C. CONTEXT

by

John B. Smith

1. CONTEXT is a package of programs that attempt to show the way in which the important terms of a natural language text inter-relate and combine to form the larger substantive themes of that text. The primary emphasis of VIA in the past has been to define major themes by finding and laying out lists of related terms that appear in a document or segment of a document using various Thesauri.* To use an analogy, this process might be considered similar to discovering and displaying the various "bones" that are present in the structure or "skeleton" of substantive ideas that underlie a text. CONTEXT attempts to show how these various segments or "themes" fit together, how they are inter-related. The patterns of inter-related terms not only mark strong stylistic characteristics but also are quite revealing as to the actual content of the piece.

More specifically, CONTEXT looks at small subsections of text (the size of the subsection is determined by the user) to see whether or not any of a list of the most "important" words of the text are present. Such a list or lists is provided by VIA as output, and may be used as CONTEXT input

*(For a detailed description of VIA see S.Y. Sedelow, et al., Automated Language Analysis: 1967-1968).

if the researcher chooses. The program then examines all such subsections to determine the consistence with which various words are used together. The factors or patterns that emerge are quite indicative of the way in which the author puts together individual words or ideas to form larger themes. These factors are determined by using a standard Principal Component Analysis program.* (Similar "canned" factor analysis procedures are available at most computation centers). Input into this program is a list of numbers, or matrix, where the numbers in each row represent the number of occurrences of each term being examined in a particular subsection of text. There are as many rows of numbers as they are subsections of text.

---

*The remainder of Section II.C is intended for the reader with very limited mathematics. Therefore, the presentation is intuitive and highly analogous. For a more detailed and rigorous mathematical treatment see Harry H. Harman's Modern Factor Analysis, University of Chicago Press, 1967, 2nd Revised Edition.

Factor analytic studies of word clusters have been successfully conducted in several other fields of research. Many of the social and behavioral science journals carry articles concerning such studies; some that might be of particular interest to the reader are The American Journal of Psychology, Educational and Psychological Measurement, and Psychometrika. One effort that warrants specific reference is that of Drs. Howard Iker and Norman Harway. While at the University of Rochester Medical School they used techniques quite similar to those of CONTEXT to examine the patterns of associated ideas in transcribed psychotherapy sessions. (See "A Computer Approach Towards the Analysis of Content," Behavioral Sciences X, # 2 (4/65), pp. 173-182).

	$word_1$	$word_2$	$word_3$	. . . . .	$word_m$
$section_1$	f_{11}	f_{12}	f_{13}	. . . . .	f_{1m}
$section_2$	f_{21}	f_{22}	f_{23}	. . . . .	f_{2m}
$section_3$	f_{31}	f_{32}	f_{33}	. . . . .	f_{3m}
.	.	.	.		
.	.	.	.		
.	.	.	.		
$section_n$	f_{n1}	f_{n2}	f_{n3}	. . . . .	f_{nm}

Thus if one is interested in M different words or "subthemes" and divides the text into N subsections, then the matrix is MxN and there are N·M individual elements in it. The factor analysis program* looks at each pair of words in all subsections and assigns the pair a value ranging from -1 to +1 (this value is called a correlation coefficient). If the terms consistently occur together in the same context the correlation coefficient would be near +1; if the terms never occur in the same environment the correlation coefficient would be near -1; a random occurrence of the terms with regard to one another would result in a correlation coefficient near 0. Thus the NxM matrix reduces to a square MxM matrix called the correlation matrix.

*CONTEXT, as I have stated, uses what is actually a principal component procedure; however, I shall use the more general term, factor analysis, in referring to this data reduction technique.

	$word_1$	$word_2$	$word_3$	. . . . .	$word_m$
$word_1$	a_{11}	a_{12}	a_{13}	. . . . .	a_{1m}
$word_2$	a_{21}	a_{22}	a_{23}	. . . . .	a_{2m}
$word_3$	a_{31}	a_{32}	a_{33}	. . . . .	a_{3m}
.	.	.	.		
.	.	.	.		
.	.	.	.		
$word_m$	a_{m1}	a_{m2}	a_{m3}	. . . . .	a_{mm}

Here it is probably easiest to understand the process if we switch to a geometric or vector model. One may regard each row of the correlation matrix as a set of numbers ordered by their position (a_{11}, a_{12}, ----, a_{1m}, etc.), or as a point in a Euclidean space of dimension M, or as a vector. If one regards each row as a vector, then the set of all M vectors (one for each row) will generate a space of dimension D, such that $D \leq M$.

For example, the three vectors

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{matrix} alpha_1 \\ alpha_2 \\ alpha_3 \end{matrix}$$

could be said to generate the usually three dimensional Euclidean space

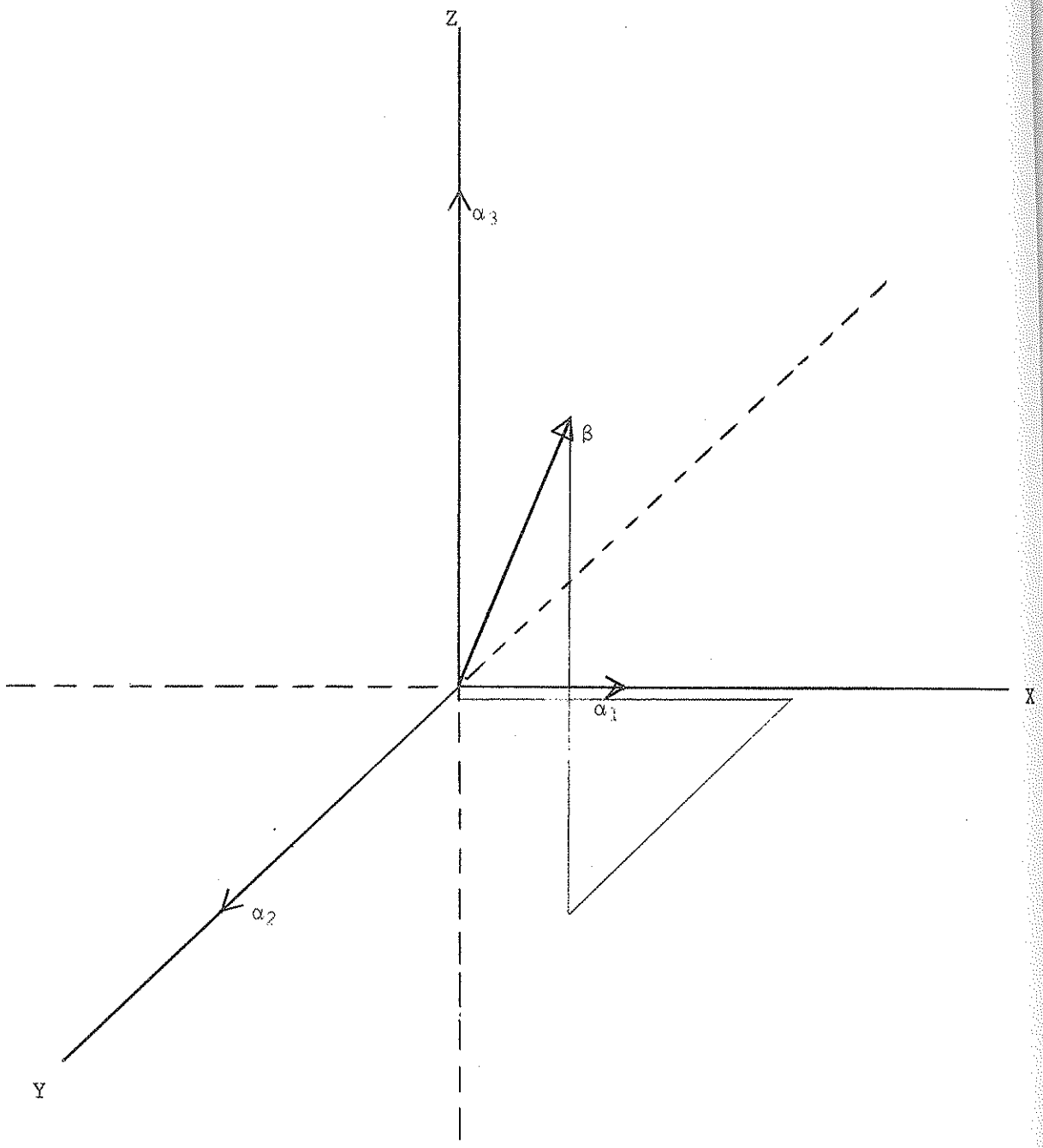


Figure 5

since any point or any vector in the space could be generated by taking linear combinations of the three vectors given. For example $\beta = (2,2,3)$ can be represented by $2\alpha_1 + \alpha_2 + \alpha_3 = 2(1,0,0) + (0,2,0) + (0,0,3) = (2,2,3)$. In general, then, N vectors will generate a space of dimensionality less than or equal to N.

The factor analysis model seeks a group of vectors, formed by various combinations of the original vectors, that comes closest to generating the original space of the correlation matrix. This approximation is close when a number of original vectors lie relatively near to one another. In 2-space this process might be represented as follows:

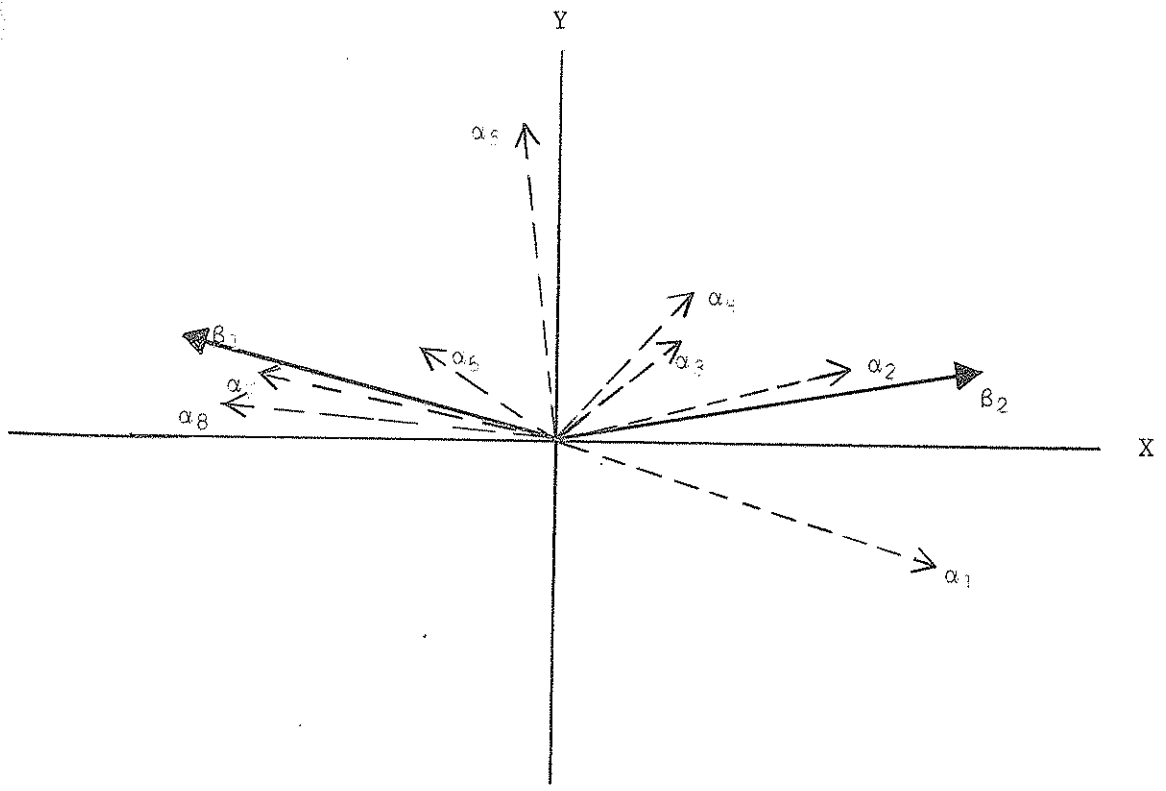


Figure 6

where the original vectors, $(\alpha_1, \alpha_2, \cdots, \alpha_8)$ might be approximated or reduced to β_1, β_2, with α_5 probably left over.

What the program actually produces is a set of column vectors (or factor loadings) of the form

$word_1$	a_{11}
$word_2$	a_{21}
$word_3$	a_{31}
.	.
$word_n$	a_{n1}

Each element or weight of the factor represents the degree to which the particular variable (word in our case) contributes to the vector. Thus individual factors can be thought to be most strongly characterized by those variables or words which contribute the largest weights. A negative weight implies the absence of that variable (or word) in conjunction with the variables or words that most strongly characterize the factor.

For example:

$word_1$	.05
$word_2$	.91
$word_3$	.63
$word_4$	.81
$word_5$	.53
$word_6$	.66
$word_7$	.32
$word_8$	.21
$word_9$	-.55
$word_{10}$	.11

This factor is most clearly defined in conjunction with words 2, 4, 6, 3, 9, and 5; however, word 9 consistently does not appear in context with the others.

2. Each factor in the CONTEXT model represents an interrelation of words or smaller themes that consistently occur together. Thus each might be thought of as a characteristic "large" theme of the document. We are currently testing CONTEXT on several texts: James Joyce's A Portrait of the Artist as a Young Man and the Praeger translation of Soviet Military Strategy. In the first chapter of the Portrait, for example, almost from the very beginning verbal motifs are introduced that play beneath the texture of the entire novel. One such motif concerns fear and retribution associated most immediately with birds and Stephan's eyes. The motif is introduced by the refrain

Pull out his eyes,
Apologise,
- - - - - - - - -

This same note of tension and fear prevades the chapter. Stephan is sent to a Jesuit school at Clongowes where he spends a period in the infirmary, goes home for Christmas, and returns to school. Upon his return a significant series of events begins. Stephan accidently breaks his glasses; because he cannot see to write he is unjustly punished with a pandybat. At the end of the chapter Stephan escapes the tyranny of Authority. What is most significant about Joyce's

narrative is not just the series of events but the patterns of associations that they raise for Stephan. These patterns of association are, of course, both a strong stylistic feature of Joyce's writing style as well as a substantive aspect of the content of the novel. Many of these patterns of associations are reflected quite strongly in the factors developed by CONTEXT.

The motif of fear mentioned above can be seen in the following factor:

#4

Apologise	.826
eyes	.807
out	.671
player	.268
light	.260

And the stay in the infirmary can be seen in the following:

#5

brother	.919
Michael	.891
queer	.541
infirmary	.294
call	.290

in which Stephan is most impressed by the unusual ("queer" is his word) habits of the attending Jesuit, Brother Michael.

Several factors reflect the events leading up to the pandybat episode:

#8		#17	
broke	.790	Father	.779
glass(es)	.783	Arnall	.766
write	.379	write	.402
did	.294	want	.286
home	.256		

These factors can be seen to reflect the association between the broken glasses and writing (it was in Father Arnall's class where Stephen was "pandied"). The punishment itself can be seen most graphically in factor #3:

pain	.800
pandybat	.712
sound	.712
loud	.569
down	.352
hand	.343
feel	.331
differ	.277
felt	.230
quick	.211

The very interesting aspect of this factor is that the pain of the pandybat striking Stephan's open hand is explicitly identified with the loud sound that the bat makes. This

identification among various senses is both a strong stylistic feature of Joyce's writing as well as an important substantive element of The Portrait. A number of other factors can be seen to reflect various aspects of the novel, but these illustrations should give the reader some idea of how CONTEXT currently functions.

3. Specific Programs of CONTEXT:

For convenience, CONTEXT has been referred to in this paper as a single program. Actually it consists of several individual procedures or programs, some of which are slight modifications of programs already in the VIA package. The normal VIA indexing program, INDEX, had to be modified slightly to facilitate the picking out of the immediate environment of words. INDEX, originally set up to allow convenient manual cross-referencing, establishes sequence in terms of numbers for volume, chapter, paragraph, sentence, and word, for prose; CONTEXT uses a modified form of INDEX, called LINDEX, in which words are merely numbered linearly--the first word is numbered 1, the second 2, etc. until the end of the text. This indexing scheme greatly simplifies the task of determing the "distance" between words. After indexing, the text is sorted alphabetically using the standard S/360 sort package. The sorted records are then fed into a suffixing program that groups words according to root. Thus, complete, completely, completing, etc. would be grouped together. All such forms are identified by a unique, five digit number called a MATCHCOUNT; however, the word itself is left as it originally appears. The words are again sorted, this time on matchcount and secondarily on the indexing sequence. The text then goes through an updating program, STATEX, in which the frequencies attached to each record are updated to correspond to the total frequency of all tokens for the same MATCHCOUNT. Also, STATEX computes the mean, standard

deviation, and prints out a distribution table. The final data preparation step occurs in PRETEXT in which all words occurring over a certain frequency (expressed either in absolute terms or in terms of m + n(s.d.)) are selected for context study. This list is edited against both inclusion and exclusion lists that are furnished by the user. PRETEXT then computes a large matrix in which each row lists the frequency of occurrence of each of the words within a particular text segment (100 word, 500 words, or whatever unit the user specifies). Finally, this matrix is fed into a standard principal component program for analysis.

LINDEX

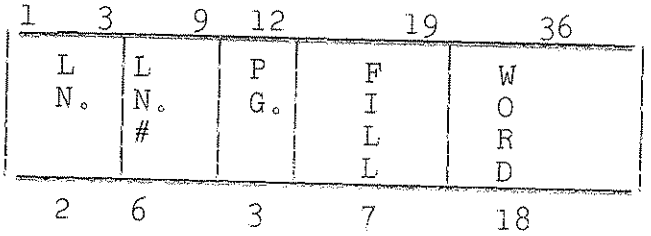
LINDEX is an indexing program based upon INDEX, described in detail in the annual report for 1967-68. It differs primarily in that the indexing information attached to each record is a six digit, sequential number. By indexing on linear sequence (as opposed to volume, chapter, paragraph, sentence, word, etc.), one can easily determine numerical units (50 words, 100 words, etc.) for designating subsections of text or word environments. This capability is essential for examining environmental correlation and for content analysis. (See section on CONTEXT).

INPUT: Input must be in 80 character records (usually punched cards or magnetic tape). The only restriction placed upon the text is that it be blank delimited. That is, every unit, including punctuation, that the user wishes the computer

to recognize must be separated by blanks from other units. (Example: . . . must be separated by blanksΔ.). Thus the program will take prose, poetry, speech transcription, etc., so long as it is blank delimited. Shift characters (such as ">", "]", or "<") may be used to indicate different type fonts, stage directions, etc.; however, if the user wishes to have these deleted from the data, he must list them individually in statement 4 of LINDEX.

OUTPUT. Output consists of fixed length records, one word or punctuation mark per record.

FORMAT.



The data set may be either put on tape directly or passed to a temporary storage location (usually a 2314 disk pack), sorted alphabetically, and then put onto tape. (Again, with slight modification of Job Control Language, data may be passed directly to subsequent programs such as PREFIX and SUFFIX.)

MAIN PROGRAM. A cardimage is read into a 1 x 80 array called CARDIMAGE. A subprocedure, FORM, then begins at column 71 (cols. 72-80 are reserved for page numbers, sequence numbers, etc.) and concatenates letters until a blank is found.

No provisions are made for word continuation from one card to the next. Therefore, if a word cannot be completed by column 71, blanks should be left at the end of the card and the word should be punched on the subsequent card. This word is returned to the main procedure. Accompanying each record is the page number of the word (for manual reference) along with a six digit linear sequence number. The latter is incremented by one for each new word returned.

ALPHABETICAL SORT: The sort used is one of the standard sorts of the System 360 sort package that is called through Job Control Language. For a detailed description of the options available, one should consult the IBM sort-merge manual. Records are sorted on the field beginning in column 19 of each record, for a field of 18 characters, with the sort in ascending order.

SUFFEX

SUFFEX is a slightly modified form of SUFFIX, described in detail in Automated Language Analysis: 1967-1968.

This form differs in the following respects. The original program, SUFFIX, printed for each word-token the indexing information, but the data set passed to later programs dropped this information. Thus the passed data set became a type (not token) data set with one entry for each unique word-type in the text. The modified version, SUFFEX, differs in that it takes an "exit" at the print statement of the older program and creates a record for each token that is passed either on tape

or disk to later programs. The record format is as follows:

1 3	9 12	17 21	38
LENGTH	LINEAR SEQ. #	PAGE	MATCH COUNT	FREQ	WORD
2	6	3	5	4	18

Thus the output is essentially an "updated" data set with matchcounts and frequency of word-type added to the records.

STATEX

STATEX is an interface program that runs between the suffix program and CONTEXT. The frequency counts attached to each record in SUFFEX were frequencies of word type; the updating of these counts so that they represent the total frequency of all word tokens for a root or matchcount (not just a word type) is done in STATEX.

In addition to updating these counts, STATEX computes the mean and standard deviation of frequencies of a text data set. For data sets with similar distributional patterns, thresholds may be set in terms of mean + n(s.d.). By doing this, the user is unable to set thresholds proportionally for data sets of different sizes. STATEX also keeps track of the number of roots (actually matchcounts) for each frequency interval. This information is printed out in table form which may in turn be used in regression procedures that "fit" a curve to the data.

Our preliminary results imply that the relative locations of various vocabulary items within the patterns for various texts would provide parameters for determining stylistic variations as well as content. Similarly the distributional patterns themselves might be useful in such analyses. For example, the distributional pattern for a fairly small section of text taken from Joyce's A Portrait of the Artist is a rather close approximation of a negative binomial expansion. If the distributional patterns for various texts can be approximated by standard statistical functions, then the defining parameters might well serve as author and content discriminators. However, let me emphasize that our research in this direction is just beginning and any results that we have at this point are tentative.

INPUT:

It is assumed that the text data set has been processed by SUFFEX and that records are in matchcount order. (See discussion of SUFFEX above).

OUTPUT:

Output is identical to input except that frequency counts have been updated so that the frequency represents that of all tokens with the same matchcount. For example, if there are 28 occurrences of complete, 16 of completely, and 4 of completed, each record of these variant forms of the same matchcount would be updated so that the frequency carried would be 48. This process would facilitate the selection process for programs

making searches based on frequency thresholds. There is additional printed output of data described in the discussion of the main programs.

MAIN PROGRAM:

Records are read into a structure until they differ in matchcount. The structure,TEMP, has room for 500 records. Each element is of the form:

```
(01 TEMP (500))
        02  JUNK    CHARACTER (11)----holds portion of record,
                                       not used in STATEX, that
                                       must be passed to PRETEXT.
        02  MATCH   FIXED DECIMAL (5)-Matchcount number
        02  FREQ    FIXED DECIMAL(4)--frequency of each word
                                       type
        02  WORD    CHARACTER(18)----text word
```

When a matchcount does not correspond with the previous matchcount, processing falls into the main execution loop. Beginning with the first entry in TEMP a check is made of subsequent pairs of words. When a mismatch occurs, the frequencies of the two words are added together since the frequencies attached to each record are the frequencies of that word type, not matchcount. The process continues as long as the matchcounts are the same. When the process is completed, the total frequency is placed in the FREQ slot for all tokens and the records are put out onto tape or disk. At this time several other counters are incremented. The formula used for

computing the standard deviation is a function of the total frequency of all tokens and also the square of the number of tokens of each matchcount. Therefore, several running totals are kept: one, of total number of tokens, is incremented by merely adding the frequency count to the previous total; the sum of frequency squared is similarly incremented, but by adding to the previous sum the square of the frequency for the matchcount. To facilitate computing the mean, a count of the number of unique matchcounts is also kept.

The frequency distribution that is computed is of the following form:

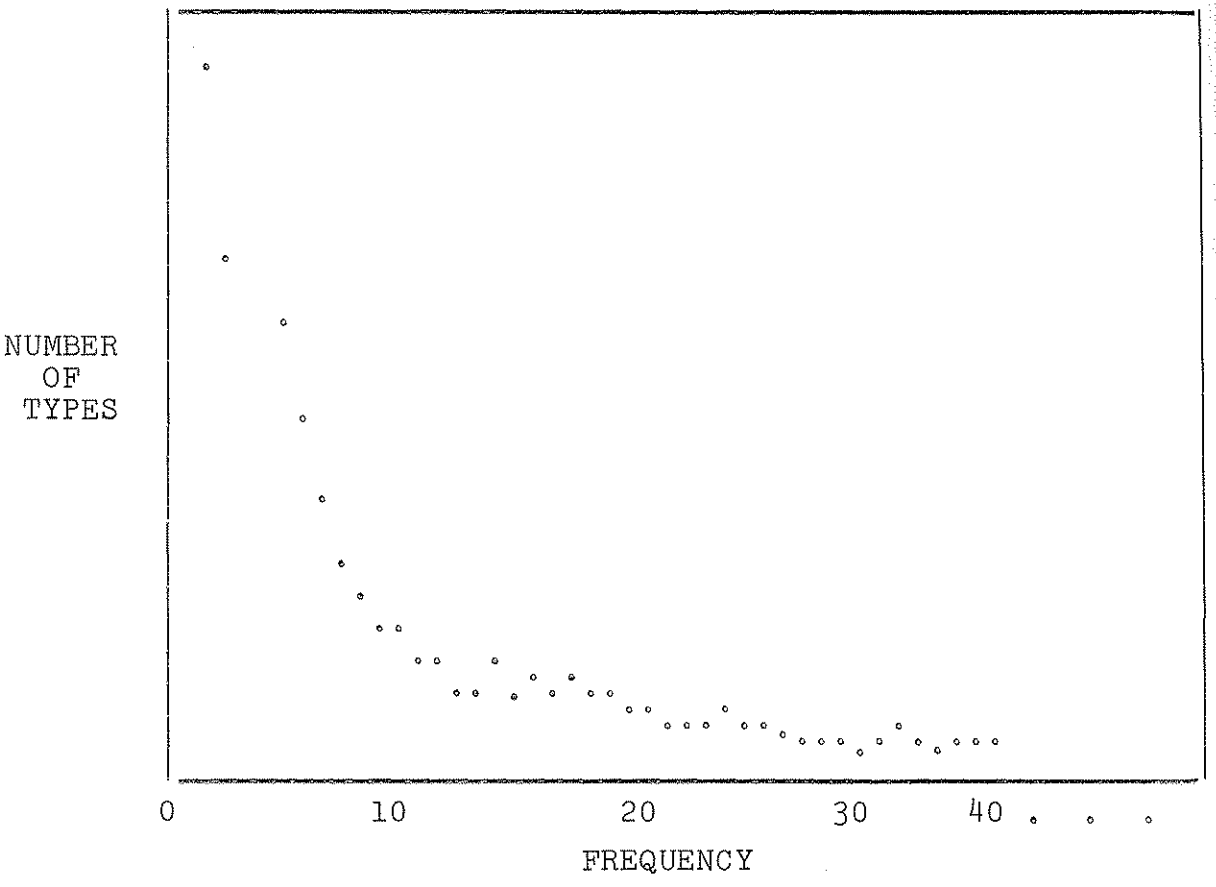


Figure 7

Each point represents the number of matchcounts for each unit frequency; i.e. the number of matchcounts that occur once, twice, etc. This information is kept in a 1 x 2000 array.*

On endfile, the program computes the mean (mean = freq. ÷ number of types) and standard deviation.

$$S.D. = \frac{\sqrt{N(\sum(freq.)^2 - (freq.)^2}}{N}$$

Thresholds for CONTEXT searches can then be expressed in terms of these statistics and passed to PRETEXT. Finally, the program prints the table of distributional data.

PRETEXT

PRETEXT is the main data preparation program in the CONTEXT sequence. It is the program that computes the data matrix that is actually passed to the "canned" principal component program.

INPUT: Data Records are the updated output of STATEX, with frequencies denoting total tokens per matchcount. It is assumed that records are sorted on matchcount, word, and linear number, all in ascending order.

Also passed from STATEX are a threshold frequency computed from the formula Threshold (CUT = mean + n(standard deviations) for a user specified n, the maximum number of tokens for a

*I assume that no matchcount will have a frequency count greater than 2000. Thus for each computational cycle, the counter in the array (called T) corresponding to the total frequency of the matchcount is incremented by one. (In the example of Complete etc. above, the corresponding counter would be in T(48).

single matchcount (used in allocating the dimensions of a major storage structure), and the number of matchcounts greater than or equal to the threshold. Also read in are two lists of matchcounts: one an inclusion list, to be used regardless of frequency, and the other an exclusion list, used similarly.

MAIN PROCEDURE:

The inclusion and exclusion edit lists are read into one dimensional arrays. Then records are read in one at a time. If the frequency of the record is greater than the frequency threshold, the program calls the EDOUT subprocedure to make sure that the user has not edited out this word.* If the word is not edited out by EDOUT, it along with the matchcount and its location are loaded into the structure LOCAT.

```
02  WORD CHARACTER (6)
02  MATCH FIXED DECIMAL (5)
02  NLOC FIXED DECIMAL (3)
02  LOC (max) FIXED DECIMAL (6)
    (one slot for each location).
```

A variable, LSTMAT, is set equal to the matchcount so that the next MATCHCOUNT can be tested directly against this variable instead of going through the whole procedure outlined above.

If the frequency of the incoming record is less than the threshold, the EDIN procedure is called. If the MATCHCOUNT is found, the record is loaded into LOCAT and processing continues as above; if not, a variable, REJECT, is set equal to the MATCHCOUNT and all subsequent records with this matchcount

*For example, words like said, here, etc. are of very high frequency in some texts, but of little thematic interest. They are not discarded in the function word edit facility of SUFFEX, but the user may wish to edit them out of the context analysis procedure.

are similarly rejected.

On Endfile a series of processes begins. First the locations under each matchcount are sorted to be sure they are in ascending order. Then a process begins that determines the number of times a particular matchcount appears within a specified environment (for example, within 50 words) of every other MATCHCOUNT. This information, stored in a square matrix called DATAC, is printed out for manual reference. After print out, this storage area is freed.

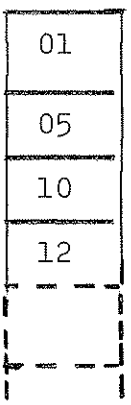
Finally, the program constructs the data matrix that serves as input into the principal component program. The user specifies the unit of text that he desires (for example, he may wish to divide the text up into 100 word chunks). Each row will have an entry for each word or matchcount selected above. Each entry represents the number of times that a particular matchcount occurs in a particular section of text. The principal component program requires that this data be received row by row. However, it turns out that with a unit of 100 words for, say, 160 matchcounts that the matrix is too large to be held in the computer. Furthermore, it will be seen that it is most convenient to compute the matrix by columns. If this is done, then the matrix cannot be easily manipulated if output is on tape or disk. The problem was solved by a rather interesting technique. Only some 20% of the elements of the matrix are non-zero. Therefore a matrix with the dimension of the matrix to be output is declared, but as a bit matrix.

(Usually a computer byte--that would hold an A, 1, etc.--consists of eight bits--0 or 1). It would have the following form:

```
0   1   0   0   1   .   .   .   .   0
1   0   0   1   0   .   .   .   .   1
1   1   0   1   0   .   .   .   .   0
.
.
.
.
.
.
1   0   0   1   0   .   .   .   .   0
```

Each element may be either 1 or 0.

Next a one dimensional array, LFIELD, is declared Fixed decimal (2) with as many elements as the total number of tokens loaded into LOCAT. It would look like this:



Similarly a dummy "column" of the matrix is declared as well as a counter for each column of the matrix.

Finally, processing begins at the top of LOCAT. Each location for a particular matchcount is divided by the unit (say 100). The (result + 1)th location in the dummy column is incremented by one. (For example, word 568/100 = 5 + 1 = 6: the 6th slot of the column is incremented to show that the particular word or matchcount occurs in the 6th unit of text). When all locations for a particular word have been processed, the corresponding elements in the large bit matrix are changed from 1 to 0; and, beginning at the top of the column, each non-zero element is loaded into the long LFIELD array. This process is repeated until all matchcounts are processed. Then the total number of 1's in each column is computed. Thus a 1600 x 1000 matrix capable of holding two digit numbers can be held in approximately 60,000 bytes instead of 320,000.

From this information, the matrix to be processed by the factor analysis program is constructed a row at a time and put out onto tape or disk. Reconstruction follows this form: for element a_{ij}

$$N = \sum_{j=1}^{i=1} \text{COLUMNTOTAL}(j) + \sum_{m=1}^{i} a_{mj} \neq 0 \quad .$$