计算机科学 1996 Vol. 23 No. 2

56-57

一种简单、高效的电子词典组织策略

何鸿君 王明昕 史晓东 郑若忠
(国防科技大学计算机系 长沙410073)

TP391.2

摘 要 It is important for electronic dictionary to speed-up searching process, especialy in real time applications. Here, we explain the characteristic distinguishes of an electronic dictionary from conventional databases. For English-Chinese dictionary, a simple organizing structure and effective "hash" algorithms are provided. This strategy has been applied in Matrix English-Chinese Translation System, and produced the desired results.

关键词 Electronic dictionary.

电子词典的应用领域越来越广，不仅可以作为系统的一部分，也可以独立形成产品。事实上，当前市场最大的就是独立的电子词典。

一般说来，电子词典的电子化对象语言的词汇量，不是一个小数目。例如，英语常用词就有二至五万，整个词汇量高达数十万之多，汉字虽然不多，但汉语的词汇量和英语相比却也相差无几。显然，词典规模如此庞大，其搜索效率不能不摆到日程上来，当面对实时应用，如同步翻译电话等时，电子词典的查找效率对提高整个系统的性能就显得格外重要了。

有效、合理地组织词典，加快查找过程，是电子词典必需解决的问题。本文就英-汉电子词典的查找，提出了一种简单、高效的组织策略。该策略已在Matrix英-汉机器翻译系统应用，效果良好。

一、电子词典的特殊性

数据库系统已开发了很多有效的组织结构及相应查询技术，将这些方法、技术借鉴到电子词典里来，当然不失是一种省事的做法，但不能保证查找的高效性和简洁性。仔细分析数据库技术所面向的问题，以及电子词典的特殊性，就会发现电子词典不同于通常的数据库，它们有着本质的区别：

●数据库所包罗的技术范围很广，操作机制往往比较复杂，电子词典只有简单的查询操作，显然不应该带上这些复杂的技术特点；

●数据库面向的是不确定的对象。电子词典所面向的对象是相对固定的，如英-汉电子词典的对象就是所有的英文单词及其对应汉语信息，单词数目和单词是基本不变的。这是电子词典区别于数据库的最重要特征；

●对数据库的查询要求很多，因人而异，需要建立不止一个入口，电子词典的查询单一，单词本身通常就是入口条件信息。

电子词典的特殊性，为提高其搜索效率提供了可能，现在可以将注意力集中到这些固定的单词身上，分析它们的特性，考虑应该采取什么策略了。

二、特征分析方法

单词本身是唯一的查找条件，所以关注的重点就应该是在构造hash函数时突出个性(单词间的不同)，弱化共性(大多数单词拥有的性质)，最初分析单词特性时，我们主要考虑了以下两个方面：1)不同字母在单词中出现的频率；2)不同首字母单词在词典中所占比例。这里应该指出的是，特征分析方法的重要之处在于它提供了一种分析问题的手段，而不仅仅是上面所提到的两个特征。事实上，单词长度也可作为一种分析特征，并且该方法还可以应用到非英-汉词典的组织中去。

为完成电子词典的组织设计工作，专门写了一个分析测试系统，进行单词的特性分析和测试对比。

三、优化组织

1. 分块思想

将整个电子词典按某个标准划分成若干块，分而治之；对于每一个块可采取相同或不同的适当的搜索策略，从而提高整个词典的搜索效率。这就是分块思想。

设词典规模为W，由N块组成，则对于某hash

何鸿君 博士生，研究领域：英-汉机器翻译，计算机支持的协同工作 王明昕 博士，研究领域：英-汉机器翻译，史晓东 博士，研究领域：英-汉机器翻译，郑若忠 教授，研究领域：数据库技术

函数有：

$$C_k=\sum_{i=1}^{w}H_i$$

式中 C_k 表示 hash 值为 k 的单词数量，H_i 表示 i 块中 hash 值为 k 的单词数量。

若单词的 hash 值为 k 的概率相等，对于规模大致相等的 N 块则有如下估计：

$H_i \approx C_k/N$

为简单起见，分块的标准可以是单词的首字母，把词典划分成二十六(不区分大小写)或七十二(区分大小写)个独立的块。我们在分析测试系统中就采取了这一划分标准。

将词典划分成不同的块，直观上带来了明显的一些好处：①分析对象减少，有利于对特征的分析和把握；②可采取一个或多个 hash 函数，非常灵活；③平均冲突次数减少，尤其是最大冲突次数明显减少。如相同的单词集合，按某再 hash 法，最大冲突次数高达数百次之多，而分块法只有几十次。④由于对象数目减少，可以避免采用复杂运算(如除法、长整数乘法等)的 hash 函数，加快运算速度。

2. 优先思想

该想法源于一个统计结果：有一批很小量的词，它们在日常生活和文章中出现的频率远高于其它单词，这些词称为高频率词。显然，将高频率词放在最先可以搜索到的位置具有很好的现实意义，并且实现简单。

四、综合效率分析

实际的查找效率受到各方面因素的影响，查找的时间不仅直接与冲突次数有关，和运算本身的复杂程度亦有很大关系，如有可能则应采用简单的运算。此外，词典的规模、语料的选择也对效率有一定的影响。

为选定 hash 函数，随意找了手头的一个小规模语料库，其中包括了《英语九百句》和《牛津词典》的动词类型的全部例句，然后针对不同的函数和策略进行了测试。测试结果表明这种词典组织策略的效果是比较令人满意的，同时也可以看到查找效率确实是多种因素共同作用的结果。

查找时间与比较次数

算法	时间		次数
	秒	hunds	
hash1	6	43	58656
hash2	4	78	104904
hash3	4	29	98952
hash4	4	17	100736
hash5	4	28	89712
hash6	5	66	133300
hash7	5	5	157888
hash8	6	15	440128
hash9	5	99	414056

Total words: 71568

注：hunds 表示百分秒，时间包括打开文件时间及词法分析时间。

图1 综合测试结果

五、数据结构及 Hash 算法

词典采用了简单的线性多重链表结构，定义如下：

```
struct ClassDictionary
  {ClassWord * header [MAX_DICTIONARY];
                    //hash 表
  ...
  }DicObj;
struct ClassWord
  {char * word; //单词串
  ClassWord * next; //指向具有相同 hash 值的另一个单词
  ...
  };
```

下面就分块和不分块方法分别给出一个 hash 函数的实例：

```
int hash_valuel(char * word_str)
{long c=0;
   short i=1,len=strlen(word_str);
   short index=tolower(word_str[0])-'a';
         int hash;
         while(word_str[i])
           c=c * len+(word_str[i++]-32);
         hash=c%mod_array[index]
              +offset_array[index];
         return(abs(hash));
}
int hash_value7(char * word_str)
{long c=0;
 short i=0,len=strlen(word_str);
 int  hash;
 while(word_str[i])
   c=c+(word_str[i++]-32)*(len-i);
 hash=c%HASH_UNITS;
 return(hash);
}
```

参 考 文 献

[1]郑若忠，《数据库原理》

[2]陈肇雄，《机器翻译研究进展》

[3]史晓东，Matrix 机译系统(硕士论文)

[4] Application of Finite Automata Representing Large Vocabularies, Software Practice & Experience, Vol. 23(1), 1993