American Journal of Computational Linguistics Microfiche 75: 43

ONE MORE STEP TOWARD COMPUTER LEXICOMETPY

NICHOLAS V. FINDLER AND SHU-HWA LEE

Department of Computer Science
State University of New York at Buffalo
4226 Ridge Lea Road
Amherst, New York 14226

ABSTRACT

We describe the continuation of an earlier work on the problem of lexical coverage. The objective is to prove experimentally certain mathematical conjectures concerning the relationship between the sizes of the covering and covered sets of words, ~~and the~~ maximum length of dictionary definitions. The data base on which the experiments are carried out has been also extended to the full contents of an existing dictionary of computer terminology. The results of the previous and present work lay the foundations for quantitative studies on lexical valence and its relation to the frequency of usage and other principles of dictionary selection.

Besides the inherent interest in these investigations, the concepts dealt with and the methods of quantifying dictionary variables may eventually lead to more efficient dictionaries with respect to precision, compactness, and computer time and memory needed for processing.

Supported by NSF Grant MCS 76-24278.

INTRODUCTION

First, we shall introduce the problem define some basic terms and provide a brief historical account of past results. In order to render this paper fairly self-sufficient, a brief summary of the previous work, Findler Viil (1974), will also have to be given.

A monolingual dictionary may be considered economical and efficient if a small set of words are used to define a relatively large set of entries. Quantitative information as to what size vocabulary is needed to cover a given number of entries is very scarce and may be characterized by two "data points":

The New Method English Dictionary published by M.P. West and J.G. Endicott in 1961 uses 1,490 self-defined basic words to explain some 18,000 words and 6,000 idioms, i.e. about 24,000 expressions. Thus, the size ratio is 0.062.

Ogden's Basic English, published in 1933, involves 850 English words and 50 "international" words to define 20,000 English words. The ratio of the covering and covered set sizes is 0.045.

The basis of selection was the "usefulness" of the words employed in the definitions, as opposed to the frequency of their occurrence in some standard texts. However, neither this concept nor other principles of selection suggested by other researchers have ever been quantitatively analyzed and made use of. We shall discuss these issues later on.

In order to approach the problem in definite terms, Findler (1970) considered three basic variables:

(i) the covering set, $\underline{R}$, of size $v_{\underline{R}}$,

(ii) the covered set, $\underline{S}$, of size $v_{\underline{S}}$,

(iii) the maximum definition length, $\underline{N}$, such that each word in $\underline{S}$ can be defined by at most $\underline{N}$ ordered words from $\underline{R}$.

The task was formulated to find

(a) $v_{\underline{R}}$ as a function of $v_{\underline{S}}$ at different parametric values of $\underline{N}$, and

(b) $v_{\underline{R}}$ as a function of $\underline{N}$ at different parametric values of $v_{\underline{S}}$

Calling $\Delta v_R/\Delta v_S$ increment ratio and v_R/v_S size ratio, the following conjectures were made concerning the first task:

(a1) The increment ratio is, in general, less than one.

(a2) The increment ratio, in general, decreases as v_S increases.

(a3) For large constant values of $\underline{N}$, $v_{\underline{R}}$ approaches a limiting value asymptotically as v_S increases.

(a4) The increment ratio never exceeds the size ratio.

Two points need to be noted in this connection. An exception to rules (a1) and (a2) would occur in a dictionary system, which does not treat polysemous words or homonyms as individual entries, every time a new word with many meanings or homonyms is introduced into the covered set. Second, the cited case is an exception to rule (a1) but not to (a4). When N=1, the covering and the covered sets are of the same size, i.e. both the increment ratio and the size ratio equal one. However, not every word is defined by itself only. If a new word is introduced that already has a synonym in the covering set, it will be defined by that synonym. In this case, the increment ratio is 0 and the size ratio becomes less than 1. (This will be clear with the description of the data base construction on page 11.)

For the second general task, (b) the following conjectures were also made:

(b1) $\underline{v_R}$ monotonically decreases as $\underline{N}$ increases.

(b2) For any fixed value of $\underline{v_S}$, $\underline{v_R}$ asymptotically approaches a lower limit as $\underline{N}$ increases without bound.

It seems reasonable to state in a qualitative sense that in the process of generating a dictionary smaller $\underline{v_R}$ values mean smaller storage requirements whereas smaller $\underline{N}$ values tend to reduce processing time and output volume. In order to answer the question "What are the optimum values of $\underline{v_R}$ and $\underline{N}$ for a given $\underline{v_S}$ for a certain (family of) computer applications on a machine with a given cost structure?' one has to consider the interrelation of the above three basic variables and to compute three entities: the <u>semantic index</u> (roughly, the number of different meanings) of the elements in the covered set, the <u>lexical valence</u> (roughly the capability of being substituted for another word) of the elements in the covering set, and the <u>frequency of occurence</u> of the elements of both sets. Quantitative investigations of the last three dictionary variables are planned to follow the present, second stage of our study.

<u>THE DATA BASE AND THE PROGRAM</u>

We have extended the data base used in our previous work, Findler and Viil (1974). The whole contents of the dictionary on computer technology, Chandor (1970), is now included in the present study. Its structure, rather simple and uniform, is described below. First, some general principles of data base construction are outlined.

Every element of the covered set is considered a single lexical item, regardless of the number of words the original dictionary entry consists of. Also, each word is coded as a string of at most 10 characters (containable in one CDC Cyber computer word). The abbreviations are still easy to read with relatively short practice.

Only the dominant meaning of polysemous terms was dealt with. Each entry has thus one meaning and one definition.

Terms in the definitions (elements of the covering set) are also considered lexical items, i.e. even multiword entities appear as a single unit and are represented by at most 10 characters.

The basic vocabulary, that is the covering set, consists of elements that also appear in the covered set. In our particular case, they are non-technical words used to define the technical terms of the computer dictionary. A definite distinction was made between <u>content words</u> and <u>function words</u> (also called operators). The latter were not included in the covering set nor were they counted in determining the length of definitions. Hence, the covering set consists only of content words.

The function words indicate grammatical and logical relationships between the words contributing to the content.

They belong to 11 categories:

1) prepositions, e.g. of, in, to;

2) conjunctions, e.g. and, or, if;

3) the relative pronoun which;

4) combinations of preposition and relative pronoun, e.g. in which, to which, by which;

5) present participles equivalent to a preposition, e.g. using, containing, representing;

6) combinations of participle and preposition, e.g. consisting of, opposed to, applied to;

7) combinations of adjective and preposition, e.g. capable of, exclusive of, equal to;

8) combinations of noun and preposition, e.g. part of, set of, number of;

9) combinations of preposition, noun, and preposition, e.g. in terms of, by means of, in the form of;

10) prepositional phrases associated with a following infinitive, e.g. used to, necessary to, in order to;

11) other frequently used purely functional expressions, e.g. for example, namely, known as.

Actually, the function words were replaced by code numbers in the dictionary. The code numbers were assigned consecutively as the function words were needed during the construction of the data base so that the order is purely random. A complete list of the 121 function words used, together with their code numbers, is given in Table I.

- -

INSERT TABLE I ABOUT HERE

- -

The original definitions were somewhat simplified and standardized. In this process, articles were omitted (many languages do very well without them). On the other hand, implicit relationships were made explicit. Nouns are represented in singular, thus avoiding another dictionary entry for plural or, what would be worse, programming a "grammar". Likewise, finite verb forms are represented in third person plural present indicative active. Avoiding the third person singular eliminates another dictionary entry, and avoiding the passive voice eliminates a great many participles, which otherwise would have had to be entered. Of course, present and past participles (the former identical to gerund in form) could not always be avoided and had to be entered in the dictionary where needed. Auxiliary verbs were automatically eliminated by avoiding compound tenses and the passive voice. Finally, "to do" associated with negation was simply omitted.

Some examples will make the encoding process clear.

Original dictionary entry:

<u>aberration</u> A defect in the electronic lens system of a cathode ray tube.

Definition in the data base:

DEFECT (in) SYSTEM (of) ELECTRONIC LENS (of) CATHRAYTUB

1. is equivalent to
2. of
3. in
4. in terms of
5. using
6. and
7. which
8. in which
9. between
10. to
11. or
12. from
13. used to
14. necessary to
15. part of
16. consisting of
17. containing
18. capable of
19. by means of
20. opposed to
21. when
22. on
23. so that
24. in order to
25. exclusive of
26. for
27. pertaining to
28. under
29. as
30. such as

62. if
63. among
64. by
65. namely
66. related to
67. concerned with
68. based on
69. constituting
70. resulting from
71. set of
72. including
73. followed by
74. provided by
75. developed by
76. assigned to
77. referred to
78. on which
79. used as
80. in the form of
81. from which
82. into which
83. number of
84. less
85. defining
86. known as
87. performing
88. performed by
89. independent of
90. chosen by
91. for which

31. equal to
32. into
33. with
34. according to
35. applied to
36. depending on
37. to which
38. whose
39. obtained by
40. inherent in
41. through
42. during
43. where
44. during which
45. out of
46. at
47. by which
48. used in
49. without
50. caused by
51. over
52. not
53. but
54. extended to
55. so as to
56. for example
57. represented by
58. along which
59. representing
60. against which
61. similar to

92. at which
93. whether
94. used by
95. about
96. before
97. per
98. having
99. formed by
100. around
101. after
102. since
103. against
104. until
105. whereupon
106. except
107. determined by
108. over which
109. in relation to
110. belonging to
111. corresponding to
112. due to
113. required for
114. type of
115. across
116. because
117. designed
118. indicating
119. produced by
120. outside
121. towards

TABLE I
List of Function Words

Note that "electronic lens system" (should be: electronic-lens system) means "system of electronic lens" (as opposed to "electronic system of lens"), and this relationship is made explicit. Note also that "cathode ray tube" is a single lexical item.

Original dictionary entry:

<u>absolute coding</u> Program instructions which have been written in absolute code, and do not require further processing before being intelligible to the computer.

Data-base entry: ABSOCODING

Definition:

PROGRAM INSTRUCTIO (which) ONE WRITE (in) ABSOLUCODE (and which not) REQUIRE FURTHER PROCESSING (before) INTELIGIBL (to) COMPUTER

Note that the first predicate in the relative clause, third person plural perfect indicative passive, is represented by the singular indefinite pronoun "one" as subject, followed by the standard plural active verb. The auxiliary "do" has been omitted and the negation is represented by a function word. The virtually redundant "being" has also been left out. In general, the copula is omitted (some languages do very well without it).

Original dictionary entry:

<u>analytical function generator</u> A function generator in which the function is a physical law. Also known as natural law function generator, natural function generator.

Data-base entry: ANLYTFNGEN

Definition:

FUNCGENRTR (in-which) FUNCTION PHYSICAL LAW

Note also the omission of the gloss "Also known as . . ."

The stylized definitions are easily understandable even to human readers as the printout of the dictionary demonstrates.

The data base was constructed by selecting the first entry, then entering all the lexical items in its definition, subsequently entering all the lexical items in the definitions of these, etc. Words that were not defined in the original dictionary were entered and defined by themselves; they constitute the basic vocabulary. This procedure was continued until everything was defined, i.e. until all the terms in the covering set were also in the covered set. Then the next entry was selected from the dictionary, and the above process was repeated.

The dictionary was arranged in the form of a SLIP list, Findler et al. (1971). Every entry (element of the covered set) occupies four cells in this list: (1) entry word (as character data, using FORTRAN format specification A10), (2) definition length (an integer), (3) type of entry (an integer), (4) sublist name.

Three types of entries were distinguished for programming convenience:

1) code 0 indicates that the entry itself is not used in any definition i.e. it occurs only in the covered set and not in the covering set;

2) code 1 indicates that the entry occurs in both sets and is not an element of the basic vocabulary;

3) code 2 indicates that the entry is defined by itself, i.e. it belongs to the basic vocabulary.

The sublist the name of which is in the fourth cell for every entry in the main list, contains the definition. This arrangement conveniently separates the entry words from those in the definitions.

A cell in this second level contains either a word (in A10 format), i.e. an element of the covering set, or a sublist name. The codes for function words (integers) are contained in the cells in the third level. This arrangement is convenient for bypassing the function words in processing when they are not needed. The general dictionary entry and an example thereof are illustrated in Figure 1.

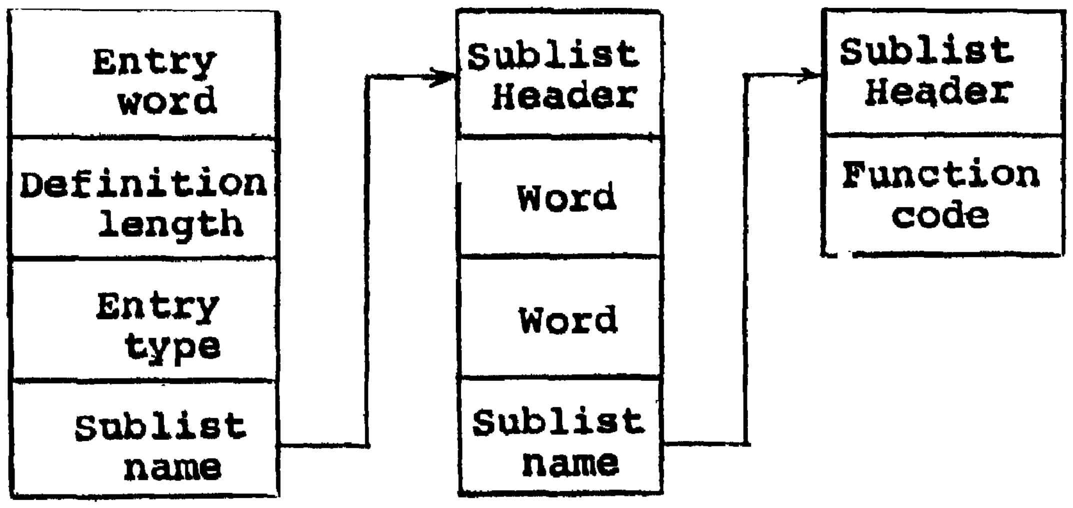
- -

INSERT FIGURE 1 ABOUT HERE

- -

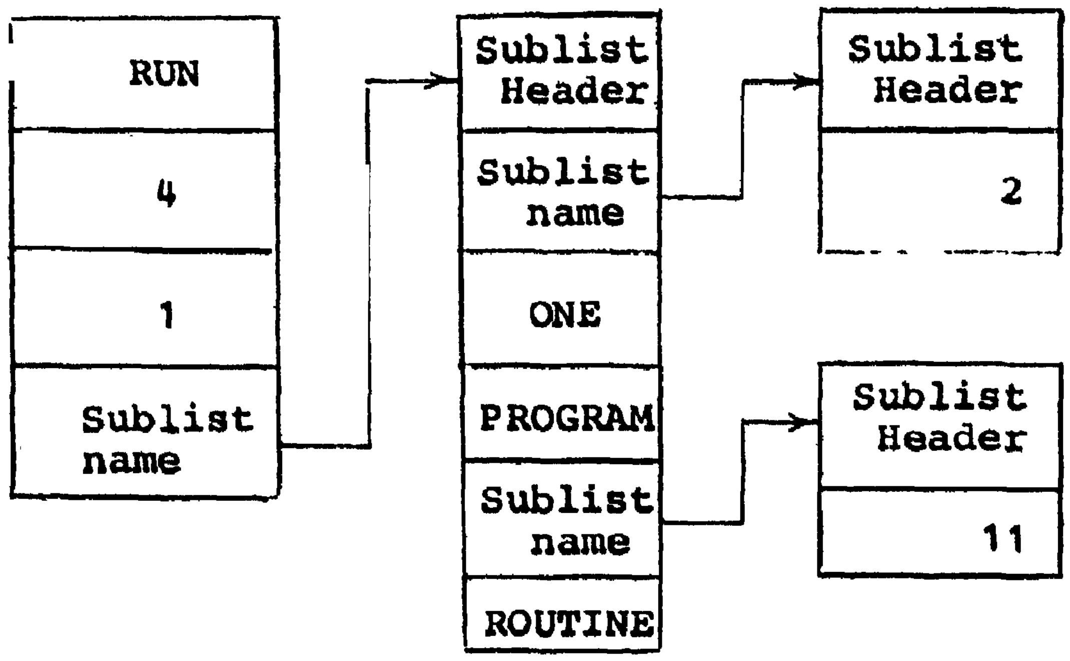
The fact that every dictionary entry owns a sublist is practical in another respect: useful information about the entry can be collected and deposited in a description list associated with the sublist. For example, if it were desired to evaluate the definition component of the lexical valence of each lexical item, a program could be developed that counts how many times a particular item occurs in the definition of other items and stores this information in the description list created for that item. Investigations of this nature will be done subsequently.

The task is to establish experimentally the relationship between $\underline{N}$ and $\underline{v_R}$ for fixed values of $\underline{v_S}$. The program starts out with the values of some fixed data point obtained in the previous



Data Structure for a Dictionary Entry

FIGURE 1a



An Exemplary Dictionary Entry

RUN =: PERFORMANCE of (=2) ONE PROGRAM or (=11) ROUTINE
Definition length: 4; Entry type: 1.

FIGURE 1b

study, Findler and Viil (1974), or one calculated for the extended data base The size of the covering set, $v_{\underline{R}}$ is then systematically reduced. Code 1 type words are replaced by their definitions of length 1, 2, 3, 4, etc. (Recall, code 1 means that such entries are not defined by themselves and occur both in the covering and the covered set.) After the substitutions are made in all definitions and the words are counted out of $\underline{v_k}$, the corresponding $\underline{N}$ values are ascertained. The process is repeated for different size covered sets, i.e. $v_{\underline{S}}$ is kept at different constant levels for each run. (We note that a quantitatively more satisfactory refinement could have been added to the dictionary-reduction program. Each basic word would be compared with all the remaining definitions, and those which do not appear in any definition are to be eliminated. Thus a basic word would occur in the dictionary only if it is needed in a definition, which is the case in the unreduced dictionary. This way, a more natural proportion between the basic words and others could be restored. However, in the present preliminary work, we did not wish to pay the considerably higher price for such refinement.)

The program is very complex for two basic reasons. First, the definitions of words to be replaced may themselves contain one or more words to be replaced. Therefore, as many as necessary iterations of replacement have to be carried out in the process. Second, the huge data base representing the whole dictionary had to be subdivided into k files only one of which can be dealt with by the program at a time. The intermediate results of one run have to be transferred to the subsequent run, which requires some tricky programming. A brief description of the multi-file handling is given in the Appendix.

Figure 2, summarizes the results for four different levels of the covered set. Although the procedure followed (leaving one and then two files out of the nine, and adjusting for the bias introduced) leads to quantitative inaccuracies, the conjectures listed in the Introduction are fully corroborated.

- -

INSERT FIGURE 2 ABOUT HERE

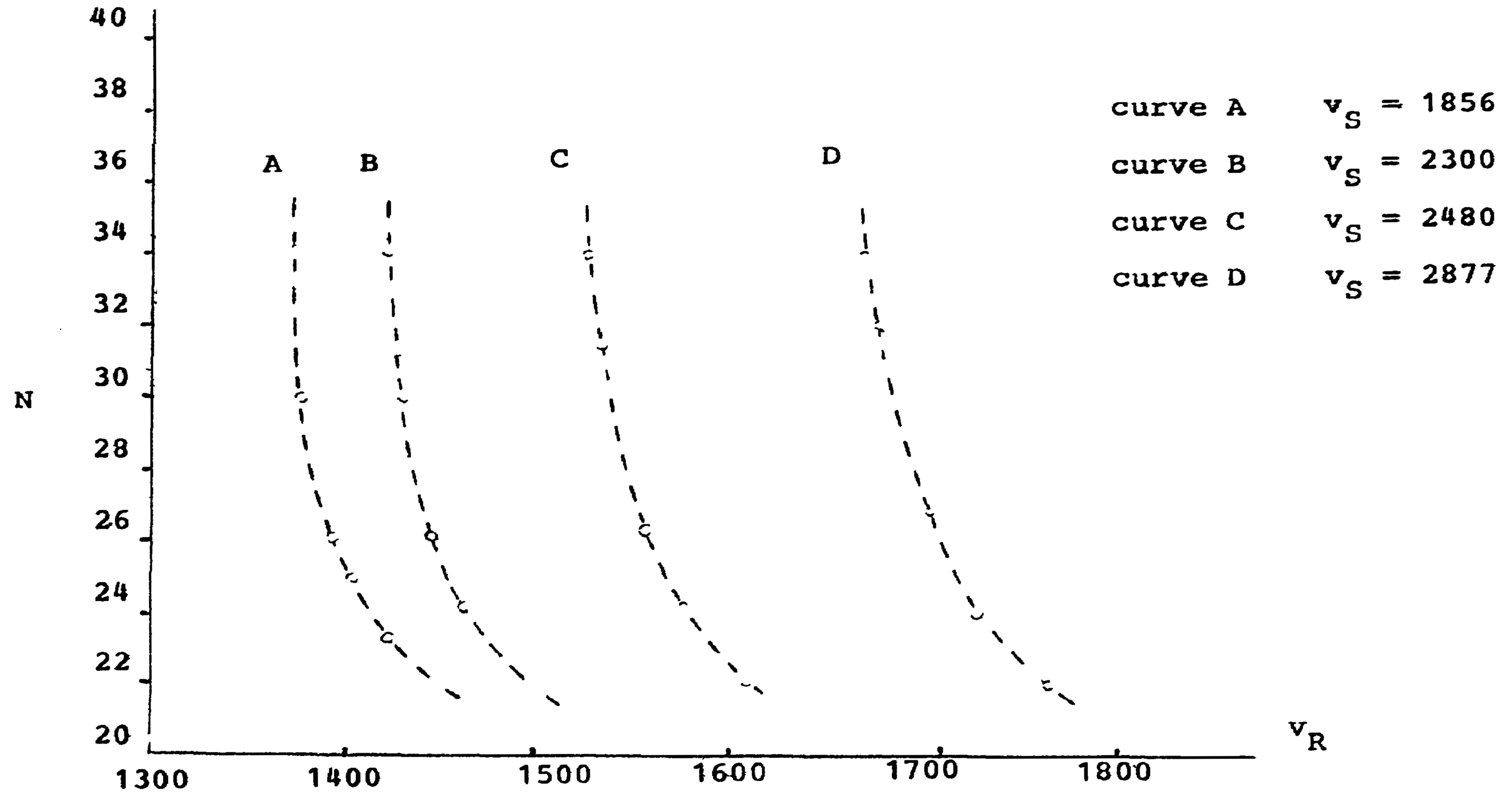
- -

FINAL COMMENTS

The data base encoded, some of the programs used and, most of all, the experience gained in dealing with dictionaries and their characteristic variables will be useful in attacking the next set of problems. The latter relate to the question on what size vocabulary is needed to cover a given number of dictionary entries (without the ubiquitous circular definitions). The answer should be given as a function of storage requirements and processing time so that an optimum solution can be obtained for a family of applications on a machine with a given cost structure. Such study will involve the semantic index of the elements of the covered set, the lexical valence of the elements of the covering set, and the frequency of occurrence of the elements of both sets.

ACKNOWLEDGEMENTS

We thank H. Viil, who co-authored with one of us (N.V.F.) the first phase of this work, for many ideas and stimulating discussions. We are also indebted to Penguin Books for their



Variation of Maximum Definition Length with the Size of Covering Set

FIGURE 2

permission to use one of their publications as our data base

REFERENCES

Chandor, A. (1970). A Dictionary of Computers. Penguin Books: Harmondsworth, England.

Findler, N.V. (1970). Some conjectures in computational linguistics. Linguistics, No. 64, pp. 5-9.

Findler, N.V., J.L. Pfaltz and H.J. Bernstein (1972). Four High-Level Extensions of FORTRAN IV: SLIP, AMPPL-II, TREETRAN and SYMBOLANG. Spartan Books: New York.

Findler, N.V. and H. Viil (1974). A few steps toward computer lexicometry. Am. J. Comp. Ling., 1, 1, 4.

APPENDIX

In the following, we give a brief description of the way multi-file handling has been organized.

It was noted before that the whole dictionary could not be fitted in the core memory at one time and, therefore, the data base had to be subdivided into 9 files to be processed separately. There was a need, however, for some flow of information between runs dealing with the different files. This was arranged by additional files constructed during processing time as well as a few control variable values being read from cards at the beginning of runs subsequent to the first one.

The variable KNTPPT indicates the section of the dictionary currently under study. The variable INCONT is set to 0 for the very first run for each N value. This tells the program to set up new lists for Covered List, Covering List, and so-called Waiting List. In all subsequent runs, its value is 1 which indicates that the program must bring these lists in from an additional, external file.

The program examines the current section of the dictionary, entry by entry. If the entry is an element of the basic vocabulary (type 2), the program bypasses it when it deals with the unreduced dictionary (it is bound to be processed as part of a definition later). Otherwise, this type of word is immediately added to both the Covered List and the Covering List (such word always covers itself), since the definitions in which they occur may have been eliminated.

If a word is not found on the Covered List, it is put there and the appropriate counter is incremented. Then all the words in the definition of the word in question are put on the Waiting List, which is subsequently processed. This is necessary because of the adopted principle that all the covering words must themselves be covered. (Tabulated data are meaningful only if this condition is satisfied.)

The program eventually examines the Waiting List word by word. If the current word is already on the Covered List (it may have occurred earlier in the dictionary), the program checks if it is also on the Covering List (it may not be because it has not yet occurred in the definition of another word). If not, it is put there and the appropriate counter is increased. All words on the Waiting List come from definitions and must therefore be added to the Covering List. After a word has been processed, it is deleted from the Waiting List (but its processing may have caused new entries to appear on the Waiting List).

If the current word is not on the Covered List, it must, of course, be put there. First, however, the program tests if the word occurs in the section of the dictionary currently in core memory (its "numerical value" is between those of the first and the last word of the section). If the word is not there, its processing is postponed and the next word on the Waiting List is examined because it is more economical to process first all the words available in the dictionary section present than to read in other sections of the dictionary as the words dictate it (memory swapping is expensive).

When the bottom of a non-empty Waiting List is reached, the words remaining there must be in other sections of the dictionary. Subsequent dictionary sections are brought in, to replace the current one, in a cyclic manner until all processing is completed.