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

THE DIFFICULTY OF DEVELOPING LOGICAL ALGORITHMS FOR THE MACHINE TRANSLATION OF NATURAL LANGUAGE

Ian M. Pigott,
Commission of the European
Communities, Luxembourg

Abstract

In studying machine translation software design, computer experts and linguists have traditionally concentrated on a number of phenomena deemed to present special problems and thus require particular attention. Among the favourites in this connection are morphological analysis, prepositional dependencies and the establishment of antecedents. These and similar subjects have been dealt with at great length in the numerous papers written over the years to demonstrate the necessity of adding one or more specific processing features to the software under design or pilot development.

Experience in the practical upgrading of operational systems has however tended to reveal a surprising variety of quite different problems and has shown that the fears of designers and theorists are frequently unfounded. Indeed, in tailoring a system for use by translators, many quite unexpected types of error emerge which, in the absence of sufficiently comprehensive studies, have to be eliminated largely on the basis of trial and error.

The paper presents several examples of translation problems of this type and explains how difficult it can be to formalize their resolution in computer programs. Special reference is made to the English-French version of Systran, under development at the European Commission in Luxembourg. Explanations are given of the identification of error types, the human effort involved in their study, and the testing procedures used to check the validity of the action taken to reduce their occurrence in routine translation work.

Finally, a number of suggestions are made for those working on design aspects of new systems in the hope that by paying less attention to problems which have already been solved, efforts can be concentrated on the specific areas which continue to cause frustration for those required to correct or use machine translations in practice.

Introduction

Despite the proliferation of operational machine translation systems in the last two or three years, the majority of linguistics and research departments working on the design of new systems continue to pay little or no attention to what has been achieved, preferring to propose totally innovative solutions to the problems considered to be of most significance.

One of the favourite arguments used to justify the new approach is based on a recommendation made in the ALPAC report in 1966, namely that as high quality machine translation is not likely to be realized for several decades, efforts should be based on the development of less ambitious aids in the area.

However, a great deal of water has flowed under the bridge since 1966. A number of extremely useful MT systems have been developed up to surprisingly high quality levels, levels which either provide a degree of intelligibility fully adapted to the use of raw machine translation for information scanning (as in the Russian-English system at the U.S. Air Force) or which produce machine output which can be post-edited by translators at rates of up to five pages per hour (as at the Commission or in the translation of equipment maintenance manuals at firms like Xerox).

Systran is just one of half a dozen systems used on a day-to-day basis to give assistance to translators and end-users. In terms of quality, it is undoubtedly still in the lead but other systems are catching up quickly as increasingly high volumes of translation are channelled through them. I shall however use Systran as an illustration for my talk today, not only because we now have over eight years' experience of its development and use, but also because, much like the IBM mainframe computers, Systran which started off as a very modest system, has grown stage by stage on the basis of user requirements into a package containing well over 100,000 lines of macro-assembler programming for each language pair.

Basic achievements

Before going into detail, at this point it may be useful to list the areas where Systran has achieved a level of success which would be difficult to beat whatever new approach were to be used.

On the morphology side, Systran is 100 per cent successful in identifying all the inflexional endings of verbs, nouns and adjectives in the source language and in re-establishing their equivalents for the target. The approach here differs somewhat according to the source language in question: for the less inflected languages such as English, full forms are automatically created from the stems and listed in the dictionary whereas for highly inflected languages like French, the endings are dynamically analysed by means of a table-driven algorithm.

The all important problem of grammatical homograph resolution (i.e. deciding whether a word such as 'light' is in fact a noun, a verb or an adjective in a given context) is also handled surprisingly successfully by Systran. The large majority of the most frequently occurring homograph types are invariably correctly resolved (noun vs verb, adverb vs preposition, finite verb vs infinitive, etc.) and even in cases where errors do still occur (past tense vs past participle or adjective), the hit rate is well over 90%.

Unfortunately, those working on the development of new systems do not appear to appreciate the importance of this aspect of MT analysis. Many seem to assume it is of relatively minor importance and can somehow be simply resolved by the establishment of semantic dependencies. This is certainly not the case and I would advise all those of you working on new systems to give this problem special attention from the very start.

Dictionary structure is another feature of Systran which functions remarkably well. The Systran dictionaries provide for about fifteen different levels of coding ranging from basic one-word entries to highly complex multi-word contextual rules which can be powerful enough to override analysis algorithms if and when this proves to be necessary. Admittedly, months if not years of experience are required for a coder to make optimal use of these dictionary features. Nevertheless, the wide variety of dictionary support available is no accident. Each and every feature has its own special function and has been specifically developed to meet a given requirement.

As a good dictionary is a prerequisite to high-quality MT, system developers would be recommended to design lexical data bases which not only provide a firm basis for the various levels of coding required but which can be updated at reasonable cost. It will be remembered that the TAUM system was discontinued mainly because of the excessive cost of increasing the size of the dictionary. TAUM entries used to cost up to $40 each while the cost per entry in other production systems, including Systran, is below $5.

Target language generation in Systran also presents very few problems at this stage of development. In other words, provided the results of source language analysis are correct, the target synthesis and rearrangement processing rarely produces any surprises. Target generation has two main functions: the first is to inflect correctly (person, number, gender, case, etc.) all nouns, verbs and adjectives in the sentence, while the second is to place all the words into the correct order for the target language in question.

Although this, like other developments, took time, people and money, the amount of effort was not nearly as great as might have been feared. The establishment of synthesis rules proved to be a relatively straightforward task, requiring no more than about three man-years for each new target language.

I have, in recent years, seen several learned accounts of the difficulties of handling target generation. I can only say that in practice this level of processing, unlike analysis of source language, has turned out to be fairly mechanical, representing not more than about ten per cent of the effort required on any language combination. Those who continue to propose new approaches based on the 'special requirements' of the target language in question would be well advised to investigate the dependability of what has already been achieved.

Finally, turning to source language analysis, which is certainly by far the most complicated part of MT, I would only say at this stage that Systran has indeed achieved a relatively high level of success. Most of the source-language sentences are satisfactorily parsed, even though some annoying errors still occur at times. I shall attempt to explain why they occur and how they could be eliminated later in my talk.

The pragmatic approach

My colleague, Peter Wheeler, also speaking at this conference has commented on the importance of meeting user requirements by a pragmatic approach to the problems in hand. I in turn should like to consider some of the linguistic problems which have caused real difficulty and explain how they were solved, before going on to argue why many of the fears of more theoretically oriented linguists working on MT developments are largely unfounded.

Let us therefore go back to February 1976 when the Commission first started to develop the English-French Systran system. With a dictionary of only a few thousand words and only a fairly small program, there was ample opportunity for eliminating errors. Indeed, in those early days, practically every sentence contained a wide variety of mistakes at all levels, enough to persuade most of those assigned to the project to give up in despair.

The errors, as might be expected, occurred at two main levels - dictionary and program.

It was quickly realized that the performance of the program could not be properly judged until an adequate amount of basic vocabulary was available. The first priority was thus to create a well-coded dictionary for the text corpus on which we were working. This happened to be the Food Science and Technology Abstracts data base which had been chosen in view of the subject area involved (agriculture) and owing to the fact that it already contained thousands of pages of text in machine-readable form, thus eliminating the need to input source text manually.

The dictionary

In this initial dictionary work, we made wide use of three types of tool, a key-word-in-context listing of all the words in a 20,000-sentence sample of text, word-frequency counts for the same sample and raw machine translations of about 1000 sentences a time complete with not-found-word lists.

The raw MT served as a basis for deciding which words and expressions required coding while the KWIC and frequency listings provided an indication of the various contexts in which a given word was likely to appear, together with its more general frequency of occurrence. The normal practice was to go through the raw MT sentence-by-sentence, code up missing vocabulary, check against the not-found-word-list in order to avoid duplication and choose the most generally acceptable basic meaning making full use of the KWIC information.

It may be thought that by working on a supposedly limited corpus of this type, problems of meaning could be avoided. In fact, this was seldom the case. We soon learnt that the data base covered all aspects of food science from farming to processing, from chemical testing to legislature and standards, and from biology to environmental pollution. As a result, it proved to be an excellent testbed for the work in hand.

Very quickly we abandoned the idea of the topical glossary approach which is available in Systran and has been used to good effect by those who deal principally with one major sector of interest. This facility quite simply allows a basic meaning to be selected on the basis of a subject-field parameter in preference to other meanings a word might have in other contexts.

There were two reasons why we decided not to use topical glossaries. On the one hand we soon discovered that constantly occurring terms such as 'plant' could, even in the field of food science have quite different meanings - either a growing vegetable organism (F - plante) or an industrial facility (F - installation). Surprisingly enough, it turned out that the second meaning was the more frequent even in this subject sector. The other reason was that although we based our initial development on the food sector, we realized from the very beginning that if MT was to be of real use at the Commission, it would have to be able to cope with practically any subject sector under the sun.

A good illustration of a general purpose basic meaning is the translation assigned to the word 'station'. 'Gare' would have been too specific, 'station' although understandable would rarely be correct, and so we finally opted for 'poste' which is correct in most contexts and understandable in many more. 'Poste de chemin de fer' is not too bad a translation of 'railway station' but 'gare de telecommunications' would be quite unintelligible.

This last example brings on to the next level of dictionary coding, the string expression, which can be assigned both a basic meaning and additional syntactic information. 'Railway station' can be given its own meaning 'gare' and coded in such a way that 'station' will in this context be systematically recognized as a noun rather than as a verb.

I might add that as often as not, the use of common nouns such as 'station' as verbs is overlooked by the dictionary coder until the day when we get a sentence such as 'France has decided to station troops in Beirut'

Many interesting pages could be written on expressions coding, but I would just like to mention here the usefulness of coding various types of string expression as a means of overcoming syntactic ambiguity. The phrase 'in order to' could theoretically have two quite different meanings, depending on whether 'order' is interpreted as a noun in its own right (as in 'He returned it in order to the owner') or simply as part of an infinitive particle expression of purpose. In practice of course, the infinitive particle occurrence is by far the most common and can be entered as the only possibility to be considered. I must admit, however, that although I once predicted the other meaning would never turn up, I have since been proved wrong, but in my opinion, 5000 hits to one miss is far better than 4800 hits and 200 misses even if it happened that one of the 4800 gave the correct resolution of the less common meaning. Indeed, I would say that it is just this kind of pragmatic approach which has been responsible for the success of our development.

The program

Once we had built up a reasonable dictionary, we were in a position to take a more objective look at the translation program. We could run new raw batches of MT and study the results of the sentences which, despite the fact that all the words were in the dictionary, still continued to produce errors.

At this stage the errors fell into four basic types, those which resulted from insufficient information in the dictionary and which could normally be eliminated without too much difficulty, those resulting from poor homograph analysis which required much more work, those requiring extension to the various stages of the parsing routines which accounted for a substantial amount of effort over the first four years, and those which necessitated the introduction of contextual dictionary rules often in conjunction with semantic markers. I shall try to give you a typical example of each in order to illustrate what is bound to happen during the development of any system and how efforts can be made to solve the problems involved.

Lack of dictionary information

As an illustration of lack of dictionary information, let us take the sentence:

- Many institutions but particularly the Commission had been considered by the study.

The raw MT might have come out:

- Plusieurs institutions mais la Commission avait été particulièrement considerée par l'étude.

The translation is of course quite wrong but at first sight, it is difficult to see why it has gone wrong. On checking the dictionary information, we find that everything appears to be correct and even the program seems to have functioned as it should. Only when we get a dump of the actual analysis do we find that 'particularly' has been marked as an adverb governing 'considered'. Had other adverbs been used (e.g. unfortunately the Commission had been considered by the study), the analysis would have been correct.

The reason things went wrong was quite simply that while the system provided for codes to mark the affinity between an adverb and a verb or an adjective, no such code existed for an adverb governing a noun. Yet in our sentence 'particularly' is indeed governing a noun. Once we had diagnosed the trouble, we were able to add a new code, slightly modify the analysis program and obtain the correct translation:

- Plusieurs institutions mais en particulier la Commission avaient été considerées par l'étude.

Not purple prose perhaps, but at least syntactically correct and quite intelligible.

Homograph errors

The type of homograph error that might have occurred in the early stages can be illustrated by the translation of:

- The laboratory analyses improved awareness of the problem.

as:

- Le laboratoire analyse la conscience ameliorée du problème.

What has happened, of course, is that 'analyses' has been resolved as a verb with the result that 'improved' becomes a past participle adjective instead of a simple past tense. This kind of error is far more difficult to eliminate and while special cases such as this could be dealt with quite simply by entering 'laboratory analysis' in the dictionary as a noun phrase, a more general approach to the problem would require systematic study of hundreds, if not thousands of sentences of the same type.

Such studies have indeed been conducted over the years and I am pleased to report that they have been largely successful. By and large, we attempt to design the program to make full use of the syntactic and semantic information available on each word in order to arrive at the most likely solution on the basis of a sizable error corpus. Once the program has been modified, similar but new material is run for checking, negative side effects are noted and further modifications are made. Slowly but surely performance increases. However, for some of the more common homograph types such as noun vs verb, a routine can easily run to thirty or forty pages of contextual programming.

Parsing errors

As an example of a typical parsing error, I would give the following example:

- The committee discussed faulty equipment and office management.

Early in development, this might have been translated:

- Le comité a étudié l'équipement et l'administration de bureau défectueux.

as the adjective 'faulty' would be analysed as qualifying both 'equipment' and 'management' rather than just 'equipment'. When we read a sentence such as this, there is absolutely no doubt in our minds that the 'faulty' refers only to 'equipment' and not to 'management', the reason being that we have all had plenty of experience of faulty equipment and we know only too well that management committees are unlikely to criticise management, however bad it may be.

Yet the computer has no such inborn intelligence. The problem can however be solved on the basis of the most likely syntactic and semantic relationships between nouns and adjectives of given types. For example in

- faulty typewriters and photocopiers

'faulty' would govern both nouns as they both come into the same semantic category (both are devices), whereas in our first example, it would qualify only 'equipment' which would not carry the same semantic markers as 'management'.

Contextual dictionary entries

Finally, to turn to the contextual dictionary entries, I will take a seemingly very simple example, the preposition 'in'. I may say that in practice the correct translation of the preposition 'in' has turned out to be one of our most difficult meaning problems. Indeed, there are some 550 contextual entries attached to this unassuming little word, not to speak of a series of special routines which deal with its translation in date structures and in connection with place names.

The basic dictionary default for the translation of 'in' is 'dans' but there are a great many cases where that meaning is incorrect.

Three simple examples will illustrate the point:

'In' governing the name of an organization (on the basis of semantic coding) will be translated 'à' (à la Commission, aux Nations Unies).

'In' immediately governing a material will be translated 'en' with no article (en acier, en bois).

'In' immediately governing an animate noun will be translated 'chez' with a definite article (chez les hommes, chez les rats).

These examples might appear childish in their simplicity, but as I said before, there are over 500 such codings of greater or lesser complexity and it has taken many man years of effort to cover them all adequately. No printed dictionary will give anything like a proper explanation of the variety of translations required. Only on the basis of working through thousands of pages of real text does the true extent of the problem become apparent, and only then can it be dealt with successfully.

Academic concerns

Now that we have looked at some typical practical problems in MT development, let us turn for a moment to some of the concerns about machine translation often expressed by academic linguists working at research centres or at the universities.

Reams and reams have been written about prepositional dependency and its importance in machine translation. How important is the problem, how easy is it to resolve and to what extent can existing systems cope with it?

First I would say that the establishment of prepositional dependencies is nowhere like as important as finding the correct translation of the prepositions once their relationships have been established. We have already seen the magnitude of the problem of dealing with the translation of 'in' but the same can be said of most common prepositions (at, to, with, by, for, etc.).

By contrast, the actual setting of relationships is a comparatively simple process, particularly as in the majority of cases in written texts, prepositions govern the noun phrase to the right and are rarely governed by the noun, verb or adjective to the left. Where special government does need to be handled, it can almost always be efficiently handled by appropriate contextual coding. It is also of interest to note here that translators seem far less concerned than might be expected by the occasional incorrect setting of prepositional dependencies resulting in the wrong translation.

On the subject of antecedents, which again has been given considerable attention in academic circles, in practice we have found that a pronoun subject normally has as its antecedent, the noun which was the subject of the last main clause. Often, of course, the last main clause with a noun, rather than a pronoun, as subject is to be found in a previous sentence. In Systran, however, information about previous sentences is stored and the problem can be easily solved.

There are of course exceptions to this general rule, some of which can be successfully handled by the program, some of which can not. And although there have been one or two complaints from translators, by and large the system appears to perform well. At the human level, this type of error is, in any case, among the easiest to correct.

I have mentioned these two examples in the interest of restoring some kind of reasonable perspective regarding the problems to be solved in developing new MT systems. Indeed, on the basis of our experience to date, the real problems reside at quite different levels.

Problems to be solved

Rather surprisingly perhaps, the ongoing quality enhancement of raw MT output at this stage seems to be more dictionary bound than program dependent. We do of course continue to enhance the programs, particularly in regard to the establishment of enumerations and the resolution of grammatical homographs and

clause boundary setting, but most of our effort goes into the addition of contextual coding entries aimed at providing the translator with as much reliable technical terminology as possible. This is a long, rather tedious process, particularly in an institution like ours where there seems to be no end to the number and complexity of subject matter covered in day to day work. However, as post-editing work consists essentially of finding the 'mot juste' for each and every context, the better the terminology provided by the machine, the easier the job for the translator.

I would therefore suggest that in the development of new systems, designers pay far more attention than they have in the past, to creating the best possible dictionary structures and terminology updating features, so as to minimize the human effort involved in dictionary improvement and expansion. Such a system could be based on the interfacing of source language analysis results with word processing systems used for post-editing. If this could be achieved, a dictionary coder would be able to avoid a great deal of analytical coding work as he would be able to make use of menu proposals as a basis for dictionary creation.

For example, the post-editing changes made by a translator could be coupled to the source text to provide potential equivalences for inclusion in the dictionary. At the simplest level, these would be at the one-word level and would simply propose as a basic meaning, the equivalent inserted by the post-editor. For example, potential information on a not found word such as 'post-editor' could be created on the basis of its ending (e.g. noun, plural in -s, human, profession, etc.) and the meaning inserted by the translator (post-editeur).

At the next level, expressions requiring post-editing corrections could be listed for approval as string technical terms. 'Word processing' could lead to information such as: noun expression, main meaning - traitement de textes.

Finally, in order to facilitate the introduction of the correct meaning of a term in context, the updating algorithm could make use of the parsing information available from analysis in a given text and propose selection criteria for the correct contextual equivalent. As an example, let us take the sentence:

- The equipment appeared to work successfully under normal operating conditions.

In the absence of contextual meanings, the verb 'work' might well have been assigned its basic meaning 'travailler' rather than the post-editor's choice 'fonctionner'. The updating proposals, based on the sentence structure, might look something like this:

- WORK, meaning 'fonctionner' when:
* a. semantic subject = DEVice
* b. noun subject = equipment
* c. modified by adverb = successfully
* d. used intransitively
* e. dependent on modal = appear

The translator would then simply select one or more of the proposals as a basis for contextual meaning selection. Here he might choose a) and c), or, if he wished to be more specific, he might only select b). Following his selection, which would only take seconds, the necessary syntactic and semantic information would be correctly formatted into a dictionary rule. After updating, which ideally should be immediate for each entry, any conflicts with existing information would be displayed, allowing the coder to make any additional changes with or without the assistance of his colleagues.

This kind of computerized aid to dictionary making would certainly be a tremendous aid to the coder and would make for cheap, rapid dictionary building. If this could be ensured, the overall cost of MT improvement would be drastically cut. And once the algorithm had been successfully developed for dictionaries alone, it could possibly be extended to interface with the program itself as an aid to program enhancement.

Logical algorithms for natural language

And this brings me to my last and most important point.

In our experience, the most time-consuming part of development work has been related to the difficulty experienced by linguists in reducing natural language to logical processing at the programming and dictionary levels. At all levels of the system, translators, linguist-programmers and systems experts have constantly had difficultly in predicting the overall consequences of an addition or modification to the program or dictionaries.

More often than not, it has been necessary to make use of the only remaining tool available, that of trial and error. In language processing there are indeed very few, if any, hard and fast rules. This has meant that each basic rule has led to a general series of exceptions, each of these has in turn led to more specific exceptions and so on, down the line. Even the seemingly most straightforward rules such as 'a pronoun immediately preceding a finite verb is the subject of that verb' can lead to dozens of general exceptions and hundreds of specific ones.

It has taken us over eight years to create software packages of well over 100,000 lines of assembler programming and some 150,000 dictionary entries for each language pair. The capacity of the computer has, by the way, not proved to be a problem, nor has running time or even running cost. The basic problem has been one of training translators and linguists to reduce their knowhow to logical patterns of thought which can be programmed into the computer, tested, amended, added to, further tested and so on until reliable results can be obtained.

The work continues year by year and as the quality of the raw MT improves, so the number of enthusiastic users expands.

However, with more and more users of MT - and remember that over 400,000 pages of MT were run in 1983 on the various production systems now in use - the problem of successfully integrating user feedback becomes ever more complex.

Conclusion

My conclusion today is, then, a very simple one.

I would urge all those concerned with the basic design or further improvement of MT systems to concentrate their efforts towards assisting dictionary makers and programmers in their routine work.

Rather than coming up with new revolutionary theories based on artificial intelligence or Chomskian linguistics, they should attempt to find ways and means of prompting the human beings concerned with development into making the best possible use of their knowledge and experience by providing them with efficient updating features enabling them to make full use of the feedback received from users.

Finally, let us never forget that MT systems are intended to help overcome language barriers by speeding up the rate at which translators can handle their work. Translators' requirements are thus of paramount importance and should be borne in mind at every stage of design and development.