Text compression using a 4 bit coding scheme

J. Pike

Holly Cottage, Chawston, Bedford MK44 3BH, UK

The most frequently used words in natural or printed English are found unexpectedly to contain only an average proportion of the most frequently used letters. This independence of the word and letter frequency distributions is used to minimise the number of bits necessary to code natural English text. It is shown that mean bit rates of less than 4 per character can be achieved for text using the full ASCII set of 96 characters, by combining a variable bit length representation of each character with a character combination dictionary of a 100 or more common words. A simple practical scheme is presented which uses 4, 8 or 12 bits to code the characters and dictionary words. Using this scheme with a 205 word dictionary, a mean code rate of 3.87 bits per character is achieved. It is indicated how even this rate might be improved with a larger dictionary or by basing the dictionary on the more numerous word prefixes.

(Received September 1980)

1. Introduction

Digital storage or transmission of text characters usually uses 8 bits per character, as for example in ASCII code. As these 8 bits can represent up to 2^8 (i.e. 256) different characters and about 100 characters are sufficient for most text, there is redundancy when using 8 bits to code each character. By removing this redundancy and using the structure of natural English text, several authors (Huffman, 1952; Lea, 1978; Shannon, 1951; White, 1967; Williams, 1978) have developed techniques for the more efficient coding of text. In general these techniques fall into two classes: those which use a variable bit length representation of the text characters according to their frequency (Huffman, 1952), and those which use the spare capacity of the 8 bit coding to represent a common word or character string dictionary (Lea, 1978; White, 1967; Williams, 1978). Results from the variable bit length approach have shown that the minimum number of bits to represent the text is given by the product of the number of text characters and the mean bit length per character (H), where

$$H = -\sum P_J \ln P_J \qquad (1)$$

in which P_J is the probability of the Jth character occurring. Applying this to natural English text (White, 1967), a lower bound of greater than 4 is found for the minimum mean bit rate per character. More recent attempts (Lea, 1978; White, 1967; Williams, 1978) to reduce the mean bit rate have concentrated on the dictionary technique, which is not subject to this lower limit. Indeed, if it is assumed that the words are ranked according to their frequency of occurrence in natural English and that the probability of a particular word being of rank R is given by the empirical rule (Zipf, 1949)

$$P_R = \frac{0{\cdot}1}{R} \qquad (2)$$

then a minimum mean bit length representation of less than 2 is theoretically possible for each character. However, to achieve this lower limit, a dictionary of over 12000 words with their appropriate bit codings is required. For more practical dictionary sizes (say up to 1000 words or character strings), even schemes which have complicated techniques for obtaining near optimised string dictionaries (Lea, 1978; Williams, 1978) seem to be unable to breach the 4 bits per character barrier (which would give a 50% compression of the text compared with ASCII coding).

Some dictionary schemes (Lea, 1978) severely restrict the character set that can be represented in the text. This type of restriction might be acceptable for certain applications, but in general it would appear desirable to include at least all the 95 non-control ASCII characters. An advantage of variable bit length schemes is that further 'rare' or unusual characters can easily be accommodated within the coding whilst making little difference to the coding efficiency. Other schemes have different restrictions. To clarify the aims of the present paper, the properties considered here to be important in coding natural English text are listed below:

(1) The coding should use a small number of bits to represent natural English text and should not use significantly more bits when the text varies due to style or content.

(2) A full range of characters should be representable and in particular text using only upper case letters should be able to be coded with similar efficiency as text using lower case letters.

(3) The storage 'overhead' for the program coding and the dictionary should be small. This requires that the text coding

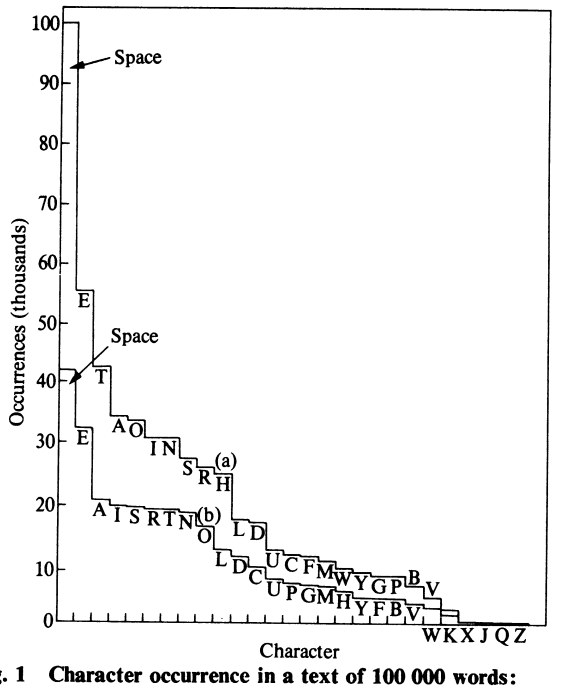


Fig. 1 Character occurrence in a text of 100 000 words: (a) full text (total 538 023), (b) text minus 200 common words (total 308800 characters)

CCC-0010-4620/81/0024-0324 $03.50

Table 1

Word occurrences in a natural English text of 100 000 words

Rank	Word	Occurrence in 100000 words	Dictionary word occurrences[a]	Rank	Word	Occurrence in 100000 words	Dictionary word occurrences[a]
1	the	7310	7425	70	other	169	194
2	of	3998	4225	71	into	166	166
3	and	3280	3280	72	men	164	194
4	to	2924	3214	73	must	163	168
5	a	2120	—[b]	74	people	163	163
6	in	2116	2533	75	said	161	161
7	that	1345	1367	76	may	160	162
8	it	1216	1488	77	man	156	214
9	is	1213	1258	78	about	153	153
10	I	1155	—	79	over	153	153
11	for	1035	1237	80	some	152	206
12	be	846	1072	81	these	152	152
13	was	839	866	82	two	146	146
14	as	782	795	83	very	145	145
15	you	773	1132	84	before	139	139
16	with	727	747	85	great	134	178
17	he	680	1294	86	could	133	147
18	on	643	1091	87	such	132	132
19	have	617	658	88	first	131	131
20	by	600	600	89	upon	129	129
21	not	589	634	90	every	124	155
22	at	585	—	91	how	124	146
23	this	572	572	92	come	123	171
24	are	542	549	93	us	123	251
25	we	529	691	94	shall	120	120
26	his	517	535	95	should	118	131
27	but	504	504	96	then	115	—
28	they	478	495	97	like	113	129
29	all	466	506	98	well	113	113
30	or	458	—	99	little	111	112
31	which	454	454	100	say	111	191
32	will	445	464	101	because	108	108
33	from	433	433	102	being	108	108
34	had	411	411	103	under	108	109
35	has	390	390	104	after	107	117
36	one	368	—	105	here	107	—
37	our	331	351	106	good	106	125
38	an	330	—	107	make	105	165
39	been	328	329	108	most	105	107
40	no	321	—	109	many	104	104
41	their	315	350	110	much	104	104
42	there	306	322	111	those	104	194
43	were	305	307	112	way	104	104
44	so	300	—	113	see	103	258
45	my	296	343	114	world	103	117
46	if	263	263	115	know	102	196
47	me	257	509	116	day	101	175
48	what	253	276	117	never	101	101
49	would	252	267	118	did	100	123
50	who	248	370	119	new	99	122
51	when	237	237	120	down	95	98
52	him	234	285	121	even	94	132
53	them	228	262	122	long	94	110
54	her	222	—	123	years	93	93
55	war	214	229	124	country	92	111
56	your	214	—	125	business	91	91
57	any	210	240	126	right	91	110
58	more	210	210	127	get	89	117
59	now	210	210	128	life	89	90
60	its	208	—	129	just	86	110
61	time	205	232	130	take	86	164
62	up	204	207	131	where	83	83
63	do	203	—	132	work	83	127
64	out	203	206	133	things	82	126
65	can	197	219	134	part	79	97
66	than	194	194	135	through	78	94
67	only	189	189	136	while	78	79
68	she	188	217	137	last	77	85
69	made	170	170	138	might	77	84

Table 1 continued

Rank	Word	Occurrence in 100000 words	Dictionary word occurrences[a]
139	am	75	124
140	back	75	83
141	old	75	84
142	own	75	90
143	three	75	75
144	against	73	73
145	go	73	247
146	think	73	99
147	came	71	71
148	matter	71	76
149	days	70	—
150	without	70	70
151	also	69	69
152	public	68	69
153	today	68	88
154	yet	68	68
155	don't	67	—
156	same	67	67
157	thought	67	113
158	each	63	63
159	far	63	100
160	home	63	71
161	put	63	67
162	again	62	62
163	always	62	62
164	nothing	61	61
165	present	61	68
166	between	60	60
167	going	60	60
168	money	60	60
169	per	60	135
170	once	59	59
171	peace	59	62
172	woman	59	59
173	year	59	—
174	another	58	58
175	away	58	58
176	cannot	58	58
177	fact	58	85
178	half	58	58
179	still	58	58
180	give	57	127
181	government	57	57
182	power	57	68
183	too	57	—
184	yours	56	—
185	found	55	56
186	few	54	56
187	possible	54	72
188	does	53	63
189	food	53	55
190	house	53	66
191	less	53	56
192	oh	53	—
193	best	52	52
194	case	52	71
195	line	52	79
196	place	52	85
197	says	52	—
198	since	52	52
199	ever	51	51
200	himself	51	51
201	let	51	—
202	military	51	51
203	tell	51	71
204	why	51	51
205	big	50	54
206	got	50	—
207	until	50	50
208	went	50	50
209	find	49	57
210	five	49	49
211	dear	48	—
212	interest	48	82
213	left	48	48
214	order	48	71
215	service	48	54
216	set	48	—
217	steel	48	48
218	women	47	48
219	among	46	46
220	front	46	49
221	given	46	—
222	hand	46	81
223	high	46	81
224	means	46	72
225	night	46	46
226	small	46	54
227	taken	46	—
228	both	45	46
229	morning	45	46
230	used	45	—
231	whole	45	52
232	city	44	—
233	enough	44	44
234	gun	44	—
235	next	44	—
236	thing	44	—
237	want	44	—
238	army	43	—
239	off	43	—
240	pay	43	—

[a] Includes prefixes (not otherwise included) in the first 1000 ranked words.
[b] words not included in the 205 word dictionary.

and decoding scheme should be relatively simple and the dictionary small.

(4) The coding and decoding of the text should be fast. The coding speed depends primarily on the dictionary search times, whereas the decoding time depends more on the complication of the scheme.

I attempt in this paper to produce a simple scheme satisfying the above criteria with a mean coding rate of better than 4 bits per character. A crucial observation is that the letter frequency distribution in natural English text is largely independent of the word frequency distribution, giving prospects of combining both dictionary and character frequency distribution techniques to gain the desired improvement. Word and letter frequency distributions in natural English are given in Dewey (1923, Table 4) from the analysis of 100000 words of text. The frequency distribution found for the letters (Dewey, 1923) is shown by (a) in **Fig. 1**, where the most common character 'space' is included on the assumption that it occurs at a mean rate of one per word. Also shown from the 100000 word text are the 240 most common words, ranked in order of frequency of occurrence in **Table 1.** Removal from the text of 200 of these words leaves the lower character frequency distribution of Fig. 1. Although the letter order is changed significantly, the shape of the distribution remains very similar to the original. Thus, it should be advantageous to optimise the word and letter representations independently and then combine the results.

A possible disadvantage of this dual technique is that the scheme will be too complicated. In Section 3 a practical scheme is developed based on a simple 4, 8 or 12 bit representation of characters and words, which approaches very closely

the optimum coding rate of the dual scheme.

It might be noted that only ranked word dictionaries with a space preceding each dictionary word are considered here. Optimisation of the dictionary members by analysing word 'prefix' frequency distributions could produce a reduction in the coding rate of 3·87 bits per character demonstrated in this paper using a 205 word dictionary. The development of such a dictionary is beyond the scope of the present paper.

2. Minimisation of mean bit rates for words and characters

The representation of words and characters using variable bit length codes is analysed theoretically. Simple expressions for the minimum number of bits to code natural English text are derived, against which the efficiency of practical schemes can be compared.

Consider a dictionary formed from the words of rank 1 to N. The proportion of the words in natural English text represented by the dictionary is given by

$$P_N = \sum_1^N P_R \tag{3}$$

where the values of P_R may be obtained from Table 1 for $N < 240$ or more generally from Eqn (2). To find P_N from Eqn (2) we need to sum $1/R$ from 1 to N. The usual summation expression in terms of Euler's constant (Abramowitz and Stegun, 1965) is not sufficiently accurate for our purposes, so expressions have been derived for $1/R$ and other sums needed later which are better than 1% accurate for $N > 1$:

$$\sum 1/R = 0{\cdot}577 + \ln(N + \tfrac{1}{2}) \tag{4}$$

$$\sum (\ln R)/R = \tfrac{1}{2}[\ln(N + \tfrac{1}{2})]^2 - 0{\cdot}073 \tag{5}$$

$$\sum R^{-0{\cdot}843} = 6{\cdot}37(N + \tfrac{1}{2})^{0{\cdot}157} - 5{\cdot}8 \tag{6}$$

The proportion of the text words which are dictionary words is thus given by

$$P_N = 0{\cdot}0577 + 0{\cdot}1 \ln(N + \tfrac{1}{2}) \tag{7}$$

The increase of P_N with $\ln N$ is shown by the lower curve in **Fig. 2.** From $P_N = 1$, we see that Eqn (2) contains the fairly arbitrary prediction that all the text will be represented by a 12367 word dictionary. At $N = 100$ about 52% of the text words are words of rank 1 to 100. This compares well with the 54% obtained by direct summation from Table 1.

Suppose we code the dictionary word of rank R using B_R bits and this combination of bits represents a fraction F_R of all the available bit combinations, such that §

$$B_R = -\operatorname{lb} F_R \tag{8}$$

then the mean bit length of the dictionary words in the text (B_N) is given by

$$B_N P_N = \sum P_R B_R = -\sum P_R \operatorname{lb} F_R \tag{9}$$

§ lb is now the recommended symbol for $\log_2$, see for example British Standard, BS 1991: Part 1, 1976.

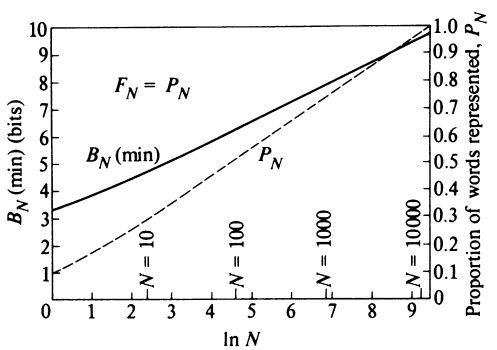


Fig. 2 Minimum mean bits per word for a coding dictionary of N members

As P_N is a function of N only, the values of B_R (or F_R) which minimise the number of bits to represent the dictionary words in the text will be the same as those obtained from minimising B_N. Minimising B_N, with the total fraction of bit combinations used by the dictionary constant (i.e. $F_N = \sum F_R = \text{constant}$), we obtain in the usual way

$$\frac{P_R}{P_N} = \frac{F_R}{F_N} \tag{10}$$

Then from Eqn (9), the minimum value of B_N is

$$B_N(\min) = -\operatorname{lb} F_N - \sum \frac{P_R}{P_N} \operatorname{lb} \frac{P_R}{P_N} \tag{11}$$

For the particular case $F_N = P_N = 1$ this equation has been derived previously (Shannon, 1951). However, the value of B_N(min) obtained (Shannon, 1951) was 11·82 bits per word, which is in error as is shown below.

The tedious summation expression in Eqn (11) can be removed using Eqn (5) to give

$$B_N(\min) = \frac{\ln 10}{\ln 2} - \frac{\ln(F_N/P_N)}{\ln 2} + \frac{[\ln(N + \tfrac{1}{2})]_2 - 0{\cdot}146}{[\ln(N + \tfrac{1}{2}) + 0{\cdot}577]\ln 4} \tag{12}$$

For $F_N = P_N$ the minimum mean bit length per word is shown as the upper curve of Fig. 2. When $N = 100$ we see that just over half the words in the text can be represented at a mean bit rate of 6·3 bits per word. When $F_N = P_N = 1$, B_N(min) is 9·7 bits per word. It is this value and not Shannon's (1951) value given above which is correct.

After the N dictionary words have been identified in the text, there remain the less common words plus a variety of punctuation marks and other odd characters. Although these latter rare characters can vary significantly in their occurrence from text to text, their comparative rarity means that however they are represented they make little difference to the overall efficiency of the coding.

White (1967) has suggested coding the free characters using digrams and trigrams. However, he needed a dictionary of 1340 words, digrams and trigrams to achieve a compression ratio of 0.53. Here an alternative approach is considered in which the bit length to represent the free characters is varied in a manner similar to that applied previously (Huffman, 1952) to the whole text. Similar analysis to that used in Huffman (1952) gives a mean bit length per character of

$$B_K(\min) = -\operatorname{lb} F_K - \sum_1^K \frac{P_J}{P_K} \operatorname{lb} \frac{P_J}{P_K} \tag{13}$$

where F_K is the total fraction of bit combinations used to represent the free characters, P_K is the probability of a character selected at random from the text being a free character and P_J is the probability that this free character is the Jth character. That is, the proportion of free characters is given by

$$P_K = \sum_1^K P_J \tag{14}$$

where K is the number of different characters to be represented. The similarity between the above expressions and those for the dictionary words is immediately apparent. An essential difference, however, is that K is a relatively small finite number (e.g. $K = 96$ for the ASCII text characters) and thus every free character may be represented by summing over K. Values of P_J can be obtained from Fig. 1 for the letters and 'space' character and from Dewey (1923) for the others. The main punctuation characters (comma, full stop, hyphen and inverted commas) were found to occur 5951, 5272, 2416 and 2216 times in 100000 words of text, respectively, and all the other rare characters occurred a total of about 12000 times (Dewey, 1923). One common 'character' not mentioned in Dewey (1923) is the 'end of line' character. It is necessary to code the end of line position if the text is to be reproduced exactly.

It is assumed that there are 10 words per line on average, giving a total of 10000 end of line characters.

When all the text characters are treated as free characters, the frequencies from Dewey (1923) as described above give a minimum mean value of 4·32 bits per character. With a 200 word dictionary, using Fig. 1 for the values of P_J in Eqn (14), we obtain

$$B_K(\text{min}) = -\text{lb}F_K + 4{\cdot}52 \qquad (15)$$

The small change between 4·32 for the full text and 4·52 above reflects the relative similarity in the distribution of the free characters as N is varied.

Having obtained the minimum number of bits per dictionary word [Eqn (12)] and the minimum number of bits per character [Eqn (15)] I now investigate how best to apportion the total bit combinations between them. If the whole text is coded and all the available bit combinations used then

$$P_N \frac{L_N}{L_\infty} + P_K = 1 \qquad (16)$$

$$F_N + F_K = 1 \qquad (17)$$

where L_N is the mean word length of all the dictionary words in the text and L_∞ is the mean word length of all the words in the text. The value of L_∞ is well known (White, 1967) to be close to 4·5 letters per word for natural English. The analysis of 100000 words in Dewey (1923) finds 4·38 letters per word. A space is included on the front of every word, except for words which begin a new line, hence 5·4 characters per word is an appropriate value for L_∞. An empirical rule for the length of the word of rank R is given by White (1967) as

$$L_R = 2{\cdot}45R^{0{\cdot}167} \qquad R > 10 \qquad (18)$$

Unfortunately, this equation does not match the line through the data from which it is derived and further the value of $L_\infty = 5{\cdot}75$ which it gives (White, 1967) is too high. A better fit to White's data is given by

$$L_R = 2{\cdot}45\,R^{0{\cdot}157} \qquad R > 10 \qquad (19)$$

which gives a more realistic value for L_∞ of 5·43. With this value of L_R, the value of L_N can be found from

$$L_N P_N = \sum L_R P_R = 1{\cdot}56(N + \tfrac{1}{2})^{0{\cdot}157} - 1{\cdot}42 \qquad N > 10 \qquad (20)$$

The mean bit rates per character to code the text using a combination of character and word coding is

$$B_T = \frac{P_N}{L_\infty} B_N + P_K B_K \qquad (21)$$

Minimising B_T gives

$$\frac{F_K}{F_N} = L_\infty \frac{P_K}{P_N} \qquad (22)$$

and a minimum mean bit rate for the text as a function of N

$$B_T(\text{min}) = \frac{P_N}{L_\infty} B_N(\text{min}) + P_K B_K(\text{min}) \qquad (23)$$

where

$$B_N(\text{min}) = \frac{\ln(10P_N + 10P_K L_\infty)}{\ln 2} + \frac{[\ln(N + \frac{1}{2})]^2 - 0{\cdot}146}{20P_N \ln 2} \qquad (24)$$

$$B_K(\text{min}) = \ln[1 + P_N/(L_\infty P_K)] + 4{\cdot}52 \qquad (25)$$

$$P_N = 0{\cdot}0577 + 0{\cdot}1 \ln(N + \tfrac{1}{2}) \qquad (26)$$

$$P_K = 1 - P_N L_N / L_\infty \qquad (27)$$

$$= 1{\cdot}26 - 0{\cdot}287(N + \tfrac{1}{2})^{0{\cdot}157} \qquad N > 10 \qquad (28)$$

with $L_\infty = 5{\cdot}4$ the value of $B_T(\text{min})$ from the above equations is shown in **Fig. 3.** Of particular significance is the intersection of the curve with the 4 bit line below $N = 100$.

Thus, for a dictionary of 100 members, representation of the text at a mean rate of less than 4 bits per character becomes theoretically possible, and for a 1000 word dictionary the mean rate is 3·3 bits per character. The practical means to realise this potential is discussed in Section 3.

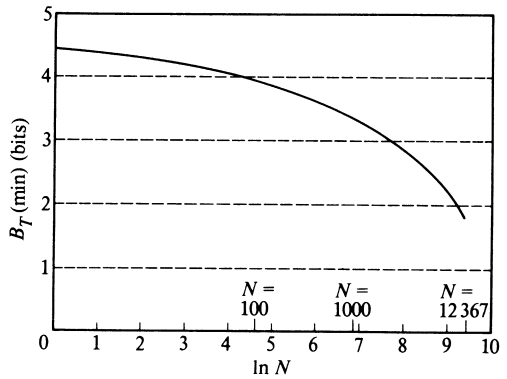
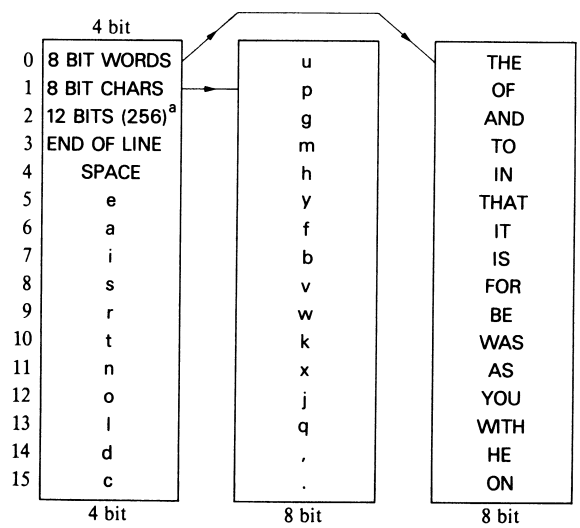


Fig. 3 Minimum mean bits per character for the combined dictionary and character coding scheme of this paper

It should be noted that the word frequency estimate of Eqn (1) used to produce Fig. 3 applies to words rather than word prefixes. Thus, for example, one would expect that the dictionary word 'be' would occur more frequently in the text than estimated in Table 1 because such a word as 'best' would be represented in part. Estimates of the occurrence of prefixes in natural English do not appear to be available. However, the occurrences of the most common words as prefixes amongst the 1000 most common words are shown in Table 1. It is difficult to estimate the improvement which would occur if the dictionary members were chosen using the most frequent prefixes rather than the word frequencies of Table 1. A more radical alternative would be to use string dictionaries (which do not necessarily start with a space) similar to those used in Lea (1978) and Williams (1978); however, as is explained in Section 3, such dictionaries have practical disadvantages compared with the 'prefix' dictionaries.

3. The 4 bit scheme

Coding schemes using unrestricted variable bit lengths to represent characters or words tend to be complicated and slow



[a] 256, 12 bit options coded as 67 rare characters and 189 dictionary words

Fig. 4 Possible representation of characters and words for the 4 bit scheme

because each bit has to be analysed individually. Schemes which use 8 bits for representing characters and dictionary words are much simpler but appear to have a lower limit of about 4 bits per character as their mean rate to represent natural English text. Here we use a 4 bit coding which retains much of the simplicity associated with the 8 bit schemes but manages to approach closer to the lower bit rates per character found in Section 2 for variable bit lengths.

The free characters are coded as shown in **Fig. 4**, with 13 characters using 4 bits, 16 characters using 8 bits and 67 characters using 12 bits. The characters using 4 bits are the letters e, a, i, s, r, t, n, o, l, d, c, plus 'space' and 'end of line' characters. From Fig. 1 it can be seen that these represent about 80% of the free characters. If the 67 rare characters represented by 12 bits form 2% of the characters, the mean rate of bits per character for the free characters is 4·88.

A scheme is now investigated using a modest 205 word dictionary which is coded with the 16 most common words using 8 bits and the rest of the words the 189 unused 12 bit options. The actual words used are those indicated in the occurrence column of Table 1, and it can be seen that they do not confirm exactly with the 205 most commonly occurring words. Consider for example the word 'a' which is ranked fifth. This should be coded on rank considerations as an 8 bit word. However, a 'space' and an 'a' only uses 8 bits when they are coded as free characters. Hence including 'a' in the dictionary does not reduce the number of bits used to record the text. Ideally the words should be reranked according to the number of bits saved in a natural English text assuming that e, a, i, s, r, t, n, o, l, d, c, use 4 bits and the other letters 8 bits. Returning to our sample dictionary of Table 1, we see that the 16 words using 8 bits can represent about 30% of all the words and that a further 30% of the words is represented by the 189 words using 12 bits. These word occurrences include words at the start of a line which do not begin with a space, and words which contain upper case letters. To enable the scheme to accommodate these words and also fulfil the requirement that both mixed and upper case text should both be coded efficiently, we introduce a number of logical rules, which although complicated to state are easily implemented in a computer program.

We assume that the 4 and 8 bit letters represent lower case letters except under the following conditions:

(1) the character is the first letter of the text;

(2) the character is the first letter following a full stop;

(3) the character is the single letter 'I' (i.e. part of 'space I space');

(4) the two previous letters have been upper case.

The last requirement enables text which is only upper case to be coded as efficiently as normal mixed upper and lower case text. The convention for the dictionary words is taken to be that each word is preceded by a space except when it is the first word in the line. Adding the spaces has the effect of increasing the mean word length of the dictionary by 0·9 characters (for the assumed 10 word line) without the need to record the 'space' in the dictionary listing. There is a second major advantage in starting all the dictionary words with a space. When coding the text it is only necessary to search the dictionary following a space or end of line character, giving a potential 80% saving in dictionary search times. The dictionary storage and search times can also be reduced by ordering the dictionary alphabetically and recording the location of the dictionary words which start with a particular letter. Only the dictionary words with the correct first letter then need be searched.

With these innovations, a 200 word dictionary requires about 600 bytes to store the necessary dictionary characters in ASCII. If dictionary storage needs to be reduced further, the letters forming the dictionary words can be coded in the same way as the free characters, at some cost in the coding and decoding speed for the dictionary words in the text.

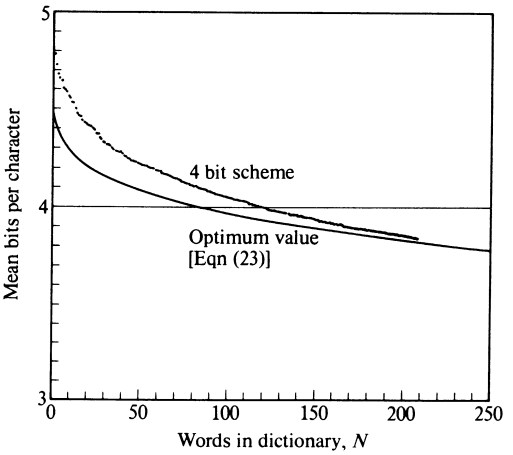


Fig. 5 Mean bits per character for coding natural English text using the 4 bit scheme

Using the dictionary of 205 words indicated in Table 1 and the character representation of Fig. 4, we can estimate the compression rate of natural English text using the present scheme. In **Fig. 5**, the mean bit rate per character for the 4 bit scheme applied to the 100000 words of text from Dewey (1923) is shown as a function of N. We see that the 4 bit scheme achieves values close to the optimised values also shown which are taken from Fig. 3. At the design value for N of 205 dictionary members, $F_N = 1$ and the rate of 3·87 bits per character is achieved, which is significantly below that of other methods.

The 205 word dictionary discussed above demonstrates the capability of practical schemes to achieve coding rates of less than 4 bits per character. Rates of less than 3·87 could be achieved either by increasing the dictionary size or optimising the dictionary members.

4. Conclusions

Variable bit length codings of both words and characters have been combined to form an efficient technique for coding natural English text. Theoretical optimisation of the word and character representation indicates that using 100 or more coded words, mean coding rates of 4 bits per character are possible and for 1000 words the rate is 3·3 bits per character. A practical coding scheme is presented which uses multiples of 4 bits to code the words and characters according to their frequency. The use of a 4 bit unit reduces the complicated processing usually associated with variable bit length schemes, whilst retaining a sufficiently good match with the frequency distributions to enable a coding rate better than 4 bits per character to be achieved with a 205 word dictionary.

The present scheme has been compared with other schemes to code natural English text. Although a considerable number of text compression schemes have appeared in the literature [see, for example, the list in Lea (1978)], many of them are only of theoretical interest or apply to reduced forms of text. From its construction, the present scheme will be superior to those which code just words or letters (Huffman, 1952). More successful schemes, however, use word fragment or string dictionaries to improve the compression rate, culminating in the near optimum dictionaries of Lea (1978) and Wolff (1978).

In Lea (1978), a compression rate of nearly 50% (equivalent to 4 bits per character) is achieved by coding a 206 member dictionary using fixed length code. This coding technique is simpler than the one used here, but the complexities encountered in searching the string dictionary were such that Lea (1978) needed to develop a hardware solution to achieve sufficient coding speed.

In Wolff (1978), compression rates which are superior to the present scheme are obtained with comparatively large dictionaries (> 500) on small samples of 2000–3000 words. Although Wolff obtains mean coding rates down to 3 bits per character, his dictionaries are derived specifically for each sample. Hence, to obtain a fair comparison the dictionary storage requirement must be included with that of the text, again giving a coding rate above 4 bits per character.

If the string search problem can be solved satisfactorily, a string dictionary could replace the word dictionary in the present scheme. However, a more immediate improvement might be achieved by using an optimised 'prefix' dictionary in which each string starts with a space. To match such a dictionary to the 4 bit scheme, the characters of each prefix need to be weighted according to their 4, 8 or 12 bit representation. These developments are beyond the scope of the present paper.

References

ABRAMOWITZ, M. and STEGUN, I. A. (1965). *Handbook of Mathematical Functions*, National Bureau of Standards, US Government Printing Office, Washington, DC.

DEWEY, G. (1923). *Relative Frequency of English Speech Sounds*. Harvard University Press, Cambridge, Mass.

HUFFMAN, D. A. (1952). A method for the construction of minimum-redundancy codes, *Proceedings of the Institute of Radio Engineers*, Vol. 40, pp. 1098–1101.

LEA, R. M. (1978). Text compression with an associative parallel processor, *The Computer Journal*, Vol. 21 No. 1, pp. 45–56.

SHANNON, C. E. (1951). Prediction and entropy of printed English, *Bell System Technical Journal*, Vol. 30, pp. 50–64.

WHITE, H. E. (1967). Printed English compression by dictionary encoding, *Proceedings of the IEEE* Vol. 55 No. 3, pp. 390–396.

WILLIAMS, P. W. (1978). Criteria for choosing subjects to obtain maximum relative entropy, *The Computer Journal*, Vol. 21 No. 1, pp.57–65.

WOLFF, J. G. (1978). Recoding of natural language for economy of transmission or storage, *The Computer Journal*, Vol. 21 No. 1, pp.42–44.

ZIPF, G. K. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Reading, Mass.

Book reviews

The Art of Electronics, by P. Horowitz and W. Hill, 1980; 716 pages. (*CUP*, £35·00, £12·50 paper)

Most technical bookshops carry a shelf full of textbooks on electronics; consequently any new textbook must offer some new material or novel treatment if it is to gain acceptance. This book largely succeeds in this aim by concentrating on the problems of circuit design and construction, and explaining the measures necessary to cope with component and supply tolerances and imperfections. All too many textbooks and university courses deal with ideal components and their behaviour, and give scant coverage to the imperfections of real components and ways of coping with them.

The book starts with chapters on basic components, operational amplifiers, active filters and oscillators, F.E.T.'s and low noise techniques, digital electronics, D-A and A-D conversion. Later chapters deal with systems assembled from these, such as minicomputers and microprocessors, and constructional techniques, high frequency and high speed techniques, measurements and signal processing. There are also 11 appendices, giving information on colour codes, using oscilloscopes, how to draw circuit diagrams, component data sheets, etc.

In all of these topics the text concentrates on the practical aspects of designing reliable and predictable circuits, and indicates the range of components and devices currently available for the task. In general the relative merits of various types of component are well presented, but the paragraph on carbon composition resistors may be a little misleading. After mentioning the well known defects of excess noise and parameter drift, the authors comment on the inductance of this type of resistor. Whilst this may indeed cause problems at high frequencies, because of their 'geometry carbon composition resistors have lower inductance for a range of resistor values than the more precise film resistors with spiral tracks. For this reason they may be preferred for some high frequency applications.

The design methods discussed are largely intended for prototype and low volume production, so little attention is paid to worst case design. Despite these minor points, the book adopts a more practical approach than the general run of electronics textbooks and should thus be of particular interest to engineers moving into the field of circuit design. It should also aid students tackling project work and those trained in other disciplines who have to interface their systems to electronic equipment.

J. C. CLULEY (Birmingham)

Fundamentals of Network Analysis, by D. T. Phillips and A. Garcia-Diaz, 1981; 474 pages. (*Prentice-Hall*, £17.50)

Network analysis is far more than the use of PERT or critical path methods to plan and control projects. A very large number of practical problems can be given network representations in areas as diverse as production sequencing, stock control, manpower planning, and the operation of transportation or distribution systems.

The authors' objectives are to provide a wide ranging review of possible applications, and descriptions of algorithms which have been developed for special network models.

The book begins with an introduction to network concepts and notation and states without proof some of the mathematical results linking networks with the more general class of linear programming problems. Chapter 2 describes several well known applications, including those of transportation, assignment, shortest routes and the infamous travelling salesman. An algorithm is offered for each problem, illustrated by at least one semi-real numerical example. The third chapter considers the versatile 'out-of-kilter' algorithm, which can be applied to many deterministic flow problems. Chapter 4 reviews the methods of critical path analysis and PERT. The book concludes with some topics designated as advanced—networks with flow gains and losses, the GERT approach to stochastic networks and multicommodity flows.

A 40 page listing of a general purpose network optimisation FORTRAN program appears in the Appendix, but it is rather doubtful whether many readers would have the patience to key in a problem of such length, even if the copious comment statements were excluded.

Despite its 474 pages and substantial price, the book is not as comprehensive as the preface suggests. No warnings are given about the invalidity of PERT probability estimates, and it is misleadingly stated that "the travelling salesman problem cannot be directly formulated and solved as a linear program".

In general the writing is clear, but the presence of contributed sections leads to some unevenness in style. The quality of production is good, apart from some incompletely cut pages in the review copy.

The book is aimed at the student market, and is worth considering as an introductory text emphasising network applications and algorithms as an alternative to the more formal graph theoretic approach.

L. CORNER (Newhaven)