[*International Conference on Machine Translation of Languages and Applied Language Analysis*, National Physical Laboratory, Teddington, UK, 5-8 September 1961]

AUTOMATIC LINGUISTIC ANALYSIS - A HEURISTIC PROBLEM

Paul L. Garvin
Ramo-Wooldridge Laboratories
Canoga Park, California

1. OUTLINE OF THE PROBLEM

An automatic linguistic analysis program is here defined as a computer program which, given as input a body of text, will produce as output a linguistic description of the system of the natural language which is represented by the text. A corollary capability of such a program will be the capacity for deciding whether or not a given input is indeed a text in a natural language. This paper reports on research now in progress at Ramo-Wooldridge.

*1.1 Parameters of Linguistic Description**

The system of a language can be conceived of as constituted by an orderly aggregate of various kinds of elements, each of which has a finite and typical set of co-occurrence possibilities with regard to other elements of the system. The elements are of different functional types and orders of complexity, as exemplified by such elements of written English as letters, syllables, words, or phrases. These elements recur in texts in a regular way, so as to form classes - called distribution classes - in terms of shared co-occurrence characteristics. The purpose of linguistic analysis is then to specify the nature and boundaries of the various types and orders of elements, as well as to describe the co-occurrence patterns serving as the criteria for the definition of the distribution classes, and to list the membership of these classes. The former aspect of linguistic analysis is often termed *segmentation,* the latter is called *distributional analysis.*

Linguistic segmentation is the first step in the analysis of raw text - that is, of spoken messages recorded from native informants. Segmentation procedures are based on the relation between the form (i.e., the phonetic shape) and the meaning (in operational terms, the translation or possible paraphrase) of the message. Their mechanization thus would require the comparative processing of two inputs - one representing the phonetic shape of the raw text, one its translation or paraphrase.

*A detailed discussion of the linguistic point of view underlying the present paper is contained in Paul L. Garvin "The Definitional Model of Language", in Paul L. Garvin, ed., "Natural Language and the Computer", in preparation by the McGraw-Hill Book Company, Inc., New York, N.Y.

A program designed for a single rather than a dual input hence cannot be expected to accomplish segmentation. On the contrary, it requires an input which has in some prior way been segmented into elements of a unified functional type. Given such a previously segmented input, the program can, however, be expected to accomplish a distributional analysis of the elements that are found to recur in the input text.

Ordinary written English, with its spaces between words and punctuation marks, is an obvious example of such a previously segmented text.

The intended output of such a distributional analysis program is a dictionary listing of all the elements (for instance, of all the printed words) found to recur in the input text, with each element in the listing accompanied by a grammar code reflecting the distributional description of the element in terms of the distribution class and subclass to which it belongs. Since the purpose of the program thus is to produce a grammar-coded dictionary listing, it is logically necessary to require that the program itself initially contain no dictionary or grammar code, but only the routines required for their compilation.

1.8 Procedures of Distributional Analysis

The basic question of distributional analysis is: does unit *a* occur in environment *b*? This question *can* be answered by a computer program. The problem is primarily one of specifying automatically what units *a* the question is to be asked about, and what environments *b* are to be considered in arriving at an answer.

In ordinary linguistic analysis, responses by native speakers (informants) are evaluated and text is examined "manually" in order to arrive at distributional descriptions. This is done by observing regularities in the recurrence of elements, from which can be inferred the presence of typical conditions of occurrence, called *diagnostic contexts,* that can serve as defining criteria for the determination of appropriate distribution classes of elements. Informant responses are manipulated in order to test for the occurrence of elements in diagnostic contexts, and in order to detect additional diagnostic contexts for further tests.

A major difficulty of these procedures of distributional analysis lies in the subjectivity inherent in such informant work. This subjectivity is maximized by the linguist using himself as an informant; it may be minimized by circumscribing the test situation very narrowly and by using a variety of informants, as well as by other controls. In spite of any safeguards that one may want to introduce, as the questions become more sophisticated, the informant's responses become more and more difficult to control and his memory becomes less and less reliable. Thus, even in ordinary linguistic analysis one, reaches a point where informant work has to be combined with the study of text.

The basic difficulty in the use of text for purposes of linguistic analysis is that large samples are required. This is understandable if one takes into account the inverse ratio of the recurrence of elements to the size of the sample; the less frequently the element recurs, the larger sample is required in order to study its distributional properties. Data processing equipment allows the processing of very large bodies of text using the same program, with the program assuming the role of the linguistic analyst.

1.3 Heuristic Aspects of Automatic Linguistic Analysis

A fully automatic distributional analysis program can be looked upon as a heuristic rather than a purely algorithmic problem. A.L. Samuel recently set forth some of the characteristics of an intellectual activity in which heuristic procedures and learning processes can play a major role. These are, as applied to the problem of playing checkers:

> "(1) The activity must not be deterministic in the practical sense. There exists no known algorithm which will guarantee a win or draw in checkers, and the complete exploitation of every possible path through a checker game would involve perhaps 10^{40} choices of moves which, at 3 choices per millimicrosecond would still take 10^{21} centuries to consider.
>
> (2) A definite goal must exist - the winning of the game - and at least one criterion of intermediate goal must exist which has bearing on the achievement of the final goal and for which the sign should be known....
>
> (3) The rules of the activity should be definite and they should be known....
>
> (4) There should be a background of knowledge of the activity against which the learning progress can be tested.
>
> (5) The activity should be one that is familiar to a substantial body of people so that the behavior of the program can be made understandable to them...."*

The above criteria seems to be applicable to automatic linguistic analysis as well. The can be paraphrased as follows:

(1) Linguistic analysis is not deterministic in the practical sense. There exists no known algorithm which will guarantee success in linguistic analysis and the complete exploitation of every possible combinatory criterion might involve an equally astronomical number of steps as the number of moves to be explored in a checkers algorithm.

(2) A definite goal does exist - a detailed distributional statement - and criteria can be formulated for intermediate goals that have bearing on the achievement of the final goal. These would be the broader distributional statements from which the ultimate, more refined classifications can be derived. Unlike checkers, the final goal can not be formulated as simply.

(3) The rules of the activity are definite and can be formulated. This, of course, presupposes that one accepts as a basic assumption the possibility of linguistic discovery procedures. The procedures which have proved to be most useful in linguistic analysis are those of substitution and dropping; they can be made computable, and they may be introduced into the heuristic linguistic analysis program after certain necessary preliminary steps have been completed. Other equally computable procedures can be formulated.

*A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal of Research and Development, July 1959, pp.211-212.

(4) There is, of course, a background of knowledge of the activity against which the machine results are tested: linguistic analysis is, or can be made into, an established field and machine results can be compared to human results.

(5) Although the activity of linguistic analysis is not one that is familiar to a substantial body of people, its results nonetheless can be compared to the intuitive behavior of an entire speech community and the behavior of the program can be explicated in terms of this observed intuitive behavior.

1.4 Summary and Evaluation

It is thus possible to envision a computer program which will process the initially segmented text by applying to it a variety of linguistic analytic procedures in turn, and will evaluate the results of each trial on the basis of certain built-in statistical or otherwise computable criteria. The program can be expected to accept certain trials and reject others on the basis of these criteria. The results of the initial tests performed by the program can then be utilized for the automatic formulation of additional tests leading to a more refined classification, until the potential of the program is exhausted and the output can be printed out for inspection by a competent linguistic evaluator.

From a linguistic standpoint, the development of automatic linguistic analysis will lead to a rigor hitherto not customary in the field of linguistics, as well as ultimately to a capability for processing much larger samples of language than the linguist can ever hope to examine manually. From the standpoint of the application of linguistic knowledge to language data processing, it will lead to a greatly increased reliability of the results of linguistic analysis.

Automatic linguistic analysis can be expected to be of particular interest to the description of languages in which word classes (that is, parts of speech as specified linguistically and not merely by traditional grammar) are not readily definable, and in which conspicuous formal marks of syntactic relations (such as clearly recognizable grammatical endings and other elements) are either absent or infrequent. Examples of such languages are Chinese, or for that matter, English.

From a computational standpoint, automatic linguistic analysis opens up a new and different area of application for heuristic programming.

2. DESIGN OF A HEURISTIC PROGRAM FOR AUTOMATIC LINGUISTIC ANALYSIS

2.l *Components of the Program*

As was intimated in Section 1.4 above, a heuristic program for automatic linguistic analysis can be designed in terms of the interplay of two major sets of routines:

(i) *Trial routines.* Sets of analytic routines can be designed in simulation of suitable procedures of linguistic analysis as presently conducted. The purpose of each particular analytic routine will be to

process the input text in order to produce a classification of the textual elements in terms of a set of criteria proper to that routine. Several routines, each based on a different set of criteria, can be used consecutively to process the same input text, yielding several alternative classifications. Each of these can be said to represent one of several trials for the completion of a particular step in the analysis. The program will then have to choose from among trial routines the one which has the greatest promise of allowing it to proceed to the next step (for details, see Section 2.3 below).

(ii) *Evaluation routines.* The purpose of these routines will be to apply to the results of alternative trial routines the criteria which the program needs for deciding which of the trials have been "successful" and which "unsuccessful" from the standpoint of the ultimate analytic goal. "Unsuccessful" results will be rejected, and "successful" results will be accepted by the program, enabling it to proceed to the next analytic step. The evaluation criteria will be derived from analytic linguistic observations and based on certain assumptions about the nature of linguistic data that are suggested by common sense and the accumulated experience of the linguistic profession (for details see Section 2.4 below).

2.2 *Over-all Design of the Program*

The distributional analysis which the program is intended to simulate, ideally constitutes a step-by-step process in which the first step yields a broad classification of the relevant elements, and every subsequent step yields a more detailed subclassification, until a set of classes and subclasses which cannot be further subdivided into smaller subclasses is arrived at. If the initial elements of the analytic process are words, the resulting classification will be one of words into linguistically defined word classes and subclasses.

Each step in the process of analysis consists of the application of one or more of a set of available analytic procedures, as well as - if more than one procedure has yielded results - of a decision as to which of the alternatively resulting classifications is to be retained. Each step subclassifies further the results of the last preceding step, and conversely produces the set of classes which are to be subclassified further by the next following step. The process of analysis can be considered completed when the procedures available at a given step no longer yield a subclassification into classes containing more than one element each: when every class resulting from a given step is a class of one, the process of analysis can be considered to have reached its logical endpoint.

In ordinary linguistic analysis, the acceptability of a given classification is usually judged intuitively, but as was stated under (ii) in Section 2.1 above, it is possible to formalize certain of the criteria for these impressionistic judgments.

The step-by-step character of linguistic analysis can be simulated by a pass method of programming.

To the steps of the analytic process can be made to correspond a series of consecutive passes at the text. Each pass will then contain

a set of trial routines together with the evaluation routines required for the choice from among the results of the trials. The output of each pass will serve to adjust the parameters for the trial and evaluation routines of the next following pass. The program will continue to operate until a given pass no longer yields trial results acceptable to the evaluation routines and by analogy with the idealized process of linguistic analysis, that pass can then be considered the final pass of the entire run.

Given appropriately chosen criteria for the trial and evaluation routines, it is reasonable to assume that, if the input text represents a natural language, the program will be able to apply at least one pass to it and produce a classification acceptable to the evaluation routines. It is equally reasonable to assume that if the text does not represent a natural language, all the trial routines of the very first pass will fall by the criteria contained in the evaluation routines.

The program can thus be expected to have the capability for deciding whether or not a given input text represents a natural language.

2.3 *Design of Trial Routines*

It was stated in the Introduction (under l.2) that distributional analysis can be based on an examination of diagnostic contexts, and (under 1.3) that the linguistic techniques used in this analysis, primarily those of substitution and dropping, are essentially computable. It was further stated under 2.2 above that the trial routines of the automatic linguistic analysis program would be developed on the basis of such linguistic procedures.

In the present section, the techniques of substitution and dropping in linguistic analysis will first be discussed, and then their adaption to the trial routines of the program will be outlined.

(i) *The substitution technique.* The linguistic technique of substitution consists in investigating what elements are substitutable for each other in the same diagnostic context. Substitutability is for purposes of the present study defined more narrowly than is necessary for ordinary linguistic analysis. It is here specified as a disjunctive relation: a set of elements are substitutable for each other if any one of them, but not two or more together and adjacently, may occur under a given set of conditions. A context is considered diagnostic if under the conditions of that context only sane, but not all, of the elements under consideration are substitutable for each other. A diagnostic context thus allows the division of the given universe of elements into two distribution classes: the class of elements which are substitutable for each other in that context, and the class of elements which are not.

Let us apply the substitution technique to the maximally simple linguistic universe of the 26 letters of the English Alphabet in order to illustrate the difference between a context that is diagnostic, and one that is not:

If we use as our test frame a context specified by a period on the left and *any* letter or a space on the right, it will be found that any one of the 26 letters of the English alphabet may in some normal English

text occupy that test frame. A context so specified is thus not a diagnostic context for the universe of letters - it is a nondistinctive context. If on the other hand the context is specified by a period on the left and the lower-case letter h on the right, it will be found - barring a very highly specialized text with borrowings from a language other than English - that only the letters a, b, c, d, e, f, g, k, o, p, r, s, t, u, w, z occur in that test frame, and that the letters h, i, j, l, m, n, q, v, x, y, do not. The context "._h" therefore is a diagnostic context.

Even for a small universe of elements, such as that of English letters, more than one diagnostic context can be specified, and each separate context will yield its own alternative distributional classification. For a larger universe of elements, such as that of printed words as delimited by spaces and/or punctuation marks, it is reasonable to assume that only a few nondistinctive contexts will be found and that the majority of contexts will be diagnostic.

(ii) *The dropping technique.* This technique is in ordinary linguistics used for the analysis of spoken language with the aid of an informant. It consists in testing whether the omission of one portion of a given sequence leaves a viable residue, that is, one which is acceptable as an utterance in the language under investigation. The dropping technique is used to ascertain a linguistic relation of dependence within the sequence, which corresponds to a logical relation of presupposition: A is dependent on B, whenever B is the necessary condition for the occurrence of A. In the dropping test, the omission of A will leave a viable residue, the omission of B will not. In the English sequence "We shall make an attempt to describe this", the omission of the auxiliary "shall" leaves a viable residue: "We make an attempt to describe this". On the other hand, the omission of "make" does not: "We shall an attempt to describe this" is not an acceptable English utterance. The results of this dropping test permit us to assume that a dependence of "shall" on "make" for its occurrence is more likely than the converse. In order to arrive at a definitive statement of such a dependence relation, many additional tests are required.

The dropping test is related to the analytic objective of classification in that all elements that exhibit the same dependence relation (that is, for which the dropping test yields the same results) can be considered members of the same class, as defined by this relation.

(iii) *Computability of substitution and dropping.* While it appears (cf., Section 1.3.) that both the linguistic techniques described above are computable, there are some differences in the design features of trial routines based on dropping as compared to those based on substitution which are of crucial significance in view of the logical requirements (stated under 1.1) that the initial program contain no dictionary or grammar code.

Substitution routines will have to be able to generate for each pass of the program, the appropriate set of test frames by which to produce the alternative classificatory listings for that pass. The test frames for the first pass will have to be generated without access to a dictionary of grammar-coded elements - this is possible; for the subsequent passes,

the program can draw upon the listings of each preceding pass to generate the frames. Since the purpose of the program is to produce increasingly more refined classifications, the test frames will have to be increasingly more specific with each pass, in order to produce more and more detailed listings as the program continues to operate. To insure that the classifications are not due to chance, a large input text is required.

The dropping test can be simulated by an essentially cumbersome series of comparisons (which, as will be elaborated further below, can be simplified, once certain conditions are met). For each comparison, the trial routines will have to identify a pair of unequally long strings of elements, such that the longer of the two strings contains all the elements of the shorter one, in the same order, plus one additional element. The one element present in the longer string and absent in the shorter one can be said to be "droppable" from the longer one, if both strings have been found to recur in the text sufficiently frequently to allow the assumption that the difference in their length is not due to chance. For every identified longer string, the droppability of each element will have to be tested by finding an appropriate shorter string. Those elements for which shorter strings are not found in the input text can then be assumed not to be droppable, provided enough recurrences have been found so that the absence of a particular shorter string is not attributable to chance. In order to allow for the necessary recurrence, a large input text would again be required, very probably several times larger than that required for a program based on substitution routines.

For a dropping routine to operate within the logically prescribed restrictions - that is, without an initial dictionary or grammar code - each trial would have to compare every string of n - 1 elements. A series of passes could be envisioned, with the value of n increasing for each pass from a minimum of 2 for the first pass to the maximum found in the text, for the last pass. Assuming a punctuated text, this maximum value of n could be made very much smaller than the total number of elements in the entire text by requiring that no string be allowed to contain a period or other final punctuation mark - this would restrict the permissible length of a string to the span of between two such punctuation marks, or between one such mark on one side, and the beginning or end of the entire text on the other. That is, the maximum permissible length of a string would be that of the longest sentence in the text.

The reverse procedure, in which the program first ascertains the maximum value for n and then decreases it with each pass, is equally thinkable.

Either procedure for applying the dropping test, to be carried to its logical conclusion, seems to require a program of quite unmanageable proportions. Neither procedure allows one to visualize, in any simple way, the kind of progression from broader to more detailed classifications that an automatic linguistic analysis program should produce in the course of its operation.

For these two reasons, a computer simulation of the linguistic dropping technique does not recommend itself for the initial trial routines of the program. Once, however, the program has created a grammar-coded dictionary by other means, a manageable application of dropping routines can be envisioned. This will be discussed under (iii) in Section £.4. below.

For the initial trial routines of the program an adaptation of the substitution technique is clearly better suited.

(iv) *Adaptation of the substitution technique to the trial routines* If - as was intimated in several places above (cf., Section 2.2) - printed English is used as input to the program, printed English words as delimited by spaces and/or punctuation marks can be chosen to constitute the universe of elements to be classified. The purpose of the program then becomes one of assigning all the words found to occur in the text to their appropriate classes and subclasses, and to indicate this assignment by producing as output a dictionary listing of words, accompanied by a grammar code indicating their class and subclass membership. As was stated in Sections 2.1 and 2.2 above, the program will attempt to achieve this by a succession of passes at the text, each pass consisting of an appropriate combination of trial and evaluation routines.

Under (iii) above, two major reasons were given for selecting, of the two linguistic techniques that were examined, substitution rather than dropping as the model on which to base the design of the trial routines; firstly, the test frames required by a substitution routine could be generated by a program of manageable proportions without violating the logical restriction of not containing an initial dictionary or grammar code; secondly, test frames of increasing specificity were envisioned for consecutive passes, allowing the anticipation of increasingly more detailed classifications with each subsequent pass.

Only the test frames of the very first pass have to be specified without recourse to a dictionary or grammar code, since each subsequent pass can draw upon the output of previous passes for the grammar-coded dictionary produced up to that point.

The first pass in turn can draw upon the typographical characteristics of the input text, specifically upon periods and other sentence-final punctuations, to produce the initial test frames. The following sequence is envisioned.

Sentence boundaries in the text can be ascertained by reading the beginning of the text, final punctuations, and the end of the text. Some provisions will have to be made to eliminate such punctuation noises as periods after abbreviations; this might initially be accomplished by key-punching instructions or pre-editing. Once this is done, sentences provide the "natural" initial test frames for the first pass.

With the sentence as a frame, all word recurrences in the text can be grouped into three "natural" classes: occurrence as the very first word of a sentence (called "initial", abbreviated I), occurrence as the very last word of a sentence (called "final", F), and occurrence in any non-first and non-last position in the sentence (called "medial", M). Occurrence in a one-word sentence can be classed as both initial and final. For each separate word of the text, the number of I, M, and F occurrences can be tabulated, and the total range of distribution of a word can be ascertained as one of the following: I only, IM, IMF, IF, M only, MF, F only. This count will be reliable provided - as was intimated under (iii) above - the textual sample is large enough to insure that this pattern of occurrence and non-occurrence is not due to chance. The I only, IM, IMF, IF, M only, MF, and F only ranges of distribution may then serve as the defining criteria for an Initial set of

classes, to which the program can assign all the recurrent words of the text.

A dictionary of all the recurrent words of the text with a grammar code consisting of appropriate class membership tags can now become part of the program and be used in the further processing of the text.

A significant linguistic observation may be introduced to serve as a criterion for the further utilization of the classes obtained in the trials of this (and any further) pass. This observation relates to the membership size of classes of linguistic elements of the morphemic (or content-bearing) type (as exemplified in this connection by words as opposed to letters): classes of such elements tend to have either extremely large membership (as, for instance, the class of verbs or nouns in traditional grammar), or severely restricted membership (as, for instance, the class of prepositions in traditional grammar). The former type of classes can be called unrestricted as to membership, the latter restricted.

In line with the above, it may be assumed (and the assumption has been borne out by preliminary manual tests) that certain of the initial seven classes will be unrestricted as to membership, others will be restricted. It then becomes reasonable to suggest that the classes of unrestricted membership lend themselves more readily to further subclassification than those of restricted membership, and conversely, that the restricted membership of the latter permits more readily the automatic selection of a finite number of individual words for the generation of test frames in subsequent passes.

In the next pass, therefore, the classes of unrestricted membership will constitute the universe of elements to be further classified, and one or more of the classes of restricted membership can be selected by the evaluation routines to serve as parameters for the more specific test frames that are required, that is, as frame-forming classes.

The distributional defining criteria of a given frame-forming class can be related to the frame-forming properties of the sentence to generate an appropriate set of frames. Assuming, for instance, that the MF class has been selected as frame-forming, test frames can be generated by relating the class as a whole or any of its members to the two fixed points of the sentence: its beginning and its end. Thus, one set of frames will delimit the slot between the beginning of the sentence and any, or a particular, MF word in the second position; another set of frames will delimit the slot between any, or a particular, MF word in the one-before-last position and the end of the sentence. This one frame-forming class alone will thus produce two major subclassifications of the universe of elements: those that do, and those that do not, occur in the two general frames as constituted by "sentence beginning _____ any MF word", and "any MF word _____ sentence end", respectively; it will also produce a further subclass of words that occur in both frames. Using only one frame for each trial, as many minor subclasses of the first two major subclasses can be produced as there are members of the MF class to serve in the more specific frames constituted by "sentence beginning _____ a particular MF word", and by a "particular MF word _____ sentence end". The number of classes can be multiplied by using the intersections of two or more frames to produce additional classes.

Each of these new classes can, if it is of unrestricted membership be used as a new universe of elements for further subdivision. It can, if it is of restricted membership, be used in the generation of additional test frames. Each of the original classes of restricted membership can be further subdivided by suitable test frames, and the resulting subclasses can in turn be used in the generation of still more test frames.

It can be seen that, as was suggested in 1.3 above, the program will have an algorithmic capability for producing an extremely large number of word classes and subclasses. In order to keep the classification in manageable bounds, as well as to insure that the classificatory output of the program will in some way be interesting linguistically, evaluation routines will have to be introduced after each classificatory step, resulting in the retention of only those classes and subclasses that are in keeping with the assumptions on which the evaluation routines are based.

It also becomes reasonable to envision that, once the program has produced a detailed classification and has included the appropriate tags in its dictionary, the dropping technique can be simulated more successfully than would have been possible in the initial passes of the program. In the later phases of the program, test strings need no longer be selected on the basis of their length alone (cf., under (iii) above), but may be specified on the basis of the previously established class memberships of the universe of elements, which can be expected to reduce drastically the scope of the required dropping routines.

2.4 *Evaluation Routines*

As was stated in 2.1 above, the purpose of the evaluation routines of the program is to provide the criteria for deciding whether the results of a particular trial have been successful or not. Since the purpose of the trial routines is to produce classes, we may stipulate as a condition of success that the classes resulting from a particular trial be linguistically relevant classes. This in turn requires the formulation of some computable differential criteria for the relevance of linguistic classes.

One element property, namely the condition relating to the membership size of the classes, has been suggested under (iv) above for inclusion in the trial routines themselves. Two further, more detailed properties are here definitely suggested for inclusion in the evaluation routines, and an additional one is considered worth investigating. It is furthermore not unreasonable to assume that in the course of the research hitherto unsuspected properties may suggest themselves for consideration in the design of the program.

The three properties alluded to above - two suggested definitely, and one suggested tentatively - are discussed in that order in the subsequent three sections.

(i) *A statistical property: the ratio of frequency of occurrence of members to size of membership of class.* It is assumed that the frequency of occurrence of a linguistic element is in an approximately inverse ratio to the size of the class to which it belongs. That is, the

frequency of occurrence in text of members of classes of restricted membership is assumed to be relatively high, that of members of classes of unrestricted membership relatively low.

Support for this assumption can be found in the observations that individual function words (such as prepositions or conjunctions), which may be expected to belong to classes of restricted membership, usually have high frequency of occurrence in texts, while on the other hand individual content words (such as verbs or nouns) have by comparison a much lower frequency. Some of these observations have been borne out by word counts.

An evaluation routine based on this property can be envisioned as follows:

The program will tabulate, along with class and subclass membership, the frequency of occurrence of each word. At the end of each test of trials, a comparison of the frequency of occurrence of the members on the one hand to the size of the membership on the other can be made for each alternative class. Only classes of restricted membership with high-frequency members, and classes of unrestricted membership with low-frequency members, will be candidates for acceptance, possibly subject to further evaluation in terms of the properties to be discussed below. Conversely, classes of restricted membership with low-frequency members and classes of unrestricted membership with high-frequency members will be candidates for rejection.

(11) *A distributional property: combinatory affinity of classes.* One of the most consistent observations about natural language is that there tend to be distinct regularities in the sequential co-occurrence of the members of the various linguistic classes and subclasses. Thus - using traditional parts of speech to illustrate - it can be readily noted that in ordinary English texts nouns are frequently accompanied by adjectives, the latter are frequently accompanied by adverbs, verbs are frequently accompanied by verbal auxiliaries, and so on for a number of other typical combinations. The frequent occurrence in text of a member of one particular class of linguistic elements adjacent to a member of another particular class can be expressed as a distributional property of the classes in question, here called the combinatory affinity of classes.

It is thus not unreasonable to assume, in view of the consistent observation of such distributional regularities in the natural languages of the world, that a machine-produced classification which allows the determination of many combinatory affinities of classes is likely to be a more successful trial result than one which does not.

An evaluation routine based on combinatory affinity would not be applied individually to the classes and subclasses resulting from particular trial routines, but to the comparison of at least two entire alternative classifications at a time, each produced by its own set of trial routines within the same pass. The routine can be looked upon as a two-phase procedure; the first phase will consist of the repeated application of the same subroutine to examine, for each alternative classification, the combinatory affinity of classes resulting from that classification, the second phase will be a subroutine comparing the results of these examinations, leading to a decision as to the candidacy

for acceptance or rejection of the classificatory alternatives.

The examination subroutine of the combinatory affinity routine can be based on the expectation that, whenever two or more classifications have resulted from a set of trials, the program will have produced (either logically or in a literal sense) a corresponding number of dictionaries, each with a grammar code derived from a different one of the alternative classifications. Each consecutive application of the examination subroutine can thus call the dictionary derived from the classification to be examined, and use it to ascertain relevant combinatory affinities.

An examination subroutine might be composed of the following operations, some of which could be programmed to run simultaneously:

(1) Call current dictionary.
(2) Bring in input text one sentence at a time, look up words in dictionary, and assign current grammar code to each word.
(3) Examine each sentence to detect whether a particular combination of adjacent grammar codes occurs more than once in a sentence. (This procedure is suggested in order to avoid having to test for all conceivable combinations.) Store all such recurrent combinations in a "dictionary of current combinations" (DCC).
(4) Use DCC to reexamine text sentences to detect and tabulate additional occurrences of current combinations (CC) not detected under (3) above.
(5) Reprocess text sentences to examine left and right neighborhoods of each occurrence of each CC. If any CC is found to occur with the same neighbor(s) more than once in the same sentence, include CC with neighbors in DCC as expanded CC (ECC).
(6) Use updated DCC to reexamine text sentences to detect and tabulate additional occurrences of ECC not detected under (5) above.
(7) Tabulate total number of running words of text contained in all recorded occurrences of CC and ECC. Compute ratio of this number to total number of words in text, yielding the value for the combinatory affinity of classes for the classification on which current grammar code was based.
(8) Call next alternative dictionary and repeat procedure, using new grammar code (i.e., return to (2) above).

The comparison subroutine will compare the values for combinatory affinity yielded by each iteration of the examination subroutine and use them to arrive at the required decision as to the candidacy for acceptance or rejection of the classifications from which these values have resulted

The CC's and ECC's tabulated from the successful classification (or from all alternative classifications) can be retained for further examination. They can be used by the routines suggested under (iii) below, or printed out for inspection.

(iii) *A relational property: dependence of classes.* A relation of dependence can be ascertained by the dropping technique, as discussed

under (ii) and (iii) in Section 2.3 above. Dropping routines were rejected for use as initial trial routines because their application to each individual word occurrence of the text without the aid of a grammar-coded dictionary would render the program unmanageable.

Once the program has produced a dictionary and grammar-code, however, the conditions for the application of a dropping routine are much more favorable. It now becomes possible to use such routines not to test for the dependence relations of individual words, but for dependences of classes - a much more restricted and manageable objective.

Given a set of classes as represented by the grammar code of a current dictionary, there are several conceivable ways of introducing into the program dropping routines for the determination of the dependences of classes. Two will be mentioned.

It is then for instance possible to apply the method of strings of lengths n and n - 1, described under (iii) in Section 2.3 above, not to strings of words, but to strings of grammar code symbols. The recurrence of these can be expected to be much higher than that of individual words, the variety of possible strings consequently much less, and the proportions of the program might under these conditions remain within manageable bounds.

Another application of the dropping routine might be to operate upon the CC's and ECC's produced by the combinatory affinity routine described under (ii) above. The longest ECC in the DCC could be taken as the string with the highest value n for length, and CC's and ECC's of n - 1 length could be compared to it, and so on for all values of n down to the minimum of 2.

Either modification of the dropping technique can conceivably serve as either a trial routine or an evaluation routine. If used as a trial routine, its purpose would be to yield additional classes for evaluation in terms of the statistical and distributional properties. If, on the other hand, the dropping routines were to be used as evaluation routines, they could be applied to alternative dictionaries, and the relative proportion of dependence relations obtained from each alternative might serve as an evaluation criterion. It might be assumed that a classification that allows for the determination of a greater number of dependence relations within the text is preferable to one that does not.