Interlude — Terminology for Coding Theory

An *alphabet* is a finite set of symbols, most commonly either $\{0,1\}$ or the ASCII character set. A *word* over an alphabet A is a finite string of symbols from A (repetitions allowed). The *length* $l(w)$ of a word w is the number of symbols in it (including repetitions: so $l(0\,0\,1\,0\,1) = 5$). There are $|A|^n$ different words of length n over A, since there are $|A|$ choices for the first symbol, for each of these there are $|A|$ choice for the second symbol, etc.

A *code* over an alphabet A is a finite set C of distinct words over A, called *codewords*. The *codemessage* corresponding to a sequence of codewords is the string obtained by concatenating these codewords; e.g., if $C = \{\overset{c_1}{0},\ \overset{c_2}{0\,1\,0},\ \overset{c_3}{0\,1\,1}\}$ then $0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0$ is the codemessage corresponding to the sequence $c_1c_3c_3c_2c_1$ of codewords, but $1\,1\,1$ is *not* a codemessage.

To *use* a code in practice, one requires also:

(1) a *source alphabet* A_s;
(2) a set of permissible *source messages* (usually, all possible words over A_s);
(3) a finite set of distinct words over A_s, called *source words* (often, all possible words over A_s of some fixed length);
(4) a procedure for dividing up an arbitrary source message into source words; and
(5) a bijection from the set of source words to the set of codewords:

$$\begin{tikzcd}
\text{source message} \arrow[d, "\text{split}"'] & \text{codemessage} \\
\text{source words} \arrow[r, "\text{encode}"] & \text{codewords} \arrow[u, "\text{concatenate}"']
\end{tikzcd}$$

However, in much of coding theory the source messages/words are irrelevant and will be ignored.

Chapter 2 Coding for Data-Compression

2.1. Instantaneous and uniquely decipherable codes.

A word w_1 is a *prefix* of a word w_2 if w_1 occurs at the start of w_2; e.g., $0\,1\,1$ is a prefix of $0\,1\,1\,0\,1$. A code C is *instantaneous* if no codeword is a prefix of any other codeword. C is *uniquely decipherable* if each codemessage corresponds to exactly one sequence of codewords. Instantaneous codes are uniquely decipherable:

Example 1. $C_1 = \{0,\ 1\,1,\ 1\,0\,0,\ 1\,0\,1\}$.

C_1 is instantaneous. To decipher a codemessage, read from the left until a codeword is completed, then iterate with what is left. E.g.,

$$0\,|\,1\,1\,|\,0\,|\,0\,|\,1\,0\,0\,|\,1\,0\,1\,|\,0\,|\,0\ .$$

Example 2. $C_2 = \{0,\ 0\,1\,0,\ 0\,1\,1\}$.

C_2 is not instantaneous (e.g., 0 is a prefix of $0\,1\,0$), but it *is* uniquely decipherable: in a codemessage, $1\,1\,1$ is impossible, $1\,1$ can occur only as part of $0\,1\,1$, and any other 1's must occur as part of $0\,1\,0$; what is left is all 0's. E.g.,

$$\underline{0}\,|\,\underline{0\,1\,1}\,|\,\underline{0\,1\,1}\,|\,\underline{0\,1\,0}\,|\,\underline{0\,1\,1}\,|\,\underline{0\,1\,0}\,|\,\underline{0}\,|\,\underline{0}\,|\,\underline{0\,1\,0}\,|\,\underline{0\,1\,1}\,|\,\underline{0}\ .$$

Example 3. $C_3 = \{0\,0,\ 0\,1,\ 1\,0,\ 0\,1\,0\}$.

C_3 is neither instantaneous nor uniquely decipherable. For example, the codemessage $0\,1\,0\,0\,0\,0\,1\,0$ has the two interpretations

$$\begin{aligned} & 0\,1|0\,0|0\,0|1\,0 \\ \text{and}\quad & 0\,1\,0|0\,0|0\,1\,0\,. \end{aligned}$$

Theorem 2.1. Let A be an alphabet of α symbols, and let m and $l_1 \leqslant \ldots \leqslant l_m$ be positive integers. Then the following three statements are equivalent.

(a) There exists an instantaneous code C over A with codewords $c_1, \ldots, c_m$ of lengths $l_1, \ldots, l_m$. (L. G. Kraft, 1949)

[MS dissertation, MIT.]

(b) There exists a uniquely decipherable code C' over A with codewords $c'_1, \ldots, c'_m$ of lengths $l_1, \ldots, l_m$. (B. McMillan, 1956)

[b. 1915; Bell Labs.]

(c) $\sum_{i=1}^{m} \alpha^{-l_i} \leqslant 1$.

Proof. (a) $\Longrightarrow$ (b) because every instantaneous code is uniquely decipherable.

(c) $\Longrightarrow$ (a): Choose c_1 to be an arbitrary word over A of length l_1. For $2 \leqslant s \leqslant m$, suppose we have chosen $c_1, \ldots, c_{s-1}$ of lengths $l_1, \ldots, l_{s-1}$ so that none is a prefix of any other. We must now choose c_s.

There are α^{l_s} possible words of length l_s, but $\alpha^{l_s - l_i}$ of them have c_i as a prefix, for each i ($1 \leqslant i \leqslant s-1$). However,

$$\sum_{i=1}^{s-1} \alpha^{l_s - l_i} < \alpha^{l_s},$$

since

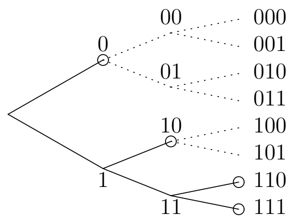
$$\sum_{i=1}^{s-1} \alpha^{-l_i} < \sum_{i=1}^{m} \alpha^{-l_i} \leqslant 1.$$

Thus there exists a word of length l_s that does not have any of $c_1, \ldots, c_{s-1}$ as a prefix. Choose c_s to be any such word. Then, for $1 \leqslant i \leqslant s-1$, c_i is not a prefix of c_s, and c_s is not a prefix of c_i since $l_s \geqslant l_i$. Continue until $s = m$.

One way of visualizing this proof is on a tree. Suppose, for example, $\alpha = 2$, $m = 4$, $(l_1, l_2, l_3, l_4) = (1, 2, 3, 3)$. Then

$$\sum_{i=1}^{m} \alpha^{-l_i} = \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \tfrac{1}{8} = 1,$$

and so there is an instantaneous code with these word lengths.



(b) $\Longrightarrow$ (c) (J. Karush, 1961): Write $x := \alpha^{-1}$ and let r be a positive integer. Then

$$\begin{aligned}\left(\sum_{i=1}^{m} \alpha^{-l_i}\right)^r = \left(\sum_{i=1}^{m} x^{l_i}\right)^r &= \left(\sum_{w \in C'} x^{l(w)}\right)^r \\ &= \sum_{w_1 \in C'} \sum_{w_2 \in C'} \cdots \sum_{w_r \in C'} x^{l(w_1)} x^{l(w_2)} \ldots x^{l(w_r)},\end{aligned}$$

since each term in the sum is obtained by choosing $x^{l(w_1)}$ from the 'first' bracket, $x^{l(w_2)}$ from the 'second', ..., $x^{l(w_r)}$ from the 'last', for some $w_1, w_2, \ldots, w_r \in C'$. But the summand is x^i, where i is the length of the codemessage $w_1 w_2 \ldots w_r$, and so the sum equals $\sum_{i=1}^{\infty} a_i x^i$, where a_i is the number of sequences of r codewords that give codemessages of length i. Note that two such sequences cannot give the same codemessage, since C' is uniquely decipherable, and so $a_i \leqslant \alpha^i$, the number of different strings of i symbols from A; and of course $a_i = 0$ if $i > r l_m$. So

$$\left(\sum_{i=1}^{m} \alpha^{-l_i}\right)^r = \sum_{i=1}^{\infty} a_i x^i = \sum_{i=1}^{\infty} a_i \alpha^{-i} \leqslant \sum_{i=1}^{r l_m} \alpha^i \alpha^{-i} = r l_m.$$

Hence

$$\sum_{i=1}^{m} \alpha^{-l_i} \leqslant (r l_m)^{1/r} \to 1 \quad \text{as} \quad r \to \infty.$$

Since r can be arbitrarily large, (c) follows. //

2.2. The noiseless coding theorem for memoryless sources. Suppose we have a source S that emits words $w_1, \ldots, w_m$

with positive probabilities $p_1, \ldots, p_m$, each word emitted being independent of all previous words (hence, 'memoryless'). Suppose we have an alphabet A of α symbols, a uniquely decipherable code C over A with codewords $c_1, \ldots, c_m$ of lengths $l_1, \ldots, l_m$, and an encoding map $\begin{array}{c} w_1 \to c_1 \\ w_2 \to c_2 \\ \vdots \\ w_m \to c_m \end{array}$.

The *average length* of C is $l(C) := \sum_{i=1}^m p_i l(c_i) = \sum_{i=1}^m p_i l_i$. C is *optimal* or *compact* if $l(C)$ is as small as possible subject to all these conditions. This means that, on average over all source messages, the length of the corresponding codemessage is minimal among all those produced by uniquely decipherable codes over alphabet A.

Theorem 2.2. *The noiseless coding theorem* (C. E. Shannon, 1948). If C is an optimal code for the above source S over the alphabet A, then

$$\frac{H(S)}{\log_2 \alpha} \leqslant l(C) < \frac{H(S)}{\log_2 \alpha} + 1,$$

where $H(S) := -\sum_{i=1}^m p_i \log_2 p_i$, the entropy or uncertainty of the source S.

Proof. Let C be *any* uniquely decipherable code for S over A, with word lengths $l_1, \ldots, l_m$. Let $\beta := \sum_{i=1}^m \alpha^{-l_i}$ and $q_i := \frac{\alpha^{-l_i}}{\beta}$ $(i = 1, \ldots, m)$, so that $(q_1, \ldots, q_m)$ is a probability distribution and $\log_2 q_i = -l_i \log_2 \alpha - \log_2 \beta$. By Lemma 1.3.1,

$$\begin{aligned} H(S) &= -\sum_{i=1}^m p_i \log_2 p_i \leqslant -\sum_{i=1}^m p_i \log_2 q_i \\ &= \left(\sum_{i=1}^m p_i l_i\right) \log_2 \alpha + \left(\sum_{i=1}^m p_i\right) \log_2 \beta \\ &= l(C) \log_2 \alpha + \log_2 \beta. \end{aligned}$$

But $\beta = \sum_{i=1}^m \alpha^{-l_i} \leqslant 1$ by Theorem 2.1, so $\log_2 \beta \leqslant 0$ and $\frac{H(S)}{\log_2 \alpha} \leqslant l(C)$, as required.

To prove the upper bound, it suffices to prove that *there exists* a uniquely decipherable code satisfying this bound, since then every optimal code must satisfy it. So, for each i, choose l_i to be the *least* integer such that $\frac{-\log_2 p_i}{\log_2 \alpha} \leqslant l_i$,

$$\text{i.e., } -\log_2 p_i \leqslant l_i \log_2 \alpha, \quad \text{i.e., } p_i^{-1} \leqslant \alpha^{l_i}, \quad \text{i.e., } \alpha^{-l_i} \leqslant p_i.$$

Then

$$\frac{-\log_2 p_i}{\log_2 \alpha} \leqslant l_i < \frac{-\log_2 p_i}{\log_2 \alpha} + 1.$$

Also, $\sum_{i=1}^m \alpha^{-l_i} \leqslant \sum_{i=1}^m p_i = 1$, and so there exists a uniquely decipherable code C with these codeword lengths by Theorem 2.1. Also, $l(C) = \sum_{i=1}^m p_i l_i < \sum_{i=1}^m \left(\frac{-p_i \log_2 p_i}{\log_2 \alpha} + p_i\right) = \frac{H(S)}{\log_2 \alpha} + 1$. //

In the binary case ($\alpha = 2$), R. G. Gallagher proved in 1978 that $l(C) \leqslant H(S) + \max_i p_i + \log_2\left(\frac{2 \log_2 \mathrm{e}}{\mathrm{e}}\right) \approx H(S) + \max_i p_i + 0.0861$.

2.3. Huffman's algorithm. This is an algorithm for constructing *instantaneous* optimal codes. We shall consider only *binary* codes, with alphabet $\{0, 1\}$. Let S be a memoryless source that emits words $w_1, \ldots, w_m$ with probabilities $p_1 \geqslant \ldots \geqslant p_m > 0$. Let S' be a memoryless source that emits words $w'_1, \ldots, w'_{m-1}$ with probabilities $p_1, \ldots, p_{m-2}, p_{m-1} + p_m$. Let C' be an instantaneous code for S' over $\{0, 1\}$ with encoding map

$$\begin{array}{c} w'_1 \to c'_1 \\ \vdots \\ w'_{m-1} \to c'_{m-1} \end{array}.$$

Let C be the code $\begin{array}{c} w_1 \to c'_1 \\ \vdots \\ w_{m-2} \to c'_{m-2} \\ w_{m-1} \to c'_{m-1}0 \\ w_m \to c'_{m-1}1 \end{array}$

for S. Clearly C is instantaneous.

Lemma 2.3.1. $l(C) = l(C') + p_{m-1} + p_m$.

Proof. $l(C) = \sum_{i=1}^{m-2} p_i l(c_i') + (p_{m-1} + p_m)(l(c_{m-1}') + 1)$

$$= l(C') + p_{m-1} + p_m. \quad //$$

Theorem 2.3. C is optimal for S if and only if C' is optimal for S'.

Proof. Let C^* be an *instantaneous* optimal code for S over $\{0,1\}$ (which exists by Theorem 2.1), with encoding map

$$\begin{array}{c} w_1 \to c_1^* \\ \vdots \\ w_m \to c_m^* \end{array} .$$

Claim 1. If $p_i > p_j$ then $l(c_i^*) \leqslant l(c_j^*)$.

Proof. If $l(c_i^*) > l(c_j^*)$, form a new instantaneous code $\hat{C}$ from C^* by interchanging c_i^* and c_j^*; then

$$\begin{aligned} l(\hat{C}) &= l(C^*) - p_i l(c_i^*) - p_j l(c_j^*) + p_j l(c_i^*) + p_i l(c_j^*) \\ &= l(C^*) - (p_i - p_j)(l(c_i^*) - l(c_j^*)) \\ &< l(C^*), \quad \Rightarrow\Leftarrow . \end{aligned}$$

Claim 2. Among the codewords of maximum length in C^*, there are two that agree in all but the last digit.

Proof. If not, then we can delete the last digit of every codeword of maximum length and still have an instantaneous code, $\Rightarrow\Leftarrow$.

By Claims 1 and 2, we may assume w.l.o.g. that c_{m-1}^* and c_m^* agree in all but the last digit, since $l(C^*)$ does not change if we interchange two longest codewords, or two codewords used with the same probability. Say $c_{m-1}^* = c^*0$ and $c_m^* = c^*1$. Let $C^{*\prime}$ be the code

$$\begin{array}{c} w_1' \to c_1^* \\ \vdots \\ w_{m-2}' \to c_{m-2}^* \\ w_{m-1}' \to c^* \end{array}$$

for S'. Then C^* is formed from $C^{*\prime}$ in the same way that C is formed from C'. By Lemma 2.3.1,

$$l(C^*) - l(C^{*\prime}) = l(C) - l(C') = p_{m-1} + p_m,$$

and so $l(C^*) - l(C) = l(C^{*\prime}) - l(C')$. Since C^* is optimal, $l(C^*) \leqslant l(C)$, and so $l(C^{*\prime}) \leqslant l(C')$, with equality iff $l(C^*) = l(C)$. Since C' was arbitrary, therefore $C^{*\prime}$ is optimal for S', and

$$\begin{aligned} C' \text{ is optimal for } S' &\iff l(C') = l(C^{*\prime}) \\ &\iff l(C) = l(C^*) \\ &\iff C \text{ is optimal for } S. \quad // \end{aligned}$$

Theorem 2.3 gives a recursive algorithm for constructing instantaneous optimal codes, which is due to D. A. Huffman (1952). (b. 1925, MIT, → Santa Cruz.) Given the source S with m words, amalgamate the two least probable words to form a source S' with $m-1$ words. Repeat until we have a source $S^{(m-2)}$ with two words. Form the code $C^{(m-2)}$ for $S^{(m-2)}$ by assigning codeword 0 to one source word and codeword 1 to the other; $C^{(m-2)}$ is clearly instantaneous and optimal. Now use Theorem 2.3 to construct instantaneous optimal codes $C^{(m-3)}, \ldots, C', C$ for $S^{(m-3)}, \ldots, S', S$ in turn.

Example 1.

	S	C		S'	C'		S''	C''		S'''	C'''
w_1	0.45	1	w_1	0.45	1	w_1	0.45	1	w_{2345}	0.55	0
w_2	0.25	0 1	w_2	0.25	0 1	w_{345}	0.3	0 0	w_1	0.45	1
w_3	0.1	0 0 1	w_{45}	0.2	0 0 0	w_2	0.25	0 1			
w_4	0.1	0 0 0 0	w_3	0.1	0 0 1						
w_5	0.1	0 0 0 1									

Example 2.

	S	C		S'	C'		S''	C''		S'''	C'''		$S^{(4)}$	$C^{(4)}$
w_1	0.3	0 0	w_1	0.3	0 0	w_1	0.3	0 0	w_{3456}	0.45	1	w_{12}	0.55	0
w_2	0.25	0 1	w_2	0.25	0 1	w_2	0.25	0 1	w_1	0.3	0 0	w_{3456}	0.45	1
w_3	0.2	1 1	w_3	0.2	1 1	w_{456}	0.25	1 0	w_2	0.25	0 1			
w_4	0.1	1 0 1	w_{56}	0.15	1 0 0	w_3	0.2	1 1						
w_5	0.1	1 0 0 0	w_4	0.1	1 0 1									
w_6	0.05	1 0 0 1												

Compression with Huffman's algorithm is limited because:

(1) the encoding map must be included with the message;

(2) the *only* form of redundancy it removes is that implied by the differing frequencies of source words; e.g., if the source words are 0 and 1 (only), then the source message 0 1 0 1 0 1 . . . is highly redundant, but Huffman's algorithm will not compress it at all.

There is also a computational disadvantage, in that the message must be scanned twice, once to choose the source words and discover their frequencies, and once to encode them.

Nevertheless, effective computer programs based on Huffman's algorithm exist and typically achieve compressions of around 45–50% on long text files and 75% on source code for computer programs.

2.4. The LZW algorithm. (A. Lempel and J. Ziv, 1977; T. A. Welch, 1984.) This algorithm avoids all the above problems. The codewords are all 2^n binary words of length n, for some n. The encoder and decoder start with the same code table, containing a numerical list of symbols in the source alphabet. At each stage, the encoder encodes the longest initial string of the remaining message for which a codeword exists in the table, and then adds that string plus the next symbol as a new source word in the table (unless the table is already full). The decoder decodes the incoming code and reconstructs the same code table one word behind the encoder.

Example with source alphabet ␣ (space), a, b.

Encoder

a	␣	b	b	a	a␣	a	b	␣	a␣a	ba	bab	␣b	a
1	0	2	2	1	3	1	2	0	8	6	13	4	1

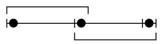
␣	0	a␣a	8
a	1	ab	9
b	2	b␣	10
a␣	3	␣a	11
␣b	4	a␣ab	12
bb	5	bab	13
ba	6	bab␣	14
aa	7	[␣ba	15]

Decoder

1	0	2	2	1	3	1	2	0	8	6	13	4	1
a	␣	b	b	a	a␣	a	b	␣	a␣a	ba	bab	␣b	a

␣	0	a␣a	8
a	1	ab	9
b	2	b␣	10
a␣	3	␣a	11
␣b	4	a␣ab	12
bb	5	bab	13
ba	6	bab␣	14
aa	7	[␣ba	15]

If the decoder receives a codeword that is not in the code table, then it must have been put into the encoder's code table at the previous word, and so corresponds to the previous source word received plus its first symbol.



The disadvantages of this method are:

(1) It takes time for the code table to build up, so the first part of the message is expanded, not compressed. (Thus the method does not work with short messages.)

(2) After a long time, the code table becomes full, and if the characteristics of the message change markedly after that, then compression is less effective.

Nevertheless, the algorithm achieves compressions of typically 55–60% on long text files and 85% on source code for computer programs.

A test on a long text file:

program	reduced to	compression	method
compress	41%	59%	LZW algorithm
gzip	28%	72%	Lempel–Ziv 1977
bzip2	23%	77%	Burrows–Wheeler block sorting + Huffman coding