Fast Approximate Matching of Words Against a Dictionary

H. Bunke, Bern

Received June 28, 1994

Abstract — Zusammenfassung

Fast Approximate Matching of Words Against a Dictionary. A new algorithm for string edit distance computation is given. The algorithm assumes that one of the two strings to be compared is a dictionary entry that is known a priori. This dictionary word is converted in an off-line phase into a deterministic finite state automaton. Given an input string and the automaton derived from the dictionary word, the computation of the edit distance between the two strings corresponds to a traversal of the states of the automaton. This procedure needs time which is only linear in the length of the input string. It is independent of the length of the dictionary word. Given not only one but \( N \) different dictionary words, their corresponding automata can be combined into a single deterministic finite state automaton. Thus the computation of the edit distance between the input word and each dictionary entry, and the determination of the nearest neighbor in the dictionary need time that is only linear in the length of the input string. However, the number of states of the automaton is exponential.

AMS Subject Classifications: 68Q20, 90C39

Key words: String edit distance, finite state automaton, nearest neighbor search, dictionary lookup.


1. Introduction

String matching is a problem that constantly receives attention in different areas of science and engineering. In computer sciences, for example, there are problems like program error correction, file difference checking, or information retrieval that require the comparison of strings of symbols [1]. Spelling correction in natural language text editing is another application that is based on the
comparison of strings of symbols [2]. In optical character recognition, for example, the results of a character classification program are usually matched to a dictionary of legal words in order to improve the recognition performance of the overall system [3, 4]. Other applications of string matching include the recognition of two-dimensional shapes [5], the computer analysis and understanding of human speech and sound waves, and molecular biology [6].

There are different classes of string matching problems. In this paper we focus on approximate string matching, i.e., on string distance computation based on a set of edit operations [7]. We are interested in measuring the similarity of pairs of strings via the minimum cost sequence of edit operations needed to transform one string into the other. The first algorithm for computing the distance, or similarity, of two strings was proposed long ago [8]. Subsequently, other authors have studied the string distance computation problem [9]. Generally, one introduces a set of basic edit operations with costs associated with each edit operation and defines the distance of two strings as the minimum cost sequence of edit operations needed to transform one string into the other. The algorithm of Wagner and Fischer [9] is usually referred to as the standard solution to the problem. It is based on dynamic programming and has a time complexity of \(O(n \cdot m)\), where \(n\) and \(m\) give the lengths of the two strings to be compared.

It has also been shown that string distance is closely related with the problem of finding the longest common subsequence of two strings. For this problem, a number of algorithms with a lower computational complexity than that of Wagner & Fischer's algorithm have been published [10–12]. The asymptotically fastest algorithm for the original string distance problem is that of Masek and Paterson [13], having a time complexity of \(O(n^2/\log n)\), where \(n\) denotes the lengths of the two strings to be compared. A recent survey on algorithms for string matching has been given in [14]. Additional approaches to approximate string matching have been described in [15–18].

In this paper a new algorithm for string edit distance computation is proposed. It is based on Wagner and Fischer's algorithm [9]. However, while in [9] the two strings to be compared may be given online, our algorithm assumes that one of the two strings to be compared is a dictionary entry that is known a priori. This dictionary word is converted in an off-line phase into a deterministic finite state automaton. Now, given an input string and the automaton derived from the dictionary word, the computation of the edit distance between the two strings corresponds to a traversal of the state of the automaton. This procedure needs time which is only linear in the length of the input string. It is independent of the length of the dictionary word. Given not only one but \(N\) different dictionary words, their corresponding automata can be combined into a single deterministic finite state automaton. Thus the computation of the edit distance between the input word and each dictionary entry, and the determination of the nearest neighbor in the dictionary need time that is only linear in the length of the input string. Particularly, the time complexity is independent of the number and the