2001 年情報論的学習理論ワークショップ
2001 Workshop on Information-Based Induction Sciences (IBIS2001)
Tokyo, Japan, July 30 - August 1, 2001.

ユニバーサルデータ圧縮アルゴリズムの変遷 —基礎から最新手法まで—

Variations on Universal Data Compression Algorithms - Basic Methods to Recent Methods -

山本 博資*
Hirosuke Yamamoto

Abstract: Universal data compression codes are surveyed. Based on used techniques, universal codes are classified into stochastic method, dictionary-based method, sorting method, or grammar-based method. For each method, basic ideas are introduced, and detailed algorithms are described for some universal codes, e.g. CTW, PPM, Bock sorting, SEQUITUR, MPM codes.

1 まえがき

IT 技術の進歩に伴い，通信回線や記録媒体の大容量化が進んでいるが，それにも増して，社会の情報化/IT 化が急速に進み，伝送/記録される情報量が増加し続けている．そのため，情報をより効率よく伝送/記録するためのデータ圧縮技術は，今後も情報化社会を支える必要不可欠な技術であり，より効率のよい符号化技術が求め続けられるであろう．

1948 年に，Shannon[55] が定常無記憶情報源に対してエントロピーまで圧縮可能であることを証明し，その後，MacMillan[37] が，定常エルゴード情報源に対してエントロピーレートまで圧縮可能なことを示したことにより，データ圧縮符号の究極の目的がエントロピーレートを達成することであることが明らかとなった．その後，FV(固定長-可変長) 符号で最適な Huffman 符号 [27] や，VF(可変長-固定長) 符号で最適な Tunstall 符号 [60] などが考案されたが，実用的にエントロピーレートを達成するには，メモリ量/計算量的に困難であった．

1976 年に，Rissanen[45] と Pasco[44] が，Shannon-Fano-Elias 符号 [28] を源流とした算術符号 (Arithmetic Code) を提案するに至り，情報源の確率構造が既知の場合には，実用的なメモリ量/計算量で圧縮限界のエントロピーレートを容易に達成できるようになった．しかし，文書, プログラム, 画像, 科学データなど，実際に圧縮したい情報の確率構造が既知であることはほとんどなく，確率構造の未知な情報を効率よく圧縮できることがデータ圧縮符号に要求されている．本稿では，そのような要求を満たすユニバーサル符号の基本概念と手法を紹介する．なお，紙面の都合上，無ひずみ圧縮を取り扱うが，ひずみを許すユニバーサル符号の符号化定理に関しては，文献 [29] で紹介されている．

*東京大学 大学院情報理工学系研究科 数理情報学専攻，〒 113-0033 東京都文京区本郷 7-3-1，E-mail: yamamoto@hy.t.u-tokyo.ac.jp
Dept.of Mathematical Informatics, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

2 ユニバーサル符号の歴史

主なデータ圧縮符号を，表 1 に示す．ユニバーサル符号の研究は，1966 年の Fitingof[24], Lynch[36], Davisson[16] らの研究に始まるが，理論的な研究から実用レベルへの大きな進展は，1976-78 年と 1983-86 年に起こっている．前者の期間に，算術符号 [44][45] と Lempel-Ziv 符号 (LZ77 符号 [74],LZ78 符号 [75]) が考案され，後者の年には，MDL 符号化 [47][48] の理論的枠組みが明らかにされる一方で，算術符号を用いた PPM 符号 [13]，LZ78 符号を改良した Welch 符号 [65]，LZ77 符号を改良した Bell 符号 (LZSS 符号)[6] などが，実用的な計算機性能で高い圧縮率を達成できることを実証し，ユニバーサル符号は実用段階に入った．

その後，コンピュータの高性能化に伴い，より多くのメモリ量や計算量を伴う符号化法も実用可能となってきているが，1990 年代中頃に提案された CTW 符号 [66] やブロックソート符号 (BS 符号)[9] などはその例である．また，1990 年代半ば以後，圧縮したいデータ系列を生成する文法に符号化する SEQUITUR 符号 [41] や MPM 符号 [31] などが提案され，注目を集めている．

表 1: 主なデータ圧縮符号　({　} 内は関連してよく知られているもの)

1948	Shannon 符号 [55]
1952	Huffman 符号 [27]
1959	Run-Length 符号化 [12]
1962	Tunstall 符号 [60]
1966	ヒストグラム符号化 [24]
1968 {1975}	正整数のユニバーサル符号 [34] {[18]}
1973	数え上げ符号 [15]
1973 {1978}	動的 Huffman 符号 [21] {[25]}
1976	算術符号 [44][45]
1977	LZ77 符号 [74]
1978	LZ78 符号 [75]
1983	MDL 符号化 [47][48]
1984	Welch 符号 (LZW 符号)[65]
1984	PPM 符号 [13]
1986	Bell 符号 (LZSS 符号)[6]
1986 {1987}	Move-to-Front 符号 (MTF 符号)[8] {Recency-Rank 符号 [19]}
1989	Fiala-Greene 符号 (LZFG 符号)[23]
1992 {1996}	動的 Tunstall 符号 [57]{[20]}
1994	ブロックソート符号 (BS 符号)[9]
1994	ACB 符号 [10]
1994 {1997}	SEQUITUR 符号 [40] {[41]}
1995	CTW 符号 [66]
1996	文脈ソート符号 (CS 符号)[73]
2000	MPM 符号 [31]

3　ユニバーサル符号の分類

データ圧縮符号は，大きく表 2 のように分類できる．

エントロピー符号化に分類される符号は，符号の構成または符号化に確率を陽に用いるものであり，仮定された確率分布を持つデータ系列に対しては性能よく圧縮できるが，実際の分布 P が仮定された分布 Q と異なるときには，ダイバージェンス $D(P\|Q)$ 以上のロスを生じる．実用的には，事前に確率分布を知ることは困難なため，エントロピー符号化を単独で用いることは少なく，ユニバーサル符号の一部として利用される場合が多い．

ユニバーサル符号は，あるクラスに属する任意の情報源出力を効率良く圧縮できる（あるいは，漸近的にエントロピーレートまで圧縮できる）符号である．ユニバーサル符号の構成方法は，統計的な手法を用いて情報源の確率を推定し，それを用いてエントロピー符号化を行う方式 (II-A) と，符号化/復号化に確率を陽に含まないアルゴリズムを用いる方式 (II-B) に分類できる．なお，動的 Huffman 符号や動的 Tunstall 符号は，任意の定常無記憶情報源に対して適応的に最適な符号を構成できるので，ユニバーサル符号と考えることもできるが，対象情報源のクラスが狭いため，一般にはユニバーサル符号とは呼ばれていない[2]．

表 2: 主なデータ圧縮符号の分類

I. エントロピー符号化

- i. 静的符号（Huffman 符号，Tunstall 符号，算術符号など）
- ii. 動的符号（動的 Huffman 符号，動的 Tunstall 符号，算術符号など）

II. ユニバーサル符号

- II-A　統計的な確率推定と算術符号の組合せ（MDL 符号, CTW 符号, PPM 符号など）
- II-B 符号化アルゴリズムに確率を陽に含まない符号
  - i. 構成要素として使用される符号（MTF 符号，正整数のユニバーサル符号[1]など）
  - ii. 辞書法（LZ77 符号, LZ78 符号, LZW 符号，LZFG 符号など）
  - iii. ソート法（BS 符号，CS 符号，ACB 符号など）
  - iv. 文法法（SEQUITUR 符号，MPM 符号など）

II-A の方式は，符号化を 2 段階で行う two-pass 方式（データ系列を先に全部調べて情報源確率を推定し，その確率に基づいて算術符号化を行う方式）と，1 段階で行う one-pass 方式（データを逐次的に読みながら，確率 $P(x_t|x_1^{t-1})$ の推定と算術符号化を逐次的に繰り返す方式）とに分類できる．

II-B の方は，符号の特徴により，ii. 辞書法，iii. ソート法，iv. 文法法に大きく分類できる．

以下の節で，これらのユニバーサル符号化法の基本的な概念および重要なアイデアを紹介する．

なお，ユニバーサル圧縮符号の性能は，(1) 圧縮率，(2) 符号化/復号化速度，(3) 必要メモリ量，(4) 対象クラスの広さ，などを考慮しなければならない．これらの性能は，互いに trade-off の関係になっていることが多く，どの性能に重点を置くかで，最適な符号が異なってくる．

[2]Huffman 符号や Tunstall 符号などの木符号の研究は，[1] に詳しく紹介されている．

4 確率推定と算術符号を用いるユニバーサル符号

情報源のモデルがパラメータ $\theta = (\theta_1, \theta_2, \cdots, \theta_k)$ を用いて表現され，そのもとでの x_1^n の生起確率が $P_\theta(x_1^n)$ で与えられるものとする．two-pass で符号化するときには，パラメータ θ を用いて算術符号化したときの符号語長 $-\log P_\theta(x_1^n)$ とパラメータ θ を記述するために必要な符号語長 $(k/2)\log n$ の和が，トータルの符号語長となる．そのため，

$$\min_{\theta,k}\{-\log P_\theta(x_1^n) + \frac{1}{2}k\log n\}$$

を最小にするモデル θ を用いて算術符号化するのが最適となる．この基準を MDL(minimum description length) [48] という．

ファイルなどの実際のデータに対して，上記の最適な θ を求めるのは一般に困難であるため，1 シンボルごとに最適なモデルを選択しながら，逐次的に符号化が行える one-pass 方式の符号化が望まれる．以下では，one-pass の主な符号化法を紹介する．

4.1 情報源モデル

情報源モデルのクラスを広くしすぎると，モデルの記述長が長くなり効率が悪くなる．しかし，クラスを狭くすると，実際のデータの確率をうまく近似できなくなり，効率が悪くなる．したがって，どのような情報源モデルを仮定するかが重要となる．

情報源のアルファベットを $\mathcal{X}$ とし，任意の $x_{t+1} \in \mathcal{X}$ に対して，$P(x_{t+1}|x_1^t) = P(x_{t+1}|s(x_1^t))$ を満たすような，有限状態集合 $\mathcal{S}$ の値をとる関数 $s(x_1^t)$ を考える．この関数が，初期状態 $s(\lambda) = s_0$ と

$$s(x_1^{t+1}) = f(s(x_1^t), x_{t+1}) \tag{1}$$

である状態遷移関数 $f : \mathcal{S} \times \mathcal{X} \to \mathcal{S}$ により定まっているものとする．このとき，初期状態 s_0 と関数 $f(s,x)$ と $|\mathcal{S}| \times (|\mathcal{X}| - 1)$ 個の条件付き確率 $P(x|s)$ で定義される情報源を，FSM(finite-state-machine-generated) 情報源という．

$S(x_1^t) = x_t x_{t-1} \cdots x_{t-m+1}$ のとき，m 次のマルコフ情報源となる．m 次のマルコフ情報源は，深さ m の木を用いて表現できるが，深さが一定でない木で表現できる FSM 情報源を，FSMX 情報源という [49]．例えば，2 値の情報源で，$\mathcal{S} = \{1, 01, 000, 001\}$ は，FSMX となる．$s(x_1^t) = x_t x_{t-1} = 01$ の状態で，$x_{t+1} = 0$ のとき $s(x_1^{t+1}) = x_{t+1}x_t x_{t-1} = 001$，$x_{t+1} = 1$ のときに，$s(x_1^{t+1}) = x_{t+1} = 1$ の状態に遷移し，$s(x_1^t)$ と x_{t+1} から次の状態 $s(x_1^{t+1})$ が一意に決まり，式 (1) が成り立つ．

状態を木で表現できる情報源でも，式 (1) を満たさない場合もある．例えば，$\mathcal{S} = \{1, 00, 010, 011\}$ の場合である．$s(x_1^t) = x_t = 1$ の状態で $x_{t+1} = 0$ のとき，これらの情報からは，次の状態 $s(x_1^{t+1})$ は決まらない．しかし，$x_{t-1} = 0$ なら $s(x_1^{t+1}) = x_{t+1}x_t x_{t-1} = 010$，$x_{t-1} = 1$ なら $s(x_1^{t+1}) = x_{t+1}x_t x_{t-1} = 011$ というように，過去の値を知ることができれば，$s(x_1^{t+1})$ を定めることができる．木の最大深さが有限で，木の根から任意の内部節点までの系列 α と，ある $x, a, b \in \mathcal{X}$ に対して，$P(x|\alpha a) \neq P(x|\alpha b)$ を満たすとき，この情報源を minimal な木情報源 (tree source) あるいは有限記憶情報源 (finite memory source) という [64][66].

4.2 Context アルゴリズムと状態選択

木情報源における各状態 (節点, 文脈) 対する頻度を，次の Context アルゴリズムで求めることができる [47].

根だけの木を T_0 とし，根の各 $x(\in \mathcal{X})$ の頻度をゼロに設定して，$t = 0, 1, 2, \cdots$ に対して以下を繰り返す．

Step-A1 x_{t+1} を読み，木 T_t を根から，$x_t x_{t-1} x_{t-2} \cdots$ というようにたどり，各節点の文字 x_{t+1} に対応した頻度を 1 増加させる．

Step-A2 最も深い節点の頻度が 0 のときは，1 に増加させる．1 のときは頻度を 2 に増加させ，そこから新しい節点（葉）をつくり，その節点の全ての頻度をゼロにセットする．

節点 s における $x \in \mathcal{X}$ の頻度を $n_s(x)$ とし，$n_s = \sum_{x\in\mathcal{X}} n_s(x)$ すると，節点 s を用いて文字 x の符号化をするとき，次式で与えられる Laplace 推定確率 $P_L(x|s)$([49] で用いられている) や Krichevsky-Trofimov 推定 (KT 推定)[35][56] 確率 $P_{KT}(x|s)$ を用いて行うことができる．

$$P_L(x|s) = \frac{n_s(x)+1}{n_s + |\mathcal{X}|}, \quad P_{KT}(x|s) = \frac{n_s(x)+1/2}{n_s + |\mathcal{X}|/2}$$

なお，確率 $P(x_1^n) = (1-\theta)^{n_0}\theta^{n-n_0}$ を持つ長さ n のベルヌーイ系列 x_1^n に対する KT 推定確率 $P_{KT}(x_1^n)$ の冗長度は，次式のように，$\theta \in [0,1]$ に依存しない上界を持つが，Laplace 推定確率はこのような性質を持たない [66].

$$\log \frac{P(x_1^n)}{P_{KT}(x_1^n)} \leq \frac{1}{2}\log n + 1$$

x_{t+1} を符号化するときに，木情報源のどの深さの文脈 s を用いて符号化するかは，次のように決めることができる．$x_t x_{t-1} \cdots$ に従って木を根からたどり，各節点 s における Stochastic Complexity(S.C.) $L(s)$ と，その子の節点 s_a の S.C. の総和 $\sum_{a\in\mathcal{X}} L(s_a)$ を比較し，後者

の方が大きくなる最初の s を用いて符号化する．なお，2 値の場合の S.C. は次式で与えられる [5].

$$L(s) = n_s \log n_s - \sum_{x=0,1} n_s(x) \log n_s(x) + \frac{1}{2} \log \frac{n_s \pi}{2} + O(\frac{1}{n_s})$$

文献 [64] には，葉の数が K で確率 $P(x_1^n)$ を有する木情報源出力 x_1^n に対して，もう少し複雑な方法で最適な状態を選び，KT 推定確率 $P_{KT}(x_1^n)$ を用いると，冗長度が確率 1 で

$$\frac{1}{n} \log \frac{P(x_1^n)}{P_{KT}(x_1^n)} \leq \frac{K(|\mathcal{X}|-1)}{2n} \log n + O(\frac{1}{n})$$

の上界を持ち，冗長度の期待値に関しても同様の上界が成立することが示されている．

4.3 混合法

今までは，データ系列 x_1^n に対して，最適なパラメータ θ を持つ確率分布 $P_\theta(x_1^n)$ を 1 つ推定して，それを用いて x_1^n を算術符号化することを考えていた．しかし，複数のモデル θ に対して確率分布を推定し，それを最適な事前分布 $w(\theta)$ で混合して得られる混合分布 $Q_w(x_1^n) = \int w(\theta) P_\theta(x_1^n) \mathrm{d}\theta$ を用いて符号化する方法が考えられる．このような符号をベイズ符号という．具体的に混合法を用いたユニバーサル符号化が幾つか示されているが [50][63]，ここでは CTW(Conext-Tree Weighting) 符号 [66][67] について，以下にそのアルゴリズムを紹介する．

$\mathcal{X} = \{0,1\}$ とし，文脈木 T は全ての葉が深さ $D_{\max}$ を持つ完全木とする．なお，内部節点 s の子供の節点を s_0, s_1 で表すことにする．また，$x_{1-D_{\max}}^0$ は与えられているものとする．

$t = 1, 2, \cdots$ に対して，以下を繰り返す．

Step-B1 $x_{t-1} x_{t-2} \cdots$ に従って木をたどり，その各節点 s に対して，頻度を更新する．

Step-B2 頻度を更新した節点 s の確率 $P_s(x_1^n)$ を KT 推定により，次ぎのように求める．

$$P_s(x_1^t) = \frac{n_s(x_t) + 1/2}{n_s(0) + n_s(1) + 1} P_s(x_1^{t-1})$$

なお，更新しなかった状態 s' は，$P_{s'}(x_1^t) = P_{s'}(x_1^{t-1})$ である．

Step-B3 各節点 s に対して，混合確率 $Q(x_1^t)$ を次のように求める．ただし，$D(s)$ は状態 s の深さを示す．

$$Q_s(x_1^t) = \begin{cases} \frac{P_s(x_1^t)}{2} + \frac{Q_{s_0}(x_1^t) Q_{s_1}(x_1^t)}{2}, & 0 \leq D(s) < D_{\max} \\ P_s(x_1^t), & D(s) = D_{\max} \end{cases}$$

x_t の符号化は，根の状態 s_r の混合確率 $Q_{s_r}(x_1^t)$ から，$Q_{s_r}(x_t | x_1^{t-1}) = Q_{s_r}(x_1^t) / Q_{s_r}(x_1^{t-1})$ を求めて算術符号化する．

なお，最大深さ $D_{\max}$ を持つ任意の木情報源から出力された系列 x_1^n を CTW で符号化したとき，その冗長度が $(|\mathcal{S}| \log n)/2 - (|\mathcal{S}| \log |\mathcal{S}|)/2 + |\mathcal{S}| - 1 + |\{s : s \in \mathcal{S}, D(s) \neq D_{\max}\}| + 3$ 以下になることが示されている [66]．また，文献 [58] には，CTW 符号の混合重み $w(\mathcal{S})$ がどのような重みになっているかが解析されている[3].

4.4 PPM

上記の手法は 2 値の木情報源に適用可能であるが，実際のファイル圧縮などに適用するためには，バイト（あるいはワード）単位で深さが少なくとも 3〜5 程度以上の木情報源を取り扱えなくてはならず，現時点の計算機性能では，計算時間/メモリ量の点から実用的ではない．

これに対して，アルファベットサイズの大きな木情報源で，モデル選択を簡単に行えるようにしたものが，PPM(Prediction by Partial Matching)[13] である．

例えば，木の最大深さが 3 で，“$x_{t-2}x_{t-1}x_t$ =sio”のときに x_{t+1} =d の文字を符号化する場合を考える．木の根から $x_t x_{t-1} x_{t-2}$ とたどったときの文字の頻度が次のようになっているもとする．

深さ	文脈	n	t	u	d	b	c	a	合計
3	sio	9	2						11
2	io	10	2	3					15
1	o	15	6	18	25	1			65
0		30	15	41	68	24	72	32	282

このとき，木の深さ D が最も深いところから考える．$D = 3$ では，文字 d はまだ出現していないので，特殊記号 ESC を符号化して非出現文字であることを示す．ESC に頻度 1 を割り当て，ESC の確率を 1/12 とする．次に，$D = 2$ では，やはり d は非出現文字であるが，$D = 3$ のときに，x_{t+1} は “n,t” でないことが分かるので，それらを除いた頻度で考え，ESC に 1/4 を割り当てる．$D = 1$ では，u も除いて考えるので，d に 25/27 を割り当てる．したがって，$P(\mathrm{d} | s(x_1^t)) = (1/12)(1/4)(25/27)$ となり，この確率を用いて文字 d を算術符号化する．なお，$D = 0$ でも頻度がゼロの場合は，d の ASCII コードを送る．

深さ D での文字 x の頻度を $n_D(x)$ で表わし，その深さで初めて頻度が 1 以上になる文字の集合を $\mathcal{A}_D$ とし，$M_D = \sum_{x \in \mathcal{A}_D} n_D(x)$ とすると，上記の方式 (Method-A[13]) の ESC の確率は $1/(M_D + 1)$，$x \in \mathcal{A}_D$ の確率は $n_D(x)/(M_D + 1)$ である．

PPM の性能は，ESC の確率の割り当て方により大き

[3] ベイズ符号を含め，確率推定を行うユニバーサル符号の理論的な解析は，[26] に紹介されている．

く変化するが，Method-A 以外に次のような ESC 確率の割り当て法が考えられている．ただし，深さ D で頻度が 1 以上の文字の種類数を r_D とし，頻度が 1 の文字の種類数を t_D とする．

Method-B[13]:$(r_D-t_D)/M_D$，Method-C[43]:$r_D/(M_D+r_D)$，Method-X[68]：t_D/M_D．

ここで，Method-X は，ESC の確率として，ポアッソン過程で出現する文字がゼロ頻度となる確率を用いる方式 Method-P[68] の第一次近似である．

PPM では，木の最大の深さが予め決まっているものとしているが，Patricia 木を使って，深さに制限のないように改良した方式 (PPM*)[14] なども提案されている．

5 辞書法

長さ n のデータ系列 x^n のうち，x_1^{s-1} は符号化が終わり，x_s^n をこれから符号化する部分とすると，辞書法では次のステップを繰り返すことにより，符号化される．

Step-C1 x_s^n から，その語頭部 x_s^t をある規則に基づいて分解 (parsing) する．

Step-C2 x_s^t を現時点の辞書を用いて符号化する．

C2-1 x_s^t が，辞書内のどの部分と一致するかを示す整数値に符号化する．

C2-2 その整数値を 2 値符号語に符号化する．

Step-C3 x_s^t を用いて辞書を更新する．

C2-2 の符号化には，動的なエントロピー符号化や正整数のユニバーサル表現が用いられる．符号化と同じルールで辞書を更新しながら，次のように復号すれば，辞書を送らなくても符号化側と同じ辞書が復号側で逐次的に作成でき，正しく復号できる．

Step-D1 符号語系列から, 元の部分系列 1 つ分に対応した符号語を取り出す.

Step-D2 現時点の辞書を用いて, その符号語から元の部分系列 x_s^t を復号する.

Step-D3 復号した x_s^t を用いて辞書を更新する.

LZ77 符号 [74] では，辞書としてバッファーを用いており，x_1^{s-1} が貯えられたバッファー内で，x_s^n と最大一致する系列で分割する．その最大一致が，$x_s^{t-1}=x_{s-p}^{t-p-1}$ のとき，x_s^t を，インターバル値 p と一致長 $t-s$，非一致シンボル x_t を用いて，$(p,t-s,x_t)$ と符号化する．このような単純なバッファーを辞書として用いる符号から，Fiala-Greene 符号 [23] のように辞書として Patricia 木を用いる符号など，辞書法に分類されるユニバーサル符号には非常に多数の符号があり，また，それらの符号には性能を改善するためのさまざまなアイデアが盛り込まれている．それらの詳細は，文献 [70] に詳しく述べてあるので，ここでは全て省略する．

Lempel-Ziv 符号の冗長度（符号語長とエントロピーレートの差）を，理論的に評価する研究も数多くされている．LZ77 符号に関しては，解析を簡単にするため，次のような固定辞書（Fixed Database）方式 (FDLZ 符号) に対する評価がなされている．データ系列 X_1^∞ に対して，それと独立でかつ同じ確率分布を持つトレーニング系列 Z_1^n を固定辞書として使用する．このとき，FDLZ 符号の冗長度は，有限次数のマルコフ情報源に対して $O(\log\log n/\log n)$ となることが，Wyner-Wyner[69] により示されている．さらに彼等は，最大一致長と 2 番目の一致長との差を符号化するように改良した Bender-Wolf[7] の符号を固定辞書化すると，冗長度が $O(1/\log n)$ に改善されることを示した．

一方，増分分解を利用した LZ78 符号 [75] に関しては，Savari[53] が，データ長が n である有限状態ユニフィラー情報源出力に対して，冗長度が $O(1/\log n)$ で与えられることを示した[4]．

これらの収束性能は，MDL 符号化が保証した $O(\log n/n)$ に比べてかなり収束が悪い．この一因は次のことに起因する．辞書法では一度に符号化される部分系列 x_s^t 内の相関はうまく利用されているが，x_s^t の直前の文脈 $\cdots x_{s-2}x_{s-1}$ を積極的に利用していない．次節で述べるソート法は，積極的に文脈を利用するような符号である．

6 ソート法

辞書法では，辞書内のデータ位置は，登録順またはデータ系列 x_s^t のアルファベット順で表現されていることが多い．ソート法は，x_s^t の文脈 x_1^{s-1} を用いてソートし，符号化は，類似文脈内における相対的な位置を指定してデータを符号化する方式である．

ソートを用いる符号には，ブロックソート符号 (BS 符号)[9], 文脈ソート符号 (CS 符号)[73], ACB 符号 [10][52] などが存在するが，そのうち本節では主にブロックソート符号を取り上げ解説を行う．

6.1 BS 符号の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．

Step-E1 $\boldsymbol{x}$ を n 回巡回シフトして，n 個の系列を作る．

Step-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作る．ソートテーブルにおける $\boldsymbol{x}$ の行番号

[4] 文献 [62] で，LZ78 符号の漸近的性能評価の解説がなされている．

表 3: 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	$racadabraab$	$r_2\cdots\cdots\cdots b_2$

を m とし，ソートテーブルの最終列の系列を $\boldsymbol{y}$ とする．

Step-E3 $\boldsymbol{y}$ を Recency Rank(RR) 符号 [19](MTF 符号 [8]) 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．

Step-E4 m と $\boldsymbol{z}$ を 2 値符号化する．

$\boldsymbol{x} = abracadabra$ に対する Step-E2 のソートテーブルを表 3 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．また，Step-E3 の RR 値の系列 $\boldsymbol{z}$ は，アルファベット $\mathcal{A}$ が $\mathcal{A} = \{a, b, c, d, r\}$ なので，次のような系列となる．

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

Step-E4 における $\boldsymbol{z}$ の符号化は，無記憶系列と見なして，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符号などが用いられる．

6.2 BS 符号の復号アルゴリズム

復号は，符号化の手順を逆に行えばよい．

Step-F1 2 値符号語系列から，m と $\boldsymbol{z}$ を求める．

Step-F2 RR 符号の復号により，$\boldsymbol{z}$ から $\boldsymbol{y}$ を求める．

Step-F3 $\boldsymbol{y}$ をソートして，ソートテーブルの最初の列 $\boldsymbol{u}$ を求める．

Step-F4 $\boldsymbol{u}$ 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．

表 4: $\boldsymbol{y}$ 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		$dabra\cdots a_5$

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 $\boldsymbol{u}$ のどの 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 4 に示すように同一種類のシンボルは $\boldsymbol{u}$ における出現順と $\boldsymbol{y}$ 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 3 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 3 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．

6.3 BS 符号の性能およびバリエーション

BS 符号で効率よく圧縮できる理由は次のように説明できる．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···” 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．また文脈の変化に対して，RR 符号化が迅速な追随特性を有している．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．

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 の情報は不必要となる [73]．

BS 符号では，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 [54]．また，Suffix array を利用する方法などもある [51]．

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 [22]．この性質を利用して，Run-Length 符号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 符号の代わりに，Inversion Frequencies という符号化法を用いる手法もある [2]．

BS 符号で作られる $\boldsymbol{y}$ は，同じ文脈を持つ文字が集り，その同一文脈内での文字の頻度が元の系列と変化しない性質を利用して，Arimura-Yamamoto[3][4] は，シンボル拡大を行えば，定常全エルゴード情報源に対して，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を示した．さらに，Effros ら [17] は，文脈の変化点を符号化し，文脈ごとで独立な算術符号で符号化するような BS 符号を考え，有限状態の定常エルゴード情報源に対して，その冗長度が $O(\log n/n)$ となることを示した．また，符号化が簡単となるように，算術符号の切り替えを文脈と無関係に長さ $w(n)$ ごとで切り替える符号化法を考え，$w(n) = O(\sqrt{n \log n})$ のときには冗長度が $O(\sqrt{\log n/n})$ となることも示している．

BS 符号は，長さ n の系列を全て読み込んでから，一度にソートを行うためオフライン方式となる．これに対して，1 シンボルごとにソートと符号化を繰り返してオンラインで動作可能なアルゴリズムに改良したものが，文脈ソート符号 (CS 符号)[73] である．BS 符号では，x_i の符号化に系列 x_1^n 全体の文脈が利用できるのに対して，CS 符号では x_1^{i-1} の文脈だけを使用することになる．しかし，BS 符号では，ソートされた $\boldsymbol{y}$ の各シンボル y_j の符号化は，$y_1 \cdots y_{j-1}$ を利用して圧縮するため，y_{j+1} 以後に対応した文脈は利用できていない．これに対して，CS 符号では，x_1^{i-1} 内の文脈を全て利用できるように工夫されている．なお，CS 符号に関しても，シンボル拡大を行えばエントロピーレートまで圧縮可能なことが示されているが，冗長度などの詳細な解析はまだ行われていない．

BS 符号と CS 符号は，$\boldsymbol{y}$ を 1 シンボル（シンボル拡大をする場合は固定長）ごとに符号化を行っている．これに対して，LZ77 符号にソートを利用した ACB 符号 [10][52] では符号化する単位が可変長となっている．LZ77 符号では，x_s^n に対する最大一致系列が $x_{s-p}^{t-p-1} = x_s^{t-1}$ のとき，$(p, t-s, x_t)$ というようにインターバル値 p を用いて符号化されるが，ACB 符号では各時刻 i ごとの文脈 x_1^i をソート順に並べ変え，その順序を利用して番号付けするように工夫されている．LZ77 符号は，実用的には Bell 符号 [6] のように，非圧縮文字 x_t を符号語に含めないとか，最大一致系列を高速に検索できるデータ構造を作るなどの工夫が必要であるが，ACB 符号に対しては，そのような検討がまだほとんどされておらず，今後の課題となっている．

7 文法法

1990 年代後半からは，文法を利用する圧縮法の研究が盛んに行われている．x_1^n を生成できる文脈自由文法を作ることができれば，その文法を符号化して送ることにより，復号器側で x_1^n を復号できる．

その代表的なものとして，SEQUITUR[40][41] と MPM [31] を紹介する．SEQUITUR はデータ系列の最初から逐次的に文法を作って行くのに対し，MPM は全系列を読み込んでから文法が作成される特徴がある．

7.1 SEQUITUR

次のような文法が与えられれば S から $\boldsymbol{x} = abcdbcabcdbcabc$ を生成できる．

$$\begin{cases} S & \rightarrow & CCB \\ A & \rightarrow & bc \\ B & \rightarrow & aA \\ C & \rightarrow & BdA \end{cases}$$

したがって，この文法を $\boldsymbol{x}$ の符号語とすることができる．$\boldsymbol{x}$ を生成する文法は無数に考えられるが，SEQUITUR では，無駄な規則を生成しないように次のアルゴリズムにより，文法を作成する．

Step-G1 データ系列から 1 文字読み込み，その文字を S のルールに追加する．

Step-G2 既にルールのどこかに存在している digram(2 文字対)$\alpha\beta$ が生成されたときは，次のようにルールを修正する．

G2-1 他方の digram が，$A_i \rightarrow \alpha\beta$ の形をしたルールの場合は，生成された digram を A_i で置き換える．

G2-2 そうでない場合は，新しいルール $A_j \rightarrow \alpha\beta$ を作り，二つの digram を A_j で置き換える．

Step-G3 一度しか使用されないルール $A_k \to \alpha\beta\cdots\gamma$ が存在した場合，そのルール A_k を削除し，A_k のところに $\alpha\beta\cdots\gamma$ を代入する．

$\boldsymbol{x} = abcdbcabcd$ の場合に対して，$S \to abcdb$ 以後の生成例を下記に示す．下線を引いたところは，上記のアルゴリズムで，ルールの変更が生じる箇所である．

$$\left\{ \begin{array}{lcl} S & \to & abcdb \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & a\underline{bc}d\underline{bc} \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & aAdA \\ A & \to & bc \end{array} \right.$$

$$\vdots$$

$$\left\{ \begin{array}{lcl} S & \to & aAdAa\underline{bc} \\ A & \to & \underline{bc} \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & \underline{aA}dA\underline{aA} \\ A & \to & bc \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & BdAB \\ A & \to & bc \\ B & \to & aA \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & \underline{Bd}A\underline{Bd} \\ A & \to & bc \\ B & \to & aA \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & CAC \\ A & \to & bc \\ B & \to & aA \\ C & \to & \underline{B}d \end{array} \right.$$

$$\left\{ \begin{array}{lcl} S & \to & CAC \\ A & \to & bc \\ C & \to & aAd \end{array} \right.$$

文法を最後に２値系列に符号化するとき，SEQUITUR では同じ digram が２度以上出現しないため，単純な無記憶モデルを用いてエントロピー符号化している．

なお，SEQUITUR を用いれば，英文テキスト，音楽の楽譜，フラクタルのグラフ構造を表現するＬシステムなどに対して，その文法的な多層構造をうまく抽出できることが，例を用いて示されている [41]．

SEQUITUR の圧縮性能は，理論的には解明されていない．しかし，Kieffer と Yang[30][72] は，より厳密な文法の簡約化規則 (reduction rule) を定めると共に，既に生成されている文法に最大一致する系列で，新しいデータを切り出して付加し，簡約化規則を適用するという greedy grammar transform を考え，ある条件を満たす既約文法を用いる場合，冗長度の最悪値が $O(\log\log n/\log n)$ となることを示した．

また，文脈自由文法を用いたデータ系列の表現方法が，pointer tree や data flow graph を用いたデータ表現と等価であり，簡単化のためにそれぞれの簡約化手法が利用できることが示されている [32]．

7.2 MPM

MPM(Multilevel Pattern Matching) の符号化は，全系列 $\boldsymbol{x}$ を読み込んだ後，次のアルゴリズムでトップダウンに文法を作成する．

r と I を与えられた整数値として，$i = 0, 1, \cdots, I$ に対して下記を繰り返す．

Step-H1 $i = 0$ のとき，$\boldsymbol{x}$ を長さ r^I の部分列に分割して，その並びを S_0 とする．(半端分の残りがあれば，それを R_0 とする．)

$i > 0$ のとき，S_{i-1} に含まれる部分系列のうち，異なるものを，それぞれ r 分割したものの並びを S_i とする．(R_{i-1} がある場合は，その中の長さ r^{I-i} の部分系列も S_i に含め，残りを R_i とする．)

Step-H2 $S_i(\boldsymbol{x})$ に含まれる部分系列のうち，同じものには同じ token を割り当て，異なるものには異なる token を割り当てる．その token の並びを T_i とする．

Step-H3 各 T_i をエントロピー符号化する．

$\boldsymbol{x} = 00100100100000010010000010010010$, $r = 2$, $I = 4$ の場合に対する具体例を下記に示す．なお下線が引かれた文字列は，左側に同じ系列が存在することを示す．

$S_0(\boldsymbol{x}) = (0010010010000001, 0010000010010010)$
$S_1(\boldsymbol{x}) = (00100100, 10000001, 00100000, 10010010)$
$S_2(\boldsymbol{x}) = (0010, 0100, 1000, 0001, \underline{0010}, 0000, 1001, \underline{0010})$
$S_3(\boldsymbol{x}) = (00, 10, 01, \underline{00}, \underline{10}, \underline{00}, \underline{00}, \underline{01}, \underline{00}, \underline{00}, \underline{10}, \underline{01})$
$S_4(\boldsymbol{x}) = (0, 0, 1, 0, 0, 1)$
$T_0 = (t_0, t_1)$
$T_1 = (t_0, t_1, t_2, t_3)$
$T_2 = (t_0, t_1, t_2, t_3, t_0, t_4, t_5, t_0)$
$T_3 = (t_0, t_1, t_2, t_0, t_1, t_0, t_0, t_2, t_0, t_0, t_1, t_2)$
$T_4 = (0, 0, 1, 0, 0, 1)$

復号は，T_i の各異なる token に T_{i+1} の r 個の token を左から順に対応付けていくことにより行える．(その対応付けが，文法規則となる．)

次に，$\boldsymbol{x}$ のデータ長が r^I の倍数でない例を示す．

$S_0(\boldsymbol{x}) = (0000000100000001), 0001011$
$S_1(\boldsymbol{x}) = (00000001, \underline{00000001}), 0001011$
$S_2(\boldsymbol{x}) = (0000, 0001, \underline{0001}), 011$
$S_3(\boldsymbol{x}) = (00, \underline{00}, \underline{00}, 01, \underline{01}), 1$

$S_4(\boldsymbol{x}) = (0,0,0,1,1)$
$T_0 = (t_0)$
$T_1 = (t_0, t_0)$
$T_2 = (t_0, t_1, t_1)$
$T_3 = (t_0, t_0, t_0, t_1, t_1)$
$T_4 = (0,0,1,1)$

MPM の冗長度に関しては最悪値で $O(1/\log n)$ であることが示されている．なお，Step-H3 において，T_i に含まれる token の系列をエントロピー符号化するときには，異なる token は $t_0, t_1, t_2, \cdots$ と順に出現するため，token の種類を符号化する必要はなく，新しい token であることを示す特殊記号 ESC を符号化すればよい．実際の圧縮率は，その ESC 確率の推定法にかなり影響を受けるため，注意する必要がある．

上で紹介した SEQUITUR や MPM 以外にも，多くの方式が研究されている．例えば，SEQUITOR をオンライン化したもの [38]，digram のうち最も頻度の大きいものを一つの文字で置き換える方式 [39]，N-gram の部分集合を用意して符号化していくもの [59] などがある．また，対象とする系列をある言語（例えば Pascal）に限り，それに適した文脈自由文法を構成してその文法を符号器復号器で共有し，どの文法が適用されたかを符号化する方式 [11] なども提案されている．

8 まとめ

本稿では，基本的なユニバーサルデータ圧縮アルゴリズムを紹介した．紙面の都合上，本稿では触れられなかったさまざまな符号やアルゴリズが [1][26] [52][61] [70][71] などに紹介されているので，あわせて参考にして頂きたい．

参考文献

[1] J.Abrahams (和訳：山本博資), “ハフマン符号木に関連した話題,” 応用数理, 8 巻 2 号（データ圧縮特集号）, pp.4-20, June 1998.

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

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

[4] M.Arimura and H.Yamamoto “Almost Sure Convergence Coding Theorem for Block Sorting Data Compression Algorithm,” *Proc. 1998 Int. Symp. Infor. Theory and Its Appl.(ISITA98)*, Mexico City, Mexico, Oct. 1998.

[5] A.Barron, J.Rissanen, and B.Yu, “The Minimum Description Length Principle in Coding and Modeling”, *IEEE Trans. Inform. Theory*, vol.44, no.6, pp.2743-2760, Oct. 1998.

[6] T.C.Bell, “Better OPM/L Text Compression,” *IEEE Trans. Communications*, vol.COM-34, no.12, pp.1176-1182, Dec. 1986.

[7] P.E.Bender and J.K.Wolf, “New Asymptotic Bounds and Improvements on the Lempel-Ziv Data Compression Algorithm,” *IEEE Trans. Inform. Theory*, vol.37, no.3, pp.721-729, May 1991.

[8] J.L.Bentley, D.D.Sleator, R.E.Tarjan, and V.K.Wei, “A Locally Adaptive Compression Scheme,” *Commun. ACM*, vol.29, no.4, pp.320-330, April 1986.

[9] 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.

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

[11] R.Cameron, “Source Encoding Using Syntactic Information Source Models,” *IEEE Trans. Inform. Theory*, vol.34, pp.843-850, Jul. 1988.

[12] J. Capon, “A Probabilistic Model for Run-Length Coding of Pictures,” *IRE Trans. Inform. Theory*, vol.5, no.5, pp.157-163, Dec. 1959.

[13] J.G.Cleary and I.H.Witten, “Data Compression Using Adaptive Coding and Partial String Matching,” *IEEE Trans. Communications*, vol.COM-32, no.4, pp.396-402, April 1984.

[14] J.G.Cleary, W.J.Teahan, and I.H.Witten, “Unbounded Length Contexts for PPM,” *Proc. IEEE DCC*, pp.52-61, 1995.

[15] T.Cover, “Enumerative Source Coding,” *IEEE Trans. Inform. Theory*, vol.IT-19, pp.73-76, Jan. 1973.

[16] L.D.Davisson, “Comments on 'Sequence Time Coding for Data Compression',” *Proc. IEEE*, vol.54, p.2010, Dec. 1966.

[17] M.Effros, K.Visweswariah, S.R.Kulkarni, and S.Verdú, “Universal Lossless Source Coding with the Burrows Wheeler Transform,” submitted to *IEEE Trans. Inform. Theory*

[18] P.Elias, “Universal Codeword Sets and Representations of the Integers”, *IEEE Trans. Inform. Theory*, vol.IT-21, no.2, pp.194-203, March 1975.

[19] P.Elias, “Interval and Recency Rank Source Coding: Two On-line Adaptive Variable-Length Schemes”, *IEEE Trans. Inform. Theory*, vol.IT-33, no.1 pp.3-10, Jan. 1987.

[20] F.Fabris, A.Sgarro, and R.Pauletti, “Tunstall adaptive coding and miscoding,” *IEEE Trans. Inform. Theory*, vol.42, no.6, pp.2167-2180 Nov. 1996.

[21] N.Faller, “An Adaptive System for Data Compression”, *Record of the 7-th Asilomar Conference on Circuits, Systems and Computers*, pp.593-597, 1973.

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

[23] E.R.Fiala and D.H.Greene, “Data Compression with Finite Windows”, Comm. of the ACM, vol.32, no.4, pp.490-505, April, 1989.

[24] B.M.Fitingof, “Optimal Coding in the Case of Unknown and Changing Message Statistics,” *Probl. Inform. Transm.*, vol.2, no.2, pp.3-11 (in Russian), pp.1-7 (English Trans.), 1966.

[25] R.G.Gallager, “Variations on a Theme by Huffman”, *IEEE Trans. Inform. Theory*, vol.IT-24, no.6, pp.668-675, Nov. 1978.

[26] 韓太舜，小林欣吾, “情報と符号化の数理”, 培風館，1999.

[27] D.A.Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proc. of IRE* vol.40, pp.1098-1101, Sep. 1952.

[28] F.Jelinek, *Probabilistic Information Theory*, macGraw-Hill, New York, 1968.

[29] 金谷文夫, “ひずみを許すユニバーサル符号化,” 電子情報通信学会和文論文誌A, vol.J84-A, no.6, pp.691-697, June 2001.

[30] J.C.Kieffer and E.Yang, “Grammar-Based Codes: A New Class of Universal Lossless Source Codes,” *IEEE Trans. Inform. Theory*, vol.46, no.3, pp.737-754, May 2000.

[31] J.C.Kieffer, E.Yang, G.Nelson, and P.Cosman, “Universal Lossless Compression via Multilevel Pattern Matching,” *IEEE Trans. Inform. Theory*, vol.46, no.4, pp.1227-1245, July 2000.

[32] J.C.Kieffer, “A Study of Advances in Hierarchical Data Compression,” 2000

[33] A.N.Kolmogorov, “Three Aproaches to the Quantitative Definition of Information,” *Prob. Inform. Trans.*, vol.1, pp.4-7, 1965.

[34] V.I.Levenshtein, “On the redundancy and delay of decodable coding of natural numbers,” Problem of Cybernetics, vol.20, pp.149-155, 1968.

[35] R.E.Krichevsky and V.K.Trofimov, “The Performance of Universal Encoding,” *IEEE Trans. Inform. Theory* vol.IT-27, no.2, pp.199-207, March 1981.

[36] T.J.Lynch, “Sequence Time Coding for Data Compression,” *Proc.IEEE*, vol.54, pp.1490-1491, Oct. 1966.

[37] B.MacMillan, “The Basic Theorems of Information Theory,” *Ann. Math. Statistics*, vol.24, pp.196-219, 1953.

[38] 永山，伊藤，橋本，“Digram に基づいた無歪みデータ圧縮,” 第18回情報理論とその応用シンポジウム予稿集，pp.573-576, 花巻, 1995.

[39] 中村，村島，“繰り返し現れる隣接文字の連結に基づくデータ圧縮法,” 電子情報通信学会論文誌A，vol.J79-A, no.7, pp.1319-1323, July 1996.

[40] C.G.Nevil-Manning, I.H.Witten, and D.L.Maulsby, “Compression by Induction Hierachical Grammars,” *Proc. IEEE DCC'94*, pp.244-253,Snowbird, Utah, USA, March 1994.

[41] C.G.Nevil-Manning and I.H.Witten, “Compression and Explanation Using Hierachical Grammars,” *The Computer Journal*, vol.40, no.2/3, pp.104-116, 1997.

[42] C.G.Nevil-Manning and I.H.Witten, “Identifying Hierarchical Structure in Sequences: A Linear-Time Algorithm,” *J. of Artificial Inteligence Research*, vol.7, pp.67-82, 1997.

[43] A.Moffat, “Implementing the PPM Data Compression Scheme,” *IEEE Trans. Commun.*, vol.38, no.11, pp.1917-192, Nov. 1990.

[44] R.Pasco, “Source Coding Algorithms for Fast Data Compression,” Ph.D. Thesis, Stanford University, 1976.

[45] J.Rissanen, “General Kraft Inequality and Arithmetic Coding,” *IBM J. Res. Devel.*, vol.20, pp.198-203, 1976.

[46] J.Rissanen, “Universal Data Compression System,” *IEEE Trans. Inform. Theory*, vol.IT-29, no.5, pp.656-664, Sept. 1983.

[47] J.Rissanen, “A Universal Prior for Integers and Estimation by Minimum Description Length,” *An. Statist.*, vol.11, no.2, pp.416-431, June, 1983.

[48] J.Rissanen, “Universal Coding, Information, Prediction, and Estimation,” *IEEE Trans. Inform. Theory*, vol.IT-30, no.4, pp.629-636, July 1984.

[49] J.Rissanen, “Complexity of Strings in the Class of Markov Sources,” *IEEE Trans. Inform. Theory*, vol.IT-32, no.4, pp.526-532, July 1986.

[50] B.Y.Ryabuko, “Twice-Universal Coding,” *Prob. Inform. Trans.*, vo.20, no.3, pp.24-28, July-Sept. 1984.

[51] 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.

[52] D.Salomon, “Data compression, The Complete Reference,” (2nd Ed.) Springer-Verlag, New York, 2000.

[53] S.A.Savari, “Redundancy of the Lempel-Ziv Incremental Parsing Rule,” *IEEE Trans. Inform. Theory*, vol.43, no.1, pp.9-21, Jan. 1997.

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

[55] C.E.Shannon, “A Mathematical Theory of Communications,” *B.S.T.J.*, vol.27, pp.379-423, vol.28, pp.623-625, 1948.

[56] Y.M.Shtarkov, “Universal Sequential Coding of Single Message,” *Prob. Inform. Transm.*, vol.23,no.3, pp.3-17, July-Sept. 1987.

[57] P.R.Stubley, “Adaptive data compression using tree codes,” Univ.of Waterloo, Dep.of Elec. Eng., PhD thesis, 1992.

[58] J.Suzuki, “A Relation between Context Tree Weighting and General Model Weighting Techniques for Tree Sources,” *IEICE Trans. on Fundamentals*, vol.E81-A, no.11, pp.2412-2417, Nov. 1998.

[59] C.Y.Teng and D.L.Neuhoff, “An Improved Hierarchical Lossless Text Compression Algorithm,” *Proc. IEEE DCC'95*, pp.292-301, Snowbird, Utah, USA, March 1995.

[60] B.P.Tunstall, “Synthesis of Noiseless Compression Codes,” Ph.D. Thesis, Georgia Institute of Technology, 1968.

[61] 植松友彦, 文書データ圧縮アルゴリズム入門，CQ 出版，1994.

[62] 植松友彦, “Lempel-Ziv 符号と情報理論,” 電子情報通信学会和文論文誌A, vol.J84-A, no.6, pp.681-690, June 2001.

[63] M.J.Weinberger, N.Merhav, M.Feder, “Optimal Sequential Probaility Assignment for Individual Sequences,” *IEEE Trans. Inform. Theory*, vol.40, no.2, pp.384-396, March 1994.

[64] M.J.Weinberger, J.J.Rissanen, and M.Feder, “A Universal Finite Memory Source,” *IEEE Trans. Inform. Theory*, vol.41, no.3, pp.643-652, May 1995.

[65] T.A.Welch, “A Technique for High-Performance Data Compression”, *IEEE Computer*, vol.17, no.6, pp.8-19, June 1984.

[66] F.M.J.Willems, Y.M.Shtarkov, and T.J.Tjalkens, “The Context Tree Weighting Method: Basic Properties,” *IEEE Trans. Inform. Theory*, vol.41, no.3, pp.653-664, May 1995.

[67] F.M.J.Willems, Y.M.Shtarkov, and T.J.Tjalkens, “Context Weighting for General Finite-Context Sources,” *IEEE Trans. Inform. Theory*, vol.42, no.5, pp.1514-1520, Sept. 1996.

[68] I.H.Witten and T.C.Bell, “The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression,” *IEEE Trans. Inform. Theory*, vol.37, no.4, pp.1085-1094, July 1991.

[69] A.D.Wyner and A.J.Wyner, “Improved Redundancy of a Variation of the Lempel-Ziv Algorithm,” *IEEE Trans. Inform. Theory*, vol.41, no.3, pp.723-732, May 1995.

[70] 山本博資, “ユニバーサルデータ圧縮アルゴリズム: 原理と手法,” 情報処理, vol.35, no.7, pp.600-608, July 1994.

[71] 山本博資，“データ圧縮と正整数のユニバーサル表現,” 「情報源符号化, 無歪みデータ圧縮」(情報理論とその応用学会編)，4章，pp.53-78, 1998.

[72] E.Yang and J.C.Kieffer, “Efficient Universal Lossless Data Compression via Textual Substitution,” *IEEE Trans. Inform. Theory*, vol.46, no.3, pp.755-777, May 2000.

[73] H.Yokoo and M.Takahashi, “Data Compression by Context Sorting,” *IEICE Trans. on Fundamentals*, Vol.E78-A, No.5, pp.681-686, May 1996.

[74] J.Ziv and A.Lempel, “A Universal Algorithm for Sequential Data Compression”, *IEEE Trans. Inform. Theory*, vol.IT-23, no.3, pp.337-343, May 1977.

[75] J.Ziv and A.Lempel, “Compression of Individual Sequences via Variable-Rate Coding”, *IEEE Trans. Inform. Theory*, vol.IT-24, no. 5, pp.530-536, Sep. 1978. (U.S. patent 4,464,650)