SOME PARAMETERS FOR COMPUTATIONAL STYLISTICS: COMPUTER AIDS TO THE USE OF TRADITIONAL CATEGORIES IN STYLISTIC ANALYSIS[1]

Sally Yeates Sedelow
Saint Louis University
St. Louis, Missouri
and consultant
System Development Corporation
Santa Monica, California

Walter A. Sedelow, Jr.
Saint Louis University
St. Louis, Missouri

Terry Ruggles[2]
System Development Corporation
Santa Monica, California

"Computational Stylistics," the use of computers for a rigorous description and analysis of pattern attributes of text, is the result of a desire to give contemporary technology focused upon language processing the benefit of analytical methods devised and used by literary critics and, at the same time, to give literary critics the benefit of tools provided by contemporary technology--i.e., the large digital computer and new programming procedures.

It has been distressing to observe that while our colleagues in linguistics have been making major contributions to such fields as machine translation and information retrieval and, in return, gaining important insights into the structure of language,[3] we in literature

[1] The research reported herein was supported in part by the Office of Naval Research, Contract Nonr-4427(00).

[2] A significant debt for the programming of the experimental LISP program described in this paper is owed to Daniel Bobrow, a member of the M.I.T. faculty.

[3] Significant work on dependency analysis undertaken at the RAND Corporation by David Hays, *et al.* is a case in point.

have offered very little and gained very little. In the field of machine translation, for instance, following the brute force "solution" to the problem of meaning (i.e., some form of dictionary look-up technique), efforts have been concentrated upon devising algorithms for syntactic analysis--a problem which obviously is of primary importance. While "Computational Stylistics" is of both general and theoretical relevance for a large segment of the spectrum of literary and linguistic investigations, machine translation is one of its currently important applications; and, prior to the appearance of the "Preface to Computational Stylistics"[4] and the work on the programs described in this paper, there was little or no attention directed to extra-syntactic aspects of style as a property of machine translation. We all know that a major concern of the skilled human translator is the search for stylistic equivalences between languages; syntactic correspondences are, to be sure, part of that concern but they are, we believe, the least of the skilled translator's worries.[5] Problems of form and texture (including semantics) and of rhythm are the paramount considerations. Without attention to the parameters of language usage which they imply, translations are likely to be so deadly dull as to be worth far less than they would be otherwise. And yet, the sad fact is that although work on machine translation has been in progress since the early 'fifties, it is only recently that

[4] Sally Yeates Sedelow and Walter A. Sedelow, Jr. System Development Corporation, SP-1534, February 17, 1964.

[5] For a good recent discussion of the problems of the translator, see Reuben A. Brower, ed., On Translation, Cambridge, Mass.: Havard University Press, 1959.

a determined effort has been made to direct attention to these important stylistic characteristics of language. It may be argued that this work should have been concurrent with that devoted to problems of syntax; because it was not, we may well have a longer wait before the achievement of adequate machine translations. Machine translation is an example of an applied research activity to which, in principle, literary critics have much to contribute; another such activity is automatic abstracting. Obviously there are many others. In many--perhaps most--phases of contemporary life some form or other of language processing is a relevant, not to say predominant, characteristic; and there is, then, good reason for believing that literary critics may have a great deal to offer to students of these socio-linguistic functions.

These opening remarks might seem to have only a tangential bearing on computers and literature, the theme of this conference. But the fact that no one would deny that some manual translations are literature serves to emphasize the very close connection and, in some cases, identity between certain traditional concerns of literary critics and new, pressing problems of such disparate groups as linguists (including structural linguists, socio-linguists, psycho-linguists, anthropological linguists, _et_ _al_.), military intelligence officers, physical scientists, and many others. It is as we bear all these needs and interests in mind that we will benefit most from the introduction or, some might say, obtrusion of computers into our midst. Others' needs oblige us to re-examine our methods and criteria with a resultant expansion of our critical repertoire or, at the least, an increased precision in its use. It may well be that precision in the use of the critics' vocabulary will be a primary serendipitous consequence of the application of artificial intelligence heuristics to the study of literature. Just as the application of the more formally rigorous symbolic logic to the comparatively more theoretically fragmented formulations of mathematicians has disclosed, and is disclosing, analytical hiatuses, the discovery of which has enriched mathematics, so in a number of the traditional academic disciplines the formulation of problems with the rigor necessary to meet the high standards imposed by computer-aided analyses has contributed to a more accurate and complete analytical structure. Experimental psychology and inter-industry economics might be instanced. The literary-critical methods of our discipline stand to benefit similarly.

Our own interest in computers has resulted from the desire to study more carefully such literary problems as change in an author's style through time, as well as the differentiating properties of individual authors' styles. The works of Milton and Hume have occupied our attention in this connection. An interest in the close analysis of conversational discourse as a structuring factor in social interaction has been another incentive.

Rhythm, Texture, and Form

Now we should like to give further point to the introductory remarks by putting them in the context of the programs we have been developing for computational stylistics. The first program is called MAPTEXT (earlier called MAPSCENE, and so named in conjunction with the initial experimental program's use on some scenes from *Hamlet*[6]). As its name suggests, MAPTEXT translates or maps an input text from its verbal form into another, abstract representation. For example, in one representation a content word may be replaced by an O, and a function word by an *. Or, if one wishes to have a non-verbal, graphic representation of the occurrence of a particular word, or particular group of words (for instance, function words such as prepositions or conjunctions; or words with a certain minimum number of syllables), each word, type of word, or group of words would be represented by an assigned symbol. MAPTEXT provides the scholar with the opportunity to look at the text in a number of different forms without the distraction, in each case, of other verbal relationships. Bearing in mind this general description of the program, we should like to talk about MAPTEXT as it might be used to look for patterns of rhythm, of texture, and of form.

[6] The experimental program is described in "A LISP Program for Use in Stylistic Analysis," Sally Yeates Sedelow and Daniel G. Bobrow, System Development Corporation, TM-1753, 17 February 1964. FORTRAN is now being used in place of LISP because FORTRAN has auxiliary storage capabilities and is widely available (*vide infra* for further comment).

With reference to the problem of rhythm, MAPTEXT can be used for broad delineations. As is the case with so many stylistic parameters, there is no very clear definition of rhythm; scholars talk about it in terms of stress, of pitch, of word length, and of units such as the sentence or paragraph. As soon as we look at the first item on this list--stress--and think about using the computer to recognize patterns of stress, we discover the indefiniteness of traditional definitions. Presumably stress is something we all perceive; but unfortunately we are not quite sure what is being perceived. (Is it syllable length, or syllable pitch, or some combination of the two? If a combination, what--operationally speaking--is it?) And perhaps more unfortunately, one person's perception of stress does not necessarily coincide with that of another person. It is instructive to notice that skilled linguists with highly trained ears indicate patterns of stress that other skilled linguists with highly trained ears do not hear that way at all. It is quite likely that the computer will be used to help resolve this problem; a computer will be programmed to examine sound spectrographs, or something of the sort, and make correlations between patterns in the spectrographs and modal human perceptions of stress. Once this is achieved there will be a standard to which to appeal and from which to depart--with defined exactness. In the meantime, so far as syllabic stress is concerned the text to be fed into the computer would have to include designations, which had been assigned manually, as to stress. Units such as word length, inter-comma string length, sentence length, or paragraph length can be very easily handled by the computer; further, the computer can cope with very lengthy sequential patterns of diverse units which could elude even the careful reader. The use of MAPTEXT helps us to see the validity in defining rhythm as the regular recurrence of, or regular alternation in, linguistic features. Given this definition and the representation of a text in terms of content and function words, for instance, one can then look for subtle content-function rhythms. Or one can explore patterns of punctuation, the use of specific words, or any linguistic feature or complex pattern of features--whether deterministic or probabilistic. Once a promising pattern has been revealed, the computer can be used to perform statistical analyses; the analyses can be performed either upon the abstracted form of the pattern or upon the input text itself. MAPTEXT will have provided a sequentially multidimensional view of the text; the specified representations may suggest others not specified but nonetheless apparent in the graphic form. Statistical summations alone, while evidently useful, are not so likely to add new dimensions to the researcher's knowledge nor, therefore, to suggest new directions for research as are a combination of the statistical summations <u>and</u> the MAPTEXT representations.

Texture has been perhaps even less rigorously defined than rhythm. The critical term the authors of this paper most closely associate with texture is another vaguely used word, tone. Insofar as texture and tone can be separated from other characteristics of style, such as rhythm and form, they probably have semantic connotations. For instance, the songs of swains provide acceptable tone in pastoral verse but one would be rather surprised to find them in naturalistic imagery of the city. MAPTEXT's ability to trace occurrences of specific words and, what is more important, a number of specific words at the same time could make it helpful in exploring this aspect of texture. While it is possible for a scholar to delineate complex mosaics of images and image fragments, this laborious task can be done more efficiently and expeditiously by the computer. Further, MAPTEXT may be of some slight aid toward the solution of the semantic problem which looms as a major obstacle to computerized analysis of language. It may be that just as certain patterns of syntax (such as the likelihood of an article preceding a noun) are apparent and have been used for automated parsing systems, so, too, patterns of more specific word usage will be found to exist--notably on a probability basis in a large population. If this should be the case, then we would have a rather different sort of semantic definition than that usually envisaged but one consistent with the traditional nature of much language usage.

In using MAPTEXT to help delineate form, one of its advantages is the variety of dimensions which it can cope with simultaneously and in full detail. The relative densities of any defined classes of words might be explored. How much more often do words of a given class occur in one part of the sentence than in another? One might do a computation such as, for example, using the middle of the sentence as the mean and seeing how far the standard deviation of the given class of words differs from that of another class. Or, since we would argue that rhythm, although discussed separately here, is a part of form, one might use MAPTEXT to see whether or not sentence markers, or major stops such as semicolons, revealed patterns of form in a text under scrutiny. One might also wish to compute distances between the recurrences of specific words, as well as between specific types of punctuation.

If one wanted to split up the words in the input text into syllables, spelled phonetically, MAPTEXT could be used for a representation of rhyme; this not only would be a good way to discover linkages between internal rhyme, head rhyme, and end rhyme, but also would be a particularly vivid way of representing or displaying them for teaching purposes. In fact, MAPTEXT representations in general may prove very powerful and suggestive pedagogical aids. Humanistic subjects have been relatively neglected in studies of computer-based aids to teaching, whether of college or pre-college classes; these diagrammatic printouts could be of sufficient use in the teaching of literary analysis to help redress the balance.

While we can use MAPTEXT for exclusively literary studies, we are also working on problems germane to machine translation and automatic abstracting. For adequate translations and abstracts, the computer will need algorithms for producing both idiomatic and idiosyncratic styles. MAPTEXT and statistical analyses subsequent to uses of MAPTEXT should provide valuable leads toward discovering the linguistic patterns necessary for such algorithms. Because a number of separate representational symbols can be used by MAPTEXT simultaneously, and because its parameters can be changed at will, it is an exceedingly flexible program.

Thematics

One aspect of form--theme, or organizing concept, or, possibly, imagery--has been reserved for the discussion of the VIA (Verbally-Indexed Associations) program. Critical procedures involved in the recognition of theme have been, at best, only hazily explicit. Presumably there would be agreement, however, that perception of theme is at least partially dependent upon the perception of semantic similarity among words in the text and partially dependent upon the perception of words contiguous with the semantically related words--so that theme, according to this minimal definition, is a function of semantic content and textual context. Because conventional thesauri are organized in terms of putative semantic relationships, we have chosen to use the thesaurus form as the basis for the VIA program. We take it, further, that semantic similarity is perceived in part in terms of word roots, i.e., words with the same root are likely to have meanings which have some connection with each other. Our VIA thesaurus, therefore, is constructed on the bases of (a) identical root, (b) synonymity and antonymity (when looking for the theme of *evil*, for instance, it is difficult to ignore--or, better, necessary to use--the contrasting theme of *good*), and (c) textual contiguity.

Any construction of any thesaurus runs head-on into the semantic problem. The VIA program copes with this problem by relying on a man-machine interaction which becomes increasingly less dependent upon human action. In other words, the VIA program can be described as a self-adapting system, or as a partially self-organizing system.[7]

In rather broad outline, the program works in the following way. First, a frequency count on all the content words in the text is performed. Words with the same root are grouped together and the frequency count is based on the root; that is, if there were three "madlys," two "madnesses" and one "mad," the count would be six, not three, two and one. A list of all the words that occur more than a specified number of times is then printed out. (If one wishes to look at every word that appears in a given text one can do so by making zero the specified number. If one wanted, for example, to look at the grouping of all the words according to root, one might want to use this option; normally, however, the specified number will be high enough to separate out frequently recurring themes, or concepts.) When the list of high frequency words is printed out, each word is accompanied by an index indicating the locations in the text where the word appeared. After the list of high-frequency words has been made available, conventional thesauri, Brown's <u>Composition of Scientific Words</u> (1956), dictionaries of synonyms, and the immediate textual context of the high-frequency words are explored for conceptually associated words. Each of the associated words which occurs is entered on a list headed by the high-frequency word with which it is conceptually associated.

[7] <u>Re</u> such systems see Heinz von Foerster and George W. Zopf, Jr. eds., "Principles of Self Organization," <u>Transactions of the University of Illinois Symposium</u>, Vol. 9 in International Tracts in Computer Science and Technology and Their Application. New York: Pergamon Press, 1962.

Thus, when we were working with a scene from <u>Hamlet</u>, the word "disease" appeared on a list headed by "mad." The lists of high-frequency words and their associates constitute the beginnings of the thesaurus, and these lists are saved (at present, on punched cards; ultimately, on tape) in a form to which the computer has access. After the next section of text is examined for high-frequency words, the computer then searches <u>its own thesaurus</u> for associated words. At present, the computer's procedure is to see whether the high-frequency word appears anywhere in its thesaurus; if it does, a search is made for all other words appearing in the same thesaurus entry. If one or more of the other words in the entry heads another entry, a textual search is made for the words appearing under that entry. For example, if the word "disease" is a high-frequency word, the input text will be searched for occurrences of the word "mad." If the word "insane" appeared in the "mad" entry and also headed another entry, the text would be searched for occurrences of "insane" as well as for occurrences of all words in the "insane" entry. As the computer's thesaurus builds up, the amount of manual searching for possible associated words is sharply reduced; this "learning" characteristic of the program is one of its great strengths and constitutes a very useful advance over other programmed techniques for thematic analysis. Searching procedures other than those described can also be used in this program; for instance, it may be desirable for the computer to save all suggested associated words, although only those appearing in the text would be part of the computer's thesaurus. Using this latter system, each subsequent input of text could be checked for the presence of words associated with earlier high-frequency words, even though some of the possible-associated-words did not occur earlier in the text. Also, optimum depths for carrying out searches within the thesaurus will need to be determined for particular applications. The end result of the VIA program is a thematic or conceptual outline, in the form of a cross-referenced thesaurus, of a particular literary work, or group of works by the same author, etc. The most important and distinctive aspects of this program are (a) the self-adapting characteristic, and (b) the construction of the thesaurus on the basis of words which appear in the work under examination. In other words, there is no pre-prepared word list, as is the case with thesauri conventionally used in information retrieval, which confines the search to those words which the prospective searcher guesses will appear.

In order to make clear certain details of the program's design and their rationale we turn now to some considerations of program system design economics. Earlier in this paper, the change from LISP to FORTRAN as a programming language was mentioned. LISP is a list-processing language and FORTRAN is sometimes described as a problem-oriented and sometimes as a procedure-oriented language. Both list-processing languages and languages such as FORTRAN can work on input data which is in the form of words. In list-processing languages, the data is conceptualized in terms of lists of words, and in a FORTRAN-type language in terms of tables or arrays of words.

In list-processing languages, the list of items can be linked to an item on another list or to another list, or to a series of lists; in FORTRAN-type languages, an item or many items in one table can be linked to an item or items in another table. By contrast with the table-using languages the chief characteristic of list-processing languages is that the items on a list are linked together "automatically," so that the programmer does not have to keep track of their location by means of still further tables. Furthermore, programming instructions are designed with list-processing in mind: one can, for instance, simply insert an item between two other items on a list, while in FORTRAN such a procedure is more complicated. The disadvantages of list-processing languages for literary data processing are (a) their operation takes more space in core memory because of the space required for linking addresses, (b) they tend to be rather slow, and (c) in the specific case of LISP, there is the disastrous disadvantage of having no access to auxiliary storage--disastrous, because the computer to which we have easiest access does not have the large central core memory required for processing large quantities of text. LISP was useful to us during the experimental phase of this investigation because the speed with which LISP programs can be written enabled us to test our ideas quickly. FORTRAN was chosen for the major project primarily because it has wide distribution and we want our programs to be usefully available to others. Also, it has fine input-output procedures and runs fairly quickly. One of its major disadvantages for us, because we are working with fixed-length computer words is that it does not yet have procedures for getting at sections of computer words; frequently one wants to look at a single textual letter, which does not require a computer word. Therefore, we have had to use machine-oriented coding for some of our procedures. Another disadvantage of FORTRAN and, with the exception of COMIT, almost all other programming languages, is that the manuals describing the languages have not described data-processing in terms of literal language; they talk in terms of numeric language. The translation from one to another is frequently not as clear as it readily could be; owing to the growth of interest in the processing of linguistic and literary data, such clarification would be a worthy project.

Another consideration in the design of programs for processing literary and linguistic data--and this is a computer-dependent variable--is that of economical system design for examining a large body of text. The two factors here are time and space--running time on the computer and adequate space for both the text one wishes to process and the programming system which processes it. We have traded off economy of time in order to insure ourselves of space adequate for the anticipated large volume of text. Our input text and our computer thesaurus are both stored on tape. Because the computer must proceed through a tape linearly, it goes much more slowly than if the text and thesaurus were stored in the individually-addressed cells of the central computer memory. Another aspect of economic system design is the sequencing of operations. For instance, for speed we should set up our MAPTEXT representation while we are doing some initial processing of our input text for the VIA program.

But because we wish ourselves and others to have separate access to the MAPTEXT program, we have written it as a separate program; further, if a large number of distinct operations are incorporated in one program, its size and complexity are not readily managed by a new user. Thus we have sacrificed the initial economy of programming time and possibly computer running time for the later economy of effort on the part of the users of the program. The design of economical searches is another problem; for instance, every time the computer processes a word, the word must be compared with the items on a list of approximately one hundred function words. Would it be better to group the most frequently appearing function words at the head of the list and process straight down the list, or would a so-called binary search be more efficient? We opted for a binary search because, on the average, only six comparisons would be necessary to locate the word. It is not likely that a grouping technique would be so efficient. Hash-indexing, the assignment of computer storage space on the basis of the value of, for instance, the sum of the first four characters of a word, was also a possibility; we rejected it because of the quantity of data being processed in a relatively small core memory.

The considerations mentioned thus far are general in nature and would apply in the course of the design of any sizeable system. There are, however, specific tasks handled by our VIA program which also may be of interest; how to get at word roots with efficiency is one of these. A procedure used by some researchers is simply to give the computer lists of suffixes, perhaps arranged first as to length and then as to alphabetical order and, for each input word, to match the last five letters against the five-letter-long suffixes, following which, if no match is found, to check the last four letters against the four-letter-long suffixes, etc. If one is dealing with a large input text, this procedure takes a lot of time. We finally settled on a procedure developed by Keren McConlogue of the System Development Corporation, and we are adapting and translating for our program her "word-matching" technique. This technique is to group together content words for which the first three letters are identical. Then additional letters are matched until the words deviate from each other; the assumption is that the point of deviation may mark the end of the root form and the beginning of the suffix. One way to check the validity of this assumption would be to check the putative suffix against a suffix list. Objections to this procedure have already been made; simply removing the deviant sections of the words would lead to too many mistakes and mispairings. The procedure, therefore, is to work with two words at a time, selecting the suffix which alphabetically precedes the other, and then searching a relatively short suffix list for the suffix. If the suffix occurs, it is linked to possible "pairing" suffixes in another table. For instance, if the program were working with the words "rain" and "rainy," it would discover that the words matched through "n." The short suffix list would then be searched for "blank," a legal suffix in this system. "Blank" would be linked to the list of possible pairing suffixes in the other table and this list would be searched for a "y." When the "y" is found, the program would switch to another linked table, which lists exceptions to the suffix pair, "blank" and "y" ("bus," for instance, would be an exception). If the word does not occur in the exception list, the "y" is removed. Thus, to remove the "y," three short lists have been examined rather than the huge lists which would otherwise need to be searched, first to find the "y" and then to look through the many exceptions to the operating rule which in this case would be, simply, that every final "y" is a suffix. If the first "suffix" had not occurred in the initial table, then the program would proceed on the assumption that it had not found a suffix and the operations on that word would be abandoned. Also, no operations are conducted on words for which there are no "matches" for the initial three letters.

* * *

The translation of the VIA and MAPTEXT programs from LISP into FORTRAN is well under way; in fact, the initial version of the VIA program is just about checked out. The input for the experimental program was a brief scene (Act IV, Scene I) from _Hamlet_. Even though the scene was short and selected at random, the brief thesaurus appearing in the output has picked up the theme of madness associated with disease, often cited as a theme in that play. Also appearing in the thesaurus are the verbs "come" and "go"--stage directions in the text which actors should not miss but literary critics sometimes do.

We might jestingly note that, in the history of literary criticism, the road to the text is strewn with good inten~~s~~tions. It would be presumptuous of us to say that we have "arrived," but it is gratifying to note that with this computational stylistics approach, we are meeting some of the elemental conditions of general scientific method in that we will have "synthesized" through computer programs a repertory of analytical techniques which have the ultimate reliability implied by perfect replication--in this instance, a replication of critical functions. On such firm foundations, each of us can build to suit his critical taste.