定数領域の頻度計算を用いたオンライン文法圧縮アルゴリズム

Online Grammar Compression by Constant Space Item Counting

徳永啓太郎 [1*]　坂本比呂志 [1]
Keitaro Tokunaga[1]　Hiroshi Sakamoto[1]

[1] 九州工業大学情報工学部
[1] Kyusyu Institute of Technology School of Computer Science and Systems Engineering

Abstract: Re-Pair is one of most efficient algorithm for grammar compression. A major drawback of Re-pair is its memory consumption, especially for frequency counting. Basically, Re-Pair uses the priority queue maintaining a list of digrams according to the frequency. We improve Re-Pair using the constant space algorithm for item counting proposed by Karp et al.

1　はじめに

本論文では，文法圧縮の手法の 1 つである Re-Pair[1] と，頻度計算の手法の一つであるシンプルアルゴリズム [2] を用いた，定数領域の頻度計算を用いたオンライン文法圧縮アルゴリズムを提案する．

文法圧縮とは，入力文字列に出現する共通するパターンを，文脈自由文法を用いて生成規則として記録し置換することで，データの冗長性を無くし，データサイズを削減する圧縮手法である．

Re-pair とは，入力文字列に出現する文字のペア全ての出現頻度を数え上げ，その内の最頻出のペアを順に用いて文法圧縮を行うアルゴリズムである．最頻出のペアを生成規則として用いることで圧縮率が良くなるが，出現する全てのペアの出現頻度の計算を行うためにメモリの使用領域が大きくなってしまうという欠点が存在する．

一方，シンプルアルゴリズムは，出現頻度の低いアイテムの情報を切り捨てることで，頻出アイテムの頻度計算を定数領域で可能とするアルゴリズムである．

そこで，本研究では，Re-pair の頻度計算にシンプルアルゴリズムを用い，さらに入力文字列を部分文字列ごとに分けて文法圧縮を行うことで，入力文字列の長さに関わらず定数領域で実行できる，頻度情報を用いたオンライン文法圧縮のアルゴリズムを提案する．

以下に本論文の構成を示す．２章では既存の手法である Re-Pair と頻度計算のためのシンプルアルゴリズムについて説明する．３章で本論文の提案手法を説明する．４章で実験結果を提示する．５章で本研究のまとめを述べる．

*連絡先：九州工業大学情報工学部知能情報工学科
〒 820-0067 福岡県飯塚市川津 680-4
E-mail: k_tokunaga@donald.ai.kyutech.ac.jp

2　準備

この章では準備として，既存のアルゴリズムについて説明する．まず 1 節，2 節では文脈自由文法（CFG）について説明し，３節では Re-Pair について，４節では頻度計算のためのシンプルアルゴリズムについて説明する．

2.1　文脈自由文法（CFG）

N を非終端記号の集合，Σ を終端記号の集合，S を開始記号，P を生成規則の集合とするとき，文脈自由文法 (CFG)G は，$G(N, \Sigma, P, S)$ と定義される．

以下に CFG の例を示す．

$$N = \{S\} \tag{1}$$

$$\Sigma = \{a, b\} \tag{2}$$

$$P = \{S \to aSb, S \to \epsilon(\text{空文字})\} \tag{3}$$

(1)，(2)，(3)，より，$\{a^n b^n : n \geq 0\}$ が生成されるが，生成される文字列が一意でないため，可逆圧縮に用いることができない．

そこで，終端記号の導出を一意に定めるため，Admissible Grammar を導入する．Admissible Grammar は文脈自由文法のサブクラスの 1 つである．Admissible Grammar は次のように定義される．

1. G は deterministic である．
   任意の変数 A について，$A \to \alpha$ となる生成規則がただ一つ存在する．
2. 生成規則の右辺に空語が現れない．
3. $L(G) \neq \emptyset$ である．
4. G は使用しない記号を持たない．
   任意の終端記号・非終端記号 $Z(Z \neq S)$ について，$S \Rightarrow_G x$ の導出の間に少なくとも 1 回は現れる．

図 1: Admissible Grammar(G) の定義

以降では，この Admissible Grammar による文脈自由文法を用いる．

2.2 CFG による圧縮

CFG による圧縮では，入力文字列を終端記号とみなし，これを新たに作成した生成規則によって新たな文字列（非終端記号を含む文字列）に変換する．これを文字列長が 1 になるまで行うと，開始記号 S と生成規則の辞書 P が求められ，これが CFG 圧縮の出力となる．

以下に，入力テキストが”*aabbabab*”の例を場合を示す．

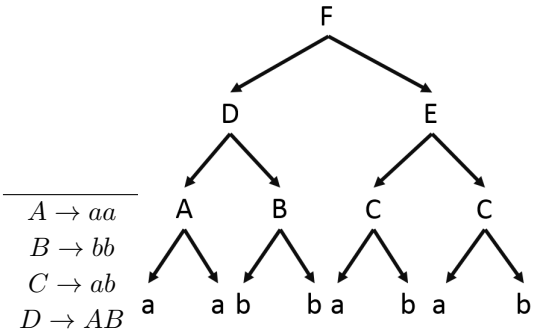

図 2: CFG の生成パターン

index	A	B	C	D	E	F
left	a	b	a	A	C	D
right	a	b	b	B	C	E

図 3: 辞書

例では，入力文字列は図 2 のように変換される．出力は開始記号である F と，生成規則の辞書である図 3 になる．

2.3 Re-Pair

Re-Pair は文法圧縮の手法の 1 つである．以下の説明は，Offline Dictionary Based Compression[1] より引用した．

Re-pair で用いられるアルゴリズムは，入力文字列中の隣接する文字のペアのうち最頻出のペアを新しい文字のペアへと置き換えること，追加された文字に関する文字のペアの出現頻度を再評価すること，2 回以上出現する隣接する文字のペアが出現しなくなるまでこのプロセスを繰り返すことから成る．

Re-Pair

1. 入力文字列において，最頻出の，隣接する 2 文字 a,b のペア ab を確認する．
2. 新しい文字 A を作り，入力文字列中に出現している全ての ab を A に置き換える．
3. ステップ 1 に戻る．

図 4: Re-Pair のアルゴリズム

入力文字列は，単体の文字と再帰的に定義された文字で表される新しい文字列となり，文字数は減少する．[引用者訳]

2.4 頻度計算のためのシンプルアルゴリズム

頻度計算のためのシンプルアルゴリズムのアルゴリズムを A Simple Algorithm for Finding Frequent Elements in Streams and Bags[2] から引用して図 5 に示す．ただし，入力アイテム列の長さを N，シンプルアルゴリズムが保持するアイテムの上限を $\frac{1}{\theta}$ $(0 < \theta < 1)$ とする．

頻度計算のためのシンプルアルゴリズム

$x[1]...x[N]$ is the input sequence
K is a set of symbols initially empty
count is an array of integers indexed by K

for i:= 1,...,N do
  { if $x[i]$ is in K then count$[x[i]]$:= count$[x[i]]$ + 1
    else {insert $x[i]$ in K, set count$[x[i]]$:= 1}
  if $|K| > \frac{1}{\theta}$ then
    for all a in K do
    { count$[a]$:= count$[a]$ - 1,
      if count$[a]$ = 0 then delete a from K}
output K

図 5: 頻度計算のためのシンプルアルゴリズム

図 5 のアルゴリズムより，以下の補題が成り立つ．

補題 2.1 頻度計算のためのシンプルアルゴリズムの領域計算量は $O(\frac{1}{\theta})$ である

補題 2.2 頻度計算のためのシンプルアルゴリズムは，入力アイテム列に登場する，出現頻度が $n\theta$ より大きいアイテムを必ず獲得する．

3 提案手法

この章では提案手法である，頻度計算のためのシンプルアルゴリズムを用いた Re-Pair について述べる．

提案手法は，以下の手順で実行する．n を一度に圧縮する入力文字列の部分文字列の文字数，$\frac{1}{\theta}$ をシンプルアルゴリズムが保持するアイテム数とする．

1. 与えられた入力文字列の先頭から，任意の文字数 n 個の部分文字列を取り出し，その内の文字のペアの出現頻度を全て数え上げ，Re-pair に入力する．
2. シンプルアルゴリズムが保持するアイテムを用いて文法圧縮を行う．さらに，新しくできた文字のペアの出現頻度を数え上げ，Re-pair に入力する．これを，シンプルアルゴリズムが保持する，部分文字列の文法圧縮に使える全てのアイテムについて行う．
3. Re-pair から，最頻出アイテムを取り出し，このアイテムを生成規則として文法圧縮を行う．さらに，新しくできた文字のペアの出現頻度を数え上げる．これを，部分文字列の文字数が 1 個になるまで繰り返す．
4. 圧縮に使用した生成規則の内，Re-pair から取り出したものをファイルに出力する．さらに，生成規則としたアイテムの頻度情報を全て，シンプルアルゴリズムに入力する．
5. 圧縮した部分文字列の次の文字を先頭とし，入力文字列の終端に到達するまで，1 から 4 を繰り返す．

上記の手順の過程で，もしシンプルアルゴリズムの空き領域がなくなった場合，アイテムの出現頻度を全て-1 し，頻度が 0 になった要素を削除することで，空き領域を作る．

シンプルアルゴリズムの領域計算量は $O(\frac{1}{\theta})$，アルゴリズム中で使用した Re-pair の領域計算量は $O(n)$ であり，それぞれの領域計算量は任意で設定可能であるため，定数領域とすることができる．

4 実験結果

Re-pair と提案アルゴリズムの圧縮率・メモリ使用領域･実行時間の比較実験を行った．実験には Pizza&Chili Corpus に公開されていた英文テキストである，einstein.en.txt を用いて行った．

einstein.en.txt は，文字種類数 139，ファイルサイズ約 445MB の，同じパターンが頻出するテキストである．また，提案アルゴリズムのパラメータは，部分文字列の文字数 n とシンプルアルゴリズムのアイテム保持数 $\frac{1}{\theta}$ を以下のように設定して行った．

- Propose1 n:100,000 $\frac{1}{\theta}$:100,000
- Propose2 n:200,000 $\frac{1}{\theta}$:100,000
- Propose3 n:100,000 $\frac{1}{\theta}$:200,000

実験で得られた最大メモリ使用量，生成規則数，実行時間をそれぞれ表 1 に示す．計算機環境は CPU:Intel Xeon E7-8837 (Quad Core、HT @2.67GHz)、Memory:1TB RAM , OS:CentOS6.7(64bit), コンパイラ:gcc 4.1.2 である．

表 1: 実験結果の比較

	heep peak(MB)	生成規則数	実行時間 (sec)
Re-pair	5,358.6	100,991	220.723
Propose1	10.0	300,080	585.58
Propose2	14.2	256,914	799.62
Propose3	15.3	288,328	628.05

まとめ

実験結果の表 1 より，提案アルゴリズムのメモリ使用領域は Re-pair の 0.3 %未満にまで削減されていることが確認できた．よって，本研究の目的であった Re-Pair のメモリ使用量の削減は達成できたと言える．

生成規則数については，提案手法は Re-pair の 2.5〜3 倍になっており，圧縮率は落ちている．しかし，比較した Re-pair を符号化まで行うと，圧縮後のファイルサイズは 549KB となり，圧縮前の $\frac{1}{1000}$ のサイズとなる．よって，提案アルゴリズムの符号化後のサイズはその 2.5〜3 倍と推定でき，十分に実用的な性能である．

一方で，実行時間については，Re-pair と比較して約 3 倍になっており，パラメータ n と $\frac{1}{\theta}$ の値が大きくなるにつれてさらに増加している．

よって，実行時間の削減が今後の主な課題となる．

参考文献

[1] N.J. Larsson and A. Moffat: Offline Dictionary Based Compression, *DCC, pp 296–305(1999)*

[2] M. Karp and S. Shenker: A Simple Algorithm for Finding Frequent Elements in Streams and Bags, *ACM Transactions on Database Systems, Vol. 28, No.1, pp 51–55 (2003)*

[3] S. Maruyama, Y. Tabei: Fully Online Grammar Compression in Constant Space. DCC 2014: 173-182