[From: *Datamation*, 13(4), 1967]

MACHINE TRANSLATION – FACT OR FANCY?

the reasonable middle ground

by PAUL L GARVIN

A few years ago, a number of news stories appeared here and there announcing "fundamental breakthroughs" in the development of machine translation from Russian into English and declaring that workable machine translation systems, if not already a reality, were just around the corner. Since then, one or two government agencies have begun to operate automatic translation facilities with which some users were partly satisfied, some not at all, and none completely. More recently, a committee constituted by the National Academy of Sciences, National Research Council—the Automatic Language Processing Advisory Committee (ALPAC)—conducted a two-year study of the field of machine translation and came to the conclusion that "without recourse to human translation or editing . . none is in immediate prospect."[1]

Where, then, do we stand in machine translation? Were the claims justified that were made in the earlier days, or is ALPAC correct in concluding that there is no prospect for its achievement in the foreseeable future?
In my opinion, both are wrong.

To substantiate my view, let me give a brief survey of the state of machine translation. First, let me make clear that the field of machine translation is (with one glaring exception—the photoscopic disc)[2] not primarily concerned with the design of a special translation machine, but with the design of translation programs to be run on large general-purpose computers. Let me add that the major effort so far in this country has been directed toward the machine translation of Russian into English, although some experimental work has also been done on other languages (such as Chinese and German).

Two extreme approaches have been taken to the field, and one which I consider a reasonable middle ground.

I call the two extremes the "brute-force" approach and the "perfectionist" approach. Let me discuss these first, since they are the ones represented in the earlier newspaper claims and in the recent ALPAC opinion.

The "brute-force" approach is based on the assumption that, given a sufficiently large memory, machine translation can be accomplished without a complex algorithm—either with a very large dictionary containing not only words but also phrases, or with a large dictionary and an equally large table of grammar rules taken from conventional Russian grammar. The dictionary approach was implemented on special hardware, the dictionary-plus-conventional-grammar approach on a general-purpose computer. Both versions of the "brute-force" approach have yielded translations on a fairly large scale, but of questionable quality. The trouble is that both systems are fundamentally unimprovable, since they allow only mechanical extensions of the tables which create as many or more new errors as they rectify. The negative opinion of ALPAC regarding the achievements of machine transla-

[1] Language and Machines, Computers in Translation and Linguistics. A Report by the Automatic Language Processing Advisory Committee, National Academy of Sciences, National Research Council. Publication 1416. National Academy of Sciences, National Research Council. Washington, D. C., 1966. p. 19.

[2]Neil Macdonald, The Photoscopic Language Translator, Computers and Automation, Aug. 1960, pp. 6-8.

tion is based largely on a study of these systems, and is of course justified to the extent to which it concerns them.

The "perfectionist" approach represents the other extreme. It is based on the assumption that without a complete theoretical knowledge of the source and target languages (based on a theoretical knowledge of language in general), as well as a perfect understanding of the process of translation, both preferably in the form of mathematical models, the task can not even be begun. Consequently, the "perfectionists" have devoted most of their energies to theoretical studies of language, sometimes using computing equipment in the process, and have deferred the development of actual translation systems into the indefinite future. ALPAC'S assessment of the future of the field reflects this view.

Clearly, neither extreme will lead to acceptable machine translation within the foreseeable future (or perhaps ever). But there is that reasonable middle ground—to which, in my opinion, ALPAC has not given sufficient attention. Not surprisingly, it is the position that I represent.

The approach which my associates and I are taking, the "Fulcrum" approach, is essentially an engineering solution to the problem. It avoids not only the naivete of the "'brute-force" approach (which by now has become evident to the professional world thanks to the poor quality of the results that it has produced), but it also avoids the lack of task-orientation of the "perfectionist" approach (which is much less evident to a professional world that stands in awe of theoretical research couched in dazzling quasi-mathematical symbolisms).

Our disagreements with the "perfectionists" deserve some further elaboration.

It is clear that the achievement of acceptable machine translation requires a very detailed and extensive knowledge of the languages concerned. Everybody agrees on this. Where the disagreement lies is in regard to the nature of this knowledge and the approach to be used in acquiring it. In my opinion, the knowledge needed for the design of machine-translation systems is not merely theoretical knowledge, but primarily empirical knowledge, and above all problem-solving engineering know-how. This knowledge cannot be acquired in the abstract, before the design of translation systems is begun. On the contrary, it is only in the process of developing a trans--lation algorithm that it becomes clear what type of knowledge of the language is required. And it is only in the process of experimenting with the algorithm that the correctness of our findings about language can be verified.

Let me now discuss some of the essential features of the "Fulcrum" approach. In the process, I want to point out some further similarities with, and differences from, other approaches.

First, some similarities.

All approaches agree that a machine translation system must contain two basic components: a machine dictionary and an algorithmic portion. All approaches further agree that the dictionary must contain not only the source-language (Russian) words and their target-language (English) equivalents, but also a set of codes by means of which the information contained in the dictionary can be used to activate the algorithmic portion. These codes will be both grammatical and semantic, since in effect the algorithmic portion must perform both automatic parsing and semantic ambiguity resolution on the input text, and therefore needs both types of information. Because of the great burden placed on it, the algorithmic portion is the essential part of the translation system and determines the design of the codes used in the machine dictionary. The system operates by first performing a dictionary lookup on the source text. As a result of it, the dictionary entries (together with their equivalents and codes) corresponding to all the words of the text are brought into the work space—one sentence, or a few sentences, at a time. There the algorithmic portion goes to work on the text and uses the codes to do the processing.

Up to here, most approaches agree. Where the disagreements come in is in regard to both the basic design and the detailed structure of the algorithmic portion of the translation program, and consequently the dictionary codes.

First, the matter of basic design.

Many researchers view the matter of a language-data-processing algorithm in a manner similar to the parsers used for the processing of computer languages and other artificial languages. That is, they consider that the processor and the grammatical and other information required for the processing should be kept separate. The information needed by the processor should be stored in a table in the form of rules, to be called by the processor when needed. The rules are stored in the table in no special order, since each time information is needed the whole table is called and the state in which the processing finds itself will determine which rule will be applied. I call this approach "tripartite," since it favors the use of programs containing three basic portions: a dictionary, a processing algorithm, and a separate table of grammatical (and, to the extent possible, semantic) rules. In theory, the tripartite approach has two great sets of advantages:

1. It separates the labor of the programmer who designs and maintains the processor from that of the linguist who designs and maintains the table of rules. The only thing they have to agree on is the format of the rules that the processor can accept. This minimizes the communication problem between linguist and programmer; since once these matters have been settled, the two portions of the program can be handled separately.

2. The same processor can be used with more than one table of rules. This means first of all that rules can be modified or changed without having to change the processor, provided of course that the format is maintained. This gives the linguist great freedom of experimentation with different types of rules. It also permits the use of the same processor for the parsing of more than one language, by simply substituting one table of rules for another.

This is indeed an impressive array of arguments in favor of the tripartite approach. Unfortunately, the theoretical advantages turn out to be illusory, whenever in practice the use of a realistically extensive grammar, and not just a few basic rules, is attempted. The grammar table then becomes so complex that it can no longer be handled by a simple algorithm. Then either the algorithm has to be adapted to the table, which reduces its generality, or a secondary algorithm has to be written which will show the processor the way around the table. In either case, the principle of the strict separation of grammar and algorithm (or more generally, of information and algorithm) has to be violated.

I might add that the tripartite approach has so far produced only limited experimental results, although, of course, they are considered promising by some of its proponents.

This is why we have chosen a different basic design, one in which the information, grammatical and otherwise, is written into the algorithm rather than being kept separate from it. This means that the over-all system consists of only two portions, a dictionary and an algorithm—hence, the term "bipartite" has been chosen for it.

The algorithm of a bipartite system is essentially not a "parser" of the type used in tripartite systems. It is instead a linguistic pattern recognition algorithm which, instead of matching portions of sentences against rules stored in a table, directs searches at the different portions of the sentence in order to identify its grammatical pattern. Thus, the essential characteristic of the algorithm is the sequencing of the searches and, in each search subroutine, only as much grammatical information is used as is appropriate to the particular search. The rules of the grammar are in fact applied by the algorithm in a definite order, and a given rule is not even called unless the previous searches have led to a point where its application becomes necessary. This means that the highly complex system of rules that makes up the real grammar of a language is distributed over a correspondingly complex algorithm which applies the rules in terms of the ordering that the structure of the language requires.

A bipartite system thus stands or falls by the manner in which the problem of the sequencing of the searches within the algorithm has been solved. This is the key problem in developing the detailed structure of the algorithm, and we have chosen the "Fulcrum" approach for its solution.

The "Fulcrum" approach is based on two fundamental principles: the concept of the fulcrum and the pass method.

The concept of the fulcrum implies the use of key elements within the sentence (fulcra) as starting points for he searches performed by the algorithm. This means hat the algorithm, in searching through a sentence, does not simply progress from word to word, but in fact skips" from fulcrum to fulcrum. It performs a little search sequence each time it has reached a fulcrum, and goes on to the next fulcrum when this particular search is completed.

The pass method means that not one but several passes are made at every sentence, each pass designed to identify a particular set of grammatical conditions pertinent to the recognition process. Consequently, each pass has its own set of fulcra and its own search sequences. The ass method reflects the orderly progression in which the determination of the structure of the sentence is made: first, the sentence components are identified individually, then the relations between components are established, and finally the structure of the sentence as a whole is established. To each of these intermediate parsing objectives there corresponds, roughly, a pass or series of passes in the algorithm. The correspondence is not exact, because there are many ambiguities and irregularities interfering with the recognition process, and the design of the "Fulcrum" algorithm reflects these added complexities.

Let me give a grossly oversimplified illustration of the operation of the "Fulcrum" algorithm:[3] imagine that the following sentence were Russian and not English.

"These various compounds of copper have been treated the technical literature on many occasions."

In the earlier passes of the algorithm, first the nominal phrase "these various compounds of copper" and the two prepositional phrases "in the technical literature" and on many occasions" are identified and labeled as to their potential functions within the sentence (the nomina1 phrase is a potential subject, the prepositional phrases are potential complements). In a later pass, the verbal phrase "have been treated" (which in Russian consists of a single word) is identified as the potential predicate. These identifications are made on the basis of the fulcrum principle. Thus, in the nominal phrase, the algorithm first identified its fulcrum, the head noun "compounds." and then directs its searches at the modifiers ("These various") and the nominal complement ("of copper"), the structure of which had been previously identified by the algorithm. Finally, in the same late pass in which the potential predicate is identified, the algorithm fits the different sentence components together to arrive at the structure of the sentence as a whole, again using the fulcrum principle. The algorithm "knows" that the fulcrum of a simple sentence is the predicate and therefore seeks out the predicate immediately. It then reads the grammar codes of the predicate to determine which are likely subjects and complements and, armed with this knowledge, can hunt up the remaining sentence components.

Semantic ambiguities are resolved by context searching wherever possible: the conditions in context are sought out which are likely to determine the choice of one rather than another of the equivalents of a given Russian word. Thus, in our sample sentence, the word represented by the English word "compound" really has two English equivalents: "compound" or "association." The algorithm will decide that "compound" is the correct equivalent, because of the complement "of copper" that follows in the immediate context.

Note that the grammatical and other information which the algorithm needs to carry out these decisions is carried in the codes that are contained in the dictionary and are made available to the algorithm by the dictionary lookup.

Obviously, not many Russian sentences are as simply structured, nor are they as similar to their English equivalents, as the one cited here. To allow the processing of all Russian sentences, simple or difficult, similar or dissimilar to English, the "Fulcrum" algorithm consists of a sophisticated interlocking system of passes and searches for fulcra. In addition, it has a capacity for generating English text "from scratch" for those sentence portions in which the differences between the two languages are so great that selection of equivalents for the Russian original is not enough to produce correct English text. Finally, we are building a heuristic capability into the "Fulcrum" algorithm, which will allow it to revise decisions made earlier in the program on the basis of information gathered later in the program. This will give it the capability of recognizing even some of the most involved Russian sentences.

An earlier experimental version of the "Fulcrum" algorithm, called "Fulcrum I," has been in existence for some time now (developed under the sponsorship of the Air Force, RADC, and of the National Science Foundation), and has served as a testbed for developing new concepts and techniques in machine translation. More recently, we have begun designing an advanced version of the "Fulcrum" algorithm, the "Fulcrum II" (under Air Force, RADC, sponsorship), which is ultimately intended to be the basis for a new production machine translation system, to replace the inadequate ones now in existence. The "Fulcrum II" will be characterized by a revised and updated sequence of passes and searches for fulcra, a heuristic capability, new techniques for producing English translated text, as well as efficiency-oriented rather than experiment-oriented programming. The plans and. major flowcharts for it are now in essence complete, though not all the details. Given the necessary funding, the "Fulcrum II" can be completed and running within four to six years, depending on the level of effort and availability of staff (which is hard to find).

[3] *For a* more detailed discussion, see *Adaptation of Advanced Fulcrum Techniques to MT Production System (Russian-English),* Final Report, Contract AF30(602)-3770 (Engineering Change "B"), Nov. 1, 1966, The Bunker-Ramo Corp. Canoga Park, Calif.