A Lower-Variance Randomized Algorithm for Approximate String Matching[*]

Mikhail J. Atallah Elena Grigorescu
Yi Wu
Department of Computer Science
Purdue University
West Lafayette, IN 47907
U.S.A.
{mja,egrigore,wu510}@cs.purdue.edu

Abstract

Several randomized algorithms make use of convolution to estimate the score vector of matches between a text string of length N and a pattern string of length M, i.e., the vector obtained when the pattern is slid along the text, and the number of matches is counted for each position. These algorithms run in deterministic time $O(kN \log M)$, and find an unbiased estimator of the scores whose variance is $(M-c)^2/k$ where c is the actual score; here k is an adjustable parameter that provides a tradeoff between computation time and lower variance. This paper provides an algorithm that also runs in in deterministic time $O(kN \log M)$ but achieves a lower variance of $\min(M/k, M-c)(M-c)/k$. For all score values c that are less than $M-(M/k)$, our variance is essentially a factor of k smaller than in previous work, and for $M-(M/k) < c \leq M$ it matches the previous work's variance. As in the previous work, our estimator is unbiased, and we make no assumption about the probabilistic characteristics of the input, or about the size of the alphabet, and our solution extends to string matching with classes, class complements, "never match" and "always match" symbols, to the weighted case and to higher dimensions.

[*]Portions of this work were supported by National Science Foundation Grants CNS-0915436, CNS-0913875, Science and Technology Center CCF-0939370; by an NPRP grant from the Qatar National Research Fund; by Grant FA9550-09-1-0223 from the Air Force Office of Scientific Research; and by sponsors of the Center for Education and Research in Information Assurance and Security. The statements made herein are solely the responsibility of the authors.

Keywords: convolution, FFT, approximate string matching, randomized algorithms.

1 Introduction

Given a text string $T = t_0 t_1 \dots t_{N-1}$ and a pattern string $P = p_0 p_1 \dots p_{M-1}$ over an alphabet Σ of size σ, we address the problem of computing the score vector of matches between T and P. This is defined as the vector C whose ith component c_i is the number of matches between the text and the pattern when the first letter of the pattern is positioned in front of the ith letter of the string.

We avoid repeating the discussion of the applicability of the above problems, which is amply discussed in the cited references and elsewhere in the literature.

As in [2, 10, 4], we seek a randomized algorithm of Monte-Carlo type to compute an unbiased estimate of the score vector, and as in the previous work we make crucial use of convolution, specifically, of the well-known fact that by doing N/M convolutions one can compute all the $N - M$ scalar products between a vector of length M and all contiguous length-M subvectors of another vector of length N (see ,e.g., [1, 9]).

Also as in the previous work, the algorithm we propose here proceeds by computing and averaging k independent equally distributed estimates for the score vector, and has the property that the expected value of the averaged estimator is equal to the exact value: The expected value of the ith component $\widehat{c_i}$ of the estimate $\widehat{C}$ of the score vector equals c_i. Our improvement is that the variance is now $\min(M/k, M-c)(M-c)/k$, whereas it was $(M-c_i)^2/k$ in the previous work, k being a parameter that can be tuned to achieve a desired level of accuracy. The time complexity is the same as in the previous work: $O(kN\gamma(M)/M)$, where $\gamma(M)$ is the time needed to perform the convolution of two vectors of length M. As is commonly done, we replace $\gamma(M)$ by $M \log M$, which corresponds to the computational model where an arithmetic operation takes constant time. This makes the time complexity $O(kN \log M)$. The following theorem summarizes our main result.

Theorem 1. *An estimate for the score C between a text string of length N and a pattern string of length M can be computed by a Monte-Carlo algorithm in time $O(kN \log M)$, where k is the number of iterations in the algorithm. The randomized result has mean C and each entry has a variance bounded by* $\min(M/k, M-c)(M-c)/k$.

2 Previous Work and New Contribution

The problem of computing the score vectors has a long history, in which the goal has always been to avoid the naive brute-force $O(MN)$ solution. Fischer and Paterson used convolution to solve the special case of finding all exact occurrences (i.e., scores that equal exactly M) in the presence of "always match" symbols [7], for the case of constant-size alphabet. Abrahamson and Kosaraju independently extended the algorithm by Fischer and Paterson into a deterministic algorithm for computing the vector C in time complexity of $O(\min(|\Sigma|, \sqrt{M/\log M})\ N \log M)$ [1, 9] for any alphabet size. An algorithm of Baeza-Yates and Gonnet, that works remarkably well in practice, solves the problem in time $O(NM \log M/ \log N)$ [5], which is better than $O(N \log M)$ for very small M, i.e., $M = o(\log N)$. Another algorithm, by Baeza-Yates and Perleberg, solves the problem in time $O(NMf_{\max})$ where $f_{\max}$ is the maximal occurrence frequency of symbols in the pattern [6].

An algorithm of deterministic time $O(N \log M)$ was given in [3], whose analysis depends on some restrictive assumptions on the probabilistic characteristics of the input, namely the Bernoulli model. Karloff studied the case when the alphabet is not known beforehand and gave a deterministic $O(N \log^3 M)$ time algorithm for estimating all the scores of mismatches [8], and also provided a randomized variant of deterministic time complexity $O(N \log^2 M)$. Karloff's estimator is intentionally biased in order to guarantee not to overestimate the number of mismatches by more than a constant multiplicative factor. The method apparently cannot be modified to estimate the number of matches (rather than of mismatches).

The work that is closest to this paper is in [2, 10, 4]. In these as in this paper, the algorithm runs in deterministic time $O(kN \log M)$. The main difference is that in the present paper the variance goes down by a factor of essentially k, for all values of a score that are $\leq M-(M/k)$ (for score values that are greater the variance bound is the same as in the previous work).

The algorithm in [2] was based on the idea of mapping the alphabet symbols into the primitive σth roots of unity, and then using convolution, relying on the nullity of the sum of all the σth roots of unity for its unbiasted-estimator property. The process was repeated k times, each with a different random mapping, and the average of the k results was taken to be the answer. In the follow-on work in [10, 4] different mappings were used. The scheme in [10] is particularly well suited for small alphabet sizes (≤ 20).

The main new idea here is to do a judicious pre-processing step that also takes $O(kN \log M)$ time, and then to prove that it results in the claimed improved variance. That is the main new ingredient, and in fact What follows the pre-processing step could be any of the schemes in [2, 10, 4], although here we chose a different variant for simplicity (of both the presentation and the variance proof). The next sections present the algorithm and analyze its properties.

3 The Algorithm

The steps of the algorithm are given below, where we assume that the symbols in T that do not occur in P have been replaced by a special "match nothing" symbol #; this means that the reduced alphabet (which we still denote Σ) has a size σ that is at most the length of M plus one.

Score-Estimation Algorithm:

INPUT: a text $T = t_0 \dots t_{N-1}$; a pattern $P = p_0 \dots p_{M-1}$ where the t_i's and the p_i's are letters from Σ; an integer k that determines the desired tradeoff between time and accuracy.

OUTPUT: an estimate for the score vector C.

1. Compute in time $O(M + \sigma \log \sigma)$ the number of occurrences in P of every symbol of Σ, and let Σ' denote the subset of Σ whose symbols occur more than M/k times in P. Note that $|\Sigma'| \leq k$.

2. For every symbol s in Σ', compute the exact contribution of symbol s to the score vector C. It is well known that this can done with N/M convolutions for each symbol, hence in $O(N \log M)$ time per symbol (see, e.g., [1, 9]). As this is done for all the symbols of Σ', it takes a total of $O(|\Sigma'| N \log M)$ time. The rest of the algorithm deals with estimating the contribution of the symbols in $\Sigma - \Sigma'$.

   Note. As stated earlier, the rest of the agorithm could have used any of the schemes in [2, 10, 4]. The variant we chose is convenient for the proof of the lower variance.

3. For $\ell = 1, 2, \cdots, k$:

   (a) From the text T and pattern P, obtain vectors T_ℓ and (respectively) P_ℓ as follows. First, replace in T and P every occurrence of a symbol from $\Sigma' \cup \{\#\}$ with a 0. Then, for each symbol $s \in \Sigma - \Sigma' - \{\#\}$, flip a fair coin and, if its outcome is head (tail), replace in T and P every occurrence of s with a $+1$ (resp., -1).

(b) Compute the vector C_ℓ such that, for $0 \leq i \leq N - M - 1$, $C_\ell[i]$ is the scalar product of P_ℓ with the sub-vector of T_ℓ of length M that starts at position i in T_ℓ; in other words:

$$C_\ell[i] = \sum_{j=0}^{M-1} P_\ell[j] * T_\ell[i+j]$$

Implementation note: This can be done using N/M convolutions, where the ith such convolution is between the following two vectors of size $2M$ each: (i) the reverse of P_ℓ padded with M zeroes; and (ii) the sub-vector of T_ℓ of length $2M$ that starts at position iM in T_ℓ.

4. Compute the vector $\widehat{C} = \sum_{\ell=1}^{k} C^\ell / k$ as the estimate of the contribution to C of symbols in $\Sigma - \Sigma'$. Adding to this the contribution of symbols in Σ' as computed in Step 2 gives the desired estimate of C.

4 Analysis of the Output Estimate

That the estimator of a c_i for that particular iteration is unbiased is obvious, because each one of the M terms of the summation that it makes up has zero mean if the two symbols are different and 1 if they are same. So we henceforth focus on the variance.

We first analyze the variance for one of the k iterations, and then divide by k to obtain the variance for their average. The variance is the sum of the M^2 entries of the $M \times M$ covariance matrix for the M terms of the summation that makes up a this iteration's estimate of c_i. For a term that corresponds to equal symbols, the corresponding row and column of the covariance matrix are zero, and because there are c_i such terms the sub-matrix that contributes to nonzero entries is no larger than $(M - c_i) \times (M - c_i)$ (it can be smaller because of the symbols from Σ', whose involvement in any of the terms causes the value of that term to be zero, and its row and column to also be all zeroes in the covariance matrix). A nonzero entry of the covariance sub-matrix contains a 1, and has a row (resp., column) that is a term whose corresponding 2 symbols are unequal and are both from $\Sigma - \Sigma'$. For any such term (say, arising out of the alignment of distinct symbols a and b), its covariance with any other such term is zero unless the other term also involves the same two symbols a and b. But the number of such repetitions of symbols a and b is bounded by M/k (by its definition $\Sigma - \Sigma'$ excludes symbols that occur more than M/k times in P). This means that no such term can have more than M/k nonzero covariances with other terms, whence the upper bound of $(M - c_i)(M/k)$ on the number of nonzero entries of the covariance matrix. Of course $(M - c_i)^2$ is also an upper bound, and combining these two gives us the desired $\min(M/k, M - c_i)(M - c_i)$ bound on the variance of the estimator of c_i for that iteration. Therefore the variance of the overall estimator (the average over the k iterations) is $\min(M/k, M - c_i)(M - c_i)/k$.

References

[1] Abrahamson, K. Generalized string matching. *SIAM J. Comput. 16*, 6 (1987), 1039–1051.

[2] Atallah, M. J., Chyzak, F., and Dumas, P. A randomized algorithm for approximate string matching. *Algorithmica 29*, 3 (2001), 468–486.

[3] Atallah, M. J., Jacquet, P., and Szpankowski, W. Pattern matching with mismatches: a simple randomized algorithm and its analysis. In *Combinatorial Pattern Matching (Proceedings of the Third Annual Symposium held in Tucson, Arizona, April 29–May 1, 1992)* (Berlin, 1992), A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, Eds., vol. 644 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 27–40.

[4] Baba, K., Shinohara, A., Takeda, M., Inenaga, S., and Arikawa, S. A note on randomized algorithm for string matching with mismatches. *Nordic J. of Computing 10*, 1 (Mar. 2003), 2–12.

[5] Baeza-Yates, R. A., and Gonnet, G. H. A new approach to text searching. *Comm. ACM 35* (1992), 74–82.

[6] Baeza-Yates, R. A., and Perleberg, C. H. Fast and practical approximate string matching. *Inform. Process. Lett. 59*, 1 (1996), 21–27.

[7] Fischer, M. J., and Paterson, M. S. String-matching and other products. In *Complexity of computation (Proc. SIAM-AMS Appl. Math. Sympos., New York, 1973)* (Providence, R. I., 1974), Amer. Math. Soc., pp. 113–125. SIAM–AMS Proc., Vol. VII.

[8] Karloff, H. Fast algorithms for approximately counting mismatches. *Inform. Process. Lett. 48*, 2 (1993), 53–60.

[9] Kosaraju, S. R. Efficient string matching. Manuscript, Johns Hopkins University, 1987.

[10] Schoenmeyr, T., and Zhang, D. Y. Fft-based algorithms for the string matching with mismatches problem. *J. Algorithms 57*, 2 (Nov. 2005), 130–139.