ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.12, December 2010, pp.3220−3236
doi: 10.3724/SP.J.1001.2010.03961

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种 Web 评论自动抽取方法*

刘 伟[1+], 严华梁[2], 肖建国[2], 曾建勋[1]

[1](中国科学技术信息研究所,北京 100038)

[2](北京大学 计算机科学技术研究所,北京 100871)

Solution for Automatic Web Review Extraction

LIU Wei[1+], YAN Hua-Liang[2], XIAO Jian-Guo[2], ZENG Jian-Xun[1]

[1](Institute of Scientific and Technical Information of China, Beijing 100038, China)

[2](Institute of Computer Science and Technology, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: liuw@istic.ac.cn

Liu W, Yan HL, Xiao JG, Zeng JX. Solution for automatic Web review extraction. *Journal of Software*, 2010,21(12):3220−3236. http://www.jos.org.cn/1000-9825/3961.htm

Abstract: Web user reviews are the important information source for many popular applications (e.g. monitoring and analysis of public opinion), and they need to be extracted accurately from Web pages. Web user reviews belong to user-generated contents, whose presentation is not restricted by the Web page template. Therefore new challenges are raised. First, the inconsistency of review contents on both DOM tree and visual appearance impair the similarity between review records; second, the review content in a review record corresponds to a complicated subtree rather than one single node in the DOM tree. To tackle these challenges, a comprehensive solution is proposed to perform automatic extraction of Web reviews by employing sophisticated techniques. The review records are extracted from Web pages based on the level-weighted tree similarity algorithm first, and then, the pure review contents in records are extracted by comparing the node consistency. The experimental results on news Web sites and forum Web sites indicate that our solution can achieve high extraction accuracy and efficiency.

Key words: Web user review; structured data record; Web data extraction

摘 要: Web用户评论是许多重要应用的信息来源,比如公众舆情的检测与分析,Web用户评论必须从网页中准确地抽取出来.用户生成内容(user-generated content)不受页面模板的限制,这就给Web数据抽取提出了新的挑战:首先,不同用户评论内容的不一致性严重影响了评论记录在DOM树和视觉上的相似性;其次,评论内容在DOM树中是一棵复杂的子树,而且彼此之间在DOM树中的结构相差巨大.为了解决这两个问题,提出了一种完整的解决方案,使用多种技术来实现对用户评论内容的抽取.抽取过程分为两个步骤,基于深度加权的树相似性算法评论记录首先从网页中抽取出来,然后通过比较DOM树中节点的一致性,将纯粹的用户评论内容从评论记录中抽取出来.在多个新闻

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2008AA01Z421 (国家高技术研究发展计划(863)); the China Postdoctoral Science Foundation Funded Project under Grant Nos.20080440256, 200902014 (中国博士后科学基金)

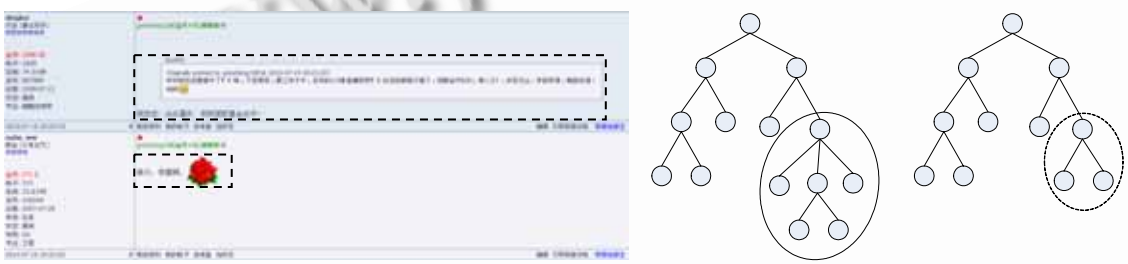
Received 2010-09-06; Revised 2010-11-08; Accepted 2010-11-24

网站和论坛网站上的实验结果表明,该方法可以达到较高的准确度和效率.

关键词: Web 用户评论;结构化数据记录;Web 数据抽取
中图法分类号: TP311 文献标识码: A

随着 Web 2.0 技术的快速发展,Web 用户可以自由地针对特定的事件或对象在网页中发表自己的观点看法.这使得 Web 用户评论以惊人的速度增长并且覆盖了现实世界的各个领域,比如经济、政治、娱乐等.Web 用户评论是许多应用的重要信息来源,比如公众舆情的监测与分析.这些应用需要一种准确高效的方法从大量不同的网站中收集舆情信息.虽然在这个任务上进行了一些努力尝试[1−3],但它们主要是关注于评论页面的爬取.由于评论页面中往往包含大量的噪音信息,严重影响了信息分析的性能.因而在实际应用中,需要将用户评论在页面中准确地抽取出来,而不是索引整个评论页面.

本文将研究从评论页面中自动抽取用户评论内容的问题,即 Web 用户评论的抽取.Web 用户评论抽取共分两个子任务:(1) 页面中评论记录的抽取;(2) 评论记录中评论内容的抽取.评论记录的抽取是指从评论页面中发现各个评论记录的边界,并将评论记录抽取出来作为下一个子任务的输入.评论内容抽取是指从评论记录中抽取纯粹的用户评论内容.图 1 展示了评论页面中的两个评论记录和它们在 DOM 树中,其中的用户评论内容用闭合的虚线分别在页面中和 DOM 树中标出.



(a) (b)

Fig.1 Two review records in one page and their subtrees in the DOM tree

图 1 同一页面中的两个评论记录以及它们在 DOM 树中对应的子树

显然,为大量不同模板的评论页面进行手工生成抽取程序是不现实的.为了满足实际应用的需要,我们提出了一种独立于模板的抽取方法,可以从不同模板的评论页面中将用户评论内容准确地抽取出来,而无需任何人工干预.众所周知,Web 评论是由不同的用户发布,而不是网站的拥有者.因此,评论内容之间通常在文本长度以及信息的格式上存在着非常大的不一致性.用户一般可以自由地使用各种字体进行修饰,而且图片、表格和视频等多种信息的格式也频繁出现在评论内容之中.Web 评论的这种特点使得抽取任务面临巨大的挑战.Web 用户评论的抽取属于 Web 数据抽取研究领域.据了解,在该研究领域已有大量的工作针对各种不同的应用和 Web 数据类型提出抽取方法,文献[4]对已有的该领域的研究工作进行了较好的综述.由于 Web 用户评论的不一致性,目前已有的方法不适合对 Web 用户评论的抽取.与其他类型的 Web 数据(比如购物网站中在搜索结果页面中列出的产品信息)相比,Web 用户评论的抽取带来了两个新的挑战:第一,由于不同评论记录之间的评论内容在信息格式上的随机性,严重影响了评论记录之间在 DOM 树和视觉上的相似性.这就使得现有的方法,比如文献[5,6]提出的方法,无法进行准确地抽取(见第 6.3 节的实验);第二,评论记录中的评论内容在 DOM 树中对应一棵结构复杂的子树不是一个叶节点,而且在 DOM 树中对应的子树结构各异.因此,从评论记录中抽取评论内容则转化成为在 DOM 树中如何识别包含评论内容的自小子树的问题,而不是以往对某个叶节点的抽取.针对这两个挑战,本文提出了一个完全自动化的抽取方案.该方案使用了多种先进的技术,包含评论记录抽取和评论内容抽取两个模块.在 Web 评论记录抽取的问题上,提出了一种基于深度加权的树相似性算法来计算子树之间的相

似度.利用该算法实现对噪音信息的去除和评论记录边界的识别.这样,每个评论记录对应 DOM 树中的一棵或多棵子树.在评论内容抽取的问题上,首先从每个评论记录中将包含评论内容的子树并进行对齐,然后通过计算对齐树中各个节点的一致性识别出包含评论内容的最小子树,从而达到对评论内容抽取的目的.根据该解决方案我们实现了 WeRE 原型系统,该系统从实际应用角度提供了两种抽取方式:直接抽取和 Wrapper 抽取.直接抽取不需要样本页面进行训练,不同模板的评论页面可以不加区分地一起处理.这种方式要求一个评论页面内必须包含至少 3 个以上的评论记录.Wrapper 抽取是针对一组相同模板的页面,首先挑选包含一个包含评论记录较多的页面作为样本页面训练生成 Wrapper,然后利用生成的 Wrapper 对其他页面进行抽取.该方式无需对样本页面进行标注,在效率上要高于前一种方式,但不能处理不同模板的评论页面.

总之,本文的贡献包括 3 个方面:首次指出了因评论内容的随机性而带来的 Web 数据抽取上的两个新挑战,已有相关工作并没有认识到该挑战,因此也没有提出相应的解决办法;第二,针对这些挑战,提出了一种轻量级的整体解决方案,在评论记录抽取以及评论内容抽取两个问题上分别提出完全自动的抽取方法.同时,提供了两种抽取方式来满足实际应用的需要;第三,在原型系统 WeRE 上进行的广泛实验表明,本文所提出的抽取方法能够达到非常高的准确性和效率,可以满足实际应用的需要.

本文第 1 节展示系统的整体框架及各模块的主要功能.第 2 节介绍系统中对评论页面的预处理以及一般评论页面具有的若干特征.第 3 节和第 4 节分别提出评论记录抽取和评论内容抽取的方法.第 5 节介绍系统提供的两种抽取方式.第 6 节报告对系统准确性和效率的评估.第 7 节是相关工作.第 8 节对本文工作进行总结.

1 WeRE 系统的整体框架

WeRE 系统的整体框架如图 2 所示.系统的输入是可在浏览器中正常显示的评论页面,输出是页面中用户提交的评论内容,以单列的表格形式展现,每行对应了一个评论记录中的评论内容.用户可以选择以直接抽取方式或 Wrapper 抽取方式进行抽取.

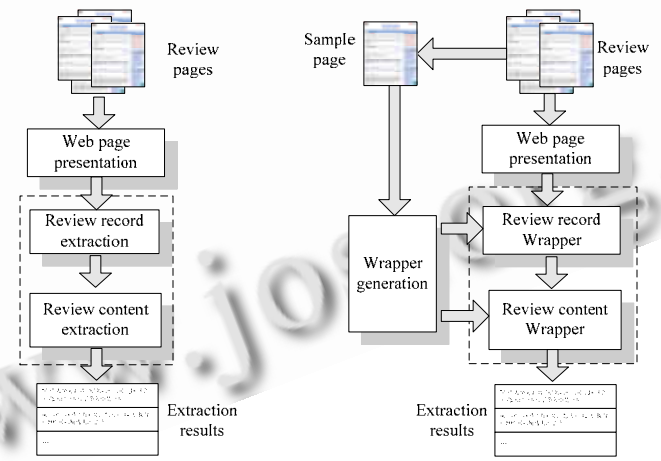


Fig.2 System architecture of WeRE

图 2 原型系统 WeRE 的整体框架

原型系统 WeRE 主要模块的功能介绍如下:

网页表示:将一个评论页面解析成 DOM 结构,同时把视觉信息附加在 DOM 树相应的节点上(比如每个节点在页面占据的矩形区域、使用的字体等);

评论记录抽取:首先从评论页面中将评论区域(DOM 树中包含所有评论记录的最小子树)识别出来,然后进一步从评论区域中将每个评论记录的边界识别出来;

Wrapper 生成:利用直接抽取方式将无标注的样本页面作为输入,分别生成评论记录 Wrapper 和评论内容

Wrapper;

评论记录 Wrapper:包含了页面中评论记录相关的标签路径,在页面中位置、字体等信息,用来从页面中抽取评论记录;

评论内容 Wrapper:包含了评论记录中评论内容相关的标签路径,在页面中位置、字体等信息,用来从评论记录中抽取评论内容.

2 网页解析和抽取特征

2.1 网页解析

我们将评论页面解析成 DOM 树的结构,并在 DOM 树上实现对用户评论内容的抽取.将网页转换成 DOM 树结构这种预处理方式已经被广泛地采用,相关的技术非常成熟,因此在本文中不再进行讨论.

DOM 树中的每个节点对应 Html 文件中一个元素.虽然网页转化成 DOM 树后具有了一定的结构特征,但 Html 标签缺乏语义表达的能力,因此我们把页面的视觉信息引入到 DOM 树中.页面的视觉信息被证明是用于 Web 数据抽取非常有效的特征,近来已经有若干基于页面视觉信息的抽取方法被提出,比如文献[6−8]中提出的方法.本文中所提出的方法用到了以下 3 类视觉信息:

面积:DOM 树中一个节点在网页中占据的矩形区域大小,包括长和宽;

位置:DOM 树中一个节点在网页中占据的矩形区域其左上角在页面中的位置坐标(x,y);

字体:DOM 树中一个节点对应 Html 文件中元素的字体信息,包括字体的大小、颜色、样式等.

原型系统 WeRE 是使用 C#语言实现,所使用到的视觉信息可以通过 WebBrowser 组件很容易地获得,技术上的具体实现细节就不再深入介绍.

2.2 抽取特征

为了便于浏览者阅读,评论记录总是非常规则地排列在页面中.根据我们对大量真实评论页面的观察,总结了一组可用于抽取的规则.为了便于对这些规则的理解,本文定义评论区为 DOM 树中包含所有评论记录的最小子树,用符号 T_{region} 和 t_{region} 分别表示该子树及其根节点.图 3 给出了一个来自真实评论页面的评论区在 DOM 树中的一般性示例.为简单起见,只显示了页面中的 3 个评论记录.图 3(a)是评论区在页面中的示例,图 3(b)是评论区在 DOM 树中的示例.在该示例中,共有 3 个评论记录,每个评论记录由两棵子树组成,我们用符号 T_{review} 来表示这种子树,即 T_2~T_7 为 T_{review};同时用符号 T_{noise} 表示噪音子树,即 T_1 和 T_8.下面给出本文用到的抽取特征.

视觉特征:

*VF*1:评论记录自上而下左对齐顺序排列,而且记录之间宽度相同;

*VF*2:同一页面内的评论记录是模板相同的;

*VF*3:除了用户评论内容外,不同评论记录中相同语义项在视觉上是相似的(包括字体和位置),比如图 1 中用户 ID 都是使用了蓝色加粗字体.

DOM 树特征:

*DTF*1:一个评论记录由评论区根节点 t_{region} 下一棵或多棵子树组成;

*DTF*2:同模板的评论记录包含 t_{region} 下相同数量的子树;

*DTF*3:同模板的评论记录中语义的次序是一致的;

*DTF*4:除了评论记录外,往往在评论区上下两端存在噪音信息,比如“共有 xxx 条回复”和“上一页: 1|2...下一页”.

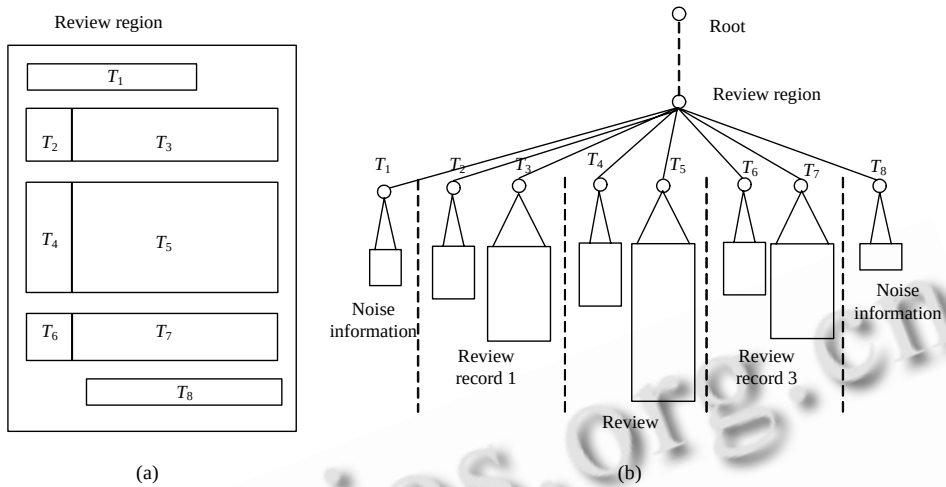


Fig.3 A general case of the review records in the Web page and in the DOM tree

图 3 评论区在页面中和 DOM 树中的一般示例

为了验证这些特征的有效性,我们在实验数据集(见第 6.1 节)上进行了统计.表 1 以百分比形式给出了统计的结果,表示评论页面满足某个特征的比例.其中,在 *VF*2,*DTF*1 和 *DTF*2 特征上仅有 1~2 个页面不符合,*DTF*4 的比例表示有 52.7%的评论页面的评论区内含有噪音信息.统计结果表明,这些特征满足绝大部分的评论页面.

Table 1 Validity of the extraction features

表 **1** 抽取特征有效性的统计

Visual features	*VF*1	100%	*VF*3	100%
	*VF*2	98.9%		
DOM tree features	*DTF*1	99.1%	*DTF*3	100%
	*DTF*2	99.3%	*DTF*4	52.7%

3 评论记录抽取

评论记录的抽取共包含 3 个步骤:首先,在页面中识别出评论区;然后,将噪音信息从评论区中去除;最后,识别评论记录的边界,即判断每个评论记录由多少棵子树组成.

在 WeRE 中,评论区的识别与文献[9]中提出的从搜索结果页面中识别搜索结果记录组的方法非常类似.文献[9]的方法使用了 4 个启发式规则进行识别.这 4 个启发式规则是:1) 占据了页面较大的区域;2) 位于页面的中央位置;3) 字符的密度非常大;4) 含有较多的记录.我们在此基础上进一步补充了评论区独有的一个特征:包含许多按升序或降序排列的日期.这是因为评论记录中都包含发表日期,并且按次序排列.评论区识别这一步骤思想非常简单,就不再过多介绍,关于规则的细节可参考文献[9].

3.1 深度加权的树相似性算法

根据视觉特征 *VF*2,同一页面中评论记录使用了相同的模板展现,因此两个评论记录中相同语义(评论内容除外)对应的子树在结构上比不同语义对应的子树更相似.但如果使用已有的树相似性算法[5],评论内容的随机性会严重削弱包含评论内容的子树之间的相似度,比如图 3 中子树 T_3,T_5 和 T_7 之间的相似度.为了避免这种情况,本文提出一种深度加权的树相似性算法 LTSA(level-weighted tree similarity algorithm).该算法的基本思想是,深层的匹配节点越多,两棵树的相似度越大.这是因为只有相同语义的子树才更有可能使深层的节点匹配上.

在 Web 数据抽取问题上,树相似性算法是为了将两棵树中相同语义的节点匹配在一起,但在已有的树相似性算法[5]中,DOM 树中节点的匹配仅要求标签一致,但 Html 标签只是用于对网页信息布局的设计,缺乏语义的

表达能力.为了改进节点匹配的准确性,我们使用了节点的视觉信息,因为相同语义的节点会使用相近的字体,比如用户的 ID 和发布时间等语义项.因此,我们定义两个节点是匹配的,当且仅当这两个节点的标签和字体同时一致.

图 4 给出了 LSTA 算法的细节.LSTA 算法由两个子算法组成:最大树相似性算法(max tree similarity algorithm,简称 MTS)和最大序列匹配算法(max sequence similarity algorithm,简称 MSS).我们使用 MTS 算法计算 DOM 树中任意两棵子树 A 和 B 的相似度,该算法的复杂度为 $O(mn)$,m 和 n 分别是两棵子树的大小(节点的数量).MTS 算法使用了递归方式的实现策略.首先,对 A 和 B 的根节点 a 和 b 进行匹配(MTS 算法中的第 3 行),如果不能够匹配上,算法停止,两棵树的相似度为 0;否则,A 和 B 的相似度为 a 和 b 的相似度加 a 和 b 的孩子序列之间的相似度(MSM 算法中的第 5 行).a 和 b 的相似度计算公式如下:

$$NI(a,b)=\exp\left(\frac{level(a,b)-avgDepth(T_{\text{region}})}{avgDepth(T_{\text{region}})}\right) \tag{1}$$

其中,$level(a,b)$是 a 或 b 在 T_{region} 中的深度,$avgDepth(T_{\text{region}})$是 t_{egion} 所有孩子子树的平均深度.从式(1)可以看出,a 和 b 的相似度 $NI(a,b)$决定于 $level(a,b)$和 $avgDepth(T_{\text{region}})$:$level(a,b)$越大或 $avgDepth(T_{\text{region}})$越小,$NI(a,b)$越大.我们并不用 $level(a,b)$来直接衡量 $NI(a,b)$是因为考虑到网页模板的差异性,否则,复杂模板的页面(T_{region} 深度大)中即使是不同语义的子树之间也可能得到很高的相似度,而简单模板的页面中即使是相同语义的子树之间的相似度也可能很低.MSM 算法用来衡量两个节点序列之间的相似度.该算法是基于动态规划的思想找到两个序列$\{a_1,a_2,\ldots,a_m\}$和$\{b_1,b_2,\ldots,b_n\}$之间这样的一个匹配,该匹配使得所有匹配节点对的相似度之和最大.算法的基本思想描述如下:假设$\{a_1,a_2,\ldots,a_m\}$和$\{b_1,b_2,\ldots,b_n\}$之间已经产生了最大相似度匹配序列,则在该匹配序列中存在两种可能性,a_1 和 b_1 匹配或者没有匹配.如果匹配,则匹配序列的相似度为 a_1 和 b_1 产生的相似度和序列$\{a_2,\ldots,a_m\}$和$\{b_2,\ldots,b_n\}$产生的相似度之和(MSS 算法中的第 3 行);如果不匹配,则匹配序列的相似度为序列$\{a_1,a_2,\ldots,a_m\}$和$\{b_2,\ldots,b_n\}$产生的相似度(MSS 算法中的第 4 行)或序列$\{a_2,\ldots,a_m\}$和$\{b_1,b_2,\ldots,b_n\}$产生的相似度(MSS 算法中的第 5 行).

Algorithm MSS //Max Sequence Similarity.
Input: $\{a_1,a_2,...,a_m\}$, $\{b_1,b_2,...,b_n\}$; //Two node sequences
Output: s; //A real value
Begin
1 If (m is 0) or (n is 0) then
2 return 0;
3 $s_0=MTS(A_1,B_1)+MSM(\{a_2,...,a_m\},\{b_2,...,b_n\})$; //$A_1$ and B_1 are the subtrees whose root nodes are a_1 and b_1 respectively
4 $s_1=MSM(\{a_1,a_2,...,a_m\},\{b_2,...,b_n\})$;
5 $s_2=MSM(\{a_2,...,a_m\},\{b_1,b_2,...,b_n\})$;
6 return $\max\{s_0,s_1,s_2\}$
End

Algorithm MTS //Max Tree Similarity.
Input: A, B; //Two subtrees
Output: s; //The similarity between A and B
Begin
1 a=the root of A;
2 b=the root of B;
3 if $NodeMatching(a,b)$ is false then
4 return 0;
5 $s=NI(a,b)+MSM(Children(a),Children(b))$;
6 return s;
End

Fig.4 LTSA algorithm

图 4 LTSA 算法

利用上面介绍的深度加权的树相似性算法,可以计算出 T_{region} 下任意两个孩子子树之间的相似度.接下来,我们分别提出了去除噪音子树的算法和评论记录边界的识别算法.

3.2 评论区中噪音信息的去除

当评论区子树下每个孩子子树与其他 n–1 棵子树之间的相似度用 LTSA 算法计算得到后,就得到了 n–1 个相似度.我们用全局相似度 gs_i(global similarity)表示子树 T_i 的 n–1 个相似度的最大值.显然,属于评论子树的全局相似性要大于噪音子树的.在实际不同模板的评论页面中,T_{review} 的全局相似度值在(3.5,25)之间,而 T_{noise} 的全局相似度值在(0,4.2)之间.因此,无法设定一个统一的阈值来区分 T_{review} 和 T_{noise},但可以通过考察全局相似度的变化趋势来进行区分,如图 5 所示.我们用相邻两棵子树全局相似度值之比 gs_i/gs_{i-1} 来表达这种变化趋势,这样就会得到 n–1 个比值.如果 T_{noise} 存在,则最大的比值产生于第 1 个 T_{review} 和恰在它之前的 T_{noise} 之间,最小的比值产生于最后一个 T_{review} 和恰在它之后的 T_{noise} 之间.基于这样的考虑,我们在图 6 中给出了噪音信息去除的算法.

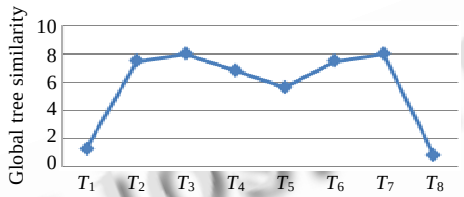


Fig.5 Change trend of the global similarities of subtrees in T_{region}

图 5 评论区中各子树全局相似度的变化曲线

Algorithm RNI //Removing Noise Information.

Input: T_{region} //The review region;

Output: *RStart*, *REnd* //*RStart* is the start position of the first review record, *REnd* is the end position of the last review record.

Begin

1 *TSet*=*getChildNodes*(T_{region});
2 *tsm*[*i*,*j*]:=0; //Initialize matrix *TSM*, *i*=0,…,*n*, *j*=0,…,*n*
3 for *i*=1 to *n* do
4 for *j*=1 to *n* do
5 if *i*≠*j* then
6 T_i=the *i*th child sub-tree of T_{dr};
7 T_i=the *i*th child sub-tree of T_{dr};
8 *m*[*i*,*j*]=*MTS*(T_i,T_j); //The tree similarity of T_i and T_j
9 for *i*=1 to *n* do
10 gs_i=Max{*m*[*i*,*j*], *j*=0,…,*n*};
11 *RStart*=Max{gs_i/gs_{i-1}}; //*i*<*n*
12 *REnd*=Min{gs_{i-1}/gs_i}; //*i*<*n*
13 If *RStart*<λ then //λ is the predefined threshold
14 *RStart*=0;
15 If *REnd*<λ then
16 *REnd*=*n*;
17 Return *Rstart* & *REnd*;
End

Fig.6 Algorithm of removing noise information

图 6 噪音信息去除算法

该算法的关键在于通过找到第 1 棵 T_{review} 和最后一棵 T_{review} 来达到去除噪音的目的.首先,使用 LTSA 算法计算 T_{region} 下任意两棵孩子子树之间的相似度,可以用矩阵的形式来存储,用 *TSM*[*n*,*n*]来表示(图 6 算法中的第 1 行~第 8 行).然后,得到每棵子树的全局相似度(图 6 算法中的第 9 行、第 10 行),并计算相邻子树之间的全局相似度之比,根据这些比值中的最大值和最小值得到第 1 棵 T_{review} 的位置 *RStart* 和最后一棵 T_{review} 的位置 *REnd*(图 6 算法中的第 11 行、第 12 行).并不是所有的评论区中都有噪音信息,为了避免将 T_{review} 误认为 T_{noise},我们设置了一个阈值 2.5,即只有当最大比值大于 2.5 时,前一棵子树才被认为是 T_{noise};当最小比值小于 1/2.5 时,后一棵子树才被认为是 T_{noise}.该阈值是在实验中通过不断调整得到的使整体抽取准确性达到最好的值.最后,*RStart* 和 *REnd* 被返回.比如,在图 3 的例子中,数字 2 和 7 被算法返回,表示第 1 棵子树 T_{review} 和最后一棵 T_{review} 分别为 T_2 和 T_7,T_1 和 T_8 作为噪音子树从 T_{region} 中删除.

3.3 评论记录边界的识别

由于评论记录的 *DTF*1 特征,在上一步把噪音信息删除后,我们需要进一步识别出评论记录之间的边界.根据 *DTF*2 特征,只要确定一个评论记录由几棵连续的 T_{review} 组成即可,设为 l 棵,$l\geq 2$.但当前页面中评论记录的数量是未知的,因此无法直接确定 l 的值.本节将介绍一种简单有效的方法来推测 l 的值.

前面已经计算出每棵 T_{review} 的全局相似度,利用这些信息可以构造一个有向图 $G(V,E)$:V 是点的集合,E 是边的集合.V 中的点与评论子树 T_i 一一对应,存在一条从 T_i 到 T_j 的边当且仅当 T_i 的全局相似度为 T_i 和 T_j 之间的相似度.在本文中,将该图称为全局相似度关系图(global similarity relationship graph,简称 GSRG).在 GSRG 中,每个节点的出度必然为 1,入度在 0 到 $n-1$ 之间.比如在图 3 的例子中,根据表 1 形成的 GSRG 如图 7 所示.下面我们基于该图提出识别评论记录边界的算法.

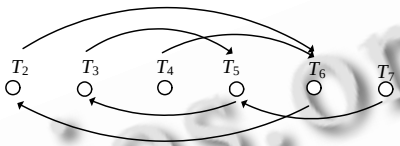


Fig.7 An illustration of global similarity relationship graph

图 7 全局相似度关系图示例

图 8 给出了评论记录边界的识别算法.该算法的目的是推测 l 的值,即一个评论记录由几个 T_{review} 组成.首先,根据图 GSRG 计算距离 $d_i=|i-j|$(第 1 行~第 3 行),T_j 是与 T_i 最相似的子树.由于 T_i 和 T_j 是不同记录中相同语义的子树,根据 *DTF*2 特征,l 必然是$\{d_1,...,d_n\}$的某个公约数.如果$\{d_1,...,d_n\}$只有一个不为 1 的公约数 l',则把该公约数 l'作为返回(图 8 中算法的第 4 行~第 6 行).如果 l'不是素数,即它可以继续分解为更小的公约数,换句话说,l 为一个比 l'更小的数.因此,n 棵 T_{review} 被分成 n/l'个子序列,然后对每个子序列继续递归,使用该算法得到一个 l',取最小的一个 l'作为 l 返回(第 8 行~第 12 行).我们利用图 8 中的例子来示例这个算法.根据图中有向边在节点之间的关系,子树 $T_2,\ldots,T_7$ 对应的距离分别为 4,2,2,2,4,2.这表示每个评论记录由两棵 T_{review} 组成.

Algorithm DBRR //Detecting the Boundaries of Review Records.
Input: $T_1, T_2,\ldots,T_n$; //n T_{review}s extracted by Algorithm RSEA
Output: l; //One review record consists of l T_{review}s
Begin
1 For i=1 to n do
2 T_j=*getMaxSimTree*(T_i); //Find the most similar sub-tree T_j of T_i based on GSRG
3 $d_i=|i-j|$;
4 l'= *MaximalCommonDivisor* ($d_1,...,d_n$);
5 If l' is a prime number then
6 $l=l'$; Return l;
7 Else
8 For k=0 to $n/l'-1$ do
9 l'=*DBRR*($T_{k+1},T_{k+2},\ldots,T_{k+l'}$);
10 If $l>l'$
11 $l=l'$;
12 Return l;
End

Fig.8 Algorithm of detecting the boundaries of review records

图 8 评论记录边界的识别算法

4 评论内容抽取

本节讨论如何从评论记录中抽取纯粹的评论内容.由于评论内容是由不同用户自由撰写,因此无论是文本长度还是在 DOM 树中子树结构上都表现出了很大的不一致性.因此,我们将基于子树结构和文本长度两个方面的不一致性来提出评论内容的抽取方法.

4.1 识别包含评论内容的T_{review}

当每个评论记录包含多棵 T_{review} 时,要首先识别出那棵 T_{review} 包含了评论内容.根据 *DTF*3,每个评论记录中第 i 棵 T_{review} 语义是相同的.因此,我们把每个评论记录中第 i 棵 T_{review} 组成一组.如果每个评论记录由 l 棵 T_{review} 组成,那么就得到了 l 个不同语义的组:$\{G_1,\ldots,G_l\}$.比如图 7 中,T_2~T_7 这 6 棵 T_{review} 被分为 2 组$\{\{T_2,T_4,T_6\},\{T_3,T_5,T_7\}\}$.下面将使用标准差的方法来确定哪一组包含各个评论记录中的评论内容.衡量一个组的标准差的公式如下

$$SD(G_i)=\sqrt{\frac{\sum_{j=1}^{|G_i|}(N_i-\bar{N})^2(W_i-\bar{W})^2}{|G_i|}} \tag{2}$$

其中,$|G_i|$是指 G_i 中的数量,N_j 是指 G_i 中第 j 棵 T_{review} 中节点的数量,$\bar{N}$ 是指 G_i 中所有 T_{review} 中节点的平均值;W_j 是指 G_i 中第 j 棵 T_{review} 中的文本长度,$\bar{W}$ 是指 G_i 中所有 T_{review} 中的文本长度平均值.很明显,包含评论内容的一组其标准差是最大的,因为相对于其他语义的组,包含评论内容的组中各个 T_{review} 在节点数量和文本长度上是最不一致的.

4.2 抽取包含评论内容的最小子树

在 DOM 树中,每个评论记录中的评论内容对应了一棵复杂的子树而不是一个简单的叶节点,因而评论内容抽取的目标是从中找到包含评论内容的最小子树,本文用 $T_{content}$ 来表示.评论内容抽取的方法采用了 3 步策略:首先,一个评论页面的所有评论记录中包含评论内容的 T_{review} 进行对齐操作,得到一棵对齐树 T_A;然后,计算 T_A 中每个节点的一致性,并进一步计算 T_A 中每棵子树的一致性;最后,根据 T_A 中每棵子树的一致性,在每个评论记录的包含评论内容的 T_{review} 中抽出其 $T_{content}$.下面是对每一步具体的介绍.

4.2.1 子树对齐

给定若干棵树,树对齐是指构造一棵超树 T_A,或者说,所有的树通过插入一些节点变成同构的,使得各个树中相同语义的节点被对齐为 T_A 中的一个节点.考虑到 T_{review} 是有序标记树,我们使用文献[10]提出的快速树对齐算法 FastScore 来对齐任意两棵包含评论内容的 T_{review}.n 棵包含评论内容的 T_{review} 的对齐过程描述如下:首先,所有的包含评论内容的 T_{review} 根据树的大小按降序排列.然后,前两棵 T_{review} 对齐为 T',第 3 棵 T_{review} 与对齐为 T'',这样,直到所有树对齐得到 T_A.在这个过程中,每个节点被对齐的次数被记录下来.如果 T_A 中一个节点被对齐了 m 次,表示该节点存在于 m 棵 T_{review} 中.图 9 展示了一个树对齐的示例,其中每个节点括号中的数字表示该节点被对齐的次数.直觉上,当一个节点不属于评论内容时,往往具有较高的对齐次数,因为该节点是由页面的模板产生;相反,当一个节点属于评论内容时,对齐次数一般较低,因为不同评论记录中 $T_{content}$ 彼此结构上的巨大差异造成 T_A 中 $T_{content}$ 中的节点较低的对齐次数.

4.2.2 节点一致性和树一致性

根据树对齐算法,每个 T_{review} 中 $T_{content}$ 的根节点必然对齐在 T_A 中的一个节点上.图 9 中虚线包围的子树即每个 T_{review} 中的 $T_{content}$,这些经过对齐就得到了 T_A 中的 $T_{content}$.因此,只要在 T_A 中抽取出 $T_{content}$ 就实现了评论内容的抽取.T_A 可以被分为两部分:$T_{content}$ 和 $T_A-T_{content}$,$T_A-T_{content}$ 表示 T_A 除去 $T_{content}$ 剩余的部分.由于 $T_{content}$ 是不同用户自主产生,不受页面模板的限制,$T_{content}$ 中的节点和 $T_A-T_{content}$ 中的节点在总体上存在两个方面的区别:(1) 平均来说,$T_{content}$ 中节点的对齐次数要远远小于 $T_A-T_{content}$ 中节点的;(2) $T_{content}$ 中节点包含文本的长度在各个评论记录中往往具有较大的差异.因此,我们基于这两方面提出节点一致性和子树一致性的概念来区分 $T_{content}$ 和 $T_A-T_{content}$.首先给出节点一致性的定义:

$$Consis(a)=\frac{m}{n}\left(\sum_{i=1}^{m}-\frac{|W_i|}{|W|}\ln\frac{|W_i|}{|W|}\right) \tag{3}$$

式(3)中 a 是 T_A 中的任一节点,n 是评论记录的数量,m 是节点对齐的次数,$|W_i|$是节点 a 在第 i 个评论记录中

包含文本的长度,$|W|$是节点 a 在所有评论记录中的总长度.公式(3)由 $\frac{m}{n}$ 和 $\sum_{i=1}^{m}-\frac{|W_i|}{|W|}\ln\frac{|W_i|}{|W|}$ 两个部分组成.前者用于衡量节点 a 在树结构上的一致性,后者用来衡量节点 a 在文本长度上的一致性.根据这个公式,如果一个节点在越多的评论记录中出现,在所出现的各个评论记录中包含的文本长度越接近,它的一致性越大.因此,在一般情况下,T_{content} 中节点的一致性要明显低于 T_A-T_{content} 中节点的一致性.

基于节点一致性的定义,我们进一步定义子树 T 的一致性为该子树中所有节点一致性的平均值,表示为 $Consis(T)$.显然,T_{content} 的一致性要小于 T_A-T_{content} 的一致性.但 T_{content} 并不一定是一致性最小的子树,因为 T_{content} 的某些子树常会比 T_{content} 具有更小的一致性.因此,并不能将 T_A 中一致性最小的子树作为 T_{content}.下一步将介绍基于子树一致性抽取 T_{content} 的方法.

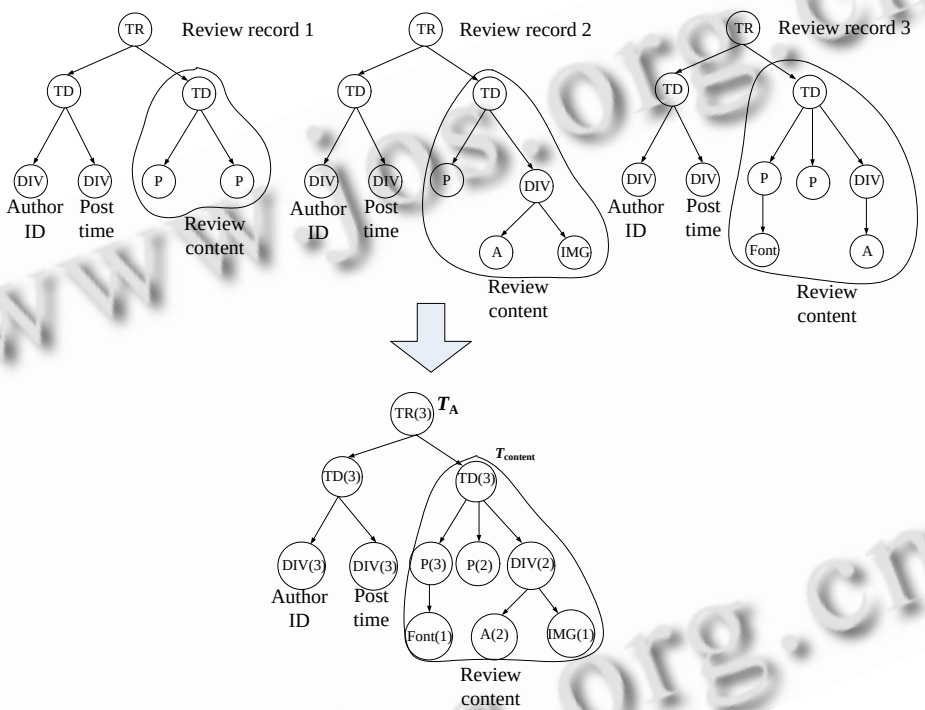


Fig.9 An example of tree alignment

图 9 树对齐示例

4.2.3 T_A 中 T_{content} 的抽取

首先,定义一致性差为

$$ConsisDiff(T_i)=Consis(T_A-T_i)-Consis(T_i) \quad (4)$$

公式(4)中,T_i 是 T_A 的一棵子树.实际上,在所有的子树中,$ConsisDiff(T_{\text{content}})$ 是最小的,我们通过反证的方式进行解释.假设 T_A 中存在一棵子树 T_i,使得 $ConsisDiff(T_i)>ConsisDiff(T_{\text{content}})$.那么 T_{content} 和 T_i 之间的关系有 3 种可能:一是 T_i 包含 T_{content};二是 T_{content} 包含 T_i;三是 T_{content} 和 T_i 之间没有共同的节点.

对于第 1 种可能性,T_i-T_{content} 中的节点不属于评论内容,因此 $Consis(T_i-T_{\text{content}})>Consis(T_{\text{content}})$,进一步可断定 $ConsisDiff(T_i)>ConsisDiff(T_{\text{content}})$.对于第 2 种可能,$T_{\text{content}}-T_i$ 中的节点属于评论内容,因此 $Consis(T_{\text{content}}-T_i)<Consis(T_A-T_{\text{content}})$,可以推出 $Consis(T_A-T_i)-Consis(T_i)<Consis(T_A-T_{\text{content}})-Consis(T_{\text{content}})$.对于第 3 种可能,显然 $Consis(T_A-T_i)<Consis(T_A-T_{\text{content}})$,因为 T_A-T_{content} 中的节点为非评论内容节点,而 T_A-T_i 包含了所有评论内容的节点.

基于上述讨论,图 10 给出了评论内容抽取的算法.在该算法中,各个评论记录中包含评论内容的 T_{review} 被对

齐为 T_A.对齐后,计算 T_A每个节点的一致性(第 1 行~第 3 行).在 T_A中,按以下策略选择一条路径 $Q\{a_0,a_1,\ldots,a_Q\}$(第 4 行~第 10 行):Q 的起始节点为 T_A 的根节点;设当前 Q 为$\{a_0,a_1,\ldots,a_i\}$,将 a_{i+1} 加入 Q 仅当 a_{i+1} 为 a_i 的孩子节点且 a_{i+1} 在 a_i 所有的孩子节点对应的子树中一致性是最小的.当 Q 生成后,对 Q 中的每个节点 a_i 分别计算 $ConsisDiff(T_i)$,T_i 是 a_i 对应的子树.如果在 Q 中所有节点中 $ConsisDiff(T_i)$是最大的,则节点 a_i 对应的子树为 T_{content}(第 11 行~第 17 行).

我们通过图 11 中的例子进一步解释这个算法.首先根节点 a 放入 Q,a 有两个孩子节点 b 和 c,如果 $ConsisDiff(T_b)<ConsisDiff(T_c)$,则将 c 加入 Q,然后继续考察孩子节点.重复这个过程,一直遇到叶子节点为止.这样就得到$\{a,c,f,i\}$,如图 11 中阴影节点所示.下一步,分别计算 $ConsisDiff(T_a)$,$ConsisDiff(T_c)$,$ConsisDiff(T_f)$和 $ConsisDiff(T_i)$.如果 $ConsisDiff(T_c)$最大,则 T_c 为 T_{content}.

Algorithm T_{content} extraction.
Input: T_1, T_2,…, T_l; //l the T_{review}s that are obtained by the method proposed in section 4.1
Output: T_{content}; //The root of the minimal tree that contains all review contents in T_A
Begin
1 T_A=*TreeAlignment*(T_1,T_2,…,T_l)
2 For each node a in T_A do
3 Compute *Consis*(a);
4 Initialize queue Q; //Storing a path (from the root to a leaf node) of T_A
5 a_p=the root of T_A;
6 Put a_p into Q;
7 While a_p is not a leaf node do
8 a_c is the child node of n_p whose *Consis*(T_c) is minimum; //T_c is a_c's subtree
9 Put a_c into Q
10 a_p=a_c;
11 $s_{\max}$=+∞;
12 While Q is not null do
13 a_i=get a node from Q;
14 If (*Consis*(T_A–T_i)–*Consis*(T_i)<$s_{\max}$) then
15 $s_{\max}$=(*Consis*(T_A–T_i)–*Consis*(T_i); //T_i is a_i's subtree
16 a_m=a_i;
17 Return a_m
End

Fig.10 Algorithm of review content extraction
图 10 评论内容抽取算法

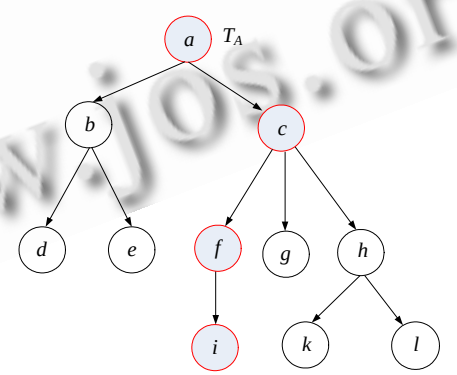


Fig.11 An example to illustrate the algorithm T_{content} extraction
图 11 评论内容抽取算法示例

5 基于 Wrapper 的抽取

由于直接抽取计算代价较高,且当出现页面中只有一个评论记录的极端情况时直接抽取方式将会失败,因

此本节将介绍原型系统 WeRE 提供的另外一种抽取方式:基于 wrapper 的抽取.该抽取方式是利用一个样本页面生成 wrapper 抽取程序来对相同模板的其他评论页面进行抽取,进行样本页面时选择包含多个评论记录的页面.WeRE 是自动选取的:把所有页面按照文件大小降序排列,选择最大的页面作为样本页面.这样保证了样本页面中包含有多个评论记录.

Wrapper 的生成. 一共需要生成两个 wrapper,分别是评论记录抽取 wrapper 和评论内容抽取 wrapper.每个 wrapper 中记录了要抽取信息的一些特征作为抽取规则.这些抽取规则共分为 3 类:Html 标签路径、位置与宽度和字体.表 2 给出这 3 类特征在两个 wrapper 中的具体含义.我们对样本页面使用直接抽取,在抽取过程中获取这些抽取规则.评论记录抽取 wrapper 中还记录了每个评论记录包含 T_{review} 的数量.

Table 2 Three types of features in wrappers

表 **2** Wrapper 中包含的 3 类抽取规则

Wrapper	Html tag path	Position and width	Font
Review record wrapper	The tag path from root of DOM tree to T_{review}	Position feature refers to the distance from the left border of the review page to the left border of review records; width feature refers to the width of review records	The shared fonts in review records
Review content wrapper	The tag path from root of T_{review} to T_{content}	Position feature refers to the distance from the left border of the review records to the left border of review contents; width feature refers to the width of review contents	The shared fonts in review contents

Wrapper 的抽取. 当两个 wrapper 生成以后,就可以用这两个 wrapper 从与样本页面相同模板的页面中依次抽取评论记录和评论内容.使用 wrapper 抽取评论记录时,首先从 DOM 树中抽取出与 Html 标签路径特征一致的子树,比如 body\div\...\table\tr,但并不能保证这些子树都是 T_{review}.然后,根据位置和宽度规则以及字体规则征将不符合的子树去除.根据每个评论记录包含 T_{review} 的数量就可以将 T_{review} 序列进行切分,得到每个评论记录了.用 wrapper 抽取评论内容的过程类似,不再赘述.

6 实验评估

6.1 实验数据集

实验中用到的数据集来自 10 个新闻网站和 20 个论坛网站的评论页面,我们实现了一个评论页面爬虫从这些网站中爬取评论页面到本地.该爬虫中评论页面识别规则是人工根据这些网站中评论页面的 URL 编写的正则表达式,因此爬取的准确性在 99.9%以上.根据来源不同,可以分为新闻评论数据集(news data set,简称 NDS)和论坛评论数据集(forum data set,简称 FDS).实验将在这两个数据集分别展开,表 3 和表 4 分别给出这两个数据集的详细信息.对数据集的标注采用可视化的标注工具进行人工标注.给定一个评论页面的 URL,该标注工具可以分别显示该页面和该页面的 DOM 树,当点击 DOM 树的某个节点时,会显示该节点包含的所有文本内容.这样,根据文本内容可以保证标注结果的正确性.

Table 3 Details on NDS

表 **3** 新闻评论数据集的详细信息

News sites	Number of review pages	Number of review records
www.xinhuanet.com	242	4 401
www.people.com.cn	999	20 889
www.southcn.com	227	1 811
www.mofcom.gov.cn	999	19 980
www.sohu.com	964	19 223
www.xinhuanet.com	1 000	19 958
www.sina.com	999	19 367
www.huanqiu.com	1 000	9 971
news.163.com	931	27 928
www.qq.com	1 000	10 025
Total	8 361	153 553

Table 4 Details on FDS

表 **4** 论坛评论数据集的详细信息

Forum sites	Number of review pages	Number of review records
forum.ubuntu.org.cn	1 001	8 911
bbs.winos.cn	1 001	8 118
wordpress.org.cn	1 000	7 983
bbs.zdnet.com.cn	1 000	1 800
bbs.tech.china.com	998	6 018
bbs.zhcw.com	999	8 172
bbs.chuguo.cn	999	5 780
www.hupub.com	1 000	8 339
www.jd-bbs.com	998	12 173
bbs.55bbs.com	1 014	8 411
bbs.imobile.com.cn	1 000	17 186
sq.k12.com.cn/discuz	1 000	16 726
bbs.cqzg.cn	999	3 343
forum.jiangmin.com	1 001	9 127
bbs.ikaka.com	1 001	6 703
itbbs.pcshow.net	981	7 428
forum.livetome.cn	1 000	4 302
bbs.mydrivers.com	1 000	5 129
huatan.net/bbs	1 000	8 594
bbs.canet.com.cn	1 000	11 401
Total	19 992	165 644

6.2 评估标准和实验设置

本文采用传统的正确率(Precision)、召回率(Recall)和 *F*-measure 对实验结果进行评估.正确率是指所有抽取出来的评论记录(或评论内容)中正确抽取的评论记录(或评论内容)所占的比例.召回率是指正确抽取的评论记录(或评论内容)占被抽取页面中所有评论记录(或评论内容)的比例.

$$F\text{-measure}=2\times Recall\times Precision/(Recall+Precision).$$

在本实验中,正确抽取的评论记录(或评论内容)严格定义为既没有缺失任何内容,也没有任何多余的内容.这 3 项标准是 Web 数据抽取研究领域普遍采用的评估标准.此外,评论记录抽取时阈值设为 2.5.在第 6.6 节,我们将量化分析评论记录抽取时阈值对抽取准确性的影响.

6.3 评论记录抽取实验分析及与相关工作比较

本节评估 WeRE 在评论记录抽取方面的准确性.输入是可在 IE 浏览器中正常显示的评论页面,输出是抽取出来的评论记录.作为对比,我们还使用两个代表性的 Web 数据抽取工具 MDR[5]和 ViDRE[6]在相同的数据集上进行测试.MDR 在 DOM 树上直接抽取 Web 数据记录,ViDRE 则只利用了页面的视觉信息并且生成 wrapper 来抽取.表 5 给出了 WeRE 在评论记录抽取上的准确性,表 6 给出了这两个 Web 数据抽取工具的实验结果.通过分析表 5 和表 6 中的实验结果可以得出 3 个结论:第一,WeRE 在评论记录抽取上已经达到了非常高的准确性,特别是在 precision 评估标准上达到了 99.9%甚至 100%,这表明,我们提出的评论记录抽取方法已经达到实际应用的要求;第二,从总体准确性来说,基于 wrapper 的抽取方式要略优于直接抽取方式,这是由于当处理页面中只有一个评论记录时,直接抽取方式无法识别;第三,在评论记录抽取方面,WeRE 明显优于已有的抽取方法 MDR 和 ViDRE.通过检查 MDR 和 ViDRE 抽取错误和失败的例子,我们发现当一个评论记录与页面中其他记录在 DOM 树结构或视觉存在较大差异时,MDR 和 ViDRE 往往会抽取错误或者将其作为噪音信息忽略.

Table 5 Experimental results on review record extraction

表 5 评论记录抽取实验结果

	Direct extraction		
	Precision	Recall	*F*-measure
NDS	99.9%	98.6%	99.2%
FDS	100%	98.9%	99.4%
	Wrapper based extraction		
	Precision	Recall	*F*-measure
NDS	99.9%	98.8%	99.9%
FDS	99.9%	99.0%	99.6%

Table 6 Experimental results on MDR and ViDRE

表 6 MDR 和 ViDRE 的实验结果

	MDR		
	Precision	Recall	*F*-measure
NDS	77.3%	89.2%	82.8%
FDS	64.6%	81.5%	71.9%
	ViDRE		
	Precision	Recall	*F*-measure
NDS	92.7%	90.3%	91.5%
FDS	86.4%	82.2%	84.2%

6.4 评论内容抽取实验分析及与相关工作比较

在本部分实验,我们评估 WeRE 在评论内容抽取方面的准确性.输入是前面评论记录抽取实验中正确抽取出来的评论记录,输出是这些评论记录中的评论内容.一个评论记录中评论内容被认为正确抽取出来当且仅当其不包含任何噪音信息而且不缺失任何属于它的信息.表 7 给出了该部分实验的结果.从该实验结果中可以得出两个结论:第一,总体来说,WeRE 在评论记录抽取方面表现出了较高的准确性,除了直接抽取方式在 FDS 数据集上表现稍低外,在 3 个评估指标上都在 97%以上;第二,直接抽取方式与基于 wrapper 的抽取方式在 NDS 数据集上实验结果较为接近,但在 FDS 数据集上基于 wrapper 的抽取方式要明显优于直接抽取方式,通过分析发现是因为论坛评论内容比新闻评论内容更为复杂,造成直接抽取方式在 FDS 数据集上的准确性比在 NDS 数据集上的准确性差一些.

文献[11]也是针对 Web 评论提出的自动抽取方法,在应用角度上与本文的研究问题最为接近.他们是在 20 个论坛网站上对所提出的方法进行实验评估.由于数据集不同,而且也没有提供可下载或可在线访问的原型系统,因此无法直接比较抽取准确性的优劣.我们从文献[11]实验中用到的 20 个论坛网站中各爬取了 100 个评论页面作为数据集进行实验.表 8 上半部分是 WeRE 在该数据集上的实验结果,表 8 下半部分是文献[11]报告的实验结果.通过对比可以发现:在评论记录抽取上,WeRE 略优于 MLNs-PVV;在评论内容抽取上,WeRE 的抽取准确性要明显高于 MLNs-PVV,特别是在 recall 评估标准上要超出 10%.

Table 7 Experimental results on review content

表 7 评论内容抽取实验结果

	Direct extraction		
	Precision	Recall	*F*-measure
NDS	97.5%	97.1%	97.2%
FDS	90.1%	87.3%	88.6%
	Wrapper based extraction		
	Precision	Recall	*F*-measure
NDS	99.9%	99.2%	99.5%
FDS	98.5%	98.3%	98.4%

Table 8 Comparison results between WeRE and MLNs-PVV

表 8 WeRE 和 MLNs-PVV 的整体对比

	WeRE		
	Precision	Recall	*F*-measure
Review record	99.4%	99.1%	99.3%
Review content	97.3%	95.8%	96.5%
	MLNs-PVV		
	Precision	Recall	*F*-measure
Review record	99.6%	94.1%	96.7%
Review content	91.4%	85.3%	88.2%

6.5 效率实验

除了抽取的准确性,效率也是评价实际应用系统的一个重要因素.因此,我们在抽取效率方面对 WeRE 进行实验.表 9 给出了 WeRE 在抽取时间方面的实验结果,从中可以得出两个重要的结论:第一,基于 wrapper 的抽取方式在时间上要明显优于直接抽取方式.平均来说,直接抽取方式每秒可以处理接近两个评论页面,而基于 wrapper 的抽取方式每秒可以处理超过 3 个.虽然基于 wrapper 的抽取需要预先训练为每个网站生成 wrapper,生成 wrapper 的时间大约为 500ms,但如果每个网站中评论页面数量很多时(比如超过 100),这个时间代价可以被忽略;第二,影响抽取效率最主要的因素是页面的解析.WeRE 是利用微软的 WebBrowser 组件来解析评论页面

并获取页面视觉信息的,该过程占了整个抽取过程 60%以上的时间.相比而言,无论是直接抽取方式还是基于 wrapper 的抽取方式,评论记录抽取和评论内容抽取只占整个抽取时间较小的比例.

Table 9 Experimental results on efficiency (ms)

表 9 抽取效率上的实验结果 (ms)

	Direct extraction	Wrapper based extraction
Page presentation	314.6	314.6
Record extraction	136.8	6.9
Content extraction	110.2	7.4
Total time per page	561.6	328.9

6.6 评论记录抽取中阈值对抽取准确性的影响

为了量化分析评论记录抽取中阈值对准确性的影响,我们使阈值在 1.5~3 之间取值,在两个数据集 FDS 和 NDS 上进行实验,分析不同阈值下 *F*-measure 的变化.我们发现,当阈值取到 2.5 时,两个数据集上的 *F*-measure 分别达到最大值 99.4%和 99.2%.这是因为如果阈值设得过低,噪音信息会被误判为评论子树,导致 precision 的下降;反之,则会把评论子树误判为噪音子树,导致 recall 的下降.如图 12 所示,横轴为阈值的取值,纵轴为 *F*-measure.

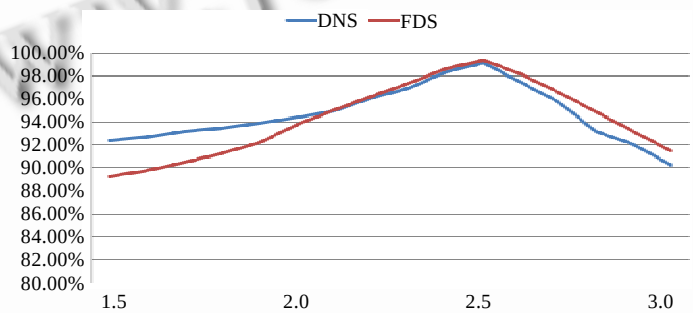


Fig.12 Effect of the threshold on the extraction performance

图 12 阈值对抽取准确性的影响

7 相关工作

Web 评论内容的抽取属于 Web 数据抽取领域,该研究领域受到了大量关注,文献[4]对该领域已有的研究成果做了较好的总结.根据自动化程度,已有的抽取方法可以分为 3 类:手工、半自动和全自动.早期的方法主要是手工的和半自动的,比如文献[12−14]提出的方法.而当前的应用需要从大量页面中抽取数据,为了改进效率以及降低人工参与的代价,目前的工作主要集中在全自动方法的研究上.本文所提出的方法就是一种全自动的方法.本节主对全自动方法进行简要的介绍和比较.

自动方法中比较有代表性的是 Omini[15],IEPAD[16],RoadRunner[17],MDR[5]和 DEPTA[18].这其中的有些方法只是对数据记录的抽取,并没有进一步从数据记录中抽取数据项,比如 Omini.有些方法并不生成 wrapper,而是通过识别模式直接从页面中抽取,无需通过训练产生抽取规则.这些工作已经在文献[4]中进行了讨论和比较.另外,还有些工作研究如何从 Web 数据记录中抽取出其中的数据项,比如 DeLa[19],DEPTA 和文献[20]提出的方法.DeLa 利用 Html 标签信息构造正则表达式 wrapper,可以从 Web 表格中抽取出数据项.与 DeLa 类似,DEPTA 仍然在 Html 标签树上操作,通过对齐两个 Web 数据记录中相同语义的数据项达到抽取的目的.这些方法只利用 DOM 树的结构和 Html 标签信息,前面第 1 节中已经指出了 Html 标签缺乏语义表达能力的缺陷.最近的研究已经注意到这个缺陷,提出的方法主要是利用页面的视觉信息来实现 Web 数据的抽取,比如 ViNTS[9],ViPERS[7].ViNTS 利用了搜索结果页面中视觉内容特征来捕获 Web 数据记录之间的规律性,然后结合 Html 标签结构提高

抽取的准确性.ViPER 同样利用了页面的视觉信息,结合多序列对其技术来实现 Web 数据记录的抽取.这两种方法并没有实现对 Web 数据记录中数据项的抽取.这些方法都是理想地假设要抽取的数据项为 DOM 树中一个的叶节点.显然,这个假设对评论内容而言并不成立,因为评论内容在 DOM 树中通常是一棵结构复杂的子树.

目前已经有一些工作研究如何从网站中自动获取 Web 用户评论,但大部分关注于评论页面的爬取,比如文献[1−3,21].众所周知,评论页面中通常是多个评论记录顺序排列,并且包含大量的像广告、导航之类的噪音信息,如果评论分析系统将整个页面作为处理对象,显然会严重影响对评论分析的质量.文献[22]提出的是评论记录抽取的问题,没有给出如何进一步从评论记录中抽取评论内容,本文是这个工作的进一步扩展.文献[11]是与本文最接近的工作,它将页面层次和网站层次的知识结合通过学习各种特征的重要性来集成.然而获取网站层次的知识代价是非常高的,为了重新构造网站地图,需要爬取大量的非评论页面,而且页面的聚类时间代价也是非常高的.因此,当需要从许多网站中抽取 Web 用户评论时,这种方法在效率上就不符合实际应用的需求了.与之相比,本文所提出的方法是轻量级的,不需要任何评论页面之外的复杂知识,仅根据评论记录的若干一般特征(见第 2.2 节)就可以实现准确的抽取.

8 结 论

本文提出了一种从评论页面中自动抽取各个评论记录中评论内容的方法,从技术上解决了因评论内容由不同的一般用户自主撰写带来的 Web 数据抽取上新的挑战.抽取方法主要分为两个步骤:评论记录的抽取和评论内容的抽取.基于该抽取方法实现了 Web 评论抽取原型系统 WeRE.在 10 个新闻网站和 20 个论坛网站中大量评论页面上的实验,表明了所提出的方法可以达到较高的准确性,优于现有的抽取方法.

References:

[1] Cai R, Yang JM, Lai W. iRobot: An intelligent crawler for Web forums. In: Huai J, Chen R, Hon H, Liu Y, Ma W, Tomkins A, Zhang X, eds. Proc. of the Int'l Conf. on World Wide Web (WWW 2008). Beijing: ACM Press, 2008. 447−456.

[2] Guo Y, Li K, Zhang K. Board forum crawling: A Web crawling method for Web forum. In: Nishida T, ed. Proc. of the Int'l Conf. on Web Intelligence (WI 2006). Hong Kong: IEEE Computer Society, 2006. 745−748.

[3] Wang Y, Yang JM, Lai W. Exploring traversal strategy for Web forum crawling. In: Myaeng S, Oard D, Sebastiani F, Chua T, Leong M, eds. Proc. of the ACM Conf. on Research and Development in Information Retrieval (SIGIR 2008). Singapore: ACM Press, 2008. 459−466.

[4] Chang CH, Kayed M, Girgis MR, Shaalan KF. A survey of Web information extraction systems. IEEE Trans. on Knowledge and Data Engineering, 2006,18(10):1411−1428.

[5] Liu B, Grossman RL, Zhai Y. Mining data records in Web pages. In: Getoor L, Senator T, Domingos P, Faloutsos C, eds. Proc. of the Int'l Conf. on Knowledge Discovery and Data Mining (KDD 2003). Washington: ACM Press, 2003. 601−606.

[6] Liu W, Meng X, Meng W. Vision-Based Web data records extraction. In: Zhou D, ed. Proc. of the Int'l Workshop on the Web and Databases (WebDB 2006). 2006. 20−25. http://db.ucsd.edu/webdb2006/camera-ready/paginated/04-144.pdf

[7] Simon K, Lausen G. ViPER: Augmenting automatic information extraction with visual perceptions. In: Herzog O, Schek H, Fuhr N, Chowdhury A, Teiken W, eds. Proc. of the Int'l Conf. on Information and Knowledge Management (CIKM 2005). Bremen: ACM Press, 2005. 381−388.

[8] Song R, Liu H, Wen JR, Ma WY. Learning block importance models for Web pages. In: Feldman S, Uretsky M, Najork M, Wills C, eds. Proc. of the Int'l Conf. on World Wide Web (WWW 2004). New York: ACM Press, 2004. 203−211.

[9] Zhao H, Meng W, Wu Z, Raghavan V, Yu CT. Fully automatic wrapper generation for search engines. In: Ellis A, Hagino T, eds. Proc. of the Int'l Conf. on World Wide Web (WWW 2005). Chiba: ACM Press, 2005. 66−75.

[10] Jansson J, Lingas A. A fast algorithm for optimal alignment between similar ordered trees. In: Amir A, Landau G, eds. Proc. of the Int'l Conf. on Combinatorial Pattern Matching (CPM2001). Jerusalem: Springer-Verlag, 2001. 232−240.

[11] Yang J, Cai R, Wang Y. Incorporating site-level knowledge to extract structured data from Web forums. In: Quemada J, León G, Maarek Y, Nejdl W, eds. Proc. of the Int'l Conf. on World Wide Web (WWW 2009). Madrid: ACM Press, 2009. 181−190.

[12] Adelberg B. NoDoSE—A tool for semi-automatically extracting semi-structured data from text documents. In: Haas L, Tiwary A, eds. Proc. of the Int'l Conf. on Management of Data (SIGMOD 1998). Seattle: ACM Press, 1998. 283−294.

[13] Hammer J, Garcia-Molina H, Nestorov S. Template-Based wrappers in the TSIMMIS system. In: Peckham J, ed. Proc. of the Int'l Conf. on Management of Data (SIGMOD 1997). Tucson: ACM Press, 1998. 532−535.

[14] Kushmerick N, Weld D, Doorenbos R. Wrapper induction for information extraction. In: Pollack M, ed. Proc. of the Int'l Joint Conf. on Artificial Intelligence (IJCAI'97). Nagoya: Morgan Kaufmann Publishers, 1997. 729−737.

[15] Buttler D, Liu L, Pu C. A fully automated object extraction system for the WWW. In: Lanus M, ed. Proc. of the Int'l Conf. on Distributed Computing Systems (ICDCS 2001). Phoenix: IEEE Computer Society, 2001. 361−370.

[16] Chang CH, Hsu CN, Lui SC. Automatic information extraction from semi-structured Web pages by pattern discovery. Decision Support Systems, 2003,35(1):129−147.

[17] Crescenzi V, Mecca G, Merialdo P. RoadRunner: Towards automatic data extraction from large Web sites. In: Apers P, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass R, eds. Proc. of the Int'l Conf. on Very Large Data Bases (VLDB 2001). Rome: Morgan Kaufmann Publishers, 2001. 109−118.

[18] Zhai Y, Liu B. Web data extraction based on partial tree alignment. In: Ellis A, Hagino T, eds. Proc. of the Int'l Conf. on World Wide Web (WWW 2005). Chiba: ACM Press, 2005. 76−85.

[19] Wang J, Lochovsky FH. Data extraction and label assignment for Web databases. In: Hencsey G, White B, eds. Proc. of the Int'l Conf. on World Wide Web (WWW 2003). Budapest: ACM Press, 2003. 187−196.

[20] Lu Y, He H, Zhao H, Meng W, Yu CT. Annotating structured data of the deep Web. In: Chirkova R, Oria V, eds. Proc. of the Int'l Conf. on Data Engineering (ICDE 2007). Istanbul: IEEE Computer Society, 2007. 376−385.

[21] Mittal N, Govil MC, Nayak R, Jain N. Reconstruction of Web forms for efficient Web search. In: Lobiyal D, Vidyarthi D, eds. Proc. of the Int'l Conf. on Methods and Models in Computer Science (ICM2CS 2009). New Delhi: IEEE Press, 2009. 1−5.

[22] Liu W, Yan H, Xiao J. Automatically mining review records from forum Web sites. In: Li M, Liang Q, Wang L, Song Y, eds. Proc. of the Int'l Conf. on Fuzzy Systems and Knowledge Discovery (FSKD 2010). Yantai: IEEE Press, 2010. 2450−2455.

刘伟(1976－),男,山东聊城人,博士,助理研究员,主要研究领域为 Web 数据抽取,Deep Web 数据集成.

严华梁(1985－),男,硕士,主要研究领域为 Web 数据抽取.

肖建国(1957－),男,教授,CCF 会员,主要研究领域为文本挖掘,图像处理.

曾建勋(1965－),男,研究馆员,主要研究领域为信息检索.