Bell Laboratories

Cover Sheet for Technical Memorandum

Title- **Webster's Second on the Head of a Pin**

Date- **July 15, 1974**

TM- **74-1271-13**

Other Keywords- **words**
text compression

Author	Location	Extension	Charging Case- **39199**
Robert Morris	**MH 2C-524**	**3878**	Filing Case- **39199-11**
Ken Thompson	**MH 2C-523**	**2394**	

ABSTRACT

We used the list of words from Webster's Second Unabridged Dictionary (without definitions) as a test case for special purpose text compression techniques.

We compressed it by a factor of 4.52 to 1.

The 234,932 words originally occupied 2,486,781 bytes and were compressed into 549,388 bytes. The size of the decoding program is 1356 bytes.

The initial characters of a word that agreed with the initial characters of the previous word were dropped and replaced by a code. Common suffixes were also coded. Finally, a variable-length code was used.

Pages Text 6	Other 0	Total 6	
No. Figures ---	No. Tables ---	No. Refs. ---	

SEE REVERSE SIDE FOR DISTRIBUTION LIST

Bell Laboratories

Subject- **Webster's Second on the Head of a Pin**

Date- **July 15, 1974**

From- **Robert Morris**
Ken Thompson

TM- **74-1271-13**

MEMORANDUM FOR FILE

The Source Document

There is a copy of the word list from Webster's Second Unabridged Dictionary on the UNIX time sharing system. The word list is stored in ASCII with one byte per character and it occupies 2,486,781 bytes of storage. There are 234,932 entries, separated by new-line characters. The word list is frequently used, and when it is used, it is usually searched from beginning to end.

The list has been modified in several ways so that it does not correspond precisely to the published dictionary. There are an unknown but small number of keypunch errors and inadvertently omitted words. The words which contain non-alphabetic characters (such as hyphen and apostrophe) and the multi-word entries have been omitted. The formal name of the source dictionary is Webster's New International Dictionary, Second Edition (1934).

The alphabet consists of the 26 lower-case letters, the 26 upper case letters, which only appear at the beginnings of words, and the new line character.

The average word length is 10.6 characters, including new-line characters.

The frequencies of the characters in the words are as follows:

nl	234932						
a	196409	n	157628	A	2528	N	650
b	39000	o	169772	B	1352	O	618
c	100857	p	75832	C	2450	P	2243
d	67110	q	3652	D	897	Q	77
e	233902	r	159848	E	898	R	641
f	23640	s	136962	F	470	S	2274
g	46018	t	151050	G	993	T	1520
h	63068	u	87006	H	1095	U	206
i	200128	v	19771	I	484	V	332
j	2662	w	13505	J	413	W	321
k	15523	x	6834	K	489	X	92
l	129149	y	51382	L	1022	Y	139
m	68680	z	8207	M	1822	Z	228

These numbers total to 2,486,781.

Prefix Compression

Since the dictionary is sorted, two adjacent words will agree in some of their initial characters if the upper-case lower-case distinction is ignored.

Our first step was to determine the extent of this agreement and to capitalize on it by replacing the initial characters of a word by a byte which tells the number of initial characters that are the same as those of the previous word (ignoring case). We added a second new-line character to the alphabet that indicates that the first character of the current line is in upper case. This means that the alphabet consists only of the 26 lower-case letters and two new-line characters.

The total savings from introducing counts for the initial characters of words was 1,259,470 bytes. There are then only 1,227,311 bytes in the compressed version, for a compression of 2.03 to 1.

Here is the distribution of the number of initial characters of each word that are the same as those of the previous word. The first group of numbers is for words beginning with a lower case letter and the second is for words beginning with an upper case letter.

0	0	0	26
1	289	1	98
2	2746	2	904
3	14521	3	3409
4	31421	4	4929
5	37525	5	4050
6	34535	6	3285
7	27772	7	2480
8	20592	8	1937
9	15225	9	1290
10	10893	10	841
11	6862	11	546
12	4034	12	248
13	2251	13	128
14	1080	14	56
15	534	15	18
16	241	16	6
17	91	17	2
18	47	18	0
19	11	19	0
20	2	20	1
21	5		
22	1		

The numbers total to 234,932 and the average is about 6.5.

When this method is used, it is no longer possible to begin reading the compressed file at an arbitrary point in the middle and make sense of it. It would be possible to synchronize the compressed text at intervals by inserting 0 counts, but we made no attempt to do so.

Here is a sample of some dictionary words and the coded version of the words:

abdomen	3omen
abdominal	5inal
Abdominales	9es
abdominalian	9ian
abdominally	9ly
abdominoanterior	7oanterior
abdominocardiac	8cardiac
abdominocentesis	9entesis
abdominocystic	9ystic

Variable Length Encoding

The next step was to take advantage of the fact that we were representing a character from a 28-letter alphabet in an 8-bit byte, when in fact it only requires 5 bits. Further, some of the characters are relatively uncommon.

We changed the unit of representation to a 4-bit half byte (called a hexadecimal digit). This provides 16 codes and permits the representation of 16 different letters. We used 14 of these codes to represent the 14 most common letters (including the two new-line characters) and reserved the two remaining codes to indicate that the next hexadecimal digit was needed to represent the letter. Thus we had available 14 single-digit codes and 32 two-digit codes.

The coding scheme that we used is expressed in terms of hexadecimal digits as follows:

0	- escape to 2 digits
1	- escape to 2 digits
2	- a
3	- c
4	- e
5	- i
6	- l
7	- n
8	- o
9	- p
A	- r
B	- s
C	- t
D	- u
E	- new-line (lower case)
F	- new-line (upper case)

This scheme provides for 32 two-digit codes; 14 of these were used for the remaining alphabetic characters.

When coded by this scheme the dictionary occupied about 773,500 bytes for a reduction in space of 3.22 to 1.

Suffix Encoding

We were encouraged by obtaining these results for a trivial investment of time and ingenuity. We pressed onward to more sophisticated techniques.

We obtained a list of all of the suffixes left over when the initial agreeing characters were suppressed and then reversed the order of the characters in the suffixes. The reversed suffixes were sorted and the duplicates were counted. The resulting file consisted of an entry for each

distinct suffix: the suffix (reversed) and its count. This file was used to prepare a list of valuable suffixes to encode. The value of a suffix is the product of the number of times it occurs with the number of hexadecimal digits saved by replacing it by a two-digit code (of which we had some left over). This depends of course on the number of hexadecimal digits (one or two) used to encode its constituent characters.

The most valuable suffixes were assigned codes, and statistics were gathered again. The letter frequencies changed and it turned out to be advantageous to assign some more letters to two-digit codes in order to get more codes for suffixes.

Finally, the suffixes did not need to be followed by explicit new-line characters, since a new-line character could be assumed to occur after every suffix.

The suffixes that were finally used were:
(remember to read them backwards)

ai	eni	lai	retem
am	en	la	re
cit	epocs	luf	ro
ci	er	mrof	sis
c	es	msi	siti
det	eta	mu	ssel
de	eti	m	ssen
di	et	nai	suoref
doo	ev	nam	suo
d	ez	na	su
eadi	foorp	ni	tnem
eaec	gni	noitaz	tn
ecn	g	noita	ts
ec	hc	noit	yl
ed	hparg	noi	ymot
eg	hs	no	yr
eki	h	pihs	ytilib
elba	kc	p	yti
el	k	ralu	y
em	laci	rekam	

The result of encoding suffixes in this way is a compression ratio of approximately 4 to 1.

The suffixes were chosen by hand, and we have not found any way to make an optimal choice of suffixes short of an exhaustive trial. The computation is made more difficult because the choice of one suffix such as "y" affects the value of choosing another suffix, like "ytiliba" which contains it.

Encoding the Count

It is wasteful to use a byte to encode the count when it does not have 256 possible values. In fact, the count and the case distinction were combined into a single code in a very efficient way by the use of a table lookup code. It turns out that over 90% of the counts are encoded in this way along with case specification into a single hexadecimal digit.

The first hexadecimal digit indicated the count and case shift or else indicated that the next digit should be read (escape). The second digit, if needed, indicates the actual count.

first digit	second digit (l.c.)	second digit (u.c.)
0 - l.c. 3	0	0
1 - l.c. 4	1	1
2 - l.c. 5	2	2
3 - l.c. 6	12	8
4 - l.c. 7	13	9
5 - l.c. 8	14	10
6 - l.c. 9	15	11
7 - l.c. 10	16	12
8 - l.c. 11	17	13
9 - l.c. escape	18	14
A - u.c. 3	19	15
B - u.c. 4	20	16
C - u.c. 5	21	17
D - u.c. 6	22	18
E - u.c. 7	23	19
F - u.c. escape	24	20

As it happens, there is no word in the dictionary which is in upper case and has more than 20 initial characters in common with its predecessor. For the delectation of word collectors, the longest common initial string in this dictionary contains 22 characters and occurs between the two words:
Pseudolamellibranchiata, and
pseudolamellibranchiate.

The final character codes needed only one kind of new-line character. These are the single-digit codes:

0 - escape to two digits
1 - ditto
2 - ditto
3 - ditto
4 - ditto
5 - ditto
6 - new line
7 - a
8 - e
9 - i
A - l
B - n
C - o
D - r
E - s
F - t

This provides 96 two-digit codes. There are 17 alphabetic characters that need two-digit codes and this leaves 79 two-digit codes for the suffixes above.

The final result was that the dictionary was compressed into 549388 bytes. This is a compression of 4.52 to 1.

The program that does the decoding occupies 1356 bytes. Purists may argue that the size of the decoding program should be added to the size of the compressed text to determine the compression. When this is done, the ratio remains at 4.52.

Further Experiments

It is clear that further improvements could be made in a number of ways. The encoding of the new-line character and of the count could be slightly improved to save a few thousand bytes. More suffixes could be added, displacing some characters from the single-digit part of the code, for an improvement of 5,000 to 10,000 bytes. These slight improvements were not worth the effort to install them.

We tried exactly the same kind of compression process on a dictionary prepared by reversing the order of the letters in each of the words and then sorting the result. This amounts to a rhyming dictionary. The results were surprisingly poor and it appeared that we could not obtain a compression ratio greater than 4 to 1.

Robert Morris

Ken Thompson