STORING TEXT USING INTEGER CODES

Raja Noor Ainon
Computer Centre
University of Malaya
59100 Kuala Lumpur, Malaysia.

Abstract

Traditionally, text is stored on computers as a stream of characters. The goal of this research is to store text in a form that facilitates word manipulation whilst reducing storage space. A word list with syntactic linear ordering is stored and words in a text are given two-byte integer codes that point to their respective positions in this list. The implementation of the encoding scheme is described and the performance statistics of this encoding scheme is presented.

1.0 Introduction

This research aims at storing text in a form that facilitates word manipulation whilst saving storage space. Although there are many text compression algorithms currently in use, the word manipulation capability has yet to be incorporated. Harris [2], in his research, compiled a 40,000 word list with syntactic linear ordering and suggested that words in a text be given two-byte integer codes that point to their respective positions in this list. In this way the coded text has inherent syntactic information, thereby making it useful for many applications including statistical linguistics and question-answering systems. In this paper, we show how such a scheme can achieve optimal compression results and present its efficient implementation.

Three text compression techniques that have been tested on English texts include (1) the Huffman variable-length encoding schemes [3] which achieve tight packing of data by giving variable-length bit code for each character, with more frequently-occurring words having shorter codes, (2) the Adaptive pattern substitution algorithm, also known as the Lempel-Ziv or LZW algorithm (see [7], [8], [9]) which converts variable-length strings of input symbols into fixed length codes by first looking for common patterns of two or more bytes occurring frequently, and then substituting an unused byte for the common long one, and (3) another technique, due to Hahn [1] encodes non-blank characters in groups of a fixed size as unique fixed point numbers.

For measuring text compression two definitions are used in this paper. One is the compression ratio defined as the ratio of size of the original text to that of the coded text. The other is the compression percentage, defined as

$$\frac{\text{(Size of original text - Size of coded text)\%}}{\text{(Size of original text)}}.$$

2.0 The Two-Byte-Word Encoding Scheme

In most computers, text is stored as a stream of characters - made up of alphabtes, spaces, digits and punctuation marks. Each word is separated from neighbouring words by delimiters such as spaces, special characters or punctuation marks. On the other hand, a number (integer or floating point) regardless of its size, is treated as a unit and is represented in computers in the form of a computer word (usually 4 bytes) or part of a word. It is addressable, hence it does not require a delimiter like a space to distinguish it from a neighbouring number.

For this encoding scheme, we endeavour to store text as a stream of fixed length computer words which is distinguishable by the computer. This can be achieved by keeping an external list of all words in the dictionary including derived ones, and assigning a unique integer code for each entry. Instead of words separated by delimiters the coded text represents words as numbers, thereby dispensing the need for representing delimiters.

In using two bytes to represent an integer it is possible to have $2^{16} - 1 = 65536$ distinct codes. However, since it is impossible to have codes for all the words in the English Language, it is necessary to include a mechanism that allows for the representation of words without codes by their individual characters. Keeping one bit for that purpose leaves $2^{15} - 1 = 32767$ possible number of combinations. Two adjoining character codes (ASCII or EBDIC) always have zero as the first bit and is therefore read as a positive integer. The first bit, being a sign bit, can be used to indicate whether the two bytes represent a code (negative integer) or two characters, as follows: -

1XXXXXXX	XXXXXXXX	a code
0XXXXXXX	XXXXXXXX	two characters

It is also necessary to show that compression can indeed be achieved for this encoding scheme. In several studies it has been shown (see Kucera [4], for example) that the word frequency distribution in natural language analysis is highly-skewed. Assuming a skewed distribution it may be seen that the 32000 most-frequently occurring words from the Cobuild* corpus constituting 60% of the corpus, account for 99% of the total word usage. Including these 32000 words in the list will imply that words without codes (not included in the list) makes up 1% of the text. Assuming that the average size of an English word [4] consists of 4.7 characters we would expect that an average occurring word would occupy, taking one more byte for a trailing space, 5.7 bytes. Thus the compression ratio is $[(2/5.7)* 0.99 + 0.01]^{-1} = 2.8$, meaning that a coded text is 35.7% of the original text.

*The corpus used in the Cobuild Project, a project in the English Department, University of Birmingham, is made of 6 million words of written text and 1.3 million words of transcribed speech.

The encoding scheme was implemented on a Honeywell computer using MULTICS operating system. Since MULTICS represents characters using ASCII with the nine bits per character, two bits are not being utilized. For our implementation, one bit is used to indicate words beginning with upper-case characters (proper names, etc) and the other available bit is kept for future development.

3.0 The Word List

Harris [2] has constructed a word list with linear ordering in a sense that all words in a group are derived forms of a baseword (the first word in the group), and its relative position implies its syntactic information (see figure 1). Because the number of derived forms is not regular, the size of a group varies. Even though the positional notation cannot be used in a strict sense as in numbers because of the length variability of the groups, the relative position of a member in a group can still provide its syntactic information.

For a comprehensive and consistent word list, some words which are not in the top 32000 have had to be included, resulting in a larger word list. As the size of the integer codes cannot exceed 32768, the whole word list may have to be reduced by excluding some words in the top 32000.

In using the above word list, it is found that two problems may arise due to (1) the occurrence of homographs (words identical in spelling but different in pronunciation and meaning) and homologs (words identical in spelling and pronunciation but different in meaning) and (2) the occurrence of words having the same meaning but different syntax. If different codes are given to the duplicate words the encoding process would need an intelligent parser to be able to differentiate between the two. Such a parser, though not impossible to implement, is not cost-justifiable for this study; hence the only alternative is to assign the code for both words and to allow for ambiguity when the coded text is used for analysis.

Set A (4 verb forms)

arch	arches	arching	arched
bid	bids	bidding	bid
round	rounds	rounding	rounded

Set B (4 verb forms + 1 noun)

abandon	abandons	abandoning	abandoned	abandonment
acclaim	acclaims	acclaiming	acclaimed	acclamation
consent	consents	consenting	consented	consent
.				
.				
.				

Fig. 1

4.0 Implementation

For encoding, a code table comprising more than 32000 distinct words in the word list indicating their codes is stored in an indexed file using the words as keys. For faster encoding, the top 200 words from the Cobuild corpus are stored in a hash table. During encoding this hash table is searched before searching the code table, therefore saving execution time when encoding common words.

For word manipulation of the encoded text the word list needs to be structured in order that the syntactic information is captured. That is, the group and the set (*set-type*) to which the word belongs and the relative position of the word in the group needs to be indicated. For our implementation a linked list is employed to store a word which has links to the base word (the first word in the group) and the next word in the group and information containing its *set-type*. Each node in the linked list is stored as a record in an indexed file using the codes as keys. In this way each word in a group can be retrieved individually and the group can be synthesized from tracing the links to the next word. There is further gain in using the codes as the search key in that the same file can be used for the decoding process.

5.0 Some Sample Statistics

Table 1 gives the performance statistics of the two-byte-word encoding scheme on four English texts. Comparison of compression ratios with other techniques is shown in Table 2.

Table 1 Performance statistics of the two-byte-word encoding scheme on several English texts

Text	Text 1[a]	Text 2[b]	Text 3[c]	Text 4[d]
1. Size of input text (in bits)	2,927,745	2,542,977	3,605,931	2,840,193
2. No. of Tokens	6005	5789	7844	7148
3. No. of uncoded words	1120	1817	2860	2069
4. Compression Ratio	2.08	2.02	2.00	1.90
5. Compression Percentage	51.93%	50.47%	49.87%	47.42%
6. Percentage of Text that is coded	91%	78%	77%	81%
7. Encoding Time (in Secs)	56	58	78	64
8. Word Frequency Count of Coded Text (in Secs)	37	39	52	45
9. Word Frequency Count of Uncoded Text (in Secs)	187	178	239	not available

a. "Small is Beautiful" by E.F. Schumacher
b. "Baby and Child Care" by Dr. B. Spock
c. "The Third World War" by Sir John Hacket
d. "The Americans" by A. Cooke

Table 2 Comparison of the Two-Byte-Word Encoding Scheme with other Algorithms

Technique		
Pechura	[5]	1.5
Welch	[7]	1.8
Huffman	[3]	1.9
Two-byte-word		2.0
Hahn	[1]	2.4
Rubin	[6]	2.4

From Table 1 it is observed that on the average the compression percentage is 82%. Although short of the 99% mentioned in section 2.0, this is expected,since not all of the top 32000 words have been included in the word list. The compression ratio as shown in the fourth row has a mean of 2.0. Comparing the last row with the sum of the 7th and 8th row it is seen that the speed for word frequency count for the coded texts is much faster than that of the coded texts.

6.0 Some Practical Applications

For purposes of storing compressed text, the two-byte-word encoding scheme can be used independent of the word list and some results are shown in the previous section. It may be seen that the performance of the scheme is comparable with other well-known techniques (see Table 2). With the word list the scheme is capable of word manipulation and therefore it can be used for more intelligent applications. For example, the scheme has been used for obtaining the lemmatized word count of several large texts - an almost impossible task when done manually. No comparison results is shown simply because no similar system currently exists.

Because the words in the coded text are represented by integers, word comparison - a common task in linguistic research involving comparison of character by character - becomes comparison of two numbers which is a quick and simple operation on digital computers. This is reflected quite dramatically on obtaining the word frequency count of the coded text.

Acknowledgements

I wish to thank Professor John Sinclair of the English Department, University of Birmingham for the use of research facilities in the Cobuild project. I am also grateful to the University of Malaya for providing the funds for my stay in Britain.

References

1. Hahn, B., "A new technique for compression and storage of data", CACM Aug. 1974, Vol. 17, No.8, pp. 434-436.

2. Harris, S., research report, English Language Department, University of Birmingham.

3. Huffman, D.A., "A method for the construction for minimum redundancy codes" Proc. IRE 40, 40, 9 (Sept. 1952), 1098-1101.

4. Kucera, H. and Francis, W.N., "Computational analysis of present-day American English", Brown Univ. Press 1967.

5. Pechura, M.," File archival techniques using data compression", CACM, Vol. 25, No. 9, Sept. 1982, pp. 605-609.

6. Rubin, F., "Experiments in text file compression" CACM, Vol. 19, No. 11, Nov. 1976, pp. 617-623.

7. Welch, T.A., "A technique for high-performance data compression", IEEE Computer, June 1984, pp. 8 - 19.

8. Ziv, J. and Lempel, A., "A Universal algorithm for sequential data compression" IEEE Trans. Information Theory, Vol. IT-23, No. 3, May 1977, pp. 337-343.

9. Ziv, J. and Lempel, A., "Compression of individual sequences via variable-rate coding", IEEE Trans. Information Theory, Vol. IT-24, No. 5, Sept. 1978, pp. 5306.