required for buffers and the sort program if efficient running times are to be achieved. We do not yet know the optimum trade-off point, but it does seem that 44K is appropriate for the sort and that a 7200 byte blocksize is best for fast input/output. This entails the use of approximately 100K bytes.

Preliminary timing estimates show that the program is indeed fast. A text of 175,000 words can be completely processed in less than 25 minutes on a Model 75. A text of 20,000 words was processed in 4 minutes.

4. A Design for a General Statistical Analyzer for Natural Language Texts

by

John B. Smith

If we assume the desirability of making a number of statistical measures over a text of natural language, we are faced with the question of what such an analyzer should look like. In this discussion I shall examine some of the general problems inherent in a procedure of this kind and then suggest what some of the characteristics of a computer program to make these counts should be.

First, we must infer what sorts of counts or measures are likely to be wanted. Many of these will fall into several fundamental categories. These include determination of whether or not a unique string occurs in a text; a count of the number

of times such a string occurs in the text; and perhaps a linear distribution of its occurrences over the text, or certain subsections of the text. Next we may want to group several such strings into categories and determine the presence of the category in the text, its frequency, and/or its linear distribution. Also, we may desire individual counts and distributions of the elements of the category, but these statistics represent only repetitions of the first class of measures mentioned. Thirdly, we may wish to determine the existence of particular configurations of categories, the frequency of occurrence of these configurations, and, again, their distributions. This third class of measures may be "simple," applying only to sentences - for example, syntactic analysis; or it may be extensive and attempt to define the entire semantic structure of a text in terms of associations or linkings among a set of categories. No assumptions are made concerning the population over which these measures are taken; therefore, inherent is the possibility of comparing various texts, canons, even languages when such amalgamations are viewed as independent categories. Thus, it should be possible to place most, if not all, statistical measures into this taxonomy. The practical result is that a program that can make the kinds of measurements we have described should have enormous flexibility. The program described below represents one attempt at defining such generality into a workable system.

There are five basic sections of the Analyzer. A driver program allocates resources, determines analytic methods, and

"oversees" the operation of the entire complex. Input and output functions are handled by independent modules. Analysis requests are scanned by a module that establishes logical control vectors to be used by the driver program. Finally, there are a number of analytic modules that perform a variety of functions including making various counts, searching for associative tendencies among elements and categories, and mapping of elements onto thesaural-like semantic structures. Not all of these analytic methods will be available from the start, but because of their modularity they can be added as they are written and as research needs expand and change. For each analysis module, there will be a separate request analyzer subprogram to examine the parameters necessary to define a particular run request. Each of the five major divisions of the program will be described in detail below.

The master control program or driver program manages the individual subroutines and the allocation of resources. Upon initiation of the program, its first task is to pass control to the run request analyzer which either reads a card from the card reader or receives a line of input from a remote terminal. Upon completion of the run request analysis, this program is returned a control vector indicating the various text analytic modules to be used, individual control vectors for them, control information for input data sets to be used, and the particular configuration of output display facilities requested. From a consideration of the number and kind of individual statistical measures to be made, the driver allocates various arrays, structures, banks of counters, etc. as permissible from resources

available. If analysis is to be performed on only a portion of a text or data set, this information is set up in logical form for the input program. Data sets are then opened and control passed to the input routine. After this step, control is passed back and forth between the input routine and the analysis modules until the analysis is complete or the ends of the data sets are reached. At this time the driver again assumes command and establishes any control necessary for post-analysis processing such as sorting, accumulating frequencies from scratch data sets, etc. The Output program then takes control, performs the "housekeeping" functions described above, and prepares the final printed or graphic display of the results of the analysis as well as machine-readable data sets of these values. If the program is being run in the conversational mode, output is directed to the terminal and the driver waits for additional instructions; if not, the driver terminates operation.

The second phase, the run request analyzer, consists of several modules that are brought into operation according to the specific analytic modules called for. In explaining this process, I shall use for an example the request analyzer associated with the COUNT analysis program that compiles a number of counts or distributions over a text. This particular analyzer is called in when the keyword COUNT appears on a run request card. What follows is a series of requests for specific distributions defined in terms of four major kinds of parameters. (In effect, the set of these parameters and the ordering procedure that determines how they can be arranged amounts to a fourth level "user language".)

The first two parameters specify a particular alphabet or set of symbols to be used and the name of a dictionary in which text words exist in terms of the symbols of this alphabet. For example, the alphabet could be the set of phonemes for English and the dictionary, "TEXT.PHONEME," could contain the text expressed in these phonemic symbols. Logically, the dictionary entry could refer to a subprocedure or algorithm that converts the character representation of a word into the appropriate alphabet-symbol representation. More will be said below in the discussion of the Input procedure concerning the assumed data format of the dictionary.

The third level of parameters specifies the linear dimensions, if any, to be used in the analysis. These values are used to develop distributions such as words per sentence, nouns per 500 words of text, etc. Within this level are five specific parameters that are arranged as follows: $n(x_1,y_1)(x_2,y_2)XN$. Starting from the inside and working out, the (x_1,y_1) refers to the upper limit of an index hierarchy for which counts are developed. Assume, for example, that the scheme of INDEX including counts for volume, chapter, paragraph, sentence, and word-in-sentence had been used for hierarchy one; then (1,3) would refer to the paragraph level. If this were the only parameter specified, the program would develop all distributions of the hierarchical scheme below this level: namely, character/word, words/sentence, character/sentence, words/paragraph, sentences/paragraph, etc. If the ordered pair (x_2,y_2) had been specified, then only those counts between the two levels would have been made. For example, (1,3)(1,4) would

develop the distribution for sentences/paragraph only. The symbol X may be used if only a particular count is desired. ((1,4)X would result in a count of words/sentence.) Without the second hierarchical level specified, the basic unit is assumed to be words; however, if the second level is indicated in conjunction with X, that unit is taken as the "atom" for the distribution. Thus (1,2)(1,4)X would result in the distribution of sentences/chapter only.

The last symbol in the sequence, N, is used if counts are to be normalized. If one were studying the distribution of function words over sentences, he would most likely wish to have this distribution normalized between 0 and 1 so that he could determine the relative, not absolute, placement of this category of words. Thus the position in the sentence when such a word appears is divided by the length of the sentence, thereby giving the relative proportion of the way through the sentence -- or normalized distribution -- at the occurrence of these function words. Finally, the first symbol, n, specifies that a group of elements is to be used as the basis of the distribution. If nothing else follows n, the unit is assumed to be words. Thus, "500" would imply that counts are to be made for each 500-word unit of text. 5(1,3) would indicate that hierarchical counts are to be accumulated for each five paragraph blocks of text.

The fourth level of parameters is similar to the third but allows the user to partition his distribution into several separate sets of counts. For example, the user could obtain separate distributions of function words per sentence for

sentences of length 1-10 words, 11-20, 21-30, etc. This facility may reveal important conditional characteristics of certain stylistic features that vary according to particular textual dimensions. The general format of this level of parameters is $m(x_3,y_3)(x_4,y_4)$. As before, the $(x_3,y_3)(x_4,y_4)$ terms specify the units and appropriate level at which these units apply. M specifies the multiple size. Thus 50(1,3) would indicate that paragraph counts are to be broken into distributions based upon 50 word units. Separate distribution would be maintained for paragraphs of fifty words or less, paragraphs of 51-100 words, etc. Similarly, 10(1,3)(1,2) would indicate that separate distributions are to be kept on the basis of the number of sentences per paragraph: i.e., paragraphs of 1-10 sentences, 11-20 sentences, etc.

Obviously, not all of these four levels of parameters will necessarily be used all of the time; and, indeed, not all combinations are meaningful to the analyzer. For example, there can be no fourth level partitioning of counts unless level three is defined for a particular analysis. A rough guide to admissible combinations and the kinds of counts they develop is indicated by the table on the following page in which a "1" indicates that a parameter of some kind is present for that level and a "0" implies the absence of that level parameter. Similar to this kind of organization, additional request analyzers will be developed for other processing modules.

The input phase of the program is kept logically independent

to allow the user to adapt his data sets to the format required for analysis. The program is set up to take as input combinations of the output data sets from the INDEX program described elsewhere in this report. From those data sets, the INPUT subprocedure prepares data lists in fixed formats required by the analysis modules. For example, a distribution of the frequency of occurrence of phonemes over a text would require that the INPUT program first pass to COUNT a type list of phonemic symbols. Then, as the data is read, it would construct lists of text words defined in those same symbols. COUNT would in turn analyze these lists. Similarly, if hierarchical counts are being made, the input program would pass a text-ordered list of logical vectors (00011, etc.) that indicate what positions in the hierarchy are changing with a particular text word or punctuation mark. If the user's data is in a different format from that provided by the INDEX package, he can substitute his own INPUT module to accept his data set and produce the necessary lists of data required by the analysis modules.

The analysis phase of the program is kept modular to allow for growth of the system. Most immediate plans call for the development of the COUNT procedure, discussed briefly above; however, it is likely that after the completion of that procedure a VIA analyzer for semantic associations and a principal component analysis program for developing tendencies among words or groups of words to appear in close proximity in a text or some section of a text will be adapted to this modular design. Since COUNT is the analysis procedure that is most likely to be developed first, more discussion of its operation is in order.

As mentioned in the description of INPUT, COUNT takes as its input an ordered list of data. On that list, according to the control parameters passed to it by the driver program, COUNT accumulates several kinds of statistical measures. First, if an alphabet is specified, it establishes a bank of counters corresponding to that list of symbol-types. Then as the text is passed to it in terms of those symbols, it increments the appropriate counter. For example, if the text word were "æ t", the counter for æ would be incremented by one as would the counter for t. If some type of hierarchical count is being made, COUNT receives a text-ordered list of logical bits that indicate what level of the hierarchy is changing with the current text word. For example, if the text word was a "." that happened to end a paragraph and if the indexing hierarchy established was for volume, chapter, paragraph, sentence, and word in sentence, then the logical vector would be (00111) indicating that word, sentence, and paragraph counts all change with this symbol. COUNT would, in turn, increment the appropriate counters for each of these levels.

The third major function of this procedure is to maintain separate distributions for various textual parameters. This function is handled by outputting the data accumulated into a temporary data set corresponding to the appropriate partition. If, for example, the data is to be partitioned according to paragraph length in multiples of ten sentences, there would be a temporary data set for counts for paragraphs of length 1-10, one for 11-20, etc. It would then be the responsibility of the OUTPUT procedure to accumulate these individual counts and print out or otherwise display the

results.

OUTPUT has four major functions. Its primary responsibility is to display in an efficient and effective format the statistical counts developed by the analysis procedures. This may be in list form or it may be the plotted output of line graphs, histograms, etc. Secondly, it should store these data in a machine-readable form so that they may be used in subsequent analyses. Formats should be written onto the data sets themselves in a form that the INPUT procedure may analyze and use and thus relieve the researcher of many of the headaches of file manipulation and compatibility. Thirdly, in some analyses, temporary data sets were created. In these cases the OUTPUT program will sort these records, if necessary, accumulate counts, and then display and store the results. Finally, in analyses that require additional utilities -- such as principal component analysis -- the OUTPUT procedure will prepare the counts accumulated in a format that can be used by the utility, write this data along with card images of required control statements onto a temporary data set, and then pass control to the system for execution of the utility program.

Because of the modularity of this package, it should provide enormous flexibility for the user. With it, he will be able to accumulate the large number of parametric values necessary to test actively complex models of style and content so that these areas may move from speculation into validated implementations.