WEEK 3

Source-Coding Theorem

An important problem in communications is the *efficient* representation of data generated by a discrete source. The process by which this representation is accomplished is called *source encoding*. The device that performs the representation is called a *source encoder*. For the source encoder to be *efficient*, we require knowledge of the statistics of the source. In particular, if some source symbols are known to be more probable than others, then we may exploit this feature in the generation of a *source code* by assigning *short* code words to *frequent* source symbols, and *long* code words to *rare* source symbols. We refer to such a source code as a *variable-length code*. The *Morse code* is an example of a variable- length code. In the Morse code, the letters of the alphabet and numerals are encoded into streams of *marks* and *spaces*, denoted as dots and dashes, respectively. In the English language, the letter *E* occurs more frequently than the letter Q, for example, so the Morse code encodes *E* into a single dot " . " the shortest code word in the code, and it encodes Q into "--.-- the longest code word in the code.

Fixed-Length and Variable-Length Codes

A decision that must be made very early in the design of a code is whether to represent all symbols with codes of the same number of bits (fixed length) or to let some symbols use shorter codes than others (variable length). There are advantages to both schemes. Fixed-length codes are usually easier to deal with because both the coder and decoder know in advance how many bits are involved, and it is only a matter of setting or reading the values. With variable-length codes, the decoder must use the values of the bits to determine when the symbol is known and therefore when to start interpreting the next bit as the beginning of the next coded symbol.

Variable-length codes are useful if some symbols occur more frequently than others. The idea is to assign short sequences to common symbols, and long sequences to uncommon symbols, thereby achieving shorter coded messages on average. For example, not all letters in the English alphabet are equally common; the letters X, J, Q, and Z are quite rarely used. Here is a table of the frequency of the letters in written English (the number of times each letter is, on average, found per 1000 letters):

132	E	61	S	24	U
104	T	53	H	20	G, P, Y
82	A	38	D	19	W
80	O	34	L	14	B
71	N	29	F	9	V
68	R	27	C	4	K
63	I	25	M	1	X, J, Q, Z

Variable-length codes intended for English text can achieve greater efficiency by choosing a short bit pattern for the letters such as E, T, A, O, and N, and letting the less commonly used letters have longer patterns. Of course the specific patterns used must be chosen "prefix-free" in the sense that a short code cannot be the same as the first part of any longer code – otherwise the decoder would not be able to tell them apart. In practice the benefit gained has not, in the case of text, been sufficient to justify the greater complexity, and text is normally coded in a fixed-length code such as ASCII. However, one exception is Morse Code.

Morse Code

Morse code was designed for use on telegraphs, and it later saw use in radio communications before AM radios could carry voice. Until 1999 it was a required mode of communication for ocean vessels, even though it was rarely used (the theory apparently was that some older craft might not have converted to more modern communications gear). Ability to send and receive Morse code is still a requirement for U.S. citizens who want a radio amateur license.

Morse code is named after Samuel F. B. Morse, the inventor of the telegraph, who is credited with its modern form. It consists of a sequence of short and long pulses or tones (dots and dashes) separated by short periods of silence. A person generates Morse code by making and breaking an electrical connection on a "key," and the person on the other end of the line listens to the sequence of dots and dashes and converts them to letters, spaces, and punctuation.

If the duration of a dot is taken to be one unit then that of a dash is three units. The space between the dots and dashes within one character is one unit, that between characters is three units, and that between words seven units. Space is not considered a character, as it is in ASCII. A comparison of these two charts shows that Morse did a fairly good job of assigning short sequences to the more common letters. It is reported that he did this not by consulting books and newspapers and counting letters, but by visiting a print shop. The printing presses at the time used movable type, with separate letters assembled by the printer into lines. Each letter was available in multiple copies

for each font and size, in the form of pieces of lead. Morse simply counted the pieces of type available for each letter of the alphabet, assuming that the printers knew their business and stocked their cases with the right quantity of each letter. The wooden type cases were arranged with two rows, the capital letters in the upper one and small letters in the lower one. Printers referred to those from the upper row of the case as "uppercase" letters.

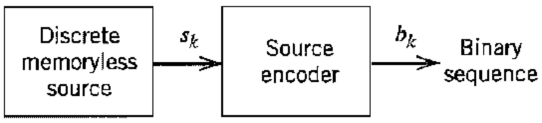
A	·—	K	—·—	U	··—	0	—————			Period	·—·—·—
B	—···	L	·—··	V	···—	1	·————			Hyphen	—····—
C	—·—·	M	——	W	·——	2	··———			Comma	——··——
D	—··	N	—·	X	—··—	3	···——			Colon	———···
E	·	O	———	Y	—·——	4	····—			Question Mark	··——··
F	··—·	P	·——·	Z	——··	5	·····			Apostrophe	·————·
G	——·	Q	——·—			6	—····			Parenthesis	—·——·—
H	····	R	·—·			7	——···			Quotation mark	·—··—·
I	··	S	···			8	———··			Fraction bar	—··—·
J	·———	T	—			9	————·			Delete prior word	·····
										End of Transmission	·—·—·

Our primary interest is in the development of an efficient source encoder that satisfies two functional requirements:

- ✓ The code words produced by the encoder are in *binary* form.
- ✓ The source code is *uniquely decodable*, so that the original source sequence can be reconstructed perfectly from the encoded binary sequence.

Consider then the scheme shown in figure, which depicts a discrete memoryless source whose output s_k is converted by the source encoder into a block of Os and 1s, denoted by b_k. We assume that the source has an alphabet with K different symbols, and that the kth symbol s_k occurs with probability p_k, $k = 0, 1, \ldots, K-1$. Let the binary code word assigned to symbol s_k *by* the encoder have length l_k measured in bits. We define the average code-word length, L, of the source encoder as:

$$L = \sum_{k=1}^{k-1} P_k l_k \quad bits$$



In physical terms, the parameter L represents the *average number of bits per source symbol* used in the source encoding process. Let L_{min} denote the *minimum* possible value of L We then define the *coding efficiency* of the source encoder as:

$$\eta = \frac{L_{min}}{L}$$

With $L \geq L_{min}$, we clearly have ≤ 1 . The source encoder is said to be *efficient* when $\boldsymbol{\eta}$ approaches unity. But how is the minimum value $\mathbf{L_{min}}$ determined? The answer to this fundamental question is included in Shannon's first theorem: the *source-coding theorem* which may be stated as follows:

Given a discrete memoryless source of entropy (δ) , the average code-word length L for any distortion-less source encoding scheme is bounded as

$$L \geq H(\delta)$$

Thus the coding efficiency will be:

$$\eta = \frac{H(\delta)}{L}$$