LINGUISTICS AND THE MACHINE TRANSLATION OF NATURAL LANGUAGE TEXTS

by Richard Hudson, University College London

INTRODUCTION

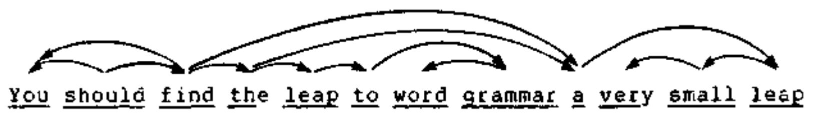
I remember hearing about the promise of MT in the days before the ALPAC report came along like the bad fairy in The Sleeping Beauty and condemned it to sleep for ten years until some Prince (who I didn't quite manage to identify) came and woke her/it/them. And now there are systems which are commercially viable, and the EEC is using MT on a large scale, and so on.

My aim is to sell linguistic theory to you; but which linguistic theory? You may well know that the discipline of linguistics includes some fairly important disputes about fundamental theoretical matters, and if you have a notion of what 'linguists' say about certain matters, it is probably only what certain linguists say. From the odd references in the newletters to trees I suspect that at least some of you think that linguistics = phrase-structure grammar, or a particular version of PSG, transformational grammar. This is a misconception, and I should like to correct it by introducing you to a linguistic theory which I think you will find much more appealing and directly relevant to your work than either PSG or TG. It is called 'word grammar'.

WORD GRAMMAR

I call it 'word grammar' because the word is the central unit - in fact, it's the only unit of which you would need to take account, apart from the letters in which words are written and various affixes you may identify within words. In particular, you need pay no attention to phrases or clauses - no trees, in other words. When your computer is working out the relations among the words in a sentence, that is literally all it has to do: it decides how one word is related to another word, without setting up larger units to mediate these relations. In relation to most modern theories of linguistics, this is a heretical position to take, but in fact it is just a return to the two-thousand year old grammatical tradition enshrined in most school grammars (and also in various European branches of what is called 'dependency theory', which flourishes in particular in West Germany and the USSR). So if you learned some grammar at school, and if this is what you have been using to guide your thinking in MT, you should find the leap to word grammar a very small leap.

Let me give you an example of what I mean by 'the relations between words'. The primary relation of syntax, according to word grammar, is dependency - the relation between a 'head' word and its 'modifier'. For example, Ian Pigott (Newsletter 10, 1981) lists the following word-pairs: subject-verb, verb-object, noun modifier-noun modified, adjective-noun, adverb-verb, preposition-object. These are all examples of modifier-head pairs, though the decision as to which is head and which modifier is a theoretical question which may lead to unexpected answers in some cases. In the diagram below, I take the last part of the last sentence in the previous paragraph, and show the dependency relations among its words. The arrows point from the head to the modifier in each modifier-head pair.



What you should be interested in is whether the structures generated reflect the actual structure of the sentence - in other words whether they reflect the structures which a human mind constructs in processing the sentence. If they do, then it should be easy to relate them to the sentence's semantic structure, because that is presumably how human processors arrive at the semantic structure. I believe this is the case for the structure given above for the example. For instance, you is connected both to should and to find. The latter connection is essential for identifying 'you' as the person in whose head the opinion (or 'finding') is located, but the connection between you and should is needed in order to link you to find (because the subject of should is always the same as the subject of the following infinitive).

You have a choice of notations for expressing the same information, and the notation I tend to use when working out bits of the grammar would be much easier for you to use. It is based on formulaic statements, which are formulated in terms of a notation which I think someone once told me is similar to the notation of LISP. Be that as it may, I hope it would be easy to implement in some computer language. If you want to say that word *n* is a modifier of word *m*, then the formula you need is: modifier(m): n. More generally, each formula consists of a labelled slot (e.g. 'modifier'), then an entity in brackets which identifies the 'owner' of the property denoted by the label, and which can be put into prose as 'of ....'; then comes a colon, to be verbalised as 'is', and lastly some entity (or Ø, standing for zero), which is the filler of the slot. This notation can be used not only in describing the structure of a sentence which the grammar generates, but also for formulating the rules of the grammar. The only difference between the two is that the former are more specific in their reference than the latter. For instance, the word *leap* occurs twice in our example string, but in each case it is an instance of the same general entity in the grammar, the word (or lexical item) *leap*. Naturally we have to distinguish these different entities from one another, so I use numbers as names for the words in a sentence, and the ordinary spelling, underlined, for the word as an entity in the grammar. Before I show how we can use formulae instead of the arrows of the dependency structure, let me deal with a more straightforward matter: the internal composition, in terms of letters, of the words. In the grammar we find this formula:

(1) composition(*leap*): *l* *e* *a* *p*

And in the sentence structure generated, we find this, in which 5 stands for the first instance of *leap*:

(2) composition(5): *l* *e* *a* *p*.

This pair of formulae shows how the structures of grammar and of sentence-structure are different only in specificity.

The way in which the human user of a word grammar makes use of the information in the grammar when processing a sentence is to find at least one model in the grammar for each word in the sentence. All this requires is a direct comparison between the known properties of the word in the sentence, and the listed properties of words in the grammar. Except for homographs, this will take us straight from the spelling of a word to a unique word in the grammar; but in the case of homographs, we need to take account of other known properties of the sentence-word, such as the kind of word we should expect it to be on the basis of the words before it. For example, our first leap occurs straight after the, and as soon as we see this word we start looking for a noun, so the easiest assumption is that leap is itself a noun. We have found a word in the sentence whose composition is l e a p, so we look through the grammar for a word whose composition is also the same; and if we find more than one, we take account of more and more known information until we can eliminate all but one of them. This is the 'model' for our word in the sentence, so we can now add a further bit of information to the analysis of the sentence.

(3) model(5): leap.

But by adding formula (3) to (2), which is the input to the system, we are opening the flood-gates for a whole collection of other formulae, namely all the formulae which involve the grammar-word leap. In other words, we start from the known properties of an entity, then we find its model in our stored knowledge, and then we assume that it has all the properties of this stored entity, including many which we cannot 'know' in any other way. In the case of leap, this means that we gain access to its meaning, which we can represent simply as LEAP for the time being, and to its syntactic classification as a noun. So we add formulae (4) and (5) to the two we already have for word 5:

(4) referent(5): LEAP

(5) model(5): noun.

The two formulae are simply copied directly from the formulae containing leap, with the substitution of 5 for leap.

I said that the only difference between the formulae found in grammars and in sentence-structures is in the specificity of the entities to which they refer; this point has already been illustrated, because the word 5 is by definition more specific than the word leap, because the former takes the latter as its model - in other words, the former is an instance of the latter. The entities referred to in a grammar-formula can be quite vaguely specified, and typically they are represented by some variable, which may or may not have conditions attached to it. For instance, we can improve on the formula given in (4) for the meaning of <u>leap</u>, by representing the referent of the word simply as leap* (a useful convention, as we shall see), and then requiring it to be an instance of the general concept LEAP. We simply tag conditions onto the formula, with a comma before them, so the formula for the meaning of the grammar-entity <u>leap</u> would be like this:

(6) referent(<u>leap</u>): leap*, model(leap*):LEAP

This tells us that if the word <u>leap</u> is used, the thing to which it refers must be an instance of a leap, but of course it does not tell us what particular instance of leaping the word refers to, because <u>leap</u> is a common noun, not a proper noun.

This last formula is the beginnings of a semantic structure for <u>leap</u>, and a fuller analysis would include an analysis of the structure of the concept LEAP. The question of how much of this structure to analyse is an entirely pragmatic one, as I happen to be one of those linguists who finds no natural boundary around the 'truly linguistic' - in our minds, the linguistic meshes inextricably with our total knowledge of the world. For instance, you may find it helpful to take the analysis up to the model for LEAP, which is (arguably) MOVEMENT. This much analysis would be helpful in eliminating some ambiguities, but more importantly it would allow us to explain the semantic connections between words, as I shall explain shortly. For example, it is probably important to show that in the phrase <u>the leap to word grammar</u>, <u>to word grammar</u> expresses the direction of (metaphorical, or mental) movement, in order to select the right translation equivalent in the target language (in contrast, say, with <u>the key to the door</u>).

We can now pick up again the question of analysing the relations between the words in a sentence. I take it that the bit of the grammar which deals with <u>leap</u> as such gives no specific information about its possible relations to other words, because it is just a typical noun, and follows all the normal rules for using nouns in relation to other words. So having identified <u>leap</u> as the model for word 5, we haven't actually learned anything directly about its relations to other words in our sentence; but we have been able to add formula (5), model(5): noun, because 5 simply inherits all the properties of <u>leap</u>, and <u>leap</u> is an instance of 'noun'. Consequently word 5 also inherits all the properties of 'noun', as well as those of <u>leap</u>, including all the information which grammars tend to include in their 'rules', in contrast with the lexical items which are in the other part of the grammar, called the 'lexicon'.

Linguists use 'grammar' to include the dictionary or lexicon of the language concerned, as well as the general rules. However, there is an important matter of principle at stake as well, because word grammar uses just the same format - again, formulae of the kind I have described - for expressing information about specific words ('lexical' information) and for expressing more general items of information which apply to whole classes of words, and which are often called 'rules'. This blurring of the distinction between lexical items and rules allows the operation of a language-processor to be very simple, and to involve nothing but 'instantiation' - i.e. the identification and exploitation of models. Just as we took <u>leap</u> as a model for word 5, and added all the stored properties of <u>leap</u> to the known (i.e. observed) properties of word 5, so we can take 'noun' as the model for <u>leap</u>, and add to the latter all the stored properties for nouns in general. The same process extends to semantics as well - we take the referent of 5, represented as 5*, as an instance of the referent of <u>leap</u>, leap*, which in turn is an instance of LEAP (and inherits any stored properties of LEAP); and LEAP is itself an instance of MOVEMENT, so it automatically inherits the latter's stored properties. I take it that this uniform mode of operation for a language processor is an attraction for a computational linguist.

What kind of information, then, should a grammar give about 'noun'? Before we come to information about syntactic relations to other words, I should mention that information about regular inflections would be attached to more general entities, such as 'noun' or 'plural noun'. I know that inflections are of great practical interest to machine-translators. The other main kind of information is about relations to other words, and can all be given in terms of dependency relations, using the terms 'head' and 'modifier' as I explained them earlier.

The most useful piece of information which is made available at this level is probably that every noun needs a head. (I take it that cases like chapter-headings, lists and captions can be coped with by some extra condition.) So as soon as you encounter a noun, you can start looking for its head. If you are lucky, you will find that on your 'work-space' you will already have some word which needs a noun as its modifier, and you can introduce the two words to one another and satisfy them both. For example, the verb *find* requires at least one modifier after it, and one of the things which this modifier is allowed to be (by the specific requirements of *find*) is a noun; so when you meet the word *leap*, and take it as a noun, you immediately have a potential head for it. (Actually, the analysis takes *the* as head of *leap*, but *the* is a noun according to my grammar, and it takes a noun as its modifier, so the same principle applies.) However, you may find a noun without already having a potential head, as with *you* at the start of the string *you should find* .... In this case, you keep a note of the need for a head for this word, and go on processing. If you come to a full-stop, and still haven't found a head for your noun, then you know you must have made a mistake, and you or your computer has to go back and try another analysis.

Anotner piece of information about nouns is that they may take any number of words such as prepositions after them, subject to the condition that each such word must make some contribution to the semantic structure. What kind of contribution this can be will depend on the semantic structure of the particular noun concerned; for example, a preposition could express the direction of movement if the noun is one like *leap*, but not if it is the name of an object such as *sausage*. The formula for this is:

(7) modifier β(noun): x, model(x):...or preposition, X(...noun*): x*.

In words, any of the n modifiers of a noun (where 0<n) is some word x, whose model may be 'preposition' (inter alia), and whose referent x* fills some slot X in the semantic structure of the referent of the noun itself, noun*. Thus, whenever you find a noun, you can tentatively open up at least one modifier slot, in case you find a potential filler for it later in the sentence; but of course if you reach a full-stop without finding one, you simply close the slots, rather than assuming that you have maoe a mistake. In the case of

leap to word grammar, you can immediately fill the modifier slot by to, and start working on the latter's semantics but some sentences need a modifier slot to be filled after considerable delay (e.g. Examples are not hard to find of the kind of sentence I have in mind.).

Thus, by putting together the various bits of information relevant to inter-word relations in the grammar, we can arrive at a coherent dependency structure for a string of words. Some of this information comes from the general entries in the grammar relevant to entities like 'noun', some of it comes from specific entries for particular lexical items, such as find; some of it is expressed in terms of syntax, some in terms of semantics, and some involves the relations between syntax and semantics. What the grammar-user has to do is to juggle the various bits of information which are potentially relevant in order to find a way of making them fit together. Of course, it is possible that this juggling trick should be performed in a different way by the human and the computer; for example, it is likely that the human will be able to muster much more contextual and real-world information than computers will be able to for some time to come, so computers may be able to make up for this by moving backwards and forwards in the sentence building up specifically syntactic structures before they start exploiting semantic structures. I have the impression that this is already happening.

The formulae for the first occurrence of leap in our example string would thus be as follows:

(8) head(5): 3

(9) modifier(5): 6

(10) direction(5*): 6*

These formulae link word 5 to word 3 (find) and 6 (to), and similar formulae would be responsible for handling all the other inter-word links which are shown in my earlier diagram. As I said there, these links provide an important step from the uninterpreted string of words to a usable semantic structure.

What I have to offer is a theory of grammar, which has already been applied in some detail to English, so the chore of working up all the missing details would take time and a certain amount of descriptive skill, but shouldn't raise many major theoretical problems. In this theory, a grammar is all about words, and consists of a large number of formulae, each of which expresses some proposition to do with some word or type of word. It is a completely static description of the structures that words can have when they occur in strings; it doesn't contain any specific recommendations for finding the structure of a particular string, but the structure of the grammar is based on the relations between instances and their models, which is precisely the relation which you are seeking when you are trying to interpret a string of words.

Consequently I think it is quite reasonable to think that word grammar should be a good basis for the decoding activities of a machine-translator. You know the meaning, so you look for a word with the right meaning, and take that as a model for the word you need; and the rest of the information, about its spelling and its syntax, will be supplied by the formulae relevant to this word and to its models. No doubt I'm naively missing some fundamental problems which computational linguists will all very kindly point out to me; but that's the obvious way to set about MT, and I can see nothing at least in my experience as a linguist which would suggest that it was the wrong way.

REFERENCES

Hudson, R A (1982) Word grammar. Preprints of the Plenary Sessions of the 13th International Linguistics Congress, Tokyo.

------ (1983) Towards a cognitive linguistics. Working Papers of the London Psycholinguistics Research Group, 5

------ (1984) Word Grammar. Oxford: Blackwell