[From: Internat.conf. *Methodology & Techniques of Machine Translation*, British Computer Society, Cranfield, 13-15 February 1984]

A Micro-computer Program for Checking Translation at Sentence Level

Brian Farrington.

University of Aberdeen, Scotland.

The program which I propose to describe in this paper was produced by the Scottish Computer-Based French Learning Project, a project financed by the Scottish Education Department.

The program, which runs on a 48k Apple II micro with one disk-drive and requires no other soft- or hard-ware, is designed to simulate a translation tutorial. It has been in use for computer-assisted learning of French in Aberdeen University, and in several Scottish schools, for some months now, replacing an earlier set of programs which were also used for practice in translating. The manner in which it works is as follows. The student is given a text of about 300 words in English to be translated into French. The text is presented, sentence by sentence, at the top of the screen. The student types her translation in answer to a prompt, one word at a time. The system either accepts or rejects it. If it accepts the student's version, the system prints it on the screen, gradually building up a French sentence in a section of the screen beneath the English sentence. If the proposed translation is rejected the system gives a brief comment indicating what category of mistake has been, or is being, made. Thereupon the faulty section of the sentence is erased from the screen.

The level of proficiency of the students, and therefore the degree of linguistic complexity that the program is capable of handling, is that of first year undergraduates or final year school pupils. Some idea of the type of language that the program can handle may be had from the information that the source texts for translation at present being used are taken from first year University examinations, Scottish SYS examination papers or English A-level, and they consist of normal natural written language dealing with subjects of general interest, usually narrative and descriptive, but containing idiomatic expressions, dialogue and so forth. Simpler material can of course be used. The upper limit of complexity is decided by

the number of possible variants, as well as the length of the sentence.

The program works by referring to a set of structures stored in a database which I shall describe in a few minutes. For the moment the point that I want to underline is that there is no one fair copy, master translation or "perfect version". On the contrary, the program is able to accept a very large number of possible translations of any given sentence. This number, for a normal two or three line sentence, can in fact easily run to several tens of millions. Not all of these translations will, of course, be equally "good"; many of them will contain awkward or inelegant turns of phrase. In fact it is part of the function of the program when used for language learning, that it should contain as large a number of formally correct renderings as possible, even though only some of these will be fully acceptable, at least from a stylistic point of view. In other words, the translations will not contain any syntactic, morphological or lexical errors: these will be stopped by the program. On the other hand, the versions accepted may vary considerably regarding suitability to the context. This variety is not inherent to the program, but it is an essential feature of the way in which it is being used for learning.

Also, though the student types in one word at a time, for reasons which will become clear when I come to describe the design of the database, there is no question of this being in any way a word-for-word translation. To the contrary, translation equivalence is established at sentence level. To give an example of the degree of linguistic complexity that the program can handle, it has only slight difficulty in accommodating, as a translation of the English sentence:

"I do wish you wouldn't look at me like that"

a large number of versions ranging from "Je n'aime pas que tu me regardes comme ca" to "Ne me regardez pas de cette maniere, s'il vous plait" and could in theory cope with "Dis done, t'as fini de te rincer l'oeil?"

The program is designed as a simulation, not as a question and answer exercise, which would simply interrogate and, if necessary, correct the learner. It should take him/her through the material to be translated, exercising his/her knowledge not <u>about</u> but <u>of</u>

the language. If it was going to do this, it was clear that it would have simply to prompt the learner, and then be able to process any well-formed sentence which s/he typed in and which translated the original material adequately. This program, moreover, would have to work with a data-file that was regular enough in its design for a composing program to be written that could be used to construct it. In this way any reasonably motivated teacher should be able to write materials for it, without having to be involved more than superficially in computing. And finally, as if this was not enough, the program must be capable of running on a small micro such as an average school could reasonably be expected to possess.

There are obviously many problems here. I will mention two in particular. Firstly, any program that is to process the composition of sentences in the manner that has been described must work with a left to right grammar. For a teaching/learning program there is a particular difficulty here when dealing with disjunctive syntax. A student who types *le bicyclette, for example, has not made a mistake on typing le; the mistake does not appear until s/he has typed bicyclette. But the mistake is not in bicyclette, it is in the article that precedes it. Our left to right grammar must in short have a backstitching capability. This was solved by inviting the student to type in the translation word by word, and then by processing, not individual words, but a chunk or string. The chunks can be of varying length. In this way if the learner sets off up a syntactic blind alley, s/he can be brought back to the point where s/he deviated from normal French syntax, or grammar, to start again.

The second problem is that of deciding to what degree the program should be designed to predict the student's response. Pusack (1983) has made the point that in computer-assisted learning materials for language, it is the manner in which the system processes a student's incorrect answer that is crucial, and likely to be the most interesting feature of any CAL program. Pusack lists three degrees of complexity of incorrect-answer processing:

i. Pattern mark-up.
ii. Prediction of errors.
iii. Parsing.

The first two of these imply that the system knows the "right" answer, or answers, and that there is such a thing as one, or at least a limited number of "right" answers. Such an assumption, however, would go

counter to the general aims of the project as regards language learning. The third degree in Pusack's list, parsing, can be excluded straightaway. If it were possible to devise a parser capable of handling all the structures likely to be input by a student of the level of proficiency we are concerned with here, the program would certainly not run on the small micro we are obliged to use. In any case it would not be a sufficient answer to our problem, since it is not simply a matter of producing a stretch of acceptable French, but a stretch of acceptable French which is a translation equivalent of the source sentence.

Pusack's three categories were drawn up in the perspective of a type of exercise in which there are more wrong answers than right ones. If the reverse is the case, or at least if there are as many right answers as there are wrong ones, it becomes more important to predict, not the incorrect responses, but the correct ones. The exercise becomes positively instead of negatively orientated, and its whole nature changes radically. Instead of the system prompting the student for suggestions which are immediately tested against a list of prepared "perfect" translations and a set of carefully anticipated "mistakes", each with its appropriate comment, it invites the student to experiment, to explore the frontiers of her/his knowledge of the target language for possible translation equivalents.

The new approach can be seen in the design of the data file. The essential feature here is the way in which the possible target sentences are broken up into units on four levels of analysis. It would have been convenient if it had been possible to take over a ready-made grammar, complete with phrase structure rules. Unhappily, though the scheme used certainly bears a strong resemblance to a scale and category, or systemic grammar, in particular in its use of rank-shifting, the elements of analysis used are not consistently isomorphic with any recognisable linguistic unit, and, with the exception of the smallest (WORD) and the largest (SENTENCE), they do not operate consistently on the same level of analysis. For this reason terms for the intermediate units are not taken from Linguistics, but, following a suggestion from my colleague, Dr Cram, of the Aberdeen University Department of Linguistics, from the construction industry.

To explain the terms used: a SENTENCE is composed of a number (max=10) of SYNTAXES. Each SYNTAX represents the structure, at sentence level, underlying one set of target, in this case French, sentences. A SYNTAX consists of a string of up to 15 BLOCKS, identified by their numbers. Each BLOCK consists of a set of strings of units, in our terminology, of LAYERS of BRICKS. In the same way as the SYNTAX represents the underlying structure of the sentence as a whole, each LAYER represents the structure of a major section of the sentence, or phrase. Similarly, each BRICK consists of a set of CONSTRUCTS representing the structure of word groups, each CONSTRUCT consisting, as in the case of the SYNTAXES and LAYERS, string of units, this time of WORDS. As in other grammars it is possible for a sentence such as for example, Stop! to consist of one SYNTAX containing one BLOCK composed of a single LAYER made up of one BRICK only and that consisting of only one WORD. Normally sentences will be much more complicated, either at BLOCK level or at BRICK level, and sometimes at both.

To consider the different levels in greater detail, each SYNTAX will represent the basic syntactic framework for one set of translation equivalents of the sentence as a whole. For reasons of clarity, a BLOCK may be considered, to start off with, as representing a section of the source sentence which will be translated and which will stand as a single unit in all the final, target, versions of the English sentence. For example in the following sentence the division into three BLOCKS is fairly simple:

We walked round the gallery together, in a fairly aimless fashion, only stopping to look at the paintings which seemed to us most striking.

BLOCK 1
We walked round the
gallery together

BLOCK 2
in a fairly aimless fashion

BLOCK 3
only stopping to look at the paintings
which seemed most striking

The SYNTAXES for this sentence might be:
1:1+2+3 (Block 1 + Block 2 + Block 3)

2: 2 + 1 + 3

3: 3 + 1 + 2

Each BLOCK in its turn consists of a set of (15) LAYERS, which are the equivalent at BLOCK level of the SYNTAXES we have just seen. Each LAYER then consists of a string of up to 15 BRICKS, again identified by numbers only.

The first BLOCK from the example above could contain the following LAYERS:

BLOCK 1

```
1: 1+2+3+4+5  (nous marchames
                    + autour
                         + de
                            + la galerie
                                + ensemble)

2: 1+5+2+3+4  (nous marchames
                   +ensemble
                       + autour
                       + de
                              + la galerie)

3: 6+4+5       (nous avons visite
                      +la galerie
                        + ensemble)

4: 5+6+4       (ensemble
                    + nous avons visite
                       + la galerie)
```

Each BRICK, finally, consists of a set (15) of CONSTRUCTS, which are the equivalent at BRICK level of the SYNTAXES and LAYERS of the other two levels. Each CONSTRUCT will consist of a string of WORDS, again identified by numbers which refer to places on a word-list. BRICK 1, from the example just cited, could contain for example:

BRICK 1

```
1: 1+2  (nous+marchames)
2: 1+3+4  (nous+avons+marche)
3: 1+1+5  (nous+nous+promenames)
4: 1+1+6+7 (nous+nous+sommes+promenes)  Etc.
```

The data file itself consists first of a list of all the words used in all the translations of the source sentence, followed by a list of predicted word-level mistakes. Apart from these two lists, and the text of the source sentence, the entire file consists exclusively of numbers. Obviously, this data file would be very difficult for us to read since it consists entirely of numbers. A set of programs have been written for composing the data file and also for printing its contents out in a readily comprehensible form. Reference to words by number, however, makes it possible to save a great deal of memory space and also to speed the search routines of the program, which I shall describe in a moment. Using word numbers also makes it possible to take advantage of the fact that the majority of the variant translations of a given sentence will share a common vocabulary, which also saves space. A sentence such as the one in the example above, containing 22 words in English and a similar number for each of most of the French translations that can be used, will use a word list of about 100 to 150 words for the total number of acceptable translations.

The program was written for the Project by Dr Brian Robertson of the Aberdeen University Computing Centre. It works, after formatting the screen and presenting the sentence to be translated, by prompting the student to type in a word. It then checks to see if the word typed is in the word-list. If the word is listed the program proceeds to search. It first checks the SYNTAXES to see which BLOCKS occur in first position, then searches the LAYERS of those BLOCKS to see which BRICKS occur in first position. Finally it checks to see if the word reference number figures in first position in any of the CONSTRUCTS of those BRICKS. If the word number is found it is printed on the screen and the system prompts for the next word. The procedure is repeated until the program reaches the end of the sentence.

It can be seen now how the program is able to process a very large number of different translations of the same sentence. To give an example, in the case of the sentence quoted a few moments ago, the program can accept 675 different French sentences. For a longer, more complex and semantically richer sentence, the number of possible acceptable sentences that can be handled is very large indeed. By combining the different CONSTRUCTS of the various BRICKS that have been put together in a given LAYER, it can in fact produce versions that the tutor who constructed the

data file has not thought of. So far this has not produced any nonsense sentences, and it does not seem likely that it might, so long as the data file is carefully put together. It should be remembered, however, that no claim is being made that all the versions accepted are equally acceptable. They are not, and the program teaches by inviting the learner to experiment and to use the system to discover a better way of doing the sentence than the first thought of, and most obvious one.

To help in this there are two commands: LIST and BACK. If the learner types LIST the system will display on the screen all those words that it will accept as a next step. The facility is deliberately not called HELP or CLUE, since there is no reason to suppose that what will be displayed will necessarily be better than those proposed by the learner. Some of the words LISTed may even lead into a syntactic blind alley, though none of them will actually be wrong in themselves. On the other hand, many of the possible translations will have been suggested by native speakers and will probably contain words and expressions, not to mention ways of structuring the sentence as a whole, which the learner may be able to recognise, but would not have been capable of finding unaided. In any case, at most points in the sentence LIST will usually display a number of suggestions that the learner will not have thought of. The onus is on the learner to decide which one to choose. It is by placing the learner before such choices that the program "teaches", rather than by inviting him/her to make mistakes for correction. The learner is encouraged to use his/her receptive competence to inform her active one, and to transfer from one skill to the other.

If the learner types BACK the system will erase from the screen whatever words were printed from the current BRICK. If BACK is typed again, the system erases everything in the current BLOCK. Pressing BACK three times erases the whole sentence. Using LIST and BACK frequently a student can use the program to discover new ways of putting a French sentence together, and can experiment with different tentative versions, even making what may turn out to be mistakes on purpose, in order to find out what the limits of acceptability are. The final version produced will not necessarily be "better" than what s/he would have produced unaided, though the absence of formal errors should all the same mean some improvement in conventional terms. The success, or failure, of the program does not hang on the quality of the final version produced, but on how much the student learnt while producing it. It may be for this reason that the program seems to work well with pairs or trios of students at a time, rather than with one alone.

To prepare a sentence for the program, a rough note is first made of all the principal target sentences. It will usually be found that most of these

make use of the same, or closely related noun phrases, verb phrases, adverbial expressions and so forth, though they may be linked together in different ways, to fit into the syntactic structure of the sentence as a whole. The process of listing all the likely variants is speeded up by having access to a hundred or so old examination scripts. Old examination scripts are also useful for indicating the most frequent errors, since, notwithstanding what has just been said, the program is in fact able to trap and comment on these, as I shall explain in a few moments. From this rough listing, which experience has shown to take anything from 30 minutes to two hours to prepare, one starts to analyse the target sentences into their constituent BLOCKS and BRICKS. Starting with the most obvious, simplest version, which is usually the one that follows most closely the syntax of the original English, and ending with those versions which have been suggested by native speakers, one gradually builds up a set of pathways through the material, putting the different BRICKS together so that each may be used by as many separate pathways as possible. This is not child's play and needs quite a lot of juggling to get right. But it is a technique that improves rapidly with practice. On average an easy sentence of, say, one line long, can be charted in half an hour. On the other hand, a long sentence containing subordinations, idiomatic expressions and complex noun or verb phrases can take several hours. The maximum number of BRICKS for any sentence is 50, and because of this limitation it is sometimes necessary to divide a sentence into two parts.

When the preparation is completed and the sentence has been "charted", to use the term we have adopted for the process, the file can be typed into the machine. For this a set of interactive composing programs have been written. The material is typed in, one BRICK at a time, in words, and the program translates these into the necessary number references and stores the information. When the BRICKS have been typed the program prompts for the BLOCK CONSTRUCTS and SYNTAX LAYERS. These are given using the number

references which reduces the amount of typing to a negligible amount. Typing a charted sentence into the system using the composing programs takes half an hour or so.

There remains one feature which has not been mentioned at all, and that is the way in which LITTRE responds to an incorrect reply to a prompt, in other words a wrong answer. There are two types of response here. The first is automatic requires no attention on the part of the tutor, and deals with unpredicted errors. It is able to distinguish automatically between word-level mistakes and faults of syntax.

If the learner types in a word which the system cannot recognise because it cannot be matched with any in the word-list, the message:

Sorry, I can't recognise that word, maybe it's misspelt.

appears on the screen. This routine will cope with misspelt words, mistakes of accent, non-existent words as well as entirely unexpected, correct, translations, though these should, if the sentence has been properly prepared, be few. It would not be feasible to have any form of pattern mark-up or even "fuzzy matching" since the learner's input has to be compared with up to 200 words. However there is a facility by which predicted word-level mistakes can be dealt with. There is a second word-list in the data file for anticipated "wrong words". Each of these refers to one of a list of about 30 Comments. If the learner types any of these words the appropriate comment will be printed and the word will not be added to the sentence.

If the learner types in a word which does figure on the word list, but which the search shows not to figure at the current position in any of the strings being searched in the file, the system will print the message:

Sorry, I can't fit that word in here, maybe your syntax is wrong.

In this way a large number of syntactic errors, mistakes of word-order and so on are automatically trapped and commented.

The second type of response to an incorrect reply to the prompt depends on accurate prediction of er-

rors. Any entry in every BRICK BLOCK or SYNTAX can have a comment attached to it. *If it* has, the system will put the whole construction on the screen as usual. When it reaches the end of the construction the relevant comment will be printed, and when the next key is pressed the whole construction is erased from the screen, leaving the earlier part of the sentence intact. In this way the program is made to appear to "backstitch", thus coping with, for example, the mistakes of gender mentioned earlier. There is a file of about thirty comments covering all the most frequent categories of error. The comments are brief, however, so that they can be made to apply to as wide a range of contexts as possible. We are currently investigating the feasibility of storing a set of grammatical and lexical explanations on the disk. Such explanations could be very much more comprehensive than any comment incorporated in the program. Storing them in this way on the disk would present an even greater advantage in that the student would be free to consult them, and above all not consult them as s/he wished. However, it still remains to be shown that the addition of explanatory material of this nature, and on the scale contemplated, would really improve the system as it stands.