汉语分词索引字数与分词效率的对比研究

何利益[1,2],郭 罡[2],郭建彬[2]
HE Li-yi[1,2],GUO Gang[2],GUO Jian-bin[2]

1.中国科学技术大学 电子工程与信息科学系,合肥 230027
2.中国人民解放军 96151 部队,安徽 黄山 245041
1.Department of Electronic Engineering and Information Science,University of Science and Technology of China,Hefei 230027,China
2.96151 Unit of PLA,Huangshan,Anhui 245041,China

HE Li-yi,GUO Gang,GUO Jian-bin.Contrast study on Chinese word segmentation efficiency with different index degree.Computer Engineering and Applications,2008,44(26):135-137.

Abstract: According to the Chinese dictionary word segmentation efficiency that based on the Double-Character-Hash-Index (DCHI) mechanism exceeds clearly based on the First-Character-Hash-Index(FCHI) mechanism,this paper lucubrates to the Chinese word-building characteristic and provides a new segmentation dictionary mechanism named Three-Character-Hash-Indexing (TCHI) mechanism,which exploits character coding index sufficiently.The results show that the TCHI dictionary mechanism can improve speed and achieve more efficiency than FCHI,DCHI and four;character-hash-index in Chinese dictionary word segmentation mechanism.

Key words: computer application;Chinese word segmentation;dictionary mechanism;Three Character Hash Index(TCHI)

摘 要:针对汉语分词词典中双字哈希索引机制未能充分利用索引分词,而分词效率又明显优于首字哈希索引机制的问题,在充分分析汉语构词特点的基础上,提出了基于三字哈希索引的分词词典机制,并通过将字串的三态标记与下一索引指针的乘积作为哈希值的链地址法,简化了词典结构,节省了内存空间。理论分析和真实语料仿真均证明了三字哈希索引机制与不同字数的其他索引机制相比,具有更好的分词效率。

关键词:计算机应用;中文分词;词典机制;三字哈希索引
DOI:10.3778/j.issn.1002-8331.2008.26.041 **文章编号:**1002-8331(2008)26-0135-03 **文献标识码:**A **中图分类号:**TP391

1 引言

词典查询是中文信息处理的一个重要基础环节,对系统效率起着关键性作用。针对词典的查询方法,前人做了大量工作,并形成了许多好的词典组织结构和相应的查询算法[1-2]。

文献[3]介绍了 3 种典型的词典查询方法:整词二分法、TRIE 索引树法、逐字二分法。(1)基于整词二分的词典机制:由词典正文、词索引表、首字哈希表 3 部分组成,通过首字哈希表的哈希定位和词索引表,确定指定词在词典正文中的可能位置范围,进而通过整词二分进行定位。该算法数据结构简单、占用空间小,构建及维护简单易行,但由于采用全词匹配的查询过程,查询效率较低。(2)基于 TRIE 索引树的词典机制:由首字哈希表和 TRIE 索引树结点两部分组成,其优点是不需预知查询词的长度,沿着树链逐字匹配即可;缺点是构造和维护比较复杂,而且都是单词树枝,浪费了一定的空间。(3)基于逐字二分的词典机制:采用了整词二分的词典组织结构和 TRIE 索引树的逐字匹配查询算法,提高了匹配效率;但由于采用整词二分的词典组织结构,效率提高有限。

文献[4]提出了基于双字哈希索引的词典机制,该机制利用了“整词二分”及“TRIE 索引树”二者的优点,仅对词语的前两个字顺次建立哈希索引,构建深度为 2 的 TRIE 子树,词的剩余字串则按序组成类似“整词二分”的“词典正文”,与文献[3]的分词效率有明显提高。

本文根据计算机对字符串比较的处理效率大大低于数字索引的处理效率,通过对汉语构词特点的深入分析,得出基于三字哈希索引的词典机制能够更充分利用索引查询,减少字符串比较的次数,具有更好的分词效率的结论,并从理论分析和真实语料仿真两方面给予了证实。

2 汉语构词特点

本文对 1998 年 1 月人民日报语料库(北京大学计算语言学研究所和富士通研究开发中心有限公司共同制作的标注语料库,以下简称 PFR 语料库)进行统计,统计信息如表 1。

2.1 索引查询

由表 1 可见,首字索引可实现对不可做前缀的单字词进行

作者简介:何利益(1976-),男,研究生,工程师,主要研究方向:中文信息处理、数据挖掘;郭罡(1963-),男,高级工程师,主要从事气象、网络信息分析与处理等方面的研究与管理;郭建彬(1978-),男,助理工程师,主要从事互联网信息搜集与处理等工作。
收稿日期:2007-11-05 **修回日期:**2008-01-21

表 1 PFR 语料库词数词频统计表

词类	词数/个	词数/(%)	词频/个	词频/(%)
不可做前缀的单字词	263	0.53	3 618	0.45
可做前缀的单字词	668	1.36	226 269	28.02
只能做前缀的单字	2 975	–	–	–
不可做前缀的双字词	25 994	52.85	253 729	31.42
可做前缀的双字词	5 159	10.50	250 323	31.10
只能做前缀的双字	6 635	–	–	–
不可做前缀的三字词	10 382	21.12	46 038	5.70
可做前缀的三字词	311	0.63	4 915	0.61
只能做前缀的三字	5 570	–	–	–
不可做前缀的四字词	5 321	10.82	17 696	2.19
可做前缀的四字词	134	0.27	2 044	0.25
只能做前缀的四字	688	–	–	–
多字词	950	1.93	2 858	0.35

索引切分,在词典中占有 0.53%的词数 0.45%的词频;第二字索引可实现对可做前缀的单字词以及不可做前缀的双字词进行索引切分，在词典中占有 54.21%的词数 59.44%的词频;第三字索引可实现对可做前缀的双字词以及不可做前缀的三字词的索引切分,在词典中占有 31.62%的词数 36.8%的词频;第四字索引可实现对可做前缀的三字词以及不可做前缀的四字词进行索引切分,在词典中占有 11.45%的词数 2.8%的词频。

2.2 字符串比较

(1)第四字索引后,对于可做前缀的四字词以及多于四字的词,共有 822 个前缀 1 084 个词,每前缀对应 1.34 个词,采用最大正向匹配算法,平均需要 1.17 次字符串比较,总词频比较(2 044+2 858)*1.17=5 732 次。

(2)第三字索引后,对于可做前缀的三字词和多于三字的词,共有 5 881 个前缀 6 716 个词,每前缀对应 1.14 个词,采用最大正向匹配算法,平均需要 1.07 次字符串比较,总词频比较 27 513*1.07=29 439 次。

(3)第二字索引后,对于可做前缀的双字词以及二字以上的词先采用一次逐字二分查找,再进行最大正向匹配算法。逐字二分：共有 11 794 个前缀 22 257 个词，每前缀对应 1.9 个词,需要 1.4 次的字符串比较,词频比较(250 323+46 038)*1.4=414 905 次;后续比较次数同(2),总词频比较 444 339 次。

(4)首字索引后,对于可做前缀的单字词和一字以上的词,采用第二字、第三字逐字二分,第三字以后最大正向匹配的算法。第二字逐字二分:共有 3 643 个前缀对应 48 918 个词,每前缀对应 13.4 个词,需要 2.5 次字符串比较,词频比较(226 269+253 729)*2.5=1 199 995 次;后续比较次数同(3),总词频比较 1 644 334 次。

由以上分析可以看出,对于未切分 PFR 语料,三字哈希索引与双字哈希索引相比,可增加 36.8%词频的索引分词,减少 414 900 次字符串比较;而四字哈希索引与三字哈希索引相比,只增加了 2.8%词频的索引分词,减少 23 707 次字符串比较。由于可做前缀的二字词和不可做前缀的三字词占有较大部分的词频,双字哈希索引无法实现这一部分词的索引切分,所以未能充分发挥索引分词的优势;而三字索引以后未切分词仅占有较少的词频,将索引分词扩展到第四字或者更高,并不能发挥相应索引的优势。因此,三字哈希索引与不同字数的其他索引机制相比,具有更好的分词效率。

3 三字哈希索引词典机制

本文根据汉语的构词特点,构建了“词的前三字哈希索引+剩余字串最大匹配”的分词词典机制,并在词的状态标记和哈希表指针的实现上进行了改进,使得词典结构更简单,实现更简便[5]。词典结构如图 1 所示。

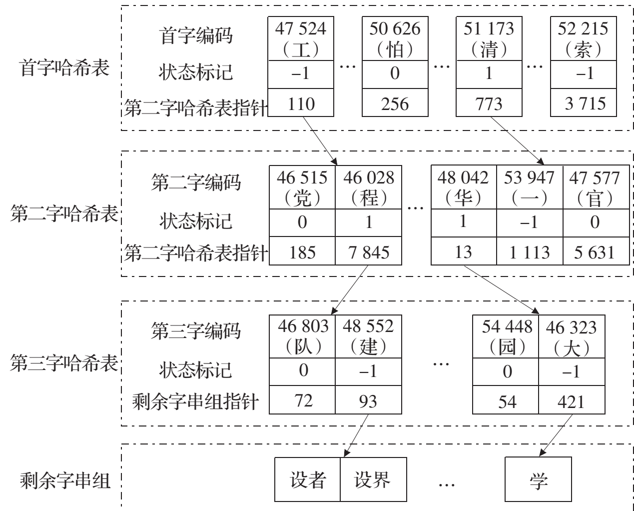

图 1 三字哈希索引分词词典机制

3.1 字符到编码的映射

本文在实现字符到编码的映射上,采用公式(1)进行映射。

$$Offset=ch1*256+ch2 \quad (1)$$

其中 $ch1$ 为编码的高字节(单字节时为 0),$ch2$ 为编码的低字节。这种编码映射方式能够实现 unicode2.0 编码规范的所有单字节、双字节编码字符,使得词典不再拘束于 GB2312–80 编码规范规定的字符范围[1]。词典中词的前三个字分别以公式(1)计算的编码值表示。

3.2 字串的三态标记

分词过程中，在词典中通过索引发现的字串可以有 3 种状态:(1)词,不能构成更长词的前缀,在词典中用 0 标记;(2)词,并能构成更长词的前缀,在词典中用 1 标记;(3)不是词,只能构成词的前缀,在词典中用–1 标记。如果仅用是否为词的二态标记,则无法一次区分一个词是单独成词还是构成词的前缀的问题。

3.3 词典结构及内存空间占用

本文词典前三个字及词典正文共用 4 张表分别存储,前 3 个字仅存储对应字符的编码值,而不存储字符本身,哈希表装载因子为 0.5;词典正文只存储词第三字以后的剩余字串。

(1)首字哈希索引。数据表包含 3 项内容,首字编码、指向第二字的指针、首字构词状态。哈希索引设置如下:

①哈希键(2 Byte):首字编码。

②哈希值(2 Byte):首字构词状态与指向第二字指针的乘积。

占用内存:3 906*4/0.5=31 248 Byte。

(2)第二字哈希索引。数据表包含 4 项内容:第二字编码、指向第三字的指针、第二字构词状态、对应的首字地址。哈希索引设置如下:

①哈希键(2 Byte):第二字编码。

②哈希值(2 Byte):第二字构词状态与指向第三字指针的乘积。

占用内存:37 788*4/0.5=302 304 Byte。

(3)第三字哈希索引。数据表包含 4 项内容:第三字编码、指向剩余字串组的指针、第三字构词状态、对应的第二字地址。哈希索引设置如下:

①哈希键（2 Byte）：第三字编码。

②哈希值（2 Byte）：第三字构词状态与指向剩余字串组指针的乘积。

占用内存：16 263*4*2=130 104 Byte。

（4）剩余字串组。数据表包括两项内容：除去词的前三个字的剩余字串、对应的第三字地址。数组设置如下：

①剩余字串（不定长 2n byte）：除去词的前三个字的剩余字串。

剩余字串组包括 5 881 个词共 7 865 个字符，占用内存：7 865*2=15 730 Byte。

词典总共占用内存：479 386 Byte。哈希表采用链地址法[5]，根据词的前缀数量设置相应的哈希表子集合初始长度，因为哈希运算采用素数作为哈希表长度具有较高的处理效率，申请的哈希表长度比实际需要的空间总是要多些，因此，词典实际占用的内存比上述计算值应略高一些。

本文设置的哈希值，既包含了词的状态标记信息，又包含了下一索引指针信息，即：值=0，对应字串是词，不是前缀；值<0，对应字串是前缀，不是词，取其正值即为下一索引的地址；值>0，对应字串是前缀，也可成词，其值即为下一索引的地址。与目前典型的分词词典机制[3-4]相比，这种方法无需单独构建哈希索引指针，大大减少了内存开销，简化了词典结构。

3.4 分词实现过程

给定文本串 $C_0C_1C_2\cdots C_n$，依据上述算法，分词实现过程如下：

（1）指针处取字符 C_0，由公式（1）计算其编码值 c0Code，判断首字哈希索引中是否存在键 c0Code。①不存在，指针后移一位，转（1）；②存在，获取对应哈希值 secondIndex。若为 0，C_0 为单字词，指针后移一位，转（1），否则转②。

（2）取指针下一位置字符 C_1，由公式（1）计算其编码值 c1Code。以 secondIndex（若为负取其正值）定位到第二字哈希索引，判断是否存在键 c1Code。①不存在，如果 secondIndex>0，C_0 为单字词；指针后移一位，转（1）。②存在，获取对应哈希值 thirdIndex，若为 0，C_0C_1 为二字词，指针后移两位，转（1），否则转（3）。

（3）取指针第三位的字符 C_2，由公式（1）计算其编码值 c2Code。以 thirdIndex（若为负取其正值）定位到第三字哈希索引，判断是否存在键 c2Code。①不存在，如果 thirdIndex>0，C_0C_1 为双字词，指针后移两位，转（1）；否则，如果 secondIndex>0，C_0 为单字词；指针后移一位，转（1）。②存在，获取哈希值 elseStringIndex。若为 0，$C_0C_1C_2$ 为三字词，指针后移三位，转（1）；否则转（4）。

（4）以 elseStringIndex（若为负取其正值）定位到剩余字串组，获取对应的字串，采用最大正向匹配算法，在切分字串中依次取对应词长与其比较。①成功匹配，则切分一多字词，将指针后移切分词长位，转（1）。②未成功匹配，如果 elseStringIndex>0，$C_0C_1C_2$ 为三字词，指针后移三位，转（1）；否则，如果 thirdIndex>0，C_0C_1 为双字词，指针后移两位，转（1）；否则，如果 secondIndex>0，C_0 为单字词；指针后移一位，转（1）。

4 实验结果

本文从空间、时间两个方面分别对首字、双字、三字、四字 4 种不同索引字数的词典机制进行考察，实验语料采用 4 个不同长度的未切分 PFR 语料（实验 1：57 K；实验 2：329 K；实验 3：2.45 M；实验 4：5.39 M。单位：Byte）。词典采用 PFR 语料的所有人工切分词共 49 182 条。为了保证实验的公正性，实验在相同的软硬件环境（C# 编程语言，.net 编程环境，Pentium 4 CPU 3.0 GHz，512 MHz Memory）来实现，哈希表均采用 C# 内置的 Hashtable 及相同的组织结构。对于首字、双字哈希索引词典机制，三字以前采用逐字二分的分词算法，三字以后采用最大正向匹配的分词算法；对于三字、四字哈希索引机制，剩余字串采用最大正向匹配的分词算法。实验结果如表 2 所示。

表 2 各词典机制空间、时间比较之实验结果

词典机制	词典空间/Byte	实验 1/ms	实验 2/ms	实验 3/ms	实验 4/ms
首字哈希	177 676	125	790	5 360	11 593
双字哈希	383 478	39	261	1 687	3 844
三字哈希	479 386	27	181	1 353	3 141
四字哈希	515 560	28	183	1 313	3 211

4.1 空间

由表 2 可以看出，随着哈希索引次数的增加，词典占用的内存也逐渐增多，但三字哈希索引总共占用 479 386 Byte 的内存空间，这在目前的硬件环境下，词典空间并不是主要问题，而且，由于在哈希值的设置上进行了改进，本文的词典机制与目前典型的词典机制相比节省了大量的内存空间。

4.2 时间

由表 2 可以看出，双字哈希索引机制比首字哈希索引机制的分词效率提高了约 2.1 倍，三字哈希索引机制比双字哈希索引机制提高了约 32%，四字哈希索引机制比三字哈希索引机制分词效率略低。由于三字哈希与双字哈希相比减少了较多的字符串比较次数，而四字哈希与三字哈希相比字符串比较次数降低不多，而搜索深度加深，因此，实验结果与本文对汉语构词特点的分析结果相吻合。

5 结论

本文认真分析了汉语构词特点，根据三字索引能够实现绝大部分词数和词频索引分词的特点，提出了三字哈希索引机制，并通过理论分析说明了该词典机制比其它索引字数的词典机制具有更好的分词效率，通过真实语料仿真实验证明了该词典机制比双字哈希索引词典机制能够提高 32%的分词效率，比首字哈希索引词典机制提高 3.1 倍的分词效率，比四字哈希索引机制也有略高的分词效率。同时，本文通过将字串做三态标记，并使用该三态标记与下一索引指针的乘积作为哈希值的链地址法，使得本文的词典机制与目前典型的词典机制相比，结构更简单，维护更简便，内存占用也大幅减少。因此，本文设计的三字哈希索引词典机制结构简单、维护方便、空间复杂度小、分词效率高，是一种优良的汉语分词词典机制。

参考文献：

[1] 朱巧明，李培峰，吴娴，等.中文信息处理技术教程[M].北京：清华大学出版社，2005.

[2] 马晏.基于评价的汉语自动分词系统的研究与实现[M]//语言信息处理专论.北京：清华大学出版社，1996.

[3] 孙茂松，左正平，黄昌宁.汉语自动分词词典机制的实验研究[J].中文信息学报，2000，14（1）：1-6.

[4] 李庆虎，陈玉健，孙家广.一种中文分词词典新机制——双字哈希机制[J].中文信息学报，2002，17（4）：13-18.

[5] 陈明.数据结构[M].北京：清华大学出版社，2005.