逆序解析DOM树及网页正文信息提取

张瑞雪 宋明秋 公衍磊
(大连理工大学系统工程研究所 大连 116023)

摘 要 一般地,从HTML网页中提取正文信息,应先将HTML网页解析成DOM树,然后遍历DOM树,依据目标信息在DOM树中的分布规律,将信息从DOM树中提取。这种传统方法将解析DOM树和从DOM树中提取信息看成两个独立的过程,制约了提取信息的速度。事实上,在准确提取目标信息的过程中,独立解析整个DOM树是没有必要的。在此,提出了逆序解析DOM树算法,并结合DOM树相似理论和传统的顺序解析算法,从部分目标信息开始分别向后顺序和向前逆序解析DOM树,同时定位并获取其他目标信息。利用该方法提取网页正文信息,一方面只需解析部分DOM树,从而减少了解析树结构花费的时间,另一方面不需要遍历整个DOM树查找目标信息,从而节省了查找时间,大大提高了信息提取速度。最后,通过实验证实了该方法的优越性。

关键词 DOM树,网页正文提取,结构相似性,逆序解析

Parsing DOM Tree Reversely and Extracting Web Main Page Information

ZHANG Rui-xue SONG Ming-qiu GONG Yan-lei
(Institute of System Engineering, Dalian University of Technology, Dalian 116023, China)

Abstract To extract main content from HTML Web page, generally, we should parse HTML, visit the whole DOM tree, and extract the data from the tree by distribution. However, this method separates the two processes of parsing and extracting and therefore restricts the speed. Actually, parsing the whole DOM tree is unnecessary. Here we supposed the algorithm of parsing DOM tree by reverse order. Then combining with the theory of DOM similarity and the traditional method of parsing DOM we parsed DOM tree with both normal order and reverse order, and at the same time we fixed the positions of other targots and got them. On the one hand, this method only parses part of DOM tree, so it reduces the time cost by parsing. On the other hand, we do not have to visit the whole tree to search the target information, as a result, it saves the searching time. Overall, this method improves the speed much. At the end of this paper, we gave the proof on the superiority of this method.

Keywords DOM tree, Web content extracting, Structural similarity, Parsing reversely

1 引言

随着Web信息资源的爆炸式增长,越来越多的网页信息呈现给用户。然而从大量的数据中筛选出用户感兴趣的信息就成了一个富有挑战性的课题。信息服务商使用网络爬虫抓取互联网中的网页,然后分析和整理网页中的信息内容,重新呈献给用户。在这个过程中,提取网页正文信息是一个不可或缺的环节。此外,网页正文还是搜索引擎、信息重构、网页分类的主要信息来源,准确并快速地提取正文信息为后继的数据挖掘和加工提供了数据基础。在传统的方法中,我们可以通过正则表达式直接提取网页内容,虽然这种方法可以快速提取网页的主要信息,但也容纳了太多的噪声信息。为精确地提取正文信息,有人提出了基于DOM树结构的页面分析方法。首先,他们通过标签之间的嵌套关系先从HTML网页中解析出DOM树结构,然后依据正文信息在DOM树结构中的分布规律确定正文的位置,包括〈table〉块分析[1]、中文与超链接比例分析[2]、节点统计分析[3]以及链路分析[4]等各种正文定位方法,最后将其提取。本文提取网页正文的方法也是建立在正文信息以块的形式聚集的分布规律之上的,另外,为消除正文中夹杂的噪音信息,本文还假定正文信息以相近的链路深度聚集[4]。由于正文信息是聚集在一起的,并在DOM树的同一个子树中,因此,若能确定任意一块正文信息,就能根据正文信息块之间的相似性从它最近的信息块中提取其他正文信息[5,6],并不需要访问整个DOM树,甚至不用解析整个DOM树。为此,本文首先提出了逆序解析DOM算法。与传统的DOM树解析算法不同,逆序解析算法是依据给定标记序列从最后一个结束标记开始依次向前解析并不断补全缺省标记的过程。在实际应用中,本文将传统解析算法和逆序解析算法相结合,以达到从HTML网页任意一处解析DOM树的目的。

为了提高算法的效率和准确率,本文的实验在解析DOM树之前对网页进行了清洗,即将不必要的注释信息、脚本信息

到稿日期:2010-05-29 返修日期:2010-07-26 本文受国家自然科学基金项目(70671016)资助。
张瑞雪(1987—),男,硕士生,主要研究方向为数据挖掘,E-mail:zhangruixue08@yahoo.cn;宋明秋(1967—),女,副教授,主要研究方向为数据挖掘、信息安全等。

等清洗掉以提高效率和准确性。

2 逆序解析 DOM 树算法

网页最常用的标记语言是 HTML 语言，它是基于标准通用标记语言(SGML)的一个庞大的文档处理系统。SGML 的基本思想是通过描述标记来提供描述文档结构的附加信息。HTML 利用 SGML 定义了一些标记，如〈div〉，〈table〉等，用于描述文本的显示方式，并对这些标记的使用都做了格式定义，对实体符号的显示和标记元素的结构也做了规范[7]。然而，HTML 还是一种非常自由的半结构化语言，它在限制文档格式的同时也容纳了太多的错误，例如元素标记的不完整、属性混乱、结束标记的缺失等。这些错误都给网页信息的提取带来了不便。文献[7,8]使用了传统的方法对 HTML 网页进行 DOM 树解析，并对标记配对方法进行了详细的阐述。本文借鉴了该配对规则，按照倒序从后向前依次进行标记识别，同时构建树节点，并根据配对规则确定节点的相对位置。

2.1 标记的识别

标记的识别过程是一个简单的字符匹配过程，从任意一个 HTML 文档末尾的右尖括号(〉)开始，每识别一个左尖括号(〈)，就记录一个标记名称，并提取标记属性信息；每识别一个右尖括号，查看其与上一个左尖括号之间的内容，若存在非空信息，则记录一个文本标记(〈text〉)，并提取该非空信息作为〈text〉元素的数据信息。

2.2 待配对结束标记栈

本文使用栈结构来存储待配对的标记，由于是逆序解析，栈中存储的待配对标记均为结束标记。在识别 HTML 文档最后一个结束标记(〈/html〉)时，将其压入待配对结束标记栈，此时栈非空。在解析整个文档过程中，根据判断识别的标记是否为开始或结束标记来删除或添加栈元素，直到识别文档第一个开始标记(〈html〉)时，将其与栈中元素(〈/html〉)配对并将栈清空，此时，HTML 文档解析结束。

2.3 配对与构造树节点

在预期结果中，每个 DOM 树节点都对应一对标记，而在 HTML 规范中，每个标记对至少都有一个开始标记，即每个结束标记至少对应着一个树节点。所以，在逆序解析 DOM 树的过程中，每遇到一个结束标记则至少要构建一个树节点，并将其添加成当前节点的第一个孩子节点。而当识别到一个开始标记时，就需要进行配对。具体配对算法如图 1 所示。

```
Algorithm: Match(CurTag, EndTagStack, ActiveNode)
{
//CurTag 是当前标记，EndTagStack 是结束标记栈，
//ActiveNode 为当前节点
if(CurTag 是文本信息或者独立标记){
  创建以文本信息为内容的叶子节点；
  新叶子节点添作为 ActiveNode 的第一个孩子；
  返回；}
if( CurTag 是一个结束标记){
  创建以当前标记为名的新节点；
  新节点添作为 ActiveNode 的第一个孩子；
  设新节点为 ActiveNode；
  返回；}
取 EndTagStack 栈顶元素 TopTag；
if( CurTag 与 TopTag 可以配对){
  设 ActiveNode 为 ActiveNode 的双亲节点；
  删除 EndTagStack 的栈顶元素；
  返回；}
if(ActiveNode 有孩子节点而且 CurTag 不能被
      ActiveNode 的第一个孩子节点所包含){
  创建以 ActiveNode 为名的新节点；
  新节点添作为 ActiveNode 的第一个孩子；
  返回；}
创建以当前标记为名的新节点；
设 ActiveNode 为 ActiveNode 的双亲节点；
新节点添作为当前节点的第一个孩子；
返回；
}
```

图 1 标记配对算法

以标记序列〈html〉〈A〉〈B〉〈C〉〈D〉〈/D〉〈E〉〈F〉〈/F〉〈/B〉〈G〉〈/G〉〈/A〉〈/html〉为例，其中标记〈E〉可嵌套〈F〉，标记〈C〉不能嵌套〈D〉。按照逆序解析算法，解析 DOM 树结构过程大致为图 2 所示，从左到右依次为逆序读取标记后解析的结果。

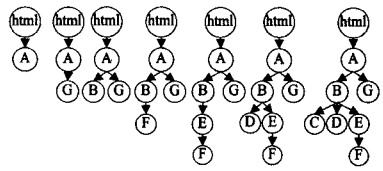

图 2 逆序解析 DOM 树过程

3 基于 DOM 树结构的正文信息提取

无论在视觉上还是 DOM 内部结构上，网页信息具有稳定的分布特征，即网页的同类信息一般都聚集在一起，并具有相似的视觉位置和逻辑位置。而正文信息有数据量大的特点，根据数据长度我们就可以确定部分正文信息。然后根据信息位置的相似性又可确定其他正文信息。

3.1 正文信息在网页中的分布特征

一般地，网页可分为导航型网页和内容型网页，其中导航型网页主要由分布均匀的超链接块组成。而内容型网页又可分为导航部分和正文部分，其中正文部分主要由大量的非链接内容聚集而成，同时又可能夹杂着少许链接内容。另外，正文信息在 DOM 树结构中一般具有相似的位置特征，它们具有相近的深度、相似的链路以及它们的父节点所对应的树具有相似的结构。

3.2 树结构的相似性度量

相似的内容分布在相似的树结构中，同样，在同一个网页中，如果两个树相似，那么它们也会有相似的内容。如果我们根据正文的特征能确定其中一部分正文信息，那么就可以再根据树结构的相似性确定其它部分正文信息。

关于 DOM 树结构的相似性度量问题，文献[5,6]分别给出了系统的算法，在此基础上，为提高计算速度，本文做了整合与简化，具体算法如图 3 所示。

```
Algorithm: Sim(NodeA, NodeB)
{
// NodeA, NodeB 分别为两树的根节点
if (NodeA 为空或 NodeB 为空)
  Return 0;
if (NodeA 和 NodeB 的标记名称不同)
```

```
    Return 0;
simscore=2.0;
Foreach cn in NodeA's children
    simscore=simscore+ Max_{j=1~n} {sim(cn,BC[j])}
    //BC[j]是 NodeB 的第 j 孩子节点
Foreach cn in NodeB's children
    simscore=simscore+ Max_{i=1~m} {sim(cn,AC[i])}
    //AC[i]是 NodeA 的第 i 孩子节点
Return simscore/(size1+size2)
//size1,size2 分别为两树的节点总数
}
```

图 3　树结构相似性度量算法

根据以上算法，根节点不同的两棵树的相似度为 0，树与本身的相似度为 1。然而判断两棵树是否相似没有统一的标准，一般是给定一个阈值，当相似度大于该阈值时可断定两棵树相似，否则不相似。而这个阈值的确定又是根据经验而定，不同的模型和不同的需求需要规定不同的阈值，一般在 0.7 至 0.9 之间，在本文后面的实验中，阈值设为 0.8。

3.3　基于 DOM 树结构的正文信息提取

要利用正文信息树结构的相似性，首先要准确定位部分正文信息。而正文信息具有数据量大的特点，我们可以根据信息的长度确定最为可能的两个正文块。由于正文信息的聚集性，从这些最为可能的正文块开始向上寻找第一个共同的根节点，则该根节点代表的子树下的大部分信息块也会是正文信息。但它不一定包括所有的正文信息，因为其兄弟子树中也有可能包含正文信息。此时，如果不存在与该根节点相似的兄弟节点，也就意味着正文信息不会出现在兄弟节点中，那么正文信息也不会出现在该根节点之外的其他所有节点中；否则将该根节点的父节点作为新的根节点，并与周边兄弟节点比较以确定其他正文信息。

以图 4 为例，提取正文信息的过程可归结为以下几个步骤：

Step 1　根据正文的长度可以判定节点〈text1〉和〈text2〉是正文信息；

Step 2　从最大的节点〈text2〉开始分别向前逆序和向后顺序解析 DOM 树，直到解析到节点〈text1〉为止，此时会解析出它们共同的根节点〈R〉，则可以确定〈R〉代表的子树里面的文本信息都是正文信息；

Step 3　继续逆序和顺序解析 DOM 树，解析出〈R〉的兄弟节点和双亲节点，并比较〈R〉节点与〈R〉节点的兄弟节点所代表的子树，由于〈R〉存在非常相似的兄弟子树，则可以肯定父节点〈T〉内包含的文本信息大部分为正文信息；

Step 4　继续解析，并用同样的方法比较〈T〉节点和〈T〉节点的兄弟节点所代表的子树，由于没有相似的子树，停止解析，并断定网页的正文信息很可能都在〈T〉节点代表的子树里；

Step 5　遍历节点〈T〉代表的子树，提取文本信息。

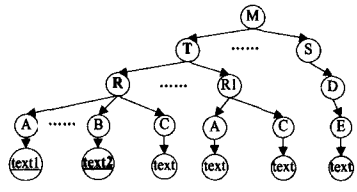

图 4　基于 DOM 树的正文信息提取

4　实验与分析

正文信息的提取大都是基于 DOM 树结构的，无论是使用节点统计方法还是链路比较方法，解析 DOM 树都是必需的。在此，本文通过实验分别实现链路相似性度量方法和基于逆序提取部分 DOM 树的子树相似性度量方法，并针对算法的时间复杂度进行统计和分析，以证明本文所提出的方法的优越性。其中链路相似性度量方法是基于全部 DOM 树结构的相似性度量算法，具有较高的准确度，算法简单并具有代表性。本文的实验均由 Python 语言实现。

本实验以国内 6 个著名的门户网站为背景，包括新浪(www.sina.com.cn)、网易(www.163.com)、搜狐(www.sohu.com)、腾讯(www.qq.com)、中国雅虎(www.cn.yahoo.com)和 TOM(www.tom.com)。分别从网站中随机地抽取 50 个网页，使用这两种不同的方法提取网页正文信息，其中 DOM 树相似度阈值设为 0.8。

图 5 统计的结果是针对每个网站提取 50 个网页所消耗的平均时间。从图中可以看出，本文提出的基于逆序解析 DOM 和子树相似性度量理论的正文提取算法比基于全 DOM 树的链路相似性度量方法具有更快的速度，提高了 1～2 倍。

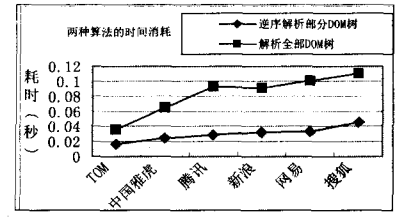

图 5　两种算法的时间消耗比较

图 6 列出了使用这两种不同算法提取正文信息需要解析的 DOM 树节点数目，从图中可以看出，本文提出的逆序解析部分 DOM 树的方法具有明显的优势，无论网页结构多么复杂，它几乎只需解析含有正文信息的子树。

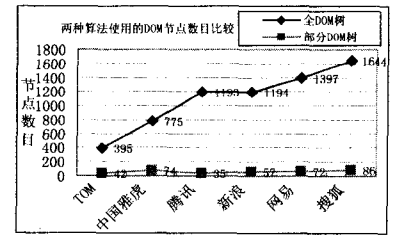

图 6　两种算法使用的节点数目比较

结束语　本文主要提出了逆序解析 DOM 树的算法以及以此为基础结合 DOM 相似性的正文提取算法。实验证明该方法比传统方法在提取时间和所需解析节点数目上具有明显的优势。逆序解析 DOM 算法也可以结合其他传统的正文信息提取方法，比如节点频率统计方法、块信息提取方法等，以提高提取正文信息的速度。除了提取正文信息外，逆序解析 DOM 树算法也可以应用在其他更复杂的半结构化数据提取上，如商品信息、用户评论。

参 考 文 献

[1] Lin Shian-hua, Ho Jan-ming. Discovering informative content blocks from web documents[C]// Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002:588-593

(下转第 225 页)

4.2 α_H 算法的局限性

在 α_H 算法步骤 5.2 中，如果 $|P_A|>1$（或 $|P_B|>1$），则工作流网中出现这种情况：两个并行结构处于选择结构的不同分支，而同时并行结构的 And-Split(And-Join)都是隐含任务，没有记录在日志中。如图 7 所示，此时 α_H 算法不能区分 $N5$ 中的两个 And-Join 隐含任务具体汇合了 B,D,E,F 中哪些任务，不能添加隐含任务。如果两个 And-Split(或 And-Join)中，只有一个是隐含任务，则 α_H 可以挖掘出来，如图 8 所示。

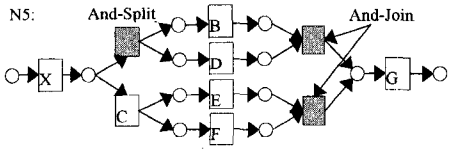

图 7　选择结构结束前出现两个隐含任务，α_H 算法不能挖掘

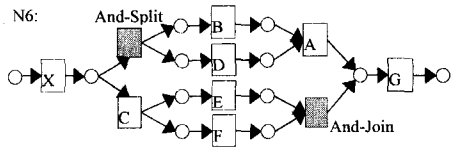

图 8　选择结构结束前只有一个隐含任务，α_H 算法可以挖掘

结束语　工作流挖掘中的 α 算法被证明能挖掘一大类工作流网，是工作流挖掘中较好的算法。但是，α 算法的缺点是不能挖掘隐含任务，而工作流网中需要一些隐含任务起路由作用。缺少隐含任务会破坏工作流网的合理性，导致死锁。本文探讨了基于 α 算法的隐含任务挖掘问题，归纳出了隐含任务可能出现的情形，提出了 α_H 算法，以在一定程度挖掘隐含任务。α_H 首先求出任务集合组对；然后根据组对中并行任务的出现位置，添加隐含任务；通过合并相同任务，去除冗余隐含任务操作来完善结果模型。本文采用 Java 语言编写了 α_H 算法程序，验证了算法的有效性，并指出了算法的不足之处。对于多个隐含任务分别处于选择结构的不同分支上的情形，α_H 不能将其挖掘出来，这将是进一步研究的工作。

参 考 文 献

[1] WFMC. Workflow Management Coalition Terminology and Glossary[Z]. Brussels, 1996

[2] Herbst J, Karagiannis D. An Inductive Ppproach to the Acquisition and Adaptation of Workflow Models[C]// Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The new Frontier for AI in Business. Stockholm, Sweden, 1999: 52-57

[3] Agrawal R, Gunopulos D, Leymann F. Mining process models from workflow logs[C]// Proceedings of the 6th. International Conference of Extending Database Technology. Valencia, Spain, 1998: 469-483

[4] Herbst J, Karagiannis D. Workflow mining with InWoLvE [J]. J. Computers in Industry, 2004, 53(3): 245-264

[5] 高昂，杨扬，王玥薇. 一种新的工作流频繁模式挖掘算法研究[J]. 计算机科学，2009，36(9)：231-233

[6] Maruster L, Weijters A, van der Aalst W, et al. Process mining: discovering direct successors in process logs[C]// Proceedings of the 5th. International Conference on Discovery Science. Lübeck, Germany, 2002: 364-373

[7] Maruster L, Weijters A, van der Aalst W. Genetic process mining: an experimental evaluation [J]. Data Mining and Knowledge Discovery, 2007, 14(2): 245-304

[8] Silva R, Zhang J, Shanahan J. Probabilistic Workflow Mining[C]// Proceedings of the 11th International Conference on Knowledge Discovery in Data Mining. Chicago, Illinois, USA: ACM, 2005: 275-284

[9] Schimm G. Mining exact models of concurrent workflows [J]. J. Computers in Industry, 2004, 53(3): 265-281

[10] 马慧，汤庸，吴凌坤. 流程增量挖掘中的模型更新方法 [J]. 计算机科学，2009，36(5)：154-157

[11] van der Aalst W, Weijters T, Maruster L. Workflow mining: Discovering process models from event logs [J]. IEEE Transactions on Knowledge and Data Engineering, 2004: 1128-1142

[12] Wen Lijie, van der Aalst W, Wang Jianmin, et al. A novel approach for process mining based on event types [J]. J. Intelligent Information Systems, 2009, 32(2): 163-190

[13] de Medeiros A K A, van Dongen B F, van der Aalst W M P, et al. Process mining: extending the α-algorithm to mine short loops[M]. BETA Working Paper Series, WP 113. Eindhoven: Eindhoven University of Technology, 2004

[14] Wen L, et al. Mining process models with non-free-choice constructs [J]. Data Mining and Knowledge Discovery, 2007, 15 (2): 145-180

[15] van der Aalst W. The application of Petri nets to workflow management [J]. J. Circuits Systems and Computers, 1998, 8(1): 21-66

[16] van der Aalst W. ProM Framework 5.2 [EB/OL]. http://prom.win.tue.nl/tools/prom/, 2009-10

[17] van der Aalst W, Weijters A. Process mining: a research agenda [J]. J. Computers in Industry, 2004, 53(3): 231-244

(上接第 215 页)

[2] Gupta S, Kaiser G, Neistadt D, et al. DOM based content extraction of HTML documents[C]// Proceedings of the 12th international conference on World Wide Web. 2003: 207-214

[3] 孙承杰，关毅. 基于统计的网页正文信息抽取方法的研究[J]. 中文信息学报，2004，18(5)：17-22

[4] 宋明秋，张瑞雪，吴新涛，等. 网页正文信息抽取新方法[J]. 大连理工大学学报，2009，49(4)：594-597

[5] 何昕，谢志鹏. 基于简单树匹配算法的 Web 页面结构相似性度量[J]. 计算机研究与发展，2007，44(z3)：1-6

[6] 潘有能. XML 文档自动聚类研究[J]. 情报学报，2006，25(2)：215-220

[7] 王志琪，王永成. HTML 文件的文本信息预处理技术[J]. 计算机工程，2006，32(5)：46-67

[8] 陈琼，苏文健. 基于网页结构树的 Web 信息抽取方法[J]. 计算机工程，2005，31(20)：54-150

[9] 常育红，姜哲，朱小燕. 基于标记树表示方法的页面结构分析[J]. 计算机工程与应用，2004，40(16)：129-132

[10] 朱明，王庆伟. 半结构化网页中多记录信息的自动抽取方法[J]. 计算机仿真，2005，22(12)：95-97