The 27th Symposium on Information Theory
and Its Applications (SITA2004)
Gero, Gifu, Japan, Dec. 14-17, 2004

最頻文脈依存 N-gram を考慮した文法生成法に基づくデータ圧縮法
Data Compression Algorithm based on the Grammar Transform using the Most Frequent N-gram with Context Models

大井 昭久* 石田 崇* 平澤 茂一*
Akihisa OHI Takashi ISHIDA Shigeichi HIRASAWA

Abstract— Recently, the compression algorithms based on the grammar have been proposed. These algorithms first generate the grammar and then apply the arithmetic code. H.Nakamura et al. proposed an algorithm which registeres the most frequent digram as a rule. While, M.Kanda et al. have proposed an algorithm which makes the grammar the optimum by calculating the ideal code length of the arithmetic code. And, we proposed an algorithm that generates the grammar for the most frequent N-gram as well as the most frequent digram. While E.H.Yang et al. proposed a context-dependent algorithm which registeres sequentially the digram as a rule. In this paper, we proposed a new context-dependent algorithm that generates the grammar for the most frequent N-gram.

Keywords— Data compression, Grammar, N-gram, Arithmetic code, Context

1 はじめに

ユニバーサルデータ圧縮法の中で文法法が近年盛んに研究されている [1][2][3][4][5][6]. 文法法はデータ系列を生成する文法を作成 (グラマー化と呼ぶ) してからそれを符号化する手法である [1]. 代表的な文法法として Craig G.Nevill-Manning らによる Sequitur[2] や, 中村ら [3], 神田ら [4] による最頻 digram 統合法がある. ここで digram とは連続する 2 記号のことである. 最頻 digram 統合法 [3][4] では全ての系列を一度読み込んだ後, 最頻の digram をルールに変換している. ここで神田ら [4] は算術符号の理想符号長を計算しながら符号長の観点で最適な文法を生成する手法を提案している. さらに大井らは神田らの手法に対し改良を施し, 最頻の N-gram を考慮した文法を生成する手法 [5] を提案しており, 神田らの手法よりさらに良い圧縮率を達成している. ここで, N-gram とは連続する N 記号のことである.

一方, 近年文脈を考慮した前方一致 digram から文法生成を行う手法 [6] が E.H. Yang らにより提案されており文脈を考慮することの有効性を示している.

そこで本研究では最頻 N-gram 統合法 [5] をさらに改良し, 文脈を考慮した最頻 N-gram から文法を生成する手法を提案する. また実際どの程度の圧縮率を達成するかをベンチマークデータ (Calgary データ [7]) に対するシミュレーションにより評価を行う. その結果, 提案手法は最頻 N-gram 統合法 [5], 前方一致文脈依存 digram による文法生成法 [6] より良い圧縮率を達成できることを示す.

2 準備

本節では, 文法の定義 [8] について述べる.
文法は 4 項組 $G=(\Sigma_\alpha, \Sigma_\beta, R, S)$ で定義される.

1. Σ_α : 終端記号集合.
2. Σ_β : 非終端記号 (ルール記号) 集合.
3. R : 文の生成規則の有限集合 (ルール).
4. $S(\in \Sigma_\beta)$: 開始記号.

ただし $\Sigma_\alpha \cap \Sigma_\beta = \phi$ とする. 以降では終端記号 (Σ_α) を小文字のアルファベット $(a, b, c, \cdots)$, ルール記号 (Σ_β) を大文字のアルファベット $(A, B, C, \cdots)$ で表記する. このとき文の生成規則を以下のように定義する.

$$\mu \to \omega \qquad \mu \in \Sigma_\beta, \quad \omega \in (\Sigma_\alpha \cup \Sigma_\beta)^*$$

以下では文の生成規則をルールと呼び, 特定の文法 G におけるルールの集合をグラマーと呼ぶ. ここで, *は任意の正整数を表す. 文法に基づく符号では, Σ_α が情報源記号に相当し, データ系列 x は $x \in \Sigma_\alpha^*$ となる. また, $\Sigma = \Sigma_\alpha \cup \Sigma_\beta$ とする.

3 最頻 N-gram 統合法 [5]

本節では理想符号長を計算しながら N-gram に基づいてグラマー化を行う最頻 N-gram 統合法 [5] を示す.

3.1 最頻 N-gram 統合法 [5] の概要

最頻 N-gram 統合法 [5] では, データ系列を全て読み込み, 各 N-gram の出現頻度を求める. この際に適切な N の値を決定する必要があり, 以下の方法で決定を行う.

例えば N=3 の時は trigram, N=4 の時は tetragram となる. しかし, digram, trigram, tetragram …の頻度はそのままでは比較できないため相対的な頻度を考慮して, 実際に出現した N-gram のパターン数で基準化する. ここで基準化した値を, 相対最頻 N-gram 数と呼ぶ.

定義 1 (相対最頻 N-gram 数) 相対最頻 N-gram 数を以下の式で定義する.

$$\text{相対最頻 } N\text{-gram 数} = \text{最頻 } N\text{-gram 数} \times \frac{\text{実際に出現した } N\text{-gram のパターン数}}{\text{実際に出現した digram のパターン数}} \tag{1}$$

相対最頻 N-gram 数の最も大きな N の値に対して最頻の N-gram をルールとして登録し, その箇所をルール記号で置換する. その際に理想符号長を計算しておく. これを全ての digram の頻度が 1 になるまで繰り返す. そして理想符号長が最小のグラマーを採用し, ルールを連接した系列 (グラマー記号列と呼ぶ) を算術符号化する. ここでは, ルールの間に連接記号をはさむことにより連接を行う.

3.2 理想符号長の計算

理想符号長の計算法について述べる. グラマー記号列 g の経験確率 $P(g)$ を式 (2) のように計算する.

$$P(g) = \prod_{g_i \in \Sigma} \left(\frac{n(g_i)}{l_g} \right)^{n(g_i)} \tag{2}$$

ただし, l_g はグラマー記号列長, g_i はグラマー記号列内で出現する記号, $n(g_i)$ は記号 g_i のグラマー記号列内の出現頻度とする. ここで, $P(g)$ を計算する確率モデルを i.i.d. と仮定する. よって, g を算術符号化したときの理想符号長は,

$$-\log_2 P(g) = -\log_2 \prod_{g_i \in \Sigma} \left(\frac{n(g_i)}{l_g} \right)^{n(g_i)} \tag{3}$$

となる.

*〒 169-8555 東京都新宿区大久保 3-4-1 早稲田大学理工学部経営システム工学科 School of Science and Engineering, Waseda University Okubo 3-4-1, Shinjuku-ku, Tokyo, 169-8555 Japan. E-mail: ohi@hirasa.mgmt.waseda.ac.jp

3.3 アルゴリズム

① データ系列を全て読み込み，各 digram の出現頻度を求める．
② 各 digram の出現頻度が全て 1 ならば⑨へ．そうでなければ N=2 とし③へ．
③ M=N+1 とし，各 M-gram の出現頻度を求め，相対最頻 M-gram 数を式 (1) より計算する．
④ 相対最頻 N-gram 数 ≤ 相対最頻 M-gram 数かつ最頻 M-gram 数が 1 でなければ N=N+1 として③へ．さもなければ⑤へ．
⑤ 最頻 N-gram をルールとして登録し，グラマー中のその N-gram をルール記号で置換する．
⑥ 各記号の出現頻度を求め，理想符号長を式 (3) より計算する．
⑦ もし置換された N-gram 中にルール記号がただ 1 箇所にのみ存在するときはそのルールを削除する．またそのルール記号をそのルールに登録されていた記号列で置換する．
⑧ 置換したルール記号も情報源記号と同様に扱い，置換された系列に対してまた digram の出現頻度を求め② へ．
⑨ 理想符号長が最小のグラマーを採用し，そのグラマー記号列を算術符号化する． □

4 前方一致文脈依存 digram による文法生成法 [6]

本節では文脈を考慮した前方一致 digram から文法生成を行う手法 [6] を示す．

4.1 前方一致文脈依存 digram による文法生成法 [6] の概要

前方一致文脈依存 digram による文法生成法 [6] では，データ系列を順次読み込んでいく．ここで文脈とは直前の 1 終端記号のことを表し，文脈依存 digram とはある文脈の元で出現した digram のことを表す．そして読み込まれた文脈依存 digram が以前に現われた文脈依存 digram と一致した時にその文脈依存 digram をルールとして登録し，その箇所をルール記号で置換する．この様にしてルール生成を行いながらデータ系列を最後まで読み込む．ただし文脈がルール記号の場合はルールの内容を終端記号列に戻した場合の末尾の記号を文脈とする．そして，それぞれのルール記号に対して 1 つのみ両側に [ ] をつけて代入することによりグラマー記号列を生成する．

4.2 アルゴリズム

① 与えられた系列を S に一記号ずつ読み込む．その際にルールとの最長一致を検索し，一致すればまとめて読み込みそのルールに変換する．
② その記号と直前の 2 記号とで生成される文脈依存 digram がグラマー中に存在しているか検索する．2 記号前を文脈とする．存在しているなら③へ．そうでなければ①へ．
③ 一致した文脈依存 digram をルールとして登録し，グラマー中のその文脈依存 digram をルール記号で置換する．
④ もし置換された 文脈依存 digram 中にルール記号がただ 1 箇所にのみ存在するときはそのルールを削除する．またそのルール記号をそのルールに登録されていた記号列で置換する．
⑤ 置換したルール記号も情報源記号と同様に扱い，その直前の記号との文脈依存 digram として②へ． □

具体的なデータに対する前方一致文脈依存 digram による文法生成法 [6] の動作例を表 1 に示す．

表 1: 前方一致文脈依存 digram による文法生成法 [6] の動作例 (S→abcabc)

symbol	String	Grammar	備考
a	a	S→a	
b	ab	S→ab	
c	abc	S→abc	
a	abca	S→abca	
b	abcab	S→abcab	
c	abcabc	S→abcabc	bc|a が一致
		S→aAaA	RuleA|a 作成
		A|a→bc	

5 提案手法

5.1 提案への着想及び提案の概要

前方一致文脈依存 digram による文法生成法 [6] では文脈を考慮することにより同じルール記号を文脈の数だけ使用することができる．そのため同じ記号が出現する割合が高くなり算術符号の観点からすると圧縮効率が良くなることが期待できる．

そこで，提案方式では文脈を考慮して最頻 N-gram による文法生成を行うことを考える．最頻 N-gram 統合法と同様にデータ系列を全て読み込み，各文脈依存 N-gram の出現頻度を求める．この際にまた適切な N の値を決定する必要があり以下の方法で行う．

文脈依存 digram，文脈依存 trigram …の頻度はそのままでは比較できないため相対的な頻度を考慮して，実際に出現した文脈依存 N-gram のパターン数で基準化する．ここで基準化した値を，相対最頻文脈依存 N-gram 数と呼ぶ．

定義 2 (相対最頻文脈依存 N-gram 数) 相対最頻文脈依存 N-gram 数を以下の式で定義する．

$$\text{相対最頻文脈依存 } N\text{-gram 数} = \text{最頻文脈依存 } N\text{-gram 数} \times \frac{\text{実際に出現した文脈依存 } N\text{-gram のパターン数}}{\text{実際に出現した文脈依存 digram のパターン数}} \quad (4)$$

相対最頻文脈依存 digram 数 ≤ 相対最頻文脈依存 trigram 数かつ最頻文脈依存 trigram 数が 1 でなければ tetragram の頻度を算出し，相対最頻文脈依存 tetragram 数を求める．このように相対最頻文脈依存 N-gram 数を繰り返し比較し N の値を決定する．そして最頻文脈依存 N-gram をルールとして登録し，その箇所をルール記号で置換する．その際にグラマー記号列の理想符号長を計算しておく．

さらに，置換された系列に対してまた文脈依存 N-gram の出現頻度を求め，最頻文脈依存 N-gram をルール記号で置換する．これを全ての文脈依存 digram の頻度が 1 になるまで繰り返す．そして，理想符号長が最小のグラマーを採用し，グラマー記号列をそれぞれのルール記号に対して 1 つのみ両側に [ ] をつけて代入することにより生成した後，算術符号化する．

5.2 提案アルゴリズム

① データ系列を全て読み込み，各文脈依存 digram の出現頻度を求める．
② 各文脈依存 digram の出現頻度が全て 1 ならば⑨へ．そうでなければ N=2 とし③へ．
③ M=N+1 とし，各文脈依存 M-gram の出現頻度を求め，相対最頻文脈依存 M-gram 数を式 (4) より計算する．
④ 相対最頻文脈依存 N-gram 数 ≤ 相対最頻文脈依存 M-gram 数かつ最頻文脈依存 M-gram 数が 1 でなければ N=N+1 として③へ．さもなければ⑤へ．

表 2: 提案手法の動作例 (Σ_α={a,b,c},Σ_β={A,B})

grammar	digram	頻度	trigram	頻度	tetragram	頻度	備考
S→cabcabc	ab|c	4	abc|c	4	bcab|a	2	ab|c が最高頻度
cabcabc	bc|a	4	bca|a	2	cabc|b	2	相対最頻文脈依存 trigram 数=4 × (6/4)=6
	ca|b	3	cab|b	2	abca|c	2	相対最頻文脈依存 trigram 数 > 最頻文脈依存
	ca|c	1	bcc|a	1	bcca|a	1	digram 数より文脈依存 tetragram の頻度を算出
			cca|b	1	ccab|b	1	相対最頻文脈依存 tetragram 数=2 × (7/4)=3.5
			cab|c	1	cabc|c	1	相対最頻文脈依存 trigram 数 >
					abcc|c	1	相対最頻文脈依存 tetragram 数より N=3 に決定
S→cAAcAA	AA|c	2	AAc|c	1			Rule A 作成，AA|c が最高頻度，理想符号長=23.7
A|c→abc	Ac|c	1	AcA|c	1			文脈依存 trigram の頻度が全て 1 より
	cA|c	1	cAA|c	1			最頻文脈依存 digram を選択
S→cBcB	Bc|c	1					Rule B 作成
A|c→abc	cB|c	1					理想符号長=27.7
B|c→AA							文脈依存 digram の頻度が全て 1
							S→cAAcAA，A|c→abc を採用する．

⑤ 最頻文脈依存 N-gram をルールとして登録し，グラマー中のその文脈依存 N-gram をルール記号で置換する．

⑥ 各記号の出現頻度を求め，理想符号長を式 (3) より計算する．

⑦ もし置換された 文脈依存 N-gram 中にルール記号がただ 1 箇所にのみ存在するときはそのルールを削除する．またそのルール記号をそのルールに登録されていた記号列で置換する．

⑧ 置換したルール記号も情報源記号と同様に扱い，置換された系列に対してまた digram の出現頻度を求め② へ．

⑨ 理想符号長が最小のグラマーを採用し，そのグラマー記号列を算術符号化する．

具体的なデータに対する提案手法の動作例を表 2 に示す．

6 シミュレーション

本節ではシミュレーションにより本提案手法の有効性を示す．

6.1 シミュレーション条件

Calgary データ [7] を対象とし，最頻 N-gram 統合法 [5](手法 A)，前方一致文脈依存 digram による文法生成法 [6](手法 B)，提案手法を用いて各ファイルのグラマー化を行う．そしてそれぞれのグラマー記号列を算術符号を用いて符号化し，圧縮率を求める．手法 A，提案手法に関しては理想符号長が最小となるグラマー記号列を採用する．

6.2 シミュレーション結果

圧縮結果を表 3 に示す．また，提案手法が手法 A を上回ったデータの中で差の大きいデータ (bib，paper6)，下回ったデータ (geo) についてルールの内容を終端記号集合のみで表した場合の平均長，理想符号長が最小となるグラマー記号列長を調べた結果をそれぞれ表 4，表 5 に示す．ここで提案手法においては，A|a，A|b のように同じルール記号で異なる文脈からなるルールは同じルールとみなして計算を行う．

表 3: シミュレーション結果 [圧縮率]

データ	byte	手法 A[5]	手法 B[6]	提案手法
bib	117,543	0.2997	0.3039	0.2679
book2	268,843	0.3230	0.3323	0.3013
paper1	54,413	0.3756	0.3835	0.3501
paper2	83,932	0.3598	0.3702	0.3397
paper3	47,628	0.4013	0.4229	0.3850
paper4	15,582	0.4402	0.4731	0.4246
paper5	12,276	0.4555	0.4783	0.4380
paper6	39,126	0.3876	0.3933	0.3577
progc	41,100	0.3658	0.3812	0.3456
progl	71,908	0.2585	0.2628	0.2317
progp	51,347	0.2432	0.2536	0.2275
trans	90,726	0.2310	0.2373	0.2062
geo	102,400	0.5767	0.7132	0.6500
obj1	21,504	0.5219	0.5416	0.5182
obj2	246,814	0.3713	0.3896	0.3526
pic	513,216	0.1068	0.1214	0.1093

表 4: ルールを終端記号集合のみで表した場合の平均長

bib		paper6		geo	
手法 A	提案	手法 A	提案	手法 A	提案
8.104	19.17	7.528	15.99	3.782	12.67

表 5: 理想符号長が最小となるグラマー記号列長

bib		paper6		geo	
手法 A	提案	手法 A	提案	手法 A	提案
29023	34578	13324	16276	52595	75179

7 考察

本節では，シミュレーション結果及び時間計算量と空間計算量に対する考察を行う．

7.1 シミュレーション結果に対する考察

(1) 表 3 より提案手法では手法 A に対しては大部分のファイルにおいて圧縮率が改善された．手法 B に対しては全てのファイルに対して圧縮率が改善された．これは提案手法では，文脈を考慮することにより文脈の数だけ同じルール記号を使用できるようになり，同じ記号が出現する割合が高くなったので算術符号によって符号化した際に手法 A と比較して圧縮率が改善されたと考えられる．

(2) 表 4，表 5 より圧縮率が手法 A を上回った中で差の大きい bib，paper6 に関しては，長いルールが手法 A より非常に多く現れた．また理想符号長が最小となるグラマー記号列の長さも手法 A より長くなったが差が小さかったため，結果的に圧縮率が上回ったと考えられる．これより，良いグラマーとは個々のルールが長く，理想符号長が最小となるグラマー記号列の長さが短いグラマーであると考えられる．

(3) 表 4，表 5 より圧縮率が手法 A より下回った geo に関しては長いルールが手法 A より多く現れたが，理想符号長が最小となるグラマー記号列の長さが非常に長くなってしまったため結果的に圧縮率が下回ったと考えられる．geo では手法 A において N=2 を選択する場合が非常に多く digram によるルール変換が効率が良いと考えられる．しかし提案手法では文脈依存 digram の個数は trigram の数と同じであるため変換回数が少なくなってしまいグラマー記号

列長が長くなると考えられる．

7.2 時間計算量と空間計算量に対する考察

本項では提案手法が最頻 N-gram 統合法 [5] と同様，時間計算量及び空間計算量が $O(n)$ (n はデータ系列長) であることを示す．

定理 1 (提案手法の時間計算量及び空間計算量) 提案手法の時間計算量及び空間計算量は $O(n)$ である．

(証明)

提案手法では以下の 3 つの操作がある．

(1) 文脈依存 N-gram の頻度を調べる（この時間計算量を c_1 とする）．
(2) 文脈依存 N-gram をルール記号で置換する（この時間計算量を c_2 とする）．
(3) グラマー記号列の理想符号長 $-\log_2 P(g)$ を計算する（この時間計算量を c_3 とする）．

証明に用いる記号を以下に示す．

＜記号＞

N：2,3, $\cdots$, N_{max}

$g^{(t)}$： t 時点 (t 時点とは t 個目のルールを生成した時点を指す．) のグラマー記号列．

g_j： グラマー記号列内で j 番目に出現する記号．

$l_g^{(t)}$： t 時点のグラマー記号列の長さ．

$n^{(t)}(x)$： t 時点の記号 $x(\in \Sigma)$ のグラマー記号列中の頻度．

まず最初に，文脈依存 N-gram の頻度を調べるためにルール S の右辺を見て頻度情報を保持するリスト構造を作成する時間計算量は，文脈依存 N-gram の数は高々 $n-N$ であるので $O(n)$ である．これが最大 N_{max} 個必要なので，文脈依存 N-gram の頻度を調べる時間計算量は結局 $c_1 = O(n)$ である．

次に，ルール生成の際にリスト構造を更新するのに必要な時間計算量 c_2 を考える．文脈依存 N-gram をルール記号で 1 回置換するとその文脈依存 N-gram の前後の文脈依存 N-gram の頻度が減り，前後に新たに文脈依存 N-gram が生成される．結果的に 1 つのルール記号の置換に必要な時間計算量は $O(1)$ となる．ここで，ルール記号の置換は最大で $n/2$ 回であるので N-gram をルール記号に置換する時間計算量は，$c_2 = O(n)$ となる．

最後に，理想符号長を計算する時間計算量であるが，グラマー記号列 $g^{(t)}$ の理想符号長 $L(g^{(t)})$ の値は以下の式によって求められる．式 (3) より，

$$
\begin{aligned}
L(g^{(t)}) &= -\log_2 P(g^{(t)}) \\
&= -\log_2 \prod_{g_j \in \Sigma} \left(\frac{n^{(t)}(g_j)}{l_g^{(t)}} \right)^{n^{(t)}(g_j)} \\
&= -\sum_{g_j \in \Sigma} n^{(t)}(g_j) \log_2 \left(\frac{n^{(t)}(g_j)}{l_g^{(t)}} \right) \\
&= \sum_{g_j \in \Sigma} n^{(t)}(g_j) \log_2 l_g^{(t)} - \sum_{g_j \in \Sigma} n^{(t)}(g_j) \log_2 n^{(t)}(g_j) \\
&= l_g^{(t)} \log_2 l_g^{(t)} - \sum_{g_j \in \Sigma} n^{(t)}(g_j) \log_2 n^{(t)}(g_j)
\end{aligned}
$$

となり，グラマー記号列長，ルールで置換する記号の出現頻度のみが更新に関与していることが導かれる．ここで，$t-1$ 時点から t 時点に更新される際に文脈依存 N-gram に置換する記号の頻度，グラマー記号列長が減少し，生成されたルール記号の頻度，[ ] の頻度が増加する．従って 1 時点前からの加減計算のみで計算可能なことより，1 回ルール記号で置換する際に理想符号長を更新する時間計算量は $O(1)$ である．これは最大で $n/2$ 回行われるので，理想符号長の計算に必要な時間計算量 c_3 は高々 $O(n)$ となる．従って，提案手法に必要な総時間計算量は $c_1 + c_2 + c_3$ より $O(n)$ である．

また空間計算量についての評価だが，頻度情報を保持するリスト構造を生成する際に，登録する N-gram の数は高々 $n-N$ 個であるので必要な空間計算量は $O(n)$ である．また，N-gram をルール記号に 1 回置換するのに必要な空間計算量とグラマー記号列の理想符号長を 1 回計算するのに必要な空間計算量は $O(1)$ である．最大で $n/2$ 回行われるので必要な空間計算量は高々 $O(n)$ である．

以上より，提案手法の時間計算量及び空間計算量は $O(n)$ であることが示された．

8 まとめと今後の課題

今回の研究では，文脈を考慮した最頻 N-gram によりグラマー化を行う手法を提案し，従来の 2 手法 [5][6] と比較して圧縮率において有効性を示すことができた．また，時間計算量と空間計算量においても最頻 N-gram 統合法 [5] と同じく $O(n)$ であることを示した．

今回は文脈依存 N-gram を考慮に入れグラマー化を行う際に相対的な頻度を考慮してルール選択を行っていったが，他に圧縮率をより向上させるような基準化方法を検討する必要がある．また，最頻 N-gram 統合法 [5] において digram を多く選択するデータに対して，提案手法では圧縮率が下回っているので最頻 digram によるルール変換を後処理として用いるなどを行い圧縮率を改善する必要がある．そして，最頻 digram 統合法 [3][4]，最頻 N-gram 統合法 [5] ではユニバーサル性を持つ事が証明されていないが，本手法がユニバーサル性を持つかについての検討は今後の課題である．

9 謝辞

著者の一人である大井は，本研究を行うにあたり，数多くのご助言，ご支援を賜りました早稲田大学平澤研究室の各氏に感謝いたします．

参考文献

[1] J.C.Kieffer, E.Yang: "Grammar Based Codes: A New Class of Universal Lossless Source Code," *IEEE Trans. Inform. Theory*, Vol.46, No.3, pp.737-754, (2000).

[2] Craig G.Nevill-Manning, Ian H.Witten: "Identifying Hierarchical Structure In Sequences a Linear-Time Algorithm," *Journal of Artificial Intelligence Research* 7, pp.67-82, (1997).

[3] 中村博文, 村島定行: "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, Vol.J79-A , No.7 , pp.1319-1323, (1996).

[4] 神田勝，石田崇，小林学，平澤茂一，"最頻 Digram 統合に基づくデータ圧縮法," 電子情報通信学会信学技法, Vol.IT-102,No.198,pp.31-36,(2002).

[5] 大井昭久，石田崇，平澤茂一，"最頻 N-gram を考慮した文法生成法に基づくデータ圧縮法," 電子情報通信学会信学技法，Vol.IT-103,No.214,pp.13-18,(2003).

[6] E.-H. Yang and Da-ke He, "Efficient universal lossless compression algorithms based on a greedy sequential grammar transform-Part two: With context models," *IEEE Trans. Inform. Theory*, Vol.49, No.11, pp.2874-2894, (2003).

[7] ftp://ftp.cpsc.ucalgary.ca/pub/projects /text.compression.corpus/

[8] 北 研二: 確率的言語モデル, 東京大学出版会, (1999).