Information Processing Letters 125 (2017) 1–4

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

# An acceleration of FFT-based algorithms for the match-count problem

Kensuke Baba

*Fujitsu Laboratories, Kawasaki, 211-8581, Japan*

ARTICLE INFO

*Article history:*
Received 28 November 2016
Received in revised form 24 April 2017
Accepted 24 April 2017
Available online 27 April 2017
Communicated by B. Doerr

*Keywords:*
Algorithms
Match-count problem
Convolution
FFT
Processing time

ABSTRACT

The match-count problem on strings is a problem of counting the matches of characters for every possible gap of the starting positions between two strings. This problem for strings of lengths $m$ and $n$ ($m \leq n$) over an alphabet of size $\sigma$ is classically solved in $O(\sigma n \log m)$ time using the algorithm based on the convolution theorem and a fast Fourier transform (FFT). This paper provides a method to reduce the number of computations of the FFT required in the FFT-based algorithm. The algorithm obtained by the proposed method still needs $O(\sigma n \log m)$ time, but the number of required FFT computations is reduced from $3\sigma$ to $2\sigma + 1$. This practical improvement of the processing time is also applicable to other algorithms based on the convolution theorem, including algorithms for the weighted version of the match-count problem.



## 1. Introduction

In this paper, we address the match-count problem on strings [1], which is, for two strings, to compute the vector whose $i$th element is the number of matches between corresponding characters in the strings aligned with the gap $i$ between the start positions.

The match-count problem for strings of lengths $m$ and $n$ ($m \leq n$) over an alphabet $\Sigma$ of size $\sigma$ is solved in $O(\sigma n \log m)$ time using the algorithm based on the convolution theorem [2] and a fast Fourier transform (FFT), while the naive algorithm requires $O(mn)$ comparisons of characters. This FFT-based approach was developed by Fischer and Paterson [3]. This algorithm is efficient for the lengths $m, n$ of input strings but is not suitable for applications with a large alphabet size $\sigma$ such as documents written in natural language.

We propose a method to reduce the number of FFT computations required in the FFT-based algorithm. The computations of FFT are the main part of the algorithm, and the number of the computations is proportional to $\sigma$. Although the algorithm obtained by the proposed method still needs $O(\sigma n \log m)$ time, the number of required FFTs is about two-thirds of the original algorithm. We describe the key idea briefly. The FFT-based algorithm computes the output vector using element-wise additions of the $\sigma$ vectors obtained by $\sigma$ convolutions. Because a convolution is computed using two FFTs, element-wise multiplications, and an inverse of the FFT (IFFT), the number of FFTs is $3\sigma$ (on the assumption that each convolution is computed without dividing vectors). We change the order of the IFFT and the element-wise additions done after the convolutions, and then the $\sigma$ IFFTs are computed as a single IFFT. Thus, the total number of FFT computations is reduced from $3\sigma$ to $2\sigma + 1$.

Another approach to improve the speed of the FFT-based algorithm is randomization. Atallah et al. [4] randomized the algorithm to reduce the processing time using a trade-off with the accuracy of the solution's estimation, and several improvements in the randomized algorithm were proposed [5–8]. The algorithm obtained by the proposed method computes the exact solution instead of an approximated one. Additionally, it is applicable to random-

*E-mail address:* baba.kensuke@jp.fujitsu.com.

http://dx.doi.org/10.1016/j.ipl.2017.04.013
0020-0190/

ized algorithms, which yields a better trade-off between the processing time and the accuracy of approximation.

The acceleration method is applicable to other algorithms that partially use the computation of convolutions. Abrahamson [9] proposed the essential idea of an $O(n\sqrt{m}\log m)$ algorithm for the match-count problem which is faster than the FFT-based algorithm when $\sigma$ is large. This algorithm is regarded as a combination of the FFT-based algorithm for characters that occur frequently in the shorter string and a straightforward algorithm to count matches for the other characters. Fredriksson and Grabowski [10] proposed a parallel computation for convolutions which improves the complexity of Abrahamson's algorithm to $O(n\sqrt{m/w}\log m)$ for a word size $w = \Omega(\log n)$. The improved algorithm also uses the FFT-based algorithm for words obtained by packing plural characters into a single word of size $w$. Therefore, the acceleration method is applicable to those algorithms also.

## 2. Problem

Let $\Sigma$ be a finite set of characters. For an integer $n > 0$, $\Sigma^n$ is the set of the strings of length $n$ over $\Sigma$. For a string $s$ of length $n$, $s_i$ for $0 \le i < n$ is the $i$th character of $s$. For strings $s$ and $t$, $st$ is the concatenation of $s$ and $t$. For a character $a$ and an integer $n > 0$, $a^n$ is the string of $n$ $a$'s.

Let $\delta$ be a function such that $\delta(a,b)$ for $a, b \in \Sigma$ is 1 if $a = b$, and 0 otherwise. Let $x \notin \Sigma$ be the *never-match character*, that is, $\delta(x,a) = \delta(a,x) = 0$ for any $a \in \Sigma$. Then, the *score vector* between $s \in \Sigma^m$ and $t \in \Sigma^n$ $(m \le n)$ is defined as the vector $C(s,t)$ whose $i$th element for $0 \le i \le m+n-2$ is

$$c_i = \sum_{j=0}^{m-1} \delta(s_j, t'_{i+j}), \tag{1}$$

where $t' = x^{m-1} t x^{m-1}$. The *match-count problem* is a problem of computing the score vector between two strings.

## 3. Algorithm

We introduce the FFT-based algorithm [3] as the basic algorithm for the match-count problem, and we present a modification to it.

### 3.1. Basic algorithm

We introduce the $O(\sigma n \log n)$ algorithm that computes the score vector between two strings in $\Sigma^n$, where $|\Sigma| = \sigma$. The algorithm can be extended to an $O(\sigma n \log m)$ algorithm for two strings of lengths $n$ and $m$ $(< n)$ by dividing the longer string in the same way as the technique used in [4].

Let $\varphi$ be a function from $\Sigma \cup \{x\}$ to $\mathbf{N}$, whose restriction from $\Sigma$ to $\{0, 1, \dots, \sigma - 1\}$ is bijective, and such that $\varphi(x) = 0$. Let $\phi$ be the function from $\Sigma \cup \{x\}$ to $\{0,1\}^\sigma$ such that the $i$th element of $\phi(a)$ for $0 \le i < \sigma$ and $a \in \Sigma \cup \{x\}$ is 1 if $i = \varphi(a)$ and $a \in \Sigma$, and 0 otherwise. Then, $\langle \phi(a), \phi(b) \rangle = \delta(a,b)$ for $a, b \in \Sigma \cup \{x\}$. Let $l = 2n - 1$. Let $S$ and $T$ be the $l \times \sigma$ matrices

$$S = \left(\phi(s_{n-1})^T, \phi(s_{n-2})^T, \dots, \phi(s_0)^T, O^T, \dots, O^T\right) \text{ and}$$
$$T = \left(\phi(t_0)^T, \phi(t_1)^T, \dots, \phi(t_{n-1})^T, O^T, \dots, O^T\right), \tag{2}$$

where $M^T$ is the transposed matrix of a matrix $M$ and $O$ is the zero vector of dimensionality $\sigma$. For any matrix $M$, we denote the $(i,j)$-element of $M$ by $M_{i,j}$ with both indices starting from 0. Then, Equation (1) is modified using Equation (2) as

$$\begin{aligned} c_i &= \sum_{j=0}^{n-1} \langle \phi(s_j), \phi(t'_{i+j}) \rangle = \sum_{j=0}^{l-1}\sum_{k=0}^{\sigma-1} S_{j,k} \cdot T_{i-j,k} \\ &= \sum_{k=0}^{\sigma-1}\sum_{j=0}^{l-1} S_{j,k} \cdot T_{i-j,k}, \end{aligned} \tag{3}$$

for $0 \le i < l$, where $T_{i,k} = T_{l+i,k}$ for any $i$ and $0 \le k < \sigma$. In Equation (3), we can see the circular convolution

$$U_{i,k} = \sum_{j=0}^{l-1} S_{j,k} \cdot T_{i-j,k} \quad (0 \le i < l) \tag{4}$$

for each $0 \le k < \sigma$. Then,

$$c_i = \sum_{k=0}^{\sigma-1} U_{i,k} \tag{5}$$

for $0 \le i < l$.

Using Equation (4), the vector $\left(U_{0,k}, U_{1,k}, \dots, U_{l-1,k}\right)$ is the circular convolution of the two vectors $(S_{0,k}, S_{1,k}, \dots, S_{l-1,k})$ and $\left(T_{0,k}, T_{1,k}, \dots, T_{l-1,k}\right)$ for each $0 \le k < \sigma$. Let $F_n$ be the matrix of the *discrete Fourier transform* (*DFT*) with $n$ sample points, that is, $(F_n)_{i,j} = \omega_n^{ij}$ for $0 \le i, j < n$, where $\omega_n = e^{2\pi\sqrt{-1}/n}$. Then, using the convolution theorem [2] with DFT,

$$U = F_l^{-1}\left(F_l S \circ F_l T\right), \tag{6}$$

where $\circ$ is the operator of the Hadamard product.

Thus, the basic algorithm to compute $C(s,t)$ is summarized as follows:

1. Convert $s$ and $t$ to $S$ and $T$, respectively;
2. Compute $F_l S$ and $F_l T$ using $2\sigma$ FFTs;
3. Compute $X = F_l S \circ F_l T$ using element-wise multiplications;
4. Compute $U = F_l^{-1} X$ using $\sigma$ FFTs; and
5. Compute $C(s,t)$ from $U$ using element-wise additions.

The processing time of the algorithm is $O(\sigma l \log l)$, which leads to $O(\sigma n \log n)$. Process 1 needs $O(l)$ evaluations of $\phi$, where an evaluation needs $O(\log \sigma)$ time. Process 2 consists of $2\sigma$ FFTs, where an FFT needs $O(l \log l)$ time. Process 3 needs $O(\sigma l)$ multiplications. Process 4 needs $\sigma$ FFTs. Process 5 needs $O(\sigma l)$ additions. Therefore, the total processing time is bound by $O(\sigma l \log l)$.