A fast decoding method for prefix codes

Ruy L Milidiú, Eduardo S Laber, Lorenza O Moreno, Julio C Duarte

{milidiu, laber, lorenza, duartejc}@inf.puc-rio.br

PUC-Rio, Departamento de Informática

Prefix codes exert a major role in well known compression schemes. They allow a text to be decoded without ambiguity, since they are variable-length codes where no codeword is a prefix of other. The problem of improving their decoding speed has received a special attention in the data compression community. Among the several techniques that have been proposed, Table Look-up is one of the most efficient. It avoids bit manipulation using a table of size 2^k, where k is the maximum codeword length. Although very fast, this decoding method requires large additional memory since its space complexity is exponential in the length of the largest codeword.

Here, we propose a scheme that employs length-restricted codes to generate the codewords and table look-up to decode them. It has the benefit that the maximum length L of a codeword can be controlled, what avoids the exponential growth of the look-up table. Let n be the size of the input alphabet. If we set L to be $\log n + \alpha$, the amount of memory required to store the look-up table is given by

$$2^L = 2^{\log n+\alpha} = 2^{\log n}.2^{\alpha} = 2^{\alpha}n.$$

If α is constant, the space needs are linear in n. A potential drawback of this approach is that length-restricted coding implies on some compression loss. However, this is very small in practice.

In order to reduce the bit manipulation, we introduce lexical expansion. By using lexical expansion, we join consecutive symbols and their codewords, creating a new symbol and updating the respective entry in the look-up table. Hence, we are able to decode more than one symbol in a single decoding step.

In our experiments, by setting α to 1, the observed average compression loss is about 17% when the look-up table size is taking into account. The observed average decoding speed is 2.1 times faster than the one for canonical codes using the same model, what makes this method very suitable for applications where some compression and extremely fast decoding speed are mandatory.

These combined techniques yield a time and space efficient algorithm for decoding prefix codes when the size of the vocabulary is large.