データ圧縮による文字列照合の高速化

2G－1

柴田 裕介　喜田 拓也　竹田 正幸　篠原 歩　有川 節夫
九州大学大学院システム情報科学研究科

1. はじめに

文字列照合は，文字列処理の最も基本的な操作の1つである．文字列照合問題とは，“パターン”と呼ばれる文字列と“テキスト”と呼ばれる文字列が与えられたときに，「テキスト中に出現する全てのパターンを見つけよ.」という問題である．この問題を効率よく解くことは非常に重要であり，多くの解法が提案されている．

一方，データ圧縮の研究も，ディスク容量や通信コストの削減を目的に古くから盛んに行われてきた．データ圧縮法の評価基準は，(1) 圧縮率が高いこと，(2) 圧縮・展開に要する時間の短いことの2つであった．これに対し著者らは，(3) 文字列照合の高速化を可能にする，という新しい評価基準に着目した．すなわち，あらかじめテキストを圧縮しておき，テキストを展開せずに直接展開することで，ディスクからのデータ転送量を削減し，検索時間を短縮するという考え方である．

圧縮テキストに対する文字列照合には2つの目標がある．1つは，圧縮テキストを展開しながら文字列照合を行うよりも高速化すること．もう1つは，非圧縮テキストに対して文字列照合を行うよりも高速化することである．

本論文で紹介する Byte Pair Encoding(BPE) 法[2]は，一般的に広く用いられている圧縮法である LZ77 法や LZW 法と比べて，圧縮率がそれほど良くなく，圧縮時間も遅いため，今まで脚光を浴びることがなかった．しかし，著者らは，符号後がバイトであること，圧縮に用いる辞書が静的であるという特徴に着目し，文字列照合の高速化の観点から，BPE 法の潜在能力を引き出し，文字列照合の高速化に成功した[4]．

2. Byte pair encoding

この節では，BPE 法のアルゴリズムを簡単に紹介し，LZW 法に基づく Unix の Compress コマンドに対して，圧縮率と圧縮時間の比較を行う．

2.1 圧縮アルゴリズム

多くの圧縮アルゴリズムは，頻度の高い文字列を，より短い文字列や数字に置き換えている．BPE アルゴリズムはより単純で，テキスト中の隣り合っているバイトペアの中で最も頻度の高いペアを見つけ，そのペア全てを，テキスト中で使われていないコードに置き換える．この処理を，頻出するペアがなくなるか，ペアを表現するために用いることのできるコードがなくなるまで繰り返し行う．

以下に，このアルゴリズムを例で示す．

$$T_0 = \texttt{ABABCDEBDEFABDEABC.}$$

最も頻度の高いペア AB を，テキスト中で未使用のコード G に置き換える．

$$T_1 = \texttt{GGCDEBDEFGDEGC.}$$

置き換えたテキスト中で最も頻度の高いペア DE を，H に置き換える．

$$T_2 = \texttt{GGCHBHFGHGC.}$$

同様にして，ペア I を，コード GC に置き換える．

$$T_3 = \texttt{GIHBHFGHI.}$$

この操作によって，テキストの長さは，$|T_0| = 18$ から $|T_3| = 9$ と短くなった．

圧縮テキストは，置き換えられた文字列と置き換えられたペアと置き換えたコードからなるペア表で構成される．BPE アルゴリズムは，使用可能な文字が256と限られているため，局所的に変化するテキストに対して圧縮率が下がる場合がある．そこで，テキストを任意のブロックサイズに分割し，ブロックごとに圧縮を行っている．

2.2 圧縮時間の高速化

2.1 節で示したアルゴリズムは，テキストを単純に1次元配列に格納し，ペアの置き換えごとにテキスト上を1回走査する．このため，テキストの長さを N，置き換えに用いたコードの数を ℓ とすると，$O(\ell N)$ 時間かかってしまう．

著者らは，BPE 法の圧縮時間の高速化と，3.1 節で示すパターン圧縮の複雑さの解消のため，以下のようなアルゴリズムの改造を行った．

1) テキスト中の未使用文字をあらかじめ調べる．
2) テキストの1ブロックのみ従来の BPE アルゴリズムを適用する．
3) 2) で構成されたペア表から，複数同時置き換え用パターン照合機械[1]を用いて，テキスト長の線形時間で全てのペアの置き換えを行う．

Accelerates pattern matchin by the data compression
Yusuke Shibata, Takuya Kida, Masayuki Takeda,
Ayumi Shinohara, Setsuo Arikawa
Department of Informatics, Kyushu University, Fukuoka, 812-8581 Japan

2.3 圧縮率，圧縮時間の比較

BPE 法と Compress の圧縮率と圧縮時間を表 1 に示す．実験に用いたデータは以下の通りである．

Brown corpus(B): 米国ブラウン大学で編集された代表的英文コーパス．ファイルサイズは6.8Mbyte.

Genbank(G): GenBank データベースの一部分で，ファイルサイズは 43.3Mbyte.

表 1 圧縮率・圧縮時間.

		BPE		Compress
		2.1 節	2.2 節	
(B)	圧縮率 (%)	51.1	59.0	43.8
	圧縮時間 (秒)	196.9	8.05	12.7
(G)	圧縮率 (%)	49.4	59.5	32.6
	圧縮時間 (秒)	1119.9	43.7	41.3

3. BPE 圧縮テキスト上でのパターン照合

BPE 法は，符号語が 1 バイトであり，圧縮に用いる辞書が静的で辞書のサイズも高々256 である．著者らは，この BPE 圧縮テキストの特徴から，2 つの手法を提案する．

3.1 パターンの符号化による文字列照合

検索対象となる BPE 圧縮テキストのペア表を利用して，与えられたパターンが圧縮テキスト上で可能性のある全てのパターンに置き換え，Aho-Corasick(AC) 法を使って文字列照合を行った．この場合，ℓ をペアの数，m をパターンの数とすると，符号化されるパターンの数は $O(\ell^2 m^2)$ となる．実用的な面から，符号化されたパターンを一般的なパターンに変換し，Shift-Or(SO) 法を用いることにより，符号化されるパターンの数を $O(m^2)$ に減らした．

表 2 に，Unix のスペルチェッカーで用いられる英単語を符号化した際の，平均と最悪時のパターンの個数を示した．

表 2 符号化されたパターン数.

	AC		SO	
	平均	最大	平均	最大
Brown Corpus	37.11	130	2.73	5

3.2 飛ばし読み遷移を使った KMP オートマトン

この方法は単純だが，実用的にとても高速である．KMP オートマトンの各状態について，各コードごとに，そのコードの表す文字列による遷移先の状態番号を格納した 2 次元の表を作成する．

一般に，辞書ベースの圧縮は，テキストの記号列をひとつのトークン (token) として符号化する．トークンはフレーズ辞書 (phrase dictionary) に対するインデックスとなる．LZW 圧縮では，トークンの数は圧縮テキスト長に比例する．しかし，BPE 法では，置き換えに用いるコードの個数である 256 を越えないため，遷移表のサイズも高々256 × m となる．

4. 実験結果

3.1 節，3.2 節で述べた 2 つの方法と，LZW 圧縮テキストに対しての直接的な手法 [3]，非圧縮テキストに対して AC 法, SO 法を用いた場合での検索時間の比較を行った．

実験は，ノートブック (Gateway2000 Solo9100, PentiumII 300MHZ) 上で行った．

表 3 検索時間 [上:Brown Corpus, 下:Genbank] (秒).

BPE (2.2 節)				LZW	非圧縮	
3.1 節		3.2 節		[3]	AC	SO
AC	SO	AC	SO			
0.76	0.82	0.76	0.66	2.42	1.27	1.43
4.73	4.23	4.23	4.23	9.81	6.98	7.86

この結果から，検索時間のほとんどが CPU 時間に依存するような環境においても，本論文で提案する手法は，非圧縮テキストに対して文字列照合を行うよりもほぼ圧縮率程度検索時間を短縮できることがわかった．

また，同じ環境でアリゾナ大学 Manber らによって開発された agrep との比較も行ったが，同様の結果を得ることができた．

本稿で提案した手法は，テキストを圧縮し，圧縮テキストの長さに比例した時間でパターン照合を行うものである．従って，高速化の度合いはほぼ圧縮率に等しい．そこで，agrep で用いられている簡素化 Boyer-Moore 法におけるテキスト文字の読み飛ばしと比較した．その結果，簡素化 Boyer-Moore 法の飛ばし読みの平均の個数が約 5.6 文字であったのに対して，本手法は 1.7 文字であった．

参考文献

1) S. Arikawa and S. Shiraishi. Pattern matching machines for replacing several character strings. *Bulletin of Informatics and Cybernetics*, 21(1–2):101–111, 1984.

2) P. Gage. A new algorithm for data compression. *The C Users Journal*, 12(2), 1994.

3) T. Kida, M. Takeda, A. Shinohara, and S. Arikawa. Shift-And approach to pattern matching in LZW compressed text. In *Proc. 10th Ann. Symp. on Combinatorial Pattern Matching*, Lecture Notes in Computer Science. Springer-Verlag, 1999. to appear.

4) Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa. Byte pair encoding: a text compression scheme that accelerates pattern matching. Technical Report DOI-TR-CS-161, Department of Informatics, Kyushu University, April 1999.