人工知能学会研究会資料
SIG-FPAI-B502-05

文法圧縮のハッシュ領域の削減

Reduction of Working Space for Hash Table of Grammar Compression

水野仁人[1*]　高畠嘉将[1] 坂本比呂志[1]
Masato Mizuno[1]　Yoshimasa Takabatake[1]　Hiroshi Sakamoto[1]

[1] 九州工業大学 情報工学府
[1] Graduate School of Computer Science and Systems Engineering,
Kyushu Institute of Technology, Japan

Abstract: Highlly repetitive texts have been increasing still now. Grammar compression is a technique for efficiently compressing highlly repetitive texts. However, working space for grammar compression is still large. The cause is data structure called *reverse phrase dictionary*. In this paper, we implemented a method for reducting working space of reverse phrase dictionary.

1 はじめに

個人のゲノムやバージョン管理されたテキストやソースコードといった繰り返しの多いテキストが今なお増え続けている．文法圧縮はそういった繰り返しの多いテキストを効率良く圧縮する圧縮手法の一つである．文法圧縮ではデータ中に頻出する共通の文字列を一つの生成規則として集約することで，入力テキストからその入力テキストを一意に導出する文脈自由文法の一種である *straightlineprogram*(SLP) を構築し，そこで得られた SLP を符号化する．SLP の構築に関して最小の文法をみつけるのは NP-hard であることが知られており，LCA[5] や REPAIR[6] といった近似アルゴリズムが提案されている．また SLP の構築と符号化を同時に行うオンライン文法圧縮も提案されており，代表的な例として Fully-online LCA(FOLCA)[1] や Online ESP-index(OESP-index)[2] がある．オンライン文法圧縮は入力長に依存しない作業領域で構築することが可能である．しかしながら，まだ文法圧縮における作業領域は大きい．文法圧縮に必要なデータ構造は主に辞書と逆引き辞書の二つであり，辞書に関しては SLP の情報理論的下限と漸近的に一致するデータ構造が FOLCA で提案されているので逆引き辞書が作業領域が大きい原因であると考えられる．そこで，FOLCA で構築される辞書の実装方法である succinct post order SLP(SPOSLP) における逆引き辞書に登録する非終端記号の数を削減する方法（RRD）[4] が提案されている．本稿では，RRD を実装し従来の逆引き辞書と比較実験，考察を行った．

*連絡先： 九州工業大学大学院 情報工学府
〒 820-8502 福岡県飯塚市川津 680 番 4
E-mail: m_mizuno@donald.ai.kyutech.ac.jp

2 準備

集合 C の要素数は $|C|$ とする．Σ を有限アルファベットの集合とし，$\sigma = |\Sigma|$ とする．文字列 R の長さを $|R|$ と表記する．入力文字列は S とし，$N = |S|$ とする．$\lg = \log_2$ である．

2.1 文法圧縮

文法圧縮とは入力テキストを一意に導出する文脈自由文法の一つ *straightlineprogram*(SLP) $G = \{V, \Sigma, P, X_s\}$ を構築する圧縮手法である．ここで，$V \subseteq X$, $P \subseteq (V \times (\Sigma \cup V)^2)$，$X_s \in V$ は SLP の開始記号であり，SLP の生成規則の形は，$X_i \to X_j X_k (X_i \in V, i < j, k)$ に制限される．各 X_i は非終端記号と呼ぶ．$n = |P|$ とする．SLP の一例を図 1 に示す．

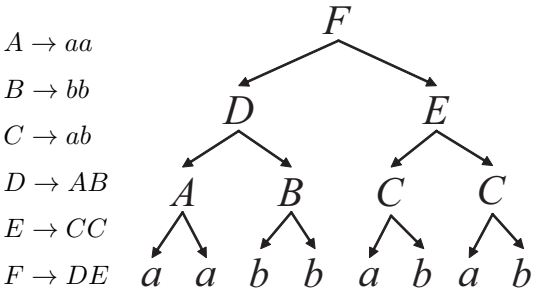

図 1: SLP

表 1: 従来の FOLCA と RRD を用いた FOLCA における領域計算量と構築時間比較．N は入力テキストの長さ，α はハッシュテーブルの loadfactor，n は生成規則数，σ はアルファベットサイズ，lg は $\log_2$，$\lg^*$ は $\lg\lg\lg\lg....$ とする．

	逆引き辞書の領域 (bits)	全体の領域 (bits)	構築時間
FOLCA	$\alpha n \lg(n+\sigma) + n(1+\lg(\alpha n))$	$(1+\alpha)n\lg(n+\sigma) + n(3+\lg(\alpha n))$	$\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))}$
FOLCA+RRD	$\frac{\alpha n}{2}\lg(n+\sigma) + \frac{n}{2}(1+\lg(\alpha n))$	$(\frac{1}{2}+\alpha)n\lg(n+\sigma) + \frac{n}{2}(5+\lg(\alpha n))$	$\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))}$

2.2 辞書と逆引き辞書

文法圧縮を行うためには一般的に辞書 D と逆引き辞書 D^{-1} の 2 つのデータ構造が必要であるので，これらのデータ構造について説明する．文法圧縮の P を表現するデータ構造は辞書 D と呼ばれ，関数 $D(X_i)$ は $X_i \to X_jX_k \in P$ のとき X_jX_k を返し，それ以外のときは $X_\infty X_\infty (X_\infty \notin V)$ を返す．一般的に D は $2n\lg(n)$ ビットの二重配列で表現され，関数 $D(X_i)$ を $O(1)$ で返す．新しいデータの追加の行われない静的な D に関しては $ImprovedESP\text{-}index(ESP\text{-}index\text{-}I)$[7] において，領域が $2n + n\lg n + o(n\lg n)$ ビットという SLP の情報理論的下限である $2n+\lg n! + o(n)$ 近いサイズで関数 $D(X_i)$ を $O(\lg\lg n)$ で返す方法が提案されている．新しいデータの追加が行われる場合に関しても $fullyonlineLCA(FOLCA)$[1] では D を $2n+n\lg(n+\alpha)+o(n)$ ビットという SLP の情報理論的下限に漸近的に一致するサイズで $O(\lg(n)/\lg(\lg(n)))$ 時間で末尾追加とアクセスする方法が提案されている．FOLCA の辞書の詳細については 2.3 節で説明する．逆引き辞書 D^{-1} は X_iX_j が与えられて，$X_i \to X_jX_k \in P$ のとき，X_i を返すデータ構造である．この逆引き辞書の関数は $D^{-1}(X_j, X_k) = X_i$ として表現する．逆引き辞書の一般的なデータ構造はチェイン法によるハッシュテーブルであり，$loadfactor$ を $\alpha \in (0,1]$ とすると，$n(1+\alpha)\lg(n+\alpha)$ ビットで表現でき，$D^{-1}(X_jX_k)$ を $O(1/\alpha)$ 期待時間で返す．静的な場合には ESP-index-I では辞書 D と同じデータ構造を用いて，$O(\lg\lg n)$ 時間で返す方法が提案されている．FOLCA ではハッシュテーブルをリストに登録される値の単調増加性を用いて圧縮しており，その領域は $\alpha n\lg(n+\sigma)+n(1+\lg(\alpha n))$ であり，$D^{-1}(X_j, X_k)$ を $O(1/\alpha)$ 期待時間で返す．

2.3 Fully-online LCA(FOLCA)

第 3 章で fully-online LCA(FOLCA)[1] の辞書のデータ構造を用いて，ハッシュの領域を削減する方法を紹介するので，FOLCA について説明する．FOLCA ではオンラインで SLP の情報理論的下限と漸近的に一致するサイズの辞書 D を構築する．その辞書は succinct post order SLP（SPOSLP）と呼ばれる．post order SLP(POSLP) とは，SLP を開始記号を根とする構文木と見たとき，各非終端記号を後置順にラベリングし，一度出現した内部ノードはその子孫を枝刈りした SLP の表現である．SPOSLP は POSLP を後置順に辿り，葉ノードなら 1，内部ノードなら 0 を並べたビット列 B と POSLP の葉ノードのラベルを後置順に並べた配列 L を用意する．B は動的区間最小最大木 [3] により符号化されている．符号化された B と L を用いると様々な操作が可能となるが，本論文では $Parent(X_i)$ と $Leftchild(X_i)$, $Rightchild(X_i)$ の 3 つの操作について説明する．$Leftchild(X_i)$, $Rightchild(X_i)$ は $O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間で実行される．$Parent(X_i)$ は後置順で見たときの非終端記号 X_i に対応する親を返す関数であり，$O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間で実行できる．FOLCA ではほぼ辞書と逆引き辞書に用いるハッシュテーブルの領域のみで構築可能であり，以下の定理が成り立つ．

定理 1 (S. Maruyama[1]) 長さ N の入力テキストから $O(\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))})$ で $POSLP$ の辞書と逆引き辞書を $(1+\alpha)n\lg(n+\alpha)+n(3+\lg(\alpha n))$ の作業領域で構築できる．

3 SPOSLP の逆引き辞書の領域削減方法（RRD）

本章では，FOLCA[1] のようにオンライン文法圧縮を行い，SPOSLP で辞書の符号化をしている場合の逆引き辞書の削減方法である RRD[4] について説明する．FOLCA に用いられる辞書 SPOSLP は，枝刈りしてあるため一つの非終端記号が内部ノードに出現する回数は一度のみである．また内部ノード X_i に対応する親ノード $Parent(X_i)$，左の子ノード $Leftchild(X_i)$，右の子ノード $Rightchild(X_i)$ を $O(\lg(n)/\lg(\lg(n)))$ 時間で発見できる性質を持つ．これら性質を利用して逆引き辞書へ登録する非終端記号を SPOSLP と対応する内部ノードの両方の子どもが葉となっているもののみとすることで逆引き辞書の領域削減を行う．従来の逆引き辞書同様 (X_j, X_k) が与えられた際のアルゴリズムを以下に記す．

Step1 SPOSLP と対応する内部ノード X_j から親ノード $Parent(X_j)$ へ辿る．

表 2: 実験データ

入力テキスト	手法	圧縮率 (%)	実行時間 (sec)	辞書の登録数	逆引き辞書の登録数	作業領域
einstein (446MB)	FOLCA	0.2	550	376,497	376,497	7.82MB
	FOLCA+RRD		611		116,870	2.94MB
jawiki (73.5GB)	FOLCA	2.61	235,692	547,903,265	547,903,265	13.66GB
	FOLCA+RRD		193,215		170,947,638	8.30GB

Step2 $Leftchild(Parent(X_j)) = X_j$ かつ $Rightchild(Parent(X_j)) = X_k$ ならば $Parent(X_j)$ を返し，アルゴリズムを終了する．

Step3 SPOSLP と対応する内部ノード X_k から親ノード $X_{ParentInternal(k)}$ へ辿る．

Step4 $Leftchild(Parent(X_k)) = X_j$ かつ $Rightchild(Parent(X_k)) = X_k$ ならば $Parent(X_k)$ を返し，アルゴリズムを終了する．

Step5 新しい非終端記号を割り当てる．

SPOSLP と対応する非終端記号 X_i の左の子ノード $Leftchild(X_i)$ または右の子ノード $Rightchild(X_i)$ に内部ノードが含まれる場合，以上のアルゴリズムにより逆引きを行うことで X_i を求めることができる．したがって，逆引き辞書に登録する非終端記号は SPOSLP と対応する内部ノードの両方の子どもが葉となっているもののみで済む．その際の逆引き辞書への登録する非終端記号の数は葉の数を n とした場合，補題 2 が成り立つ．

補題 2 (Y. takabatake[4]) 文法圧縮における *SLP* を *SPOSLP* で表現した場合，逆引き辞書に登録する非終端記号の数は高々 $\frac{n}{2}$ 個である．

また，$Parent(X_i)$, $Leftchild(X_i)$, $Rightchild(X_i)$ の回数が定数回なので $O(\lg(n)/\lg(\lg(n)))$ 時間である．

RRD を用いた FOLCA の作業領域は以下の定理 3 の通りである．

定理 3 (Y. takabatake[4]) 長さ N の入力テキストから $O(\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))})$ で *POSLP* の辞書と逆引き辞書を $(1+\frac{\alpha}{2})n\lg(n+\alpha)+\frac{n}{2}(5+\lg(\alpha n))$ の作業領域で構築できる．

従来の FOLCA と RRD を用いた FOLCA の計算量を比較したものを表 1 に記す．

4 実験

第 3 章の RRD を実装し，実験を行った．全ての実験は，OS:CentOS 5.11，CPU:Intel Xeon E5504(2.00GHz, Quad Core，HT)，Memory：144GB RAM，Compiler：gcc 4.1.2 上で行った．入力テキストとして，圧縮率の高い 2 つのテキスト einstein[1]，jawiki[2] を選択した．従来の FOLCA と RRD を用いた FOLCA を比較する．実験結果を表 2 に示した．

RRD を用いた FOLCA は従来の FOLCA に比べ，逆引き辞書への非終端記号の登録数は einstein が 31%，jawiki も 31%という結果となった．全体の作業領域においては einstein が 38%，jawiki が 61%という結果となった．入力テキストサイズが大きい jawiki では従来の FOLCA より RRD を用いた FOLCA の方が実行時間が短いという結果となった．

5 考察

実験結果は逆引き辞書への非終端記号の登録数が 31% 程度であった，FOLCA[1] 全体の使用領域を大きく削減することができたといえる．補題 2 の通り $\frac{1}{2}$ 以下となることが確認できた．また入力テキストが jawiki の場合，RRD を用いた FOLCA の方が実行時間が短いという結果となった．これは逆引き辞書への非終端記号の登録数が増えるにつれてハッシュの衝突が起こるため従来の FOLCA の方が遅くなったと考えられる．したがって，従来の FOLCA に比べ RRD を用いた FOLCA は生成規則が増加し，逆引き辞書への非終端記号の登録数が増加するほどに実行時間に差が生じる．

6 おわりに

本稿では，ハッシュ領域削減の手法 RRD[4] を実装し，それを FOLCA[1] に用いて従来の FOLCA との比較実験を行った．RRD を用いた FOLCA における逆引き辞書の領域計算量は $\frac{\alpha n}{2}\lg(n+\sigma)+\frac{n}{2}(1+\lg(\alpha n))$，領域計算量は $(\frac{1}{2}+\alpha)n\lg(n+\sigma)+\frac{n}{2}(5+\lg(\alpha n))$，構築時間は $\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))}$ である．今後は，RRD を用いた逆引き辞書の返答時間を高速化することが目標である．

[1] http://pizzachili.dcc.uchile.cl/repcorpus/real
[2] http://dumps.wikimedia.org/jawiki/20151202

参考文献

[1] S. Maruyama, Y. Tabei, H. Sakamoto, and K. Sadakane. Fully-Online Grammar Compression. *In SPIRE*, pages 218-229, 2013.

[2] Y. Takabatake, Y. Tabei, and H. Sakamoto. Online Eelf-Indexed Grammar Compression. *In SPIRE*, pages 258-269, 2015.

[3] G. Navarro and K. Sadakane. Fully-Functional Static and Dynamic Succinct Trees. *ACM Transactions on Algorithms*, 10(3), 2014.

[4] Y. Takabatake and H. Sakamoto. 文法圧縮のための逆引き辞書の省スペース化. (Space Efficient Reverse Phrase Dictionary for Grammar Compression) 人工知能基本問題研究会 *98*, pages 42-47, 2015

[5] H. Sakamoto, S. Maruyama, T. Kida, and S.Shimozono. A Space-Saving Approximation Algorithm for Grammar-Based Compression. *IEICE trans. inf. syst.*, E92-D:158-165, 2009.

[6] N.J. Larsson and A. Moffat. Offline Dictionary-Based Compression. *Proceedings of the IEEE*, 88(11):1722-1732, 2000.

[7] Y. Takabatake, Y. tabei, and H. Sakamoto. Improved ESP-Index: A Practical Self-Index for Highly Repetitive Texts. *In SEA*, pages 338-350, 2014.