"Linguistic" Sentences and "Real" Sentences

Masaru Tomita
Computer Science Department and
Center for Machine Translation
Carnegie Mellon University

This paper identifies two kinds of sentences: "linguistic" sentences and "real" sentences. The former is a kind of sentences that are often discussed in (computational) linguistic literatures, such as those in Figure 1. The latter, on the other hand, is a kind of sentences that appear in practical applications, such as those in Figure 2. Whereas both are grammatical English sentences, they appear to be significantly different. In this paper, we discuss the characteristics of those two kinds of sentences, and claim that a different approach is necessary to parse each kind of sentences.

John hit Mary.
Every man who owns a donkey beats it.
I saw a man with a telescope.
The horse raced past the barn fell.
Time flies like an arrow.
The mouse the cat the dog chased ate died.
John persuaded Mary to expect that he believes
that she likes an apple.

Figure 1: "Linguistic" Sentences

All processes (programs) in the destroyed window (or icon) are killed (except *nohuped* processes; see *nohup(1)* in the *HP-UX Reference*); therefore, make sure you really wish to destroy a window or an icon before you perform this task.

This window contains an HP-UX shell (either a Bourne shell or C-shell, depending on the value of the SHELL environment variable; for details, see the "Concepts" section of the "Using Commands" chapter).

Figure 2: "Real" Sentences

It seems that problems in parsing sentences can be classified into two categories: linguistically "interesting" problems and linguistically "uninteresting" problems. Linguistically "interesting" problems are those for which there are no obvious solutions, and reasonably sophisticated theories are required to solve them, or those behind which there are general linguistic principles, and a small number of general rules can cope with them (e.g., relativization, causativization, ambiguity, movement, garden-path, etc). On the other hand, linguistically "uninteresting" problems are those for which there exist obvious solutions, or those behind which there is no general linguistic principle, and it is just a matter of writing and adding rules to cope with these problems (e.g., punctuation, date and time expressions, idioms, etc).

Figures 3 and 4 show example "interesting" and "uninteresting" problems, respectively. While one could give an elegant explanation of why the second sentence in figure 3 is ungrammatical, there is no particular reason why "15th July" is ungrammatica, other than that it is simply not English.

John expects Mary to kiss herself.
* John expects Mary to kiss himself.
John expects Mary to kiss him.

Figure 3: An Interesting Problem

on July 15th
on the fifteenth of July
on 7/15
* on 15th July
* in July 15th

Figure 4: An Uninteresting Problem

"Linguistic" sentences usually contain one or more linguistically interesting problems, with few or no linguisctically uninteresting problems. "Real" sentences, on the other hand, contain many uninteresting problems, but fewer interesting problems. In (computational) linguistic literatures, uninteresting problems can be ignored, as long as everybody agrees that there are obvious solutions for them. In practical applications, on the other hand, we cannot ignore uninteresting problems, or systems simply do not work.

One of the projects at the Center for Machine Translation at Carnegie-Mellon University is to translate personal computer manuals from English to Japanese and from Japanese to English. In this project, and perhaps in any other practical projects that have to deal with "real" sentences, the system's failures are caused by a few interesting problems and tons of uninteresting problems. There often exist reasonable approximate solutions to interesting problems in practical applications; for example, it is quite acceptable to assume that there are no embedded relative clauses in computer manuals, in order to simplify the (interesting) problem of relativization. On the other hand, there are no quick solutions to uninteresting problems other than writing a banch of rules.

- We can never anticipate and prepare for all of these uninteresting problems in advance. It seems as if there will be always these problems no matter how carefully and how many times we test and debug the system and its grammar.
- The quantity of the knowledge sources (i.e., grammars/rules) has to be very large; unlike interesting problems, rules for uninteresting problems can hardly generalized into a smaller number of rules, as each of them represents an uninteresting problem with no general linguistic principles behind it.
- It is more difficult for humans to test, debug, and maintain a larger amount of knowledge sources accurately and consistently.
- It is more difficult for a system to access a larger amount of knowledge sources efficiently.

These problems are much more serious than linguistically "interesting" problems, and directly affect performance of practical systems.