PET: PROCESSING ENGLISH TEXT

F. Oppacher

Dept. of Computer Science, Concordia University, Montreal, Canada

This paper describes a new parser that combines top down and bottom up strategies and the natural language processing system PET which uses the parser. PET is designed to facilitate the interactive construction of natural language front ends and to support experiments in computational linguistics. The system, which has been implemented in UT-LISP, provides facilities for performing the following tasks: natural language parsing according to context-free and transformational grammars; disambiguation of word senses by pattern--directed inference; construction of a semantic network data base from English sentences; deductive information retrieval to answer simple English questions.

Most extant natural language processing systems have very complicated control structures and are, therefore, difficult to extend modularly. It appears that the lack of modularity of these systems is due to the fact that their syntactic expertise is not conveniently located in one routine but, in effect, distributed throughout the entire program.

In the system described here, modularization is achieved by making the control structure largely transparent, i.e. by allowing it to reside in the parser.

The method, in a nutshell, is this: the parser outputs a phrase structure tree, or, if the analyzed sentence is structurally ambiguous, a list of several phrase structure trees. A tree for a sentence is generated in bottom up fashion under the control of context-free rules and restricted context--sensitive rules. The latter consist of tests and tree-modifying functions. The context-sensitive tests inspect the immediate environment of a node about to be entered into the tree. Depending on the outcome of such tests, the tree-modifying functions can change partially constructed parse trees. The interior nodes of the tree are occupied by functions corresponding to transformational and/or semantic rules, and the leaf nodes are occupied by the dictionary entries of the words in the surface string. Since LISP, unlike other high level languages, makes no distinction between programs and data structures, the tree generated by the parser can be immediately executed as a program. The tree, interpreted as program, constitutes the control structure referred to above, and governs the semantic interpretation of the sentence whose structure it reflects.

As can be seen from this very rough description, control issues involved in processing natural language text are indeed largely taken care of by the syntactic component. However, to eliminate semantically uninterpretable parses and to help resolve subtle syntactic ambiguities, the parser must occasionally communicate with the semantic component.

The parser attempts to eliminate the overhead incurred by pure top down (TD) or bottom up (UP) algorithms. TD algorithms may have to do a lot of backtracking because of wrongly predicted goals. BU algorithms build many temporary structures which will not figure in the final parse. Both backtracking and the generation of all possible BU interpretations can be avoided by suitably combining TD and BU strategies: TD expectations based on the grammar itself and on what has been parsed so far guide and constrain the BU search, while BU results are used at once to refute or confirm TD expectations.

The parser is described and contrasted with several other TD and BU parsers.

The most notable features of the operation of the parser are the following. As the sentence is traversed from left to right, TD expectations are associated with each position in the sentence. Each word in the sentence is thought of as lying between a beginning and an ending position, so that the i-th word lies between positions i-1 and i. Nodes are built only when they agree with prior expectations and when they meet additional context-sensitive tests.

At each moment, the parser attempts to keep the forest of partial trees as shallow as possible. After the input words have been processed from left to right, the roots of the trees constructed so far are visited in alternating right to left and left to right order. With each pass the height of some trees in the parse forest increases until the root for the entire sentence is built.