Frequency encoding: an alternative to common entropy encoding techniques used in lossless data compression

Christopher W. Chunick
Draft, August 05, 2013

Abstract

There are currently only a small number of entropy encoding techniques in use today to generate a prefix code that reduces the redundancy of a message being transmitted. I am introducing a new technique that creates a codeword to be used in encoding and decoding of information specifically but not limited to the area of lossless data compression and telecommunications. The new encoding method is a variant of range encoding [1] that involves sorted frequencies of symbols instead of probabilities to determine the outputted prefix code.

The Frequency Encoding Algorithm

The algorithm can be applied to a message or string of symbols as long as the adaptive or static frequencies of all symbols or previously appearing symbols are available. A message using binary storage, referred to as M, will be used to verify the algorithm. The start placeholder, sp, is the leftmost placeholder, the finish placeholder being fp, and width of the placeholder range will be referred to respectively as w = (sp-fp+1) and range as r. The range that determines the code to output will be the difference of a bottom and a top of the width, referred to as b and t with b always required to be less than t, and r = t - b. Frequencies of each symbol in this example will be referred to as f0 and f1 with total of all frequencies being tf. The sorted symbols from lowest to highest will be referred to as s0 and s1. The example will start with a total width of 16 placeholders (2^16 possibilities), a message of 10 bytes in length, 2 symbols (c and d), a bottom value of 0, a top value of 65535, and a range of 65535 with zero always being quantitative.

w = 10
b = 0
t = 2^w -1
r = t - b

m = cccddccccc

To encode the message ‘cccddccccc’ the frequency of each symbol is required:

Symbol	Frequency
c	f0=8
d	f1=2

tf=f0+f1

The frequencies are sorted from lowest to highest:

Sorted	Symbol	Frequency
s0	d	f1
s1	c	f2

[2] entropy = -1(((2/10)*(log(2/10)/log(2))+(8/10)*(log(8/10)/log(2))))=0.7219 bits/byte

The proposed message of 10 bytes requires at least 7.219 bits of information storage to encode. Frequency encoding does not use fractions so there is precision loss which directly affects how close to entropy that the algorithm can reach.

To encode the first symbol, replace the symbol with the sorted symbol, f1 becomes s0. The total frequency is 10 which gives a probability of P(sx) = sx/tf. Starting from the leftmost placeholder, check to see if the total frequency is less than the range of the width.

b = **0**000000000000000
t = **1**111111111111111
sp = 16 (16th placeholder)
fp = 16 (16th placeholder)
w = sp - fp + 1 = 1
r = t - b = 1

If the total frequency does not fit within the range continue to move the finish placeholder over one to increase the range until tf < (r+1).

b = **0000**000000000000
t = **1111**111111111111
sp = 16 (16th placeholder)
fp = 13 (13th placeholder)
w = sp - fp + 1 = 4
tf < (r = t - b + 1 = 16)

To encode the symbol split the range up by the frequencies of s0. The first symbol is c, using s1 as a reference only the bottom of the range needs to change, if the symbol was s0 then the top changes and the bottom stays the same.

b(w) = b(w) + s0 = **0100**
t(w) = **1111**

b = 0100000000
t = 1111111111

The new total range is now: r = t - b = 57343, continue to the next symbol using the next width that fits the total frequency. The table below illustrates each symbol as it is encoded and the final output code.

Symbol	Sort	Frq	w	r(w)	t(w)	b(w)	formula	t(w)	b(w)	r(w)	Range	Top	Bot
cccddccccc	S1	8	4	15	15	0	b(w)=b(w)+s0	15	2	13	57343	65535	8192
c**c**cddccccc	S1	8	4	13	15	2	b(w)=b(w)+s0	15	4	11	49151	65535	16384
cc**c**ddccccc	S1	8	4	11	15	4	b(w)=b(w)+s0	15	6	9	40959	65535	24576
ccc**d**dccccc	S0	2	5	19	31	12	t(w)=b(w)+s0-1	13	12	1	4095	28671	24576
cccd**d**ccccc	S0	2	8	15	111	96	t(w)=b(w)+s0-1	97	96	1	511	25087	24576
cccdd**c**cccc	S1	8	11	20	783	768	b(w)=b(w)+s0	783	770	13	447	25087	24640
cccddc**c**ccc	S1	8	11	13	783	770	b(w)=b(w)+s0	783	772	11	383	25087	24704
cccddcc**c**cc	S1	8	11	11	783	772	b(w)=b(w)+s0	783	774	9	319	25087	24768
cccddccc**c**c	S1	8	12	19	1567	1548	b(w)=b(w)+s0	1567	1550	17	287	25087	24800
cccddcccc**c**	S1	8	12	19	1567	1550	b(w)=b(w)+s0	1567	1552	15	255	25087	**24832**

Encoding the particular message of 10 bytes only requires 2 bytes of storage or 12 bits, a savings of 8 bytes. Any bit beyond the width is superfluous and can be discarded. The bottom set of bits can be stored and used as the code to decode the message, minus the superfluous bits. The final binary code stored is “011000010000” . Decoding the message requires following the same steps as encoding and comparing the standard formula b(w)=b(w)+s0 to the same width of the outputted code. If it is lower than the code, then s1 is the original input, otherwise it is s0. Knowing this information you can continue along the same process as above while decoding the message.

Conclusion

Frequency Encoding provides a new entropy encoding algorithm that removes division and the need for irrational numbers from the process. Reducing the operations to additions, subtractions, and binary shifts could be useful in not requiring superscalar processors to encode and decode digital transmissions in the future. This example is limited to a binary set of symbols and further investigation is required for a larger set of symbols.

References

[1] G. N. N. Martin Presented in March 1979 to the Video & Data Recording Conference,
IBM UK Scientific Center held in Southampton July 24-27 1979. "Range encoding: an algorithm for removing redundancy from a digitised message."

[2] Shannon, Claude E. (July/October 1948). "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379–423.