



ENTROPY TECHNIQUES FOR DIGITAL TRANSMISSION OF SPEECH

J. M. Pérez-García*, A. J. Rubio-Ayuso*

ABSTRACT

The object of this work is to obtain a binary code with an important reduction of the bit rate for speech transmission purposes. This reduction must be carried out in a simple way and giving an exact reproduction of the input signal. In order to make this codification we have considered a 12 bits uniform quantizer and the signal both with and without preemphasis. A Huffman code has been obtained in both cases considering the voice as a symbol source of 4096 symbols. Due to this number of symbols a special algorithm has been implemented in order to automatically compute the Huffman codes.

INTRODUCTION

The scope of this work is to study a suitable digital coding of the human speech. There are a lot of possibilities for the discrete representation of the voice which can be classified in two wide groups: Waveform representation and Parametric representation. Parametric representation fundamentally consists on transmitting the parameters of a model for speech production. In waveform representation the signal is coded and sent after a process of sampling and quantifying. The fact of electing one or another method uses to be related to other factors as cost, quality or bit-rate.

We pretend to obtain a code for the waveform representation of the speech in such a way that an important reduction of the bit-rate is reached (versus an uniform quantizer). This reduction of bit-rate must be achieved without any loss of quality of the voice signal. The proposed method exactly reproduces the signal as in an uniform quantizer. In order to do that we must code the voice signal in a less redundant manner. On the other hand, we code both the signal amplitude and the signal derivative (pre-emphasis).

THEORETICAL BASIS

Let S be an information source with a fixed finite alphabet, $S = \{s_1, s_2, \dots, s_q\}$ and where the symbols s_i are emitted with a probability given by $P(s_i)$. The entropy $H(S)$ of the source is defined as (Ref 1)

$$H(S) = \sum_{i=1}^q P(s_i) \log_2(1/P(s_i)) \text{ bit/symbol} \quad (1)$$

A code is a relationship between sequences of symbols of S and sequences of symbols of another code alphabet $X = \{x_1, x_2, \dots, x_r\}$. A code is called block code if there is a fixed sequence of symbols of X for every symbol of S . These sequences are called code words. A block code is non-singular if all of their words are different. A non-singular code is

*University of Granada, Spain.

called univocal if all its extensions (Ref 1) are non-singular (i. e., it is univocally decodable). An univocal code is said to be instantaneous when it is possible to decode a word without needing to know the following symbol. Obviously, our information source is the output of an uniform cuantizer (a 12 bits analog to digital converter) and we are interested in finding an instantaneous code for it. The symbols of our source are the 4096 possible values for the output of the A/D converter.

$$\text{The code mean length is defined as } L = \sum_{i=1}^q P(s_i)l(s_i) \quad (2)$$

where $l(s_i)$ is the length of the code word assigned to the source symbol s_i . It can be proved (Ref 1) that the code mean length for an instantaneous code for an information source must be $L \geq H(S)$. So, we define the efficiency of the code as $\eta = H(S)/L$. An instantaneous code is said to be compact if its mean length is minimum. Normally, this minimum does not equal $H(S)$. Thus, the problem is how to find a compact code for our information source. Huffman solved this problem giving a method to systematically find a compact code for a source (Ref 2). Let us see how it works. Let us consider an information source S with probabilities P_i and assume that the symbols are ordered in such a way that $P_1 \geq P_2 \geq \dots \geq P_q$. The Huffman algorithm begins forming a reduced source S_1 by gathering the two last symbols of S in only one of S_1 . Again the symbols of S_1 must be reordered according its probabilities (the probability of the new symbol in S_1 is the sum of the probabilities of the two gathered symbols of S ; see Figure 1 for an example). After that, a new reduced source S_2 is formed by gathering the two last symbols of S_1 an so on. The number of necessary reductions is $q-2$. The last source always has two symbols. They can be simply coded by the words 0 and 1. The process continues undoing the reductions until the original source S . In the example of figure 1, S_3 is coded by adding a bit to the symbol of S_4 which emanated from two symbols of S_3 .

	S		S ₁		S ₂		S ₃		S ₄	
s ₁	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
s ₂	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
s ₃	0.1	0100	0.1	011	0.2	010	0.3	01		
s ₄	0.1	0101	0.1	0100	0.1	11				
s ₅	0.06	0110	0.1	0101						
s ₆	0.04	0111								

Figure 1

HUFFMAN CODE: ALGORITHM FOR LARGE NUMBER OF SYMBOLS

A computational problem arises when the number of symbols becomes larger. Effectively, we must compute and store the $q-2$ sets of probabilities corresponding to the $q-2$ reductions of S . So, following the original algorithm we need a memory size proportional to q^2 . For $q=4094$ we need memory for about 16×10^6 data. In order to reduce this great amount of memory, we have implemented an algorithm which only needs a memory amount proportional to q . We are going to explain the working of our algorithm with the aid of the example of

Figure 1. We use a probability matrix, $P(i)$ and a support matrix $M(i)$ which is initialized to $M(i)=0$ for all i . In the coding phase (Figure 2) we also use another matrix $C(i)$ which contains the code words symbols. To make the first reduction, S_1 , we gather together the two last symbols of S . The new symbol of S_1 is reordered and marked by adding 1 to it in the support matrix. The last symbol of S is left in its place in order to be able to reconstruct the original source in the coding phase. In Figure 2 it can be seen that every reduced source S_i has two parts. The upper one really contains the information relative to that reduction. The lower part only contains information relative to the reconstruction of the reduction S_{i-1} . These two parts are separated in the figure by an horizontal line. The coding phase begins with the last reduction. Actually, the coding process is the same before explained. The only change is in the way for obtaining the preceding reduction since the only information we have is in matrix M and matrix P . To obtain S_{i-1} from S_i we must examine the support matrix M looking for the first symbol s_j with $M(s_j) > 0$. The symbol corresponding to S_{i-1} has a probability equal to $P(s_j)$ minus the probability of the first symbol under the horizontal line. This symbol must be incorporated to the source S_{j-1} , shifting the horizontal line one position below. Also, we must make $M(s_j) = M(s_j) - 1$ and reorder the symbols. This process must be repeated and continued until arriving to the original source S .

	S		S ₁		S ₂		S ₃		S ₄	
	P	M	P	M	P	M	P	M	P	
s ₁	0.4	0	0.4	0	0.4	0	0.4	1	0.6	
s ₂	0.3	0	0.3	0	0.3	0	0.3	0	0.4	
s ₃	0.1	0	0.1	1	0.2	2	0.3	2	0.3	
s ₄	0.1	0	0.1	0	0.1	0	0.1	0	0.1	
s ₅	0.06	1	0.1	1	0.1	1	0.1	1	0.1	
s ₆	0.04	0	0.04	0	0.04	0	0.04	0	0.04	

Figure 2

EXPERIMENTAL WORK AND RESULTS

According to the concepts in the preceding Section, the first task is to compute the probability of the 4096 symbols of our information source. Hereafter, we separately explain two cases which will be referred to as "Amplitude coding" and "Increments coding".

In order to compute the probabilities of the symbols for amplitude coding, 100 spanish words from three different speakers have been sampled. A total number of about 500.000 12-bits samples have been taken, corresponding to about 50 seconds of speech. Each of the words was normalized in amplitude in such a way that we always have $A_{max} = 4 * SD$, where A_{max} is the maximal amplitude (value 2048) and SD is the Standard Deviation of the samples of the word after normalization. Thus, after normalization all the words have the same Standard Deviation and can be correctly compared and mixed. This normalization ensures that the percentage of samples saturating the A/D converter is about 0.35 % (Ref 3). With these

samples, an histogram for the amplitudes was computed and designed and then, a likelihood functions was computed by means of a regression adjust method. We have tested several curves of the type

$$y = \frac{C}{|x|^A} e^{-B|x|} \quad (3)$$

where A, B, C are constants, x is the value of the source symbols (from -2047 to 2048) and y is the corresponding statistical absolute frequency. Several values for A have been tested in the range 0 to 1. The best results were obtained for $A=1/3$. They are $B=-2.4 \times 10^{-3}$ and $C=3300$ with a correlation coefficient of $r=0.94$. We have assigned the experimental values $y(0)=6308$ (because in this point the theoretical curve is singular) and $y(2048)=y(-2047)=1767$ (to take into account the real saturation of the A/D converter). The relative probabilities and the entropy of the source can be computed giving a value of $H(S_a)=10.43$ bits/symbols (S_a stands for source of amplitudes).

The histogram for increment coding has been computed in the same way as for amplitudes but considering the difference between samples instead of the own samples. In this case, the experimental adjustment of the histogram is carried out by means of two curves:

- For $x < 79$, $A=0.25$; $B=0.01$; $C=5600$ ($r=0.72$)
- For $x > 78$, $A=1$; $B=2.4 \times 10^{-3}$; $C=81000$ ($r=0.88$)

As in the case of amplitudes coding, in order to adequate the function to the reality we have assigned the experimental values $y(0)=10589$ and $y(2048)=y(-2047)=380$. The entropy of this increment source results to be $H(S_i)=9.27$ bits/symbol.

Applying the algorithm here presented to the source for amplitudes coding we obtain a code whose first word (for $x=0$) is 110011 and the last one (for $x=2047$) is 011011101000110101. The mean length for the amplitudes code is $L_a=10.46$ bits/symbol. Thus, the efficiency of the code obtained is $\eta_a=0.9971$. By comparing the code obtained with the usual 12-bits uniform code we can say that we obtain a reduction of 12.83 % in channel occupation. For the case of increment coding we obtain a code whose first word (for $x=0$) is 000100 and the last one (for $x=2047$) is 001011001110011000101. The mean length for the increments code is $L_i=9.30$ bits/symbol. Thus, the efficiency for this code is $\eta_i=0.9968$. By comparing the code obtained with the usual 12-bits uniform code we can say that we obtain a reduction of 22.5 % in channel occupation. We must point out that this reductions of channel occupation are achieved without any degradation of the quality of the speech signal because it is reproduced exactly as the original one.

REFERENCES

1. N. Abramson "Teoría de la información y codificación" Paraninfo, Madrid 1975.
2. D. A. Huffman, "A method for the construction of minimum redundancy codes" Proc. IRE, vol 40, n. 10, pp. 1098-1101. Sept 1952.
3. L. R. Rabiner, R. W. Schafer "Digital processing of speech signals" Prentice-Hall, 1978.