ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.2, February 2008, pp.275−290
DOI: 10.3724/SP.J.1001.2008.00275

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

基于页面Block的Web档案采集和存储*

宋 杰+, 王大玲, 鲍玉斌, 申德荣

(东北大学 信息科学与工程学院,辽宁 沈阳 100004)

Collecting and Storing Web Archive Based on Page Block

SONG Jie+, WANG Da-Ling, BAO Yu-Bin, SHEN De-Rong

(School of Information Science and Engineering, Northeastern University, Shenyang 110004, China)

+ Corresponding author: Phn: +86-24-83687776, E-mail: sy_songjie@163.net, http://www.neu.edu.cn

Song J, Wang DL, Bao YB, Shen DR. Collecting and storing Web archive based on page block. *Journal of Software*, 2008,19(2):275−290. http://www.jos.org.cn/1000-9825/19/275.htm

Abstract: In this paper, the page block based Web archive collecting and storing approach is proposed. The algorithms of layout-based page partition, extracting topic from block, version comparison and incremental storage implementation are introduced in detail. The prototype system is implemented and tested to verify the proposed approach. Theoretics and experiments show that, the proposed approach adapts the Web archive management well, and provides a valuable data resource to the Web archive based query, search, data mining and knowledge discovering applications.

Key words: Web archive; page partition, page block

摘 要: 提出了基于页面 Block 对 Web 页面的采集和存储方式,并详细表述了该方法如何完成基于布局页面分区、Block 主题的抽取、版本和差异的比较以及增量存储的方式.实现了一个 Web 归档原型系统,并对所提出的算法进行了详细的测试.理论和实验表明,所提出的基于页面 Block 的 Web 档案(Web archive)采集和存储方法能够很好地适应 Web 档案的管理方式,并对基于 Web 档案的查询、搜索、知识发现和数据挖掘等应用提供有利的数据资源.

关键词: Web 档案;页面分区;页块

中图法分类号: TP393 **文献标识码**: A

由于 Web 搜索的成功,大量 Web 用户都习惯于通过搜索引擎来检索信息,但是,这种方式仅能检索到当前的网页.像新闻、Blogs、论坛这样的 Web 信息在一年内就会消失[1].由于 Web 信息的流逝性,如何利用信息技术来搜集、保存和查询逝去的 Web 信息成为 Web 领域中一个崭新的课题.Web 历史或 Web 档案(Web archives)是一种基于 Web 的信息服务,它每隔一段时间就从 Internet 中抓取网页并以快照的形式保存起来,逐渐形成了一个 Web 历史网页的博物馆.这些网页不只是被收集和存储起来,还要进行分类和处理,以方便历史浏览和检索.

现有的 Web 档案的采集主要有两种方式.一种方式是网页发布商主动地汇缴给 Web 档案收集者.这种方式并不普遍,因为网页发布商部署网页的目的并不是为了归档和收藏,这样做会带来额外的成本且不能产生直接

* Supported by the National Natural Science Foundation of China under Grant Nos.60573090, 60673139 (国家自然科学基金)
Received 2007-08-31; Accepted 2007-10-19

效益.此外,在国家之间汇缴 Web 页面还存在法律问题[2].另一种方式就是 Web 档案收集者利用网络爬虫在 Internet 中采集.这种采集方式较为常用但难度较大.传统的收集方式是采集和保存网页的循环,即在一个固定的时间周期中采集所有的网页,并且按照 URL 地址分类保存起来.这种方法存在很大的研究空间:首先应该研究页面的转化和处理方式,是直接以单个页面的形式存储还是对页面进行分割和转化后再存储,处理方式取决于 Web 档案的使用方式;其次,该方法需要一个良好的版本比较机制,如果不进行版本比较,势必导致存储了大量时间不同但内容相同的网页,且版本比较算法必须成功地迅速检测到网页的部分变化,或者是哪些部分变化,是内容的变化,还是背景、格式或是广告信息的变化;最后是网页的存储方式的研究,由于 Web 历史信息量非常大,以国内的 Web InfoMall 为例,其规模以平均每天 150 万个网页的速度扩大,5 年已达到了 25 亿个网页(约 1 300GB)[3],因此,Web 页面存储应该支持增量的存储方式,即只存储变化的部分.

针对上述问题,本文提出了一种基于页面 Block 的 Web 档案采集和存储方法.我们分析了 Web 档案的使用方式,认为页面应该按照显示和主题分区,这样有利于 Web 档案搜索、问题回答(QA)或知识发现.该方法在使用网络爬虫获取页面以后,根据页面的布局特征将页面分为若干个显示和主题相对独立的矩形区域(Block)以及剔除这些区域剩下的页面框架(Layout),按照这些 Block 进行版本比较和存储并且只存储发生变化的 Block.本研究还实现了一个 Web 归档系统的原型,并用实验数据说明该方法的正确性和可用性.

本文第 1 节介绍研究背景.第 2 节阐述基于 Block 的页面采集和处理方式.第 3 节介绍原型系统的实现方式和测试结果.第 4 节对前人的相关工作进行简要分析.第 5 节总结全文并提出下一步的工作.

1 研究背景

目前,Web 归档系统大多选择整个 Web 页面作为最小的处理和存储单元,即依赖网页整体内容变化来检测新网页,并且以整个网页快照形式存储页面.这种方式已逐渐不能适应 Web 的快速发展,因此,我们提出把页面按照一定的算法分为若干个区域,把这些区域作为基本的存储和处理单元.提出该方法的理由如下:(1) 当今的 Web 页面更新迅速,页面内容丰富多样,很多页面的内容不是由同一主题的大量平文本而是由多个不相关主题的页面区域构成,并且页面的更新也分解成部分区域的更新,因此,以页面区域为最小单位来采集和存储 Web 档案更符合实际情况,粒度上明显优于以页面为最小单位的采集和存储.(2) 此外,为了方便用户浏览,Web 页面中加入了大量与内容无关的元素,比如导航菜单、输入区、装饰性的标题和图片或者站点相关信息,这些元素信息量较少且相对稳定.如果采用基于整个页面的 Web 档案采集和存储,则对这些内容无关的元素的处理和存储势必浪费大量的时间和空间,而如果基于页面区域进行采集和存储,就可以适当忽略这些稳定的且内容无关的区域,节省处理和存储代价.(3) Web 档案的意义在于强大的档案资料库的搜索功能.对于搜索引擎、QA 系统或知识发现系统,以 Web 网页作为独立语义单元的最大问题在于忽略了页面上存在着多个主题,当查询语句分散在各个主题上时,搜索精度就会显著下降.因此,按照页面区域来采集和存储 Web 档案,有利于为档案信息搜索和提取而创建主题相关的数据集,并且排除广告、导航等噪音数据,进而降低搜索难度、提高搜索质量.

网页布局是网页的一个重要的内部结构,从网页的布局我们可以分辨内容相关的信息和内容无关的噪音.我们根据网页布局把网页划分成若干个区域,在 Web 档案采集和存储过程中,检测网页整体布局的变化或者部分区域的变化,并把变化的部分保存起来,以达到存储 Web 档案的目的.为了方便阐述,本文从 Web Archive[4]网站选取了 java.net 网站的若干历史网页.图 1 中,左图是 2007 年 1 月 2 日该网站的某网页快照,右图是该网页 5 天后(同年 1 月 7 日)的网页快照,为了节省篇幅,在不影响算法说明的前提下,我们对页面作了简化处理.

直观上可以看出,这两个网页结构非常相似.简单分析页面可以看出,有很多区域与内容信息无关,对信息检索没有帮助,而且相对比较稳定.本页面主要的内容为虚线矩形区域,该区域又分为若干个主题.我们分析了大量 Web 历史页面后发现,每个页面总是存在一些频繁更新的区域,而剩下的区域则很少更新.在本例中,频繁更新的只有虚线矩形区域.

可以看出,在采集周期(5 天)内网页发生了变化,如果时间周期缩短为 2 天,则很有可能得到两个完全相同的页面.因此,本研究需要解决以下几个问题:(1) 如何对采集的网页自动进行版本比较以确定网页是否更新;

(2) 根据直观的页面分区我们发现,只有中央虚线矩形部分发生了变化,那么,如何以一种合理的方式分割页面并找到变化的部分;(3) 显然,为了减少存储开销,Web 归档系统可以仅存储发生变化的部分,其存储方式也是本文研究的问题之一.

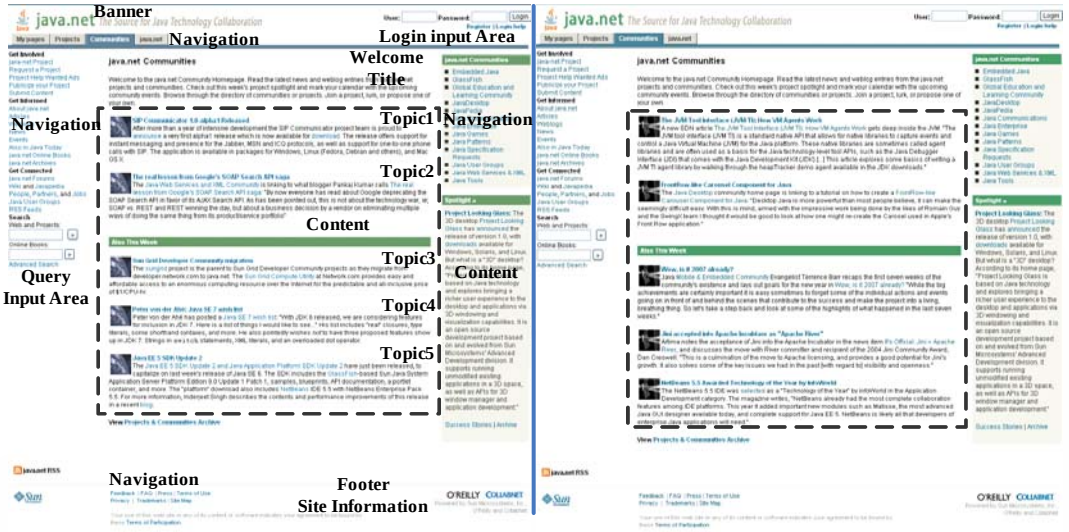

Fig.1 Two Web history snapshots as example

图 1 作为例子的两个历史网页快照

2 基于Block的页面采集

本节将描述 Web 页面的分区、主题生成、版本比较和存储算法.首先我们给出页面 Block 的相关定义.

定义 1(Block). Block 是页面中在内容和显示上独立的、闭合的矩形区域.Web 页面可以分割为若干个互不相交的 Block,我们把这个过程称为页面分区.

定义 2(Block 树). 我们递归定义一个 Block 可以由多个相互不重叠的 Block 组成,则图 2 所示的页面可以分解为图 3 所示的树形结构,称为 Block 树.其中,节点 Block 是整个页面,是最大粒度的 Block,而叶子节点则是最小粒度的 Block.

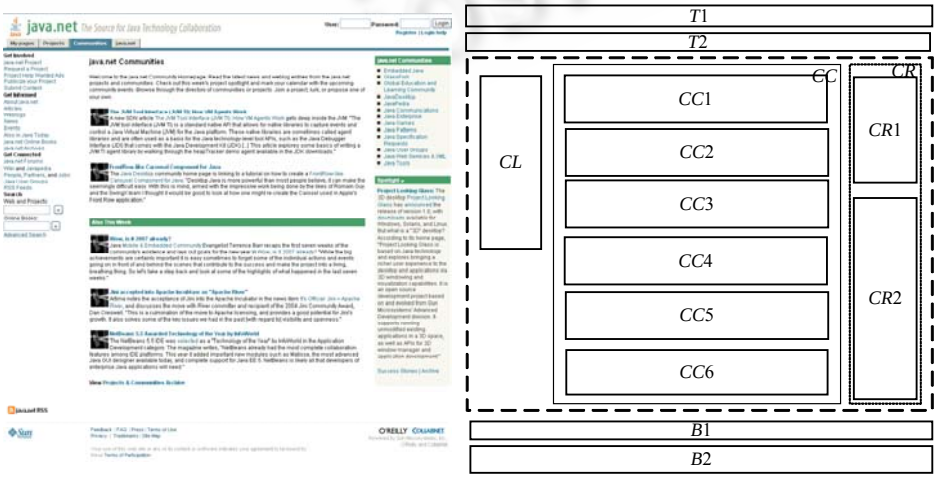

Fig.2 Block structure of an example page

图 2 示例页面的 Block 结构

定义 3(分区级别). 分区级别表示该分区算法产生的 Block 树的最大深度,用 Level 表示.由图 4 可以看出,在任意深度下,所有叶子节点均能组成页面的一种分区策略.因此,选择合适的分区级别有助于得到最理想的分区结果.

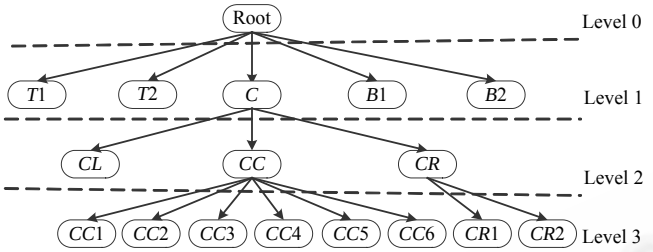

Fig.3 Block-Tree of an example page

图 3 示例页面的 Block 树

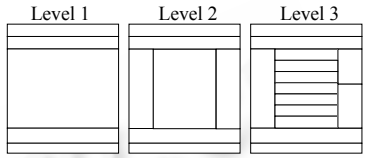

Fig.4 Partition result of different level

图 4 不同分区级别的分区结果

定义 4(Block 主题和内容). 设 content 为 Block 内容,即 Block 中存储的 HTML 文档,topic 为 Block 主题,即从 Block 内容中查找或抽取出的能够表示 Block 内容概要的一段文字,则 Block 可以定义成一个由主题、内容组成的二元组:

$$Block=\langle topic,content\rangle.$$

定义 5(Layout). 设 Layout 为网页的布局,即剔除了 Block 的网页框架,其中,Block 部分使用占位符填充.

Layout 是一个完整的网页,主要包括除了 Block 部分页面框架以外的像〈BODY〉这样的一些全局架构标签,或者是 Block 之间的分割部分.

定义 6(Version). 设 Version 为某个 URL 地址在某一个时间点的 Web 历史记录.Version 可以定义成一个由 Layout 和若干个 Block 以及元数据信息组成的三元组:

$$Version=\langle Layout,\{Block\},MetaData\rangle.$$

2.1 页面分区算法

页面分区算法的目标是把页面分割为一组不重叠的 Block,这种分割算法基于 DOM(document object model).一个 Web 页面经过解析后可以转换为 DOM[5].DOM 提供了树形结构的页面模型,因此,我们可以基于 DOM 来建立 Block 树[6,7].我们当然可以借助大量 HTML 标签来获取布局和位置信息,如〈P〉,〈TABLE〉,〈TR〉,〈HR〉,〈UL〉等.然而,由于 HTML 语法的灵活性,很多 Web 页面并不完全遵守 W3C 的 HTML 规范.此外,依据 DOM 的分区只能代表布局结构独立的区域而不能完全代表语义上独立的区域,比如,同一个父节点下面的两个子节点并不一定代表相同的主题.因此,分区算法不能完全依赖 DOM,还需要考虑其他一些因素.通常,显示上独立的区域一般表示相同的主题,Web 中提供了大量可见的元素来划分页面[8],例如字体、颜色、图像、空白,这些都是页面分区算法需要考虑的元素.页面分区的整个过程如图 5 所示.

首先,DOM 树和一些字体、颜色信息经抽取 Block 算法抽取出第 n 级 Block(n 初始化为 1),组成深度为 n 的 Block 树,同时保存 Layout.

然后,判断 n 是否满足给定的分区级别,如果小于该级别,则依次把 Block 树中的每个叶子节点 Block 的内容(content)作为新的 DOM,重新抽取 Block 并组装 n+1 级 Block 树,如果某个 Block 已经无法再分割,则忽略这个 Block.

如此反复,直到 Block 树达到给定分区级别为止.给定分区级别可以根据经验预先设置,也可以根据历史记录动态调整.

图 6 给出了图 1 中例子页面的 3 次分区算法迭代过程中每次生成的 Block 树,图中深颜色的节点表示不可再分的 Block,在迭代中应该忽略.

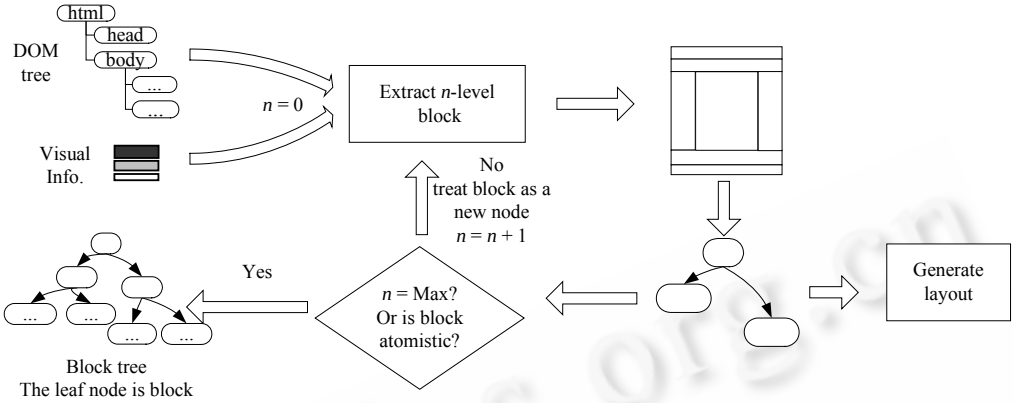

Fig.5 Block-Based page partition algorithm

图 5 基于 Block 的页面分区算法

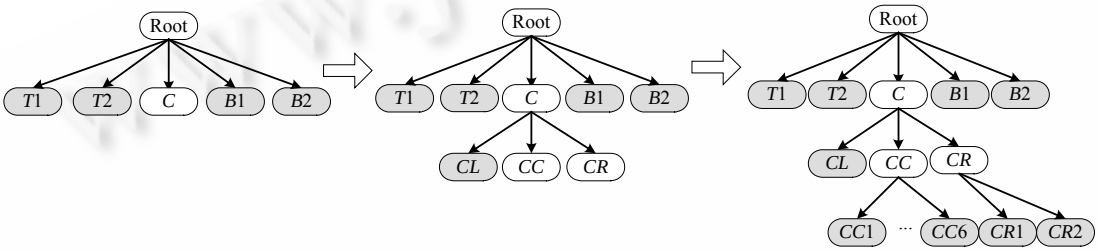

Fig.6 Block-Trees generated by three overlaps of page partition algorithm

图 6 页面分区算法 3 次迭代产生的 Block 树

抽取 *n* 级 Block 算法是一个递归的过程,其主算法为算法 1.在该算法中,*depth* 不同于分区级别,分区级别是 Block 树中最终的深度,*depth* 是当前节点在 DOM 数中的深度.*depth* 的最大值 Max*Depth* 参照分区级别设置,但二者之间没有必然联系.

算法 1. 抽取 Block.

Input: *pNode* is DOM, *depth* is depth of *pNode* in DOM tree;

Output: 抽取的 Block.

ExtractBlock(*pNode*,*depth*)

```
1  If depth>MaxDepth
2     Return;
3  End;
3  If IsBlock(pNode)==True
4     Save pNode as a Block into Block pool;
5  Else
6     For each child of pNode
7           ExtractBlock(child,depth+1);
8     End;
9  End;
```

Block 抽取算法的关键在于,遍历 DOM 树时,如何判断一个节点是否是一个 Block,即 IsBlock 算法.在描述 IsBlock 算法之前首先定义以下几个函数.设 *node* 为 DOM 树中的某一个节点,则:

- *node.properties* 表示取当前节点的属性,如 bgcolor,font,width 和 height 分别表示背景颜色、字体、宽度和高度.
- *node.tag* 表示当前节点的 HTML 标签.
- *node.type* 表示节点的类别.我们把 HTML 标签分为布局标签、格式标签和功能标签 3 类,布局标签多数与网页的布局有关,如⟨TABLE⟩,⟨TD⟩,⟨TR⟩,⟨HR⟩,⟨P⟩等;格式标签多与文字、图片等页面元素的格式有关,如⟨B⟩,⟨BIG⟩,⟨EM⟩,⟨FONT⟩,⟨I⟩,⟨STRONG⟩,⟨U⟩,⟨DIV⟩等;功能标签多数是显示特定的非文本元素,如⟨FORM⟩,⟨A⟩,⟨INPUT⟩,⟨IMG⟩,⟨BUTTON⟩等.

$$node.type=\begin{cases}\text{Schema}, & node.tag\in \text{布局标签}\\ \text{Format}, & node.tag\in \text{格式标签}\\ \text{Function}, & node.tag\in \text{功能标签}\end{cases}.$$

- *node.subNode* 表示当前节点的子节点集合,该集合的长度用 *node.subNode.length* 表示.

下面的 Rule 1~Rule 11 是判断一个节点是否是 Block 所依次遵循的法则.

Rule 1. 如果当前节点没有子节点或子节点大小均为 0,则该节点不是 Block.否则

Rule 2. 如果当前节点是 Function 节点,则该节点不是 Block.否则

Rule 3. 如果当前节点是 Format 节点,则该节点不是 Block.否则

Rule 4. 如果当前节点只有 1 个子节点,则该节点不是 Block.否则

Rule 5. 如果当前节点的尺寸小于标签的给定最小值,则该节点不是 Block.否则

Rule 6. 如果当前节点的颜色与子节点颜色不同,则该节点不是 Block.否则

Rule 7. 如果当前节点是 Schema 节点,且子节点中没有 Format 或 Function 节点,则该节点不是 Block.否则

Rule 8. 如果当前节点是 Schema 节点,且子节点只包含 Format 或 Function 节点,则该节点是 Block.否则

Rule 9. 如果当前节点是 Schema 节点,且子节点中的 Schema 节点递归满足 Rule 8 或 Rule 9,则该节点是 Block.否则

Rule 10. 如果当前节点的前一个兄弟节点是 Block,则该节点是 Block.否则

Rule 11. 该节点不是 Block.

根据 Rule 1~Rule 11,我们重新考虑图 2 所示的例子.如图 6 和图 7 所示,如果设分区级别为 3,那么,首先抽取深度为 1 的 Block 树,*T*1,*T*2,*C*,*B*1,*B*2 很容易被抽取出来.第 2 轮抽取深度为 2 的 Block 树,由于 *T*1,*T*2,*B*1,*B*2 都已经是最小粒度,无法再分割,所以考虑继续分割 *C*,*C* 的第 1 个子节点⟨TD⟩有 24 个⟨DIV⟩子标签,⟨DIV⟩子标签多是链接标签,⟨TD⟩符合 Rule 8,抽取名为 *CL* 的 Block.根据 Rule 10 很容易得到 *CC* 和 *CR* 这两个 Block.最后抽取级别为 3 的 Block.根据 Rule 2 和 Rule 3,可以判断 *CL* 无法进一步分割.*CC* 可以分割为 *CC*1~*CC*6,CR 可以分割为 *CR*1~*CR*6.

2.2 主题生成算法

为了便于对 Block 进行检索、分类和管理,我们提供了一种简单的方法为每一个 Block 生成一句主题文字.主题生成算法要求具有唯一性,即相同的内容抽取的主题相同,但主题相同对应的内容则不一定相同.通常可以采用两种算法生成主题:一种是通过字体、颜色和位置等信息查找主题;另一种是利用数据挖掘的方法抽取主题.在介绍这两种方法之前,首先对一些操作进行定义.

由 Block 定义可知,Block 中包含的 content 也是一段 HTML 文档,包含着文字和格式信息,我们不难把 content 中每个含有文字的 Format 标签抽取出来.如果用 *T* 表示这些 Format 标签的集合,设有 *n* 个 Format 标签,每个标签 $t_i(1\le i\le n)$又可以分为格式信息 $format_i$ 和文字信息 $text_i$,设 *Format* 是所有 *format* 的集合,*Text* 是所有 *text* 的集合,那么,集合 *T* 可以表示为 *Format* 集合和 *Text* 集合组成的二元组.

$$T=\{t_i\mid 1\le i\le n\}=\langle Format,Text\rangle=\langle\{t_i.format\},\{t_i.text\}\rangle=\langle\{format_i\},\{text_i\}\rangle,\ 1\le i\le n.$$

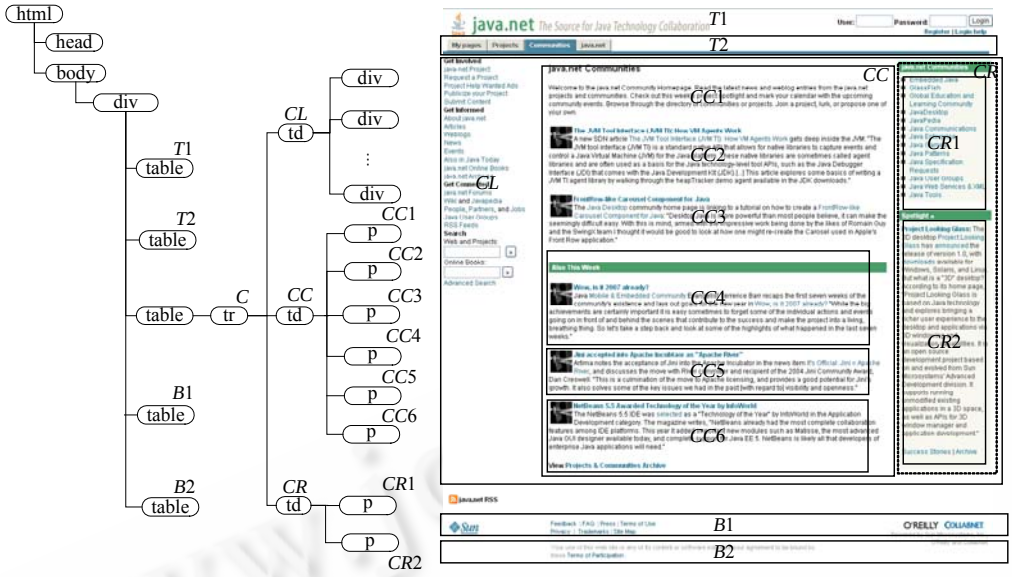

Fig.7 DOM structure and corresponding Block structure of example page

图 7 示例页面的 DOM 结构和对应的 Block 结构

定义 7(长度函数). 以参数 $text_i$ 为例,$length(text_i)$返回 $text_i$ 中单词的个数,或者以参数 *Text* 为例:

$$length(Text)=\sum_{i=1}^{n} length(text_i)\ .$$

1) 查找 Topic

在大多数 Web 页的 Block 中已经存在一个主题,它以较大的字体、显著的位置或者不同的颜色来显示.这种情况下,主题生成算法的重点就是如何找到一个已存在的主题.主题 topic 的查找步骤如下:

步骤 1: 遍历集合 *Text* 并找到一段文字 $text_i$,其对应的 $format_i$ 是集合 *Format* 中字体最大或者位置独立且靠前的,或者颜色与其他不同的;

步骤 2: 如果θ是主题中的单词数目最大限定值,那么,

$$topic=\begin{cases} text_i, & length(text_i)\le\theta \\ trim(text_i), & length(text_i)>\theta \end{cases}.$$

这里,$trim(text_i)$是一个能够将一个段落缩减到只含θ个单词的函数.算法 2 给出了 $trim(text_i)$函数:

算法 2. 裁减内容.

Input: $text_i$ as a paragraph, a stoplist file *S*, word number threshold θ.

Output: A set *TW* consisting of all words of topic phrase.

$trim(text_i)$

```
1  Delete stopwords from text_i according to stoplist;
2  TW={};
3  For j=1 to θ Step 1
4     word_j=word of text_i which index is j;
5     If word_j∉TW
6        Put word_j into TW;
7     End;
8  End;
9  Return TW;
```

在算法 2 中,第 1 行采用 stoplist[9]进行了预处理.

2) 抽取主题

在一些 Block 中,有些主题没有用特殊的格式突显出来,甚至根本没有主题.在这种情况下,主题生成算法的重点是如何从现有的文字中抽取出一个主题.许多 Web 挖掘研究提供了抽取 Web 页面主题的算法,这些算法可能适用于含有大量文字的 Block.但是,对于文字量很少的 Block 来说,数据挖掘算法很难有意义.抽取主题的算法如下:

设δ为一个区间,表示适合 Web 挖掘算法的最小单词数,$text_i \in Text(1 \le i \le n)$,则

$$topic = \begin{cases} trim(text_i), & length(text_i) < \delta \\ extract(text_i), & length(text_i) \ge \delta \end{cases}.$$

算法 3 描述了抽取主题的过程.

算法 3. 抽取主题.

Input: $text_i$ as a paragraph, a stoplist file S, a frequency threshold ε.

Output: A set TW consisting of all words of topic phrase.

ExtractTopic($text_i$)

1) Delete stopwords from $text_i$ according to stoplist;
2) Process $text_i$ using stemming algorithm;
3) TW={};
4) **For** j=1 **to** θ Step 1
5) $word_j$=word of $text_i$ which index is j;
6) If $word_j \notin T$ and $tf(text_i, word_j) > \varepsilon$
7) **Put** $word_j$ into TW;
8) **End**;
9) **End**;
10) **Return** TW;

在算法 3 中,1~2 行是用 stoplist[9]和 stemming[10]对 *text* 进行预处理.这里,*Text* 是整个 Block 中包含的文字,$text_i$ 是每个 Block 中每个标签包含的文字($text_i \in Text$),$word_j$ 是 $text_i$ 文字中的一个单词,$tf(text_i, word_j)$是 $word_j$ 在文档 $text_i$ 出现的绝对频率[11].

2.3 页面比较算法

页面比较算法用来比较新采集的页面和上一个版本的差异.当采集到一个新版本的页面快照(version)时,需要对其进行分区处理,获得其所有 Block 和 Layout,然后与最近一次历史 Version 进行版本比较,找出差异的部分存储,如果新、旧页面完全一致,则表示页面没有更新,本次采集无效.

设α和β为 Web 历史采集的两个连续的时间点($\alpha < \beta$),在此时间点下,同一 URL 采集的 Version 分别是 v^α和 v^β,对应的 Layout 为 l^α和 l^β,对应的 Block 集合为 bs^α和 bs^β,$bs^\alpha.size$ 表示 bs^α中包含的 Block 个数.同理,$bs^\beta.size$ 表示 bs^β中包含的 Block 个数. b_i^α 表示 bs^α集合中的第 i 个 Block,与之对应的是 b_i^β. *ByteEqual*(x,y)函数返回 True,表示 x 和 y 两个元素按字节比较相等.

v^α=v^β的充要条件是

$$\begin{cases} v^\alpha = v^\beta \Leftrightarrow ByteEqual(l^\alpha, l^\beta) \cup bs^\alpha = bs^\beta \\ bs^\alpha = bs^\beta \Leftrightarrow bs^\alpha.size = bs^\beta.size \cup (\forall b_i^\alpha \in bs^\alpha, \forall b_i^\beta \in bs^\beta, b_i^\alpha = b_i^\beta) \\ b_i^\alpha = b_i^\beta \Leftrightarrow ByteEqual(b_i^\alpha.content, b_i^\beta.content) = \text{True} \\ b_i^\alpha = b_i^\beta \Rightarrow b_i^\alpha.topic = b_i^\beta.topic \end{cases} \quad (1)$$

尽管 Block 的 topic 相等不是 Block 相等的充要条件,但一般情况下,当网页内容发生实质性的变化时均会

反映在 topic 上.不同文字生成相同 topic 的概率较低,所以,我们可以近似地使用公式(1)来判断 Block 是否相等.

2.4 增量存储算法

完成版本比较以后,就要考虑采集的网页如何存储.我们借鉴了 Delta 存储算法的思想,以 Block 为存储单元进行 Web 档案的存储.Delta 存储是一种仅存储差异的文档存储技术[12],在版本控制系统(如 CVS[13])中常常使用 Delta 存储技术来减少磁盘开销.在版本控制系统中,每个文件都以一个唯一的文件路径作为根,并以一个派生的树形结构来记录每个版本分支.对于 Web 档案的存储也可以采用 Delta 存储的方式来实现,但不是以字节为粒度考虑差异,而是按 Block 粒度考虑差异;不是以唯一的文件路径作为根,而是以唯一的 URL 作为根.基于 Block 的增量存储方式还集成了高性能的目录[14]来访问元数据.

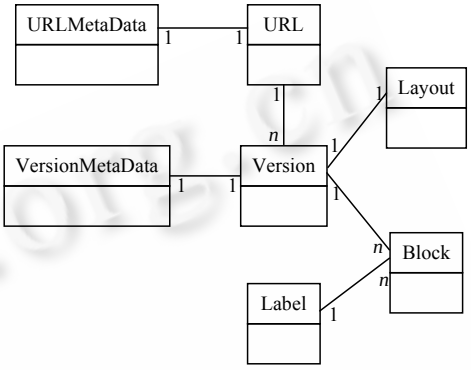

Fig.8 Schema of Web archive repository

图 8 Web 档案存储的模式

图 8 为 Web 档案存储的结构示意图,其中每个部分的含义如下:

- URL:表示 Web 档案收集的 URL 信息.
- URLMetaData:URL 的元数据,一条 URL 对应一个 URLMetaData.
- Version:代表某个 URL 地址某一个时间点的 Web 档案记录,可以表现为若干个 Block 的集合,也可以把这些 Block 填充到 Layout 中得到一个完整的页面.
- VersionMetaData:Version 的元数据,描述每条 Web 历史信息(version),与 Version 一一对应.
- Block:采集的页面快照经分区算法处理后得到的若干 Block.Version 与 Block 是一对多的关系,同一个 URL 下的不同 Version 对应的所有 Block 不会重复,但是,Block 存储会因不同的 URL 中出现相似的内容而重复,这种重复可以通过文献[15]提出的 Volumes 技术来解决.
- Layout:由页面分区算法可知,Block 是页面的一部分,如果需要还原整个页面,仅有一组 Block 是不够的,还应该保存网页的整体框架.Layout 代表了一种网页的布局,存储的是除了 Block 部分以外的页面框架.由于网页的 Layout 很少发生变化,大量 Version 都对应一个 Layout.
- Label:Label 是 Block 的分类信息,表明了 Block 中内容的类别,为以后的历史搜索、知识发现和数据挖掘提供信息.

以第 2.3 节中α和β的两个连续的时间点($\alpha<\beta$)为例,算法 4 描述了 v^{β}的存储处理算法.

算法 4. 存储采集的页面.

Store()

```
1   Bool flag=false;
2   If ByteEqual(l^α,l^β)==True
3     For each b^α,b^β in bs^α and bs^β,
4       If b^α!=b^β
5           Save b^β and b^β's Label
6           flag=true;
7       End
8     End
9   End
10  Else
11    Save l^β;
12    flag=true;
```

```
13    For each b^β in bs^β,
14        Save b^β and b^β's Label
15    End
16  End
17  If flag==true
18    Save v^β and v^β's VersionMataData
19    construct all relationships
20  End
```

算法首先判断 Layout 是否相等:如果相等,则存储差异的 Block;如果不等,则存储全部的 Layout;然后,存储分类信息和元数据;最后,建立各个部分之间的关系.对于图 2 给出的例子,我们第 2 次采集只需要存储 *CC*2~*CC*6 这 5 个 Block.

2.5 总体流程

本文第 2 节介绍了基于页面 Block 的 Web 档案采集和存储的核心算法,当网络爬虫采集到页面以后,首先根据当前的 URL 地址查找最近的一个 Version:如果不存在该 URL 地址,说明这是一个新的页面,则按照系统默认的分区级别抽取 Block 和 Layout,给每个 Block 抽取主题,存储图 8 中表述的每个结构;如果找到当前 URL 最近的 Version 记录,则根据该 Version 的分区级别抽取当前页面的 Block 和 Layout,然后根据算法 4 描述的过程存储采集的网页.

Web 归档系统还可以利用大量的历史信息来指导自身,比如利用某个网页的历史分区方式和经常变化的 Block 来指导网页分区.如图 2 给出的例子,如果网页频繁变化的是 *CC*2~*CC*6 这 5 个 Block,我们可以设置分区级别为 2 级,这样,*CC*2 和 *CC*6 就合为一个 Block,简化了版本比较(但是,这种合并只是从采集存储角度来进行,并非一定有利于 Web 档案的应用).此外,网站的更新频率一般是有规律的,根据文献[16]提出的思想,我们可以通过同一 URL 的各个 Version 之间的间隔来推断最佳的采集频率,减少采集不到网页变化的可能性.

3 原型系统和实验分析

为了证明本文提出的基于页面 Block 的 Web 档案管理方法的正确性和有效性,我们设计了一个原型系统 Block-based Web Archive Prototype(BWAP),并且对其进行了详细的测试.首先,我们采用软件测试的方法对系统进行测试;依据现有的知识,我们没有发现同类工作的运行性能数据,因此没有进行性能比较,而只是在测试中对系统性能作了定性的衡量.相对于性能因素而言,本研究更关心的是系统的精确度,因此,我们组织了一个实验对系统的精确度加以验证.

3.1 原型系统结构

图 9 说明了 BWAP 的主要构件以及它们之间的关系,图中箭头的方向是主要数据的流动方向,采用的例子是本文第 2 节反复提及的示例.上标 α 表示当前采集的页面版本,β 表示该 URL 的 Web 档案信息中最近的一个版本.由图中可以看出,采集的页面最终保存 *CC*2~*CC*6 这 5 个 Block 和相关元数据到 Web 档案库中.图中相关组件的功能如下:

- Page Cache:页面缓存用来缓存 Crawler 获取的页面,使采集和页面处理可以异步进行,降低耦合度.
- Thread Controller:线程控制器,可以多线程地同时对多个页面执行处理和存储.
- Page Processor:页面处理器,用来分割页面,获取 Blocks 和 Layout.
- Topic Generator:主题生成器,为每个 Block 查找或抽取主题.
- MetaData Generator:元数据生成器,根据用户设置的元数据生成各种元数据.
- Store Agent:存储代理,完成 Layout,Blocks 和元数据的存储工作.
- Version Manager:版本管理器,完成基于 Block 的页面版本比较(见第 2.3 节).

- Level Manager:分区级别管理器,能够选择、查找历史或适当地调整每个网页的分区级别.
- Query& Storage Proxy:Web 存档库的查询和存储代理.

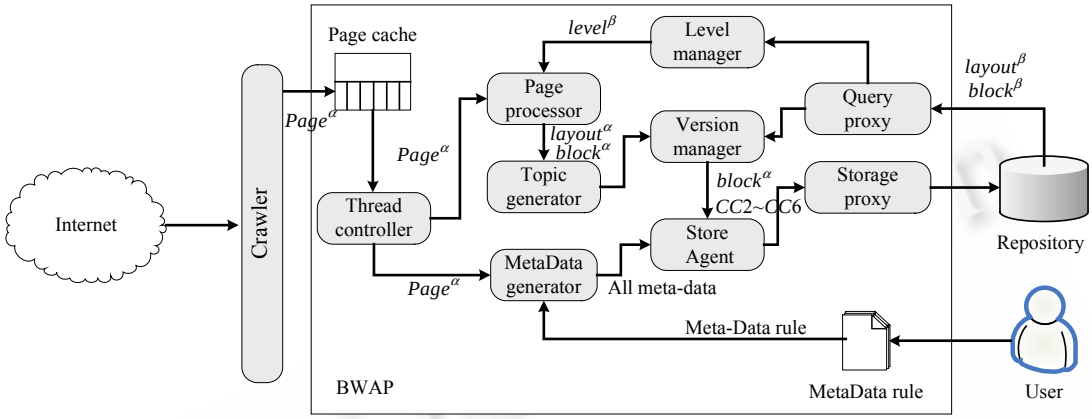

Fig.9 Overview of BWAP

图 9 BWAP 总体结构

当 Crawler 获取了页面并存入 Page Cache 以后,Thread Controller 启动一个线程来处理 $page^{\alpha}$.首先处理 $page^{\alpha}$的是 Page Processor,这时,Level Manager 根据 $page^{\alpha}$的 URL,通过 Query Proxy 在 Repository 中查找合适的分区级别;然后,Page Processor 根据分区级别把 $page^{\alpha}$分割成 $layout^{\alpha}$和 $blocks^{\alpha}$;$blocks^{\alpha}$通过 Topic Generator 生成主题后,与 $layout^{\alpha}$一同转到 Version Manage 模块进行版本比较;Version Manage 通过 Query Proxy 在 Repository 中找到 $layout^{\beta}$和 $blocks^{\beta}$后,按照第 2.3 节中的比较算法进行版本比较;最后,MetaData Generator 根据管理员输入的元数据生成规则和 $page^{\alpha}$相关信息生成相关元数据,并与版本比较结果(有差异的 Blocks)一同由 Store Agent 通过 Storage Proxy 存入 Repository.整个系统的时序图如图 10 所示.

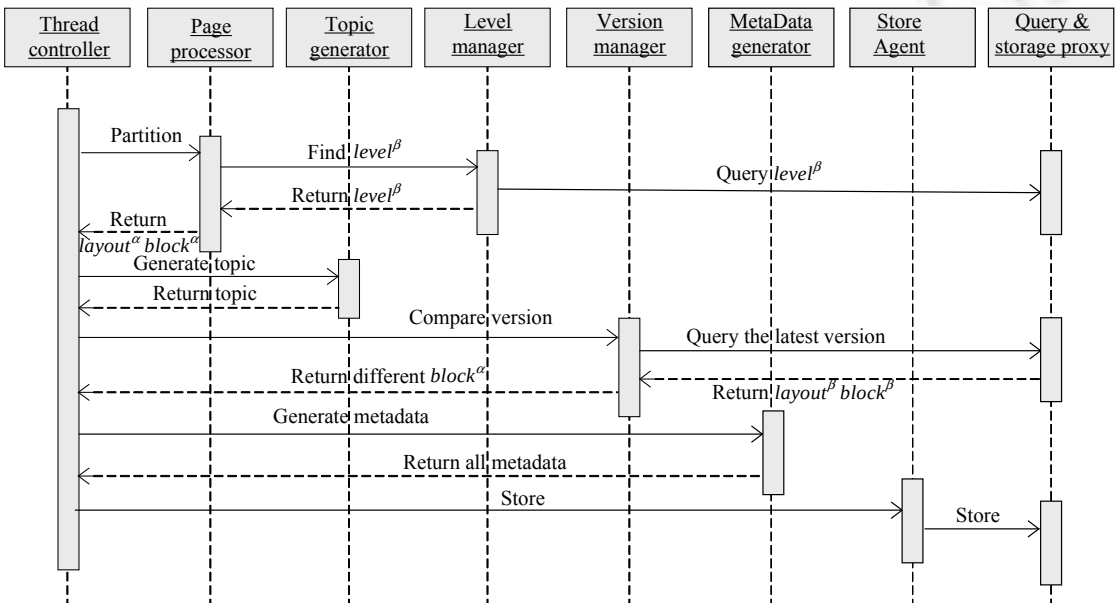

Fig.10 Sequence diagram of BWAP

图 10 BWAP 时序图

3.2 系统测试

为了说明 BWAP 的功能完整性,我们为 BWAP 设计了详尽的测试.BWAP 通过了很多严格的测试用例,本节列出了部分测试用例和测试结果,它们涵盖以下几个方面:

- 功能测试:保证 BWAP 可以在各种输入数据集下完成既定的功能逻辑;
- 白盒测试:对 BWAP 中每一个构件以及构件内部的特定代码块进行测试;
- 性能测试:在各种条件下,对系统处理和响应时间进行测试;
- 集成测试:主要是测试 BWAP 内部构件的集成性以及与其他构件或框架的兼容性.

表 1 列出了部分测试用例和测试结果.

Table 1 Test cases and test results

表 1 测试用例和测试结果

Category	Test case	Result
Functional testing	Test the system can deal with the pages whose layouts change frequently.	Success
	Test the system can deal with the pages which have tiny change.	Success
	Test the system can deal with the pages whose structures are abnormally complex.	Success
	Test the system can deal with the pages whose structures are abnormally simple.	Success
	Test the system can deal with the pages which include many scripts.	Success
	Test the system can deal with the pages which include text mostly.	Failure
	Test the system can deal with the pages which include a lot of blanks.	Success
	Test the system can deal with the fragmentary pages.	Success
	Test the system can deal with the pages which contain lots of images.	Success
	Test the system can deal with the pages by great partition level.	Success
	Test the system can deal with the pages by small partition level.	Success
	Test the system can deal with the pages by unstable partition level.	Success
White box testing	Walk through the code and review it for such qualities as logic, adherence to development standards and best practices, and readable comments.	Done
	Run the code against a code-coverage tool to make sure that all the available code is being tested.	Done
	Perform memory profiling of the code to make sure that objects are correctly released and garbage is collected.	Done
Performance testing	Test to see if system processing pages takes much prolonged much more time than crawler collection pages.	No
	Test to see if storing takes much prolonged time than processing.	No
	Test the performance of each component to find bottleneck.	None
	Test performance by dealing with several thousands pages one time.	Good
	Test performance when process and store pages which have many images.	Better
	Test performance when process and store large pages.	Good
	Test performance when process and store pages with limited memory.	Normal
Integration testing	Test the server running at Windows, Linux, and Unix system.	Success
	Test the framework can integrating with key components of J2EE such as JTA and JAAS.	Partly success
	Test the framework can running with various database such as Oracle, SQLServer and DB2.	Success

表 1 所列结果表明:BWAP 在各种条件和数据环境下均能准确地完成分割页面、定位变化的页面区域、生成主题关键字、比较版本以及增量存储等功能;BWAP 通过了代码走查、评审、内存泄漏检测和代码覆盖软件的逻辑覆盖测试;BWAP 在高并发或大数据量的测试下运行性能良好;BWAP 能够在各种平台下运行,并且不依赖特定的数据库种类.

3.3 实验分析

基于页面 Block 的 Web 档案的采集和存储算法中最关键的一步就是定位页面发生变化的区域.以图 1 为例,如果采用观察的方法,则能很准确地确定变化的最小区域(虚线矩形).如果采用算法处理,若分区级别设为 1,则得到 *C* 区是变化的区域;若分区级别设为 2,则得到 *CC* 区是变化的区域;若分区级别设为 3,则得到 *CC*2~*CC*6 这 5 个区域是变化的区域(如图 2 和图 3 所示).因此,算法的判断与人的判断存在一定的差异.检验算法是否、合理准确的最好办法是手工验证.因此,我们首先选择了 200 对来自于文献[4]的历史网页,每对网页的采集时间非常接近,并且这些网页分属于不同的主题,以确保网页结构广泛、多样.首先,我们通过 BWAP 把每对网页中时间较早的一个置入 Web 档案库中,然后再装入时间较晚的一个.我们收集 BWAP 报告的后一个页面的分区结果和变

化的 Block,同时,我们选择 4 位志愿者分别来检查这些结果,并且以 Perfect,Good,Normal,Bad 这 4 个级别给以评价.图 11 和图 12 给出了评价结果.

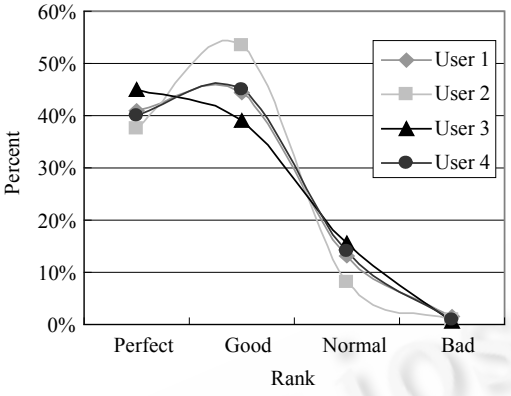

Fig.11 Test results of each user

图 11 每个用户的测试结果

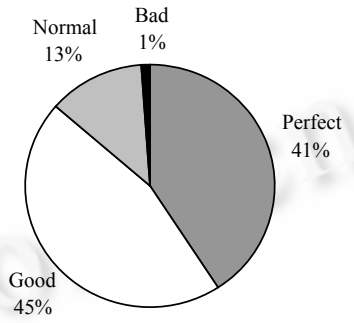

Fig.12 Average proportion of each rank

图 12 每个级别的平均比例

从图 11 的平滑曲线可以看出,4 位志愿者的评价结果非常相似,这是由于每人评价的数据集是相同的,都是算法处理这 200 个页面的结果;由于从优到劣只分出 4 个级别,曲线特征并不明显,但仍可以看出,除了 User 3 以外,其他曲线近似于一种正态分布,峰值位于 Good 级别,这些都表明算法的准确性和合理性.从图 12 可以看出,这 800 次评价中结果是 Perfect 和 Good 的占 86%,仅有 1%为 Bad.我们考察了评价为 Normal 以下的约 14%的页面发现,导致算法不精确的原因是多方面的:一些是浏览器和开源 HTML 解析工具的客观原因,主观原因及算法不足之处主要是无法区分那些用图片、Flash 和文字来分割的区域,因此,该算法还有很大的提升空间.尽管如此,不足 1%的 Bad 评价率已经证明该算法的正确性.此外,在这 800 次评价中并没有发现算法不能识别的页面变化.少量的页面分区不合理可能会对存储带来负面影响,但不会影响对 Web 页面的变化的采集和保存.

4 相关工作

Web Archive 属于一种新兴的技术.根据澳大利亚国家图书馆在 2003 年的统计,仅有 16 个国家拥有完整的国内 Web 归档系统[17].对 Web 档案的处理也处于发展阶段.我们在研究 Web 档案处理成果的同时,也参考了大量 Web 领域相关的工作.

首先是 Web 归档系统的实现技术.前人在 Web 归档系统的实现技术上作了大量研究.The Internet Archive 是 Web 归档系统的先锋,它通过一个叫做 Heritrix 的开源爬虫实现对页面的抓取[18].文献[19,20]描述了实现一个 Web 归档系统的系统结构和相关标准,并且实现了一个名为 Tomba 的原型系统.本文学习了它的很多思想以及实现方式,尤其在 Web 页面的存储结构设计上借鉴了很多.文献[21]实现了 NWA 工具集来访问 Nordic Web Archive 并进行查询搜索.此外,本研究也广泛参考了各国[22−26]的 Web 归档系统的实现技术.这些文献大多是在介绍 Web 档案的管理技术,或者在总体结构、存储和使用手段上进行研究.与此不同的是,本研究主要提出基于页面 Block 来管理 Web 档案,是对传统 Web 档案管理方法的一种改进.

其次是 Web 页面的分区技术.在传统的文本处理中,通常使用固定长度的段落或者窗口(window)来分割页面[27,28].一个固定长度的段落包含一定数目的连续单词.对 Web 文档来说,如果移除 HTML 标签和属性,则可以使用这种办法来分割页面.固定长度的分割虽然精简并具有高性能,但也损失了大量 HTML 信息,不宜用作 Web 档案页面的分割算法.此外,有许多学者考虑到运用 HTML 标签来分割页面,这些标签包括〈P〉,〈TABLE〉,〈UL〉等.文献[29]在一个 Web 查询处理系统中利用这些标签对页面进行了分割.文献[30]只把〈TABLE〉标签和〈TABLE〉标签内的所有子标签分为 1 个内容块,这显然忽略了大量信息,只使用〈TABLE〉标签是远远不够的.文献[31,32]

用一些像〈P〉,〈TABLE〉和〈UL〉这样的简单标签将网页分割成若干区域,并进一步进行转化和抽取摘要,然而,它们只是考虑标签类别而忽视了格式、样式等信息.此外,文献[33]对页面分解算法作出了大量评论,但没有涉及到技术细节.文献[34]对 Web 页面分区作了透彻的分析,并提出依赖可视化元素来分割页面的 VIPS 算法.该研究工作对我们有很大的启发,然而 VIPS 算法过于复杂,不宜用于 Web 归档系统.相比之下,本文提出的算法比较简单.另外,这些工作均未把网页分区技术运用到 Web 历史档案信息处理领域.尽管这些方法还都存在不足,但对本文的页面分区算法仍有很大的帮助.

本文也参考了一些 Web 存储技术[35,36]和 Web 挖掘以及主题抽取技术[37,38].这些技术并非是针对 Web 档案的采集和处理,或者属于一些底层的存储技术.我们借鉴了其思想,并灵活运用到 Web 档案的处理上.

5 结论与展望

本文描述了基于页面 Block 的 Web 档案的采集和存储方法,文中详细阐述了对页面进行分区、主题抽取、版本比较和增量存储这一过程中每个步骤的算法.本文提出的基于页面 Block 的管理思路将便于 Web 档案的管理,并对基于 Web 档案的查询、搜索、知识发现和数据挖掘应用提供丰富的数据资源.本研究的主要贡献有以下几点:

- 提出基于页面 Block 的 Web 档案的管理方式;
- 描述了页面分割算法、主题生成算法、页面比较算法和增量存储算法;
- 实现了一个 Web 归档原型系统 BWAP 并用实验数据来验证算法的正确性.

本研究还处于初步阶段.进一步的工作主要包括两个方向:一是进一步完善分区算法,包括能够适应页面在不改变总体布局的情况下增减 Block 的数目,Block 的 Label 信息的生成以及通过历史分区信息自动学习、调整、优化的分区算法;二是如何利用 Block 资源完成基于 Block 的历史搜索、问答系统和历史知识发现.

References:

[1] Ntoulas A, Cho J, Olston C. What's new on the Web? The evolution of the Web from a search engine perspective. In: Chen YR, Kovács L, Lawrence S, eds. Proc. of the 13th Int'l Conf. on World Wide Web. New York: ACM Press, 2004. 1−12.

[2] National Liberaray of Australia. Padi-Web archiving. 2006. http://www.nla.gov.au/padi/topics/92.html

[3] Web InfoMall. 2006 (in Chinese). http://www.infomall.cn/

[4] Internet archive WayBack machine. http://www.archive.org/Web/Web.php

[5] Gupta S, Kaiser G, Stolfo S. Extracting context to improve accuracy for HTML content extraction. In: Ellis A, Tatsuya H, eds. Proc. of the 14th Int'l Conf. on World Wide Web—Special Interest Tracks and Posters. New York: ACM Press, 2005. 1114−1115.

[6] Lin SH, Ho JM. Discovering informative content blocks from Web documents. In: Hand D, Keim D, eds. Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2002. 588−593.

[7] Wong WC, Fu AW. Finding structure and characteristics of Web documents for classification. In: Gunopulos D, Rastogi R, eds. Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. New York: ACM Press, 2000. 96−105.

[8] Yang YD, Zhang HJ. HTML page analysis based on visual cues. In: Antonacopoulos A, Gatos B, eds. Proc. of the 6th Int'l Conf. on Document Analysis and Recognition. Washington: IEEE Computer Society, 2001. 859−864.

[9] http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words

[10] http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/porter.c

[11] Kantrowitz M, Mohit B, Mittal V. Stemming and its effects on TFIDF ranking. In: Nicholas J, Peter I, Mun-Kew L, eds. Proc. of the 23rd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. New York: ACM Press, 2000. 357−359.

[12] MacDonald J. Versioned file archiving, compression, and distribution. UC Berkeley, 1999. http://www.cs.berkeley.edu/~jmacd/

[13] Berliner B. CVS II: Parallelizing software development. In: Proc. of the USENIX Winter 1990 Technical Conf. Berkeley: USENIX Association, 1990. 341−352.

[14] Gomes D, Campos JP, Silva MJ. Versus: A Web repository. 2003. http://xldb.fc.ul.pt/referencias

[15] Gomes D, LSantos A, Silva MJ. Managing duplicates in a Web archive. In: Liebrock LM, ed. Proc. of the 21st Annual ACM Symp. on Applied Computing. New York: ACM Press, 2006. 818−825.

[16] Cho J, Garcia-Molina H. Estimating frequency of change. ACM Trans. on Internet Technology (TOIT), 2003,3(3):256−290.

[17] Phillips M. PANDORA, Australia's Web archive, and the digital archiving system that supports it. DigiCULT.info, 2003,(6):24−30. http://www.nla.gov.au/nla/staffpaper/2003/mphillips1.html

[18] Halse JE, Mohr G, Sigurdsson K, Stack M, Jack P. Heritrix developer documentation. 2005. http://crawler.archive.org/articles/developer_manual/index.html

[19] Gomes D, Freitas S, Silva MJ. Design and selection criteria for a national Web archive. In: Thanos C, Gonzalo J, eds. Proc. of the 10th European Conf. of Research and Advanced Technology for Digital Libraries (ECDL). Berlin, Heidelberg: Springer-Verlag, 2006. 196−207.

[20] Silva MJ. Searching and archiving the Web with tumba!. In: Proc. of the 4th Conf. on Association Portugal of System and Information (CAPSI). 2003. http://xldb.fc.ul.pt/data/Publications_attach/tumba-search+archive-capsi-final.pdf

[21] Hallgrimsson D, Bang S. Nordic Web archive. In: Michael D, ed. Proc. of the 3rd ECDL Workshop on Web Archives. 2003. http://bibnum.bnf.fr/ECDL/2003/proceedings.php?f=ecdl2003

[22] National Diet Library (Japan). Web archiving project. 2007. http://warp.ndl.go.jp

[23] UK Web archiving consortium. 2006. http://info.Webarchive.org.uk

[24] The Library of Congress. Minerva Web archiving project. 2006. http://lcWeb2.loc.gov/cocoon/minerva/html/ minerva-home.html

[25] McCown F. Dynamic Web file format transformations with grace. In: Proc. of the 5th Int'l Web Archiving Workshop and Digital Preservation (IWAW 2005). 2005. 22−23. http://www.iwaw.net/05/papers/iwaw05-mccown2.pdf

[26] Lampos C, Eirinaki M, Jevtuchova D, Vazirgiannis M. Archiving the greek Web. In: Proc. of the 4th Int'l Web Archiving Workshop (IWAW 2004). 2004. http://www.iwaw.net/04/Lampos.pdf

[27] Callan J. Passage-Level evidence in document retrieval. In: Croft BW, Rijsbergen V, eds. Proc. of the 7th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. New York: ACM Press, 1994. 302−310.

[28] Kaszkiel M, Zobel J. Effective ranking with arbitrary passages. Journal of the American Society for Information Science, 2001, 52(4):344−364.

[29] Diao YL, Lu HJ, Chen ST, Tian ZP. Toward learning based Web query processing. In: Abbadi AE, Brodie ML, Chakravarthy S, Dayal U, Kamel N, Schlageter G, Whang KY, eds. Proc. of the 26th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 2000. 317−328.

[30] Li SH, Ho JM. Discovering informative content blocks from Web documents. In: Hand D, Keim D, Ng R, eds. Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data mining. New York: ACM Press, 2002. 588−593.

[31] Kaasinen E, Aaltonen M, Kolari J, Melakoski S, Laakko T. Two approaches to bringing Internet services to WAP devices. Computer Networks: The Int'l Journal of Computer and Telecommunications Networking, 2000,33(1-6):231−246.

[32] Buyukkokten O, Garcia H, Paepche A. Accordion summarization for end-game browsing on PDAs and cellular phones. In: Rosson MB, Gilmore DJ, eds. Proc. of the SIG-CHI on Human Factors in Computing Systems. New York: ACM Press, 2001.

[33] Rahman A, Alam H, Hartono R. Content extraction from HTML documents. In: Hu JY, ed. Proc. of the 1st Int'l Workshop on Web Document Analysis (WDA 2001). New York: ACM Press, 2001. 3−10.

[34] Cai D, Yu S, Wen JR, Ma WY. Extracting content structure for Web pages based on visual representation. In: Zhou XF, Zhang YC, Orlowska ME, eds. Proc. of the 5th Asia Pacific Web Conf. Berlin, Heidelberg: Springer-Verlag, 2003. 406−417.

[35] Burner M, Kahle B. WWW archive file format specification. Alexa Internet Inc., 1996. http://pages.alexa.com/company/arcformat.html

[36] Gomes D, Santos AL, Silva MJ. Webstore: A manager for incremental storage of contents. Technical Report, DI/FCUL TR 04−15, Lisbon: University of Lisbon, 2004.

[37] Sekiguchi Y, Kawashima H, Okuda H, Oku M. Topic detection from Blog documents using users' interests. In: Aberer K, Hara T, eds. Proc. of the 7th Int'l Conf. on Mobile Data Management (MDM 2006). Washington: IEEE Computer Society, 2006. 108−111.

[38] Wang XY, Xiong FY, Ling B, Zhou A. A similarity-based algorithm for topic exploration and distillation. Journal of Software, 2003,14(9):1578−1585 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1578.htm

附中文参考文献:

[3] 中国 Web 信息博物馆.2006. http://www.infomall.cn/

[38] 王晓宇,熊方,凌波,周傲英.一种基于相似度分析的主题提取和发现算法.软件学报,2003,14(9):1578−1585. http://www.jos.org.cn/1000-9825/14/1578.htm

宋杰(1980－),男,安徽淮北人,博士生,主要研究领域为 Web 数据仓库,软件工程.

鲍玉斌(1968－),男,博士,副教授,CCF 高级会员,主要研究领域为 Web 数据仓库.

王大玲(1962－),女,博士,教授,CCF 高级会员,主要研究领域为 Web 挖掘.

申德荣(1962－),女,博士,教授,CCF 高级会员,主要研究领域为 Web 服务.

第 5 届智能 CAD 与数字娱乐学术会议

征 文 通 知

由中国图象图形学会计算机动画与数字娱乐专业委员会、中国人工智能学会智能 CAD 与数字艺术专业委员会、以及中国工程图学学会国际联络工作委员会联合主办，大连大学承办的第 5 届智能 CAD 与数字娱乐学术会议，将于 2008 年 7 月 22 日在美丽的海滨城市大连举行。

一、征文内容(主要包括，但不限于)

- 智能 CAD
- 数字艺术
- 计算机动画
- 虚拟现实
- 网络游戏
- 可视化技术
- 模式识别
- 人机交互
- 计算机图形学
- 图像处理
- 信息融合
- 多媒体技术
- 计算机视觉
- 人工智能
- 数字内容管理
- 交互式玩具
- E-Home
- 运动捕获动画
- 数字博物馆
- 人脸表情跟踪与识别

二、论文格式及注意事项

论文相关要求请登陆会议网站：http://202.199.159.247/cide2008/

电子投稿，请将 WORD 格式的文件发到：xpwei@dlu.edu.cn，投稿时务必在电子邮件正文中留下通讯作者的详细通讯地址、邮政编码、电话，以便联系。

三、重要日期

截稿日期：2008 年 3 月 31 日

录用日期：2008 年 4 月 30 日

修改稿接收及注册截止日期：2008 年 5 月 31 日

四、联系方式

联系人及联系电话：张强(0411-87403733)

电子信箱：zhangq30@gmail.com