Rep. Fac. Sci., Kagoshima Univ., (Math., Phys., & Chem.),
No. 16, p. 27-30, 1983

AN IMPROVEMENT OF A PATTERN MATCHING ALGORITHM BY MAKING USE OF A STATISTICAL PROPERTY OF LANGUAGES

Akira TOGASI* and Hirotoki KAWASAKI*

(Received September 9, 1983)

Abstract

In this paper we give an algorithm improving processing times for a pattern matching problem, which is often used to information retrieval, by using statistical properties of languages.

1. Introduction

The statistical properties of the natural languages have been investigated by several researchers. Especially there are many papers describing the investigation about properties relating to consecutive characters in a word, and also the properties of the vocal sounds in it, since the alphabets used in the India-European language being very small sets, we can easily seek statistical properties of languages.

In this paper we consider an improvement about the number of handling times of the pattern matching to the natural language.

2. Pattern Matching

Suppose that τ is a string whose length is m, π is a string whose length is n, and assume that $m \geq n$.

In this paper, we say τ a text and π a pattern. Moreover suppose τ and π are strings on the alphabet $\Sigma = \{a_1, \ldots, a_s\}$ i.e. $\tau, \pi \in \Sigma^+ = \Sigma^* - \{\lambda\}$.

Then the problem whether a pattern π exists in a text τ or not is said to be a pattern matching problem. And it is the problem we consider in this paper.

3. Basic Algorithm

We give one of the most primitive algorithms for the pattern matching (algorithm 1).

First, let us denote the i-th character of τ and π by $\tau(i)$ and $\pi(i)$, respectively. Fig. 1 is the flowchart of algorithm 1.

* Department of Mathematics, Faculty of Science, Kagoshima University, Kagoshima, Japan.

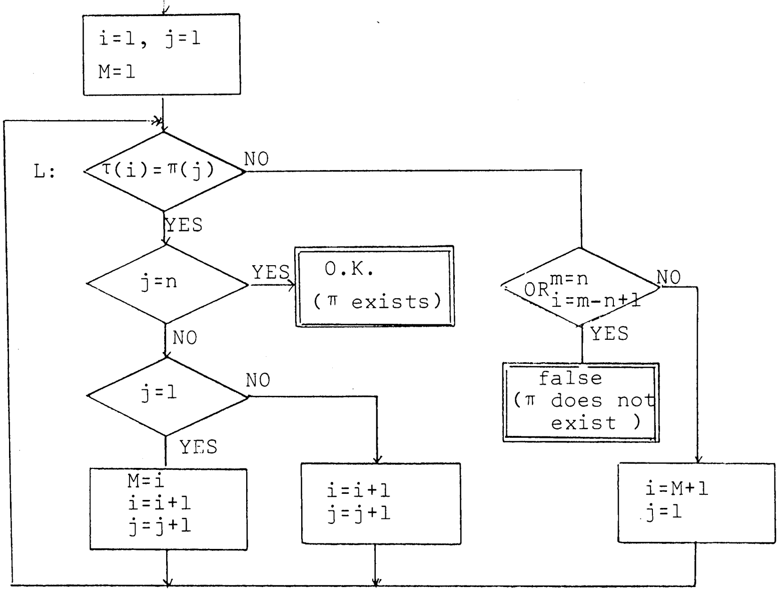

Fig. 1 The flowchart of Algorithm 1.

The program of Algorithm 1.

```
      $i=1$, $j=1$, $M=1$
L : IF $\tau(i)=\pi(j)$
        THEN    IF $j=n$
                THEN    O.K.
                ELSE    IF $j=1$
                        THEN    $M=i$
                                $i=i+1$
                                $j=j+1$
                                GO TO L
                        ELSE    $i=i+1$
                                $j=j+1$
                                GO TO L
        ELSE    IF $i=m-n+1 \vee m=n$
                THEN    false
                ELSE    $i=M+1$
                        $j=1$
                        GO TO L
```

Algorithm 1 starts from $M=1$, compares $\tau(i+M-1)$ with $\pi(i)$. If the matching succeeds to $i=n$, it turns on O.K. (i.e. π exists in τ), and if the unmatching happens on the halfway, adds one to M and does the same things. When $M=m-n+1$, if the unmatching happens, as it is impossible to increase M more, so it turns on false(π does not exist in τ). Then let T_N be the number of times comparing $\tau(i+M-1)$ with $\pi(i)$ until it turns on O.K. or false.

The number T_N is n at least (when it turns on O.K. on $M=1$), and $n\times(m-n+1)$ at most (when it turns on O.K. or false on $M=m-n+1$, $i=n$ after the unmatchings happen on $i=n$ from $M=1$ to $M=m+n$), that is,

$$n\leq T_N\leq n\times(m-n+1).$$

Definition 1. Given a text τ and a pattern π, we define *the referring order* by the number of times going through the label L of algorithm 1.

4. An improvement of the algorithm

When the text τ is a string on the natural language, the appearance pattern of τ is not uniform. The probability of $\tau(i)=a_j$ is not generally equal to that of $\tau(i)=a_j'(j\neq j')$. So let us assume that p_{ij} is the probability of $\tau(i)=a_j$ and p_{ij}' that of $\tau(i)=a_j'$.

Then we can define an appearance pattern matrix ;

$$\mathrm{P}=\begin{pmatrix} p_{11} & \cdots\cdots\cdots & p_{m1} \\ \vdots & & \vdots \\ p_{1s} & \cdots\cdots\cdots & p_{ms} \end{pmatrix}$$

in this, trivially we have

$$\sum_{j=1}^{s} p_{ij}=1 \qquad (i=1,\ 2,\ \ldots,\ m)$$

Given π such that

$$\pi=a_{i1}\ldots a_{in}\in\Sigma^{+},$$

the probability that π exists from k-th in τ is

$$\prod_{j=1}^{n} p_{j+k-1,ij} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (1)$$

Definition 2. Given the pattern π, we define *the averaged referring order* by the average of the referring order of matching by the text τ which has an appearance pattern matrix P.

Moreover, suppose T_k is the averaged referring order that π exists from k-th in τ. Let us denote by F_r the averaged referring order that the unmatching happens on $M=r(r=1,\ 2,\ \ldots,\ k-1)$. Then we have immediately

$$F_r=\sum_{h=1}^{n} h\Big(\prod_{j=1}^{h-1} p_{j+r-1,ij}\times(1-p_{h+r,\ i_h})\Big) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

(F_r is independent of k), and

$$\therefore\quad T_k=\sum_{i=1}^{k-1} F_i+n \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (3)$$

Therefore the averaged referring order T of Algorithm 1 is

$$T=\sum_{k=1}^{t} T_k\cdot\prod_{j=1}^{n} p_{j+k-1,\ ij} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (4)$$

$$(t=m-n+1)$$

Now we know that T is dependent on $\{1,\ 2,\ \ldots,\ t\}$ which is the order M moves in the algorithm 1. So we can decrease T by changing the order M moves.

Suppose that

$$W=\{w\mid w \text{ is a permutation of } \{1,\ 2,\ \ldots,\ t\}\},$$

and when we take a permutation w such that

$$W \ni w = \{w_1, \ldots, w_t\}$$

as the new order M moves, we denote the new T_k by $T_k{}^w$.

Then $T_k{}^w$ is

$$T_k{}^w = \sum_{i=1}^{k-1} F_{w_i} + n \cdots\cdots (5)$$

So T^w is

$$T^w = \sum_{k=1}^{t} T_k{}^w \cdot \prod_{j=1}^{n} p_{j+k-1,\ ij} \cdots\cdots (6)$$

Therefore choosing w^* such that

$$T^{w^*} = \min_{w \in W} T^w$$

as new order M moves, then

$$T^{w^*} \leq T.$$

So we can improve the algorithm 1. We show the algorithm 2 improving the algorithm 1.

The program of Algorithm 2

```
    Search w* such that $\min_{w \in T} T^w$
    $i = w_1$, $j = 1$, $M = 1$
L : IF $\tau(i) = \pi(j)$
        THEN    IF $j = n$
                THEN    O.K.
                ELSE    IF $j = 1$
                        THEN    $M = i$
                                $i = i + 1$
                                $j = j + 1$
                                GO TO L
                        ELSE    $i = i + 1$
                                $j = j + 1$
                                GO TO L
        ELSE    IF $i = t \vee m = n$
                THEN    false
                ELSE    $i = w_{M+1}$
                        $j = 1$
                        GO TO L
```

References

[1] Suen, C.Y : *n-Gram Statistics for Natural Language Understanding and Test Processing*, IEEE Trans. Vol. PAMI=1, No. 2, April 1979

[2] Knuth, E.E., Morris, J.H.Jr., Pratt, V.R. : *Fast Pattern Matching in Strings,* SIAM J. Comput Vol. 6, No. 2, pp. 323-350 (1977).

[3] Boyer, R.S. and Moore, J.S. : *A Fast String Searching Algorithm,* Comm. ACM, Vol. 20, No. 10, pp. 762-772 (1977).

[4] Guibas, L.J. and Odlyzko, A.M. : *A New Proof of the Linearity of the Boyer-Moore String Searching Algorithm,* SIAM J. Comput., Vol. 9, No. 4, pp. 672-682 (1980).