Panel on Natural Language and Databases

Daniel P. Flickinger
Computer Research Center
Hewlett-Packard Company
1501 Page Mill Road
Palo Alto, California 94304 USA

While I disagree with the proposition that database query has outlived its usefulness as a test environment for natural language processing (for reasons that I give below), I believe there are other reasonable tasks which can also spur new research in NL processing. In particular, I will suggest that the task of providing a natural language interface to a rich programming environment offers a convenient yet challenging extension of work already being done with database query.

First I recite some of the merits of continuing research on natural language within the confines of constructing an interface for ordinary databases. One advantage is that the speed of processing is not of overwhelming importance in this application, since one who requests information from a database can expect the retrieval to take time, with or without a natural language interface. Of course speed is desirable, and waiting for answers to apparently simple requests will be irritating, but some delay will be tolerable. This tolerance on the part of the user will, I suggest, disappear in applications where an attempt is made to engage a system in dialogue with the user, as would be the case in some expert systems, or in teaching systems. Assuming that natural language systems will not by themselves get faster as they are made to cope with larger fragments of a natural language, it will be useful to continue with database query while we wait for miracles of technology to fill our demands for ever greater processing speed.

A second reason for not yet abandoning the database query as a test environment is that a great deal of important natural language processing research remains to be done in generalizing systems to cope with more than one natural language. Work on language universals gives reason to believe that some significant part of a natural language system for English should be recyclable in constructing a system for some other language. How much these cross-linguistic concerns ought to affect the construction of a particular system is itself one of the questions deserving of attention, but our experience to date suggests that it pays to avoid language-particular solutions in an implementation which aspires to treatment of any sizable fragment of a language, even a single language like English. The degree of language-independence that a natural language system can boast may also prove to be one useful metric for evaluating and comparing such systems. It seems clear that even the task of answering database queries will provide a more than adequate supply of linguistically interesting problems for this line of research.

Finally, it has simply not been our experience at Hewlett-Packard that there is any shortage of theoretically interesting problems to solve in constructing a natural language interface for databases. For example, in building such an interface, we have recently designed and implemented a hierarchically structured lexicon for a fragment of English, together with a set of lexical rules that can be run either when loading the system, or when parsing, to greatly expand the size of the lexicon actually used in the system. Several questions of theoretical interest that arose in that process remain unanswered; at least some can be answered by experimenting with our present system, functioning simply as an interface to an ordinary relational database.

Having argued that significant work remains to be done in natural language processing as an interface to databases, I nonetheless believe that it would be fruitful to expand the scope of a natural language interface, to permit some manipulation of a programming environment, allowing not only the retrieval of information describing the state of the system, but also some modification of the system via natural language. Of course, such a task would be facilitated by having the information about the environment stored in a manner similar to that of a database, so that our attention could be devoted to the new range of linguistic issues raised, rather than to details of how the whole programming environment is structured and maintained. I will not offer an account of how such a merging of database and general programming environment might be accomplished, but instead will offer some motivation for stretching natural language research in this direction.

It seems clear, first of all, that such an interface would be useful, given that even a common programming environment provides a wide array of tools, not all of which are familiar to any one user. While it is usually the case that one who is accustomed to a given facility would be hampered by having to employ only a natural language interface to accomplish familiar tasks (e.g., imagine typing "Move down to the beginning of the next line" every time a carriage-return was required), such an interface would be invaluable when trying to utilize an unfamiliar part of the system. A related benefit would be the ability of a user new to a programming environment to customize that environment without any detailed knowledge of it. This indirect access to the multitude of parameters that determine the behavior of a complex environment would also be convenient for an experienced user attempting to alter some rarely-changed aspect of the environment. Such a natural language interface might also cope with "how to" questions, at least serving as another link to on-line documentation.

The linguistically interesting issues that such an extended interface would raise include a greater need for some language production capability (where the ordinary database query system can get by with only language understanding), and a greater need for some discourse representation. I suspect that some new syntactic constructions might also appear, rare in a database application but more common in programming applications.

Using an extended interface of this kind, some dialogue between the user and the system would be useful, especially in cases where a request was too vague, and the system (like an expert system) could present a series of choice points to the user in order to reduce the original request to a manageable one. Presenting these choices would provide a convenient forum for research in language production, while suffering the disadvantage mentioned above of forcing us to worry more about the speed with which the system performs.

Issues concerning discourse representation could be studied with this kind of task in a fairly natural way, since questions about a programming environment would have to do in part with the changes taking place during a session, so that the system would want to keep track of at least some history of a session, both previous events and previous discourse. In addition to providing a testbed for discourse-related research, a system like this would also offer a good setting for study of tense and aspect issues which are not so readily raised in a simple database query application.

A final advantage of extending a natural language interface to include the programming environment is that if the interface were being developed in such an environment, one could use natural language to develop the natural language system itself, a property that would be not only useful but also elegant.