A MODEL FOR PREFERENCE

Dominique Petitpierre

ISSCO
University of Geneva
54 route des Acacias
CH-1227 Geneva, Switzerland

Steven Krauwer
Louis des Tombe

Instituut voor Algemene Taalwetenschap
Rijksuniversiteit Utrecht
Trans 14
3512 JK Utrecht, The Netherlands

Doug Arnold

Centre for Cognitive Studies
University of Essex
Colchester, CO4 3SQ, England

Giovanni B. Varile

DG XIII, Batiment Jean Monnet
Commission of the European communities
P.O. Box 1907, Luxembourg, Luxembourg

Abstract

In this paper we address the problem of choosing the best solution(s) from a set of interpretations of the same object (in our case a segment of text). A notion of preference is stated, based on pairwise comparisons of complete interpretations in order to obtain a partial order among the competing interpretations. An experimental implementation is described, which uses Prolog-like preference statements.

1. Introduction

In this paper we address the problem of choosing the best solution(s) from a set of interpretations of the same text segment (For the sake of brevity, throughout this text we use the term interpretation, where in fact we should write representation of an interpretation). Although developed in the context of a machine translation system (the Eurotra project, Arnold 1986, Arnold and des Tombe 1987), we believe that our approach is suited to many other fields of computational linguistics and even outside (pattern recognition, etc.).

After a brief overview of the problem (section 2), we suggest a general method to deal with preference (section 3) and then describe a possible implementation (section 4). An appendix gives actual examples of preference statements.

2. What is preference?

In the computational linguistics literature, the term 'preference' has been used in different contexts. We shall mention a few, selectively, (in section 2.1 which may be skipped) and then state our own view (in section 2.2).

2.1. Various approaches

Preference strategies have often been used for dealing with the problem of ill-formed input (a particular case of robustness, cf below section 2.2) (AJCL 1983, Charniak 1983). Following Weischedel and Sondheimer (1983) we distinguish the cases where preference is part of the particular computation being performed (Wilks 1973, Fass and Wilks 1983, Pereira 1985) from the case where it is a separate process, run after the results of the computation have been obtained (Jensen et al 1983, Weischedel and Sondheimer 1983).

A frequent approach to preference is scoring. A numeric score is calculated, independently, for each competing interpretation and is then used to rank the interpretations. The best interpretations are then chosen. The score can be the number of constraints satisfied by the interpretation (Wilks 1973, Fass & Wilks 1983), where these constraints might be assigned relative weights by the linguist (Robinson 1982, Charniak 1983, Bennett and Slocum 1985) or calculated by the computer (Papegaaij 1986). Such techniques have been used extensively for speech recognition (Paxton 1977, Walker et al 1978) and in the field of expert systems (such as Mycin, Buchanan & Shortliffe 1984), where the calculation of both score and ranking become quite complex with probabilities and thresholds.

The problem with scoring is that it seems quite unnatural for a linguist to associate a score (or weight or probability) to a particular rule or piece of data when the knowledge being encoded is in fact qualitative. Furthermore, combining the scores based on different types of reasoning to calculate a global score for a representation seems a rather arbitrary procedure. Such a uniform metric, even if it can model actual linguistic knowledge, forces the grammar writer to juggle with numbers to get the behaviour he wants, thus making the preference process obscure.
A further disadvantage of this approach is that the score is often based on the way interpretations are built, rather than on the properties of the interpretations themselves.

Preference is also mentioned in a linguistic controversy started by Frazier and Fodor (1979) with their principles of right association and minimal attachment (Schubert 1984). There the problem is to disambiguate many readings (or interpretations) of a sentence in order to find the good (preferred) one(s). Various contributions on that issue have in common that bad interpretations are abandoned before being finished, during computation (Shieber 1983, Pereira 1985). Although this method speeds up the computation, there is a risk that a possiblity will be abandoned too early, before the relevant information has been found. This is shown by Wilks et al (1985) who claim to have the ideal solution in Preference Semantics, which uses as part of its computation scoring and ranking.

2.2. Our notion of preference

Our approach, although stemming from earlier work in the Eurotra project (McNaught et al 1983, Johnson et al 1985), is, we believe, new and original.

We make the following assumptions:

i the relation 'translation of' between texts as established by a machine translation system has to be one to one (1-1)?

ii There is apriori no formal or linguistic guarantee that this will be the case for the relation as a whole or for the translation steps between intermediate levels of representation. (An attempt to formalize this can be found in Krauwer and des Tombe 1984 or in section 4 of Johnson et al 1985).

The problem we want to address here is the following:
Given the fact that one to many (1-n) translations do occur, how do we ensure that the final result is still 1-1.

This problem is not restricted to machine translation:
Often a program (for example a parser or a text generator) produces many interpretations of the same object (usually a text segment) when in the ideal case only one is wanted. In the following we refer to a '1-n translation' for this general phenomenon.

We see two types of solutions to this problem, each of them applicable to specific classes of cases:

i Spurious results can be eliminated on the basis of their own individual properties (e.g. well-formedness, completeness); for this we will use the term 'filtering'.

ii Spurious results can be eliminated via comparison of competing representations, where only the best one(s) will have the right to survive; for this we will use the term 'preference'.

It is important to note that we restrict ourselves to reducing 1-n translations to (ideally) 1-1. We will assume that the 'good' translation is one of the candidates. The problem of forcing the system to come up with at least 1 translation (i.e. do something about possible 1-0 cases) will not be addressed here. In order to avoid confusion we will use the term 'robustness' to refer to this type of problem. We are aware of the fact that we deviate slightly from the standard use of the term preference.

There are two main types of 1-n -ness:

i linguistically motivated (i.e. real ambiguity in analysis, or true synonymy in generation).

ii accidental, caused by overgeneration of the descriptive devices that define the resulting (or intermediate) interpretations.

Note that overgeneration and ambiguity or synonymy may hide cases of undergeneration (cf the robustness problem).

We define the application of preference as the selection of the best element(s) from a set of competing interpretations of the same object.

According to this definition the scoring and ranking mechanism described in the previous section is a case of preference.

In the rest of this paper we will describe a preference device that is different from the scoring and ranking mechanism in the sense that it is not based on the way interpretations are built, but rather on linguistic properties of the objects themselves. Its main characteristics are that:

i it applies to complete and sound (well formed) interpretations only. That is, all the other modules of construction, transformation and filtering have been applied (Ex: parsing, Wh-movement, etc). Thus, for these modules all competing representations are equivalent, and all the information needed for comparing them has been found.

ii it is based on pairwise comparison between alternative (competing) interpretations of the same object.

The problem can then be stated as follows:
How do we make use of the linguistic knowledge in order to insure a 1-1 translation?
It is our basic belief that it is impossible for the linguist to know the exact nature of a class of competing interpretations in advance. This implies that he cannot in general formulate one single rule that picks out the best one.

3. The proposed method

3.1. Basic idea

Our proposal is the following:

- It should be possible to make (linguistic) statements of the type: if representation A has property X, and B property Y, then A is to be preferred over B (e.g. 'in law texts declarative sentences are better than questions', or 'sentences with a main verb are better than sentences without one').

- On the basis of a set of such statements it should be possible to establish a partial order over the set of competing representations.

- And in that case the number of candidates can be reduced by, for example, letting only the maximal elements survive, or discarding the minimal ones.

3.2. Problems with the method

The first (but least serious) problem is that it is not certain that linguists will always be able to make such statements (we will call them 'preference statements') over pairs of representations. Experimentation is necessary.

The second one is more serious: it would be highly unrealistic to expect that the result of applying of the preference statements will be a linear order, in fact there is not even a guarantee that the order will be partial. In general the outcome will be a directed graph. There are three ways of tackling this problem:

i The linguist should try to make the set of preference statements homogeneous and constrained, and should have control over the way in which they are applied, so that he can avoid contradictory statements.

ii One tries to make a formal device that checks whether contradictions can occur.

iii One tries to compare pairs of competitors in a specific order such that it can be guaranteed that the result is always a partial order.

At the moment (iii) is the most feasible, (ii) the most ambitious, and (i) the most desirable solution. Currently we envisage a combination of (i) and (iii).

The third problem is that of the maximal elements. Ideally there would be just one maximal element, i.e. the preferred representation. This cannot be guaranteed to be true.

The problems sketched here are by no means trivial. That is why we want to experiment with a first implementation of this method, to identify the various relevant parameters in the specific context of Eurotra.

4. The proposed implementation

The implementation proposed here is described in very general terms, and can be adapted for a wide range of applications. We give in the appendix some commented examples specific to our particular context.

4.1. Preference rules

Preference statements are expressed by the user in the form of rules (preference rules). There are three types of preference rules: simple rules, predefined rules and composite rules. A preference rule applied to two representations of interpretation tries to decide which one is better than the other (preferred to the other). It is not guaranteed that a rule can always take a decision.

A simple preference rule is of the form

```
 p = (Pattern1 > Pattern2)
```

The name of the rule is p, and Pattern1 and Pattern2 are current patterns. When given two arguments (two representations or subparts) A and B (written p(A,B)) the system will try to match Pattern1 with A and Pattern2 with B. If this succeeds then A is better than B (or A is preferred to B or A>B). If it fails then the system will try to match A with Pattern2 and B with Pattern1. If this succeeds then B is better than A.

Predefined rules are provided for the cases where simple rules cannot express some useful basic preference statement. For example, in our actual implementation (cf appendix), two predefined rules say that a tree structure with fewer (more) branches than the other is to be preferred to one with more (fewer) branches. This cannot be expressed with the particular language for patterns.

A composite preference rule is of the form

```
 p = (Pattern1,Pattern2)
     => (p1($V,$W),
         p2($X,$Y),
         ...)
```

Identifiers p, p1, p2, ... are rule names, Pattern1 and Pattern2 are actual patterns, and $V, $W, $X, $Y, ... are variable identifiers, that should also occur in Pattern1 ($V,$X) and Pattern2 ($W,$Y) where they identify sub-parts of the interpretations. When given two arguments A and B, the system tries to match A with Pattern1 and B with Pattern2. If this succeeds, the variables $V,$X,.. occurring in Pattern1 and $W,$Y,... occurring in Pattern2 are instantiated to sub-parts of A and B respectively. Then the system tries each preference rule of the list, with the instantiated arguments, till one rule can decide. In this case the relationship holding between A and B is the same as that holding between the sub-part of A and the sub-part of B. If no rule of the list can decide then preference is not decided. If the initial match doesn't succeed, then an attempt will be made to match A with Pattern2 and B with Pattern1. If this succeeds the system tries the rules of the list in the same way as above. Composite preference rules allow recursion.

This formalism is very much inspired by the programming language Prolog: a preference rule is analogous to a three argument predicate (two interpretations and the resulting relationship), a simple rule to an assertion, and a composite rule to a clause with sub-goals.

4.2. General algorithm

Initially, all competing objects are in the set of non ordered objects N and the set of ordered objects O is empty. Then, the following is repeated until N is empty: an object is removed from N and is compared to each object of O (if any), then it is added to O.

This algorithm does not ensure that the resulting directed graph of preference relationships among the competing objects has no cycle. Anyway, maximal (minimal) elements can be defined in the following way:

> An object E is a maximal (minimal) element if no competing object is better (worse) than E.

Thus an object in a cycle of the graph cannot be maximal (minimal).

To give the user control of how rules are tried on the competing objects, only one distinguished rule is applied to each competing pair. In the general case it should be a composite rule that just passes its two arguments to the rules of the list, thus ensuring that only these rules are tried and in that order.

The pattern matching mechanism of composite rules is quite powerful. (see also the appendix): It allows some preferences rule to be applied only to selected objects (satisfying a precondition). It also allows (recursive) exploration of sub-parts of representations (a derivation tree for example), in parallel or not. Finally it enables the user to give priority to some preference rules over some others.

4.3. Problems with the implementation

Although we decided that this model is good enough for preliminary experimentation, certain problems are already apparent:

- The system takes arbitrary decisions in the case of a contradiction, that is if

some rule can be applied to a pair of arguments in both orders (if p(A,B) and p(B,A) are both possible). In particular a preference decision should not be taken between identical objects.
- Infinite recursion can occur with composite preference rules.
- Maximal (minimal) elements may not exist in the resulting graph of preference relationships (for example if all elements are in a cycle).
- Arbitrary decisions may be taken if the patterns allow multiple matches: the current model will stop with the first match that produces a decision.

Currently it is the user's responsibility to avoid these problems by writing "sensible" rules. In the next section we sketch some possible solutions that are considered for a future implementation.

5. Future directions

The implementation of this preference model has been written in Prolog. To facilitate experimentation, a mechanism is provided for tracing the preference rules application to observe their behaviour.

The model described above is very flexible. We are currently studying the implementation of variants of the basic comparison algorithm:

We are investigating algorithms that would:
- reduce the number of comparisons, by aiming at extracting only the maximal (minimal) elements, without trying to order all elements.
- calculate the transitive closure of the directed graph, and then remove all contradictory relationships, thereby removing all cycles. This amounts to saying that two interpretations are not comparable if their comparison leads to contradictory decisions.
- compare the competing interpretations stepwise, that is all comparisons are performed with the first rule in a list, then only the pairs for which there is no decision yet are compared with the second rule, and so on.

ACKNOWLEDGEMENTS

We would like to thank Paul Bennett, Maghi King, Gertjan Van Noord, Mike Rosner and Susan Warwick for their fruitful comments and their support.

APPENDIX

In the current framework of EUROTRA (Arnold and des Tombe 1987), representation of interpretations are derivation trees, containing at each node a set of attribute-value pairs. Here is a very sketchy and intuitive description of the syntax used in the patterns:
- The identifiers s, np, vp etc. are values of the distinguished attribute of the node (in these examples, the syntactic category).
- Curly brackets delimit a set of conditions to be satisfied by a node. For example {s,f=declarative} indicate the required conditions on the node for the distinguished attribute (should have value s) and for an f attribute (should have value declarative).
- $A, $B, etc. are variable identifiers.
- s.[np,vp] indicates a tree with root s and two daughters np and vp.
- ? or (?) indicates an unspecified node.
- * indicates a list of unspecified nodes.
- $A!Pattern indicates that the variable $A is instantiated to the sub-tree that matches Pattern
- $more_branches (and $less_branches) is a predefined preference rule that prefer the argument that has more (less) branches than the other.
- The first rule declared becomes the distinguished rule applied to the competing interpretations.

Example 1

```
p0 = ($A!(?),$B!(?)
      => (p1($A,$B),
          p2($A,$B)),

p1 = ({s,f=declarative}
     > {s,f=interrogative}),

p2 = (s.[np,v,$A!s,*],
      s.[np,v,$B!s,*])
      => (p1($A,$B),
          p2($A,$B)) .
```

This set of preference rules will explore, in parallel, two trees, from top to bottom, always taking the 's' branch, and prefer the tree in which it finds a declarative sentence (opposed to an interrogative).If one inverts the order of p1 and p2 in the distinguished composite rule p0 the trees would be explored from bottom to top.

Rule p0 just passes its arguments to p1 or p2.
Rule p1 prefers a declarative s over an interrogative s.
Rule p2 identifies the embedded s in each argument and passes them to p1 or p2.

Example 2

```
p0 = (s.[np,vp.[*,$A!(?)]],
      s.[np,vp.[*,$B!(?)]])
     => (p1($A,$B),
         p2($A,$B),
         p3($A,$B)),

p1 = (np.[*,pp] > pp),
p2 = (np.[*,$A!np] , $B!pp)
     => (p1($A,$B),
         p2($A,$B),
         p3($A,$B)),

p3 = (np.[*,$A!(?)],
      np.[*,$B!(?)])
     => (p1($A,$B),
         p2($A,$B),
         p3($A,$B)).
```

Given two sentences, this set of rules will prefer the one that has the pp attached deeper in the structure than the other (right attachment). This example is restricted to explore only embedded nps.

For both arguments, rule p0 identifies the last daughters of the vp of a sentence s, and passes them to preference rules p1 or p2 or p3.
Rule p1 will prefer a pp attached under an np to a pp (which was attached higher in the structure).
Rule p2 will be tried only if p1 was not applicable. It is there for the case the pp is imbedded deeper in the np.
Rule p3 is similar to rule p0, except that it takes the last daughters of a np. It is tried only if p1 and p2 are not applicable.

REFERENCES

AJCL. 1983 Special issue on ill-formed input. American journal of computational linguistics 9(3-4).

Arnold, Doug. 1986 Eurotra: A European Perspective On Machine Translation. Proceedings of the IEEE 74(7): 979-992.

Arnold, Doug and des Tombe, Louis. 1987 Basic Theory and Methodology in EUROTRA. In: Nirenburg, Sergei, Ed., Machine Translation. Cambridge University Press, Cambridge, England: 114-135.

Bennett, Winfield S. and Slocum, Jonathan. 1985 The LRC machine Translation System. Computational linguistics 11(2-3): 111-121.

Buchanan, Bruce G. and Shortliffe, Edward H. 1984 Rule-based Expert Systems. Addison Wesley, Reading, Massachusetts.

Charniak, Eugene. 1983 A Parser With Something for Everyone. In: King, Margaret, Ed., Parsing Natural Language. Academic Press, London, England: 117-149.

Fass, Dan and Wilks, Yorick. 1983 Preference Semantics, Ill-Formedness, and Metaphor. American journal of computational linguistics 9(3-4): 178-187.

Frazier, Lyn and Fodor, Janet D. 1978 The Sausage Machine: A New Two-Stage Parsing Model. Cognition 6: 291-325.

Jensen, K.; Heidorn, G. E.; Miller, L. A. and Ravin, Y. 1983 Parse Fitting and Prose Fixing: Getting a Hold on Ill-Formedness. American journal of computational linguistics 9(3-4): 147-160.

Johnson, Rod; King, Margaret and des Tombe, Louis. 1985 EUROTRA: A Multilingual System Under Development. Computational linguistics 11(2-3): 155-169.

Krauwer, Steven and des Tombe, Louis. 1984 Transfer in a Multilingual Machine Translation System. In: Proceedings of Coling84, Stanford, California: 464-467.

Mc Naught, Jock; Arnold, Doug; Bennett, Paul; Fass, Dan; Grover, Claire; Huang, Xiuming; Johnson, Rod; Somers, Harry; Whitelock, Pete and Wilks, Yorick 1983 Structure, Strategies and Taxonomy. Eurotra contract report ETL-1, Commission of the European Communities, Luxembourg, Luxembourg.

Papegaaij, Bart; Sadler, Victor and Witkam, Toon. 1986 Word Expert Semantics; an Interlingual Knowledge Based Approach. Foris, Dordrecht, Holland.

Paxton, W.H. 1977 A Framework for Speech Understanding. Ph.D. Dissertation, Stanford University, Stanford, California.

Pereira, Fernando C. 1985 A New Characterization of Attachment Preferences. In: Dowty, David R.; Kartunnen, Lauri and Zwicky, Arnold M., Eds., Natural language parsing. Cambridge University Press, Cambridge, England: 307-319.

Robinson, Jane J. 1982 DIAGRAM: A Grammar for Dialogues. Communications of the ACM 25(1): 27-47.

Schubert, Lenhart K. 1984 On Parsing Preferences. In: Proceedings of COLING84 Stanford, California: 247-250.

Shieber, Stuart. 1983 Sentence Disambiguation by a Shift-Reduce Parsing Technique. In: Proceedings of IJCAI-83 Karlsruhe, West Germany: 699-703.

Walker, D.E., Ed., 1978 Understanding Spoken Language. North Holland, New York, New York.

Weischedel, Ralph M. and Sondheimer, Norman K. 1983 Meta-rules as a Basis for Processing Ill-Formed Input. American journal of computational linguistics 9(3-4): 161-177.

Wilks, Yorick. 1973 An Artificial Intelligence Approach to Machine Translation, In: Schank, Roger C. and Colby, Mark Kenneth, Eds., Computer Models of Thought and Language. W.H. Freeman and Co, San Francisco, California: 114-151.

Wilks, Yorick; Huang, Xiuming and Fass Dan. 1985 Syntax, Preference and Right Attachment. MCCS-85-5, July 1985, Computing Research Laboratory, New Mexico State University.