文章编号:1006-544X(2002)04-0425-05

基于单词的 Huffman 压缩方法

陈基漓[1], 严小卫[2], 杨 祥[1]

(1. 桂林工学院 电子与计算机系, 广西 桂林 541004; 2. 广西师范大学 计算机科学系, 广西 桂林 541004)

摘 要: 基于单词的文本压缩算法是一种新型的算法, 它与传统文本压缩算法的主要区别是以单词取代单个字符作为处理对象. 介绍了基于单词的 Huffman 压缩算法, 给出了多叉 Huffman 树的构造算法, 用一个例子进行了说明, 比较了两种不同宽度的 Huffman 编码树. 宽度为 256 的算法使用整个字节进行编码, 而宽度为 128 的算法只使用一个字节中的 7 位进行编码, 故前者具有更好的压缩率, 并通过实验进行了说明.

关键词: 基于单词的压缩算法; Huffman 编码; Huffman 树

中图分类号: TP391; TP274 **文献标识码**: A①

传统的文本压缩算法多以字符作为基本处理单位, 基于字符的压缩方法具有简单容易实现的特点. 首先由于出现在文本中的单个字符的个数是有限的, 基于字符的压缩方法中使用的一些辅助结构就相对简单而且便于维护, 如 Huffman 压缩方法中构造的 Huffman 树, LZW 方法中构造的字典. 其次在基于字符的压缩方法中, 每次处理的都是数据源中的下一个字符, 直接从数据源中读取即可, 不涉及任何其它的操作. 由于以上原因, 如果压缩的目的主要是为了减少数据源的存储空间, 以及传输的数据, 基于字符的压缩算法是一种很好的选择. 对于自然语言的文本来说, 从检索、理解等各个方面来考虑, 以单词作为基本单位比以字符作为基本单位更符合人们的使用习惯以及自然语言本身的特点, 而且以单词作为基本元素考虑, 还可以使用语言中单词之间的相关性, 单词相对字符而言, 更具有倾向性[1]. 在这样的背景下, 对于以自然语言为基础的文本进行压缩, 更好的选择是以单词替代字符作为每次处理的一个基本单位. 基于单词的压缩方法算法的主要思想与基于字符的压缩方法是相同的, 只是在压缩过程中每次处理的是一个单词.

本文介绍了基于单词的 Huffman 编码压缩方法, 指出基于单词的 Huffman 压缩算法适合采用按字节运算的编码模式, 在二叉 Huffman 树的构造算法基础上给出了多叉 Huffman 树的构造算法, 并以一个四叉树的构造举例说明了多叉 Huffman 树的构造算法, 最后比较了两种不同宽度的 Huffman 编码树.

1 基于单词的 Huffman 编码压缩方法

算法的实现与基于字符的 Huffman 编码方式是一样的, 也可以采用静态和动态两种方式. 常规的 Huffman 编码方法采用的是按位运算的方式, 以存储空间的最小单位“位”来作为编码的最小单位, 每个字符根据编码确定所需要的位数, 这种方式的好处是能够充分利用存储空间. 以位运算为基础的 Huffman 编码正好是二叉树的最好实现途径, 每一位上的“0”、“1”两种状态恰好对应一个结点的两个分支. 由于每个字符的编码长度是不固定的, 在解码或查找的过程中, 需要进行位运算来获得不同字符的编码, 需要频繁地进

① **收稿日期**: 2002-06-19; **修订日期**: 2002-08-19

基金项目: 中国科学院计算技术研究所智能信息开放实验室开放课题 (IIP2001-4)

作者简介: 陈基漓 (1972-), 女, 广西兴安人, 讲师, 硕士, 主要研究方向: 信息检索及数据库.

行移位和位屏蔽运算，在运算上要花费一定的时间，使得解码的效率降低. 基于单词的压缩方法中，单词个体较多，会导致构造的 Huffman 树深度增大，使得对应深层结点单词的编码位数过大，按位运算的 Huffman 编码方式更适合于以字符为处理对象压缩方法.

E. Moura 等人提出了按字节运算的 Huffman 编码方式[2,3]，以字节作为编码的最小单位，用整字节取代位来对字符进行编码. 实际上是将传统 Huffman 编码中的 Huffman 树用多叉树的形式来实现，以多叉树取代二叉树. 按字节运算的 Huffman 编码方式，每个结点对应的编码总是整个的字节，树中的第一层结点用一个字节表示，第二层结点用二个字节表示……. 对压缩文本进行解码时只需要将压缩信息按整字节的形式读出，不需要进行移位和屏蔽位的运算，使得解码的效率大为提高. 使用按字节运算的 Huffman 编码方式的另一个好处是，可以支持直接在压缩的文本上进行快速的查找. 将按字节运算的 Huffman 编码应用于以字符为主体的模式中，由于字符个数太少，而所有字符的编码最少也需要一个字节，会造成编码中多余的位出现，使得多余的位比较多，压缩率比按位运算的编码方式要稍低. 而对于以单词为主体的模式中，以整个字节表示 Huffman 树中的一个结点，正好降低了树的深度，增加了宽度，单词的个数很多，对宽度的增加，不会对压缩比有明显的影响，按字节运算的 Huffman 编码方式，更适合于基于单词的压缩模式.

在基于单词的模式中应用以字节运算为基础的 Huffman 编码方式，单词可以用不同的整数个字节表示. 虽然压缩效率不如按位运算的 Huffman 编码，但因为往往只用一个或两个字节来表示单词，这远比单词所占的实际字节数要少得多，所以压缩效率还是很可观的，特别是当文本中较长的单词出现的次数比较多的时候.

2 多叉 Huffman 树构造算法

以字节为基础的 Huffman 编码方式中，关键的问题是 Huffman 树的构造，如何使构造的 Huffman 树所形成的编码是最优的形式. 如果直接采用构造二叉 Huffman 树的算法来产生多叉 Huffman 树，会导致编码形式不是最优[3]. 下面以简单的四叉 Huffman 树（即一个结点的子结点个数最多为 4 个，结点的最大宽度为 4）为例进行说明.

2.1 一个简单的实例

算法思想为：先取 4 个频率最低的单词为一组，形成一个新的符号，它们的频率和作为新符号的频率，从剩下的单词和符号中再选取 4 个频率最低的形成一组，…，重复以上操作，每次形成的新符号作为下一次操作时一个独立的单词处理，直到将所有的符号处理完毕.

为简单起见，假设每个单词出现的频率是相同的，共有 18 个单词，分别用 1，2，…，17，18 表示，用传统的处理方法构造四叉 Huffman 树过程如下.

先选取 1，2，3，4 四个单词形成一组，该组的频率值为 4，在剩下的 14 个单词加 1 个符号中选择四个频率最小的，依次为 5，6，7，8 形成一组，…，一共形成了 4 个组，还剩下两个单词 17，18，如图 1 所示. 下一步选择 17，18 两个单词，再加上频率为 4 的两个组作为新的一组，频率为 4 +4 +1 +1 =10，此时只剩下 3 个符号，频率分别为 4，4，10，将三个符号与一个空结点组成一组，形成 Huffman 树的第一层结点，所有符号及单词都处理完毕，得到的 Huffman 树如图 2 所示.

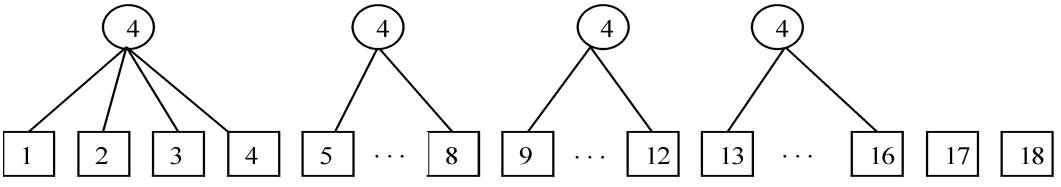


图 1 四叉树结点的分组过程

Fig. 1 Grouping process of multi – fork tree's nodes

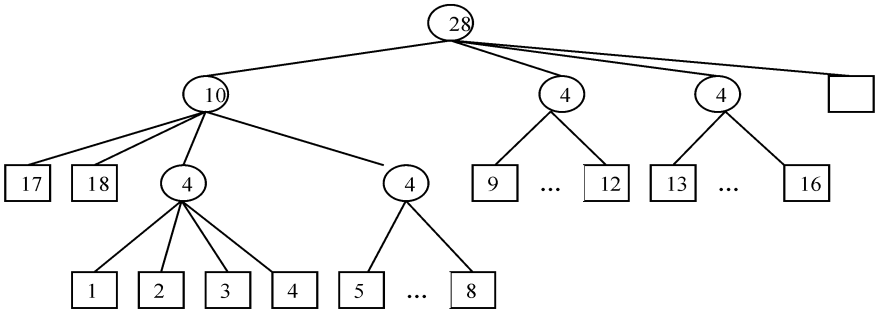


图 2　传统方法得到的四叉 Huffman 树

Fig. 2　Multi – fork tree in traditional method

按照以上算法得到的 Huffman 树，为四叉树形式，每个结点需要用 4 位表示. 可以看出，编号为 1，2，3，4，5，6，7，8 的单词位于树中的第三层，这些单词的表示需要 6 位，编号为 9，10，…，17，18 的单词位于树中第二层，需要用 4 位表示. 树中的第一层出现了一个空结点，如果将任意一个单词结点放在空结点处，得到的表示方法都会比这种形式要好一些. 主要原因是没有充分利用第一层的结点，按照 Huffman 树的原则，应该尽可能地使频率高的结点靠近根结点，频率低的结点远离根结点. 出现的空结点应该在最后一层，而不是在第一层.

解决这个问题的方法是设法使第一层的空结点移至最后一层. 首先确定可能产生空结点的个数，由公式 $4-[1+(n-4)\bmod 3]$[3]得，在以上例子中空结点的个数为 $4-[1+(18-4)\bmod 3]=1$. 空结点要置于最后一层，即最后一层中必定存在这样的一组点，包括一个空结点和代表三个频率最低的单词的结点. 即先选取频率最小的三个结点和一个空结点构成一组，形成一个新的符号，从剩下的 15 个单词和 1 个新符号中继续选择 4 个频率最小的作为一组，……，直至将所有单词与符号都处理完毕为止. 上述例子改进的算法产生的 Huffman 树如图 3 所示.

可以看出，用改进算法得到的 Huffman 树，只有编号为 1，2，3 的单词需要用 6 位表示，其余 15 个单词都只需要用 4 位表示，表示单词的平均位数要比原来的算法低.

2.2　算法的具体描述

将以上改进的思路用于结点最大宽度为 256 或 128 的 Huffman 编码算法中，可以得到改进的算法如下：（以宽度为 256 的情况表述算法）

（1）将各个单词出现的频率进行排序.

（2）利用公式 $m=256-[1+(n-256)\bmod(256-1)]$，（其中 n 为不同单词的个数），求出 Huffman 树中出现的空结点数 m，得离根结点最远的结点包含的空叶结点个数为 m，该结点中包含的非空叶结点为 $256-m$，依次选取 $256-m$ 个频率最小的单词作为结点，与 m 个空结点组成 Huffman 树中离根结点最远的结点的子结点，并将该结点

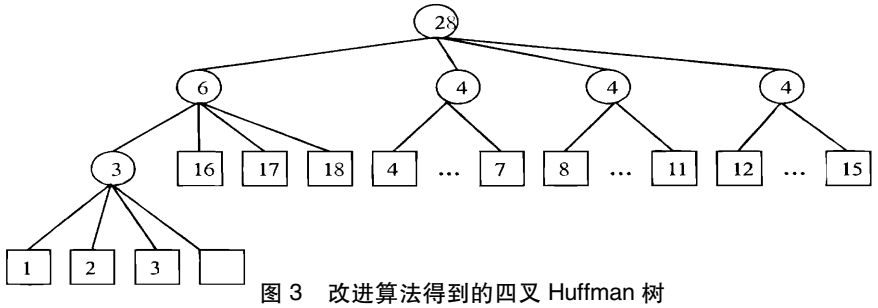


图 3　改进算法得到的四叉 Huffman 树

Fig. 3　Multi – fork tree from method improvement

当作一个符号，该结点中所有子结点的频率和为该结点所代表符号的频率，该结点与剩余的单词作为独立的符号进入下一步.

（3）从剩余的独立符号中依次选取频率最小的 256 个符号作为一个新结点的子结点，同样将所有子结点的频率和作为新结点的频率，新结点与剩余的独立符号进入下一步. 若所有的独立符号都处理完毕，则转（4），若还没有处理完，则转（3）.

（4）根据得到的 Huffman 树，对所有的单词进行编码，每个结点的代码用一个字节表示，代码为该结点在其父结点所有的子结点中的位置，规定从左向右依次为 1，2，…，256. 从根结点出发，到达每一个非空的叶结点，所经过的路径中各个结点的代码组成叶结点对应的单词编码.

3 两种不同宽度的 Huffman 编码树

Huffman 树中结点的最大宽度为 256，当 Huffman 树的深度为 2，且没有空结点时，即所有的单词结点都分布在第二层，每个单词对应的叶结点用 2 个字节表示，则可以表示 $256 \times 256 = 65\ 536$ 个不同的单词. 一个字节的 8 位全部用来表示单词的编码，当 Huffman 树的叶结点在第一、二、三层中都有分布时，单词的编码长度有 1，2，3 个字节三种，虽然构造 Huffman 树的算法可以保证，任何一个单词的编码都不会成为另一个单词编码的前缀，但是有可能出现类似这样的情况，单词 a 编码的后一个字节，与单词 b 编码的前一个（或两个）字节恰好组成单词 c 的编码. 如果文本中单词 a，b 连续出现，单词 c 未出现，在压缩文本中单词 a，b 的编码也连续出现，如果是做解压处理，不会出现任何误差，但如果是在压缩的文本上直接进行检索，就可能将压缩文本中表示单词 a，b 的代码理解成是单词 c 的代码，会影响检索的精确性.

解决这个问题的一个方法是对各个结点进行编码时，只使用一个字节中的 7 位，将最高位留出作为每个单词编码的起始标志位，每个结点的子结点个数最多为 128[2]. 对单词进行编码时，从根结点出发到单词对应叶结点的路径中每个结点均用一个字节表示，其中表示第一层结点的字节中最高位为 1，其余层结点的字节中最高位为 0，即每个单词编码的第一个字节最高位为 1，其余字节最高位为 0. 用这种方法得到的压缩后的文本编码，单词与单词之间有明确的分界，而且可以从压缩文本的任何部位开始进行检索. 算法的不足是压缩文本的每个字节都需要 1 位作为标识，生成的 Huffman 树的深度较大，压缩率不如前面一种算法. 为方便描述，将上述两种算法称为 Huffman256、Huffman128.

我们对压缩算法在文档检索中的应用进行了一些实验，实验采用的数据从以下网址获得，ftp：//ils. unc. edu/pub/…. 实验数据主要为 cf74，cf75，cf76，cf77，cf78，cf79 六个文档文件，包含一些文档的基本信息及摘要. 实验数据大小如表 1 所示.

表 1 测试文件大小

Table 1 Size of test documents

文件名	文件大小/kB	包含的文档/个
Cf74	608	168
Cf75	770	190
Cf76	1090	230
Cf77	848	200
Cf78	689	200
Cf79	875	265

对上述文档文件进行压缩，分别使用了基于字符的 Huffman 编码，基于单词的 Huffman128，Huffman256 编码方法，结果如表 2 所示.

表 2 基于字符与基于单词的 Huffman 压缩算法结果比较

Table 2 Comparison between the compress algorithm based on character and word

压缩算法	平均压缩率 /%（压缩文件大小/源文件大小）
字符 Huffman	26. 6
Huffman128	35. 15
Huffman256	30. 81

从以上结果可以看出，Huffman256 压缩效果要比 Huffman128 的效果好，主要原因是 Huffman128 中单词编码只用了一个字节中的 7 位，留出一位作为标识位.

4 结束语

基于单词的文本压缩算法适用于信息检索系统，有广阔的应用前景. 下一步的研究工作是将基于单词的文本压缩算法具体应用于文本检索系

统中，以此提高文本检索的效率.

参考文献：

[1] Moffat A. Word – based text compression [J]. Software Practive and Experience, 1989, 19 (2): 185 ~ 198.

[2] Nivio Zwiani, Edleno Silva de Moura, Gonzalo Navarro Ricardo. Compression: A key for next – generation text retrieval systems [J]. IEEE Computer, 2000 (11): 37 ~ 44.

[3] Edleno Silva de Moura, Nivio Ziviani, Ricado Baeza – Yates. Fast and flexible word searching on compressed text [J]. ACM Transactions Information Systems, 2000, 18 (2): 113 ~ 139.

Word – based Huffman compression algorithm

CHEN Ji-li[1], YAN Xiao-wei[2], YANG Xiang[1]

(1. *Department of Electronics and Computer*, *Guilin Institute of Technology*, *Guilin* 541004, *China*;
2. *Department of Computer Science*, *Guangxi Normal University*, *Guilin* 541004, *China*)

Abstract: Word – based compression algorithm is a new type of text compression algorithm. The traditional text compression algorithm is based on characters. First, a word – based Huffman compression algorithm is introduced. The construction algorithm of multi – fork tree is presented and demonstrated by an example. Two kinds of Huffman coding trees with different breadths are compared. Algorithm with 256 breadth uses whole byte for coding, while algorithm with 128 breadth uses only 7 bits for coding. The conclusion that the compression ration of the first algorithm is high is proved by experiments.

Key words: word – based compression; Huffman coding; Huffman tree

河北丰宁发现一批距今 1.3 亿年的白垩纪动物化石

河北省丰宁满族自治县发现一批距今 1.3 亿年白垩纪时代的动物化石，引起文物考古专家的高度关注。

这批动物化石是在丰宁县外沟门乡骡子沟村东土窖后山发现的。经文物考古专家初步鉴定，这是一处大面积古动物化石群，大约形成于 1.3 亿年前的白垩纪时代。这批古化石有 10 多个品种，以鱼类为主，多为狼鳍鱼，此外还有蝴蝶、蚌、鸟类等。这些古化石虽然历史久远，但图案十分清晰、精美。

文物考古专家认为，在首都北京周边发现大规模古动物化石群并不多见，这对研究北京地区的地质构成及自然气候变化，具有很高的考古研究价值。

目前，这批古动物化石已被妥善保护。