1T－10 仮想接尾辞木: テキストデータマイニングのための接尾辞配列を用いた高速な部分語頻度計算法

笠井　透　有村博紀　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

大規模テキストデータベースの急速な発展によって，テキストデータから規則性やパタンを発見する研究が注目されている．従来のパタン発見や文字列解析のアルゴリズムの多くは，接尾辞木 (suffix tree)[1] とよばれる索引構造を対象としている．最近，よりコンパクトなデータ構造(*1)である接尾辞配列 (suffix array)[5] が提案され，大規模テキストデータベースにおける先進的索引構造として注目されている．

本研究では，大規模テキストデータベースからの効率よいパタン発見を実現するために，接尾辞配列上で高速なパタン発見を可能にするための基本的実装法について考察する．接尾辞配列を左から右へ一度走査するだけで接尾辞木を仮想的巡回をおこない，テキスト中のすべての部分語の頻度を計算する高速なアルゴリズムを提案する．このアルゴリズムの時間計算量は $O(n)$ であり，素朴なアルゴリズムの $O(n^2)$ に比べて小さいので，パタン探索問題やテキストデータマイニングの高速化に有効である．また，計算機実験の結果も示す．

2 テキスト，接尾辞木，接尾辞配列

本稿では，任意の文字列を語とよぶ．語 s に対して，$s = uvw$ を満たす語 u, v, w を，それぞれ，s の**接頭辞**，**部分語**，**接尾辞**とよぶ．

長さ n のテキストとは，文字列 $A = a_1 a_2 \cdots a_{n-1}\$$ である ($a_i \neq \$$)．位置 i からはじまる A の接尾辞を $A_i = a_i \cdots a_{n-1}\$$ で表し，A_i と A_j の最長共通接頭辞の長さを $lcp(i,j)$ と書く．

テキスト A の**接尾辞木** T_A とは，A の空でない接尾辞全体 $\{A_1, A_2, \ldots, A_n\}$ を表す圧縮トライである [1]．圧縮トライとは，通常のトライから子を一つしかもたない内部節点を取り除き，辺のラベルを合併することを繰り返して得られる木である．左から i 番目の葉を l_i と書き，節点 u, v の**最近共通先祖**を $nca(u,v)$ と書く．

テキスト A の**接尾辞配列** $Pos[1..n]$ とは，接尾辞木の葉だけを一次元整数配列に格納した索引構造である [5]．$Pos[i]$ には，辞書式順序で順位が i の接尾辞の A 中での開始位置が格納される．

2.1 パタン探索問題

パタン発見に関わる問題の多くは，接尾辞木の節点の巡回と動的計画法の適用によって効率よく計算できる．このクラスの問題を一般化する．

値の集合 D と，テキストのすべての位置への初期の割り当て $B : \{1, \ldots, n\} \to D$, D 上の結合的2項演算 $\oplus$ が与えられたとする．**部分語統計問題** (subword statistics problem) とは，テキスト A と割り当て B が与えられたとき，A のすべての部分語 α に対して $C(\alpha) = B(i_1) \oplus \cdots \oplus B(i_m)$ を計算する問題である．ここに，α の出現位置全体を $\{i_1, \ldots, i_m\}, m \geq 0$ とおく．以下の問題は，上記の部分語統計問題の具体例とそれを用いて解ける問題である．

> 部分語枚挙問題，部分語頻度計算問題，出現文書数計算問題 [4]，最長共通部分語問題，最長反復部分語問題

3 接尾辞配列を用いたパタン探索

3.1 単純なアルゴリズム

節点の部分区間の計算は，接尾辞木の深さ優先探索を用いて，つぎのように計算できる．

Binary_Traverse

1. 葉を左から右に走査しながら，葉 l_i にリスト $C_{l_i} := B(Pos[l_i])$ を対応づける．
2. 葉から根へと走査しながら，各内部節点 v に対して，リスト $C_v = C_{v_1} \oplus C_{v_2} \oplus \cdots \oplus C_{v_m}$ を対応づける．v_i は，v の i 番目の子である．

接尾辞配列上での実現には，葉に対応した区間 (L, R) で各節点 u を表現する．さらに，節点での分

Virtual suffix tree: fast computation of subword frequency using suffix array for text data mining

Department of Informatics, Kyushu University, Fukuoka 812-8581, Japan

(*1) 接尾辞木の 1/2 ～ 1/3 の記憶容量である

岐を接尾辞配列上の2分探索で模倣し，スタックを用いて深さ優先探索をおこなう．しかしこの方法の計算時間は $O(n \log n + Q)$ 時間となる．ここに,Q は非圧縮接尾辞トライ $\bar{T}_A$ の節点数 $Q = O(n^2)$ である．

3.2 高速なアルゴリズム

提案のアルゴリズムを図1に示す．このアルゴリズムは，接尾辞配列を左から右に一度だけ走査しながら，スタック S を用いて木の後置順巡回 (post-order traversal) を模倣し，すべての節点に対して部分語統計情報を計算する．

模倣では，対 (L_{new}, H_{new}) が，現在訪問している節点 を葉 L_{new}の先祖で深さ H_{new}をもつ節点として表す．各 $1 \leq i \leq n$ に対して，葉 l_iからの最長最右枝 (rightmost branch) を，葉 l から根方向への，最右辺だけからなる最長の枝と定義し，この枝上の節点のリストをΠ_i とかく．

補題 1 T_Aの後置順巡回は,$\Pi_1, \Pi_2, \ldots, \Pi_n$ をこの順で連結したリストに一致する．

まずアルゴリズムは，$Hgt[i] = lcp(Pos[i], Pos[i+1])$ で定義される**高さ配列** $Hgt[1..n]$ を計算する．ただし，$Hgt[n] = -1$ とする．単純な方法では，計算に $O(n^2)$ 時間必要だが,Pos の逆関数 $Prm = Pos^{-1}$ を用いて $O(n)$ 時間で計算できる．

For ループの i 回目の実行では，スタック S の再上部にはΠ_iが積まれている．そこで，現在の葉 l_i から出発して，節点の深さ比較を用いて，最近共通先祖 $p_i = nca(l_i, l_{i+1})$ までのパス上の節点 q を順にポップしていき，p_iのひとつ手前に達したら,p_iを S にプッシュする．

補題 2 *For* ループの i 回目 $(1 \leq i \leq n)$ の実行で，リストΠ_i の節点が順に出力される．

上の2つの補題から，正当性が示せる．

定理 3 図1のアルゴリズム **Fast_Traverse** は，テキスト A の接尾辞配列 $Pos[1..n]$ から，A 中のすべての部分語の出現頻度を $O(n)$ 時間と $O(n)$ 領域で計算する．

4 計算時間

図1のアルゴリズム Fast_Traverse によって，2.2節の部分語統計問題の各具体例を，長さ n の接尾辞配列上で計算時間 $O(n + M(n))$ で解くことが可能である．ここに，$M(n)$ は，3.1節の接尾辞木を用いた標準アルゴリズムでの演算$\oplus$の合計数である．使用

Algorithm Fast_Traverse

1. Compute $Hgt[1..n]$ and $S := \phi$.
   Push $(\perp, (0, -1))$ into S.
2. For each $i = 1, \ldots, n$, do:
   (a) Push $(B(i), (i, |A_{Pos[i]}|))$.
   (b) $C_{new} := \perp$ and $(L_{new}, H_{new}) := (i, Hgt[i])$.
   Let $(C, (L, H))$ be the top of S.
   (c) While $H > H_{new}$, do:
   (i) Report $(C \oplus C_{new}, H)$.
   (ii) $C_{new} := C \oplus C_{new}$. Then, pop S.
   Let $(C, (L, H))$ be the top of S.
   (d) If $H = H_{new}$ then
   Pop $(C, (L, H))$ from S, and then push $(C \oplus C_{new}, H)$ into S.
   (e) Else if $H < H_{new}$ then
   Push $(C_{new}, (L_{new}, H_{new}))$ into S.

図 1: 接尾辞木のすべての節点の部分区間を計算するアルゴリズム

領域は，Pos と Hgtがそれぞれ $4n$ と $2n$ バイトであり，スタック平均長は n よりかなり小さい．主記憶上の計算機実験の結果を次表に示す．

Algorithm	Fast_Traverse	Binary_Traverse
Time (sec)	2.20	15.06

English text (6MB)[3]. Sun Enterprise3000 (Solaris)

本アルゴリズムは，ディスク上での実装にも適している．深さ配列 Hgt を前処理で計算し，スタックを主記憶上におけばよい．このとき，テキスト配列と接尾辞配列へのアクセスは必要とせず，ディスク上の Hgt 配列のアクセスパタンは逐次的である．

5 おわりに

本稿では，高速なパタン探索手法について論じ，接尾辞配列上で接尾辞木の巡回を実現する線形時間アルゴリズムを与えた．テキストデータマイニングに関して，文献 [2] の語相関パタン発見問題が，接尾辞木を用いた場合と同じ時間計算量で実装可能である．詳細に関しては，別の機会に述べたい．

参考文献

[1] E.M.McCreight. A space-economical suffix tree construction algorithm. *JACM*, 23(2):262-272 (1976).

[2] H.Arimura, S.Shimozono. Maximizing agreement between a classification and bounded or unbounded number of associated words. In *Proc. ISSAC*, (1998).

[3] R. Harris, Abstract Index, Monash Univ (1998).

[4] L.C.K. Hui. Color set size problem with applications to string matching. In *Proc. of 3rd CPM*, 230-243 (1992).

[5] U.Manber, G.Myers. Suffix arrays: A new method for on-line string searches. *SIAM J. Computing*, 22(5):935-948 (1993).