Vol.2014-AL-149 No.5
2014/9/12

Dense符号化のための文法圧縮分割

正木 拓也[1]　笹川 裕人[1]　喜田 拓也[1]

概要：入力テキストを単語毎に符号化する End-Tagged Dense 符号 (ETDC) は，バイト単位の可変長符号を用いる，符号語の抽出が容易な検索向きのデータ圧縮法である．本稿では，単語毎に分かち書きされていないテキストに対して ETDC で符号化する手法を提案した．提案手法は，テキストに対して文法圧縮の一つである Re-Pair アルゴリズムを利用した分かち書きを行い，その後に ETDC で符号化を行う．その際，Re-Pair アルゴリズムの再帰処理において，後段の ETDC の符号化の効率を見積もる評価指標を導入し，その指標に基づいて再帰を打ち切る．それにより，日本語テキストやゲノムデータなどに対しても，検索や圧縮率の観点から効率よいデータ圧縮が実現できる．今回，実データに対して提案手法を適用することで，gzip や bzip2 に匹敵する圧縮率を達成できることを実証した．

Grammar Compression Parsing for Dense Coding

Abstract: End-Tagged Dense Code (ETDC) is a word-based compression method that uses byte-oriented codewords. It is easy to extract codewords from compressed texts, and thus suitable for searching keywords on them. In this paper, we proposed a method for ETDC on texts which are not separated by spaces. In the proposed method, an input text is parsed by utilizing Re-Pair algorithm proposed by Larsson and Moffat, and then encoded by ETDC. We introduced a criterion that estimates the efficiency of ETDC in order to truncate the recursion of Re-Pair according to the criterion. This truncation realizes efficient data compression from the viewpoint of searching and compression ratio even for Japanese texts and gene data. In this time, we showed that our method achieved good compression ratios comparable with gzip and bzip2.

1. 序論

テキストの圧縮においては，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 [1]．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 [8, 9]．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 [5, 6]．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が，日本語テキストなど，分かち書きされていないテキストの場合，単語毎の明確な区切りがないため，直接の適用は難しい．

Yoshida ら [10] は，日本語テキストに対して**形態素解析**による分かち書きを行い，各単語に **Canonical Huffman 符号** (**CHC**) [7] を割り当てる単語ベースの圧縮法を提案している．ただし，彼らの手法では圧縮率を向上させるために，可変長ビットである CHC で符号化を行っているため，パターン照合などの処理を行う際にビットの切り出し操作が必要となる．それに加えて，符号語の切り出しは圧縮テキストを先頭から走査する必要があるため，任意の位置へのデータアクセスが困難である．また，形態素解析は辞書に依存する面が大きく，対象のテキストに合う辞書をその都度用意しなければならない．

本稿では，形態素解析を行わずに，分かち書きされていないテキストに対して，バイト単位の可変長符号である **End-Tagged Dense 符号** (**ETDC**) [2] で符号化する手法を提案する．ETDC による単語ベースの圧縮は，符号語と元テキストの単語が一対一に対応しており，かつ各バイトの先頭ビットを参照することで，符号語の切れ目も簡単に判断できるため，検索向きのデータ圧縮である．

本稿で提案する手法は，分かち書きされていないテキストに対して，**Re-Pair アルゴリズム** (以下 **Re-Pair**) [4] を

[1] 北海道大学大学院情報科学研究科
Hokkaido University, Graduate School of Information Science and Technology, {tmasaki, sasakawa, kida}@ist.hokudai.ac.jp

利用した分かち書きを行い，各単語を ETDC で符号化する．Re-Pair とは，元テキストを一意に導出する形式文法を構築することで圧縮する**文法圧縮**の一つであり，テキスト中の連続する 2 文字の中で最も頻度の大きいものを，新しい文字に置き換えていく処理を再帰的に繰り返すことで**文脈自由文法**を構築する．このとき，Re-Pair の再帰をいつ打ち切るかで単語の分かち書き方が変化し，ETDC で符号化したときの圧縮率も高くなったり低くなったりを不規則に変動する．そこで，分かち書きに利用する Re-Pair アルゴリズムにおいて，再帰処理を行う前後の圧縮データサイズに基づいた評価指標を導入し，その指標に従って再帰を打ち切る．実験では，日本語テキスト，dna データ，protein データの 3 つに対して提案手法で圧縮を行い，提案手法と既存の圧縮法との圧縮率の比較を行った．

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 [3] という．形式文法としては文脈自由文法 (CFG) が用いられることが多い．CFG は $G = (\Sigma, V, \sigma, R)$ の 4 つの組で定義される．ここで，Σ は**終端アルファベット**，V は**非終端アルファベット**，σ は**開始記号**，R は**生成規則** $\alpha \to \beta$ の有限集合であり，$\Sigma \cup V = \emptyset$，$\sigma \in V$，$\alpha \in V$ かつ $\beta \in (\Sigma \cup V)^*$ が成立する．

文法圧縮の 1 つとして Re-Pair がある．本稿では，任意の 1 文字を**ユニグラム**，連続する 2 文字を**バイグラム**と呼ぶ．

2.2 Re-Pair アルゴリズム

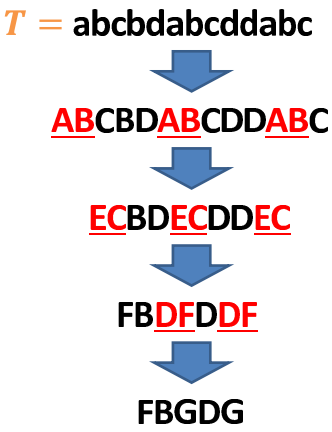
形式的には，入力テキスト T に対して Re-Pair は 4 つ組 (Σ, V, σ, R) で表される CFG G を一つ生成する．ここで，$\Sigma = \{a_0, a_1, \cdots, a_{|\Sigma|-1}\}$ は終端記号の集合であり，$V = \{\alpha_0, \alpha_1, \cdots, \alpha_{|V|-1}\}$ は非終端記号の集合．また，$\sigma \in V$ は開始記号，R は $V \times (\Sigma \cup V)^*$ 上の有限部分集合である．R の要素は生成規則と呼ばれ，$\alpha \in V, \gamma \in (\Sigma \cup V)^*$ に対し $\alpha \to \gamma$ のように記述される．

Re-Pair によって生成された CFG G は次のような生成規則を含む．

$$\sigma \to \alpha_{i_0}\alpha_{i_1}\cdots\alpha_{i_{m-1}} \quad (\forall i_k \in \{0, \cdots, |\Sigma| + |V| - 2\}),$$

$$\alpha_i \to \begin{cases} a_i & (0 \le i < |\Sigma| \text{ の場合}), \\ \alpha_j\alpha_k \quad (0 \le j, k < i) & (i \ge |\Sigma| \text{ の場合}). \end{cases}$$

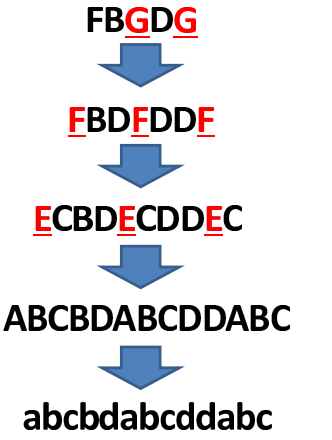
ここで，生成規則の右辺に出現するバイグラムはすべて異なり，$\sigma \to T$ である．Re-Pair は入力テキスト T 中に出現するバイグラムの中から最頻出のもの $\alpha\beta$ を一つ選び，その最頻のバイグラムの出現を左から順番に新しい記号 A に置き換える．そして，生成規則 $A \to \alpha\beta$ を R に追加する．この手続きを，テキスト上のすべてのバイグラムの頻度が 1 になるまで繰り返す．上記の置換え手続き後のテキスト T' を開始記号 σ から生成する生成規則 $\sigma \to T'$ を R に加えることで，元のテキスト T を唯一に導出する CFG G を得る．最終的には，G に適当な符号化を行い圧縮データを得る．Re-Pair の圧縮の流れを図 1 に示す．圧縮テキスト



辞書配列D	
D[0]	A→a
D[1]	B→b
D[2]	C→c
D[3]	D→d
D[4]	E→AB
D[5]	F→EC
D[6]	G→DF

図 1 Re-Pair の圧縮の流れ

を解凍するには，各非終端文字を辞書配列 D にある生成規則によって終端文字へと到達するまで展開すればよい．Re-Pair の解凍の流れを図 2 に示す．



辞書配列D	
D[0]	A→a
D[1]	B→b
D[2]	C→c
D[3]	D→d
D[4]	E→AB
D[5]	F→EC
D[6]	G→DF

図 2 Re-Pair の解凍の流れ

Larsson と Moffat ら [4] によって，Re-Pair は入力テキスト長 n に対し，$O(n)$ 時間で CFG の構築を完了することが示されている．その計算時間を達成するためには，入力テキストを，同一のバイグラムどうしを前後でリンクさせた双方向連結リストに変換する必要がある．さらに，任意のバイグラムを保持する記述子へ $O(1)$ 時間でアクセスするためのハッシュテーブルとバイグラムの頻度を管理するための優先度付きキューを必要とする．

2.3 End-Tagged Dense 符号 (ETDC)

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．ETDC [2] 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．以下に，ETDC の符号化の手順を示す．

(1) テキストデータ中の各単語を，生起確率の降順に並べ，生起確率の最も大きい単語には「10000000」の符号語を割り当てる．

(2) 残りの各単語について，生起確率が降順に以下のように符号語を割り当てる．前の単語の符号語の最後尾バイトに 1 を加算したものを符号とする．ただし，最後尾バイトが「11111111」を越える場合は繰り上げ，最後尾バイトは「10000000」に書き換えて，その前のバイトに 1 を加算する．また，最後尾以外のバイトが「01111111」を越える場合も同様に繰り上げ，そのバイトは「00000000」に書き換え，その前のバイトに 1 を加算する．繰り上げる際に前にバイトが無い場合は新しく先頭に「00000000」を付け加える．

(3) すべての単語に符号語が割り当たるまで (2) を生起確率の降順に繰り返す．

以上の手順により，ETDC が割り当てられる．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 ETDC の符号化の例を表 1 に示す．

表 1 ETDC の符号化の例 (3 ビット単位)

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	0.05	000101
G	0.01	000110

また，生起確率の順位が t である単語を ETDC で符号化したときのバイト長は，$\sum_{i=0}^{k-1} 128^i \leq t < \sum_{i=0}^{k} 128^i$ を満たす k として簡単に求められる．

3. 提案手法

本節では，ETDC を分かち書きされていないテキストに適用させる手法を提案する．提案手法では，各単語をユニグラムとみなして Re-Pair で最頻のバイグラムを計算し，そのバイグラムを結合して新しい 1 単語とする．この操作を再帰的に b 回繰り返し，最後に各単語を ETDC で符号化する．図 3 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 $b = 3$ とする．

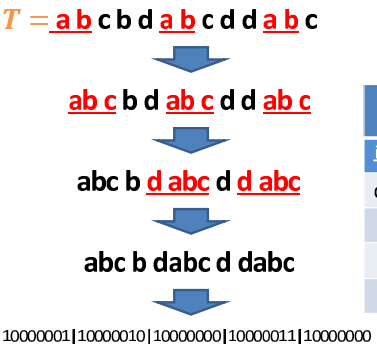


図 3 提案手法

次に，再帰回数 b の最適な値について考察する．予備実験として，実際に b 値を 10000 ずつ増やしたときの圧縮率の変化を図 4 に，図 4 の極小値付近から b 値を 1 ずつ増やしたときの圧縮率の変化を図 5 に示す． 図からわかるように，b の値に対して極小値まで圧縮率が単調減少しているように見えるが，実際には細かく上下している．従って，単純な方法では極小値のときの b の値を求めることは難しそうである．

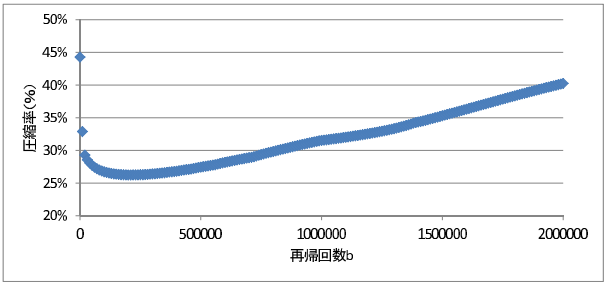


図 4 b を 10000 ずつ増やした時の圧縮率の変化

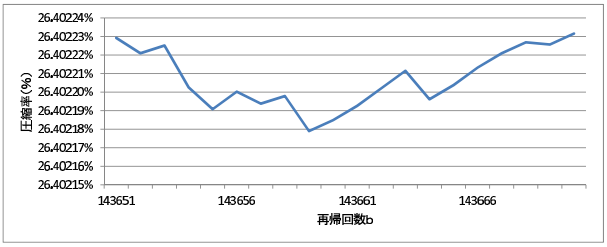


図 5 b を 1 ずつ増やした時の圧縮率の変化

$i-1$ ステップ目の再帰処理でバイグラム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の差を考えてみる．ここで，i ステップ目で選択されるバイグラムを $\alpha\beta$ とする．バイグラム $\alpha\beta$ の結合を行うと，α

と β の頻度が $\alpha\beta$ の頻度分減少し，新しい単語として $\alpha\beta$ が生まれる．このとき，ETDC の辞書には新たな単語 $\alpha\beta$ が追加され，さらに場合によっては α と β の片方もしくは両方が，$\alpha\beta$ にすべて使用されたことで ETDC の辞書から消滅する可能性もある．したがって，この 3 つの単語の頻度が変わることで，特定の範囲の単語の頻度の順位がすべて変動するため，正確にデータサイズの変動を見積もるには，すべての単語の頻度を保持した上で，再帰の都度順序を入れ替える必要がある．Re-Pair はバイグラムの頻度を優先度付きキューなどで管理しているが，各単語の頻度を管理するには余分なメモリと計算時間が必要である．

ここで，再帰処理で結合するバイグラム $\alpha\beta$ の頻度は，そのときのすべての単語中で最下位の頻度となり，$\alpha\beta$ の結合によって，α と β どちらの頻度の順位にも変動は起きないものと仮定する．この仮定において，i ステップ目のバイグラム $\alpha\beta$ の結合の際，ETDC の辞書に新たに追加することになる $\alpha\beta$ の長さと，すべての $\alpha\beta$ を ETDC で符号化したときの符号語の長さの総計との和を Inc_i，結合によって消滅したすべての ETDC で符号化された α と β の符号語の長さの総計を Dec_i とする．i ステップ目でバイグラムを置き換えたことによるデータの増加量を Δ_i とおくと，Inc_i，Dec_i，Δ_i は以下のように求まる．

$$\begin{cases} Inc_i &= |w_i| + |\mathcal{E}(w_i)|f_i \\ Dec_i &= (|\mathcal{E}(L(w_i))| + |\mathcal{E}(R(w_i))|)f_i \\ \Delta_i &= Inc_i - Dec_i \end{cases}$$

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バイグラム $\alpha\beta$ 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ユニグラム α と右側のユニグラム β である．すなわち，$|w_i|$ が ETDC の辞書に登録する $\alpha\beta$ の長さ，$|\mathcal{E}(w_i)|f_i$ がすべての $\alpha\beta$ を ETDC で符号化したときの符号語の長さの総計，$(|\mathcal{E}(L(w_i))| + |\mathcal{E}(R(w_i))|)f_i$ が $\alpha\beta$ の結合によって消えた ETDC で符号化された α と β の符号語の長さの総計である．

もし Δ_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，$b = i - 1$ とすることで，この仮定での最適な圧縮率が得られる．しかし，この値は選択するバイグラムの頻度だけでなく，バイグラムのバイト長と ETDC のバイト長，バイグラムの左側と右側のユニグラムを ETDC で符号化したときのバイト長も影響するため，単調にはならない．

ここで，i ステップ目で選択されるバイグラムが，すべての単語の平均の長さのユニグラムを 2 つ合わせたものであるとする．この条件下での Δ_i を Δ'_i とすると，Δ'_i は以下のように求まる．

$$\begin{cases} \Delta'_i &= 2l_{i-1} + |\mathcal{E}(w_i)|f_i - |\mathcal{E}_{i-1}|2f_i \\ |\mathcal{E}_i| &= (|\mathcal{E}_{i-1}|b_{i-1} - (|\mathcal{E}(L(w_i))| + |\mathcal{E}(R(w_i))|)f_i \\ &\quad + |\mathcal{E}(w_i)|f_i)/b_i \end{cases}$$

ただし，$l_i = (|T|/b_i)$ で，l_i と b_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，$|\mathcal{E}_i|$ は i ステップ目の再帰処理後に各ユニグラムに割り当てられている ETDC の符号語長の平均である．また，$|\mathcal{E}(w_i)|$ は ETDC の特性より，単語の種類数から簡単に求められる．

よって，$\Delta'_i > 0$ ならば，その時点でバイグラムを置き換えたとき，データサイズの増加量の期待値が正，$\Delta'_i < 0$ ならば，その時点でバイグラムを置き換えたとき，データサイズの増加量の期待値が負であると判別できる．この Δ'_i を用いて，初めて $\Delta'_i > 0$ となった i について $b = i - 1$ とすることで最適な b に近い値を推定する．ただし，単語の種類が 128 種類であるとき，$\Delta'_i = 2l_{i-1} > 0$ となり，このときだけは確実に圧縮率が悪化してしまうが，$\Delta'_i = 2l_{i-1} < 0$ とみなして再帰処理を続けることにする．

4. 実験

提案手法の有用性を評価するため，データセットを用いて計算機実験を行った．まず，実験で使用するデータについて述べ，実験方法について説明する．その後，実験の結果と考察について述べる．

4.1 データ

データセットとして以下の 4 つのデータを使用した．dna と protein は Pizza&Chili Corpus(http://pizzachili.dcc.uchile.cl) のサイトより入手した．

- mai2000
  2000 年の毎日新聞記事の日本語テキストからホワイトスペースを除去したデータ．文字コードは UTF-8 で，サイズは約 169MB.
- mai2001
  2001 年の毎日新聞記事の日本語テキストからホワイトスペースを除去したデータ．文字コードは UTF-8 で，サイズは約 88MB.
- dna
  dna の塩基配列を並べたデータ．サイズは約 105MB.
- protein
  タンパク質の組成配列を並べたデータ．サイズは約 105MB.

4.2 実験方法

各データセットに対して，提案手法を用いて圧縮を行った．ただし，mai2000 と mai2001 に対しては，元テキストの 1 バイトではなく，日本語 1 文字を 1 単語とみなして提

案手法を用いた．その後，提案手法の圧縮率と，既存手法である Re-Pair，gzip，bzip2 による圧縮率を比較した．

4.3 考察

圧縮率の比較結果を表 2 に示す．

表 2 圧縮率 (%) の比較

	提案手法	Re-Pair	gzip	bzip2
mai2000	27.68	24.13	37.91	26.24
mai2001	26.39	23.01	37.92	26.10
dna	29.37	31.81	28.25	26.00
protein	58.62	52.72	49.34	46.99

提案手法は，日本語テキストに対しては，gzip に優り，bzip2 と同等程度で，Re-Pair に劣る圧縮率を達成した．dna に対しては，Re-Pair に優り，gzip と同等程度で，bzip2 に劣る圧縮率を達成した．protein に対しては，3 つの既存手法全てに劣る圧縮率であった．

基本的に Re-Pair よりも圧縮率が劣るのは ETDC と Re-Pair の辞書構造の差が主な原因であると推定される．ETDC では元テキストから切り取ったそのままの文字列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 はユニグラムを全て bit 単位の可変長符号化して辞書に保持している．このため，提案手法は Re-Pair と比べて圧縮率が劣るが，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を容易に行える点が優れている．

dna に対して Re-Pair の方が提案手法よりも圧縮率が劣っているのは，dna の $|\Sigma|$ が 4 と小さく，すべてのバイグラムがユニークになるまでに，多くのバイグラムの置き換えを行うことになり，途中で，辞書に追加するバイグラム中の各ユニグラムの長さの和が，バイグラムの置き換えによって縮まったテキストの長さよりも大きくなってしまったためである．従って，Re-Pair においては，バイグラムお置き換えの回数が著しく多くなる場合は，すべてのバイグラムがユニークになるまでバイグラムの置き換えを行うべきではなく，途中で打ち切るべきである．

protein に対しては提案手法も Re-Pair も圧縮率が良くなかった．これは，protein 内での共通の部分文字列の数が少なかったためである．よって，このデータに対して辞書を用いたデータ圧縮は向いていなく，gzip や bzip2 の方が優れた圧縮率となったと考えられる．

5. 結論

本論文では，単語毎の分かち書きされていないテキストに対して，ETDC で符号化する手法を提案した．提案手法の圧縮は，分かち書きされていないテキストでも，テキストの種類によって向き不向きがあった．提案手法は文法圧縮を元にしているので，dna や protein のようなデータよりも，何かしらの文法に基づいて記述されたテキストの方が，より効果を発揮できる手法である．

また，mai2001 のデータに対して，b 値を細かく変動させて最も良かった時の圧縮率と，提案手法による圧縮率を比較したところ，0.005%程しか差がなく，提案手法によってほぼ最適に近い b 値で再帰処理の打ち切りを行うことができていた．

提案手法ではバイグラムの置き換えを行ったときのデータの増加量を，再帰処理で結合するバイグラム $\alpha\beta$ の頻度は，そのときのすべての単語中で最下位の頻度となり，$\alpha\beta$ の結合によって，α と β どちらの頻度の順位にも変動は起きないという仮定の元で計算していたが，正確な値を見積もることができれば圧縮率の向上が期待できる．

提案手法と異なる手法として，正確に見積もった Δ_i を元に，$\Delta_i \geq 0$ ならばバイグラムの置き換えを行わず，$\Delta_i < 0$ ならばバイグラムの置き換えを行うという再帰処理を，バイグラムの頻度順に行っていく手法を考察中である．

参考文献

[1] Amir, A. and Benson, C.: Efficient two-dimensional compressed matching, *Data Compression Conference, 1992. DCC'92.*, IEEE, pp. 279–288 (1992).
[2] Brisaboa, N., Iglesias, E., Navarro, G. and Paramá, J.: An Efficient Compression Code for Text Databases, *Proc. 25th European Conference on Information Retrieval Research (ECIR)*, LNCS 2633, pp. 468–481 (2003).
[3] Kieffer, J. C. and Yang, E.-H.: Grammar-based codes: a new class of universal lossless source codes, *Information Theory, IEEE Transactions on*, Vol. 46, No. 3, pp. 737–754 (2000).
[4] Larsson, N. J. and Moffat, A.: Off-line dictionary-based compression, *Proceedings of the IEEE*, Vol. 88, No. 11, pp. 1722–1732 (2000).
[5] Moffat, A.: Word-based text compression, *Software: Practice and Experience*, Vol. 19, No. 2, pp. 185–198 (1989).
[6] Moura, E., Navarro, G., Ziviani, N. and Baeza-Yates, R.: Fast and flexible word searching on compressed text, *ACM Transactions on Information Systems (TOIS)*, Vol. 18, No. 2, pp. 113–139 (2000).
[7] Schwartz, E. and Kallick, B.: Generating a canonical prefix encoding, *Communications of the ACM*, Vol. 7, No. 3, pp. 166–169 (1964).
[8] Shibata, Y., Kida, T., Fukamachi, S., Takeda, M., Shinohara, A., Shinohara, T. and Arikawa, S.: Speeding up pattern matching by text compression, *Algorithms and Complexity*, Springer, pp. 306–315 (2000).
[9] Shibata, Y., Matsumoto, T., Takeda, M., Shinohara, A. and Arikawa, S.: A BoyerMoore Type Algorithm for Compressed Pattern Matching, *Combinatorial Pattern Matching*, Springer, pp. 181–194 (2000).
[10] Yoshida, S., Morihara, T., Yahagi, H. and Itani, N.: Application of a word-based text compression method to Japanese and Chinese texts, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 85, No. 12, pp. 2933–2938 (2002).