A-011

日本語テキストに対する検索指向符号化のための文法圧縮分割
Grammar Compression Parsing for Search Oriented Codes on Japanese Texts

正木 拓也 * **笹川 裕人** * **喜田 拓也** *
Takuya Masaki Hirohito Sasakawa Takuya Kida

1. はじめに

テキスト圧縮においては，圧縮率の良さだけでなく，圧縮テキスト上で文字列検索 (**圧縮検索**) が可能かどうかも重要な点である．選択する圧縮法によっては，圧縮前のテキスト上で文字列検索を行うよりも，圧縮テキスト上で検索を行う方が高速になる場合もある [4]．

テキスト中の単語毎に符号語を割り当てる**単語ベースの文字列圧縮法**は，圧縮検索が可能な検索向きのデータ圧縮であり，また優れた圧縮率を達成できることが知られている [3]．特に，バイト単位の可変長符号を用いる **End-Tagged Dense 符号** (**ETDC**)[1] などを用いると，符号語の切れ目が容易に判断できるため，圧縮データの一部分のみを直接に取り出せるといった利点もある．

英文のようにスペースなどで単語毎に分かち書きされている場合，こうした符号化で効率良く圧縮できる．しかしながら，日本語テキストの場合，単語毎の明確な区切りがないため，単語ベースの文字列圧縮を直接に適用することは難しい．

本稿では，分かち書きされていない日本語テキストに対して，ETDC で符号化する手法を提案する．提案手法では，日本語テキストに対し，**文法圧縮**の一つである **Re-Pair アルゴリズム** [2] を利用して分かち書きを行い，その結果得られるテキストを ETDC で符号化する．また，分かち書きに利用する Re-Pair アルゴリズムにおいて，文字の置換えによる圧縮率の変化を評価する指標を導入し，その指標に基づいて再帰処理を打ち切ることで，bzip2 に匹敵する圧縮率が得られることを示す．

2. 準備

2.1 文法圧縮

文字列データからそれを一意に導出する形式文法を構築し，その文法を符号化することでデータ圧縮を実現する手法を**文法圧縮**という．形式文法としては**文脈自由文法**が用いられることが多い．文脈自由文法は $G = (T, N, \sigma, P)$ の 4 つの組で定義される．ここで，T は**終端アルファベット**，N は**非終端アルファベット**，σ は**開始記号**，P は**生成規則** $\alpha \to \beta$ の有限集合であり，$T \cup N = \emptyset$，$\sigma \in N$，$\alpha \in N$ かつ $\beta \in (N \cup T)^*$ が成立する．

文法圧縮の 1 つとして **Re-Pair アルゴリズム** [2] がある．テキストの任意の 1 文字を **uni-gram**，連続する 2 文字を **bi-gram** と呼ぶとすると，Re-Pair アルゴリズムは，入力テキスト中の最頻の bi-gram を新しい非終端文字に置き換えて uni-gram とし，その置換え規則を生成規則として辞書に登録する．その後，置き換えたテキストを次の入力として，この置換操作を全ての bi-gram が unique になるまで再帰的に繰り返すことで文法を構築する．Re-Pair アルゴリズムの処理の流れを図 1 に示す．圧縮テキストを展開するには，各非終端文字を辞書

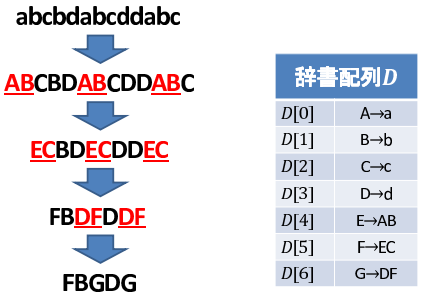


図 1: Re-Pair アルゴリズムの流れ

表 1: End-Tagged Dense 符号化の例

単語	生起確率	符号語
A	0.34	100
B	0.30	101
C	0.12	110
D	0.10	111
E	0.08	000100
F	0.06	000101

配列 D の生成規則にしたがって非終端記号がなくなるまで導出すれば良い．

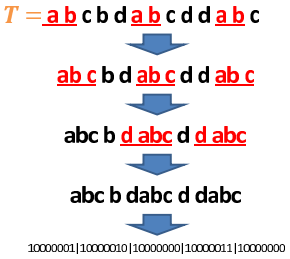
2.2 End-Tagged Dense 符号

テキスト中の各単語に対し，生起確率の降順に短い符号語を割り当てる符号化を **Dense 符号**という．End-Tagged Dense 符号 [1] はバイト単位の符号語を用いる Dense 符号の一種で，符号語の最後尾バイトの先頭ビットを 1 に，他のバイトの先頭ビットは 0 に固定することで符号の切れ目がわかるようになっている．よって，各バイトで扱える数値は 7 ビットの 128 種類となる．生起確率の大きい順に”10000000”から 1 ずつ加算して符号語を割り当てていく．先頭ビットは固定されているので，8 桁目に繰り上がる時は 8 桁目を飛ばして一つ前のバイトに繰り上げて 1 を加算する．説明簡略化のため，バイト単位ではなく 3 ビット単位としたときの End-Tagged Dense 符号化の例を表 1 に示す．

3. 提案手法

本節では，ETDC を日本語テキストに適用させるための分かち書き法を提案する．提案手法では，入力テキスト T に対して，1 バイトではなく，**マルチバイト文字**の日本語の各 1 文字を 1 単語とみなす．次に，各単語を uni-gram とみなして Re-Pair アルゴリズムで最頻の bi-gram を計算し，その bi-gram を結合して新しい 1 単語とする．この操作を再帰的に b 回繰り返し，各単語を ETDC で符号化する．図 2 に，図 1 の文字列に対する提案手法による圧縮の例を示す．ただし，再帰回数 $b = 3$

*北海道大学， Hokkaido University



$b = 3$

*ETDC*辞書

単語	頻度	符号語
dabc	2	10000000
abc	1	10000001
b	1	10000010
d	1	10000011

図 2: $b = 3$ のときの提案手法の流れ

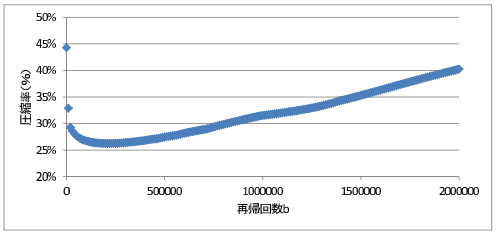


図 3: b を 10000 ずつ増やした時の圧縮率の変化

とする．

次に，再帰回数 b の最適な値について考察する．$i-1$ ステップ目の再帰処理で bi-gram の結合を行った後に ETDC で符号化したときと，i ステップ目の再帰処理後に ETDC で符号化したときの圧縮後のデータサイズを比較すると，データの増加分は $Inc_i = |w_i| + f_i|\mathcal{E}(w_i)|$，減少分は $Dec_i = f_i(|\mathcal{E}(L(w_i))| + |\mathcal{E}(R(w_i))|)$ で与えられる．ただし，w_i，f_i はそれぞれ i ステップ目で選択される最頻の bi-gram を結合した単語と頻度，$\mathcal{E}(w_i)$ は ETDC で符号化したときの w_i の符号語，$L(w_i)$，$R(w_i)$ はそれぞれ w_i の結合する前の左側の uni-gram と右側の uni-gram である．もし $\Delta_i = Inc_i - Dec_i$ が i について単調増加ならば，初めて正の値になった時点の i について，$b = i - 1$ で再帰を終了することで最適に近い圧縮率が得られる．しかし，この値は選択する bi-gram の頻度だけでなく bi-gram のバイト長と ETDC のバイト長，bi-gram の左側と右側の uni-gram の ETDC のバイト長も影響するため，単調にはならない．予備実験による，b に対する圧縮率の変化を図 3 と図 4 に示す．

そこで，Δ_i のおよその期待値として，$\Delta'_i = 2l_{i-1} + f_i|\mathcal{E}(w_i)| - 2f_i\overline{\mathcal{E}_{i-1}}$ を定義する．ただし，$l_i = (|T|/n_i)$ であり，l_i と n_i はそれぞれ i ステップ目の再帰処理後の平均ブロック長とブロック数である．また，i ステップ目の再帰処理後の ETDC の平均符号語長 $\overline{\mathcal{E}_i}$ を，$\overline{\mathcal{E}_i} = ((l_{i-1} - 2f_i)\overline{\mathcal{E}_{i-1}} + f_i|\mathcal{E}(w_i)|)/l_i$ として見積もることにする．ここで，単語 w_i の符号語長 $|\mathcal{E}(w_i)|$ は，ETDC の特性より単語の種類数から簡単に求められることに注意する．初めて $\Delta'_i > 0$ となった i について，$b = i - 1$ とすることで b の最適値を推定する．

4. 実験

日本語テキストデータを用いて実験を行った．データには UTF-8 の文字コードで記述された毎日新聞の記事を用いた．データサイズは約 169MB．データに対して提案手法による圧縮率と Re-Pair，gzip，bzip2 による圧縮率を比較した．圧縮率の比較結果を表 2 に示す．

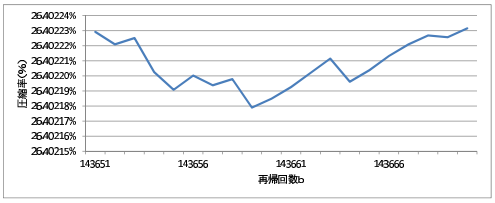


図 4: b を 1 ずつ増やした時の圧縮率の変化

表 2: 日本語テキストに対する圧縮率 (%) の比較

提案手法	Re-Pair	gzip	bzip2
26.50	23.01	37.92	26.10

5. 結論

提案手法が，ETDC を通して，bzip2 と同等程度の圧縮率を達成する分かち書きを与えることを示した．元の Re-Pair に比べて若干圧縮率が劣る一因としては，ETDC と Re-Pair の辞書構造の差が考えられる．すなわち，ETDC では単語と符号語を一対一に対応させた辞書を保持しているが，Re-Pair では辞書に相当する部分もコンパクトな文法によって表現されており，なおかつビット単位での可変長符号化が行われている．ただし，こうした圧縮率向上の工夫は，圧縮後のデータの取り扱いを複雑にしてしまう．一方で，ETDC は，圧縮テキストから符号語の切れ目が簡単に判断できる点と，辞書から符号語の検索が容易に行える点で優れている．

また，今回の実験では，再帰回数 b について，最適な b と提案手法で求めた b では圧縮率の差は 0.1%程度であった．様々なデータに対して提案手法を適用し，その挙動を調査することは今後の課題の一つである．

参考文献

[1] N. Brisaboa, E. Iglesias, G. Navarro, and J. Paramá. An efficient compression code for text databases. In *Proc. 25th European Conference on Information Retrieval Research (ECIR)*, LNCS 2633, pp. 468–481, 2003.

[2] N.J. Larsson and A. Moffat. Off-line dictionary-based compression. Vol. 88, pp. 1722–1732. IEEE, 2000.

[3] A. Moffat. Word-based text compression. *Software: Practice and Experience*, Vol. 19, No. 2, pp. 185–198, 1989.

[4] M. Takeda, Y. Shibata, T. Matsumoto, T. Kida, A. Shinohara, S. Fukamachi, T. Shinohara, and S. Arikawa. Speeding up string pattern matching by text compression: The dawn of a new era. *IPSJ Journal*, Vol. 42, No. 3, pp. 370–384, mar 2001.