

Portability of Natural Language Systems

Peter Bosch
Institute for Knowledge-Based Systems
Scientific Centre
IBM Deutschland GmbH
Postfach 80 08 80
7000 Stuttgart 80

One major problem for the processing of human language on the computer is that the general public as well as industrial management and funding agencies do not fully understand the current limitations of natural language processing (NLP). This tends to result in frustrations for the researchers as well as for everybody else. The non-specialist is often lead to generalize too quickly from smart demonstrations and easily overestimates current technology. I don't mean to suggest that anybody intentionally misleads or deceives anybody else. The fact of the matter rather is that the functioning of natural language is a good deal more complicated than is generally appreciated.

In this paper I want to make a small contribution to help the non-specialist, including colleagues from neighbouring disciplines, to understand where we stand in the machine processing of natural language, and what will and what will not be possible in the coming period.

0 What's the problem?

Learning a language means to learn the grammar and the vocabulary - this at least is the common sense theory of learning foreign languages. And probably most people would come up with a theory very much like this one at first blush.

Now if this theory is true, what then is the problem in building natural language systems on the computer? And why do computational linguists and AI researchers claim that a natural language system they have built, say, for hotel reservation is useless for other applications, such as making airline reservations? If the language is the same, then, one ought to think, at least the linguistic modules of the system should be the same. - This paper will try to explain some of the reasons why this view is wrong.

We shall start by looking at computer programs that process natural language on a very superficial level, word processors, and we shall discuss some the current inadequacies of these systems as well as the deeper reasons for these inadequacies. Then we shall look at the LILOG text understanding system as an example of an advanced research system and shall particularly look at its limitations in order to see where the problems of natural language systems lie and what progress can be expected in the coming years.

1 What there is

1.1 Non-linguistic devices

There is a good variety of very good reasons for the automatic processing of human language -

written as well as spoken - and everybody is at least familiar with the blessings of word processing.

But what happens in word processing? Not very much that has anything to do with language. The whole spiel is about how to arrange characters on a page and the big advantage over the good old typewriter is only that we can re-arrange, cut and paste, correct mistakes, and all this with no scissors, glue or correction fluid. But in an ordinary word processor there is no more linguistics than in a typewriter.

The next step up is often called electronic publishing or desk-top publishing; but also here, language plays no role. The point is rather that while ordinary word processing replaces the typewriter, electronic publishing attempts to replace the composer or type setting machine. In either case, the device does not care whether you type in words of a language or arrange them in the form of sentences or whether you just type in some gibberish - but this is no different for the typewriter.

1.2 Words

The first step where language enters the scene is when a word processor has a capability for hyphenation or spelling correction. For spelling correction the thing must be able to distinguish words from non-words and inform the typist about what it thinks are non-words, and, possibly, about what words it knows that look similar and hence may be the words that the typist intended to type but misspelled or mistyped. The spelling correction device needs at least a word list of the relevant language to do this job and an algorithm that tells it something about likely spelling or typing errors. Similarly, a hyphenation device may just use a word list where for each word the potential hyphenation points are marked.

Eventually, however, just a word list won't do. Because a word comes in many shapes. In the extreme case, these shapes have no part of their spelling in common: the English word *go* not only turns up as *goes* or *gone* but also as *went*. So what the spelling dictionary needs is either a morphological algorithm that derives at least the regular forms of a word from its root or a list of each and every possible word form of the language, and not just the few irregular cases. The latter may mean, when we consider highly inflected languages, like French or German, rather than English, that the word list will become four or five times as long. - So here we have, for purely technical reasons, a good case for the introduction of some linguistics, in this case, morphology. A morphology component can keep the word list small and hence save memory and, more importantly, will make the program easier and cheaper to maintain.

Another reason for the introduction of some linguistics is that word lists can never be complete. There are many words that everyone understands but that cannot be found in a dictionary. I do not mean four-letter words, which we could well do without, but words that people make up as they go along, by ordinary rules for word formation that everyone masters who masters the language. New words may be formed by morphological derivation: we may add prefixes, such as *de-*, *re-*, *anti-* etc. to familiar verbs or nouns, or suffixes like *-ize*, *-arize*, *-ization*, *-arization*, *-ify*, *-ification*, etc. New words formed in this way are understood with no difficulty and usually pass unnoticed. Another way of forming new words is by composition of already familiar words. In many languages, German is a particularly notorious case, such compound words are written with no blanks or hyphens, all in one word.

Compounding and morphological derivation are ordinary forms of linguistic creativity, i.e. they form part of the ordinary use of language. But neither compounds nor morphologically derived words can satisfactorily be handled by most spelling correction programs, because they are based on word list. What is required, in addition to the word list is proper linguistic knowledge: morphology as well as the regularities of compound formation, plus, as we shall see below, semantics.

1.3 Grammar

But there are more problems, still in the fairly simple area of spelling correction. Spelling correction devices are fundamentally dictionary based. This means, first of all that the name *spelling correction* arouses the wrong expectations. For what the devices do is only to check for each individual string that occurs between two blanks whether it corresponds to a word in the word list, or, if there is a morphology component, if the string is a regular derivation from a word that is in the dictionary. But this means that spelling mistakes are not recognized whenever the incorrectness of a string only follows from the context. Typical cases in English are words like *its* and *it's* or *their* and *there*, which are frequently mixed up. But only syntactic analysis can tell us which is which and which is right in a particular place in a sentence.

A further problem in spelling that cannot be dealt with on the basis of a dictionary (with or without morphology) is the problem of whether to write an expression as one or two words or with a hyphen. The mere fact that two words occur singly in a dictionary does not imply that there is not also a word that consists of the concatenation of the two strings (*in* and *deed* happen to occur singly in the dictionary, but there is also a word *indeed*). Another problem of the same sort is capitalization: one and the same word form may occur in the language capitalized or non-capitalized and only the syntactic context, or worse, the meaning, will tell us which is at stake.

It is clear from these observations that there is quite a way to go until we will reach a stage in automatic spelling correction that approaches a stage at which we could honestly speak of "spelling correction" or even "automatic proof reading". And it is also clear that a good deal of linguistics is required in order to get closer to this goal: syntax, morphology, and, eventually, semantics.

2 Why not be happy with what there is?

2.1 Taking the user serious

Why should one proceed this direction and add more linguistic knowledge to word processors rather than just be happy with what there is? The major reason to be dissatisfied with what there is is that there is too much of a difference between the user's expectations and what current programs can do. When I write a French text and ask a Frenchman to correct the spelling, I expect at least that he gets rid of mistakes of the type just mentioned, for this is what we ordinarily mean by "correcting the spelling". But when I run the spelling correction program on my word processor, I have to be aware that it does a good deal less: that the device does not detect or correct mistakes on the basis of any form of linguistic understanding but merely uses a bunch of tricks. Hence, in order to appreciate what exactly the program does, the user must learn to understand the program.

We find ourselves back in the familiar situation which we would rather like to get away from: the user has to adjust to the machine, has to first understand the machine, rather than having machines that are adjusted to the user and fit in with the user's expectations.

That the problem is real is quickly appreciated when we look at customer complaints about spelling correction devices. In one case a large factory complained that a whole set of technical terms they (and nobody else) use for some of their products were not in their word processor's dictionary. "These are perfectly ordinary everyday words", they said and were dissatisfied that the spelling correction device kept marking them as spelling mistakes. In another case, a customer complained that their spelling correction program not only kept marking such "perfectly ordinary everyday words" wrong, but also suggested "silly" corrections. An example the customer gave was the name of the then Prime Minister of the GDR, *de Maizière*, which the program preferred to correct as *Malzbier* [brown ale]. In another case in the German word for liability insurance *Haftpflichtver-*

sicherung the typist had missed out an *e* and got proposals for correction that included *Haftpflichtversdichtung* and *Haftpflichtverschwörung* ["liability versified poetry" and "liability conspiracy" respectively].

Of course such proposals, which result already from the inclusion of a facility that treats compound words, can only be prevented with the addition of a semantic module that recognizes that the proposed compounds are just regular nonsense. - But how should the user know? Cases of this kind were regularly passed on to my department when we had the responsibility for the word lists included in some of IBM's word processors.

The point is that we have to take the user seriously: either educate him to understand how the systems work, or build systems that do not need this understanding because they process language in a way sufficiently similar to the human being, i.e. systems that propose corrections on the basis of at least a superficial form of linguistic understanding. The only option we have, in the long run, is the latter. A user can perhaps be brought to accept that particular technical jobs just have to be done with the help of an awkward tool that never quite does what it should do, and the user may even blame his own incompetence in such cases. But this does not work for things everybody is an expert in: the use of ordinary language.

2.2 Taking human language serious

A more general point is that computer programs are tools tailored for particular tasks and applications. A program will work fine within the boundaries of what the designers of the program, at the design stage, considered part of the application and it will work less satisfactorily when it is used for borderline cases. For many conventional or standardized applications, there is no problem in designing the program so that it fits exactly the user's conception of the application. The user will not expect anything of the device that is beyond its capabilities, because the designers knew in advance what the user would expect. In some, surely less ideal, cases, the user accepts that a certain amount of training is required to use a program comfortably and understand its capabilities.

But human language is not limited to conventionalized or standard uses but is a tool of nearly universal capabilities. It does not care what you speak about nor how much you know about what are talking about. You can use the same English language to inquire about the train schedule, to write a love poem, or to give somebody instructions for the repair of a diesel engine. There is a little difference in the vocabulary, but this seems about all; it is still English.

Now why can computational linguists not just model exactly this linguistic knowledge that allows us to inquire about train schedules as well as write love poems and give instructions about diesel engines? Why can we not have an English language module that does for the computer precisely what our knowledge of a language does for us? In other words: why can we not have a fully portable English language engine that can be connected to whatever program you please: say a database, to allow us to query the database or add information to it, or to another language engine, say one for German, to give us automatic translation between the two languages, or to some complicated technical system in order to control the relevant processes by the word of our mouth rather than by some silly Fortran or Pascal program that nobody understands?

Before we proceed and try to answer this question, I would like to present to you a computer system that understands German text and try to show, by discussing the limitations of this system, what can and what can currently not be done in this line of business.

3 Full text understanding

3.1 Goals of the LILOG project

The LILOG project was set by IBM Germany in 1985 in order to investigate the possibilities of semantic information processing for natural language, in particular for German. The initial goal was not to develop a product, but rather to do research and develop new technologies for natural language processing and for knowledge-based systems more generally. The idea of the project was to jump in at the deep end and accept the challenge of the most complex and most difficult task in natural language processing: full text understanding. But why start with the most difficult task, rather than continue, say, in improving spelling correction devices?

In text understanding we meet many, if not all, of the problems that occur also in simpler tasks of natural language processing, and, what is more, we can investigate these tasks in their mutual interaction. The crucial point, of course, is that in text understanding the two traditionally distinct disciplines of linguistics and logic (hence LI-LOG), or, more specifically, computational linguistics and artificial intelligence, have to be brought together in order to achieve interesting results.

The research task of the LILOG project then has been to investigate text understanding and to implement a system on the computer that actually understands written text, German text in this case. - But how can we judge whether a computer has understood a text or has even read it? Well, the same way a teacher checks whether a student has read and understood a text: by asking questions about the contents. Hence the system that had to be built not only had to be able to read and understand texts, but also to understand questions about the texts and answer them.

3.2 What the LILOG system does

The current version of the LILOG system, LEU/2, reads texts from tourist guides about the City of Düsseldorf like the following:

Im Palais Nesselrode ist das Hetjensmuseum, das 1909 eröffnet wurde, untergebracht. Es befindet sich an der Ecke Schulstraße und Hafenstraße. Die Keramiksammlung umfaßt zehntausend Objekte. Der Eintritt der Ausstellung, die von 10 Uhr bis 17 Uhr geöffnet ist, beträgt 2 DM.

[The Hetjens Museum, which was opened in 1909, is housed in Palais Nesselrode. It is located at the corner of Schulstrasse and Hafenstrasse. The ceramics collection contains ten thousand items. The admission to the exhibition, which is open from 10 a.m. til 5 p.m., is DM 2.]

When this text is processed a representation of its content is stored in the system's text knowledge base, or, in a little more detail: first each word in the text is looked up in the system's dictionary and the morphological, grammatical, and semantic information is handed on to the parser, which produces a linguistic analysis for the first sentence. Then a number of semantic operations are performed on the sentence, linking the contents of the sentence to information in the knowledge base of the system, and finally a translation of the sentence into a logical knowledge representation language (LLILOG) is produced and put into the text memory. The following sentences are processed in a similar fashion, except that here the semantic analysis takes into account the contents of the preceding sentences.

The importance of this latter point is seen when we want the system to answer questions like for instance the following:

Wann hat das Hetjensmuseum geöffnet?
[When is the Hetjens Museum opened?]

The answer of the system,

Von 10 Uhr bis 17 Uhr.
[From 10 a.m. til 5 p.m.]

is evident to anyone who has read the text, but how does the system arrive at this answer? Explicitly the text only gives the opening times of an exhibition. So the system must figure out that the exhibition is the exhibition of the ceramics collection, that the ceramics collection is presumably part of the museum and that, whenever the exhibition is opened, presumably the museum will be open too - a fairly complex task of resolving anaphorically definite NPs, which relies on fairly sophisticated knowledge representation and reasoning. In particular, we need common sense knowledge (e.g. that museums tend to have collections and exhibitions) and the knowledge of the various sentences of the text must be integrated - otherwise the exhibition would have nothing to do with the ceramics collection and the museum, and there would be no way of answering the above question.

Also in a dialogue with the user the system must keep track of what was just said before. Otherwise a question like

Ist es um 14 Uhr geöffnet?
[Is it open at 2 p.m.?)]

following the above question, would make no sense. The *es* [it] would not be related to the museum. Similarly questions like

Wieviel kostet der Eintritt?
[How much is the admission?]

require an appropriate preceding discourse, because only then *the admission* can be interpreted in the intended fashion as *the admission to the Hetjens Museum*.

The LILOG text understanding system does a good many other things, like tell the user how to get from A to B in Düsseldorf, in words as well as with the help of a map, find its way round words it doesn't know, and it has very interesting capabilities in reasoning, ordinary logical reasoning as well as reasoning in an analogue fashion with mental images. But here we want to stick to the more strictly linguistic capabilities and the knowledge that is required for them.

3.3 What the LILOG system cannot do

At a demonstrations of the LILOG system at a technology show we were approached by an officer of the Inland Revenue with the proposal that we might apply the system to the German laws and regulations for income tax. The idea being that once all these texts have been understood by the system, it would be quite easy to ask questions like *Can someone who is self-employed deduct the expenses for their camping van from their income tax?* - Perhaps the system would initiate a dialogue with the user in order to clarify all the relevant details, but eventually it should tell the user whether and under what conditions the intended tax deduction is possible.

The application is certainly attractive and one of the longer term goals of text understanding systems of course is that they should be able to understand in principle any old text, even the law. But at present there are two important obstacles that make it inconceivable to consider such an application.

One is that understanding a text not only requires knowledge of the language, but also a conceptual

model of the subject matter of the text. This is why most people do not understand the tax laws and need professional help (and not from linguists), despite their good knowledge of German. In fact, the situation is worse, particularly in law: there is not only no assurance that the law is consistent, but we know that jurisdiction interprets the law in the sense of deciding cases that are not clearly decided by what it says in the text of the law.

But even if the application was not to legal texts, but to a subject matter that is simpler in the sense that we can be reasonably sure that there is a consistent model, there would still be difficulties. In the first instance in actually building the relevant conceptual model, i.e. in carrying out the knowledge engineering for that domain. This is not as task with any principled obstacles, at least for many applications it is not, but the challenge is in the amount of work. - If we know that the technical documentation for a jumbo jet, printed out on paper, is heavier than the plane could carry, then we get a good idea of what it means to represent a complex domain of technical knowledge. But whenever natural language enters the scene, a good deal of common sense knowledge is required as well. Now, nobody knows anything about how large the common sense knowledge of an average person is, but what we do know, is that it is probably less consistent than a complex legal text. In sum: as long as research into natural language systems is a matter of pure research, it is very unlikely that anybody will fund the amount of work that is required just on the knowledge engineering side for either complex technical application or generic, i.e. application-independent, natural language systems that require full common sense knowledge.

3.4 The problems of ambiguity and polysemy

The other obstacle is the linguistic one. Already when we want to apply the LILOG system to much smaller and less complicated domains than the one just sketched, we run into difficulties, in particular with word meaning.

Consider, for instance, the meaning of *Eintritt* [admission], as it is relevant for the current application. What does the system know about the semantics of this word? It basically knows just one type of context, the one in which *Eintritt* means *admission fee* and it knows that museums as well as other art institutions charge for admission. Now suppose we were to tell the system that someone "was not granted admission to" a particular museum. Then the first problem is that the construction *to grant admission to* will not be understood. For two reasons: one is that the verb *to grant* has a very large number of senses and this particular one is not included in the current linguistic module. The other reason is that the sense of *admission* that is required, is not in the dictionary. As far as the LEU/2 system is concerned, *Eintritt*, or *admission*, is a certain amount of money you pay to be allowed into a museum. - Clearly, these two senses are only the tip of the iceberg and there are many more.

The way natural language systems start with the problem of word senses is probably also how we, as human beings, once started with most words: we learnt them in one particular context and gradually we found more applications in other contexts so that eventually we came to develop a representation of the meaning of words that more or less covers the regular uses the word has in our language. There is, however, no assurance, that we will not hit upon a use of the word that is actually new to us. And it is this situation where the LILOG system, as well as any current natural language system, differs from the human being: the system has no capabilities for relaxing the conditions the meaning of a word imposes, or slightly modifying them, to cover a newly encountered variant.

This means that the only option we have for natural language systems, at the moment, is to actually explicitly code each and every known sense of a word. The reason that we don't do this is not only that no-one would be prepared to fund this gigantic activity or that we were genuinely lazy. A more

important reason is that we could currently not cope with the resulting ambiguity in processing. The more word senses we have in a system the more ambiguity we create. Each time an ambiguous word is encountered in a text, the ambiguity must be resolved, and since ambiguity resolution is not a matter of single words but a matter of a word in a context, we first get a combinatorial explosion of ambiguity (each ambiguous words combines, in principle, in each of its senses with each sense of each other word).

The pragmatic approach that is taken for all current natural language systems to cope with the problem of combinatorial explosion of lexical ambiguity or polysemy is to keep the systems relatively dum: the fewer word senses the system knows, the less of a problem it has with the explosion of ambiguity. (It seems that Nature takes a similar approach with children: they discover more word senses as their capability to handle them in language processing increases.) - Now, plainly, when we restrict a natural language system to a small and reasonably limited application, its limited knowledge of word senses yields no problems at all and has the additional advantage that also the knowledge engineering for the application remains manageable.

A consequence from these considerations is that in the near future there is no realistic option for large generic natural language systems, i.e. application-independent systems. This is no news to the specialist, but it is a priori not easy to understand for the non-specialist. The non-specialist is not usually aware of the considerable dependence of linguistic understanding on non-linguistic knowledge of the relevant subject matter, even though also he is aware of extreme cases: in order to understand the German tax regulations, mere knowledge of German is plainly insufficient. But also in less specialized areas the dependence on knowledge is just the same, the only difference being that there we are concerned with common sense or everyday knowledge that is generally and naturally available to anyone and that we acquire as we grow up, simultaneous with the acquisition of our mother tongue. The non-specialist is also not usually aware of the considerable number of relevant differences in the specification of word senses as they are required whenever explicit representation of meaning is necessary, as in computer systems. To someone who masters a language, the differences between various word senses are normally unnoticeable, just as a fluent speaker of a language is entirely unaware of any rules of grammar.

3.5 The way ahead

Let me illustrate the case of the almost uncontrollable number of word senses, with an example: the colour adjective *red*. When we look up this word in a dictionary, we only get a very rough impression of what is at stake, and will not appreciate the full scope of the problem. But consider the following uses of *red*:

red tomato

red apple

red hair

red wine

red grapefruit

red indian

red nose

We are here not concerned with an ambiguity of the kind we find with words like *bank*, where there is a straightforward choice between either the financial institution and the bank of a river. There is good reason for the suspicion that there are not just the seven variants of the concept of redness that we find in the examples, but that this list can be extended ad libitum. And the equivocation we observe is definitely an equivocation that is relevant for language understanding, for the inferences

our system must draw, and is not just a matter of shades of meaning one might just as well ignore. If the adjective *red* were to make the same contribution to the inferences of the system in all these cases, the conditions for calling a tomato, an apple, hair, wine, etc. *red* should be the same, which they are not, and one would surely want to avoid the system's conclusion that tomato, hair, grapefruit, etc. are all of the same colour. When a tomato is called red, you may infer that it is ripe - not an appropriate conclusion for either the Indian or the nose. We may also infer that it is red all over (perhaps with the exception of a couple of green leaves) - and this conclusion is not appropriate for the red apple: apples may be called red even when they are partly white or yellow. And the red grapefruit, of course is yellow all over, only the pulp is sort of pink. So not only are not all these *red* objects of the same colour, but there are quite different ways in which the qualification *red* applies to them.

What we find here, perhaps more clearly than in many other cases, is an obvious interaction between the senses of a word and our knowledge of the world: someone who doesn't know about red grapefruits, for instance, will never be able to predict what *red* means in application to grapefruits.

But how do we get ambiguity and polysemy under control? - The approach required is in fact the same we saw at the beginning of this paper for spelling correction devices. There we saw that we get nowhere as long as we attempt to just list all the word forms of a language and build our device on this basis. Similarly, the mere listing of word senses is a hopeless and, as the example of *red* shows, never-ending exercise. What is required rather is a way of generating new senses from ones already known (a semantic "morphology", if you want to keep up the analogy to the spelling correction problem). And this is by no means a hopeless task. But it is a task that spans the handling of world knowledge and its conceptualization as well as traditional linguistic semantic research, approaches from cognitive psychology and more recent work from machine learning. Another side of the problem, as we saw above, is the handling of combinatorial explosion of ambiguity. Thus it is not enough to have just any old mechanism that generates all possible senses of a given word, but we want this mechanisms to generate, at the right time in linguistic processing, exactly those senses that are required at that point in processing, ideally only one word sense: a great challenge for the interaction of linguistic and logical modules in a natural language system.

But a good amount of work will be required until we are anywhere near this goal. - The orientation I have sketched comes from an argument about the research problems that lie behind actual deficiencies in available products. But this does not imply that this is an orientation that leaves out applications. The research work ahead can, I believe, only successfully be carried out when intermediate results are regularly tested in application work. However, the guiding principles have to come from a good understanding of the research situation.

This way of tackling current problems in natural language processing and working towards increasingly less application-dependent systems is in fact suitable for industrial as well as academic research environments. Next to the longer-term research topics of a more basic orientation, small applications can be developed in which intermediate results are used to demonstrate the practical usefulness and actual progress of the more basic research. The only serious danger in this situation is that overdrawn expectations based on insufficient understanding of the problems that lie at the roots of natural language systems will make us too impatient and go in for a purely application driven research, with no perspective for the general problems.