚类中 2 个节点编号之间的间隔。

$$\Delta Span_{i,j,k}=\left|x_{i,j}-x_{i,k}\right| \tag{2}$$

式(2)表示第 i 个 Xpath 聚类的第 j 个与第 k 个节点之间的编号间隔。

定义平均周期为一个 Xpath 聚类中相邻节点间距的均值。

$$\Delta T_i=\frac{1}{n-1}\sum_{j=1}^{n-1}\Delta Span_{i,j,j+1} \tag{3}$$

作者简介：刘云峰(1974－)，男，讲师、硕士，主研方向：数据库技术
收稿日期：2010-01-19 **E-mail：**sxdtlyf@163.com

定义间距方差为考察一个聚类中各个节点离散程度的量。

$$\sigma^2(\Delta T_i)_j = \frac{1}{n-1}\sum_{j=1}^{n-1}(\Delta Span_{i,j,j+1} - \Delta T_i)^2 \quad (4)$$

(2)分割点，将一个聚类中的不连续点称为分割点。为了反映分割点的具体位置定义了一个变量 θ，它是前后 2 个间隔之间的比值。

$$\theta = \frac{\Delta Span_{i,(j+2),(j+1)}}{\Delta Span_{i,(j+1),j}} = \frac{x_{i,(j+2)} - x_{i,(j+1)}}{x_{i,(j+1)} - x_{i,j}} \quad (5)$$

为了增强分割鲁棒性，为 θ 设定一个阈值范围。实验表明当 $\theta \in [0.85, 2]$ 时可以得到较好的分割效果。

算法采用如下启发式规则：

(1)如果 $\theta \notin [0.85, 2]$，则将向量 $\boldsymbol{X}_i$ 在分割点处分割开。

(2)如果一个向量的平均周期 $\Delta T > PreSpan$，且没有进行分割，节点数目大于预定义值，则认为已经到达网页内嵌块聚类的边界。

2.4 算法描述

2.4.1 Xpath 聚类算法

将一个目标页面表示为 DOM 树结构，采用深度优先遍历策略，提取 DOM 树中的每个叶节点。对于每次遍历的叶节点，通过比较其 Xpath，将其序号添加到具有最大相似度的 Xpath 聚类中。具体算法描述如下：

Input DOMTree
Output XpathCluster
Cluster(DOM Tree)
{ XpathCluster = ∅;
For each xpath of leaf node
{
if (XpathCluster.xpath.Find(xpath))
{XpathClusfer.xpath.Insert(node);}
Else
{XpathCluster.Insert(xpath);
XpathCluster.xpath.Insert(node);
}
}
Return XpathCluster;
}

由于在聚类过程中，可能将非正文信息聚类到正文信息类中，因此先分析其方差。若一个聚类中的方差很大，则利用式(5)定位到分割点，将目标正文信息块与其周围的分隔噪音块分割开。另外，利用文本信息块的聚类平均周期、信息长度和 HUB 判别等统计参数帮助定位分割信息条。当第 1 个满足全部启发式规则和统计信息的聚类出现时，可以认为已经找到了正文信息块，完成分割任务。

分割算法描述如下：

Input XpathCluster //Xapth 聚类
Output SegBoundary //分割边界
Variables: Integer: Length_Threshold; //正文长度的最小阈值
Float：C_n_ Threshold;// C_n 列表噪音判定系数的阈值
WebPageSeg
{ SegBoundary = ∅;
Count=0;
While(Count!=XpathCluster.size())
{
If(XpathCluster.at(count).var0 is within threshold)
{
If(xpathCluster.at(count).size()>MAXSIZE&&xpathCluster.at(count).length> Length_Threshold
&& xpathCluster.at(count). C_n > C_n_Threshold && ΔT > PreSpan) //check
{SegBoundary.insert(each node within XpathCluster. at(count))
Break;
}
Else Count++;
}
}Else{//利用启发式规则(1)进行分割
Detect segment point use(2.3.4)
Sort(new cluser);
Count++;
}
}
Return SegBoundary;
}

2.4.2 节点集合内的文本抽取算法

节点集合内的文本抽取算法描述如下：

Input SegBoundary[]；//分割出来的符合条件的文本块
Output TextHashMap<tagpath,table textchunk,document frequency> //基于 HashMap 的文本块模板映射
Variables Integer: Frequency_Threshold; //table/div 嵌套次数的阈值
StringBuffer: textChunk; //文本块
For each chunk p in SegBoundary[]
While p has more HTML nodes
nNode=p.nextnode;
If nNode is not table/div Tag
textChunk= textChunk+extracted text from nNode; //抽取 nNode 间的文本信息
else if nNode is table/div Tag
{
if TextHashMap.contains(tagpath)==true
{ documentfrequency++;}
Else{
Documentfrequency=1;
}
TextHashMap.put(tagpath,textChunk, documentfrequency);
}
While TextHashMap has more {tagpath,textChunk, document frequency}
h is TextHashMap's item
If document frequency of h≥Frequency_Threshold
Print textChunk of item h

2.5 阈值的确定

在上述算法中，需要设定 3 个阈值参数：Length_Threshold，C_n_Threshold，Frequency_Threshold，它们对算法的时间复杂度和抽取效果具有一定调节作用，处理网页结构相似的网页时，可以通过训练样本自适应地算出相应的阈值。

对不同类型网页的阈值，3 个参数的数据分布有较大不同，Length、C_n 的数据分布绝大多数处于较小范围内，这些数据也是需要去掉的噪音数据，因此，使用 K-means[4]对样本数据进行聚类处理，而 frequency 数据相对前 2 个参数没有明显的分布趋势，数据量不大，而且也处在{1-10}这样的一个较窄的局部区间中，实验表明，聚类分析效果不明显，因此，本文用算数平均值求解。

(下转第 87 页)