Chapter 6
Fault–Tolerant Text Data Compression Algorithms

L. Robert
Government Arts College, India

R. Nadarajan
PSG College of Technology, India

ABSTRACT
There has been an unparalleled explosion of textual information flow over the internet through electronic mail, web browsing, digital library and information retrieval systems, etc. Since there is a persistent increase in the amount of data that needs to be transmitted or archived, the importance of data compression is likely to increase in the near future. Virtually, all modern compression methods are adaptive models and generate variable-bit-length codes that must be decoded sequentially from beginning to end. If there is any error during transmission, the entire file cannot be retrieved safely. In this article we propose few fault-tolerant methods of text compression that facilitate decoding to begin with any part of compressed file not necessarily from the beginning. If any sequence of one or more bytes is changed during transmission of compressed file due to various reasons, the remaining data can be retrieved safely. These algorithms also support reversible decompression.

INTRODUCTION
The data compression field has always been an important part of computer science and it is becoming increasingly popular and important today. Although computers become faster and data storage becomes less expensive and more efficient, the increased importance of usage of data necessitates the use of at least a small measure of data compression due to vast storage and transmission requirements. The question in many applications is no longer whether to compress data, but what compression method should be applied. Many modern compression methods are
adaptive models and generate variable-bit-length codes (Maxime & Thierry, 1999). The Huffman algorithm (Huffman, 1952; Knuth, 1985) uses the notion of variable length prefix code for replacing characters of the text. The code depends on the input text, and more precisely on the frequencies of characters in the text. The most frequent characters are given shortest code words, while the least frequent symbols correspond to the longest code words. The basic idea of Arithmetic Coding (Moffat, Alistair, Witten, & Neal, 1998) is to consider symbols as digits of a numeration system, and texts as decimal parts of numbers between 0 and 1. The length of the interval attributed to a digit is made proportional to the frequency of the digit in the text. Ziv and Lempel (1977) designed a compression method using encoding segments. These segments of the original text are stored in a dictionary that is built during the compression process. In this model where portions of the text are replaced by pointers on previous occurrences, the Ziv-Lempel compression can be proved to be asymptotically optimal. LZW (Welch, 1984) is another popular variant of Ziv-Lempel compression which eliminates a second field of a token. Chu (2002) describes a method derived from LZ family called LZAC. The objective of LZAC is to improve the compression ratios of the LZ77 family still retaining the family’s key characteristics: simple, universal, fast in decoding, and economical in memory. LZAC presents new ideas of composite fixed-variable-length coding and offset difference coding. Albert Apostolico (2005, 2006) describes variants of classical data compression paradigms by Ziv, Lempel and Welch. The phrases used for compression are strings of intermittently solid and “don’t care” characters produced in a deterministic fashion, by certain autocorrelations of the source string generated by the very mechanics of parsing. Joaquín Adiego et al. (2004) describe a new LZ based approach called LZCS in which the main idea is the replacement of frequently repeated subtrees by a backward reference to their first occurrence.

Michael Burrows and David Wheeler (1994) released the details of a transformation function that opens the door to some revolutionary new compression techniques. The Burrows-Wheeler Transformation (BWT) transforms a block of data into a format that is extremely well suited for compression. It takes a block of data and rearranges it using a sorting algorithm. The resulting output block contains exactly the same data elements that it started with, differing only in their ordering. The transformation is reversible. M. Hosang (2002) describes a variant of BWT, character elimination algorithm for lossless data compression intended for use on files with non-uniform character distributions. This algorithm takes advantage of the relatively small distances between character occurrences after the removal of less frequent characters.

J. Cleary and I. Witten (1984) developed Prediction by Partial Matching (PPM) which is an adaptive statistical data compression technique based on context modeling and prediction. PPM models use a set of previous symbols in the uncompressed symbol stream to predict the next symbol in the stream. PPM is capable of very high compression rates, encoding English text in as little as 2.2 b/character. Moffat (1990) describes a variant of PPM which encodes and decodes at over 4 kbps on a small workstation and operates within a few hundred kilobytes of data space, but still obtaining compression of about 2.4 b/character for English text. Eibe Frank et al. (2000) describe text categorization using compression models. Text categorization is the assignment of natural language texts to predefined categories based on their content. It has been observed that compression seems to provide a very promising approach to categorization.

Drinic and Kirovski (2002) present PPMexe - a set of compression mechanisms for executables that explore their syntax and semantics to achieve superior compression rates. The fundamental principle of PPMexe is the generic paradigm of prediction by partial matching. They combine