汉语词语的两字 hash 算法

朱晓丹 刁倩 周富秋

英特尔中国研究中心

E-mail: {xiaodan.zhu; qian.diao; joe.f.zhou}@intel.com

摘 要: 通过首字 hash 在词典中查找单词是汉语计算中很常用的一种算法，但该方法有明显的缺陷。本文提出两字 hash 算法，在相同的空间占用下，查找效率有很大的提高，而且 hash 表大小可以自由设定，可以用在汉语计算的很多领域。

关键字: hash 函数，hash 表，汉语计算，中文处理。

A Two-character Hash Function For Chinese Words

Xiaodan Zhu, Qian Diao & Zhou Joe F

Intel China Research Center

{xiaodan.zhu; qian.diao; joe.f.zhou}@intel.com

ABSTRACT: Using first-character hash function to search words in a dictionary is a widely used algorithm in Chinese computing. However, its disadvantage is obvious. In this paper we put forward a two-character hash function, which decrease the Average Search Length a great many. This algorithm can be used in many fields of Chinese computing.

Keywords: hash function，hash table，Chinese computing，Chinese processing.

1. 引言

通过首字查找单词被广泛地用在汉语计算的各个领域。其基本数据结构如下图所示（图 1）。

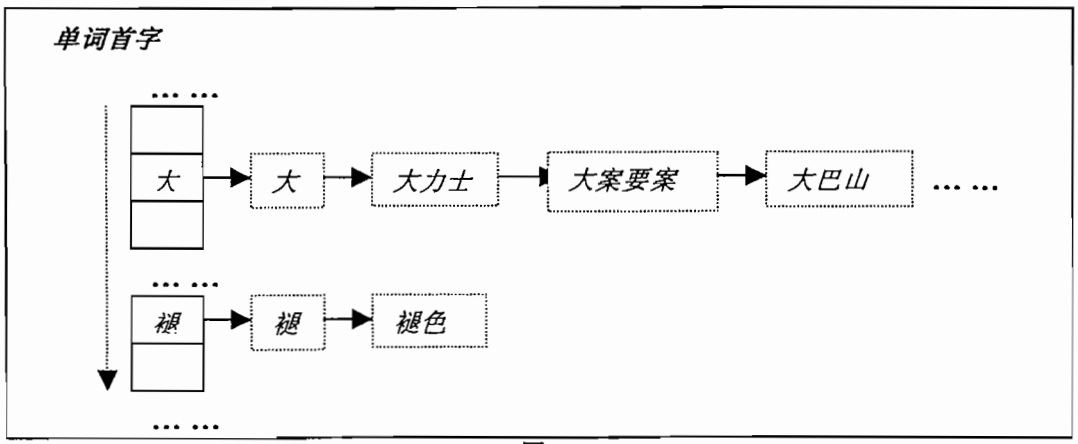

图 1

该方法的主要思想是通过词语首字的内码来找到该词语所在的桶（Bucket），然后使用顺序查找（文献[1]），折半查找（文献[3]）或两者结合（文献[2]）的方法在桶中继续查找词语的具体位置。文献[1][2][3]将它用于快速分词，文献[5]使用该结构进行汉字字串的快速去重，文献[6]基于该结构提出了一种高效的词典组织结构。

然而，该算法有明显的缺陷：词语在 hash 表中分布非常不均匀。例如，对于《现代汉语语法信息词典》（文献[10]）而言，有大约 500 个词语是以“大”字开头，这 500 个词语被存放在同一个桶中，而仅仅两个词语以“褪”字为首字，也就是说对应的桶中只有两个词。这样的不均匀性大大地影响了单词的查找效率。文献[1][2]在计算访问效率时，都假设了 hash 表的数据分布是均匀的，在下面的讨论中我们会看到：这样的假设与实际情况有一定的出入。因此，本文的目的就是在于从一定程度上解决分布的不均匀性，从而提高查找效率。

2. 两字 hash 算法

为了解决上面提出的问题，我们引入了两字 hash 的概念----取词语中两个汉字来进行 hash 映射（如果该词语只有一个汉字，则该汉字被取两次，如：下图中的“褪”）。Hash 表的结构见下图：

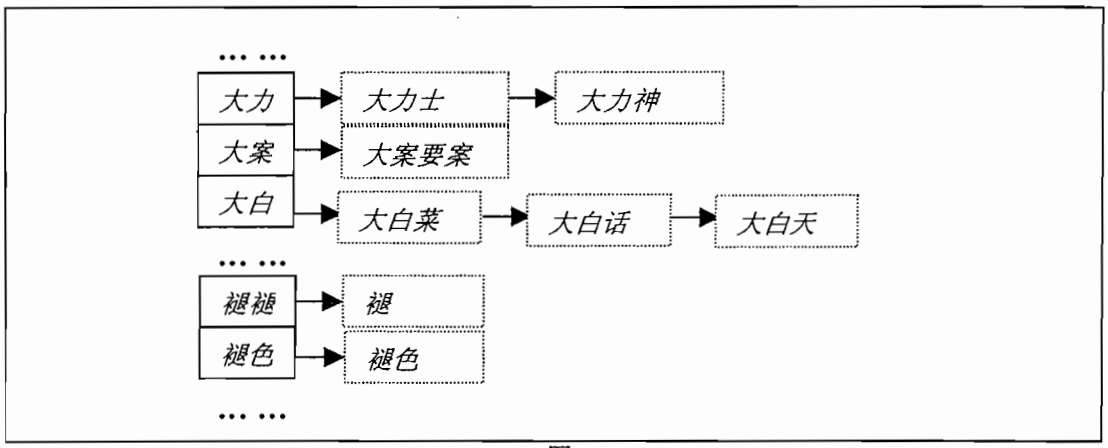

图 2

明显地，图 2 中词语的散列比图 1 好，但图 2 这种简单的两字 hash 需要极大的空间消耗。如果只考虑汉字的一级字库(有 6763 个汉字)，对于图 1，需要开 6763 个桶来存放所有的词语。而对于图 2，如果直接使用汉字内码定址，则需要开 6763*6763 个桶。

我们认为，空间消耗太大正是影响两字 hash 使用的最主要的原因。为了解决这个问题，我们构造了一个 hash 函数，使得桶的数目可以由自由设定。这样，就可以在桶数相同的情况下，比较首字 hash 和两字 hash：哪一种能将单词更均匀地散列到 hash 表中？实验证明：本文提出的算法使得单词的分布比首字 hash 均匀很多。下面是我们构造的 hash 函数为：

Address_in_HashTable=((InnerCode of ch1*InnerCode of ch2)/10) mod N

------Formula (1)

Address_in_HashTable：某词语在 hash 表中的地址（第几个桶）。

Ch1：　　该词语的首字

Ch2：　　该词语的第二字. 如果该词语只有一个汉字，Ch2=Ch1

N　：　　hash 表中桶的数目。用户可自行指定

为了减少不必要的冲突，N 应当取素数。整除 10 的目的是取两汉字内码乘积的中间部分。

InnerCode　　汉字内码.其计算公式为：

InnerCode（ch）=((unsigned char)(HighByte(ch))-176)*94+(unsigned char)(LowByte(ch)) - 161 ;

整个算法的程序流程图见图 3：

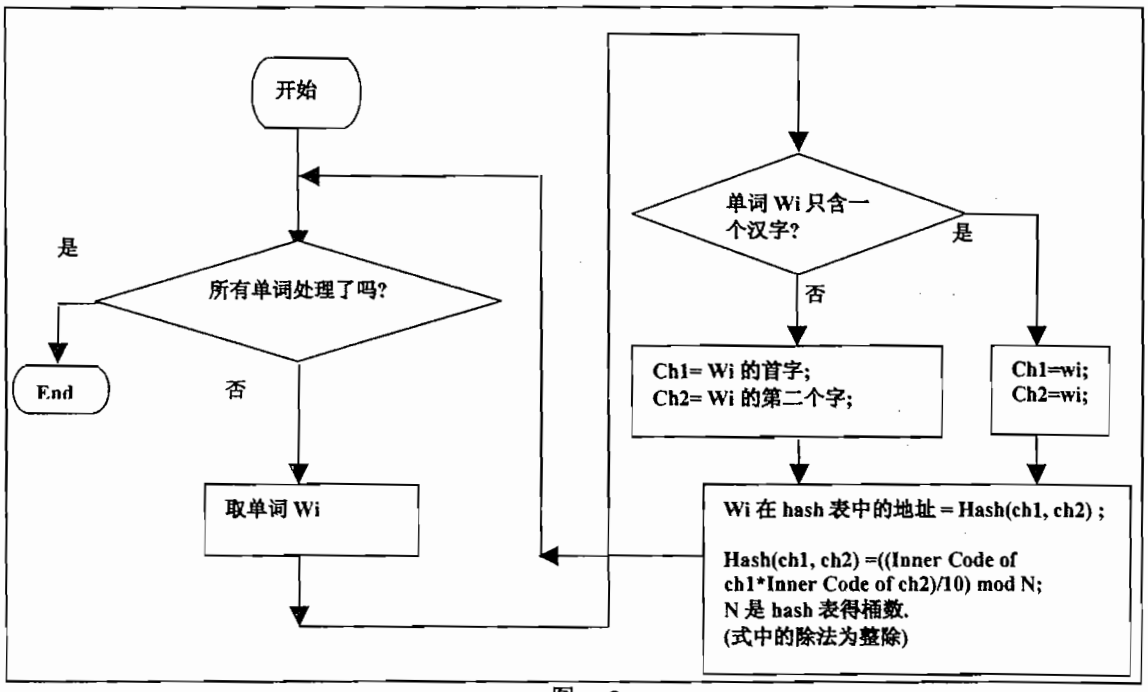

图　3

现在，我们可以在相同的空间占用下比较单词在 hash 表中分布的均匀程度。我们将 formula(1)中的 N 设为 6763，使得首字 hash 表和两字 hash 表的桶数是相同的。我们使用标准方差来度量单词在 hash 表中分布的均匀程度。实验使用三个著名的词典《同义词词林》[8]（简称“词林”），《现代汉语词海》[9]（简称“词海”），《现代汉语语法信息词典》[10]（由北京大学编著，简称“北大词典”）。实验结果见表 1：

标准差	*首字 hash*	*两字 hash*
北大词典	24.4961	*3.69661*
词林	33.7319	*4.37421*
词海	3.18338	*0.969182*

表　1

标准差的计算公式是：

$$SD=\sqrt{\frac{(bn_1-\bar{x})^2+(bn_2-\bar{x})^2+\ldots+(bn_{6763}-\bar{x})^2}{6763}}$$

bn_i：*第 i 个桶中的单词数*. $bn_1+bn_2+\ldots+bn_{pn}=$某词典总词数

$\bar{x}$ ：*桶中平均单词数*, $\bar{x}=\frac{\text{词典总词数}}{6763}$

由上表，我们可以得出结论：在相同的空间占用下，相对于首字 hash，formula（1）将使得汉语单词在 hash 表中分布的均匀程度有很大的提高。例如，对于北大词典，如果使用首字 hash 算法，标准差为 24.5。而对于两字 hash，只有 3.7。词分布的均匀程度直接影响了 hash 表的查找效率，分布越均匀，查找效率越高。

3. 算法的时间效率

我们用平均查找长度来衡量 hash 表的查找效率。下面是平均查找长度的定义：

3．1 平均查找长度

有两种典型的平均查找长度：

(1) 顺序平均查找长度（ASL1）：指的是当找到词语所在的桶之后，使用顺序查找在桶中找到该单词的平均查找长度。

$$ASL1=\sum_{i=1}^{ub}(\frac{bn_i}{wn}*\frac{bn_i+1}{2})$$

bn_i：*第 i 个桶中的词语数目*. $bn_1+bn_2+\ldots+bn_{wn}=wn$.

ub：*非空桶数：含有单词的桶的数目。*

wn：*词表中的词语数目*.

(2) 折半平均查找长度（ASL2）：指的是当找到词语所在的桶之后，使用折半查找在桶中找到该单词的平均查找长度。

$$ASL2=\sum_{i=1}^{ub}\{(\frac{bn_i+1}{wn}Log_2(bn_i+1)-1)\}$$

参数含义同上。

这两种查找方法被用在不同的应用中。

3．2 性能比较

（1）N=6763 时与上面相同，我们将 formula(1)中的 N 设为 6763（GB 中一级汉字的个数），这样，两种方法的存储空间相同。然后我们在三个词典上使用 ASL1 和 ASL2 来衡量两种方法的时间效率。

	桶数（N）	首字hash	两字hash	最优值
北大	6763	32.3477/4.31668	6.41636/2.9438	5.7776/2.5305
词林	6763	42.9268/4.67027	8.11525/3.23961	7.4386/2.8950
词海	6763	3.68671/2.09881	1.4692/1.32138	1/1

表 2

方格中第一个数是 ASL1，第二个是 ASL2。 “最优值”指该词典中的词语完全均匀地分布在 hash 表中时的 ASL1 和 ASL2。从实验结果上看。在相同的存储消耗下，使用两字 hash 比首字 hash 的查找速度有很大提高。 而且很接近“最优值”。

（2）N=词典总词数

如果我们将桶数设置为词典中词条数时，平均查找长度见下表。

	桶数(N)	首字hash	两字hash	最优值
北大	71387	32.3477/4.31668	1.64633/1.41046	1/1
词林	93851	42.9268/4.67027	1.68377/1.42126	1/1
词海	6593	3.68671/2.09881	1.51669/1.34717	1/1

表 3

这里的最优值仍然是指词语完全均匀分布时的平均查找长度。实际上，“两字 hash”在“最好的情况”下也不会达到该值。因为许多词的首尾两字相同，这些词是区分不开的。这里所谓的“最好情况”是指：直接开 6763*6763 个桶(N=6763*6763)，用两个汉字的内码直接将词语映射到 hash 表。下表是“最好的情况”下的平均查找长度：

	桶数(N)	两字hash 理论最优值 （直接以汉字内码定址）
北大	6763*6763	1.42849/1.22394
词林	6763*6763	1.34269/1.18302
词海	6763*6763	1.00015/1.00011

表 4

实际应用中，N 值还可以比表 3 中的大，那样结果会更接近表 4 。

4. 算法的应用

本文提出的算法具有较广泛的用途。该算法可以用来对文献[1][2][3][4]中提出的分词算法进行改进，也可以对[5][6]算法进行改进，以提高他们的时间效率。

文献[1][2]提出的分词算法在计算查找的时间复杂度时都假设词条在 hash 表中是均匀分布的，本算法能使实际结果接近这一假设。文献[4]提出了基于两字簇的汉语快速自动分词算法，该算法根据“在汉语中，两字词大约占 75%”这一统计结果，使用单词的前两字找到该单词所在的节点（第一步查找），然后通过该节点引出的指针继续查找（第二步查找）。第一步查找是该算法的主要耗时环节，时间复杂度为 O(N)，本文提出的算法将使第一步查找的时间复杂度降为常数。

目前，我们使用 formula(1)来创建文档的倒排索引表，以及在该表中进行单词的快速查询。针对该应用，我们使用单词的首尾汉字建立 hash 函数，而不是单词的前两个汉字（hash 表见图 4）。Hash 表的结构是：

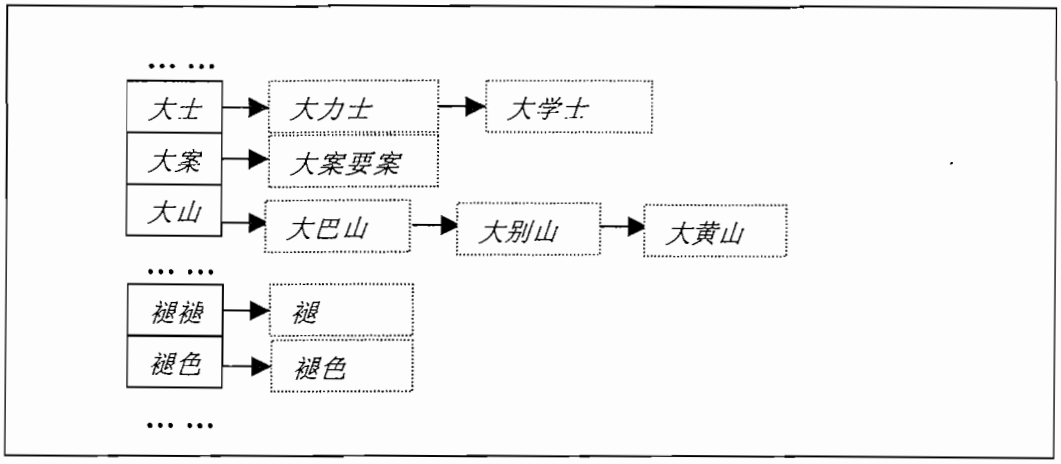

图 4

这样的选择是基于实验得到的结论：对于通用词表（不含过多的地名和机构名），使用首尾两字进行词语的查找，其访问效率比用其它“汉字对”（如本文提出的前两个汉字）要好。直观的理解是：首尾两个汉字比其它“汉字对”对词语有更好的区分力。这些研究我们将另文介绍。

5. 结 论

本文用一个具体的 hash 函数实现了汉语两字 hash 的思想。通过针对三个著名词典进行的实验发现： 在相同的空间效率的前提下，两字 hash 算法相对传统的首字 hash 算法，在时间效率上有很大的改进。而且，hash 表的大小可以自由设定。因此，在汉语计算的很多领域可以取代首字 hash 算法。

参考文献

[1] 吴胜远. 一种汉语分词方法. 计算机研究与发展：第 33 卷，第 4 期，1996 年.

[2] 陈桂林，王永成等. 一种改进的快速分词算法. 计算机研究与发展：第 37 卷，第四期，2000 年 4 月.

[3] 张国煊，王小华，周必水. 快速书面汉语自动分词系统及其算法设计. 计算机研究与发展：第 30 卷，第一期，1993 年 1 月.

[4] 吴祥昊，钟义信. 基于两字簇的汉语快速自动分词算法. 情报学报：第 17 卷，第 5 期，1998 年.

[5] 陈桂林，王永成. 汉字字串的快速去重算法. 情报学报：第 19 卷，第三期，2000 年 6 月.

[6] 陈桂林，王永成等. 一种高效的中文电子词表数据结构. 计算机研究与发展：第 37 卷，第一期，2000 年 1 月.

[7] 严蔚敏，吴伟民. 《数据结构》（第二版）：清华大学出版社，北京，1992.

[8] 梅家驹，《同义词词林》，上海辞书出版社 ，上海，1983.

[9] 倪文杰. 《现代汉语词海》，人民中国出版社 ，1994.

[10] 俞士汶. 《现代汉语语法信息词典详解》，清华大学出版社，广西科学技术出版社 ，1998.