B. V. SUKHOTIN

DECIPHERING METHODS AS A MEANS OF LINGUISTIC RESEARCH

Methods of linguistic deciphering may be regarded as a set of procedures aimed at the recognition of linguistic objects in a text whose language is not known to the investigator.

They combine many advantages of the formal approach to language. Assuming that each deciphering procedure may serve as a definition of the respective linguistic object we may view the set of such procedures as a certain linguistic theory which has the following properties:

1) A great degree of generalization, because its definitions should be valid both for the known and unknown languages.

2) Formality, because naturally enough, the deciphering procedures should be presented in the shape of algorithms.

3) Constructivity, i.e. the possibility of identifying a certain linguistic object with the help of a deciphering procedure within a reasonable (finite) time interval.

To identify a linguistic object a deciphering algorithm makes use of a set of its features. Those features are sufficient for a non-constructive definition of the object under investigation and to a very great extent determine the kind of the recognition algorithm.

It seems obvious that a linguistic object cannot be defined by means of binary features alone. A definition based on binary features will be either too specific and valid only for a chosen language, or too abstract and insufficient for identifying the object in a given text.

The following scheme seems to be better founded:

(1) Binary features are used to determine the general type of certain linguistic objects. The objects belonging to that type form the set of admissible solutions of a deciphering problem.

(2) An objective function which estimates the quality of each solution is introduced on the set of admissible solutions. The values of the objective function are calculated with the help of the investigated text. They reflect the individuality of the given language.

A maximum or a minimum of the objective function should correspond to the linguistic object which is to be defined.

(3) It follows that a recognition procedure should be an optimization algorithm which finds "the best" admissible solution – from the point of view of the objective function.

Thus, the set of admissible solutions, the objective function and the optimization algorithm constitute the definition of a linguistic object which may be used for the purposes of deciphering. A definition of this kind will be further referred to as a deciphering algorithm, or simply, an algorithm.

There is a natural hierarchy of deciphering algorithms. An algorithm B is senior to an algorithm A if the former makes use of the information provided by the latter. If A and B work alternatively, each time improving the output, then the seniority is determined by the first iteration. Consequently, taking into account the fact that the set of essentially different deciphering algorithms should be finite, it appears that there must exist "zero" algorithms which use no information produced by any other deciphering algorithm.

Zero algorithms should be different due to the fact that the physical substances of different languages may be different too. Thus the zero algorithm for the analysis of the written form of languages should be able to discriminate between a dark spot and a light one and to find the set of alphabetic symbols of the language. A similar algorithm adjusted to the analysis of audible speech should produce an alphabet of phonemes, exploiting its capacity to discern certain minimal differencies of sonation. The plurality of zero algorithms may be reduced by converting signals of different nature into a set of curves. As is well known such algorithms are the goal of pattern recognition theory.

Senior algorithms should be used for the analysis of grammar; the highest level corresponds to the problems of semantics and translation.

Many algorithms of different levels display great similarity and sometimes even identity, their only difference consisting in the linguistic material which serves as the input.

Thus the algorithms that classify letters according to their pronunciation may closely resemble the algorithms that classify morphemes according to their grammatical role; the algorithms that find the boundaries between sentences may be similar to those that find boundaries between words and so on.

The following types may be pointed out:

1) Algorithms of classification, which divide the set of investi-

gated objects into several subsets. For instance, the letters are classified into vowels and consonants, morphemes – into auxiliary and root morphemes, words – into parts of speech.

2) Algorithms of aggregation which form larger units from smaller ones. For instance, they put together letters into morphemes or syllables, morphemes into words and words into sentences.

3) Algorithms of connection, which find out some relation of partial ordering. A typical example is provided by different algorithms of discovering the dependency graph of a sentence.

4) Algorithms of mapping the elements of an unknown language into the elements of a known one. Algorithms of this type should solve problems of translation and discover the kinship of languages.

The most simple classification algorithm is that which classify letters into vowels and consonants.[1]

In effect this algorithm is valid for any string composed of objects of two different classes and characterized by the fact that objects of the same class co-occur rather rarely whereas objects of different classes co-occur relatively more often.

The set of admissible solutions in this case is a set of divisions of the list of objects into two subsets; the quality Q of a division $D = \{ K_1, K_2 \}$ is evaluated by the following formula:

$$Q = \sum_i \sum_j f(e_i, e_j).$$

Here $f(e_i, e_j)$ denotes the frequency of co-occurrence of objects e_i, e_j from classes K_1 and K_2 respectively. The maximum of Q corresponds to the optimal classification. An appropriate optimization procedure reduces the amount of divisions that should be evaluated to a reasonable number. This algorithm has been thoroughly tested in a number of computer experiments and in every case yielded almost entirely correct results.

The most important algorithm of aggregation is the morpheme identification algorithm. Apart from identifying morphemes this al-

---

[1] See Б. В. Сухотин, Проблемы структурной лингвистики, 1962. Later on appeared the works of V. V. Ševoroškin, R. Caianiello and A. Tretiakoff.

[2] Since the pioneering work of Z. Harris, *From phoneme to morpheme*, in «Language», XXXI (1955), pp. 190-222, attempts for solving this problem were made by N. D. Andreyev, A. Y. Šaikevič. The author's first paper on this problem appeared in 1963 (Проблемы структурной лингвистики).

gorithm discovers an IC graph which shows the way in which morphemes are combined into words. The algorithm is valid even for texts which have no special devices for marking the boundaries between words and may be used in order to find those boundaries.

An admissible solution in this case is a series of divisions $D_1, \ldots, D_n$ of the text, each class of D_i being included in a certain class of D_{i+1}. Morphemes form the classes of the smallest division D_1, the classes of the biggest division D_n corresponding to words.

The objective function is set up by ascribing to each class K_i of D_j a certain number $p(K_{ij})$ which shows the strength of mutual prediction of the components of K_{ij} and by adding up all $p(K_{ij})$:

$$Q = \sum_i \sum_j p(K_{ij}).$$

$p(K_{ij})$ is the product of $St_{in}(K_{ij})$ (internal stability) and $St_{ex}(K_{ij})$ (external stability).

$St_{in}(K_{ij})$ is the mean conditional probability

$$\frac{1}{2L} \sum_x \left(\frac{f(l_x)}{f(K_{ij})} + \frac{f(r_x)}{f(K_{ij})} \right),$$

where the string K_{ij} of the length L is divided into the left part l_x and the right part r_x ($l_x r_x = K_{ij}$) in all possible ways; $f(l_x)$, $f(r_x)$, $f(K_{ij})$ denoting the frequencies of the respective strings.

$St_{ex}(K_{ij})$ is equal to zero if there is a string K such that $K_{ij} \subset K$ and $f(K_{ij}) = f(K)$, and equal to 1 in other cases.

This algorithm was tested in a number of manual experiments. A large computer experiment is going on at the present time.

It is only statistically that the immediate neighbourhood of the words in a text reflects their semantic connections. The understanding of the text is greatly facilitated by the discovery of those connections, a procedure carried out by the connection algorithms.

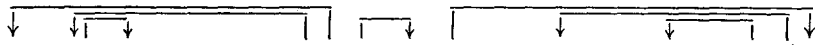
Representative of these is the algorithm of finding the dependency graph of a sentence. For this purpose the words of the language should be classified into parts of speech so that we may consider a word v to be included in a class K_v. The conditional probability $p(K_v/K_w)$ of occurrence of K_v near K_w is calculated with the help of the text.

The set of admissible solutions in the set of all possible dependency trees which may be ascribed to a given sentence. The conditional probabilities provide the weights for the arcs of the tree. The quality of

a tree is the sum (or the mean) of the weights of all arcs. The optimal tree presumably has the maximum quality. Some algorithms of this type has recently been tested in computer experiments and yielded good results.

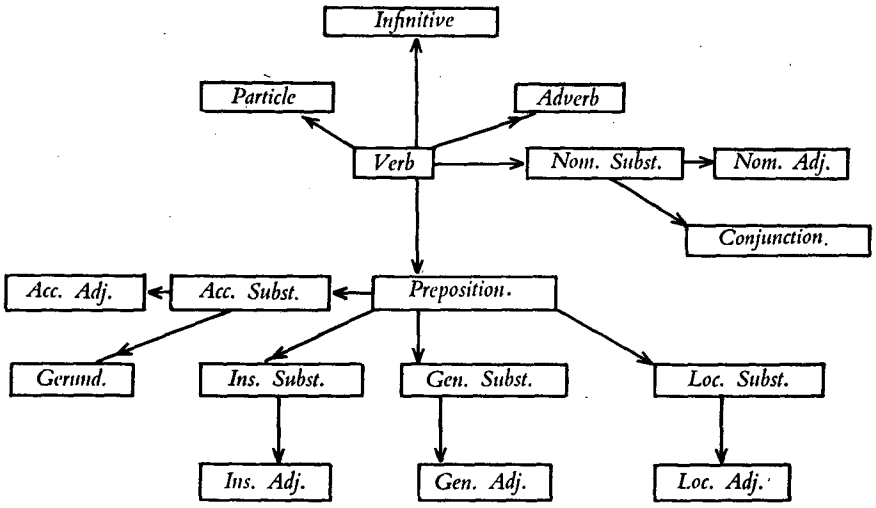
One such experiment which employed 19 syntactic classes was carried out for a Russian text of about 10000 words. It has established about 80 % correct connections. Here are some typical examples:

Однажды играли в карты у конногвардейца Нарумова
Adv. Verb. Prep. Acc. Sub. Prep. Gen. Sub. Gen. Sub.



Прочие в рассеянности сидели перед пустыми своими приборами
Nom. S. Pr. Loc. Sub. Verb. Pr. Ins. Ad. Ins. Ad. Ins. Sub.

Applying the optimization algorithm to the alphabet of syntactic classes, we get the " representating graph " which shows the typical connections, recognized by the given algorithm. For the algorithm mentioned above such representating graph looks as follows:



Algorithms of this kind may be used for the purposes of machine translation, in which case a greater amount of the input information is needed.

A typical example of an algorithm which obtains mapping $M = \{ E_i \rightarrow E'_i \}$ (E_i being some elements of the unknown language, E'_i – the respective elements of the known one) is furnished by the algorithm which discovers the pronunciation of letters.

It is based on the hypothesis that letters of two different languages which have similar pronunciation possess similar combinatory power in their respective languages as well.

The combinatory power of letter l_i may be described by the vector of conditional probabilities $\{ p(l_i/l_j) \}$ which characterizes the occurrences of l_i in the neighbourhood of l_j. In the same way, vector $\{ p(l'_i/l'_j) \}$ characterizes the combinatory power of l'_i.

The quality of a mapping may be estimated by the formula:

$$Q(M) = \sum_i d(l_i, l'_i).$$

Here d denotes the distance (e.g. Euclidean) between vectors $\{ p(l_i/l_j) \}$ and $\{ p(l'_i/l'_j) \}$. All pairs $l_i \rightarrow l'_i$ $l_j \rightarrow l'_j$ belong to mapping M, so that d may be calculated by the formula:

$$d(l_i, l'_i) = \sqrt{\sum_j (p(l_i/l_j) - p(l'_i/l'_j))^2}$$

The minimum of Q corresponds to the optimal mapping. Some algorithms of this type have been tested with good results. It is obvious that a similar algorithm will be able to compile a bilingual dictionary with the entries in the unknown language, although the latter problem is, naturally, far more difficult.