Data Compression Using Encrypted Text

Robert W. Franceschini  Amar Mukherjee
francesc@cs.ucf.edu amar@cs.ucf.edu

Department of Computer Science
University of Central Florida
Orlando, Florida 32816

Abstract

In this paper, we present a new algorithm for text compression that exploits the properties of the words in a dictionary to produce an encryption of given text. The basic idea of our algorithm is to define a unique encryption or signature of each word in the dictionary by replacing certain characters in the words by a special character ‘s’ and retain a few characters so that the word is still retrievable. The question is whether we can develop a better signature of the text before compression so that the compressed signature uses less storage than the original compressed text. This indeed is possible as our experimental results confirm. For any cryptic text the most frequently used character is ‘s’ and the standard compression algorithms can effectively exploit this redundancy in an effective way. Our algorithm produces the best lossless compression rate reported to date in the literature: it beats the widely used UNIX "compress" method, which is based on the LZW algorithm, by about 7% and GNU-zip by about 5%. We have observed compression rates as high as 75% for some sample texts using this method. Considering the fact that GNU-zip produces about 1.3% improvement in compression rate over UNIX compress, our algorithm produces a compression improvement over GNU-zip comparable to three times that of GNU-zip over UNIX compress. On average, our compression rate translates into 68% which is equivalent to about 2.56 bits/character. One basic assumption of our algorithm is that the system has access to a dictionary of words used in all the texts along with a corresponding "cryptic" dictionary. The cost of this dictionary is amortized over the compression savings for all the text files handled by the organization. If two organizations wish to exchange information using our compression algorithm, they must share a common dictionary. We used ten text files from the English text domain to test our algorithm. These ten text files were based on publicly available electronic English novels, obtained from the World Wide Web.