Vol. 34 No. 4 情報処理学会論文誌 Apr. 1993

ショートノート

大型データベースのための最長共通部分列の一高速抽出法

丹波直人[†] 田中榮一[††]
増田澄男[††] 細島美智子[†††]

過去に幾つかの最長共通部分列の抽出法が提案されているが，それらを直接大量データに用いると抽出速度が遅く実用的でない．そこで，類似商標語句の高速検索を目的に提案された Kuo-Cross 法と誤り訂正を高速に行える階層ファイルを結合し，高速に最長共通部分列を抽出する方法を提案する．単語長 6～8 の合計 12,178 語について実験を行い，その結果，平均して Kuo-Cross 法の約 57% の時間で抽出でき，記憶量の増加も Kuo-Cross 法に対し約 0.08% にすぎなかった．

A Fast Method for Finding Longest Common Subsequences in a Large Database

Naoto Tamba,[†] Eiichi Tanaka,[††] Sumio Masuda[††]
and Michiko Hosojima[†††]

Several methods for extracting longest common subsequences were proposed, but they were not so fast to apply them to a large database. Recently Kuo and Cross proposed a faster method that will be used for a retrieval system of trademark words or phrases. Furthermore a high speed error-correcting method using a hierarchical file was known. This paper describes a new method that is obtained by combining Kuo-Cross's method and a hierarchical file to find longest common subsequences in a large database. The experimental result for 12,178 words of length 6, 7 and 8 shows that the computing time can be reduced to 57% of Kuo-Cross's method. On the contrary, the increase of memory requirement is only 0.08% in comparison with Kuo-Cross's method.

1. はじめに

系列間の最長共通部分列 (longest common subsequence : LCS) を求めることは類似商標語句の検出の他，多くの問題で必要になる[1]．最近 Kuo と Cross[2] は，必要な記憶量は多いが高速に LCS を抽出する方法を提案している．本文の方法は類名表記を用いた階層ファイル[3]と Kuo-Cross の方法を結合するもので，Kuo-Cross の方法と比べてかなり速く，しかも記憶量はほとんど変わらない．

† 神戸大学大学院工学研究科電子工学専攻
Division of Electronics Engineering, Graduate School, Kobe University
†† 神戸大学工学部電気電子工学科
Department of Electrical and Electronics, Engineering, Faculty of Engineering, Kobe University
††† 宇都宮大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Utsunomiya University

2. LCS の抽出法

2 つの文字列を $A=a_1a_2\cdots a_m$, $B=b_1b_2\cdots b_n$ とし，A と B の LCS を $LCS(A, B)$ と書く，A をアルファベット順にソートしたものを $\tilde{A}=\tilde{a}_1\tilde{a}_2\cdots\tilde{a}_m$, A の長さを $|A|$ と書く．このとき，

$$|LCS(\tilde{A}, \tilde{B})| \geq |LCS(A, B)|$$

であり，$u=\max(m, n)$ として $LCS(\tilde{A}, \tilde{B})$ を求める手数は $O(u)$, $LCS(A, B)$ のそれは $O(mn)$ である[1]．そこで，すべての文字列をあらかじめソートしておき，ソートした文字列と元の文字列の対 $(\tilde{A}, A)$ を記憶しておく．入力を I, 文字列を A_k とし，$LCS(I, A_k)$ を求めるとき，まず $|LCS(\tilde{I}, \tilde{A}_k)|$ を計算し，それが閾値以上なら $LCS(I, A_k)$ を計算することにすると計算時間を短縮できる．これをパスカル風に書くと次のよ

うになる．ここで，EMPTY(x) は集合変数 x の値を空集合とする手続きである．

```
threshold :=0;
EMPTY (maxlcs);
{maxlcs は複数の LCS を記憶する集合変数}
for k:=1 to n do begin
{n : 文字列数}
  if|LCS(Ĩ, Ãk)|≥threshold then
    if|LCS(I, Ak)|=threshold then
      maxlcs←LCS(I, Ak);
    else if|LCS(I, Ak)|>threshold then begin
      threshold :=|LCS(I, Ak)|;
      EMPTY(maxlcs); maxlcs←LCS(I, Ak)
    end;
end.
```

これが Kuo-Cross 法である．

文字の出現頻度[4]を基にアルファベットを次のように分類する．

分類 2

A＝{e, s, a, r, i, n}

B＝{o, t, l, d, u, c, m, g, h, p, b, y, f, k, w, v, ', j, x, z, q, -}

分類 3

A＝{e, s, a, r}，　B＝{i, n, o, t, l}

C＝{d, u, c, m, g, h, p, b, y, f, k, w, v, ', j, x, z, q, -}

ここで，分類2ではA，Bの，分類3ではA，B，Cの発生頻度がほぼ等しくなるようにしている．A，B，Cを類名という．2類の類名で文字列を書くと，たとえば people は BABBBA となる．これを people の第1種の類名表記という．ここにはAは2個，Bは4個含まれる．この類名の個数を用いて E2(people)＝(2, 4) と書き，これを people の第2種の類名表記という．このとき E2(people)$_1$＝2，E2(people)$_2$＝4 のように書く．第1種の類名表記を用いると文字列集合を**図1**のように2段の階層ファイルにまとめることができる[3]．

階層ファイルと Kuo-Cross 法は独立な考え方に基づいているから，両者の長所を取り入れた方法を作ることができる．第2種の類名表記を用い，文字列をソートしたものも加えて**図2**のようなファイルを作る．一般に，第2種の類名表記は複数の第1種の類名表記を代表しているから，第1種の類名表記による文字列の分類と第2種の類名表記によるそれとは同じではない．ここで，文字が s 類に分類されているとして次の関数を定義する．

$$NE2(I, E_k)=\sum_{j=1}^{s}\min\{E2(I)_j, E_{kj}\}$$

ここで，E_k は k 番目の第2種の類名表記，E_{kj} は E_k の j 番目の値である．

$NE2(I, E_k)$ の計算例として，E_k に図2の類名表記 (3, 3) および (2, 4) を，I を people とした場合を挙げておく．

$$\begin{aligned}NE2(I, E_k)&=\sum_{j=1}^{2}\min\{E2(\text{people})_j, E_{kj}\}\\&=\min\{(2,4)_1, (3,3)_1\}\\&\quad+\min\{(2,4)_2, (3,3)_2\}\\&=\min\{2,3\}+\min\{4,3\}\\&=5\end{aligned}$$

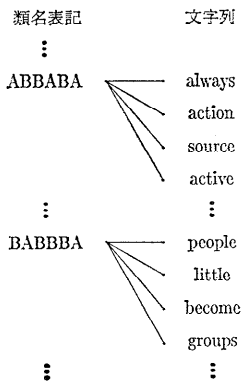


図 1　階層ファイルの一部
Fig. 1　A part of a hierarchical file.

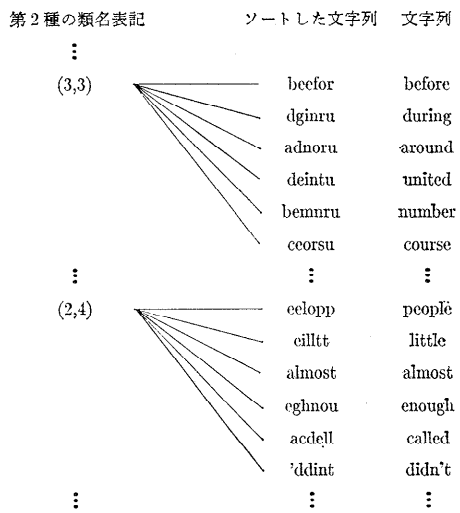


図 2　LCS 用の階層ファイルの一部
Fig. 2　A part of a hierarchical file for LCS.

$$NE2(I, E_k) = \sum_{j=1}^{2} \min\{E2(\text{peple})_j, E_{kj}\}$$
$$= \min\{(2, 4)_1, (2, 4)_1\} + \min\{(2, 4)_2, (2, 4)_2\}$$
$$= \min\{2, 2\} + \min\{4, 4\}$$
$$= 6$$

このファイルを用いて，次のような LCS の抽出法が考えられる．

threshold :=0 ;
EMPTY(*maxlcs*) ;
{*maxlcs* は複数の LCS を記憶する集合変数}
for k:= 1 **to** N_c **do begin**
{N_c : 類名表記数，N_k : k 番目の部分辞書の単語数}
{A_{kj} : k 番目の部分辞書の j 番目の文字列}
  if $NE2(I, E_k) \geq threshold$ **then**
    for j:=1 **to** N_k **do begin**
      if $|LCS(\tilde{I}, \tilde{A}_{kj})| \geq threshold$ **then**
        if $|LCS(I, A_{kj})| = threshold$ **then**
          $maxlcs \leftarrow LCS(I, A_{kj})$;
        else if $|LCS(I, A_{kj})| > threshold$ **then**
          begin
            $threshold := |LCS(I, A_{kj})|$;
            EMPTY(*maxlcs*) ; $maxlcs \leftarrow LCS(I, A_{kj})$
          end ;
    end
end.

3. 実　　験

実験は英単語 12,178 語（6〜8 文字）の辞書を使用し，各単語長別に単語を 100 個選んで辞書中の単語との LCS の抽出を行った．また，辞書中のすべての単語と直接 LCS を求める全辞書法および Kuo-Cross 法についても実験した．用いた計算機は X 68000，プログラムはC言語で記述した．用いた辞書の情報を**表 1** に，実験結果を**表 2** と**図 3** に示す．第 2 種の類名表記の部分を第 1 種の類名表記をソートしたものに置き替えた方法も考えられる．表 2 で第 1 種類名表記と書いているのはこれを指している．図 3 においてM2, M3 はそれぞれ第 2 種の類名表記を用いた 2 類，3 類の階層ファイル法である．結果は，速度では 3 類の方が高速であり，記憶量の増加は類名表記数の少ない 2 類の方が少なかった．高速だった 3 類では，全辞書法の約 7.4%，Kuo-Cross 法の約 53% の時間で実行されており，記憶量の増加は Kuo-Cross 法に対して約 0.12% の増加にすぎなかった．2 類，3 類全体の平均でも Kuo-Cross 法の約 57% の時間で実行でき，記憶量の増加は約 0.08% にすぎなかった．

表 1 辞書の単語数と類名表記数
Table 1 Numbers of words in the dictionary and class name expressions.

単語長	単語数	類名表記数(2類)	類名表記数(3類)
6	3,859	7	28
7	4,277	8	34
8	4,042	9	39
合計	12,178	24	101

表 2 実験結果
Table 2 The result of the experiment.

方法			処理時間	記憶容量
全辞書法			1	1
Kuo-Cross 法			0.1390	2.0000
本文の方法	第1種類名表記	2類	0.0840	2.0020
		3類	0.0740	2.0083
	第2種類名表記	2類	0.0838	2.0006
		3類	0.0739	2.0024

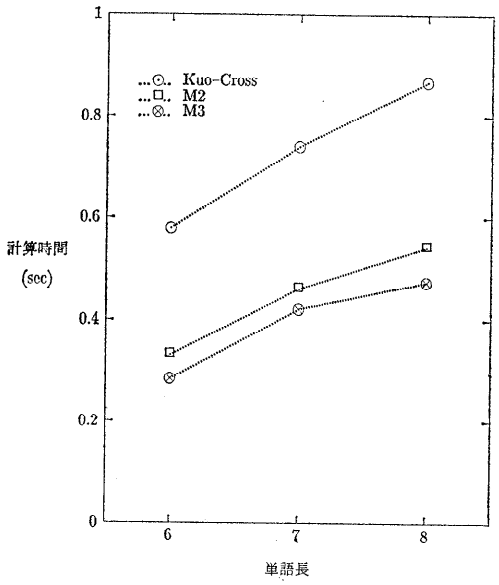


図 3 本文の方法と Kuo-Cross 法の計算時間
Fig. 3 Computing times by the proposed method and Kuo-Cross's method.

4. おわりに

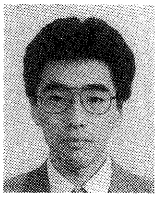
Kuo-Cross 法と階層ファイルを結合した最長共通部分列抽出法を提案し，実験してその有効性を確かめた．長さ 6〜8 の合計 12,178 語の辞書について実験した結果，本方法は平均で全辞書法の約 7.9%，Kuo-Cross 法の約 57% の時間で実行でき，記憶量の増加も Kuo-Cross 法に対し約 0.08% の増加にすぎなかった．

より大量のデータに対しては，多段階層ファイルを用いるとよいと思われる．

参考文献

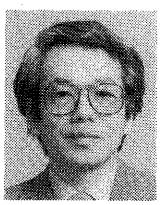
1) Sankoff, D. and Kruskal, J. B.: *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, p. 382, Addison-Wesley, Mass. (1983).
2) Kuo, S. and Cross, G. R.: A Two-step String-matching Procedure, *Pattern Recognition*, Vol. 24, No. 7, pp. 711-716 (1991).
3) Tanaka, E., Toyama, T. and Kawai, S.: High Speed Error-correction of Phoneme Sequence, *Pattern Recognition*, Vol. 19, No. 5, pp. 407-412 (1986).
4) Kucera, H. and Francis, W. N.: *Computational Analysis of Present-day American English*, Brown Univ. Press (1967).

(平成4年4月13日受付)
(平成5年2月12日採録)



丹波 直人

昭和44年生．平成4年神戸大学工学部電気工学科卒業．現在同大大学院修士課程在学中．階層ファイル，最長共通部分列に関する研究に従事．電子情報通信学会会員．



田中 榮一（正会員）

昭和 37 年大阪府立大学工学部電気工学科卒業．昭和 43 年大阪大学大学院工学研究科博士課程修了．昭和 42 年大阪府立大学工学部電気工学科助手，講師，宇都宮大学工学部情報工学科教授を経て，現在，神戸大学工学部電気電子工学科教授．アルゴリズム等の研究に従事．電子情報通信学会，ソフトウェア科学会，IEEE，日本数学会，各会員．

増田 澄男（正会員）

昭和31年生．昭和54年大阪大学基礎工学部情報工学科卒業．昭和59年同大大学院後期課程修了．工学博士．同大学基礎工学部情報工学科助手，講師を経て，現在，神戸大学工学部電気電子工学科助教授．主としてグラフアルゴリズムの設計に従事．電子情報通信学会，IEEE 各会員．

細島美智子

昭和 53 年宇都宮大学農学部農芸化学科卒業．同年宇都宮大学工学部情報工学科技官．主に同学科計算機室の管理，ソフトウェアの設計・制作に従事．