ブロックソーティングによる無歪みデータ圧縮とその周辺

The Block-Sorting Lossless Data Compression Algorithm and Related Topics

山本博資
Hirosuke Yamamoto
東京大学大学院工学系研究科
School of Engineering, University of Tokyo

1 はじめに

ユニバーサルな無歪みデータ圧縮符号としては，従来 Lempel-Ziv 符号を始めとする辞書法が数多く提案され，compress, gzip, LHA などの圧縮ツールとして広く使用されてきている．しかし，1990 年代中ごろから，新たにソートを用いてデータ系列の文脈を並び換え，類似文脈内における相対的な位置情報を用いてデータを符号化することにより，さらに性能のよい圧縮を実現する符号が提案され，注目を集めている．ソートを用いる新しい符号化法には，ブロックソート法 [1], 文脈ソート法 [2], ACB 法 [3][4] などが存在するが，そのうち本稿ではブロックソート法を取り上げ解説を行う．基本的な符号化・復号化アルゴリズムを説明すると共に，ブロックソート法のバリエーションや，情報理論的な性能評価などを紹介する．なお，ブロックソート法は圧縮ツール bzip としてインプリメントされている．

2 符号化アルゴリズム

$\boldsymbol{x} = abracadabra$ を符号化する場合を例にとって，符号化アルゴリズムを説明する．

E1: $\boldsymbol{x}$ を n 回巡回シフトして，n 個の系列を求める．

E2: n 個の系列をソートして並び換えたソートテーブルを求める．ソートテーブルにおける $\boldsymbol{x}$ の行番号を m とし，ソートテーブルの最終列の系列を $\boldsymbol{y}$ とする．

$\boldsymbol{x} = abracadabra$ に対するソートテーブルを表 1 に示す．$\boldsymbol{x}$ はソートテーブル内で 3 行目となるので $m = 3$ となり，最終列より $\boldsymbol{y} = rdarcaaaabb$ となる．

E3: $\boldsymbol{y}$ を Recency Rank(RR) 符号化を用いて，正整数値系列 $\boldsymbol{z}$ に符号化する．つまり，$\boldsymbol{y}$ の各シンボルが何種類前のシンボルと一致しているかを示す整数値で符号化する．なお，RR 符号化は Move-to-front 符号化と等価である．上の例の場合，アルファベット $\mathcal{A}$ が $\mathcal{A} = \{a, b, c, d, r\}$ なので，初期値系列を $abcdr$ とすれば，$\boldsymbol{y}$ は次のように整数値系列 $\boldsymbol{z}$ に符号化される．

$$\begin{array}{ll} abcdr & rdarcaaaabb \ (= \boldsymbol{y}) \\ & 12534311151 \ (= \boldsymbol{z}) \end{array}$$

E4: m と $\boldsymbol{z}$ を 2 値符号化する．$\boldsymbol{z}$ の符号化はハフマン符号や算術符号などのようなエントロピー符号化が用いられたり，正整数のユニバーサルな 2 値表現（2 値符号）などが用いられる．

表 1: ソートテーブルおよびその第 1 列 $\boldsymbol{u}$ と最終列 $\boldsymbol{y}$

行番号	ソートテーブル	$\boldsymbol{u} \cdots\cdots\cdots \boldsymbol{y}$
1	$aabracadabr$	$a_1 \cdots\cdots\cdots r_1$
2	$abraabracad$	$a_2 \cdots\cdots\cdots d$
3	$\underline{abracadabra}$	$\underline{a_3 \cdots\cdots\cdots a_1}$
4	$acadabraabr$	$a_4 \cdots\cdots\cdots r_2$
5	$adabraabrac$	$a_5 \cdots\cdots\cdots c$
6	$braabracada$	$b_1 \cdots\cdots\cdots a_2$
7	$bracadabraa$	$b_2 \cdots\cdots\cdots a_3$
8	$cadabraabra$	$c \cdots\cdots\cdots a_4$
9	$dabraabraca$	$d \cdots\cdots\cdots a_5$
10	$raabracadab$	$r_1 \cdots\cdots\cdots b_1$
11	$racadabraab$	$r_2 \cdots\cdots\cdots b_2$

3 復号化アルゴリズム

復号は，符号化の手順を逆に行えばよい．

D1: 2 値符号語系列から，m と $\boldsymbol{z}$ を求める．

D2: RR 符号の復号により，$\boldsymbol{z}$ から $\boldsymbol{y}$ を求める．

D3: $\boldsymbol{y}$ をソートして，ソートテーブルの最初の列 $\boldsymbol{u}$ を求める．

D4: $\boldsymbol{u}$ と $\boldsymbol{y}$ の対応から，m 行目の各シンボルを逐次的に求める．

ソートテーブルの各行は巡回シフトして作られているため，第 1 列 $\boldsymbol{u}$ と最終列 $\boldsymbol{y}$ の各シンボルは一対一に対応しており，ソートテーブルがソートして作られていることから，$\boldsymbol{y}$ をソートすることにより $\boldsymbol{u}$ が求まる．

c のように一度しか現れないシンボルは，$\boldsymbol{y}$ と $\boldsymbol{u}$ における対応がすぐ分かるが，a のように同一シンボルが幾つも現れている場合は，$\boldsymbol{y}$ のどの a と $\boldsymbol{u}$ どの a が対応しているかが分からないように思える．しかし，表 2 に示すように同一種類のシンボルは $\boldsymbol{u}$ におけるの出現順と $\boldsymbol{y}$ における出現順が完全に一致している．これは，$ax_2x_3 \cdots x_n$ と，それを巡回シフトした $x_2x_3 \cdots x_na$ のソート順がどちらも $x_2x_3 \cdots x_n$ のソート順で決まるためである．

以上より，表 1 の右側に示した $\boldsymbol{u}$ と $\boldsymbol{y}$ の情報を得ることができる．他方，$m = 3$ の情報から，$\boldsymbol{x} = a_3 \cdots\cdots a_1$ であることが分かる．さらに，ソートテーブルの各行が巡回シフトして作られているので，表 1 の右側より，$\boldsymbol{x}$ には r_1a_1, da_2, a_1a_3, r_2a_4, ca_5, a_2b_1, a_3b_2, a_4c, a_5d, b_1r_1, b_2r_2 のシンボル対が含まれていることが分かり，もとの $\boldsymbol{x} = a_3b_2r_2a_4ca_5da_2b_1r_1a_1 = abracadabra$ が復元できる．

ブロックソート法で効率よく圧縮できる理由は次のように

表 2: $\boldsymbol{y}$ と $\boldsymbol{u}$ における a の対応

行番号	a で始まる行	a で終わる行
1	$a_1\, abrac\cdots$	
2	$a_2\, braab\cdots$	
3	$a_3\, braca\cdots$	$abraca\cdots a_1$
4	$a_4\, cadab\cdots$	
5	$a_5\, dabra\cdots$	
6		$braab\cdots a_2$
7		$braca\cdots a_3$
8		$cadab\cdots a_4$
9		$dabra\cdots a_5$

説明できる．例えば，the, they, these などを多く含む英語の文章データから作ったソートテーブルでは，“he$\cdots$”で始まる行の多くは最終シンボルが “t” となる．同様に，語頭部分（つまり文脈）が同じ文字列で始まる行の最終シンボルは同じシンボルとなりやすい．そのため，最終列 $\boldsymbol{y}$ を RR 符号化した $\boldsymbol{z}$ では 1 が最も多く現れ，z の値が大きくなるにつれて出現しにくくなり，整数値 z に大きな頻度の偏りが生じるからである．

4 バリエーションおよび理論解析

前節で説明した符号化では，行番号 m により $\boldsymbol{x}$ がソートテーブルのどの行に存在するかを指定した．m を用いる代わりに，アルファベット $\mathcal{A}$ に含まれない先頭記号\$を $\boldsymbol{x}$ の前に付加し，\$は任意の文字より辞書順に早いと定義すると，$\$\boldsymbol{x}$ は常にソートテーブルの第 1 行目に現れるため，m の情報が必要でなくなる [2].

ブロックソート法では，全ブロック長を用いてソートするため，符号化に時間がかかる．ソートする範囲を最初の k 文字に制限して，符号化速度を高速化する手法が提案されている [5]．また，Suffix array を利用する方法などもある [6].

RR 符号化された後の $\boldsymbol{z}$ に含まれる整数値は，1 が連続しやすく，整数値 z の生起確率は実験的に z の値の 2 乗に反比例することが知られている [7, etc.]．この性質を利用して，Run-Length 符号化法やその分布に適合した正整数の 2 値符号を用いて符号化する方法が提案されている．また，RR 符号化の代わりに，Inversion Frequencies という符号化法を用いる手法もある [8].

その他，ブロックソート法と他の符号（文脈ソート法 [2], PPM(Prediction by Partial Matching) 法 [9]，Hershkovits-Ziv 符号 [10]，ACB 法 [3]，etc.）などの関係が考察されている．

ブロックソート法ではソートにより $\boldsymbol{x}$ の確率構造が壊されてしまうため，情報理論的な性能評価が進んでいなかった．しかし最近，ソートテーブルでは同じ文脈系列が集まる特性を利用して，シンボル拡大を行えば，定常情報源に対して，期待値および almost sure の意味で漸近的にエントロピーレートまで圧縮できることが理論的に証明されている [11][12]．しかし，シンボル拡大を行わない場合，符号化アルゴリズムの E3 で RR 符号化が行われると，マルコフ情報源のような単純な情報源の場合でも特殊な場合を除いてエントロピーレートまでは圧縮できないことも判明している．

参考文献

[1] M. Burrows and D. J. Wheeler, “A Block-Sorting Lossless Data Compression Algorithm,” *SRC Research Report 124*, Digital Systems Research Center, Palo Alto, CA., May 1994.

[2] H. Yokoo and M. Takahashi, “Data Compression by Context Sorting,” *IEICE Transactions on Fundamentals*, Vol. E78-A, No. 5, pp. 681–686, May 1995.

[3] G. Buyanovski (translated by Z. Agazarian) “Associative Coding,” *Monitor*, Moscow, pp. 10–19, Aug. 1994.

[4] D. Salomon, “Data compression,” Springer-Verlag, New York, 1997

[5] D. M. Schindler, “A Fast Block-Sorting Algorithm for Lossless Data Compression,” *Proc. Data Compression Conference*, Snowbird, Utah, p.469, March 1997.

[6] K. Sadakane, “A Fast Algorithm for Making Suffix Arrays and for Burrows-Wheeler Transformation,” *Proc. 1998 Data Compression Conference (DCC'98)*, Snowbird, Utah, pp. 129–138, 1998.

[7] P. Fenwick, “Block Sorting Text Compression — Final Report,” *Technical Report 130*, Dept. of Computer Science, The University of Auchland, New Zealand, April 1996.

[8] Z. A. Arnavut and S. S. Magliveras, “Block Sorting and Compression,” *Proc. 1997 Data Compression Conference (DCC'97)*, Snowbird, Utah, pp.181–190, March 1997.

[9] T. C. Bell, J. G. Cleary, and I. H. Witten, “Text Compression, Prentice-Hall, Inc., New Jersey, 1990

[10] Y. Hershkovits and Ja. Ziv, “On Sliding-Window Universal Data Compression with Limited-Memory,” *IEEE Transactions on Information Theory*, Vol. 44, No. 1, pp. 66–78, Jan. 1998.

[11] M. Arimura and H. Yamamoto “Asymptotic Optimality of the Block Sorting Data Compression Algorithm,” *IEICE Transactions on Fundamentals*, Vol. E81-A, No. 10, pp. 2117-2122, Oct. 1998.

[12] M. Arimura and H. Yamamoto “Almost Sure Convergence Coding Theorem for Block Sorting Data Compression Algorithm,” *Proc. 1998 International Simposium on Information Theory and Its Applications (ISITA98)*, Mexico City, Mexico, Oct. 1998.