ROBUST PROCESSING IN MACHINE TRANSLATION

Doug Arnold,

Centre for Cognitive Studies,
University of Essex,
Colchester, CO4 3SQ, U.K.

Rod Johnson,

Centre for Computational Linguistics
UMIST, Manchester,
M60 8QD, U.K.

ABSTRACT

In this paper we provide an abstract characterisation of different kinds of robust processing in Machine Translation and Natural Language Processing systems in terms of the kinds of problem they are supposed to solve. We focus on one problem which is typically exacerbated by robust processing, and for which we know of no existing solutions. We discuss two possible approaches to this, emphasising the need to correct or repair processing malfunctions.

ROBUST PROCESSING IN MACHINE TRANSLATION

This paper is an attempt to provide part of the basis for a general theory of robust processing in Machine Translation (MT) with relevance to other areas of Natural Language Processing (NLP). That is, processing which is resistant to malfunctioning however caused. The background to the paper is work on a general purpose fully automatic multi-lingual MT system within a highly decentralised organisational framework (specifically, the Eurotra system under development by the EEC). This influences us in a number of ways.

Decentralised development, and the fact that the system is to be general purpose motivate the formulation of a general theory, which abstracts away from matters of purely local relevance, and does not e.g. depend on exploiting special properties of a particular subject field (compare [7], e.g.).

The fact that we consider robustness at all can be seen as a result of the difficulty of MT, and the aim of full automation is reflected in our concentration on a theory of robust processing, rather than 'developmental robustness'. We will not be concerned here with problems that arise in designing systems so that they are capable of extension and repair (e.g. not being prone to unforseen 'ripple effects' under modification). Developmental robustness is clearly essential, and such problems are serious, but no system which relies on this kind of robustness can ever be fully automatic. For the same reason, we will not consider the use of 'interactive' approaches to robustness such as that of [10].

Finally, the fact that we are concerned with translation militates against the kind of disregard for input that is characteristic of some robust systems (PARRY [4] is an extreme example), and motivates a concern with the repair or correction of errors. It is not enough that a translation system produces superficially acceptable output for a wide class of inputs, it should aim to produce outputs which represent as nearly as possible translations of the inputs. If it cannot do this, then in some cases it will be better if it indicates as much, so that other action can be taken.

From the point of view we adopt, it is possible to regard MT and NLP systems generally as sets of processes implementing relations between representations (texts can be considered representations of themselves). It is important to distinguish:

(i) R: the correct, or intended relation that holds between representations (e.g. the relation 'is a (correct) translation of', or 'is the surface constituent structure of'): we have only fairly vague, pre-theoretical ideas about Rs, in virtue of being bi-lingual speakers, or having some intuitive grasp of the semantics of artificial representations;
(ii) T: a theoretical construct which is supposed to embody R;
(iii) P: a process or program that is supposed to implement T.

By a robust process P, we mean one which operates error free for all inputs. Clearly, the notion of error or correctness of P depends on the independent standard provided by T and R. If, for the sake of simplicity we ignore the possibility of ambiguous inputs here, we can define correctness thus:

(1) Given P(x)=y, and a set W such that for all w in W, R(w)=y, then y is correct with respect to R and w iff x is a member of W.

Intuitively, W is the set of items for which y is the correct representation according to R. One possible source of errors in P would be if P correctly implemented T, but T did not embody R. Clearly, in this case, the only sensible solution is to modify T. Since we can imagine no automatic way of finding such errors and doing this, we will

ignore this possibility, and assume that T is a well-defined, correct and complete embodiment of R. We can thus replace R by T in (1), and treat T as the standard of correctness below.

There appear to be two possible sources of error in P:

Problem (i): where P is not a correct implementation of T. One would expect this to be common where (as often in MT and NLP) T is very complex, and serious problems arise in devising implementations for them.

Problem (ii): where P is a correct implementation so far as it goes, but is incomplete, so that the domain of P is a proper-subset of the domain of T. This will also be very common: in reality processes are often faced with inputs that violate the expectations implicit in an implementation.

If we disregard hardware errors, low level bugs and such malfunctions as non-termination of P (for which there are well-known solutions), there are three possible manifestations of malfunction. We will discuss them in turn.

case (a): $P(x)=\emptyset$, where $T(x)\neq\emptyset$

i.e. P halts producing $\emptyset$ output for input x, where this is not the intended output. This would be a typical response to unforseen or illformed input, and is the case of process fragility that is most often dealt with.

There are two obvious solutions: (i) to manipulate the input so that it conforms to the expectations implicit in P (cf. the LIFER [8] approach to ellipsis), or to change P itself, modifying (generally relaxing) its expectations (cf. e.g. the approaches of [7], [9], [10] and [11]). If successful, these guarantee that P produces some output for input x. However, there is of course no guarantee that it is correct with respect to T. It may be that P plus the input manipulation process, or P with relaxed expectations is simply a more correct or complete implementation of T, but this will be fortuitous. It is more likely that making P robust in these ways will lead to errors of another kind:

case (b): $P(x)=z$ where z is not a legal output for P according to T (i.e. z is not in the range of T.

Typically, such an error will show itself by malfunctioning in a process that P feeds. Detection of such errors is straightforward: a well-formedness check on the output of P is sufficient. By itself, of course, this will lead to a proliferation of case-(a) errors in P. These can be avoided by a number of methods, in particular: (i) introducing some process to manipulate the output of P to make it well-formed according to T, or (ii) attempting to set up processes that feed on P so that they can use 'abnormal' or 'non-standard' output from P (e.g. partial representations, or complete intermediate representations produced within P, or alternative representations constructed within P which can be more reliably computed than the 'normal' intended output of P (the representational theories of GETA and Eurotra are designed with this in mind: cf. [2], [3], [5], [6], and references there, and see [1] for fuller discussion of these issues). Again, it is conceivable that the result of this may be to produce a robust P that implements T more correctly or completely, but again this will be fortuitous. The most likely result will be robust P will now produce errors of the third type:

case (c): $P(x)=y$, where y is a legal output for P according to T, but is not the intended output according to T. i.e. y is in the range of T, but $y\neq T(x)$.

Suppose both input x and output y of some process are legal objects, it nevertheless does not follow that they have been correctly paired by the process: e.g.in the case of a parsing process, x may be some sentence and y some representation. Obviously, the fact that x and y are legal objects for the parsing process and that y is the output of the parser for input x does not guarantee that y is a correct representation of x. Of course, robust processing should be resistant to this kind of malfunctioning also.

Case-(c) errors are by far the most serious and resistant to solution because they are the hardest to detect, and because in many cases no output is preferable to superficially (misleadingly) well-formed but incorrect output. Notice also that while any process may be subject to this kind of error, making a system robust in response to case-(a) and case-(b) errors will make this class of errors more widespread: we have suggested that the likely result of changing P to make it robust will be that it no longer pairs respresentations in the manner required by T, but since any process that takes the output of P should be set up so as to expect inputs that conform to T (since this is the 'correct' embodiment of R, we have assumed), we can expect that in general making a process robust will lead to cascades of errors. If we assume that a system is resistant to case-(a) and case-(b) errors, then it follows that inputs for which the system has to resort to robust processing will be likely to lead to case-(c) errors.

Moreover, we can expect that making P robust will have made case-(c) errors more difficult to deal with. The likely result of making P robust is that it no longer implements T, but some T' which is distinct from T, and for which assumptions about correctness in relation to R no longer hold. It is obvious that the possibility of detecting case-(c) errors depends on the possibility of distinguishing T from T'. Theoretically, this is unproblematic. However, in a domain such as MT it will be rather unusual for T and T' to exist separately from the processes that implement them. Thus, if we are to have any chance of detecting case-(c) errors, we must be able to clearly distinguish those aspects of a process that relate to 'normal' processing from

those that relate to robust processing. This distinction is not one that is made in most robust systems.

We know of no existing solutions to case-(c) malfunctions. Here we will outline two possible approaches.

To begin with we might consider a partial solution derived from a well-known technique in systems theory: insuring against the effect of faulty components in crucial parts of a system by computing the result for a given input by a number of different routes. For our purposes, the method would consist essentially in implementing the same theory T as a number of distinct processes $P_1,...P_n$, etc. to be run in parallel, comparing outputs and using statistical criteria to determine the correctness of processing. We will call this the 'statistical solution'. (Notice that certain kinds of system architecture make this quite feasible, even given real time constraints).

Clearly, while this should significantly improve the chances that output will be correct, it can provide no guarantee. Moreover, the kind of situation we are considering is more complex than that arising given failure of relatively simple pieces of hardware. In particular, to make this worthwhile, we must be able to ensure that the different Ps are genuinely distinct, and that they are reasonably complete and correct implementations of T, at the very least sufficiently complete and correct that their outputs can be sensibly compared.

Unfortunately, this will be very difficult to ensure, particularly in a field such as MT, where Ts are generally very complex, and (as we have noted) are often not stated separately from the processes that implement them.

The statistical approach is attractive because it seems to provide a simultaneous solution to both the detection and repair of case-(c) errors, and we consider such solutions are certainly worth further consideration. However, realistically, we expect the normal situation to be that it is difficult to produce reasonably correct and compelete distinct implementations, so that we are forced to look for an alternative approach to the detection of case-(c) errors.

It is obvious that reliable detection of (c)-type errors requires the implementation of a relation that pairs representations in exactly the same way as T: the obvious candidate is a process P^{-1}, implementing T^{-1}, the inverse of T.

The basic method here would be to compute an enumeration of the set of all possible inputs W that could have yielded the actual output, given T, and some hypothetical ideal P which correctly implements it. (Again, this is not unrealistic; certain system architectures would allow forward computation to procede while this inverse processing is carried out).

To make this worthwhile would involve two assumptions:

(i) That P^{-1} terminates in reasonable time. This cannot be guaranteed, but the assumption can be rendered more reasonable by observing characteristics of the input, and thus restricting W (e.g. restricting the members of W in relation to the length of the input to P^{-1}).

(ii) That construction of P^{-1} is somehow more straightforward than construction of P, so that P^{-1} is likely to be more reliable (correct and complete) than P. In fact this is not implausible for some applications (e.g. consider the case where P is a parser: it is a widely held idea that generators are easier to build than parsers).

Granted these assumptions, detection of case-(c) errors is straightforward given this 'inverse mapping' approach: one simply examines the enumeration for the actual input if it is present. If it is present, then given that P^{-1} is likely to be more reliable than P, then it is likely that the output of P was T-correct, and hence did not constitute a case-(c) error. At least, the chances of the output of P being correct have been increased. If the input is not present, then it is likely that P has produced a case-(c) error. The response to this will depend on the domain and application -- e.g. on whether incorrect but superficially well-formed output is preferable to no output at all.

In the nature of things, we will ultimately be lead to the original problems of robustness, but now in connection with P^{-1}. For this reason we cannot forsee any complete solution to problems of robustness generally. What we have seen is that solutions to one sort of fragility are normally only partly successful, leading to errors of another kind elsewhere. Clearly, what we have to hope is that each attempt to eliminate a source of error nevertheless leads to a net decrease in the overall number of errors.

On the one hand, this hope is reasonable, since sometimes the faults that give rise to processing errors are actually fixed. But there can be no general guarantee of this, so that it seems clear that merely making systems or processes robust in the ways described provides only a partial solution to the problem of processing errors.

This should not be surprising. Because our primary concern is with automatic error detection and repair, we have assumed throughout that T could be considered a correct and complete embodiment of R. Of course, this is unrealistic, and in fact it is probable that for many processes, at least as many processing errors will arise from the inadequacy of T with respect to R as arise from the inadequacy of P with respect to T. Our pre-theoretical and intuitive ability to relate representations far exceeds our ability to formulate clear theoretical statements about these relations. Given this, it would seem that error free processing depends at least as much on the correctness of theoretical models as the capacity

of a system to take advantage of the techniques described above.

We should emphasise this because it sometimes appears as though techniques for ensuring process robustness might have a wider importance. We assumed above that T was to be regarded as a correct embodiment of R. Suppose this assumption is relaxed, and in addition that (as we have argued is likely to be the case) the robust version of P implements a relation T' which is distinct from T. Now, it could, in principle, turn out that T' is a better embodiment of R than T. It is worth saying that this possiblility is remote, because it is a possibility that seems to be taken seriously elsewhere: almost all the strategies we have mentioned as enhancing process robustness were originally proposed as theoretical devices to increase the adequacy of Ts in relation to Rs (e.g. by providing an account of metaphorical or other 'problematic' usage). There can be no question that apart from improvements of T, such theoretical developments can have the side effect of increasing robustness. But notice that their justification is then not to do with robustness, but with theoretical adequacy. What must be emphasised is that the chances that a modification of a process to enhance robustness (and improve reliability) will also have the effect of improving the quality of its performance are extremely slim. We cannot expect robust processing to produce results which are as good as those that would result from 'ideal' (optimal/non-robust) processing. In fact, we have suggested that existing techniques for ensuring process robustness typically have the effect of changing the theory the process implements, changing the relationship between representations that the system defines in ways which do not preserve the relationship relationship between representations that the designers intended, so that processes that have been made robust by existing methods can be expected to produce output of lower than intended quality.

These remarks are intended to emphasise the importance of clear, complete, and correct theoretical models of the pre-theoretical relationships between the representations involved in systems for which error free 'robust' operation important, and to emphasise the need for approaches to robustness (such as the two we have outlined above) that make it more likely that robust processes will maintain the relationship between representations that the designers of the 'normal/optimal' processes intended. That is, to emphasise the need to detect and repair malfunctions, so as to promote correct processing.

AKNOWLEDGEMENTS

Our debt to the Eurotra project is great: collaboration on this paper developed out of work on Eurotra and has only been possible because of opportunities made available by the project. Some of the ideas in this paper were first aired in Eurotra report ETL-3 ([4]), and in a paper presented at the Cranfield conference on MT earlier this year. We would like to thank all our friends and colleagues in the project and our institutions. The views (and, in particular, the errors) in this paper are our own responsibility, and should not be interpreted as 'official' Eurotra doctrine.

REFERENCES

1. ARNOLD, D.J. & JOHNSON, R. (1984) "Approaches to Robust Processing in Machine Translation" Cognitive Studies Memo, University of Essex.

2. BOITET, CH. (1984) "Research and Development on MT and Related Techniques at Grenoble University" paper presented at Lugano MT tutorial April 1984.

3. BOITET, CH. & NEDOBEJKINE, N. (1980) "Russian-French at GETA: an outline of method and a detailed example" RR 219, GETA, Grenoble.

4. COLBY, K. (1975) Artificial Paranoia Pergamon Press, Oxford.

5. ETL-1-NL/B "Transfer (Taxonomy, Safety Nets, Strategy), Report by the Belgo-Dutch Eurotra Group, August 1983.

6. ETL-3 Final 'Trio' Report by the Eurotra Central Linguistics Team (Arnold, Jaspaert, Des Tombe), February 1984.

7. HAYES, P.J. and MOURADIAN, G.V. (1981): "Flexible parsing", AJCL 7, 4:232-242.

8. HENDRIX, G.G. (1977) "Human Engineering for Applied Natural Language Processing" Proc. 5th IJCAI, 183-191, MIT Press.

9. KWASNY, S.C. and SONDHEIMER, N.K. (1981): "Relaxation Techniques for Parsing Grammatically Ill-formed Input in Natural Language Understanding Systems". AJCL 7, 2:99-108.

10. WEISCHEDEL, R.M, and BLACK, J. (1980) 'Responding Intelligently to Unparsable Inputs' AJCL 6.2: 97-109.

11. WILKS, Y. (1975): "A Preferential Pattern Matching Semantics for Natural Language". A.I. 6:53-74.