A Black Box for Online Approximate Pattern Matching

Raphaël Clifford[1], Klim Efremenko[2], Benny Porat[3], and Ely Porat[3]

[1] University of Bristol, Dept. of Computer Science, Bristol, BS8 1UB, UK
clifford@cs.bris.ac.uk
[2] Bar-Ilan University, Dept. of Computer Science, 52900 Ramat-Gan and Weizman Institute, Dept. of Computer Science and Applied Mathematics, Rehovot, Israel
klimefrem@gmail.com
[3] Bar-Ilan University, Dept. of Computer Science, 52900 Ramat-Gan, Israel
bennyporat@gmail.com, porately@cs.biu.ac.il

Abstract. We present a deterministic black box solution for online approximate matching. Given a pattern of length m and a streaming text of length n that arrives one character at a time, the task is to report the distance between the pattern and a sliding window of the text as soon as the new character arrives. Our solution requires $O(\Sigma_{j=1}^{\log_2 m} T(n, 2^{j-1})/n)$ time for each input character, where $T(n, m)$ is the total running time of the best offline algorithm. The types of approximation that are supported include exact matching with wildcards, matching under the Hamming norm, approximating the Hamming norm, k-mismatch and numerical measures such as the L_2 and L_1 norms. For these examples, the resulting online algorithms take $O(\log^2 m)$, $O(\sqrt{m \log m})$, $O(\log^2 m/\epsilon^2)$, $O(\sqrt{k \log k} \log m)$, $O(\log^2 m)$ and $O(\sqrt{m \log m})$ time per character respectively. The space overhead is $O(m)$ which we show is optimal.

1 Introduction

Fast approximate string matching is a central problem of modern data intensive applications. Its applications are many and varied, from computational biology and large scale web searching to searching multimedia databases and digital libraries. As a result, string matching has to continuously adapt itself to the problem at hand. Simultaneously, the need for asymptotically fast algorithms grows every year with the explosion of data available in digital form.

A great deal of progress has been made in finding fast algorithms for a variety of important forms of approximate matching. One of the most studied is the Hamming distance which measures the number of mismatches between two strings. Given a text t of length n and a pattern p of length m, the task is to report the Hamming distance at every possible alignment. $O(n\sqrt{m \log m})$ time solutions based on repeated applications of the FFT were given independently by both Abrahamson and Kosaraju in 1987 [1, 17]. Particular interest has been paid to a bounded version of this problem called the k-mismatch problem. Here a bound k is given and we need only report the Hamming distance if it is less

P. Ferragina and G. Landau (Eds.): CPM 2008, LNCS 5029, pp. 143–151, 2008.

than or equal to k. If the number of mismatches is greater than the bound, the algorithm need only report that fact and not give the actual Hamming distance. In 1985 Landau and Vishkin gave a beautiful $O(nk)$ algorithm that is not FFT based which uses constant time LCA operations on the suffix tree of p and t [18]. This was subsequently improved to $O(n\sqrt{k \log k})$ time by a method based on filtering and FFTs again [4]. Approximations within a multiplicative factor of $(1+\epsilon)$ to the Hamming distance can also be found in $O(n/\epsilon^2 \log m)$ time [15].

The problem of determining the time complexity of exact matching with don't cares has also been well studied over many years [13, 15, 16, 12, 6], culminating in two related deterministic $O(n \log m)$ time solutions. This has been accompanied by recent advances for the problem of k-mismatch problem with don't cares [10, 9] as well as a surge in interest in provably fast algorithms for distance calculation and approximate matching between numerical strings. Many different metrics have been considered, with for example the L_1 distance [5, 7, 3] and less-than matching [2] problems both being solvable in $O(n\sqrt{m \log m})$ time and a bounded version of the L_∞ norm which was first discussed in [8] and then improved in [7, 19] requiring $O(\delta n \log m)$ time.

In almost every one of these cases and in many others beside, the algorithms make extensive use of the fast Fourier transform (FFT). The property of the FFT that is required is that in the RAM model, the cross-correlation,

$$(t \otimes p)[i] \stackrel{\text{def}}{=} \sum_{j=1}^{m} p_j t_{i+j-1}, \quad 0 \le i \le n-m+1,$$

can be calculated accurately and efficiently in $O(n \log n)$ time (see e.g. [11], Chapter 32). By a standard trick of splitting the text into overlapping substrings of length $2m$, the running time can be further reduced to $O(n \log m)$.

Although the FFT is a very powerful and successful tool, it also brings with it a number of disadvantages. Perhaps most significant of these in this context is that the cross-correlation computation using the FFT is very much an offline algorithm. It requires the entire pattern and text to be available before any search can be performed. Of course, it is not only the FFT that causes this difficulty. For example, the only fast algorithm for the k-mismatch algorithm which does not employ the FFT uses constant time LCA queries [18]. As it is not known how to perform the necessary preprocessing of a suffix tree to compute the LCA online, this algorithm suffers from the same limitations as those that depend on the FFT.

In many situations such as when monitoring Internet traffic or telecommunications networks this model of computation may not be feasible. It is not sufficient simply that a pattern matching algorithm runs fast. It should also require considerably less space than the input and update at least as quickly as the new data are arriving while still maintaining an overall time complexity which is as close as possible to the full offline algorithm. One approach to handle this situation is the data streaming model where it is assumed that it is not possible to ever store all the data seen and in some variants that only one pass over the data is ever allowed. This very successful model has been the source of a great