統計型圧縮とrepetition finderを併用する高速データ圧縮方式

2G－3

佐藤宣子　矢作裕紀　吉田茂

（株）富士通研究所　ペリフェラルシステム研究所　入出力研究部

1. はじめに

データの統計的性質を事前に仮定しない統計型ロスレス圧縮において、高圧縮を得るために高次の文脈を用いたとき、ビット単位の動的可変長符号化では、的中率の高い文字の符号化効率が問題になる。

本稿では、動的可変長符号化にrepetition finderの原理を取り入れて、符号化効率を改善し、ビット単位符号で高速処理と高圧縮率を両立できる方式を得たので報告する。

2. 高次文脈での動的可変長符号化の圧縮特性

文脈の長さと圧縮率の関係を図１に示す（英文テキストのとき）。１文字に１ビット以下の符号を割り当てることができる算術符号[2]と、ビット単位符号のハフマン符号とスプレイ符号[3]を比較する。ここで、スプレイ符号とは、符号木を自己組織化規則で更新する、高速の動的可変長符号化である。

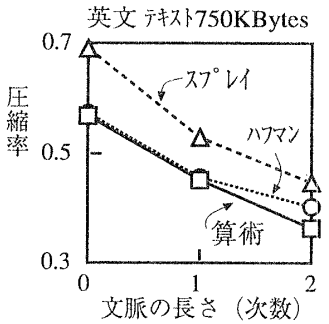

図１　文脈の長さと圧縮率

算術符号は、次数を上げるにつれて高圧縮になる。ハフマン符号は、０次１次では算術符号と同じ圧縮率だが、２次では次数に見合う圧縮率が得られない。高次になると注目文字が0.5を超える高い的中率[4]で出現し、ビット単位の符号では十分な符号化効率が得られないためである。スプレイ符号は、０次１次では他の２符号と圧縮率に差があるが、２次ではハフマン符号と僅差になる。高次では注目文字が高い的中率で確定的に予測されるため、ハフマン符号のように厳密でなくても、同程度の圧縮率が得られる。

3. 新方式の提案

文字間の繋がりから高い的中率が予想されるときに、１文字に１ビット以下の符号を割り当てる方式を検討した。図２に示すように、(a)文字単位の符号化と(b)文字列単位の符号化とを併用する。(b)の符号化では、repetition finder[1]を用いて、高い的中率の文字の繋がりを符号化する。符号化対象の注目文字に対し、直前のn文字列が過去に出現していれば、それに続くn+1文字以降は、過去の文字列と同じ文字の並びが確定的に出るものと仮定する。

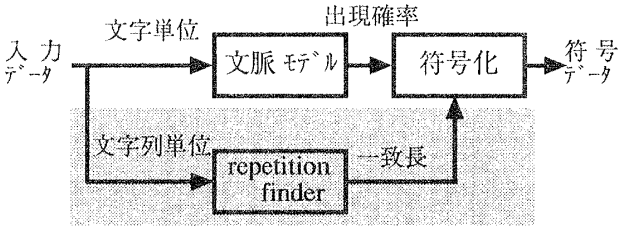

図２　長い文字列のみ一致長符号で符号化

高次相関を取るため、n=3とし、図３のように、４文字以降の一致長を求め、この一致長を符号化する。過去に直前３文字列と同じ文字列がなければ、注目文字１文字を文脈モデルで符号化する。

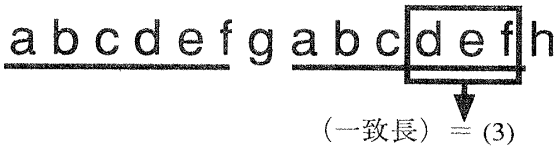

図３　検討方式の符号形態

過去の一致文字列の検出には、高速化に向くハッシュ法[5]を用いる。図４のように、直前３文字列からハッシュコードを生成し、ハッシュテーブルを引いて、過去の出現位置を求める。

"A Fast LossLess Data Compression Method Using Statisitical Context Modeling and Repetition Finder"
Noriko Satoh, Hironori Yahagi, Shigeru Yoshida
I/O systems lab., Peripheral systems Laboratories,
FUJITSU Laboratories Ltd.

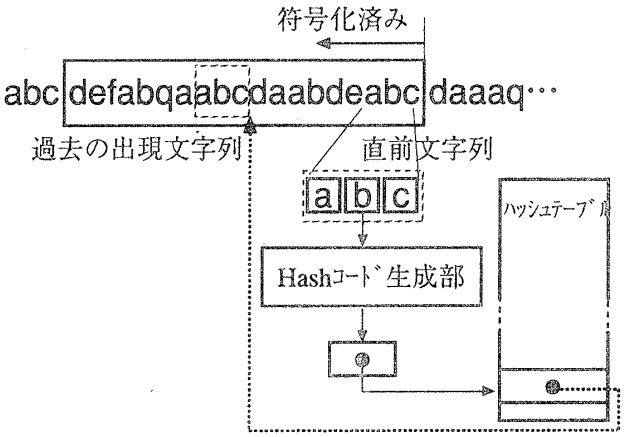

図４　一致検索

4. 方式の評価

標準的な評価データcalgary corpus[2]を用いて、圧縮率（圧縮後のﾌｧｲﾙｻｲｽﾞ／圧縮前のﾌｧｲﾙｻｲｽﾞ）を求め、１次文脈の提案方式の圧縮性能を評価した。

文字間の繋がりが強いデータにおいて、改善効果があり、２次の算術符号と同等の圧縮率が得られた（図５）。算術とハフマンとの圧縮率の差が特に大きかった２値画像(pic)、プログラムデータ(progl, progp)で高い圧縮率が得られた。また、提案方式は、ハフマン符号でもスプレイ符号でも圧縮率は同様になった。

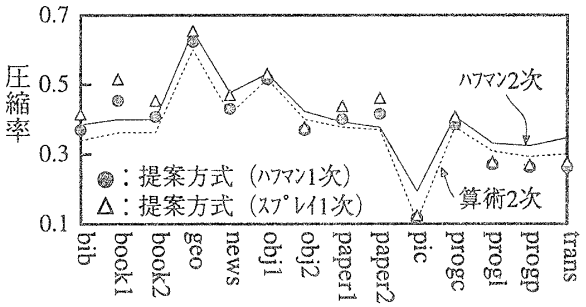

図５　提案方式の圧縮率

一方、英文テキスト（book1-2, paper1-2）では、過去の文字列と同じ文字の並びが必ずしも確定的に出ないため、符号化効率改善は小さかったものの、通常の１次より良い圧縮率が得られた。

実用性をみるため、フリーソフトLHAと提案方式（スプレイ１次にrepetition finderを併用した方式）と圧縮性能を比較した。提案方式は、単純な１次文脈、ハッシュ法によるrepetition finderという簡素な構成を取り、高速のスプレイ符号を用いるため、圧縮、復元ともに高速である。

表１　圧縮性能比較

評価データ（本原稿A4-2枚）ﾊﾞｲﾄｻｲｽﾞ	圧縮率		圧縮/復元速度(KB/s)	
	提案方式	LHA v1.0	提案方式	LHA v1.0
ｸﾗﾘｽ 文書 262K	0.33	0.31	718 725	136 661
PDF 文書 46K	0.33	0.30	420 430	95 243

S-4/20H(2CPU 151MHz), Solaris 2.5.1 で測定

5. まとめ

統計型方式において、ビット単位符号を用いたとき、高次文脈でこれに見合う圧縮率が得られない弱点を、repetition finderによる文字列単位の符号化によって改善した。このrepetition finderに高速のスプレイ符号を併用し、高圧縮率と高速処理を両立させた。本方式は、辞書型のように文字列単位の符号化を行うが、統計型における符号化効率改善という視点により、文字間の相関を常に１次以上利用できる構成になっており、高圧縮率が得られる。

本方式は、簡素かつ高性能であり、統計型の実用的方式として期待できる。

［参考文献］

[1] Hidetoshi Yokoo, "An Improvement of Dynamic Huffman Coding with a simple Repetition Finder", IEEE trans. on comm. vol. 39, No.1, Jun, 1991

[2] T. C. Bell, J. G. Cleary, I H. Witten, "Text Compression", PRENTICE HALL 1990

[3] D. W. Jones, "Application of Splay trees to data compression", Communications of ACM, Vol.31 No.8 p.996-1007, Aug. 1988

[4] Damras Wongsawang, Masayuki Okamoto, "Dependency Data Compression with Self-Organizing Lists", Trans. of Info. Processing Society of Japan , Jun. 1994

[5] R. N. Williams, "An Extremely Fast Ziv-Lempel Data Compression Algorithm", Data Compression Conf. p.362-371, 1991