Mother Tongue To Boolean

Since its early 1950s beginnings, computer-based natural language understanding has been inextricably linked to theoretical linguistics, a young science no older than computing itself.

And one of the crucial lessons so far learnt from both disciplines is the measure of just how complex human language is. Human beings understand each other with such ease that it's hard to see the machine's problem – until you try to write a computer program to imitate that understanding.

Early research, based on a blithe underestimation of the complexity of language, was crude. The pioneers of computational linguistics assumed, for instance, that fully automatic machine translation would require little more than a bilingual dictionary and a few rules of inflection.

But steady progress has been made. The commercial products having made their appearance during the past decade all do their best to address the major input text stumbling blocks of ambiguity, vagueness, and ill-formedness.

The spur to succeed has lain in the enormous potential benefits of computers that can understand natural language. If your computer can talk to you in your own native tongue, then such applications as database consultation and computer-aided teaching will be made both easier and many times more effective.

And the understanding of full written texts will be the basis for such applications as machine translation, expert systems, and online knowledge bases.

Understanding is itself a nebulous concept. What does it mean for a computer to "understand?" In a paper published in 1950, not long before his death, computing pioneer Alan Turing suggested that behavior was what counted. If a computer responded to a piece of linguistic input just as a human would do in the same situation, then it had understood it.

While no one would suggest that today's natural language understanding programs are as intuitive as humans – certainly, they lack the breadth and flexibility of human understanding – they can nevertheless perform their tasks well, even if limited to narrow domains.

FIRST PARSE

A basic natural language understanding system typically contains a parser and grammar, a semantic interpreter, and an application module. In addition, it needs a lexicon or dictionary, and possibly a knowledge base.

The job of the parser is to analyze the input's syntax, creating a tree structure that shows the part of speech of every word in it and how each word fits into the larger structure of the sentence – much as used to be taught in secondary school English. For example, figure 1 shows the result of parsing the sentence "Peter Piper picked a peck of pickled peppers."

The system's parser should be independent of any one language. It is given a grammar and lexicon for each language upon which it will operate. The way grammar is represented in the computer varies according to the parsing method, but most methods explicitly or implicitly use rules of constituency.

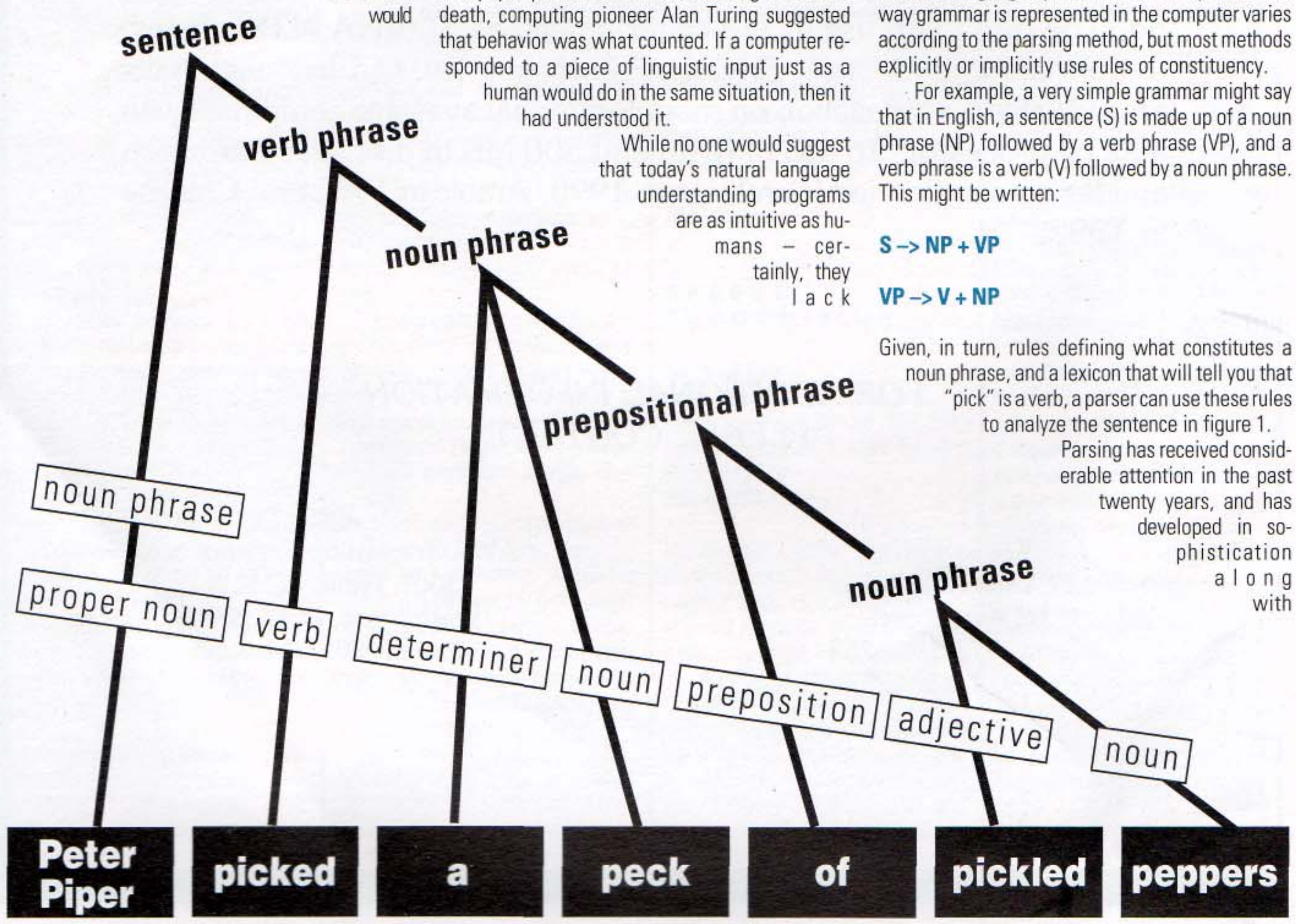
For example, a very simple grammar might say that in English, a sentence (S) is made up of a noun phrase (NP) followed by a verb phrase (VP), and a verb phrase is a verb (V) followed by a noun phrase. This might be written:

S –> NP + VP

VP –> V + NP

Given, in turn, rules defining what constitutes a noun phrase, and a lexicon that will tell you that "pick" is a verb, a parser can use these rules to analyze the sentence in figure 1.

Parsing has received considerable attention in the past twenty years, and has developed in sophistication along with



It's one thing to be able to use your computer to search for a word in a termbank or typeset a text. It's quite another for your computer to determine the meaning of your text.

Research into computer understanding of human, or natural, language has been going on for the past thirty years – though it's only during the past decade that products making use of NLU have appeared on the market.

How far are we in getting computers to understand the flawed inconsistencies of human language and translate them into the Boolean formulas of their own "native tongue?" To cast a long eye on the current state of natural language understanding, Electric Word turned to Graeme Hirst, head of NLU research at the University of Toronto.

modern theories of syntax. Indeed, there has been considerable interaction between the two disciplines. The result is that there are now grammars of considerable sophistication for English and other major languages.

The stage after parsing is semantic interpretation: taking the parsing tree and figuring out what the sentence actually means. This includes deciding which sense the speaker or writer has intended when a word or construction is ambiguous, and deciding what the antecedents of pronouns are.

Generally, semantic interpretation involves translating the sentence into some kind of representation of its meaning. This representation may use formal logic – possibly one of the many formalisms for knowledge representation developed in artificial intelligence – or be expressed in the input format of some other interfaced program.

In practice, parsing and semantic interpretation often run in parallel, because the meaning of the early parts of a sentence will often influence how the later parts are to be parsed. For example, consider these sentences:

The tourists told the guide that they couldn't understand.

The tourists complained about the guide that they couldn't understand.

In the first sentence, "that they couldn't understand" is the object of the verb "told"; in the second, it is a relative clause describing the guide. In both cases, deciding how the sentence should be parsed requires knowing what the early part of the sentence means.

The final stage of understanding is for the system to act upon what the sentence is saying. If the sentence is a question, the correct course of action will probably be to answer it; if it's an assertion of fact, it will probably be to store it. This final stage is often supplied by a separate, pre-existing program with which the natural language program interfaces.

BETWEEN YOU AND YOUR DATABASE

The simplest application of natural language understanding is in interactive interfaces to databases. The most basic interface needs to understand only a single sentence at a time. But if a piece of input defeats it, it should be able to ask the user to render a clarification or choose from one of several interpretations.

Typically, such interfaces parse the input, translate it into a database query language, and send it to the database, whose response is then presented to the user. The kinds of questions it will

YEAH, BUT . . . WHAT DOES THIS STUFF DO?

Electric Word readers will be familiar with such products as style/grammar checkers and the more sophisticated spelling correctors, which make non-trivial use of natural language processing technology.

These products use linguistic knowledge about natural language syntax (mainly word order), the phonological make-up of words, and even in some cases crude semantic information concerning meaningful, or normal, lexical collocations ("rancid butter," but not "rancid beef").

Other products, such as machine translation systems, natural language interfaces to databases, and natural-language-driven "advisor" systems, appear to require a deeper level of linguistic processing, which – at a pinch – might be called "understanding."

In fact, all such products available today achieve acceptable rather than perfect performance by circumventing rather than tackling the problem of genuine computer-based natural language understanding.

Nevertheless, even in the apparently humble domain of spell checking/correction, understanding – or at least a sophisticated level of linguistic processing – would seem essential if anything like 100% performance is to be achieved.

Consider how you might go about building a system capable of spotting and correcting the "spelling" error (substitution of "effect" for "affect") in "Smoking effects health."

BREAK IT DOWN

What then are the currently available text-based natural language interfaces capable of?

Despite considerable low-level differences of detail, all extant commercial systems and all but a few very recent research and prototype systems work by treating natural language as a sort of bad programming language with too many words and too much inherent ambiguity.

For example, imagine an interface to a database containing information about specialist magazines. Faced with a query like "Where do I write to to get a copy of Electric Word?" the system should respond with the appropriate address.

To do so, it has to work out that "copy" is being used in its nominal sense with a meaning similar to "edition," that "get" means "obtain," that "where" is the object of "write to," that one of the things you can write to is an "address," and that the two occurrences of "to" have different syntactic and semantic functions.

This is achieved first by looking up the words in the system's dictionary and second by syntactically analyzing them into meaningful groups.

Looking up the words may involve some morphological analysis of word variants and will yield syntactic and semantic information about them. This can be used to drive the syntactic analysis and subsequent production of a semantic representation.

Syntactic analysis is important because a word's part of speech and position in the sentence give useful clues to its semantic role: the relevant semantic sense and the way that sense combines with those of the other words.

For example, you can work out that "copy" is a noun because it occurs between "a" and "of," that it means "edition" because this is the relevant sense in the context "copy of <magazine>," and that "get" is a main verb meaning "obtain" because it is followed by a noun phrase referring to a magazine.

The end result of this process is an explicit and unambiguous representation of the meaning of the question in some formally specified semantic representation, which might look something like this:

(WH? X (INDEF Y (WRITE USER (ADDRESS X) (IN-ORDER-TO (OBTAIN USER (EDITION Y LT)))))).

Hopefully, a further round of processing will be able to match this query to a database search for the address of Electric Word – no trivial matter, since it involves many issues concerning the structure of particular databases.

CRUDE HEURISTICS

Of course, the more senses of words a system knows about, the harder the problem of selecting the relevant one. Current products "solve" this problem by having small core dictionaries and only adding other words with appropriate senses relevant to the application.

Artificial Intelligence Corp.'s Intellect product, for instance, comes with 400 words; the rest are added when the system is customized to the client's application.

But there are limits to this approach. In order to answer "Which magazine copied Electric Word and got taken over?" my system would need to know about other uses of "get" and "copy." Further problems arise when users attempt to ask questions that depend on the preceding dialogue for correct interpretation. For instance, having been given Electric Word's Amsterdam address, I might ask: "Is there a distributor in the UK?"

To answer this question correctly, the system must work out that I mean a distributor of Electric Word and furthermore that I am interested in having the address of such a distributor if one exists.

Systems rely on crude heuristics to solve these problems, which often fail to recover the intended interpretation. Small wonder then that one leading NLU researcher, Martin Kay of XeroxParc research foundation (Palo Alto, CA, USA), recently described NLU as "high-level compiling."

In order to produce better NLU, it is generally accepted that systems will need to go beyond the syntactically-guided semantic interpretation of individual sentences and aim to dynamically track the user's goals through a developing dialogue.

Ted Briscoe is an NLU researcher at University of Cambridge Computer Laboratory (UK).