PANEL

Language Engineering : The Real Bottle Neck of Natural Language Processing

Panel Organizer, Makoto Nagao
Department of Electrical Engineering
Kyoto University, Sakyo, Kyoto, Japan

The bottle neck in building a practical natural language processing system is not those problems which have been often discussed in research papers, but in handling much more dirty, exceptional (for theoreticians, but we frequently encounter) expressions. This panel will focus on the problem which has been rarely written but has been argued informally among researchers who have tried to build a practical natural language processing system at least once.

Theory is important and valuable for the explanation and understanding, but is essentially the first order approximation of a target object. As for language, current theories are just for the basic part of the language structure. Real language usage is quite different from the basic language structure and a supposed mechanism of interpretation. Natural language processing system must cover real language usage as much as possible. The system model must be designed in such a way that it is clearly understandable by the support of a powerful linguistic theory, and still can accept varieties of exceptional linguistic phenomena which the theory is difficult to treat. How we can design such a system is a major problem in natural language processing, especially for machine translation between the languages of different linguistic families. We have to be concerned with both linguistic and non-linguistic world. While we have to study these difficult problems, we must not forget about the realizability of a useful system from the standpoint of engineering.

I received valuable comments from Dr. Karen Jensen who cannot participate in our panel, and kindly offered me to use her comments freely in our panel. I want to cite her comments in the followings.

Why Computational Grammarians Can Be Skeptical About Existing Linguistic Theories

Karen Jensen
IBM TJ Watson Research Center
Yorktown Heights, NY10598, U.S.A

1. We need to deal with huge amounts of data (number of sentences, paragraphs, etc.). Existing linguistic theories (LTs) play with small amounts of data.
2. The data involve many (and messy) details. LTs are prematurely fond of simplicity. For example: punctuation is very important for processing real text, but LTs have nothing to say about it. (This is actually strange, since punctuation represents -- to some extent -- intonational contours, and these are certainly linguistically significant.)
3. There is no accepted criterion for when to abandon an LT; one can always modify theory to fit counterexamples. We have fairly clear criteria: if a computational system cannot do its job in real time, then it fails.
4. We need to use complex attribute-value structures, which cannot be manipulated on paper or on a blackboard. "Trees" are only superficially involved. This means we are absolutely committed to computation. LTs have various degrees of commitment.
5. We are not interested in using the most constrained/restricted formalism. LTs generally are, because of supposed claims about language processing mechanisms.
6. We are interested in uniqueness as much as in generality. LTs usually are not.
7. We are more interested in coverage of the grammar than in completeness of the grammar. LTs generally pursue completeness.
8. We aim for "all," but not "only" the grammatical constructions of a natural language. Defining ungrammatical structures is, by and large, a futile task (Alexis Manaster-Ramer, Wlodzimierz Zadrozny).
9. Existing LTs give at best a high-level specification of the structure of natural language. Writing a computational grammar is like writing a real program given very abstract specs (Nelson Correa).
10. We are not skeptical of theory, just of existing theories.

Existing linguistic theories are of limited usefulness to broad-coverage, real-world computational grammars, perhaps largely because existing theorists focus on limited notions of "grammaticality," rather than on the goal of dealing, in some fashion, with any piece of input text. Therefore, existing theories play the game of ruling out many strings of a language, rather than the game of trying to assign plausible structures to all strings. We suggest that the proper goal of a working computational grammar is not to accept or reject strings, but to assign the most reasonable structure to every input string, and to comment on it, when necessary. (This goal does not seem to be psychologically implausible for human beings, either.)

For years it has seemed theoretically sound to assume that the proper business of a grammar is to describe all of the grammatical structures of its language, and only those structures that are grammatical:

> The grammar of L will thus be a device that generates all of the grammatical sequences of L and none of the ungrammatical ones. (Chomsky 1957, p. 13)

At first blush, it seems unnecessary to conjure up any justification for this claim. Almost by definition, the proper business of a grammar should be grammaticality. However, it has been notoriously difficult to draw a line between "grammatical" sequences and "ungrammatical" sequences, for any natural human language. It may even be provably impossible to define precisely the notion of grammaticality for any language. Natural language deals with vague predicates, and might itself be called a vague predicator.

This being true, it still seems worthwhile to aim at parsing ALL of the grammatical strings of a language, but parsing ONLY the grammatical strings becomes a dubious enterprise at best. Arguments for doing so reduce either to dogma, or to some general notion of propriety. Arguments against, however, are easy to come by. Leaving theoretical considerations aside for the moment, consider these pragmatic ones:

(a) The diachronic argument. The creativity of human use of language is great, and language systems are always changing. A construction that was once unacceptable becomes acceptable over time, and vice versa. Even if a grammar could