INFORMATION AND CONTROL 3, 327–334 (1960)

On the Efficiency of a New Method of Dictionary Construction

A. D. BOOTH AND A. J. T. COLIN

Department of Numerical Automation, Birkbeck College, University of London

Programs for constructing dictionaries of texts, with computers, have sometimes been adaptations of methods suitable for manual construction with card indexes. With all card index methods it is customary to keep the different words collected in alphabetical order, for the structure of a card index lends itself to such a process: all that is necessary to insert a new word into the index between two existing ones is to make out a new card and to put it in the correct place. However, the insertion of a new word in the store of a computer where the words are kept in alphabetical order is a time-consuming process, for all the words below the one which is inserted have to be "moved down" by one place.

If, however, a computer method is used where the words are not stored in alphabetical order but in the order in which they occur, the position is even worse, for although the shifting of words is eliminated, the dictionary search which is necessary for each word in the text, to establish whether it is a new word or not, must involve all the words collected so far, and not just a small number of them, as would be the case if the words were in alphabetical order, and a logarithmic search (Booth, 1955) could be used.

The method of construction described in this paper overcomes both these problems. It is not an adaptation of a manual method, but is designed specifically for computers. The method is based on a tree structure, which is discussed below.

THE LOGICAL TREE

The logical tree is based on the fact that if α and β are two different words, α is either alphabetically less or alphabetically greater than β. The construction of a tree is illustrated by taking a sentence and making a tree from it.

"Friends, Romans, countrymen, lend me your ears;
I come to bury Caesar, not to praise him."
(*Julius Caesar*, III ii 81–82)

Stage 1: Place the first word in the center of the page, thus:

FRIENDS

Stage 2: Compare the second word with the first word. It is greater, so place it below and to the right of the first word, with a connecting arrow (a "forward" arrow)

FRIENDS
↘
ROMANS

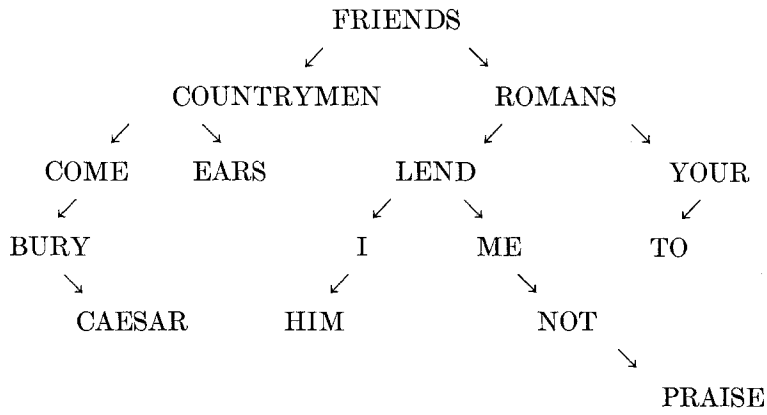
Stage 3: Similarly with the third word, except that a "backward" arrow is used.

FRIENDS
↙ ↘
COUNTRYMEN ROMANS

Stage 4: Compare the fourth word with the first word; it is greater, but a forward arrow from the first word already exists, so compare the fourth word with the word at the end of this arrow and record it appropriately, thus:

FRIENDS
↙ ↘
COUNTRYMEN ROMANS
↙
LEND

Stage 5 and onwards: Repeat the process with successive words from the text. The final result is:—



This process, which becomes unwieldy on paper if the text is long, is easily applied to a computer. Each new word is recorded in any unoccupied position in the store, space being left for the forward and backward references which replace the arrows. The address of the location where any new word is stored is itself stored in the appropriate blank space associated with the word with which the new word was last compared.

AN ANALYSIS OF THE EFFICIENCY OF THE PROCESS

The tree structure is as shown in Fig. 1. The apex of the tree will be called level 1 and it is clear that the ith level can contain, at most 2^{i-1} entries.

FIG. 1

Suppose that, at some stage, a tree constructed according to the principles described above contains N words. Let $n_{i,N}$ be the number of words at level i for this tree, then clearly:

$$\sum_{i=1}^{N} n_{i,N} = N. \tag{1}$$

Again, if a tree contains N words, the next word may be added in $N + 1$ possible places, for the addition of each new word fills one avail-

able site but creates two new ones. Thus the net result of adding a word is to increase the number of available sites by 1; since initially there is only one site (the apex of the tree), the result follows by induction.

To derive the probable structure of the tree at any stage it is necessary to make some assumption regarding the distribution of words in running text. The particular assumption which we make here is that any new entry to the tree is equally likely to occupy any of the vacant sites. This means that the probability of any particular site being filled when the tree contains $(N - 1)$ words is $1/N$.

We define the efficiency of the given process, η, in the following way:

$$\eta = \frac{C_{PN}}{C_N} \tag{2}$$

where C_{PN} = number of references required to reach each entry in a perfect (i.e. completely branched) tree with N entries;
C_N = number of references required to reach each entry in a given tree when it contains N entries.

For a completely branched tree, clearly,

$$C_{PN} = \sum_{i=1}^{M+1} i \cdot n_{i,N}$$

where M = integral part of $\log_2(N)$. Now,

$$n_{i,N} = 2^{i-1} \quad (i < M + 1)$$
$$n_{M+1,N} = N - (2^M - 1)$$

whence

$$\begin{aligned} C_{PN} &= (M + 1)(N + 1 - 2^M) + \sum_{i=1}^{M} i \cdot 2^{i-1} \\ &= (M + 1)(N + 1 - 2^M) + (M - 1)2^M + 1 \\ &= (M + 1)(N + 1) - 2^{M+1} + 1. \end{aligned} \tag{3}$$

The situation with respect to C_N is more complex. Suppose that the construction of the tree has proceeded until it contains $(N - 1)$ entries and that a further entry is about to be added. It is easy to see that the number of vacant sites at level i is simply

$$2n_{i-1,N-1} - n_{i,N-1}$$

whence, since the $(N - 1)$ entry tree contains N equiprobable sites as

shown above, the probable "number" of sites added to level i when a new word is added to the $(N-1)$ word tree is

$$\frac{1}{N}(2n_{i-1,N-1} - n_{i,N-1}).$$

It follows that

$$\begin{aligned} C_N &= C_{N-1} + \sum_{i=2}^{N} \frac{i}{N}(2n_{i-1,N-1} - n_{i,N-1}) \\ &= C_{N-1} + \frac{1}{N}\sum_{i=2}^{N} i(2n_{i-1,N-1} - n_{i,N-1}) \\ &= C_{N-1} + \frac{2}{N}\sum_{i=1}^{N-1} i.n_{i,N-1} + \frac{2}{N}\sum_{i=1}^{N-1} n_{i,N-1} - \frac{1}{N}\sum_{i=2}^{N} n_{i,N-1} \end{aligned}$$

whence, using (1), and noticing that $n_{N,N-1} = 0$,

$$C_N = \left(1 + \frac{1}{N}\right) C_{N-1} + 2 - \frac{1}{N}. \tag{4}$$

Now (4) is a difference equation whose solution is easily shown to be

$$C_N = 2(N+1)\sum_{r=1}^{N} \frac{1}{r} - 3N \tag{5}$$

which provides a simple method for calculating C_N for any value of N.

For the present investigation, however, we are interested in the behavior of C_N for large values of N and hence the approximation

$$\sum_{r=1}^{N} \frac{1}{r} \approx \log_e N + \gamma \tag{6}$$

where $\gamma = 0.5772 \cdots$ is appropriate.

Thus, using (2), (3), (5), and (6) we have, for large N,

$$\eta = \frac{C_{PN}}{C_P} \approx \frac{1}{2\log_e 2} = 0.721 \cdots. \tag{7}$$

A MODIFICATION AND ITS EFFECT

One way in which the tree construction process just described can be improved is to arrange, artificially, for the first i (say) levels of the tree to be perfect in structure and divide the "interval" (A,Z) into 2^{i-1} equal parts.

The effect of this is to replace the recurrence relation (4) by

$$C_{i,N} = \left(1 + \frac{1}{N}\right) C_{i,N-1} + 2 - \frac{(i-2)2^{i-1}}{N} \qquad (N > 2^i - 1) \quad (8)$$

with

$$C_{i,2^i-1} = (i-1)2^i + 1.$$

The solution of (8) is

$$C_{i,n} = (N+1)\left(\frac{i}{2} + \frac{1}{2^i}\right) + 2 + (i-2)2^{i-1} + 2(N+1)\left(\sum_1^N \frac{1}{r} - \sum_1^{2^i} \frac{1}{r}\right) \quad (9)$$

whence, using the approximation (6)

$$C_{i,N} \to 2(N+1)\ (\log_e N - \log_e 2^i)$$

so that, for large values of N, the ratio

$$\eta = \frac{C_{PN}}{C_{iN}}$$

again tends to 0.721 $\cdots$. For values of 2^i which are not too far from N, however, an appreciable improvement can result. Thus, in an experiment which used a text containing about 300 different words, the presence of a starting sequence with $i = 5$ improved the efficiency factor by about 17 %.

THE EFFECT OF ZIPF'S LAW

One attractive feature of the tree dictionary is that it enables use to be made of Zipf's law (Zipf, 1949) to reduce the number of access operations required to locate the most frequent words.

The magnitude of the improvement which is to be expected can be seen from the following analysis. Suppose that the words are arranged in order of frequency in the following way:

.1

½. .⅓

·¼ ·⅕ ·⅙ ·⅐

Taking account of this frequency distribution, the number of look-up operations required to visit each word in an M level tree is thus

$$1 + 2(\tfrac{1}{2} + \tfrac{1}{3}) + 3(\tfrac{1}{4} + \tfrac{1}{5} + \tfrac{1}{6} + \tfrac{1}{7}) + \cdots + M \sum_{2^{M-1}}^{2^M - 1} \frac{1}{r}$$

To reduce this number to the same basis as those previously used (i.e., visiting exactly the $N = 2^M - 1$ words in the tree) this number must be multiplied by a normalizing factor

$$(2^M - 1) \Big/ \sum_{r=1}^{2^M-1} \frac{1}{r}.$$

We thus obtain for the "Zipf count," $C_{Z,N}$

$$C_{Z,N} = \frac{2^M - 1}{\sum_{r=1}^{2^M-1} \frac{1}{r}} \left[1 + 2(\tfrac{1}{2} + \tfrac{1}{3}) + 3(\tfrac{1}{4} + \tfrac{1}{5} + \tfrac{1}{6} + \tfrac{1}{7}) + \cdots + M \sum_{2^{M-1}}^{2^M - 1} \frac{1}{r} \right] \tag{10}$$

Again, making use of the approximation (6), this becomes

$$C_{Z,N} \approx \frac{2^M - 1}{\log_e(2^M - 1) + \gamma} [1 - \log_e(2^1 - 1) - \log_e(2^2 - 1) \cdots - \log_e(2^{M-1} - 1) + M \log_e(2^M - 1)]$$

or, for large N,

$$C_{Z,N} \to MN/2$$

whence

$$\eta = \frac{C_{PN}}{C_{ZN}} \to 2.$$

CONCLUSION

It is clear from the above discussion that the new method of dictionary construction can offer advantages not only in the actual compilation stage but also in practical use. Assuming that the tree method is used without a special starting procedure, it seems likely that the resulting efficiency will be about $2 \times 0.721 = 1.422$ times that which would be achieved by using a standard linear dictionary and the partitioning method.

Some practical details of the programming techniques which are required to use the method on an automatic digital calculator and statis-

tical data obtained during an application of the method in textual analysis will be presented in a later paper.

RECEIVED: July 25, 1960.

REFERENCES

BOOTH, A. D., (1955). *Nature* **176,** 565.

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