アルゴリズム 75－9
（2000．11．10）

CTW 圧縮における重み付け変数の最適化

岡崎 巧　今井 浩

東京大学大学院理学系研究科情報科学専攻
〒113-0033 東京都文京区本郷 7-3-1
TEL: 03-5841-4102　FAX: 03-5800-6933
{takumi,imai}@is.s.u-tokyo.ac.jp

要旨

現在，様々な文書圧縮法が開発されているが，計算機の性能の向上につれ，圧縮率の高い手法が実用的になってきている．CTW 圧縮は，速度には難があるものの，特に圧縮率の優れた圧縮方法の一つである．CTW では符号化の直前の文字列 (文脈) から符号化確率を推定するが，何文字の文脈についての情報を重視するかを左右する重み付け変数の値の決め方が圧縮効率を左右する問題の一つである．本研究では，その変数の値を各長さの文脈の情報に従って定義する手法を提案し，実験により安定した圧縮率を達成ができることを示す．

Optimization of Weighting Parameters for CTW Data Compression

Takumi Okazaki　Hiroshi Imai

Department of Information Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
TEL: 03-5841-4102　FAX: 03-5800-6933
{takumi,imai}@is.s.u-tokyo.ac.jp

Abstract

Nowadays, many data compression methods are developed. Data compression methods which have good compression ratio are being useful because the performance of computers is improved. It is well known that CTW data compression is one of the best compression schemes regarding the compression performance, although the compression speed of CTW is slow. CTW predicts the encoding probabilities from the strings(context) just before the encoding symbol. The weighting parameters of CTW control the compression performance. We propose a method to define these weighting parameters according to the information of context of each length, and show its stability.

1　はじめに

文章などのデータを圧縮する際には，画像や音声などのデータを圧縮するのとは違い，欠損のない圧縮方法をとる必要がある．その方法としては現在，`gzip`，`compress` などの Lempel, Ziv の提案した辞書式の圧縮の手法が広く用いられている．これらの手法が広まった一因としては，他の圧縮法に比べ，圧縮・伸張の速度の点で有利であるため，と考えることができる．

だが，昨今のマシン環境の向上により，速度では辞書式圧縮法に劣るが，より高い圧縮率を達成する bzip2 などの Block-sorting 法が広まりつつある．今後，さらに計算機の性能があがるにつれて，より高い圧縮率をもつ圧縮法が実用的になってくるということができよう．

そういった背景の中で，CTW (Context Tree Weighting) 圧縮 [13] はバイナリデータについて圧縮率の特に優れた手法として提案されており，その方法を改良することの意義も大きい．

CTW は符号化しようとする文字の直前の文字列 (文脈) から符号化の確率を推定している．与えられた文字列から数文字についての文脈モデルをつくり，そこから次の文字を予測する．その予測確率は各文字の各文脈における出現頻度によって決定される．

CTW 等の文脈情報を用いる圧縮方法の問題点として，何文字分の文脈を読めば最も圧縮率を高くできるか，ということがあげられる．圧縮するデータの種類によって，考慮すべき文脈の文字数は異なるからである．CTW については，この問題を含んだ重み付け変数が存在する．従って，この変数の定め方が CTW の問題点となっているが，本稿では，この変数を制御する手法を提案し，実験によって，その有用性を検証する．実験はサンプルデータとして英文を用いて行い，重み付け変数を一定値に定めた場合よりも安定した圧縮率が達成されることを示す．

2 CTW 圧縮

CTW [13] は FSMX 情報源に対する符号化法で，モデルが未知の場合でも漸近的に最適な圧縮方法となっている．

CTW はもともとバイナリデータに対しての符号化法である．これを多値のアルファベットに拡張することもできるが，まず，[13] でのアルファベットが 2 値の場合の符号化法を述べた上で，多値の場合への対応の手法を述べる．

2.1 文脈木

情報源アルファベット $\mathcal{A}$ について文脈木 $T \subset \mathcal{A}^*$ とは，任意の $s \in T$ において，s の任意の語尾が T の要素となることをいう．この条件を満たしてさえいれば文脈木はどんな形であってもよいのだが，簡単のため，文脈木として，深さが D で一定な二分木を考えることにする．

文脈木において，各節が文脈をあらわすが，それぞれの文脈 s について，その後の文字が 0 が a_s 回，1 が b_s 回現れたとし，これらの値を木に保存するようにする．親節 s の子は $0s, 1s$ で与えられ，子における文字の頻度については，その定義から $a_{0s} + a_{1s} = a_s$, $b_{0s} + b_{1s} = b_s$ が成り立っている．

2.2 KT- 推定量

文脈木の頻度情報をもとに，各文脈ごとに，文字列中に 0 が a_s 個，1 が b_s 個出現するような特定の文字列の確率の推定値 P_e^s を求める．この推定には Krichevsky-Trofimov(KT)- 推定量がよく用いられる．KT- 推定量は Dirichlet 分布の特別な場合を示したもので，定義としては，

$$P_e^s(a_s, b_s) := \int_0^1 \frac{1}{\pi\sqrt{(1-\theta)\theta}}(1-\theta)^a\theta^b d\theta$$

であるが，実際の解としては，以下の式で与えられる．

$$P_e^s(a_s, b_s) = \frac{(a_s - \frac{1}{2}) \cdot \cdots \cdot \frac{3}{2} \cdot \frac{1}{2} \cdot (b_s - \frac{1}{2}) \cdot \cdots \cdot \frac{3}{2} \cdot \frac{1}{2}}{(a_s + b_s)!}$$

CTW の符号化の際に P_e^s は更新されていくが，実際に上式を使う必要はない．まず，初期条件を $P_e^s(0,0) = 1$ として，1 文字情報が増えるたびに以下の漸化式

$$P_e^s(a_s + 1, b_s) = \frac{a_s + \frac{1}{2}}{a_s + b_s + 1} P_e^s(a_s, b_s)$$

$$P_e^s(a_s, b_s + 1) = \frac{b_s + \frac{1}{2}}{a_s + b_s + 1} P_e^s(a_s, b_s)$$

を使うことで容易に P_e^s の値を更新することができる．

上式の意味を以下のように捉えることもできよう．0 が a_s 個，1 が b_s 個出現したときに，次に 0 が来る確率は

$$\frac{P_e^s(a_s + 1, b_s)}{P_e^s(a_s, b_s)} = \frac{a_s}{a_s + b_s}$$

としたいところであるが，これでは $a_s = 0$ の時に 0 の符号化が不可能になってしまう．b_s についても同様のことがいえるため，互いの頻度を $\frac{1}{2}$ ずつ足すことによって，確率が 0 になるのを防いでいる．

KT- 推定量の特長としては，この確率を 0 になるのを防ぐためにかかるコストとしての冗長度が $\frac{1}{2}\log(a_s + b_s) + 1$ で常に抑えられるということが知られている．

2.3 重み付き確率

次に，P_e^s の値を基に重みづけをした確率 P_w^s を求める．これは以下のように再帰的に求められる．

$$P_w^s(x_1^t) = \begin{cases} \gamma P_e^s(x_1^t) + (1-\gamma) P_w^{0s}(x_1^t) P_w^{1s}(x_1^t) \\ \quad (\text{節 } s \text{ は葉でない}) \\ P_e^s(x_1^t) \quad (\text{節 } s \text{ は葉}) \end{cases} \tag{1}$$

ここで，γ は $0 < \gamma < 1$ を満たす実数である．このパラメタ γ はある長さの文脈からの情報と，1 文字分長い文脈からの情報をどれくらいの割合で足し合わせるかを決める値である．文脈 s よりも 1 文字分長い文脈の情報は，$0s$ の後に来る頻度と $1s$ の後に来る頻度の情報をかけあわせたものと考えられるため，

$$P_w^{0s}(x_1^t) P_w^{1s}(x_1^t)$$

の形の項が漸化式に含まれる．

x_1^t の符号は文脈 λ (λ: 空文字列．文脈木においては根に相当) での値 $P_w^\lambda(x_1^t)$ の確率を算術符号化することによって得ることができる．

図 1 は，上で述べた文脈木の例を示している．各節に並んでいる数字は上から (a_s, b_s), P_e^s, P_w^s を表している．(a_s, b_s) は，文脈 s の上で 0 が a_s 回，1 が b_s 回出現していることを意味する．枝は文脈を表しており，一番左の上から 2 段目の節 (葉) は，文脈 01 の後の情報であることを示している．上から 2 段目の P_e^s は KT- 推定量であり，(a_s, b_s) の情報のみによって計算することができる．

一番左の列は葉であるので，P_w^s は P_e^s の値と等しいように定義する．中央の列については，P_w^s は自らの P_e^s と左の列の P_w^{0s}, P_w^{1s} によ

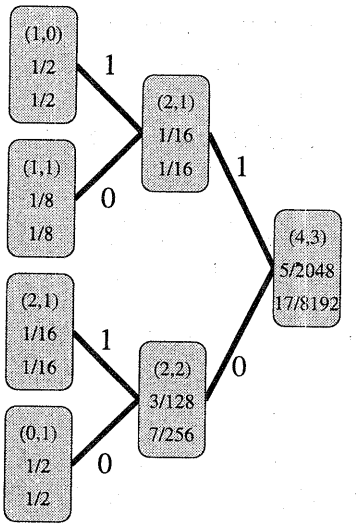

図 1: $x_1^T = 0110100$, $x_{1-D}^0 = 10$ についての 2 次 ($D = 2$) の重みつき文脈木

る漸化式によって定義される．下の段の 7/256 は P_w^0 を意味し，

$$\begin{aligned} P_w^0 &= \frac{1}{2} P_e^0 + \frac{1}{2} P_w^{00} P_w^{10} \\ &= \frac{1}{2} \cdot \frac{3}{128} + \frac{1}{2} \cdot \frac{1}{16} \cdot \frac{1}{2} = \frac{7}{256} \end{aligned}$$

によって計算できる．このように全ての節において P_w を計算することによって，一番右端の P_w の値を計算することができ，この値が算術符号に割り当てる確率となる．

2.4 アルゴリズム

[13] では，T 文字のブロック確率を算術符号化する以下のようなアルゴリズムを示している．

符号化アルゴリズム

1. 算術符号の区間の下の境界値 $B(\phi) := 0$ とおく
2. $t = 1, 2, \cdots, T$ について以下を実行する
   (a) 存在しうる文脈に対応した節を必要な分作る

(b) 算術符号の区間の計算のため，次の文字を0と仮定して $P_w^\lambda(x_1^{t-1},0)$ を計算する

(c) B を以下のように更新する

$$B(x_1^t) := \begin{cases} B(x_1^{t-1}) & (x_t=0) \\ B(x_1^{t-1}) + P_w^\lambda(x_1^{t-1},0) & (x_t=1) \end{cases}$$

(d) x_t についての情報を含むように文脈木のデータを更新し，$P_w^\lambda(x_1^t)$ を得る

3. $x_1x_2\cdots x_T$ について以上の手続きが終わったら，$B(x_1^T)$ と $P_w^\lambda(x_1^T)$ から算術符号を求める

上記のような確率の更新が，確率を過不足なく分配することは証明されている．つまり，この符号化アルゴリズムは無駄なく，さらに一意に符号化することができることがわかっている．

基本的には復号化は符号化の逆をたどればよい．

復号化アルゴリズム

1. $B(\phi) := 0$ とおき，F_∞ を求める

2. $t=1,2,\cdots,T$ について以下を実行する

(a) $d=0,1,\cdots,D$ について
節 $s(d) := x_{t-d}^{t-1}$ を必要な分作る

(b) $x^t=0$ と仮定して
$P_w^\lambda(x_1^t)$ を計算する

(c) F_∞ と $B(x_1^{t-1})+P_w^\lambda(x_1^t)$ を比較し x_t を求める

(d) B を更新する

(e) $d=0,1,\cdots,D$ について
実際の x_t で節 $s(d)$ を書き換える

3. $P_w^\lambda(x_1^T)$ から符号長を求め
次の T 文字の符号との境界を定める

文脈木のデータの更新については，全ての節を調べる必要はなく，その時の文脈に関係のある節だけを更新すればよい．その個数は木の深さ D に相当するから，このアルゴリズム全体の計算量は $O(DT)$ ですむことになる．

ブロック確率を用いれば，より理論的な数式に従った正確な確率を求めることができる反面，正確な確率計算には多倍長演算が伴い，実装が難しく，また，その計算コストも無視できない．

2.5 条件つき確率による実装

Willems ら [13] は文字の条件つき確率を利用して n 文字に対するブロック確率を求めてから算術符号化しているが，[9] では文字の条件つき確率をそのまま算術符号化する．

文字 x_t の文脈が $0s$ を含む時，文脈 s で x_t が1である確率を $P_w^s(x_t)$ は次のように計算される．

$$\begin{aligned} P_w^s(x_t) &= \frac{P_w^s(x_1^t)}{P_w^s(x_1^{t-1})} \\ &= \frac{\gamma P_e^s(x_1^t) + (1-\gamma)P_w^{0s}(x_1^t)P_w^{1s}(x_1^t)}{\gamma P_e^s(x_1^{t-1}) + (1-\gamma)P_w^{0s}(x_1^{t-1})P_w^{1s}(x_1^{t-1})} \\ &= \frac{\gamma P_e^s(x_1^{t-1})P_e^s(x_t)}{\gamma P_e^s(x_1^{t-1}) + (1-\gamma)P_w^{0s}(x_1^{t-1})P_w^{1s}(x_1^{t-1})} \\ &+ \frac{(1-\gamma)P_w^{0s}(x_1^{t-1})P_w^{0s}(x_t)P_w^{1s}(x_1^{t-1})}{\gamma P_e^s(x_1^{t-1}) + (1-\gamma)P_w^{0s}(x_1^{t-1})P_w^{1s}(x_1^{t-1})} \end{aligned}$$

ここで

$$\beta = \beta_s(x_1^{t-1}) = \frac{P_e^s(x_1^{t-1})}{P_w^{0s}(x_1^{t-1})P_w^{1s}(x_1^{t-1})}$$

と定義すると，

$$\begin{aligned} P_w^s(x_t) &= \frac{\gamma\beta P_e^s(x_t) + (1-\gamma)P_w^{0s}(x_t)}{\gamma\beta + (1-\gamma)} \\ &= \frac{\gamma\beta}{\gamma\beta+(1-\gamma)}P_e^s(x_t) + \frac{1-\gamma}{\gamma\beta+(1-\gamma)}P_w^{0s}(x_t) \end{aligned}$$

と書ける．$P_e^s(x_t)$ は x_1^{t-1} 中の文脈 s での0,1の数 a_s, b_s から

$$P_e^s(x_t) = \frac{b_s + \frac{1}{2}}{a_s + b_s + 1}$$

と計算される．また β は

$$\begin{aligned} \beta_s(x_1^t) &= \frac{P_e^s(x_1^t)}{P_w^{0s}(x_1^t)P_w^{1s}(x_1^t)} \\ &= \beta_s(x_1^{t-1}) \cdot \frac{P_e^s(x_t)}{P_w^{0s}(x_t)} \end{aligned}$$

と逐次的に計算できる．β の初期値は 1 になる．文字 x_t の符号化は 1, 0 に $[0, P_w^{\lambda}(x_t))$, $[P_w^{\lambda}(x_t), 1)$ の区間を割り当てる算術符号で行なう．算術符号化には Witten ら [14] の方法などの汎用的なものが利用できる．

β_s は 2 つの文脈での x_t が 1 である確率を混合する割合を表しているが，その値が極端に大きいまたは小さい場合には混合しても確率の値はほとんど変化しないため，β_s の値は一定の範囲に丸めても実際の圧縮率にはほとんど影響しない．

2.6 PPM の導入

CTW はバイナリのデータに対して強い威力を発揮するが，文章などのデータの場合，文脈は各ビットごとに依存しているというよりは，むしろ文字単位 (例えば 8 ビット単位) で依存している．従って，CTW を多値アルファベットに対応させることができたら，文書データに適した圧縮法が実現できることになる．

先に述べた CTW の式 (1) はそのまま多値アルファベットに応用できる．アルファベット全体の集合を $\mathcal{A}$ とすると，

$$P_w^s(x_1^t) = \begin{cases} \gamma P_e^s(x_1^t) + (1-\gamma) \prod_{a \in \mathcal{A}} P_w^{as}(x_1^t) \\ \quad (\text{節 } s \text{ は葉でない}) \\ P_e^s(x_1^t) \quad (\text{節 } s \text{ は葉}) \end{cases}$$

ここで，P_e の値をどのように推定するかが問題となる．KT-推定量のような方法では，全ての文字について確率を割り当てる必要があるため，出現頻度の極端に少ない文字などの扱いが難しい．特に特定の文脈の後は決まった文字しか来ないような文書データの場合，出現する文字と出現しない文字を同等に扱うは賢い方法とはいえない．

この問題を解決する方法として，PPM (Prediction by Partial Match) [5][6] があげられる [2][8]．PPM では escape という考えを導入する．ある文脈における文字の生起確率を予測する際，その文脈後にどの文字がどれくらいの頻度であらわれているか，という情報を用いたいところであるが，例えば，a が 4 回，b が 6 回あらわれていたからといって，同じ文脈において次に a が来る確率を 4/10 と予測することはできない．a, b 以外の文字へ割り当てられる確率が 0 となり，a, b 以外の文字が符号化できなくなってしまうためである．これを解決するために PPM では escape という仮の文字を定義する．注目している文脈後に出現したことのない文字を符号化する場合に，escape 文字を符号化し，1 文字少ない文脈の情報を見てやることにする．escape の頻度を適当な手法によって定めることによって，出現したことのない文字への確率の割り当てが実現できることになる．

もともと，PPM は単体で `gzip` などの広く用いられている辞書式圧縮法と比較して高い圧縮率をもつことが知られているため，その確率予測もより正確なものである．この PPM によって推定される確率を P_e^s とおけばより高い圧縮率を達成することができる．

3 γ の意味

もう一度，式 (1) の意味を考える．

$$\gamma P_e^s(x_1^t) + (1-\gamma) P_w^{0s}(x_1^t) P_w^{1s}(x_1^t)$$

この式の意味するところとしては，注目している文脈のみから純粋に推定できる確率と，1 文字多くした文脈についての情報をまとめることで推定できる確率を γ というパラメタを用いて重み付けをしてあげていることになる．この γ の値が大きいと，短い方の文脈を重視し，γ の値が小さいと，長い文脈を重視しているわけである．

3.1 既存研究

CTW を提案した Willems ら [13] は $\gamma = \frac{1}{2}$ の例を示し，また，γ の値のの持つ可能性について言及しているが，特にどのような γ の値が最適かが示されているわけではない．定兼ら [8] は，一文字符号化するごとに，いくつかの γ の候補値全てについて文字の生起推定確率を計算し，そのエントロピーが最も少ないときの γ を採用する方法を提案しているが，この方法では，γ の値の候補の数の回数，計算が必要になり，単純に考えてもその数の分だけ倍の時間が

かかることになり，また，その候補の数や値の根拠も薄い．

また，これらの研究では，γ は，ある文字を符号化するときにどの文脈でも同じ γ を用いているが，ある特定の文脈で特定の文字のみが出現するような場合などを考えると，γ の最適値は文脈の長さによって偏っていると推測する方が，より自然であるといえる．つまり，γ は文脈 s の関数とみなすことができれば，より正確な確率予測ができることになる．

3.2 提案手法

どの長さの文脈に重みを大きくかければ最適な γ に近づくかを考える．ある文脈には必ずこの文字が来るという文脈には高い重みをかけて，曖昧な推定しかしない文脈には低い重みをかけることが考えられる．

このような条件を実現する道具としてはエントロピーがあげられる．エントロピーは予想されるビット長をあらわしているので，ビット長を短く予想するような文脈を重い配分にするように設定してやればよい．

だが，ここで PPM の問題を見てみると，文脈の長さを決定する問題の解決法としてエントロピーよりも，その文脈中で最も頻度が高く出現する文字の確率を考える方が計算が平易であり，また，圧縮率もよくなる [4] という主張がある．理由は不明であるが，どちらを使うにせよ，偏った頻度分布の文脈ほど高い重みで確率が割り当てられる，ということがいえる．

また，これらの確率は，そのまま「その文脈はどれくらい次の文字のことをわかっているか」という指針になるため，この確率に対する比をそのまま γ の比とする．

だが，ここで問題となるのは (1) 式の形である．長い文脈の確率を見ていく時に γ が入れ子になっているため，単純に上記の確率を合計 1 に正規化した値を γ とするのは好ましくない．

この問題に対しては以下のようにする．(1) 式の形に着目すると，最も短い文脈 P_e^{λ} に対しては γ は 1 回しかかからないため，この γ を，考慮している全ての長さの文脈に対して文脈 λ の情報がとるべき重みをとることにする．つまり，上記の確率の総和中の，文脈 λ 上での上記の確率の割合を γ とする．

title\γ	0.5	0.2	0.05	mix	Entropy
bib	2.131	1.926	1.878	**1.860**	1.884
book1	2.266	2.181	2.179	2.164	**2.163**
book2	2.052	1.929	1.916	**1.899**	1.907
geo	4.370	4.366	4.394	4.384	**4.327**
news	2.649	2.450	2.426	**2.397**	2.412
obj1	4.164	3.898	3.925	**3.860**	3.941
obj2	2.842	2.589	2.555	**2.523**	2.530
paper1	2.624	2.412	2.406	**2.361**	2.369
paper2	2.473	2.300	2.293	**2.260**	**2.260**
pic	0.777	0.772	0.776	0.772	**0.766**
progc	2.746	2.485	2.471	**2.426**	2.448
progl	2.006	1.770	1.706	**1.688**	1.729
progp	2.057	1.832	1.795	**1.765**	1.780
trans	1.909	1.630	1.564	**1.540**	1.579

表 1: CTW の各 γ の値と比較した圧縮率

さらに，(1) 式の形より，一番短い文脈をはずした中で，最も短い文脈は γ がそこで 1 回かかって計算外となるためこの γ は，短い文脈を除外したものの確率の中で正規化された値を用いる．これを繰り返すことによって，全長さの文脈における γ を求めることができる．

これをまとめると以下のようになる．

アルゴリズム

1. 長さ $d(0 \leq d \leq D)$ 毎の文脈 s_d につきエントロピーの逆数または最も頻度が高い文字の確率 L_{s_d} を求める
2. L_{s_d} の総和 $S = \sum_{d=0}^{D} L_{s_d}$ を求める
3. $d = 0, 1, \cdots, D$ とし，以下のことを行う
   (a) $\gamma_{s_d} = \frac{L_{s_d}}{S}$
   (b) $S \leftarrow S - L_{s_d}$

4 実験結果

英文に対する実験を行った．実験にはワークステーション Sun Ultra60(メモリ 2048MB) を

使用した．英文のサンプルデータとしては，圧縮率の比較の際に論文で広くとりあげられている Calgary corpus[3] を利用した．

表 1 ～ 3 がその結果である．表中にある圧縮率を示す数字は単位 bpc で表され，これは，bit per character, 1 文字あたりの bit 数を意味している．この数を 8 で割ってやれば圧縮後のデータの割合を表す数字が求められる．また，表 1 の mix は，1 文字符号化させるごとに，γ の値を $0.1(0.8)^i$ $(i = 0, 1, 2, 3, 4)$ の 5 種類で変化させ，それぞれの確率の推定値 P_w^λ のうち，最もエントロピーが少なくなる γ の値を用いたものである．圧縮率については向上が期待できるが，当然処理時間もその分遅くなってしまう．また，文脈木の深さは $D = 64$ とした．

表 1 は，CTW 単独のものに本研究の手法をとりいれた結果である．CTW は [9] のもので，binary decomposition[11] という CTW に特有の，多値アルファベットへ対するテクニックを組み合わせたものを用いた．

γ を一定の値にした場合，γ の値によって圧縮率に偏りがうまれ，また，最適な γ の値もサンプルデータによって異なっている．エントロピーを γ の配分の尺度として用いた場合，全てというわけではないが，最適な γ の値によらず，安定して高い圧縮率を達成することができている．

また，mix に比べても圧縮率がいいサンプルがある．mix と本研究で述べている手法は，エントロピーを小さくするという目標において共通しているが，mix のような時間のかかる方法を用いなくても，ある程度の圧縮率は達成できるところが本手法の利点である，ということができよう．

表 2 は，PPM による確率予測を CTW に組み込んだ時の圧縮率の変化である．こちらでは最も頻度の高い文字の確率を γ の尺度として用いる手法の有用性が示されている．PPM で多値アルファベットを扱えるようになったことで，文字に依存する文脈を表現するのに必要なアルファベットの数が減り，無用に重みが散らばることがなくなったため，より正確な重みの配分ができたことになる，といえよう．

title\γ	0.5	0.3	0.2	0.1	最頻出
bib	1.8030	1.7943	1.7945	1.8015	**1.7888**
book1	2.2080	2.2097	2.2131	2.2203	**2.2053**
book2	1.8728	1.8717	1.8747	1.8832	**1.8660**
geo	4.4543	4.4555	4.4569	4.4594	**4.4523**
news	2.2952	2.2889	2.2895	2.2955	**2.2772**
obj1	3.6719	3.6678	3.6682	3.6734	**3.6626**
obj2	2.2674	2.2539	2.2510	2.2536	**2.2503**
paper1	2.2773	2.2722	2.2751	2.2857	**2.2674**
paper2	2.2355	2.2340	2.2378	2.2480	**2.2269**
pic	0.7688	0.7687	0.7690	0.7697	**0.7621**
progc	2.3040	2.2969	2.2988	2.3082	**2.2891**
progl	1.5744	1.5618	1.5600	1.5665	**1.5599**
progp	1.5824	1.5660	1.5616	1.5646	**1.5607**
trans	1.3695	1.3525	**1.3495**	1.3566	1.3574

表 2: PPM を導入した CTW の圧縮率

また，表 3 は，γ の配分を決める際に使う値をエントロピーにするのと最も頻度の高い文字の確率にするのかの比較をあらわした図である．この表を見る限り，Bloom[4] が，PPM に関しては最も頻度の高い文字の確率を基準にするのが良いという主張は，PPM を組み込んだ CTW に対しては当てはまっているが，CTW そのもに対しては当てはまっているとは言えない．短い文字数の文脈を処理するときには，PPM の技法が応用しやすいが，長い文脈を扱うときはうまくあてはまらなそうである．

また時間に関していえば，γ を調整するアルゴリズムにかかる時間は，他の確率 P などの計算に比べてわずかな時間しかかからず，時間の欠損が少ない状態で γ を予測している．

5 まとめ

本稿では，CTW において文脈中の最も高い頻度で出現する文字の確率をその文脈の推測の自信の度合として，その値を用いて CTW の重み付けのパラメタ γ の値を定める手法を提案した．この方法によって，γ の値を固定するよりも良い圧縮率を達成することを実験して確かめた．

ここで注目したいのは，この手法に対応す

title	CTW		PPM の CTW	
	Entropy	最頻出	Entropy	最頻出
bib	**1.884**	1.896	1.811	**1.7888**
book1	**2.163**	2.170	2.2128	**2.2053**
book2	**1.907**	1.915	1.8806	**1.8660**
geo	**4.327**	4.331	4.4525	**4.4523**
news	**2.412**	2.423	2.2919	**2.2772**
obj1	3.941	**3.929**	3.6715	**3.6626**
obj2	**2.530**	2.544	2.2731	**2.2503**
paper1	**2.369**	2.388	2.2875	**2.2674**
paper2	**2.260**	2.274	2.2407	**2.2269**
pic	**0.766**	**0.766**	0.7639	**0.7621**
progc	**2.448**	2.468	2.3129	**2.2891**
progl	**1.729**	1.744	1.5908	**1.5599**
progp	**1.780**	1.804	1.5837	**1.5607**
trans	**1.579**	1.601	1.3874	**1.3574**

表 3: 重み配分の基準による圧縮率の違い

る PPM の技法があることである．PPM も CTW も文脈を用いた圧縮方法であるという意味で共通しており，互いの改良技術が応用できる可能性を示しているとも言えよう．

この立場から，今後の予定として，PPM についての問題点の解決法を CTW に適用する手法などを考慮し，PPM と CTW を融合した圧縮法を構築することがあげられる．

また，尺度としてエントロピーを用いるのと最も頻度の高い文字の確率を用いるので起きる圧縮率の差についての理由について，深く検討する必要があろう．

謝辞

CTW 等，圧縮の手法に関して情報を提供してくださった Lund 大学の Åberg 氏，日頃からご助言をいただいている東北大の定兼邦彦氏に深く感謝いたします．

参考文献

[1] J. Åberg. *A Universal Source Coding Perspective on PPM.* PhD thesis, Lund University, October 1999.

[2] J. Åberg and Y. M. Shtarkov. Text compression by context tree weighting. In *IEEE Data Compression Conference*, pages 377–386, March 1997.

[3] The Calgary corpus. `http://corpus.canterbury.ac.nz/`.

[4] C. Bloom. Solving the problems of context modeling, March 1998. `http://www.cco.caltech.edu/~bloom/papers/ppmz.zip`.

[5] J. G. Cleary, W. J. Teahan, and I. H. Witten. Unbounded length contexts for PPM. In *IEEE Data Compression Conference*, pages 52–61, April 1995.

[6] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.

[7] 岡崎巧 定兼邦彦 今井浩．PPM 圧縮におけるエスケープ確率と算術符号の改良．信学技報，IT99-26, July 1999.

[8] K. Sadakane, T. Okazaki, and H. Imai. Implementing the context tree weighting method for text compression. In *IEEE Data Compression Conference*, March 2000.

[9] 定兼邦彦 岡崎巧 松本俊子 今井浩．文脈木重み付け法の条件付き確率による実装．In 第 *22* 回情報理論とその応用シンポジウム．SITA, December 1999.

[10] 坂口浩章 川端勉．有限窓を用いた文脈木重みづけ法．信学論 (A)，J80-A(12):2155–2163, 1997.

[11] T. J. Tjalkens, P. A. J. Volf, and F. M. J. Willems. A context-tree weighting method for text generating sources. In *IEEE Data Compression Conference*, page 472, March 1997.

[12] Tj. J. Tjalkens and F. M. J. Willems. Implementing the context-tree weighting method: Arithmetic coding. In *International Conference on Combinatorics, Information Theory & Statistics*, in Portland, Maine, July 1997.

[13] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Trans. Inform. Theory*, IT-41(3):653–664, May 1995.

[14] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.