Behavior Research Methods & Instrumentation
1974, Vol. 6 (4), 430-438

COMPUTER TECHNOLOGY

WORDS: A computer system for the analysis of content*

HOWARD P. IKER and ROBERT H. KLEIN
Department of Psychiatry, University of Rochester, School of Medicine and Dentistry, Rochester, New York 14642

WORDS is a computer-oriented system for content analysis designed to elicit major content themes without recourse to a priori categorization systems. The system consists of a number of modular and independent programs that the user can configure in any fashion to process the data to be analyzed. This paper presents current information on WORDS, WORDS programs, WORDS systems logic, and on the availability of the system.

The initial implementation of the WORDS system was first described about 10 years ago (Iker & Harway, 1965). In the intervening years, a continuous research program has been conducted, under the auspices of NIMH and NSF, which has been designed to increase the efficiency, scope, validity, and generalizability of the system.

The theory of WORDS was first described by Harway and Iker (1964). The mnemonic–WORDS–refers to the collection of programs comprising the system and is based upon the pivotal logic of the method itself, viz, that sufficient information exists within the word and within the temporal associations among and between words to allow data-generated elicitation of major content themes and materials.

WORDS is based on a contiguity association logic. We take the word as our *unit of information*. An input document is divided into segments of time, or segments of equal length, or paragraphs, etc., and our *unit of observation* is defined on these segments. Within each segment, the frequency with which each word occurs is determined. Covariation in frequency of occurrence is then quantified between each and every word (across the n observational units) in an intercorrelation matrix. Operationally, these intercorrelations represent the degree of association among words as they are observed across successive units of the data. This matrix is then reduced by multivariate procedures (factor analysis, cluster analysis, etc.) to locate, in a systematic fashion, the presence of common word groups. Our research has demonstrated that these word groups both represent and identify the major content themes in the raw data. Elaboration and analysis of these thematic data can then be conducted by using such methods as cluster or factor scoring to locate and inspect highly saturated sections of the raw data, by configurational analyses to examine changes in content profile patternings in different portions of the data, and so on.

WORDS does not require a priori category systems or dictionaries. This concept is at once both the central motivation behind the approach and the major characteristic that tends to distinguish this system from other content analytic methods–whether computer based or not. In brief, the relationships among and between the data themselves generate their own thematic "centroids." WORDS is designed expressly to avoid dependence on any methodology which predetermines these thematic findings.

Our research, through 1967, is summarized by Iker and Harway (1968). Since its inception, WORDS has been used for: analysis of psychotherapeutic interview materials (Harway & Iker, 1965, 1969; Harway, Iker, & Leibowitz, 1969), studies in verbal productivity and changes in productivity in manic patients (Harway, Warren, Leibowitz, Tinling, & Iker, 1973), studies in the area of the humanities and linguistics (Jandt, 1972; Jonas, 1971), and an analysis of the entire Memoirs of Daniel Paul Schreber–a landmark case in the historical development of psychoanalytic theory–(Klein & Iker, 1974). A generalized historical perspective on the use of word-word intercorrelations is presented by Iker (1974) as well as an operations manual and a series of studies reporting on methodological progress in the development of the system.[1]

WORDS SYSTEMS LOGIC

The Concept of a System

WORDS is a system. By this, we mean that it is composed of a collection of compatible and interrelated–yet independent–programs capable of passing and receiving data, one from the other, which will manage the data flow for the user from the beginning until the end of his computer use.

If a computer-oriented content-analytic method is not encompassed within a systems approach, the user will inevitably find that he must leave the particular program(s) with which he began in order to make use of other programs and other methods for further analyses and refinement of his data. When such a departure is

*Supported in part by NSF Grant GS-32241.

forced upon the user, he must, typically, begin to cope with the "library program." Many people who might wish to make use of computer-aided content-analytic methods have little sophistication in computer matters; further, the statistical methodology required in the analyses may be complex enough to raise problems in decisions as to which technique, what parameters, etc., are appropriate and desirable. Often, library programs cannot be easily interfaced with existing or generated output from other programs; data must then be reformatted, and this may, on occasion, be complex enough to force the user into ad hoc programming for alterations in input format. Appropriate library programs may be difficult to access: the user desiring analysis of very large matrices may have to scale down his request since no available program will handle his requirements; the user content with approximation approaches must often make do with high-precision but slow-speed exact routines.

Few people interested in computer-aided content analysis are going to use methods which entail problems of this nature. While no systems logic will obviate all of these problems, such an approach can come very close to reducing their impact on the casual user to the point where he can comfortably and economically make use of the method. WORDS has been designed with such a systems concept in mind.

DATA PREPARATION

The user must first select the observational unit. Whenever the material for analysis has paragraphing provided by the speaker or author, our research confirms the logical expectation that the paragraph is the appropriate observational unit. If paragraphing is not available, the user must make a choice predicated in part by the logic of the research and in part by the nature of the data. We have found that 1-min elapsed time units make for a workable and efficient segment size with psychotherapy data; where the data do not have an accompanying time function, equal numbers of words or equal numbers of sentences may be used. While there is no hard and fast rule as to what constitutes the "best" segment size, our research has demonstrated that multiple analyses of the same raw data have tended to converge on the same thematic centroids regardless of widely differing sampling rates.

Once a segment size is determined and a unique number assigned to each of these segments, data are then punched onto cards in their original form with the following changes: (1) contractions are expanded into separate words, (2) proper nouns are identified by preceding them with a dollar sign, and (3) quotation marks are either deleted or rewritten as two successive apostrophes. Words are then keypunched as though being typed. As many or as few words as feasible may be punched anywhere on the card, with the constraint that all data on any card must originate from the same observational unit (segment). A segment, on the other hand, may be continued across as many cards as required. When punched in this fashion, WORDS will accept the data for input processing.

THE STANDARD WORDS RECORD

WORDS operations hinge on the ability of the system to manipulate words statistically. Operationally, this demands that WORDS be able to use the various words independently. The system, therefore, must be able to handle each word as an independent unit, but must, at the same time, carry with each word sufficient information to allow accurate location of the point of origin of the word, its part of speech, the emitting speaker, and so on. These simultaneous needs generate the makeup and format of the standard WORDS record.

The standard record is internally structured as a set of fields whose format and structural positioning is set when data is first read into the system. While any of the data in any of the fields can be changed by other programs, the format, itself, is completely fixed.

With only a few exceptions, all data input to and output from the system's nonstatistical programs are WORDS standard format. The exceptions are those programs requiring special lists or files which are not properly part of the data being analyzed; such nonstandard data are automatically generated by and read in by those programs requiring them, and the idiosyncratic format is thus transparent to the user.

On input to WORDS, each word and each punctuation mark (including parentheses, quotation marks, etc.) is separated into an independent standard record. This output set is the raw data with which the system begins analysis. The standard record is composed of 10 discrete fields:

WORD. Contains the word that is the basic system datum. The field accommodates up to 16 characters.

INTV. A three-digit field indicating the interview in which the word was found.[2]

SEGM. A three-digit number which identifies the segment in the interview.

SEQ. A five-digit number indicating the sequence of the word in the segment.

GTAG. A one-character field set (by the parsing programs) to designate the word's major part of speech.

FREQ. A five-digit field set to 00001 at input to indicate the word's frequency of occurrence.

SPKR. A single-character field designating the speaker who emitted the word.

AUX1. A three-digit auxilliary field open to various uses depending upon the programs called.

AUX2. A one-character field, initially blank, reset by the parsing programs for ancillary GTAG information.

AUX3. A one-character field open to use by various programs. It is analogous to AUX1.

WORDS PROGRAMS

The logic of the computer implementation of WORDS is based on the availability of a series of generally independent and modular programs, each designed to do a specific job. The high degree of flexibility of the system is based upon the user's ability to configure a series of program calls that, in their successive operations upon the data, will produce the desired kinds of outputs.

Each of the system programs has a mnemonic by which it is called. Each of the programs, alphabetized by mnemonic, is described briefly below; there are a number of programs in WORDS which can be called only by the system itself, and they are not detailed here. Timing data estimates, in seconds, are indicated for each program.[3]

CLOSE. Called by the user to indicate the end of a WORDS run. It retrieves and outputs (for printing) any messages left by other programs in the run.[4] Called automatically by any program in the event of an abort. [1]

CLUST. A multivariate reduction program designed to accept an intercorrelation matrix for cluster analysis. The algorithm is recursive in that it will group clusters into higher order clusters as a function of a parameter selected by the user. The algorithm, its development, and results obtaining from its use are available (cf. fn. 1). The program is much faster than either of the factoring programs in the system. [.011 V^2 log(I + 1)]

COPY. Basically, a card-to-disk transfer routine. In addition to the transfer of card images, it also allows reformatting of cards into standard WORDS records as well as into "striplist" (cf. STRIP) format. [.002C]

CORR1. The system intercorrelation matrix program. Designed for high speed, it carries out all operations in core. As a result of this constraint, it handles a maximum of 215 variables. [.000015NV(V – 1)]

DECOD. Used to produce output listings from CORR1 results. Using each variable, in turn, as the "independent" variable, the program lists an ordered (absolute descending) set of the correlations between that variable and all others in the run. The number of correlations reported for each variable is determined by the user either as a fixed number or as the number of correlations exceeding some specified screening value. All correlations are identified by the word they represent. [.0015V^2 + 10]

DELETE. Deletes any data set from the WORDS private disk volume.[5] [1]

EDIT. The major record modification program of the system. It may be used to delete, change portions of locate, insert, or pass records. [20]

FCTR1. A principal components factoring algorithm for data matrices where the number of observations equals or exceeds the number of variables (words). It will accept matrices of order 215 or less and will extract up to 99 factors. [.0023V^2 FI]

FCTR2. A principal components factoring algorithm for data matrices where the number of observations is less than the number of variables (words). It will accept matrices of order 215 or less and will extract up to 99 factors. [.0023N^2 FI]

FIXST. Used to make changes in the "striplist" that can then be used by the STRIP program. [.004C]

JOIN. Allows the user to concatenate from two to four data sets into one. The combined output may then be treated as an independent file. [2]

LSTR1. The main interface between the word-processing and statistical processing programs in the system. It prepares observational data matrices, Z-score matrices, means, standard deviations, and so on. The program also allows the user to adjust the sampling rate by combining adjacent input segments. [.001W]

OMITS. A program which utilizes a sorting rearrangement of an input file for the purpose of deleting all records in a string that are equal to the first record of that string on the specified sorting parameters. Using OMITS with a sorting parameter of "word" would produce an output file containing one record for each input type. [.0011W]

OMITX. Identical to OMITS except that the file is assumed already in order on the specified sort parameters. [.0004W]

PARS1. The first part of the procedure designed to affix parts of speech to the words in the data base. The program couples dictionary searches along with extensive logical multiplications to assign a unique part of speech to the word or to assign the three most probable parts of speech for those words which cannot be uniquely coded without context information. [.003W]

PARS2. The second part of the parsing procedure. This program uses the sentence context in an effort to assign unique part-of-speech coding for those words not uniquely coded by PARS1. [.006W]

PICK. This program allows the user to select n records from the beginning of the specified data set. The size of n is specified by the user. [1]

PRINT. The major print program of the system. Allows the user to print any set of fields from the standard record and to print those fields in any order. It allows extensive titling and identification information to be placed on each page as well as giving the user variable spacing which, itself, can be determined on the basis of specified field contents. [.002W]

REFER. Intended for the inspection of the context of individual words in which the user is interested. It is used primarily for referencing. The program contains a list of all pronouns so that when invoked without options, all occurrences of such words may be seen in their context. If not explicitly disabled, the program also punches a card, for each occurrence of the found words, containing sufficient information for EDIT to use that card for change purposes. The user may add to the list of words contained by the program or override the list completely. [.005W]

RENAME. Allows renaming of any data set on the WORDS private volume. [1]

RERYT. Produces a readable copy of any standard record file that has been presorted into original[6] order. Data other than the word are stripped off and the words are formatted into print lines allowing an easily read output of the file contents in the form in which they were originally found. [.001W]

RESEG. The program allows resegmentation of data already within the system. Resegmenting may be established on the word itself (any user-defined set of n successive words constituting a segment), the beginning of each new sentence, or the occurrence of any user-defined marker in, or added to, the data. [.00075W]

SCORE. The factor-scoring program for the system. All factors are scored for all observations. One output orders observations on factors while a second orders factors on observations. The user can thus determine those locations where a given factor or factors are gaining their highest scores and can also locate the characteristic factor profile for given observations. [.0002VFN]

SELECT. Used to select words for intercorrelation and then multivariate reduction. The program chooses a set of words which maximize the associational richness of the data. The algorithm and results are available (cf. fn. 1.[.00003NV^2]

SORTS. A program to produce a rearranged file with no alteration in the contents of that file. [.0008W]

SPLIT. Used for initial data entry into WORDS. Words, punched successively along a card, are split off and formatted into separate machine records with sufficient identifying information supplied by the program to identify each record uniquely so that original input can always be reproduced unless records have been changed or removed from the file. [.0013W]

STRIP. Used to deinflect (lemmatize) words to root form. An extensive dictionary, the "striplist," is accumulated by the user to indicate how and where such reinflections are to be made. [.0006W]

SUMMS. A program to rearrange and then summarize records that are equal on the sorting parameters furnished. Using SUMMS with a sorting parameter of "word" would produce an output file containing one record for each input type; the frequency field of each record would specify the number of occurrences of the type. [.0011W]

SUMMX. Identical to SUMMS except that no rearrangement takes place. [.0004W]

TPOSE. Used to prepare data for FCTR2. It transposes the Z-matrix from LSTR1, and computes (the equivalent of) a covariance matrix from that data. [.0035NV]

VDCOD. Analogous to DECOD except that VDCOD operates with input from VRMX1. It converts the rotated output into easily read listings, orders loadings absolute descending, replaces variable numbers with the word itself, and so on. [.00015FV^2]

VRMX1. The varimax factor rotation program. Uses the Kaiser algorithm for rotation to simple structure of up to 215 x 99 input matrices. [.04VF]

THE JOBS PROGRAM

Purpose

WORDS achieves its flexibility by allowing configuration of a run with programs called in any order, with any degree of repetitiveness, and in any number, to accomplish the desired goal. This high degree of program modularity is, at once, both an asset and a liability. While it affords almost unlimited flexibility, it also generates complex problems when the user attempts to set up actual runs on the computer; not the least of these are input/output (I/O) manipulations, since any program may be asked to "hand-off" its data to any other program.

For the target computer to handle this kind of I/O maintenance, it is necessary that the user communicate with the control program of the machine. In all modern computers, the handling of I/O, the scheduling of jobs, the execution of separate jobs concurrently, etc., is under supervision of an executive system. For the IBM S/360 series of machines, this general executive is known as the operating system (OS). A special language is required for the user to inform OS of his requirements, of the locations and dispositions of the files generated during the run, of the handoff between one program call and another, etc. For the IBM S/360, this control language is JCL (Job Control Language).

Use of WORDS, as just a series of programs, would require not only a thorough understanding of WORDS itself, but also of JCL. IBM's JCL is a complex language, requiring a highly detailed form of specification and keypunching—with an attendant high probability of keypunch errors—and is totally intolerant of error. Since a full WORDS run will frequently produce more than 100 JCL cards, a program, JOBS, has been written to allow the user to communicate with OS via a series of relatively easily prepared control cards rather than by having to use the complex JCL functions.

Method

An overall view of a run on WORDS which makes use of JOBS involves two separate runs on the computer insofar as OS is concerned. The user submits a deck of control cards to JOBS; JOBS then utilizes that deck to emit the complex series of JCL statements actually required to communicate with OS. The second run—the real run on WORDS—then takes place as JOBS arranges for the emitted JCL deck to be read into the system as though it had been prepared directly by the user.

While JOBS, itself, requires some JCL for its operation, the amount needed is minimal and is constant from run to run. Thus, the user needs but once to prepare the JCL deck for JOBS and may, with few

changes between runs, utilize that deck repetitively.

Options and Parameters

For a system such as WORDS to be viable, it must be able to run at different installations without requiring extensive programming changes. As particular run requirements change as a function, say, of the size of the input data sets, the user needs to have the flexibility of making changes in his demands and requests on the system even within his own installation. To meet these needs, JOBS contains a large number of changeable options and parameters. The majority of these conditional specifications can be defaulted. The number of options and parameters available is over 30 and too large to enumerate here. Some examples, to give the flavor of the kinds of control available, are: changes in the name of the WORDS private volume, stipulation of whether disk storage is to be recorded in cylinder or track mode, ability to specify maximum amount of core available to the various programs, etc.

The sense of the option and parameter set is such that JOBS is able to tailor both the general configuration of the run, as well as some of the individual programs in the system, to the implementation restrictions operating at the time. The impact of such flexibility can be considerable. For example, the factor analytic programs handle large matrices and are programmed to accommodate matrix storage either on an in-core or an I/O basis–with in-core execution being significantly faster. When JOBS receives a request for, say, the FCTR1 program, it computes the amount of core required to handle the program request (in terms of the program itself, the number of factors requested, the number of variables submitted, etc.), and then compares this figure with the minimum available core size indicated (or defaulted) by the user. If available core is below that required, JOBS recomputes required core based on an I/O logic; if available core is still too small, JOBS schedules an abort for the actual WORDS run and emits a message indicating the amount of core required to process. If the I/O method will allow execution, JOBS sets up the calling sequence for the program with parameter specifications, indicating that that method is to be followed. In so doing, JOBS also generates all of the additional I/O JCL information required to handle the extra files needed by the method. This kind of interaction between user-specified overrides and/or defaults, the program calling configuration, etc., allows a high degree of "tailoring" between the WORDS run and the constraints of the computer installation at that point in time.

I/O Scheduling

One of the more demanding problems in JCL involves the punching and determination of format, specifications, record length, etc., involved in cards which define the files of data created and handed off during the run. JOBS totally frees the user from any requirements for such card preparation. By using a letter designation for each of the program calls submitted to JOBS, the user may request that output from any particular program call be handed off as input to any other subsequent program call. JOBS assigns internal descriptors to identify the file, determines the necessary data control block information for the file, computes and maintains necessary space requests for the file, and will, unless the user indicates otherwise, delete the file from the system as soon as it is determined that it will no longer be used again in the run. The user, of course, always has the option to nominate any file for permanent maintenance. On request, JOBS will place any file into "kept" status so that it goes to the private WORDS volume and is thus available at later times on other runs.

Procedure Calls

While WORDS programs are technically independent of each other, calls on the system generate sequences which are far from random. Almost inevitably, some programs are either usually preceded or followed by certain other programs. The presence of such fixed-configurations, in the context of jobs, allows the incorporation of "procedure" calls. As used here, a procedure call means the invocation of a series of programs by a single call on JOBS. JOBS currently has incorporated five such procedures.

The CORR1 PROC invokes programs to prepare WORDS standard record data for intercorrelation, correlate the data, and then submit the results to the DECOD program. The ENTRY PROC calls out a series, beginning with SPLIT, to RERYT the data, obtain a list of different words and their frequencies, and to obtain a trial STRIP on the data (to determine if additional entries are required for the striplist). The PARSE PROC allows pre- and postparsing sorts on the data, as well as calling the two parse programs and their required intervening sort. The STAT1 PROC invokes a seven-program sequence of LSTR1, CORR1, DECOD, FCTR1, VRMX1, VDCOD, and SCORE, while the STAT2 PROC calls out the analogous program series for the FCTR2 program. Use of these procedures not only saves time for the user when preparing calling sequences for JOBS, but guards against errors in defining correct input files to the various programs. The SCORE program, for example, requires four inputs. Determination of inputs, the program producing the inputs, questions of retention, etc., are all handled automatically by use of a procedure call.

Error Detection

JOBS also acts to produce diagnostic messages whenever errors are determined. While JOBS, itself, is never terminated if it is possible to continue, any error will elicit a diagnostic message on the one hand and a signal to abort the second run as soon as JOBS has

completely scanned all of the incoming data. Unless the error detected by JOBS is disastrous, the program continues to make as many diagnostic checks as is possible on all remaining input.

Whenever an error does occur, JOBS will emit at least one, and on certain occasions several, diagnostic messages. Where feasible, messages are substantive and point out the nature of the error; if a substantive message would be too long, a coded message is substituted which is elaborated fully in the WORDS manual. JOBS currently contains well over 100 diagnostic checks which are imposed on each of the cards input to the system.

Jobs and Systems Logic

JOBS has a number of options, tasks, and uses other than those mentioned. They are too extensive to be detailed here and are fully covered in the system manual (cf. fn. 1). We summarize our presentation of JOBS by noting that it, in a very real sense, operationalizes much of the systems logic of WORDS. Clearly, there is nothing about WORDS which requires JOBS for its execution. Were users willing and able to take care of the necessary JCL, WORDS would run just as well one way as the other. JOBS is designed as the most efficient method available for freeing the user from the complexities of JCL and is designed, where possible, to protect the user against his own errors by aborting a run which might cost considerable money were it to go into execution with errors revealed only later. JOBS represents our best current attempt to make the use of WORDS a relatively straightforward task requiring a minimum of computer sophistication. It is further developed to minimize the problems of implementing WORDS at installations other than that at which the system was developed. Our experience has been that JOBS can substantially reduce the effect of installation differences through use of its many options and parameters.

DATA FLOW IN WORDS

Computer Sorting

Before presenting an example of a WORDS run, it is first necessary to understand the logic of computer sorting. In a sort operation, WORDS is able to use any of the 10 fields of the standard record, in any order, in any number, as a sorting string to determine output order. The sort program treats each record as an independent entity. The user specifies the field(s) containing the data on which ordering is to be established. A sort on the "word" field alone, for example, would produce a new file alphabetized on the word portion of each record. A sort on "intv" would produce a new file with all records from interview 001 preceding those from interview 002, and so on. When multiple parameters are specified, ordering is established on the leftmost paratmeter with subordering on each parameter that follows. Thus, a sort on "word" by "intv" would establish an alphabetically ordered file (just as with the sort by "word" alone). Within identical words, however, subordering would take place on the "intv" field–all cases of, say, "the" in interview 001 would precede all "the" in 002, and so on. With a sort of "intv" by "word," all data from interview 001 precedes that from 002 (as was output from a sort on "intv" alone); now, however, all words within each interview will be alphabetized. Extensions beyond two parameters utilize exactly the same principle.

The ability to comprise a sort on any subset of fields within the standard record and the ability to submit these ordered data to summarizing programs makes WORDS able to accomplish many tasks additional to the basic purpose of the system–data-generated content analyses. It is feasible, for example, to ask the system to obtain a list of all word types in the data and the frequency associated with each, to ask how many types occur within each separate interview, to ask how many types occur for each speaker, in the first vs the last half of the data, to determine how many tokens occur in each interview, for each speaker, to list all adjectives in the data, obtain a list of types in order of frequency (low to high or high to low), etc. There is no need, because there is very nearly no end, to continue these types of specifications. All one needs to do is to envision the fact that the 10 fields may be submitted as sorting strings in any order, in any number, with or without summarizing and/or deletion operations, to see the total flexibility of the system.

A Configuration Example

The modularity and independence of the several programs in WORDS is such that an innumerable number of perfectly legal configurations are possible. There is, then, no single configuration which would completely describe the system. Rather, we present here a single, large configuration that will submit raw input data to procedures necessary for cluster-analysis in a totally automatic fashion.[7] Table 1 presents the 17-step calling configuration. An approximate time, in minutes, is noted for each call. Calls A and B are procedures involving five program calls each.[8]

The A step makes use of the ENTRY PROC for initial system input. The titling information, "EXAMPLE," will appear on all printed output, and JOBS is instructed to place the alphabetized file into "kept" status on the WORDS private volume under the title "EXAMPLE." Output from SORTS is automatically input to the PARSE PROC in the B step. PARS1 receives the alphabetized data and subjects it to initial part of speech affixing routines. The procedure takes this output to another SORT, which orders the data by sentences and then channels it into the PARS2 program. After PARS2 has finished assigning parts of speech, the "POST.SORTS" keyword on the B step invokes SORTS to alphabetize output. STRIP receives the sorted output, deinflects the data, and hands off to SORTS in Step D, which once more alphabetizes. Output from SORTS

Table 1
WORDS Configuration Example

**A.	ENTRY PROC TITLE=/*EXAMPLE*/ SORTS=/EXAMPLE/	(4.58')
B.	PARSE PROC (I1=/EXAMPLE/) POST.SORTS	(8.83')
C.	STRIP	(0.50')
D.	SORTS /WORD/	(0.67')
E.	EDIT (I1=D,I2=/UHHEDIT/) KEY	(0.25')
F.	SORTS /WORD/	(0.33')
G.	SUMMX /WORD/	(0.17')
H.	SORTS /–FREQ/WORD/	(0.07')
I.	PICK V=500 PASS	(0.02')
J.	SORTS /EDIT/	(0.03')
K.	EDIT (I1=F,I2=J) PASS KEY	(0.25')
L.	OMITX /WORD/	(0.17')
M.	SUMMS (I1=K) /INTV/SEGM/WORD/	(0.47')
N.	LSTR1 (I1=L,I2=N) V=500	(0.33')
***P.	SELECT (I1=N.O2,I2=L,I3=N.O3) V=500,MAX=215	(12.50')
Q.	CLUST (I1=P,I2=P.O4,I3=P.O2) V=215	(6.00')
R.	CLOSE	(0.02')
		(35.19')

***Input data would immediately follow step Δ.*
****An "O" step is omitted in order to avoid confusion between the step designator and the output file designators.*

goes to EDIT in Step E by nominating the input file as Step D output ("I1=D"). EDIT takes its "editor" data file from a systems file, "UHHEDIT," which makes such changes as deletion of all function words, changes of "not," "non," to "no," etc. Output from EDIT is alphabetized in Step F, and this sorted output is summarized, on the "word" field, in Step G to produce a file containing one record for every type in the data to that point; the "FREQ" field contains the frequency with which that type has occurred. The file of types is rearranged into descending order on the frequency field in Step H (with words alphabetized within equal frequencies) and then handed off to the PICK program in Step I, which creates a new file containing the 500 highest frequency words; each word becomes a screening record for the EDIT program soon to follow. The 500 editor records from PICK are sorted into special order for the EDIT program (by use of the sorting mnemonic "/EDIT/") and then handed off to EDIT in Step K. Note that EDIT, in this case (as compared with Step E), uses the alphabetized total data file produced by the SORT in Step F, but takes editor records from the data produced by PICK and its subsequent SORT. EDIT produces a new file containing all instances of the 500 highest frequency words. The output from EDIT is moved first into Step L to produce a list of alphabetized types and also into Step M to sort and summarize the data into an appropriate order for the statistical interface program, LSTR1. Using both the type list and the segmentally summed occurrence data, LSTR1 arranges and formats the variables for input into the statistical routines. SELECT is then called in Step P to choose a set of 215 words from the 500 available that best maximize the associational networks in the data. SELECT uses the file of types from Step L, and the second and third outputs from LSTR1 (a Z-matrix and the set of means and standard deviations). SELECT chooses the 215 words which best maximize the associational networks in the data and outputs a 215 by 215 correlation, as well as the appropriate Z-matrix, set of means and standard deviations and word list associated with the reduced word set. The correlation matrix, word list, and Z-matrix are handed off to CLUST, which employs a recursive clustering algorithm to produce a set of hierarchical clusters defining the thematic content areas in the submitted data. Step R indicates that JOBS is to end the run at that point.

RESEARCH WITH WORDS

The asset of WORDS is that the themes and content areas it produces are much richer, more detailed, and, by definition, more in tune with the original data than those which would be produced by the sole use of a priori category systems. At the same time, one of the liabilities of WORDS is that these results are inevitably defined in the words of the speaker. This is a reasonable characteristic so long as WORDS is used to describe solely the data base under analysis. When, however, one tries to generalize from one set of data to another, to compare one data base with another, even to compare two separate analyses on different parts of the same data, substantial problems can arise, because the results are difficult to put onto a common measuring dimension or set of dimensions.

Category systems, on the other hand, always guarantee comparability between any two data bases; their asset is equivalent to WORDS' liability. By the same token, however, their liabilities cluster around errors of comission (combining words evaluated differently by the speaker), omission (not combining words evaluated similarly by the speaker), great generality, and lack of detail–all problems which are solved adequately by use of WORDS.

We plan to investigate the possibility of using category

systems–precisely for their generality–*after* that is, independently of, standard WORDS analysis. In such analyses, the category data would be employed only after WORDS-defined materials have been extracted. There are several methods by which such analyses could be undertaken. One, for example, would be to score the raw data on the WORDS-derived themes, rescore the data on the basis of the category measures, and then correlate the WORDS data with the a priori category data. One could then speak to the relationship bewteen each category and each theme. Clearly, the category system could in no way impinge upon or affect the WORDS extraction procedures.

The availability of such ancillary data may be of great utility to WORDS. Recourse to the kinds of dimensions provided by such systems would make it possible to compare different data, different speakers, different points in time, etc. They would allow, for example, analyses of the first, as compared to the last, sessions of psychotherapy and would allow inferences, say, that much the same kind of affective structure is present at both points in time but in very different thematic areas. They would allow one, for example, to stipulate that ANXIETY for one patient is represented by certain types of thematic material, while ANXIETY for another patient manifests itself in very different areas. Within a single analysis, one could speak to the extent that ANXIETY is present in each of the thematic areas being separately extracted by WORDS. Research into such methodology is now under way.

THE AVAILABILITY OF WORDS

WORDS is functional and available. The potential user, however, must be aware that use of the system will require a considerable investment of time. Despite the fact that the system has been written in a fashion designed to make its use as simple as possible, WORDS still remains a highly complex collection of programs. Without going into detail, this fact may be demonstrated by noting that the WORDS manual is currently over 200 pages long. This is not to say that WORDS is intolerably difficult to use; rather, we wish to stress that one should not expect to make use of the system as though it could be acquired in one day, run up on the target computer on the next, and put into operation on the third. Understanding the interrelationship among and between programs, learning to use the JOBS program, making changes to satisfy installation differences by appropriate option and parameter changes, all require substantial amounts of time. While the project itself sets no constraints on potential users, we have always advised interested researchers to consider carefully whether they wish to invest the time needed to learn to use WORDS adequately.

Installation constraints, of course, represent a minimum prerequisite for implementation. WORDS is written completely in PL/1. Since PL/1 is currently implemented fully only for the IBM S/360-370 systems, the requirements for making use of the system are, then, as follows: an IBM S/360 Model 50 (or higher) computer with at least 256K bytes of available core, which operates under full OS, has a PL/1 compiler of Level 5 or greater, and has available a 2314 disk device or interchangeable counterpart. If these requirements can be satisfied, WORDS can be implemented at the target installation. The project has several memoranda which detail specific requirements; they are available on request.

REFERENCES

Harway, N. I., & Iker, H. P. Computer analysis of content in psychotherapy. Psychological Reports, 1964, 14, 720-722. [Also in G. E. Stollack, B. G. Guerney, and M. A. Rothbert (Eds.), *Psychotherapy research: Selected readings.* New York: Rand-McNally, 1966.]

Harway, N. I., & Iker, H. P. Objective content analysis of psychotherapy by computer. In K. Enslein (Ed.), *Data acquisition and processing in biology and medicine.* Vol. 4. New York: Pergamon Press, 1965.

Harway, N. I., & Iker, H. P. Content analysis and psychotherapy. Psychotherapy: Therapy, Research & Practice, 1969, 6, 97-104.

Harway, N. I., Iker, H. P., & Leibowitz, G. Association structures in natural language. Proceedings of the XIX International Congress of Psychology, London, 1969.

Harway, N. I., Warren, S., Leibowitz, G., Tinling, D., & Iker, H. P. Some aspects of language style of manic patients. Paper presented at American Psychological Association, Montreal, August 1973.

Iker, H. P. An historical note on the use of word frequency contiguities in content analysis. Computers & the Humanities, 1974, in press.

Iker, H. P., & Harway, N. I. A computer approach towards the analysis of content. Behavioral Science, 1965, 10, 173-183.

Iker, H. P., & Harway, N. I. A computer systems approach to the recognition and analysis of content. Computer Studies in the Humanities and Verbal Behavior, 1968, 1, 134-154. [Also in G. Gerbner et al (Eds.), *Analysis of communication content.* New York: Wiley, 1969.]

Jandt, F. E. Sources for computer utilization in interpersonal communication instruction and research. Today's Speech, 1972, 20, 25-31.

Jonas, T. J. The WORDS system: A computer assisted content analysis of Chaim Perelman's "New Rhetoric." Unpublished doctoral dissertation, Bowling Green State University, 1971.

Klein, R. H., & Iker, H. P. The lack of differentiation between male and female in Schreber's autobiography. Journal of Abnormal Psychology, 1974, in press.

NOTES

1. Studies reporting development of the SELECT program and the results of using a recursive clustering algorithm for data reduction are available as a series of manuscripts.

2. WORDS was initially designed for the processing of psychotherapeutic interviews. The "intv" field may, of course, be used to identify whatever the user desires. Basically, it stipulates the major breakdown area so that its use, for example, with books is typically set for chapter identification.

3. Following the substantive information for each program, a timing estimate is furnished. If a program runs with reasonable independence of the amount of data, e.g., CLOSE, this estimate is stipulated as a numeric quantity. If a program's timing is extremely dependent upon a multiplicity of factors, e.g., EDIT (number of words input, number of editor cards, type of each of the editor cards, etc.), an average figure with a substantial safety margin is presented. In other cases, a formula is listed with the following abbreviations representing the number of: C = cards, F = factors, I = iterations, N = observations (segments), V = variables (total number of different words), and W = total number of words (tokens). All timing is computed in seconds. These data are based upon our experience with an IBM 360/65 computer with 512K high-speed core and 2314 disk packs for random access storage.

4. All WORDS programs produce messages informing the user of how many variables were processed, how many segments input, and so on. These messages are written to a systems

communication block, where they are retrieved at the termination of the run by CLOSE.

5. The entire WORDS system resides on a private disk pack which must be dedicated to WORDS use. This pack is also used to hold all data sets produced by WORDS which are to be retained across runs.

6. "Original" order is operationally represented by a sorting operation on the string "/intv/segm/seq/." If data have not been altered by other programs, a sort on this string always returns the file to an order which, if printed sequentially, would produce the words in the same order as input.

7. This configuration is presented to display the relative simplicity of preparing control information for JOBS and to illustrate the power of program modularity in a configurational context. In fact, however, we would seldom employ such a run. The example assumes that input is totally without error, already referenced (pronoun replacement), and contains no words new to the system so that deinflection will take place accurately. We should ordinarily require at least three runs to make necessary changes, with the third finishing alterations to the data and then moving into statistical analyses. Since JOBS can place any file into "kept" status, intermediate outputs would typically be kept to be retrieved and then altered on the next run in the series.

8. The timing data is based on the assumption of an input data base of 50,000 words (tokens) comprising about 2,500 types after deinflection and editing. The data are assumed to have been based on 100 successive observations of approximately 500 words each.

(Received for publication February 8, 1974;
revision received April 20, 1974.)