STRATEGIES AND HEURISTICS IN THE ANALYSIS OF A NATURAL LANGUAGE IN MACHINE TRANSLATION
(In the memory of Bernard Vauquois)

Zaharin Yusoff

Groupe d'Etudes pour la Traduction Automatique
BP n° 68
Université de Grenoble
38402 Saint-Martin-d'Hères
FRANCE

ABSTRACT

The analysis phase in an indirect, transfer and global approach to machine translation is studied. The analysis conducted can be described as exhaustive (meaning with backtracking), depth-first and strategically and heuristically driven, while the grammar used is an augmented context free grammar. The problem areas, being pattern matching, ambiguities, forward propagation, checking for correctness and backtracking, are highlighted. Established results found in the literature are employed whenever adaptable, while suggestions are given otherwise.

1. INTRODUCTION

We interest ourselves in the analysis phase of a machine translation system which adopts the indirect, transfer and global approach (see [Slocum 84]). The aim of this paper is to clarify the problem areas, suggest a few solutions, and point out the loose ends. There is no current implementation for the analyser we describe, and the ideas are basically a reflection of what we would like to see in an MT system, based on previous experience in the field. A very important issue is to separate the linguistic knowledge (the grammar) from the algorithmic and technical knowledge (the organisation of the analyser, pattern matching, etc.). "Approximate" linguistic knowledge is also separated and used as a means to guide the analysis rather than considered as absolute (semantics and context constraints as heuristics instead of grammar rules).

Due to space restrictions, we shall immediately specify the basic type of analyser we shall be working with, without giving any reasons for the choice. The interested reader is referred to [Zaharin 85] for an uncondensed version of this paper, and [Zaharin 86] for more details.

2. THE ANALYSER

In general, an analyser can be viewed as a black box with two holes, where we insert the input text through one and it gives the output linguistic structure through the other (in our case, an annotated tree representing the "meaning" of the input text). Peeping into the box, we would notice that it works in cycles doing the following five steps until it triggers off some stopping mechanism and hence furnishing the output :

a) computing the object set ;
b) choosing an object ;
c) computing the rule set ;
d) choosing a rule ;
e) applying the chosen rule on the chosen object.

Naturally, depending on the various models, these steps need not be executed in the given order, nor are they necessarily as clear out. Indeed, some may even execute the cycles in parallel.

Our analyser will do the five steps in the following manner. Steps (a) and (c) will be done together, computing all objects on which some rule is applicable, and to each of these objects, the set of all applicable rules is computed. The result is a set of linked pairs O-R where R is a rule applicable on the object O. A linked pair is then picked, i.e. steps (b) and (d) together, and the chosen rule applied on the chosen object. The cycle repeats.

The motivation for the above choice is that we are aiming for a one-go analysis, for which we shall be needing the maximum of information before we apply a rule, hence the computation of all candidate objects and rules. Strategies and heuristics are then needed for the critical choice of object-rule pair in each cycle.

The natural language treated will be described by a grammar containing a set of rewrite rules with a context free base of the form $X_1 \ldots X_n \rightarrow X$ where $X_1, \ldots, X_n, X$ are annotated trees ; in other words, an augmented context free grammar. What we actually have in mind is a grammar containing rules of the form given in figure 1, as discussed in [Zaharin 86]. Nevertheless, the discussion remains valid for any system using a similar representation of data.

As in most machine translation systems, the analysis looks for only a single solution, i.e. a single representation of meaning for each input text. If the text is ambiguous, the "best" solution is taken. In the search for a solution, a depth first approach is taken, and the analyser allows for backtracking in case the solution is not found in one go. Backtracking is also required in cases where an input sentence is not in the language of the grammar, but most important, to ensure that the analyser finds a solution if there is one.

3. THE PROBLEM AREAS

With the type of analyser we have chosen, the problems that arise are basically the following :

- pattern matching ;
- ambiguities ;
- forward propagation ;
- checking for correctness ;
- backtracking.

Pattern matching seems to be the bottleneck of the realisation of any system. Fortunately, the literature already contains some efficient pattern matching procedures that can be modified to suit our model.

The choice of an augmented context free grammar means that the rules are basically in the form of strings of symbols, where each symbol is augmented with an annotated tree structure. Figure 1 gives an example of a rule we use (see [Zaharin 86]). In this form, the pattern matching can be carried out in two stages : one for strings, followed by one for trees, where the latter (the more costly one) is triggered only in cases of success of the former.

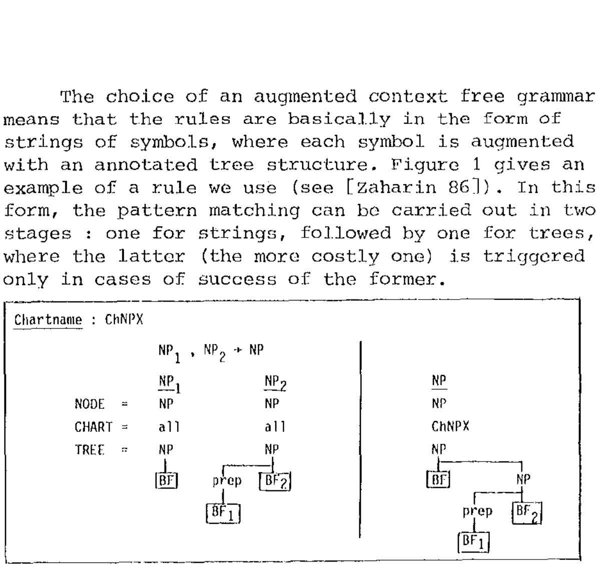


Figure 1

For instance, at the string level, this falls into the category of many pattern/many object pattern matching for strings, for which the procedure of [Aho & Corasick 75] which finds all patterns and all objects in one pass seems suitable. Only in cases of success do we pass on to a tree pattern matching process, for instance that of [Hoffmann & O'Donnel 79]. Repetitive work can be avoided if we factorise the results of the pattern matching from one cycle to the other.

Ambiguities and forward propagation are the two major problems for the model. We defer the discussion on these to the next two sections.

In general, it is very difficult to describe a natural language exactly by means of a formal grammar, no matter how sophisticated the formalism. In spite of this, the criteria for the correctness of the result of the analysis is usually with respect to the natural language treated and not that of the grammar, i.e. finding an axiom in the grammar may not be sufficient. So, rather than writing very strict rules at the risk of excluding correct structures, it is better to have more general rules which may accept anomalous structures, and then provide a filter to reject such results. [Boitet & Gerber 84] suggests an expert system to do this post-analysis checking.

For backtracking analysis, a simple model is to store failed configurations (dead states) in some file. At the beginning of each cycle, the new configuration can be checked against this file, backtracking further if it compares. This may seem a huge effort, but natural language analysis is such that many identical nodes may be found in different parts of the search space. As for forward propagation in the backtracking analysis, the priority orderings to be discussed in 5 can be preserved and made use of here.

4. AMBIGUITIES

Ambiguities haunt every treatment of a natural language. [Lepage 85] summarises the types of ambiguities that we face, both lexical and structural, while [Van Klinken 84] writes on the methods used to solve some of the cases. However, until a formal treatment of ambiguities can be proposed, the solutions will remain ad hoc, treating case by case as we meet them.

In general, lexical ambiguities are solved either grammatically, with context, or with semantics. Grammatically is as in the sense of using agreement in number to obtain "that" as a conjunction instead of a determiner in the sentence :

We know <u>that</u> ambiguities are difficult to solve.

We use context to distinguish the past participle "collected" from the verb in the two sentences :

The corals <u>collected</u> at the bottom of the sea are beautiful.
The corals <u>collected</u> at the bottom of the sea.

Finally, [Lytinen 85] pointed out the need of semantics to determine the attachment of the prepositional noun phrase "for $10" in the two sentences (based on the verb "found") :

The cleaners dry-cleaned the coat that Mary <u>found</u> at the rummage sale <u>for $10</u>.
The cleaners dry-cleaned the coat that Mary <u>found</u> in the garbage <u>for $10</u>.

Whereas we can be quite certain of the solutions obtained grammatically, the use of context or semantics does not inspire the same confidence. Context can pose problems when locating the elements the context refers to, which can be arbitrarily far away from the ambiguous word. Furthermore, the problem can be aggravated by the elements looked for being ambiguous themselves. Sometimes, negative constraints are used in context elements, and these can pose interpretation problems (see [Zaharin 86]). As for semantics, the arguments can be endless.

Bearing the above in mind, we prefer to treat the solution of lexical ambiguities as heuristics rather than steadfast rules. By this we mean that context and semantics should not be incorporated into the grammar rules used to describe the language treated, but instead should be placed in related heuristic rules which advise on the applicability of their counterparts. This also means that if their advice has not led to a success, it is possible to backtrack to the same rule and recommence, this time ignoring the advice. The case would not have been possible if the grammar rule and the context and semantics had been put together in one rule.

In the case of structural ambiguities, the sentence can be inherently ambiguous, in which case context and semantics heuristic rules can only aid to pick the preferred reading. It is also possible that structural ambiguities occur only at the level of substrings of the sentence, but some of the possibilities will not lead to a solution. In such a case, heuristic rules for preferred readings will also help, but the problem is more of choosing a rule or object to avoid leading to a dead end. This falls into the category of problems to be discussed in the next section.

5. FORWARD PROPAGATION

Forward propagation is the problem of choosing a rule and an object for application in each cycle of the analyser. This is the execution of steps (b) and (d) in section 2, which is then followed by step (e), completing the cycle. As we are aiming for a solution in one pass of the analysis, the choice is critical, as even a wrong choice of a sequence of applications may lead to a dead end. This can be seen in the following example where the grammar contains the rules (omitting the details) :

R_1 : NP VK NP → VCL ; R_3 : RELCL PNP → RELCL
R_2 : NP RELCL → NP ; R_4 : VCL PNP → VCL

Taking an example in [Lytinen 85], the analysis may find itself in the state given in figure 2 (the candidate objects are circled and the corresponding rule indicated). The sequence of applications needed in this example is R_2 R_1 R_4. If we happen to choose R_1, before R_2, we will find that the analysis will not lead to a solution.

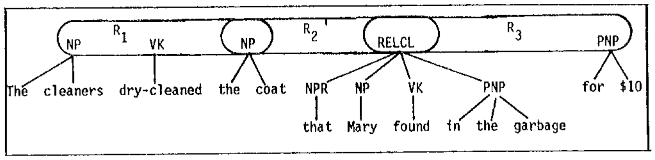

NP
R1
VK
NP
R2
RELCL
R3
PNP
The cleaners dry-cleaned the coat
NPR NP VK PNP
that Mary found in the garbage
for $10

Figure 2

The situation given here is one of the major problems faced by analyses which predefine the sequence of applications of rules. There is nothing more frustrating than not obtaining a complete analysis and yet knowing that the required rules are present in the grammar.

Instead of predefining a sequence of rule applications, we prefer using heuristic rules which apply independently in each cycle of the analyser. These heuristic rules act to determine a priority ordering of the candidate rules and objects (each rule will be tied to the object it is applicable on), the highest priority rule or object being chosen for application (taking along the object or rule it is tied to).

The big question is, what should these heuristic rules contain ? First and foremost, coming from the discussion on solving ambiguities in section 4, we need the treatment of semantics put down as heuristics. An example of such a case is in figure 2 where rule R_3 should be accompanied by a heuristic rule to check for semantics. Here, one does not "find" something "in the garbage" "for $10", and so the heuristics would advise that R_3 should not apply (unless, as discussed before, following this heuristics leads to a dead end, and so we come back to apply R_3).

We shall refer to the type of heuristics just used as the "to-apply-or-not-to-apply" heuristics. The type of heuristics mainly needed is the "after-you-or-after-me" heuristics. This is the case for the choice between R_1 and R_2 in figure 2.

For the said problem, one may argue that VCLs are higher up in the hierarchy of phrases and clauses than NPs [Vauquois & Chappuy 85], and so rules building NPs should be applied before rules building VCLs. This may be true in this example, but care must be taken when we deal with complex clauses and phrases (the hierarchy given in the reference is for simple clauses and phrases). For complex clauses and phrases, we may obtain cyclic hierarchies between NPs and RELCLs, APs and NPs, etc. For such cases, ad hoc heuristics are needed, for instance, rules building RELCLs should apply before rules building NPs if the former is found to the right of the latter, and the inverse otherwise.

Apart from some hierarchy given, context can also be used to solve the "after-you-or-after-me" problem. (Recall that context is also needed to solve ambiguities). As examples, suppose the grammar for figure 2 also contains the rules (still omitting details) :

R5 : NP VK → VCL ; R8 : SCL VCL → VCL
R6 : NP VK AP → VCL ; R9 : SCL VK AP → VCL
R7 : CONJ VCL → SCL

Checking the context, namely the conjunction "when" or "that", can be used to choose R5 on "the king rides" in figure 3, while R1 is chosen in figure 4 (this also gives an example as to why we would not use heuristics like "apply the rule with the longer LHS").

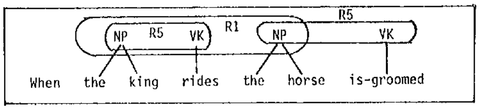


Figure 3

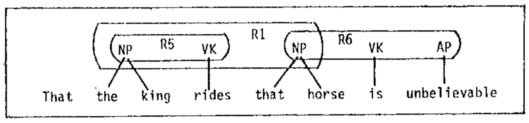


Figure 4

In the example in figure 3, it so happens that the two occurrences of the rules R5 are independent, in the sense that the application of one before the other has no great consequence, and so an arbitrary choice will do. However, not competing on intersecting objects does not necessarily guarantee independence. Had we been two steps before figure 2 with rules :

R10 : NPR NP VK → RELCL ; R11 : PNP PNP → PNP

included in the grammar, the situation would be as given in figure 5. Here, semantic heuristics can advise that R11 should not apply. However, we need to make sure that R10 applies before R1, otherwise we can never arrive at a complete analysis even though these objects seem to be independent.

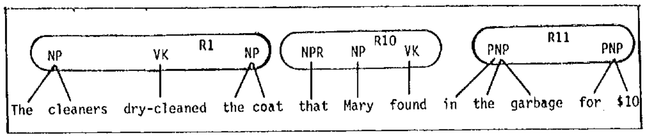

NP VK R1 NP
NPR NP R10 VK
PNP R11 PNP
The cleaners dry-cleaned the coat that Mary found in the garbage for $10

Figure 5

No doubt the above problem can be solved using the same heuristics giving the hierarchy of clauses and phrases, but this situation brings up two important questions which necessitate answering : firstly, how do we expect such situations ? And secondly, do we have to know all such situations before we can write a good set of heuristic rules ?

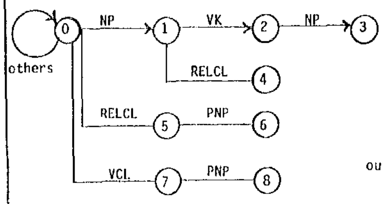
Before attempting to answer these questions, we wish to highlight the heuristics used by [Nagao & Nakamura 82] which looks very promising but may suffer from the same drawbacks. The reference uses sentential patterns (SP) that express the global structure of a sentence. If these SPs are satisfied by a certain configuration, the said configuration is chosen for expansion (they use a best-first search where a configuration is then a node in the search space).

For our purposes, SPs can also be adapted either to place checks testing whether the analysis is on the right track, or to create subgoals if some of the constituents of the SP are satisfied.

This can be useful for configurations containing specific words which can determine its neighbours. For example, a conjunction necessitates a VCL or PARTCL to its right.

Going back to the two questions posed earlier on, the problem of expecting the situations where heuristic rules can be written is not a simple one. For a given derivation tree in a context free grammar, any cut in this tree is a possible configuration of the correct analysis. Passing this cut through the pattern matcher will give the complete configuration. Looking at all possible substrings of this cut, and multiply this by the number of all cuts in the derivation tree will give us the situations we need to predict. The result is by no means negligible, to say the least.

Fortunately, rules that can apply on intersecting objects can be precomputed. In particular, if we use the pattern matching procedure of [Aho & Corasick 75] as mentioned earlier, the procedure produces a network equipped with a failure function indicating to which part of another rule (say Rb) the pattern matcher is to go to after successfully finding a pattern (say rule Ra). This gives a possible clash between rules Ra and Rb, where Ra is on the left of Rb. For example, the clash between R1 and R2 in figure 2 can be predicted by the pattern matcher, to which a heuristic rule can be written, say the one given in figure 7. Figure 6 gives the network for the pattern matcher of the reference for the rules R1 to R4 in our example. The failure function is given by f(i) where i is a state of the network while the output of applicable rules is given by output(i) (again we omit the details of augmenting each arc by the TREE value). We refer the reader to the reference for further details.



failure function :

i	1	2	3	4	5	6	7	8
f(i)	0	0	1	5	0	0	0	0

output

i	3	4	6	8
output(i)	R1	R2	R3	R4

Figure 6

```
IF (a) R1 is applicable
   (b) R2 is applicable
   (c) R1 is on the left of R2
THEN
   priority(R2) := priority(R2) + 1    (the higher the score, the higher
                                        the priority).
```

Figure 7

As for having to predict on possible situations, we can cut down on some work by making the analyser "reason" a little. For example, the analyser should be able to deduce from the situation in figure 5 that it can get to the situation in figure 2 and hence apply the heuristic rule already written for figure 2 (in this case the one given in figure 7). This reasoning can be done in the following manner : with R1 applicable on "NP VK NP" (see figure 5), the failure function for R1 points to state 1 (see figure 6), and with the applicable rule to the immediate right of R1 being R10 which produces a RELCL, this gets us to state 4 with output R2. We then obtain a slightly different situation from figure 2 but the heuristic rule can still apply, giving priority to rule R2 hence R10.

Despite the title, we have hesitated on discussing strategies, because experience tells us that it is very difficult to write admissible strategies (i.e. set sequences of heuristic rules). Furthermore, strategies may be as risky as procedural methods unless they are flexible enough. This means that they can be halted, created, interrupted and resumed during the analysis. Furthermore, they ought to be global rather than particular. For example, the hierarchy of clauses and phrases can serve to choose between rules having the same priority after other heuristic rules have applied, and halted when complex structures are treated. An interesting discussion on global and flexible strategies is found in [Hayes-Roth 85] for the expert system OPM.

REFERENCES

A.V. Aho and M.J. Corasick, "Efficient string matching : an aid to bibliographic search". CACM, June 1975, vol. 18, n° 6, pp 333-340.

Ch. Boitet and R. Gerber, "Expert systems and other new techniques in MT systems". COLING-1984.

B. Hayes-Roth, "A blackboard architecture for control". AI 26 (1985), pp 251-321.

C.M. Hoffmann and M.J. O'Donnell, "Pattern matching in trees". Computer science department, Purdue Univ.

Y. Lepage, "Ambiguïtés et traduction automatiques, les méthodes du GETA". BRISES, Octobre 1985.

S.L. Lytinen, "Integrating syntax and semantics". Proceedings of the conference on theoretical and methodological issues in machine translation of natural languages. COLGATE Univ., New-York, August 1985.

M. Nagao and J. Nakamura, "A parser which learns the application order or rewriting rules". COLING-1982.

J. Slocum, "Machine translation : its history, current status and future prospects". COLING-1984.

C. Van Klinken, "Disambiguation strategy in English structural analysis". Laporan Teknikal, Projek Terjemahan Automatik. Universiti Sains Malaysia, Penang, December 1984.

B. Vauquois and S. Chappuy, "Static grammars : a formalism for the description of linguistic models". Proceedings of the conference on theoretical and methodological issues in machine translation of natural languages. COLGATE Univ., New-York, August 1985.

Zaharin Y., "Strategies and Heuristics in the analysis of a natural language in Machine Translation". GETA document, November 1985.

Zaharin Y., "Strategies and Heuristics in the analysis of a natural language in Machine Translation". PhD thesis, Universiti Sains Malaysia, Penang, March 1986. Research conducted under the GETA-USM Cooperation (GETA document).