动态字母表算术编码*

王忠效, 范植华

(中国科学院 软件研究所,北京 100080)

E-mail: wenux@email.com.cn

http://www.ios.ac.cn

摘要: 研究了动态字母表统计模型的有关性质以及建立动态字母表模型应予以注意的问题. 理论与实验表明,动态字母表模型在没有牺牲时间性能的情况下,能够提高预测的准确性,从而获得更好的编码效率. 动态字母表对于建立大字符集文种(如汉语)文本压缩的统计模型具有重要意义.

关键词: 数据压缩;算术编码;统计模型;汉语文本压缩;编辑距离;局部自适应

中图法分类号: TP391 **文献标识码**: A

算术编码(arithmetic coding)根据信源符号的累积概率对符号进行编码. 这一思想最早见诸Shannon1948年的经典著作[1],但是,现代算术编码的基本思想直到20世纪80年代初才真正清晰. 1987年,Witten等人[2]发表了现代算术编码的第一个纯软件版本,即著名的CACM87. CACM87被广泛加以介绍[3~5],而且被用做H.263的组成部分.

算术编码向基于统计模型的编码系统提供编码器. 统计模型向编码器提供信源符号的概率分布,编码器据此产生被编码符号的比特流. 模型提供的概率越准确,编码器的编码效率越高. 在过去的有关算术编码的文献中,虽然也强调了模型的意义,但多数还是专注于研究编码算法自身——编码效率[6]与时间性能[7~9]以及自适应模型的更新算法的时间性能[2,10]. 与此不同,本文研究基于动态字母表的模型及其性质. 动态字母表模型能有效地提高模型预测的准确性,因而就能够明显地提高编码效率.

定义1. 对于字母表S,如果对于同一范畴中的任意信源中的任意符号s,在任何时刻都有$s\in S$,则称S为完全字母表.

显然,完全字母表的大小是恒定不变的. 如果把计算机中存储的所有文件视为同一范畴内的对象,并把字节作为基本符号单位,则完全字母表S由256个符号组成,每个符号对应于一个字节可能有的全部256种状态中的唯一一种. ASCII表即是这种完全字母表.

定义2. 对于字母表S,如果对于同一范畴中的任意信源中的符号s,在某一时刻t有$s\notin S$,在另一时刻t'有$s\in S$,称S是动态的.

当符号s不在动态字母表S中时,显然S不能描述s. 为了在任何时刻都能描述s,系统必须拥有另一字母表S',使得$s\in S'$. 我们称S'为S的辅助字母表.

研究基于统计模型的汉语文本压缩算法,第1个难题就是如何建立汉语大字符集的统计模型. 撇开文字学上我们甚至根本不可能用来建立汉字的完全字母表不谈,其庞大的字符集本身决定了

* 收稿日期: 1999-04-07; 修改日期: 1999-11-12

基金项目: 国家自然科学基金资助项目(69773023);中国科学院军工重点项目基金资助项目

作者简介: 王忠效(1963-),男,江西南昌人,博士,副教授,主要研究领域为中文信息处理,汉语文本压缩;范植华(1942-),男,江苏南京人,研究员,博士生导师,主要研究领域为并行处理,实时处理.

我们不可能只是简单地平移已经建立在以字节为单位的小字符集之上的数据结构和相应算法.由于汉字在实际使用中具有良好的收敛性,即在具体的篇章中汉字实际上可能仅局限于一个很小的子集;另一方面,不同篇章中的汉字构成的子集不一定相同,我们不得不视汉字字符集为动态字母表.实际上,本项研究的原始目的恰恰是因为探讨基于统计模型的汉语文本压缩算法的需要.

算术编码并不要求对累积概率表按符号概率的大小排序.在自适应模型中,编/解码时必须频繁访问和维护符号的累积概率表,因此,访问和维护累积概率表的时间开销将严重影响编/解码的时间性能.如果对累积概率表按符号概率从大到小排序,则符号的出现频率越高,对累积概率表的访问以及维护的工作量越小.CACM87的自适应模型就是这样实现的,它使得操作累积概率表的工作量减少了一个数量级[5].更有意义的是,CACM87因此给出的数据结构及算法特别适合运用到动态字母表的情形——累积概率表与完全字母表之间简单的映射关系被打乱,动态字母表中符号的机器内码值可以不再连续.鉴于此,我们针对CACM87探讨动态字母表的算术编码问题.

本文仅讨论基于单字节编码的动态字母表的算术编码,而对于基于双字节的GB2312-80编码的汉语文本压缩问题将另文进行详细的讨论.令人振奋的是,这里所讨论的技术仅在0阶模型这个最初级的层次,就已经达到或超过了流行的、基于LZ和Huffman两步压缩过程的通用压缩工具压缩汉语文本的水平.

1 动态字母表与完全字母表的编码效率分析

我们按字节对250万字的汉语语料库进行统计,结果表明,153个符号的覆盖率超过99.99%,199个符号的覆盖率达到100%.对数据压缩领域普遍认可的Calgary语料库中的英语文本文件及计算机源程序文件做统计,100个符号的覆盖率超过99.99%,104个符号则实现完全覆盖.因此,探讨动态字母表模型,不仅仅是对汉语文本压缩有意义.

在CACM87中,字母表包括ASCII的256个符号以及1个流结束符.后者用以输出标明结束信源编码过程的位流,解码器据此终止解码过程.从实现的效率考虑,概率和累积概率用整数表示的频率计数来表示.流结束符的频率不变,恒为1.在整个编/解码过程中,模型跟踪字母表中符号的频率,由于模型必须保证对任何一个符号都能编码,因此必须保证每个符号的频率计数不为0——CACM87将每个符号的频率初始化为1.此外,对于任意长度的信源输入而言,模型必然面临用整数表示的累积频率计数溢出的危险,CACM87在检测到累积频率达到其最大值时,将使全部符号的频率计数折半,并保证每个符号的频率计数最小为1.这样一来,对于不出现在实际信源中的n个符号($n\geqslant 0$),在CACM87的模型中,它们在任何单一时刻都将占有值为n的累积频率.

为便于讨论,约定L表示信源的长度(假定L充分大),F表示累积频率的最大值*,$s_i(i\in[1,256])$表示ASCII完全字母表中的符号,C表示CACM87信源符号的累积频率,并记s_i的频率为c_i.另记S为信源中实际出现的不同符号所组成的字母表.

定义3. 称信源中从累积频率的一次折半过程到下一次折半过程所经历的信源段为一个编辑距离.特别地,视开始时对字母表频率计数的初始化为一次折半过程.

显然,一个编辑距离的长度等于在其中编码的所有符号的计数,即新增累积频率;CACM87的第1个编辑距离的长度$d_1=F-257$.考虑到频率折半时奇数值的余数被舍弃,其他编辑距离的长

* CACM87采用16位整型数表示频率和码字.显然,$F\geqslant 1$.由于编/解码符号时需对F进行乘法或左移位运算,为避免溢出,F的理论最大值为$2^{14}-1$.F可以取其他值,但必须以此为上限.

度 d_i 为

$$\frac{F-n}{2}=F-\left(\frac{F-n}{2}+n\right)\leqslant d_i\leqslant F-\left[\frac{F-n}{2}-n-(257-n)\right]=\frac{F-3n+514}{2}. \tag{1}$$

由式(1),信源包含的编辑距离的数目 m 为

$$\frac{2L-F-3n+1028}{F-3n+514}\leqslant m=1+\left\lceil\frac{L-d_1}{d_i}\right\rceil\leqslant 2-\frac{L-d_1}{d_i}=\frac{2L-2n+514}{F-n}. \tag{2}$$

注意到未出现在实际信源中的 n 个符号的频率,在系统每转移到一个新的编辑距离时实际各增加了 1,到信源编码结束时,它们的累积频率为 mn. 因此,$C=L+mn$.

那么,从 CACM87 出发可计算出根据信源熵编码的总码长(比特)为

$$L'_{\text{code}}=-\sum_{i=1}^{256}c_i\log_2\frac{c_i}{C}=-\sum_{s_i\notin S}m\log_2\frac{m}{C}-\sum_{s_i\in S}c_i\log_2\frac{c_i}{C}=-mn\log_2\frac{m}{C}-\sum_{s_i\in S}c_i\log_2\frac{c_i}{C}.$$

但从动态字母出发(此时未受不存在的 n 个符号的影响),根据信源熵编码的总码长(比特):

$$L_{\text{code}}=-\sum_{s_i\in S}c_i\log_2\frac{c_i}{L}=-\sum_{s_i\in S}c_i\log_2\frac{c_i}{C-mn},$$

于是

$$\begin{aligned}L'_{\text{code}}-L_{\text{code}}&=-mn\log_2\frac{m}{C}-\sum_{s_i\in S}c_i\log_2\frac{c_i}{C}+\sum_{s_i\in S}c_2\log_2\frac{c_i}{C-mn}\\&=-mn\log_2\frac{m}{C}+(C-mn)\log_2\frac{C}{C-mn}\\&=mn\log_2\left(n+\frac{L}{m}\right)+L\log_2\left(1+\frac{mn}{L}\right)>0.\end{aligned} \tag{3}$$

根据式(2),当信源长度 L 充分大时,

$$L'_{\text{code}}-L_{\text{code}}>\frac{nL}{F-n+514}\log_2\frac{F+n}{2}+L\log_2\left(1+\frac{n}{F-3n+514}\right). \tag{4}$$

根据式(3),我们有以下定理.

定理 1. 动态字母表的编码效率一定比完全字母表的编码效率高.

由定理 1,我们容易得到定理 2.

定理 2. 对于充分长的信源,如果用机器整型数表示信源的频率计数,则对于动态字母表统计模型,F(或编辑距离)越大,编码效率越低.

证明:假定信源充分长,则可以设想将较大的编辑距离 S_i 划分为互不重叠的小的编辑距离 S_{ij} ($j=1,2,3,\ldots$),$0<|S_{ij}|<|S_i|$. 考虑按两种编辑距离长度 $|S_i|$ 和 $|S_{ij}|$ 对信源编码(如图 1 所示). 对于后一种情形,我们不妨将 S_i 看做是一个完整的信源,并可以将 S_i 上的动态字母表 E_i 视为全部 S_{ij} ($j=1,2,3,\ldots$) 上的完全字母表. 显然,对于 S_{ij} 上出现的符号构成的字母表 E_{ij},必有 $E_{ij}\subseteq E_i$,即 E_{ij} 构成完全字母表 E_i 上的动态字母表. 由定理 1,用完全字母表 E_i 对 S_i 进行编码,与通过动态字母表 E_{ij} 对 S_{ij} ($j=1,2,3,\ldots$) 编码来实现对 S_i 编码相比,编码效率更低. 定理得证. □

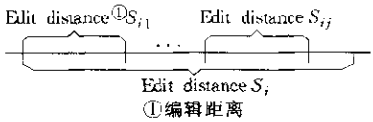


Fig. 1
图1

定理 2 表明,对于动态字母表模型,采用 32 位或更多位的整型数表示频率计数,编码效率未必比采用 16 位整型数更好. 实际上,我们的实验表明,对于以字节为单位的统计模型,取 $F=4096$ 与取 $F=16383$(用 16 位整型数存储累积频率时的最大许可值)相比,几乎总是获得更高的编码效率.

换句话说,定理2使得大字符集(如果模型视单词为基本单位,则任何自然语言的文本皆属于此范畴)对存储空间的要求能够得到更好的保证.

定理3. 对于充分长的信源,如果用机器整型数表示信源的频率计数,则对于完全字母表统计模型,只要 $F\geqslant 9$,F(或编辑距离)越大,编码效率越高*.

证明:根据式(3)有

$$L'_{code}=L_{code}+mn\log_2\left(n+\frac{L}{m}\right)+L\log_2\left(1+\frac{mn}{L}\right). \tag{5}$$

由于在具体信源上 $L_{code}=-\sum_{s_i\in S}c_i\log_2\frac{c_i}{L}$ 为一常量,要证明定理,等价于证明:当信源长度 L 充分大时,F 越大($F\geqslant 9$),式(5)中后两项之和越小.由于 $L>0$,$m\geqslant 1$,$n\geqslant 0$,而根据式(2),F 越大,m 越小,显然,式(5)的第3项加数越小.因此,如果能证明当 L 充分大并且 $F\geqslant 9$ 时 m 越小,式(5)中第2项加数越小,则定理得证.

根据式(2)及定义3,我们考察函数 $f(x)=x\log_2(n+L/x)$ $(n\geqslant 0,L>0)$ 在区间 $[1,(2L-2n+514)/(F-n)]$ 上的性质.由于其导函数为

$$f'(x)=\log_2\left(n+\frac{L}{x}\right)-\frac{L}{(nx+L)\ln 2}\geqslant\log_2\left[n+\frac{L(F-n)}{2L-2n+514}\right]-\frac{1}{\ln 2}.$$

当 $L\geqslant 514-2n$ 时,有

$$f'(x)\geqslant\log_2\left[n+\frac{L(F-n)}{3L}\right]-\frac{1}{\ln 2}=\log_2\frac{F+2n}{3}-\frac{1}{\ln 2}.$$

显然,当 $F\geqslant 9$ 时,恒有 $f'(x)>0$,即 $f(x)$ 在所讨论的区间上为 x 的严格单调增函数.因而在所讨论的区间上,x 越小,$f(x)$ 越小.定理3证毕. □

最后,注意到这样的事实:完全字母表越大,在定长的信源段上实际出现的符号相对于完全字母表越少,因而 n 越大.例如将任何信源放到完全字母表{0,1}上考察,几乎总有 $n=0$.在ASCII表上,对于前述Calgary语料,$n=256-104=152$.而对于汉字大字符集,完全字母表中符号数超过56 000(《汉语大字典》所收汉字数),人们在实际信源中使用到的字数通常远远不到6 000,这时,$n>50000$.而根据式(4)不难得到定理4.

定理4. 一种信源的完全字母表越大,动态字母表模型的编码效率越高.

定理4表明,对于任何大字符集语言(如果视单词为基本的"符号"单位,则任何自然语言皆属于此范畴),如汉语、日语、朝鲜语等,动态字母表模型是基于统计模型的文本压缩算法的必然选择.

2 数据结构

CACM87采用4个顺序表来记录和跟踪模型中的数据变化,通过其中的cumFreq向编/解码器提供信源符号的频率分布.freq:记录信源符号的频率数,与ASCII表(外加一个流结束符)1-1对应.cumFreq:记录各个符号对应的累积频率数,元素与freq元素顺序1-1对应.indexToChar:将检索freq和cumFreq的索引值翻译到对应的信源符号.charToIndex:将信源符号翻译到检索freq和cumFreq的索引值.

动态字母表模型除了继续沿用这些顺序表之外,新引入了一个重要的顺序表charExistStatus,我们将在下面予以介绍.通过它,可以看到动态字母表编码的一些重要思想.这些表一同描述动态

* F 实际上永远不可能取9这样小的值,定理3中可以去掉条件 $F\geqslant 9$.

字母表模型.与动态字母表配套的辅助字母表,也需要同样的数据结构支持.从数据封装以及各自相似的功能上看,我们有时也称动态字母表和辅助字母表为动态字母表的主模型和辅助模型.如果用16位整型数表示频率,主、辅助模型的全部表总共占据比4K字节略多的空间.

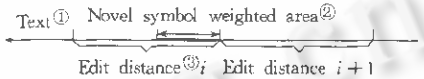
首先,让我们来看看表charExistStatus的作用.

动态字母表模型必然是自适应性质的.它应该能够吸收(或学习)新的符号,并开始对它进行频率计数,使得后续编码过程能够经济地对它进行编码.然而,做到这一点还不彻底,模型的自适应能力还只体现在一个侧面.

每个编辑距离内可能各自拥有不同的字母表,也就是说,在第1个编辑距离内出现的符号可能并不出现在第2个编辑距离内.比如,同一部小说中前后两个章节,甚至同一章节的不同部分,由于话题的变化,用字可能呈现显著的差异.由第2部分的讨论可知,第2个编辑距离最好维护自己的动态字母表,其中不应该保持存在于第1个编辑距离,但不再出现在第2个编辑距离的符号.因此,模型还应该能够溢出(或遗忘)过时的符号.

然而,同一作者在同一篇文字中的用字风格通常总要保持连续性,比如,在编辑距离之间转移时,两个编辑距离内保持不变的符号数与被溢出的符号数相比似乎总是更多(这将在另一篇关于汉字自适应查找算法的文章中加以讨论).而在具体实现方面,当折半频率计数的时候,必须避免新近识别的符号立即溢出,以防在新的编辑距离内被再次识别为新的符号——需要向输出码流中输出额外的信息.总之,符号的溢出并非突发现象,存在一个过程,用"最近最少使用"(LRU)可以很好地描述——这样,符号最新出现的位置信息就至关重要了.

我们决定是否淘汰一个符号,除了主要检查它的频率以外,还要参照它是否出现在一个有效的范围内——我们称之为新符号加权区(如图2所示).当从一个编辑距离进入下一个编辑距离的时候,很可能存在多个新近识别的符号,其频率为1,折半后结果为0.此时,如果判定它们出现在新符号加权区,我们为其加权,以确保它们不溢出,并作为下一个编辑距离内动态字母表的符号.实验结果显示,采用新符号加权区机制与简单地溢出低频符号相比,能够显著地提高编码效率.有趣的是,在一个编辑距离内先期高频出现的符号,虽然在加权区没有出现,但要溢出它们似乎很危险——实验表明,这样做的编码效率几乎总是比不这么做低许多,而且编码速度明显减慢.这也许可以说是对用字风格"连续性"的印证:高频字的死亡要经历更长的过程.



①文本,②新符号加权区,③编辑距离.

Fig.2 Edit distance and novel symbol weighted area
图2 编辑距离与新符号加权区

为了配合新符号加权区,我们引入整数类型的顺序表charExistStatus,其大小与ASCII完全字母表相同,并且下标索引值与ASCII符号之间存在简单的1-1对应关系:Hash(c)=c的ASCII码值,其中c表示ASCII符号.我们在charExistStatus[Hash(c)]中存放符号c最近出现的位置信息.

符号的位置信息可以采用它在信源中的绝对位置来描述,但对于任意长度的信源,则必然面临着整数表示溢出的危险,所以不妥.实际上,完全可以用符号在编辑距离内的偏移地址来表示位置信息,这样自然可以避免整数溢出的问题.但是,最好是用符号最新出现时动态字母表的累积频率来表示,原因很简单,统计模型必须计算和跟踪符号出现时的累积频率,因此,这里只是拿来主义,不增加任何额外的工作量.注意:当从一个编辑距离转入下一个编辑距离的时候,必须重设所有得以保持下去的符号的累积频率为新的编辑距离的起始值.特别地,注意到累积频率值恒为非负数,可以巧借charExistStatus记录、跟踪动态字母表.

新符号加权区则通过其长度值来刻画,没有专门的数据结构相对应,更无须对它进行维护.将charExistStatus[Hash(c)]与该值比较,可以判断符号c是否出现在新符号加权区.显然,0≤加权区的长度≤编辑距离的长度.其中,当加权区的长度等于0时,表明不考虑符号统计特征的连续性,动态字母表单凭频率计数溢出符号;当它取最大值时,动态字母表不能溢出符号.具体取值如何,最好由实验结果来确定.

我们再来看动态字母表与辅助字母表之间的关系.动态字母表的所有数据结构用于记录和跟踪字母表的频率变化;辅助字母表则用作它的一个接应,以保证由动态字母表不能编码的符号最终能够得到编码.实际上,两个字母表之间的关系与变阶模型中的0阶、-1阶模型的关系相似.但是,作为最后一道屏障,-1阶模型是静态的,保持为一个完全字母表.其实,为了实现快速自适应,编辑距离可能变得很短(参见定理2),因此,当从一个编辑距离转入下一个编辑距离时,会有大量符号从动态字母表中溢出,也可能会遇到大量新的符号需要辅助字母表编码,这样,辅助字母表提供的频率分布的准确性将至关重要.因此,能够被动态字母表编码的符号不应该出现在辅助字母表中,即动态字母表与辅助字母表在任何时刻的交集为空,并集为完全字母表.于是,辅助字母表本身也是动态字母表.

3 编/解码过程及分析

编码过程描述如下:

```
WHILE 信源输入未结束 DO
    char←读取下一个符号;
    IF char 在动态字母表中 THEN
        由动态字母表对 char 编码;
        更新动态字母表主模型
    ELSE
        由动态字母表对其结束符编码;
        更新动态字母表主模型;
        由辅助字母表对 char 编码;
        将 char 从辅助字母表移入动态字母表
    ENDIF
    记录 char 出现的最新位置
ENDWHILE
由动态字母表对其结束符编码;
由辅助字母表对其结束符编码
```

解码过程描述如下:

```
WHILE TRUE DO
    char←由动态字母表解码下一符号;
    IF char=动态字母表结束符 THEN
        更新动态字母表主模型;
        char←由辅助字母表解码当前符号;
        IF char=辅助字母表结束符 THEN
            RETURN
        ENDIF
        将 char 从辅助字母表移入动态字母表
    ELSE
        更新动态字母表主模型
    ENDIF
    输出 char;
    记录 char 出现的最新位置
ENDWHILE
```

编/解码过程使用相同的模型,保证二者数据的同步.编/解码之初,动态字母表仅包含一个"虚拟"的字母表结束符;辅助字母表包含全部256个符号以及1个同样是"虚拟"的字母表结束符.两个结束符的"ASCII"值都是256(ASCII值∈[0,255]).

动态字母表结束符对应于变阶统计模型中由n阶模型转入$n-1$阶模型编码的转义符.关于转义符的频率的估计,是一个很有意义的问题,文献[11]讨论了几种不同的估算方法.我们将动态字母表结束符的频率初始化为1.不像CACM87,我们完全像对待其他符号一样(必须保证其值大于0),每对它编码一次,均增加它的频率.显然,当连续出现不在动态字母表中的新符号时,它的频率迅速增加,从而对后续新符号产生好的编码效果;当一段时间很少、甚至不出现新符号时,随着其他符号频率的增长,它的频率相对降低,因而,对编码效率的影响逐渐下降.这种方式既简单,而且看

起来似乎工作得又很好.

更新模型的算法是自适应模型的关键所在.影响算术编码速度的关键因素是编码自身涉及大量的乘法运算;而对于自适应模型来讲,对模型进行更新的工作量主要表现在当从一个编辑距离转入下一个编辑距离时对频率的折半操作,其工作量相当惊人,可能涉及对所有表的维护.由定理2,动态字母表要获得较好的编码效率,编辑距离不能取最大值,似乎越小越好.然而,在实际实现时,一方面,过小的编辑距离意味着需要过于频繁地更新模型,算法的时间性能将难以保证;另一方面,此时新符号加权区(如图2所示)必须收缩,新近出现、然后被立即溢出的符号增多,从而动态字母表中只能保存很少量的符号,结果,在下一个编辑距离内将大量出现动态字母表所不能编码的符号.根据上述编码过程,系统将输出相应的转义码,转而求助辅助字母表,从而严重损害编码效率.折衷的做法是选取一个适当大小的编辑距离,这又体现在选取一个合适的"最大"累积频率 F.实验表明:对于以字节为单位的统计模型,取 $F=4096$ 较为理想.

更新模型的作用包括:增加输入符号的频率计数,递增字母表的累积频率;如果累积频率达到最大许可值 F,还将对频率计数作折半操作.无论是因为增加符号的频率计数从而导致需要调整频率表 freq 元素间顺序,还是溢出过时的符号,都需要维护表 charToIndex 和 indexToChar.也就是说,更新模型可能涉及所有的数据结构,但全部工作量几乎都集中在对表 freq,cumFreq 的维护上.

动态字母表模型与 CACM87 相比,当更新模型的时候只在发生符号溢出时产生新的工作量.注意到被溢出的符号的频率计数总是1,因此,所有被溢出的以及所有在加权区新出现且仅出现一次的符号,都连续地处于表 freq 和 cumFreq 的尾端.这表明对于适当规模的字母表来讲,因为溢出而产生的维护工作量相对于维护整个表不会很大;特别是,溢出多个符号与溢出1个符号相比,维护工作量甚至更少,因为表中需要向前移动、调整符号与索引间映射关系以及需要计算累积频率的元素减少了.另一方面,虽然溢出过程是 CACM87 所没有的,但 CACM87 维护的是一个完全字母表,其大小甚至可能是动态字母表的若干倍(比如,对于 Calgary 语料中英语文本而言,在自适应编码过程中的任何单一时刻,动态字母表所包含的符号个数≤104,也就是说,完全字母表至少是它的2.46倍),所以,在通常情况下,我们可以设想维护较小的表所减少的工作量应该足够抵消溢出过程所导致的工作量(见表1).

4 实验结果

将1987年建立的 Calgary 语料库用于测试无损压缩算法已经获得广泛的认可.目前,一个新的 Canterbury 测试语料库正在开发过程中[12].这两个语料库可被质疑的地方是显然的,比如,都没有包括大字符集文种的文本.然而,从可比性及被认可的程度来看,它们仍不失为首选.表1是用它们进行测试的结果.

表1表明,动态字母表比 CACM87 每个符号平均少输出0.0775比特,编码效率提高1.89%.另一方面,动态字母表在编码的时间性能方面比 CACM87 略差,解码则略强,但是这种差别小到可以忽略不计.这与前面的分析相符.实际上,任何一个数据压缩算法都会是数据敏感的.从《邓小平文选》中随机选取9篇稍长的文件(大于16K 字节)进行测试,无论是编码还是解码,动态字母表的时间性能始终保持优势.

表1中有两个现象需要解释.首先,CACM87 和动态字母表的编码效率可能突破信息论熵的极限.实际上,CACM87 能做到这一点,恰恰是在跨越编辑距离时对频率计数进行折半操作所带来的意外收获.折半操作降低了一个编辑距离内的统计特征对远距离的其他编辑距离内统计特征的影

响,使得后者的统计更少偏离自身的真实情形,也就是说,系统的自适应能力比所预想的更好.而根据定理2和定理3,动态字母表的最佳编辑距离的长度比完全字母表要小,其自适应速度自然更快.

Table 1 Experimental results with Calgary and Canterbury corpuses

表1 针对 Calgary 和 Canterbury 语料库的实验结果

File name①	Bytes②	Entropy③	Bits/symbol④		Coding efficiency⑤		Compression ratio⑥		Encoding time (s)⑦		Decoding time (s)⑧	
			87	DAC	87	DAC	87	DAC	87	DAC	87	DAC
geo	102 400	5.646 4	5.656 3	5.664 6	0.998 2	0.996 8	1.414	1.412	0.230	0.241	0.260	0.270
obj1	2 1504	5.948 2	5.966 5	5.698 3	0.996 9	1.043 8	1.341	1.404	0.050	0.050	0.060	0.050
obj2	246 814	6.260 4	6.070 8	5.905 7	1.031 2	1.060 1	1.318	1.355	0.551	0.611	0.611	0.581
pic	513 216	1.210 2	1.166 0	1.104 8	1.037 9	1.095 4	6.861	7.241	0.440	0.471	0.501	0.490
bib	111 261	5.200 7	5.233 7	5.220 6	0.993 7	0.996 2	1.529	1.532	0.211	0.220	0.230	0.211
book1	768 771	4.527 2	4.546 3	4.542 0	0.995 8	0.996 7	1.760	1.761	1.322	1.372	1.412	1.362
book2	610 856	4.792 6	4.776 5	4.736 6	1.003 4	1.011 8	1.675	1.689	1.101	1.142	1.122	1.121
news	377 109	5.189 6	5.186 2	5.139 1	1.000 7	1.009 8	1.543	1.557	0.721	0.741	0.751	0.741
paper1	53 161	4.983 0	4.984 1	4.909 3	0.999 8	1.015 0	1.605	1.630	0.101	0.100	0.100	0.100
paper2	821 99	4.601 4	4.626 2	4.609 9	0.994 6	0.998 2	1.729	1.735	0.140	0.151	0.150	0.140
paper3	46 526	4.665 1	4.710 1	4.684 2	0.990 4	0.995 9	1.698	1.708	0.080	0.080	0.090	0.090
paper4	13 286	4.699 7	4.815 9	4.732 8	0.975 9	0.993 0	1.661	1.690	0.030	0.021	0.030	0.030
paper5	11 954	4.936 2	5.058 7	4.971 1	0.975 8	0.993 0	1.581	1.609	0.030	0.050	0.020	0.020
paper6	38 105	5.009 5	5.003 4	4.878 1	1.001 2	1.026 9	1.599	1.640	0.080	0.070	0.070	0.070
progc	39 611	5.199 0	5.234 9	5.162 0	0.993 1	1.007 2	1.528	1.550	0.080	0.081	0.070	0.080
progl	71 646	4.770 1	4.758 8	4.683 5	1.002 4	1.018 5	1.681	1.708	0.120	0.150	0.130	0.130
progp	49 379	4.868 8	4.894 2	4.814 5	0.994 8	1.011 3	1.635	1.662	0.091	0.090	0.100	0.090
trans	93 695	5.532 8	5.492 4	5.394 1	1.007 4	1.025 7	1.457	1.483	0.180	0.190	0.251	0.180
BIBLE.TXT	4 077 775	4.373 8	4.382 7	4.370 9	0.998 0	1.000 7	1.825	1.830	6.890	7.521	7.411	7.510
E.COLI	4 638 690	1.999 8	2.029 5	1.996 8	0.985 4	1.001 5	3.942	4.006	5.167	5.518	6.039	5.929
WORLD192.TXT	2 473 400	4.998 3	4.998 7	4.972 9	0.999 9	1.005 1	1.600	1.609	4.597	4.917	4.917	4.897
ALICE29.TXT	152 089	4.567 7	4.593 9	4.581 0	0.994 3	0.997 1	1.741	1.746	0.301	0.271	0.260	0.270
ASYOULIK.TXT	125 179	4.808 1	4.839 3	4.821 5	0.993 6	0.997 2	1.653	1.659	0.230	0.230	0.231	0.230
CP.HTM	24 603	5.229 1	5.299 5	5.261 8	0.986 7	0.993 8	1.510	1.520	0.050	0.050	0.050	0.051
FIELDS.C	11 150	5.007 7	5.135 8	5.023 9	0.975 1	0.996 8	1.558	1.592	0.020	0.030	0.020	0.020
GRAMMAR.LSP	3 721	4.632 3	4.942 8	4.785 8	0.937 2	0.967 9	1.619	1.672	0.010	0.010	0.010	0.010
KENNEDY.XLS	1 029 744	3.573 5	3.368 2	3.260 8	1.060 9	1.095 9	2.375	2.453	1.582	1.762	1.743	1.753
LCET10.TXT	426 754	4.669 1	4.660 9	4.633 1	1.001 8	1.007 8	1.716	1.727	0.731	0.781	0.771	0.781
PLRABN12.TXT	481 861	4.531 4	4.555 9	4.553 8	0.994 6	0.995 1	1.756	1.757	0.841	0.861	0.842	0.871
PTT5	513 216	1.210 2	1.166 0	1.104 8	1.037 9	1.095 4	6.861	7.241	0.441	0.470	0.501	0.511
SUM	38 240	5.329 0	5.175 5	4.761 3	1.029 7	1.119 2	1.546	1.680	0.080	0.080	0.080	0.080
XARGS.1	4 227	4.898 4	5.180 0	5.049 4	0.945 6	0.970 1	1.544	1.584	0.010	0.010	0.011	0.010
AVERAGE⑨	539 129	4.620 9	4.640 9	4.563 4	0.997 9	1.016 8	2.027	2.076	0.828	0.886	0.901	0.896

①文件名,②节数,③熵,④比特/符号,⑤编码效率,⑥压缩比,⑦编码时间(秒),⑧解码时间(秒),⑨平均.

注:(1) 简称 CACM87为87、动态字母表算术编码为 DAC. (2) 硬件环境:Celeron 300A,128M RAM;软件环境:Windows NT Workstation 4.0. (3) 编码效率=熵÷每符号实际编码比特[13],压缩比=信源长度÷信源编码后长度. (4) CACM87取 $F=16383$,动态字母表取 $F=4096$,新符号加权区长度=$F\div 2$. (5) 动态字母表折半除数=4. (6) Calgary 语料库文件名用小写字母标识.此外,各项"平均"系简单均值,未依文件长度做加权处理.

实际上,为了降低一个编辑距离内的统计特征对后继编辑距离的影响,我们不妨将"折半"操作理解为将频率计数除以除数 $d(d=2,3,4,\ldots)$.表2仍是针对上述语料库的实验结果.虽然动态字母表的编码效率在 $d=4$时比在 $d=6$时低0.006%,考虑到此时可以用位移运算替代除法运算,表1中动态字母表取 $d=4$(CACM87取 $d=2$).此时,动态字母表模型的平均编码效率比 $d=2$提高了

0.119%,完全字母表模型仅提高0.019%.当d逐渐增长时,两个模型的平均编码效率均大幅度下降.究其原因,整除导致动态字母表中频率介于1与$d-1$之间的符号被溢出,显然,d越大,被溢出的符号越多,从而许多在一个编辑距离内频率稍低但出现在下一编辑距离的符号,在下一编辑距离只得作为新符号编码,进而降低编码效率;而在完全字母表方面,d越大,意味着当从一个编辑距离转入下一个编辑距离时,不在字母表中出现的符号的频率被放大得越多,根据第2节的分析可知,模型的编码效率损失也就越大.当$d=4$时,完全字母表的平均编码效率之所以有微弱提高,其实是因为快速自适应带来的效益能够恰好抵消频率被适当放大而产生的负面作用.

Table 2 Coding efficiency and divisor for "halving" frequencies

表2 折半除数对编码效率的影响

Divisor① d	Average coding efficiency of whole alphabet②	Average coding efficiency of dynamic-alphabet③
2	0.997 93	1.015 65
3	0.997 98	1.016 45
4	0.998 12	1.016 84
5	0.997 87	1.016 86
6	0.997 54	1.016 99
7	0.997 53	1.016 13
8	0.997 14	1.016 33
9	0.945 64	0.970 45

①折半除数,②完全字母表平均编码效率,③动态字母表平均编码效率.

其次,针对文件geo(地球物理数据),尽管动态字母表的编码效率高达0.996 8,但不及CACM87——虽然只差微弱的0.001 4.这是否与定理1矛盾?原来,定理1只是纯理论上的阐述,没有涉及具体实现时可能需要向输出码流中插入额外的转义码的问题.由公式(4),如果动态字母表越大,即n越小,动态字母表所呈现的优势越小.在具体实现时,由于需要输出转义码,当字母表逼近完全字母表时,转义码的累积作用甚至有可能使得动态字母表的编码效率比不上完全字母表.文件geo的统计结果显示:前90个符号的累积概率为85.01%,前230个为99.05%,频率最低的第256个符号出现18次.可见,此时$n=0$为最小值.当然,信源中出现全部256个符号并不是,甚至根本就不是动态字母表编码效率降低的原因.如果低频符号分散在整个信源,而不是集中出现在一个或少数几个编辑距离内,那么,在最坏的情况下,它们在一个编辑距离的符号加权区之外作为新符号被动态字母表吸纳,编码器付出了输出转义码的代价,在没有获得任何补偿的情况下,随着转入下一个编辑距离,它们又被溢出.如果这种现象累积到一定的程度,动态字母表编码的效率势必退居完全字母表之下.当然,也正因为其低频,以致甚低频,这些符号对整体编码效率的影响必然局限在一个很小的范围内,所以,动态字母表的编码效率依然很高.

5 进一步的工作

第4节曾谈到由于采用较短的编辑距离,动态字母表的自适应速度加快.遗憾的是,按照本文提供的编码算法,当编辑距离过小(比如$F=2000$)时,编码效率非但不能提高,反而受到严重损害.试想编码一个1K字节的C++源程序文本.假如$F=2000$,显然,编码过程将不发生折半操作.设想文本中将出现由80个连续的符号"/"构成的字符串S的情形,并假定文本中没有其他的"/".如果S出现在文本的开始位置,S将获得高效率的编码(不到5个字节),但是,符号"/"将以高达7.81%的概率影响后继944个符号的编码质量;如果出现在文本的末尾,编码器将用低于1/945、2/946、3/947、...的概率对符号"/"编码,也就是说,S得不到有效的编码.事实上,我们完全有理由既要求

对 S 高效编码,又要求 S 对编码其他符号产生最小的影响.这就要求模型是局部自适应的.

我们定义一个模型是局部自适应的,要求模型:(1) 保证在较短的信源段上的数据得到高效率的编码;(2) 一个信源段上的数据对信源的其他部分施加的影响应该很小、甚至没有.然而,无论从理论上还是从实现的难度上看,完全的局部自适应模型都是不存在的.但是,要实现针对具有普遍意义的诸如游程等现象的、适度的局部自适应应该是可以的,而且应该成为建立统计模型时追求的一个目标.

References:

[1] Shannon, C.E. A mathematical theory of communication. Bell System Technical Journal, 1948,27(3):379~423.

[2] Witten, I.H., Neal, R.M., Cleary, J.G. Arithmetic coding for data compression. Communications of the ACM, 1987,30(6):520~541.

[3] Gao, Wen. Multimedia Data Compression Techniques. Beijing: Electrical Industry Press, 1994 (in Chinese).

[4] Bell, T.C. Text Compression. New Jersey: Prentice Hall, Inc., 1990.

[5] Nelson, M. The Data Compression Book. California: M & T Books, 1991.

[6] Xue, Xiao-hui, Gao, Wen. Improved arithmetic coding algorithm. Chinese Journal of Computers, 1997,20(11):966~973 (in Chinese).

[7] Rissanen, J., Mohuiddin, K.M. A multiplication-free multialphabet arithmetic code. IEEE Transactions on Communication, 1989,37(2):93~98.

[8] Chevion, D., Karnin, E.D., Walach, E. High efficiency, multiplication-free approximation of arithmetic coding. In: Storer, J.A., ed. Proceedings of the Data Compression Conference (DCC'91). Washington, DC: IEEE Computer Society Press, 1991. 43~52.

[9] Printz, H., Stubley, P. Multialphabet arithmetic coding at 18 Mbytes/sec. In: Storer, J.A., ed. Proceedings of the Data Compression Conference (DCC'93). Washington, DC: IEEE Computer Society Press, 1993. 128~137.

[10] Jones, D.W. Application of splay trees to data compression. Communications of the ACM, 1988,31(8):996~1007.

[11] Witten, I.H., Bell, T.C. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory, 1991,37(4):1085~1094.

[12] Arnold, R., Bell, T.C. A corpus for the evaluation of lossless compression algorithms. In: Storer, J.A., ed. Proceedings of the Data Compression Conference (DCC'97). Washington, DC: IEEE Computer Society Press, 1997. 201~210.

[13] Jin, Fan. Information Theory and Encoding Methods. Beijing: Chinese Railway Press, 1990 (in Chinese).

附中文参考文献:

[3] 高文.多媒体数据压缩技术.北京:电子工业出版社,1994.

[6] 薛晓辉,高文.改进的算术编码.计算机学报,1997,20(11):966~973.

[13] 靳蕃.信息论与编码方法.北京:中国铁道出版社,1990.

Dynamic-Alphabet Arithmetic Coding*

WANG Zhong-xiao, FAN Zhi-hua

(*Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: wenux@email.com.cn

http://www.ios.ac.cn

Abstract: This paper addressed the features of dynamic-alphabet model for arithmetic coding and problems pertaining to model building. Both theory and experiments show that without loss of time performance, the dynamic-alphabet model provides more accurate prediction and consequently has better coding efficiency. Dynamic alphabet is a fresh but key concept to the building of statistical model of text compression for any natural language of large alphabet set, such as Chinese.

Key words: data compression; arithmetic coding; statistical model; Chinese text compression; edit distance; local adaptation

* Received April 7, 1999; accepted November 12, 1999

Supported by the National Natural Science Foundation of China under Grant No. 6973023; the Military Industry Key Project of the Chinese Academy of Sciences