JFY-IV MACHINE TRANSLATION SYSTEM

LIU Zhuo FU Aiping LI Wei

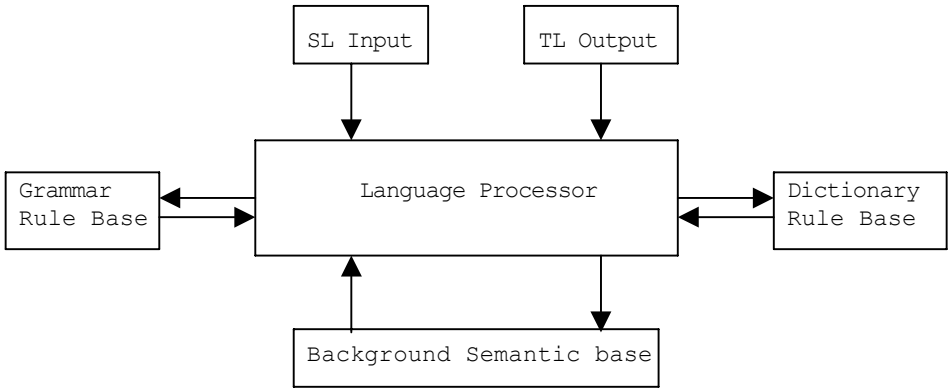
Institute of Linguistics
Chinese Academy of Social Sciences
5, Jianguomennei St., Beijing, P.R. China

1. History of Development of JFY-systems

JFY MT systems refer to the four systems named JFY-I, JFY-II, JFY-III and JFY-IV, which we began to develop as early as 13 years ago. JFY-I (1976-1978), which is only to translate titles, JFY-II (1978-1980) and JFY-III (1980-1982) are all experimental systems, aiming at exploring, both technically and theoretically, the ways and means for MT system development. But JFY-IV (1982-), now under active development, is market-oriented. Since 1982, it has experienced three major changes in its system design, and we got its final shape of overall design in 1987. Up to now, it has experimentally translated hundreds of various sample sentences during the 2-year system debugging and function inspecting, and it is proved that the theoretical basis of linguistics and the algorithm design techniques are sound and highly effective.

2. Characteristics of JFY-IV MT-system

2.1 Flow Chart for JFY-IV



The analysis and transfer work is accomplished through one-time bottom-up data-driven scanning from right to left, realized by bidirectional pattern matching; and the target sentence generation is also achieved by a right-left scanning of processing.

2.2 One-dictionary Strategy

There is designed in JFY-IV only one dictionary called SL-TL Usage Contrast Dictionary, in which rules are combined with lexical items, and word meanings and parameters are involved in the respective rules. The

dictionary of this type will help concretize universal rules and meanwhile abstract individual rules, and it will also help, during the process, extract word parameters dynamically.

At present, most MT systems seem to focus their attention on the establishment of linear abstract rule system for language analysis based on sort of meta language so as to satisfy the requirements of a regular analytical processing. They are, therefore, likely to overlook the experiential elements in language rules, i.e. the individual phenomena resulted either by language customs or by contrast differences between languages. This will of course heavily affect the translation quality. The ideal solution to this problem lies in that the abstractness of lexical rules be limited as far as possible on condition that grammar rules keep on their high generality. To achieve the goal, rule hierarchy should be used instead of linear rule table. For example, transitive VP rules can at least be classified as in the following 4 levels: (1) READ + BOOK; (2) READ + N (written material); (3) V + N (written material); (4) V + N. The difference in between lies not only in abstractness but also in the way to establish the rules. Rule level (1) is a concrete model structure, which serves as a starting point and basis for the other rules. Rule level (2) is directly related to a lexical item and, furthermore, to its certain meaning. Both (1) and (2) are dictionary rules. The level (3) and (4) rules, however, are independent of concrete words, and therefore should be included in Grammar Rule Base. Dictionary rules are also called model rules, and grammar rules called operational rules. It will bring about two advantages to establish language rules in a hierarchy of model rules up to operational rules. First, it will be possible to design the language processing algorithm as a gradual embodying process of information conversion. Second, it provides the possibility to combine rules with lexical items and their meanings without laying too heavy a burden on dictionary as quite some MT experts have worried about. And this idea of hierarchy is considered in JFY-IV as a guideline and principle for the establishment of both syntactic rules and semantic rules. In this way the precision of rules is not only guaranteed but selfprovable as well. To put it in further details, the combination of rules with words and their meanings will benefit in many ways:

(1) As a starting point and a necessary element in a model rule, the processed word implicitly introduces semantics into formal analysis, for, during its analysis, it is unnecessary to use its semantic parameters, which are only to be possibly used in the processing of other words.

(2) To lay rules onto words is the most effective way to guarantee the correctness and exactness of rules and to raise the search precision.

(3) The usages of each word are relatively limited. To enumerate all the model rules of a word seems no impossible even in the inductive method. Meanwhile, for each type of operational rules, one might deduce out inclusively the possible number of rules upon the permutation of the necessary rule elements, e.g. the permutation for AB is 2, for ABC, 6. The whole set of rules in a language can therefore be acquired by deleting the impossible permutations and by adding rules of the same permutation but with different content. This way, the set of operational rules of each kind, just as the set of model rules for each word, is itself a closed definite subset of the language rule base, which is well independent of,

i.e. not crossed with, other subsets. Such independence of rule subsets proves extraordinarily advantageous for the gradual establishment of the final rule base, and furthermore helps avoid much trouble and many mistakes caused by the crossing of rules, as shown in the previous MT systems.

(4) In the conventional MT systems, the various static parameters for each word are set up out of two requirements, one for the processing of itself, the other for the processing of other words. But in JFY-IV, thanks to the combination of rules with words, the first need is no longer necessary, for the word itself, being an indispensable element in its rule, is naturally a more concrete condition than its any parameters. Thus, one might only set up those parameters to meet the second requirement, which provides a possibility to change parameters dynamically according to the yes/no of the execution of a rule. It seems even more reasonable for a system to be able to dynamically change such codes as semantic parameters, which are varied as the word meaning is varied.

2.3 Background Semantic Base

Background semantic base is designed on two principles: distributive features and thesaurus-like hierarchy. Its establishment helps harmonize the relation between syntactic analysis and semantic analysis. Such a rule base proves convenient in two ways.

(1) In pattern matching, the involvement of the background semantic base in the condition search is in reality an extension of the model rule. During the search for the semantic condition required by the model rule, the matching will be counted as successful if either a semantic parameter of the searched word or one in the corresponding semantic chain in the base meets the requirement. Trouble is saved when you otherwise will have to be worried about how many bytes in the data structure are suitable and enough for lexical semantic parameters and about unnecessary repetition or frequent change of semantic parameters under lexical entries in the dictionary.

(2) It is very easy to set up and maintain such a hierarchy base because, once the principles are established, the rules can well be added to, or revised in, the base one by one. The number of the rules in the base does not affect the normal functioning of pattern matching, nor does even a null base.

2.4 Transformational Mechanism

The introduction of transformational mechanism helps to properly solve the problems concerning the intricate relation between individuality and generality inherent in a language.

We deem it extremely important for an efficient MT system to cleverly manage the relation between individuality and generality. There exist various manifestations for the relation, and therefore there should be different kinds of means. Our rule hierarchy strategy, which is realized with the help of the techniques like recursive call, function call and subset call applied in the establishment and execution of linguistic rules, is a very powerful and effective way to achieve the goal. The exploitation of the background semantic base is another remedy. In order to correctly

and reasonably deal with the relation of individuality and generality in linguistic pattern recognition, transformational mechanism is introduced in pattern analysis of the source language. Linguistic patterns are classified in two types: basic form and transform, respectively representing the individual aspect and general aspect for patterns. And it is stipulated that basic form is not computable, and therefore should be changed into transforms before it can be executed.

2.5 Support of Language JFY

The software we have used in developing our JFY-IV system is Language JFY, a highly flexible problem-oriented functional production language, defined and designed by ourselves on the basis of COBOL.

Language JFY is defined as of production because production is most suitable for pattern recognition; and its structure designed as functional just in order to achieve a modular system for rules. Language JFY, oriented to natural language processing, is not only of the similar characteristics to ordinary high-level computer languages, but enjoys its own particular functions to meet the needs in language processing: (1) Powerful transformation functions to realize such transformations as omission (ABC —>AB), substitution (ABC—>ADC), order change (ABC—>BAC), extension (ABC—>ABDC) and complex transformation (ABC—>BDAC); (2) Various flexible address search and word locating functions; (3) Easy and practical functions for leaping and backtracking processing; (4) Controllable go-to function; (5) Recursive function; (6) Functions for the let-pass, sieving and covering of language units according to linguistic rules; (7) Copy function for words and their related information; (8) Various perfect trace functions.

It must be emphasized that a perfect trace function is an indispensable tool in debugging for a practical MT system. Language JFY can automatically fulfill the following trace functions: (1) Overall trace: to trace the whole process of the execution of rules; (2) Word trace: to trace the process of working a certain word in the processed sentence; (3) Rule type trace: to trace the execution of a certain type of operational rules related to a certain word; (4) Conclusion trace: only to trace the process of those rules whose conditions are satisfied, i.e. the successful rules. The above four kinds of trace can be realized in two modes: foreground trace and fore-back-ground trace. The former simply traces the process of rules execution while the latter also traces the dynamical changes of all the parameters and variables concerned. Such a flexible trace function benefits us greatly in our debugging of the software, the test of all types of the rules and the whole experiment in general.

2.6 Simple and Easy-to-learn Interface for the User

A practical MT system should provide for the user the easy possibility to revise and expand the dictionary so as to gradually raise its translating ability and to suit the needs for the change and expansion of the vocabulary. Such a possibility, of course, only involves open words such as nouns, verbs and adjectives, as for function words, which are quite limited, they can only be dealt with by MT experts. There are two prerequisites for the realization of this possibility. First, the user doesn't need to know the specialized techniques for machine translation,

though a rough knowledge of the basic principles and process of machine translation should be necessary, which can be easily achieved by a week's training. Second, the system should have the mechanism to relieve the user of the disturbance of various parameters and variables. As our rules are organized in a system of hierarchy, so our lexical rules in the dictionary can be compiled very close to the way man compiles a usage dictionary. The rules in JFY-IV machine dictionary are in fact very similar in form to the descriptions for the usages of each word in some more recently published dictionaries, e.g. A Dictionary of English Collocations (by Suzhou University, 1988, Nanjing). Therefore, common language workers can well undertake the task of compiling the main part of the dictionary.

2.7 Compatibility for the Algorithm and Grammar Rule Base

JFY-IV translating system is designed to finally achieve the automatic translation from the main Indo-European languages such as English, French, German and Russian into Chinese, with the goal at present of the practical development from English to Chinese. After the completion of English Chinese part, if we go on to develop, say, a German Chinese system for JFY-IV, the main work will be the compilation of dictionary, and it on principle needs no great change, only with a few necessary complement and revision, for the already well established algorithm, i.e. the language processor, and the background semantic base and the grammar rule base.

3. Software Design for JFY-IV System

JFY-IV software involves three main parts: a working field, rule banks, and a language processor, which is the kernel.

The working field is a two dimensional data table in the computer memory set by the language processor. It is used to store the processed sentence and its related data. Before translation begins, the field is almost null only with the word string on the left side of the table. The data in the field are in constant change during the recursive calls in execution in such processes as dictionary looking-up, pattern matching, inferring, decision making, and knowledge acquiring, before finally are produced in the field all the parameters needed in the generation of a correct readable target sentence.

Except the background semantic base, the rule banks are where production rules are stored. Each rule, whether general or individual, is composed of condition and conclusion. Individual rules are initiated into operation through successful search for the related word by the language processor. General rules, on the other hand, can either be initiated in the search for words or by means of function call, with the rule type as the function name to be called, in the midst of another rule's execution.

Language processor is divided into two parts: arithmetic unit and control unit. The arithmetic unit executes such operations as pattern recognition and matching, inferring, etc. The control unit accomplishes process control, matching control, action control and trace control.

There have now been quite a number of pattern matching systems based on a production language. Compared with other production systems, our software

system has its own distinguishing features.

(1) The order between productions means little. Each production or group of productions stands independently of another production or group, creating a very favourable environment in which to expand, delete, revise and maintain the production rules.

(2) Although the operation of the system is mainly data-driven, however, the data-driven working mode can also be dynamically changed into the objective-driven one because in the working field are also stored by the system, besides various attribute parameters for the related words, some function parameters reflecting the designer's language analysis strategy.

(3) There is designed in our system a bidirectional inferring mechanism. Pattern matching therefore can be realized either in the direction from right to left or from left to right.

(4) Both condition and conclusion in a production can be complex. They can be made of a number of items; the content of each item can further be either single or complex. What is more, the content may also be a function call which helps greatly in dealing with the relation between individuality and generality of language rules. One can make a complex item whose internal relation is logical OR, AND or NOT. As the writing of a production is so flexible and varied, a complicated linguistic rule or a number of related rules might be expressed in the form of a single production in a concise and straightforward way. A man even without software training will have no much difficulty reading such production rules.

(5) In a production, condition is naturally not to be confused with conclusion. But in our system there is a mechanism which can help realize some conclusion in the midst of condition search or achieve some further condition search in the process of conclusion execution. Thanks to this know-how, a single production can in content be equal to a group of productions, and, still more significantly, generality can thus be more easily separated from individuality.

(6) The control unit in our language processor not simply controls a group of switches, but also embodies the linguistic design idea for the system.In this sense, the control unit is rightly compared to the nerve center for JFY-IV.

4. Experiment and Results in Executing JFY-IV

Since June 1978, we have undertaken four kinds of experiments to test and debug our system: (1) Software debugging: to debug logical errors in our language processor; (2) Functional test: to test function after function, problem after problem, in order to perfect every important rule module; (3) Overall test: to test the whole process of translation and all the interfaces on a corpus of sentences in which some are selected in succession from a technical text, some are typical sentences for different patterns or of other linguistic value, and still some with more or less extragramaticality to help test the robustness of the system; (4) Test for compatibility: to establish an experimental MT model from Esperanto to Chinese based on the existing JFY-IV English-Chinese system.