汉语自动分词词典新机制—词值哈希机制①

韩 莹, 王茂发, 陈新房, 潘志安, 张艳霞

(防灾科技学院 灾害信息工程系, 北京 101601)

摘 要: 汉语词典查询是中文信息处理系统的重要基础部分, 对系统效率有重要的影响. 国内自 80 年代中后期就开展了中文分词词典机制的研究, 为了提高现有基于词典的分词机制的查询效率, 对于词长不超过 4 字的词提出了一种全新的分词词典机制——基于汉字串进制值的拉链式哈希机制即词值哈希机制. 对每个汉字的机内码从新编码, 利用进制原理, 计算出一个词语的词值, 建立一个拉链式词值哈希机制, 从而提高查询匹配速度.

关键词: 中文信息处理; 中文分词; 词典机制; 2000 进制; 拉链式词值哈希机制

New Dictionary Mechanism for Chinese Word Segmentation

HAN Ying, WANG Mao-Fa, CHEN Xin-Fang, PAN Zhi-An, ZHANG Yan-Xia

(Department of Disaster Information Engineering, Institute of Disaster Prevention, Beijing 101601, China)

Abstract: Word query in Chinese Dictionary is essential part in Chinese information processing system. It has a great impact on system efficiency. The Chinese word segmentation has been studied since the late 1980s. In order to improve the existing word query efficiency, for short word of no more than 4 Chinese characters, a new hash algorithm is proposed, named Zipper-style hash indexing based on the value of each characters in Chinese word. The hash value is calculated according to machine code of each character, the weight of the left character is big than the right. The weight is equal to the maximum value of all Chinese characters minus the minimum value. The speed of word query is improved with this kind of Zipper-style Chinese word value hash indexing.

Key words: Chinese information processing; Chinese word segmentation; dictionary mechanism; two thousand decimal; zipper-style Chinese word value hash indexing

汉语自动分词是汉语信息处理的前提, 广泛应用与中文全文检索、中文自动全文翻译、中文文语转换(TTS)等领域. 自动分词的基本算法主要分为两大类: 基于词典的分词方法[1,2]和基于频度统计的分词方法[3,4]. 基于词典的分词方法是以汉语词典为基础对中文语句通过匹配进行切分, 这种方法包括 3 种基本算法[5]:正向最大匹配法、逆向最大匹配法、全切分法. 不论哪种基于词典的分词方法, 词典的查询速度是决定分词算法效率的决定性因素[6]. 先今众多的中小型搜索引擎仍然沿用树形搜索结构作为其搜索核心, 理论查询效率只是 O(log(n)), 而且实现技术复杂. 散列结构能够实现 O(1)的搜索效率, 并且维护方便. 不论是哪种基于词典的分词方法, 分词词典的查询速度是匹配算法效率的直接决定因素, 因而建立高效快速的分词词典机制势在必行[7].

在汉语词汇中, 对常用词典(人们日报)的统计发现, 在 108783 个词汇中, 51%为 2 字词, 31%为 3 字词, 13%为 4 字词, 3%左右为单字词, 多余 4 字以上的词汇只占 2%左右[8]. 为了进一步提高查询效率, 对于词长不超过 4 字的词, 本文提出了一种全新的分词词典机制——拉链式词值哈希词典算法, 并描述了该数据结构的主要思想和实现过程.

1 基本思想

首先了解一下国际码的概念, 国标码是指我国 1981 年公布的“中华人民共和国国家标准信息交换汉

① 收稿时间:2012-08-03;收到修改稿时间:2012-09-06

字编码”, 代号为“GB2312-80”, 由连续的两个字节组成. 汉字在计算机内部其内码是唯一的. 因为汉字处理系统要保证中英文的兼容, 当系统中同时存在ASCII码和汉字国标码时, 将会产生二义性. 例如: 有两个字节的内容为 30H 和 21H, 它既可表示汉字“啊”的国标码, 又可表示西文“0”和“!”的 ASCII 码. 为此, 汉字机内码应对国标码加以适当处理和变换. GB 18030 所收录汉字完全对应于 Unicode 的“中日韩统一汉字”, 收录的汉字是20924个. GB码的机内码为二字节长的代码, 它是在相应 GB 码的每个字节最高位上加“1”, 即

汉字机内码＝汉字国标码＋8080H

例如, “啊”字的国标码是 3021H, 其汉字机内码则是 B0A1H, 其他汉字的机内码值是连续顺序增加的. 根据这个特点每个汉字都可以唯一地从机内码映射到 0-19999 间的一个序列码, 从而每个汉字串都可以唯一地映射到一个数字, 这样对词语的查询就可以转化为基于数字的查询. 例如: 汉字机内码表的第一个汉字“啊”, 机内码是十六进制的 B0A1, 转化成十进制值为 45217; 第二个汉字“阿”, 机内码是十六进制的 B0A1, 转化成十进制值为 45218. 让每个汉字计算出的机内码十进制值减去机内码值最小的“啊”的机内码十进制值 45217, 得到的这个偏移量作为一个该汉字新的序列码. 那么“啊”字的序列码就是 0, “阿”字的序列码就是 1……其他的汉字如“文”字的序列码计算方法是: 首先将“文”字的机内码转化成十进制为: (CCEC)16=(52460)10, 然后 52460-45217=7243, “天”字的序列码为 7243.

因为简体汉字有 17000, 便于程序扩充, 我们假设汉字有 20000, 对汉字进行重新编码, 即让每个汉字的编码减去编码最小的汉字“啊”的编码所得到的值为该汉字的编号. 把所有的汉字重新计算就得到一个从 0 到 19999 的一个汉字字符序列码.

另外我们知道, 二进制由 0, 1 两个基数来表示, 逢 2 进 1. 八进制由 0-7 这 8 个基数来表示, 逢 8 进一. 十进制由 0-9 这 10 个数表示, 逢 10 进 1, 同样道理十六进制也是如此. 二进制、八进制及十六进制转化成十进制的方法是:

$(101)_2=1*2^2+0*2^1+1*2^0=(5)_{10}$

$(123)_8=1*8^2+2*8^1+3*8^0=(83)_{10}$

$(123)_{16}=1*16^2+2*16^1+3*16^0=(291)_{10}$

同理, 我们已经有 20000 个汉字的 0-19999 的序列码, 用这些序列码来表示汉字串, 汉字串就转化成了数字串, 逢 19999 进 1, 根据进制名称定义的方法, 本文称之为 20000 进制. 然后按照进制转换原理把该 19999 进制表示的数字串转换成十进制, 这样任何两个不同的汉字串就有了互不相同的整数值. 例如: 现在有词“天文”, 按照前面提到方法求得“天”字的序列码 7243, “文”字的序列码为: 7715, 那么“天文”的数字串为(7243 7751), 把 $(7243\ \ 7751)_{20000}$ 转换成十进制 $Num=7243*20000^1+7715*20000^0$,即得到 $(7243\ \ 7751)_{20000}=(14493715)_{10}$. 通过这种方式, 可以把每个词语转换成一个互不相等的整数值, 本文称之为词值.

2 索引机制的建立

对整个词典进行编码之后, 每个词语就对应着一个词值, 为了提高检索的效率, 建立一个词值对应的 Hash 机制. 建立 Hash 抽象数据类型要解决两个方面的内容: (1)Hash 函数的选取, (2)解决冲突得方法. 本实验采用拉链法解决冲突, 另外采用静态分配桶(Hash 表的长度)的方法构建 Hash 表.

2.1 用除留余数法构造 Hash 函数

本文取词值被某个质数 P 除后所得的余数为哈希地址.

H(Key)= Key MOD P(P<=m)

Key 为词值, 根据大量统计数据发现本实验哈希表中静态分配桶即哈希表表长 m 和质数 P 的值都取 999907 能使冲突次数相对比较少.

2.2 使用链地址法处理冲突

拉链法解决冲突的做法是: 通过词值计算该词语的 Hash 值, 将所有 Hash 值相同的结点链接在同一个单链表中. 若选定的散列表长度为 m, 则可将散列表定义为一个由 m 个头指针组成的指针数组 T[0..m-1]. 凡是散列地址为 i 的结点, 均插入到以 T[i]为头指针的单链表中. 对于链表中每个结点的结构如下定义.

2.3 Hash 表的定义

```
typedef   struct   HashNode /*Hash 链结点*/
{
  unsigned __int64 Word_ Key;/*词值*/
  HashNode  *HashNode_Next;/*指向下一个结点的指针  */
}HashNode;
typedef struct/*Hash 静态分配桶的定义*/
{
  HashNode *HashNode_Next;/*头指针    */
}HashTable,*HashTablep;
```

数据结构如图 1 所示:

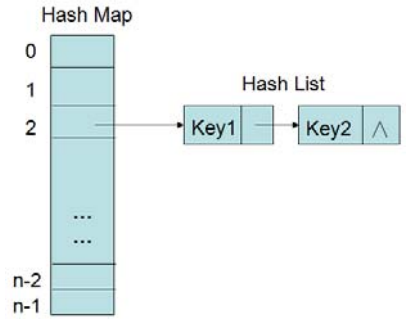

图 1 词值的拉链式 Hash 词典机制

3 实验结果及分析

为了验证本机制的效率, 对本实验结果进行了详细的分析, 同时提出了本实验的不足之处.

3.1 空间分析

① 存放指针的静态分配桶即哈希表长度 999907;

② Hash 链: 根据词语的多少决定结点的个数, 每个词语占用一个结点.

3.2 时间分析

在测试程序时, 测试词库使用一个含有 120546 个词的基本通用词表, 词表中的词长 L<=4. 测试结果显示, 在采用拉链法解决数据冲突时, 哈希链最长的含有 4 个结点. 其中哈希链长为 1 的有 106933 条, 这就意味着有 106933/120546=%88.71 的词通过一次比较可以查找成功; 哈希链长为 2 的有 12840 条, 占%10.65; 哈希链长为 3 和 4 的分别为的是 729 条和 44 条, 总共占 0.64%.

3.3 本机制存在的不足及改进

因为计算机对数字大小的存储限制, 本文所设计的词典机制只适合存储不超过四字的词语的词值. 文献[9]对汉语中的词条分布进行了统计发现, 中文词语中双字词语最多, 三字词语和四字词语也较多, 对于五字及五字以上的词, 可以采用其他的传统的存储方法, 这样既能有效的节省存储空间, 又能最大限度的提高匹配的时间效率.

由于在进行分词过程中, 每篇文章都会存在大量的重复数据, 所以可以在 Hash 链表的结点结构体中加入词频成员, 并且按照词频由高到低的顺序排列, 让使用频率高的词语排在链表的前面.

```
typedef  struct  HashNode  /*Hash 链结点*/
{
  unsigned __int64 Word_ Key;/*词值*/
  HashNode  *HashNode_Next;/*指向下一个结点的指针 */
  int  Word_Fre;/*词频*/
}HashNode;
```

改进后的数据结构如图 2 所示:

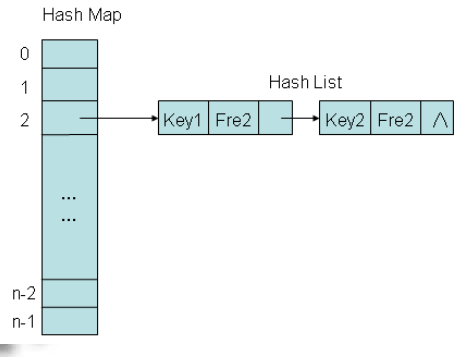

图 2 词值的改进拉链式 Hash 词典机制

4 结论

综上所述, 根据汉字中词长 L<=4 的所占比例比较多的特点, 本文采用了拉链式词值哈希机制, 该机制数据结构简单、占用空间小, 提高了一定的匹配效率, 是一种较简洁、易维护、更高效的词典组织机制.

参考文献

1 孙茂松,邹嘉彦.汉语自动分词中的若干理论问题.语言文字应用, 1995,(4).

2 梁南元.书面汉语自动分词系统—CDWS.中文信息学报,1987,2(2).

3 Choi A, Cheng CH, Ko YL. Word extraction from Chinese documents by occurrence counts. 1988 Int. Conference on computer Processing of Chinese and Oriental Languages, Toronto, Canada,488–491.

4 Fan CK, Tsai WH. Automatic word identification in Chinese sentences by the relaxation technique. Computer Processing of Chinese and Oriental Languages,1988,4(1):33–56.

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

6 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.中文信息学报,2003,4.

7 孙茂松,左正平,黄昌宁.汉语自动分词词典新机制的实验研究.中文信息报,2000,(1).

8 吴晶晶,荆继武,聂晓峰,等.一种快速中文分词词典机制.中国科学院研究生院学报,2009,26(5):703−711.

9 李振星,徐泽平,唐卫清,唐荣锡.全二分最大匹配快速分词算法.计算机工程与应用,2002,(11).