文章编号：2096-1472(2017)-06-05-03

一种用于藏英文混合文本压缩的改进LZW算法

李加才让,安见才让
(青海民族大学计算机学院,青海 西宁 810007)

摘　要：随着藏文信息处理技术的发展，藏文压缩也成了必不可少的一门研究内容。但是目前的研究成果只有一个，然而现实环境中需要一个适应于不同场合的藏文文本压缩技术。该文根据藏文文本的特点，提出两种改进的LZW数据压缩算法对藏英文混合文本进行数据压缩并无损解压。通过实验结果表明，该算法是一个适应于不同场合的文本压缩技术。

关键词：LZW算法；藏文；英文；文本压缩
中图分类号：TP391　　**文献标识码**：A

An Improved LZW Algorithm for English-Tibetan Bilingual Text Compression

LIJIA Cairang,ANJIAN Cairang
(*Computer Science Qinghai University for Nationalities*,*Xining* 810007,*China*)

Abstract: With the development of Tibetan information processing technology,Tibetan compression has become an essential research content.However,there is only one research result at present.But in reality,the Tibetan text compression technology is needed for different occasions.Based on the characteristics of Tibetan text,two improved LZW data compression algorithms are proposed for both compression and lossless decompression of Tibetan-English bilingual text. Experimental results show that the algorithm is an effective text compression technique adapted to different occasions.

Keywords:LZW algorithm;Tibetan;English;text compression

1 引言(Introduction)

为了应对信息爆炸带来的数据处理难题，迫切需要一些数据压缩技术。虽然国内外对于各种数据的压缩研究已经如火如荼地进行着，但是目前关于藏文文本压缩的已发表的研究成果只有一个[1-3]。学者边巴旺堆等人在《基于LZ77算法的藏文文本压缩算法设计与实现》[4]中以音节分隔符的消除和修改藏文编码算法并结合LZ77压缩算法提出了一个藏文文本压缩算法。尽管上述论文中的藏文文本压缩的压缩效果不错，但本文所研究的藏英文混合文本压缩算法是根据LZ77算法的升级版LZW算法结合藏文自身的特点提出的。为了适应于不同场合的文本压缩，本文算法又构造了三种不同的压缩字典构造方法。

2 经典LZW压缩算法(Classic LZW compression algorithm)

2.1 算法相关概念

(1)CharStream:字符数据流。数据文件中的字符流，一个待编码的字符序列。

(2)Character:字符。一种基础数据元素，Charstream中基本数据元素。

(3)Prefix:前缀。一个Character的最直接的前一个字符。一个前缀字符长度可以为0。

(4)Suffix:后缀。它是一个Character。

(5)String:字符串。一个字符串可以由(A,B)来组成，A是前缀，B是后缀。当A长度为0的时候，代表字符数据流中的第一个字符。

(6)Codestream:编码数据流。一个由Code组成的序列，即经过编码算法处理后的输出数据流。

(7)Code:码。在Codestream中的基本数据元素，它代表Dictionary中的一个元素。

(8)Dictionary:字典。一个由String和Code构成的表，每个String都被分配了一个Code。

基金项目：青海省科技厅（2016-ZJ-Y04）项目资助。

2.2 经典LZW压缩算法

在压缩开始时，Dictionary中仅包含基本的Character及其Code。压缩开始后，读入CharStream，取一个Prefix与当前Character组成String(Prefix,Suffix)按顺序与Dictionary中已有String作循环匹配，直到不能进一步匹配为止。将匹配成功部分的Prefix对应Code输出，当前Suffix作为下一次的Prefix，并将String作为新元素加入Dictionary，给以相应Code。如此继续，直到最后一个Character时，给其编码并返回总输出Codestream。LZW压缩算法流程如图1所示。

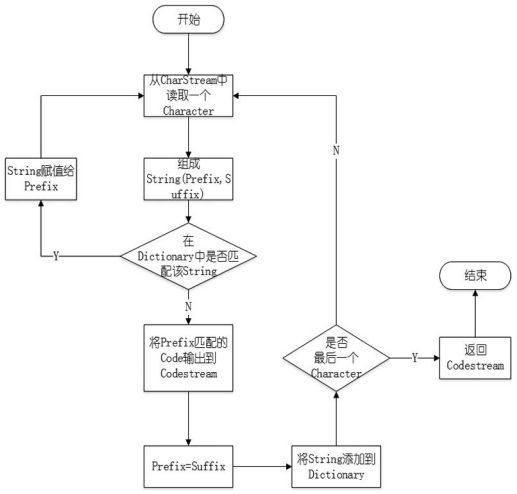

图1 LZW压缩算法流程图

Fig.1 LZW compression algorithm flow

在解压开始时，Dictionary与压缩是一样。解压开始后，读入Codestream，将Code匹配的Character输出到CharStream，直到Prefix不为空。取一个Prefix与当前Character组成String(Prefix,Character的第一个字符)。并将String作为新元素加入Dictionary，给以相应Character。如此继续，直到最后一个Code时，给其匹配Character并返回总输出Codestream。LZW解压算法流程如图2所示。

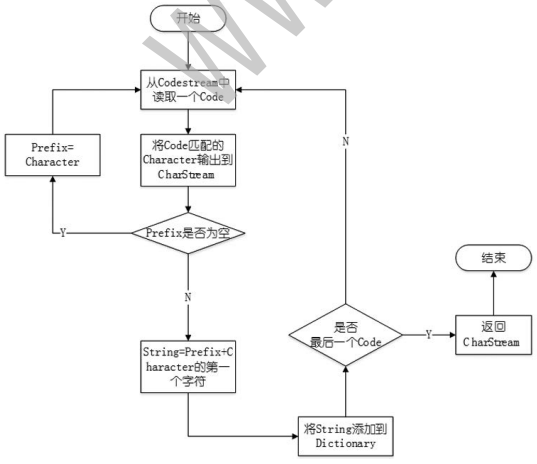

图2 LZW解压算法流程图

Fig.2 LZW decompression algorithm flow

3 藏文拼写语法下的多种构造类别(A variety of structural categories under the Tibetan spelling grammar)

藏文是一种拼音文字，由辅音字母、元音符号、藏文数字和一些标点符号构成。其中有30个辅音字母，4个元音符号。藏文书写习惯为从左到右，从上到下，以一个辅音字母为核心，其余字母均以此为基础前后附加和上下叠加，组成一个完整的字表结构。藏文独特的拼写结构产生了多种构造类别，本文用到的有以下几类。

词：藏文词可分为实词和虚词，实词是从词汇的角度确定的基本单位。虚词指语法功能词[5]。本文所用的分词规范为《青海民族大学藏语语料库基本加工规范》。

字丁：藏文拼写时把一个横向单位称为字丁，一个字丁可以是纵向叠加的组合体，也可以是独立的辅音字符。本文统计的藏文字丁共有550个。

字符：字符是计算机处理藏文的最小处理单位。便于计算机处理，藏文字符在被收入Unicode编码中。在Unicode编码已收录了211个藏文字符，包括辅音字符、元音符号、藏文组合用字符、藏文数字符号、标点符号和一些其他符号[6]。

如字符串ཆོད་མ所示：ཆོད་མ是构成该字符串的词；ཆོ、ད、མ是构成该字符串的字丁；ཅ、ྷ、ོ、ད、་、མ是构成该字符串的字符。

4 LZWTB藏文文本压缩(LZWTB Tibetan text compression)

本文针对藏文独特的拼写结构字符、字丁、词的不同类别。构建这三种类别的字典，作为LZW算法的基本字典，提出一个基本的LZW压缩算法和两个改进的LZW压缩算法。即以字符为LZW字典的LZWTB1压缩算法，以字丁为LZW字典的LZWTB2压缩算法和以词为LZW字典的LZWTB3压缩算法。通过对三种算法的比较，查看各自的压缩效果和压缩效率。

4.1 LZWTB1压缩算法

LZWTB1压缩算法建立Dictionary字典时，首先录入ASCII字符编码，其次录入Unicode的0F00到0FDA的编码范围内的藏文字符编码。并且在CharStream字符数据流读入时以字符为单位读取文本，是一种简单的基本的LZW压缩算法。这个方法的压缩结果令人满意，且无多余的操作，动态建立字典不用保存多余的数据，不耗多余的时间。适合文本的动态压缩，性价比优良。

伪代码实现过程：

```
j=1;                    //编码从1开始
```

for(i=NUT;i<=DEL;i++)　//ASCII字符编码录入Dictionary

{Dictionary.Add(i,j);j++;}

for(i=ཀ;i<=ཪ;i++)　//Unicode藏文字符编码录入Dictionary

{Dictionary.Add(i,j);j++;}

4.2 LZWTB2压缩算法

LZWTB2压缩算法是对LZWTB1压缩算法的一个扩充，建立LZW压缩算法的Dictionary字典时，在字符字典（ASCII和Unicode藏文编码）的基础上扩充字丁字典。在CharStream字符数据流读入时仍以字符为单位读取文本。此方法需要预先统计藏文的字丁，是一种具有预处理部分的文本压缩方法，这种方法的压缩结果同基本LZW压缩算法相差无几，虽然能动态建立字典不用保存多余的数据，但需要预处理统计藏文字丁，性价比不是很好。

伪代码实现过程：

j=1;　//编码从1开始

for(i=NUT;i<=DEL;i++)　//ASCII字符编码录入Dictionary

{Dictionary.Add(i,j);j++;}

for(i=ཀ;i<=ཪ;i++)　//Unicode藏文字符编码录入Dictionary

{Dictionary.Add(i,j);j++;}

for(i=0;i<ZiDing.Length;i++)//预先统计过的字丁数组ZiDing录入Dictionary

{Dictionary.Add(ZiDing[i],j);j++;}

4.3 LZWTB3压缩算法

LZWTB3压缩算法是不同于上述两种算法，该算法在建立LZW压缩算法的Dictionary字典时先录入ASCII的编码，在此基础上对待压缩文本分词，以词为单位录入到Dictionary字典。并保存分词结果在解码时需要再次用到。在CharStream字符数据流读入时以词为单位读取文本。此方法压缩压缩效果最好。但是需要预先分词，并且保存分词结果，是一种具有预处理部分且需要保存额外数据的文本压缩方法，这种方法相比上述两种压缩效果虽然最好，但性价比不是很好。只能对已分词文件压缩解压，不能动态建立字典对所有文本进行压缩。

伪代码实现过程：

j=1;　//编码从1开始

for(i=NUT;i<=DEL;i++)//ASCII字符编码录入Dictionary

{Dictionary.Add(i,j);j++;}

for(i=0;i<Ci.Length;i++)　//预先预先分词结果的数组Ci录入Dictionary

{Dictionary.Add(Ci[i],j);j++;}

5 测试(Test)

5.1 实验环境

系统平台：Microsoft Windows 7

开发工具：Microsoft Visual Studio

开发语言：C#

数据库：Microsoft SQL Server 2014

5.2 实验结果

以下是对三种压缩方法和不同文本大小的压缩效果对比表，其中压缩率=(压缩后文本大小/原文本大小)*100%，压缩率越小压缩效果越好。

表1 LZWTB1压缩算法结果

Tab.1 LZWTB1 compression algorithm results

文本名	文本大小	压缩后大小	压缩率	耗时
文本1	334k	149k	44.61%	0.78s
文本2	658k	278k	42.24%	0.99s
文本3	2943k	1186k	40.29%	1.24s

表2 LZWTB2压缩算法结果

Tab.2 LZWTB2 compression algorithm results

文本名	文本大小	压缩后大小	压缩率	耗时
文本1	334k	151k	45.20%	1.09s
文本2	658k	277k	42.09%	1.40s
文本3	2943k	1186k	40.26%	1.66s

表3 LZWTB3压缩算法结果

Tab.3 LZWTB3 compression algorithm results

文本名	文本大小	压缩后大小	压缩率	耗时
文本1	334k	81k	23.25%	3.27s
文本2	658k	162k	24.62%	5.58s
文本3	2943k	739k	25.11%	50.19s

6 结论(Conclusion)

根据算法的构造复杂度与实验结果的分析比较，本文提出的三种藏英文混合文本压缩方法中的LZWTB1压缩算法是三种算法中性价比最好的一种压缩算法，上述实验数据表示它有不错的压缩效果，并且能在压缩过程中动态地建立字典，无需额外的数据保存。该压缩算法适用于急需的、远程的文本压缩。LZWTB2压缩算法因为预处理统计藏文字丁而

（下转第4页）