Lempel-Zev-Welch Compression Algorithm

Sasank Chilamkurthy, Tharun Kumar Reddy, Varun Bairaboina

25 April 2014

1 Introduction

This is a source coding algorithm or in other words, lossless compression algorithm. This is a dictionary based algorithm. It stores previously appeared string patterns in a dictionary. If a pattern in dictionary appears again, we transmit the index of the entry instead of the entire pattern. This is how compression is achieved. We implemented a very basic version of this algorithm. Many variants of lz algorithm are available which incorporate various optimisations in dictionary management etc.

2 Encoding

2.1 High Level Description

1. Maintain a list of substrings appeared before in dictionary.
2. Store the output string from the source in a buffer.
3. Check if the present string at the buffer is there in the dictionary.
    - If yes, then wait for one more symbol to come into the buffer and then go back to step 2.
    - If no, then
        (a) Find the substring (buffer string excluding the last symbol) in the dictionary and transmit its index.
        (b) Transmit the last symbol.
        (c) Empty the buffer.

2.2 Implementation details

- Dictionary here requires searching by substrings. So we implemented dictionary as a `hashMap<String>` which has strings as its keys and integer indices as its values
- We will not transmit integer index in base 10 or 2 because that will be waste of character space as ascii characters can take values in range 0-127. So, we transmit index in base 128 system by converting it to corresponding character sequence.

3 Decoding

Decoding is simmilar to encoding.

1. Maintain a list of substrings appeared before in dictionary.
2. Store the input string in buffer. Fill buffer with incoming characters until it is of expected length. We calculate expected length from current dictionary size.
3. Convert buffer[0:length-2] to integer using base 128 system. This was the index transmitted.
4. Access string at this index from dictionary and append buffer[length-1] to it and return this.

4 Analysis

Everything (decoding and encoding) is done in constant time given performance of hashmap.