Menu-based natural language understanding

by HARRY TENNANT
Texas Instruments
Dallas, Texas

ABSTRACT

Menu-Based Natural Language Understanding is a new approach to building natural language interfaces. It retains the main goals of natural language systems: flexibility, expressive power, learnability and mnemonicity. However, it solves most of the problems inherent to conventional natural language systems. All queries are understood by the system, interface generation is much simpler, and less computing power is required. Many interfaces have been built using the menu-based natural language technology.

INTRODUCTION

We at Texas Instruments have developed a new approach to building natural language interfaces. We call it Menu-Based Natural Language Understanding (NLMenu).

NLMenu grew out of research on building conventional natural language interfaces—the kind where users are invited to ask whatever questions they have and the natural language understanding system will do its best to decipher what the user means. We were attempting to build a natural language interface to a help system. We had run simulations of the help system where users were to perform an editing task. When they ran into difficulties, they were to type questions, in English, into the help system. But instead of a help system, we routed the questions to a person. He would then send answers back to the user. The glich was that users had great problems expressing their difficulties. If they were having difficulties with the editing task, they seemed to have even more difficulty expressing the problems, in English, to the simulated help system. Now, add to this the problems that we knew the users would have in making a natural language system (instead of a person) understand what they were trying to express. We reluctantly concluded that the users would have more trouble trying to use a help system with a natural language front-end than they were likely to have with the original application for which they required help.

From this experiment, coupled with other problems of natural language interfaces, it was concluded that a new approach was needed. We wanted to keep the advantages of natural language: It is highly expressive, requires no learning time, and it will not be forgotten over a period of disuse. However, we wanted to eliminate some of the many problems of conventional natural language systems. The result was NLMenu. It retains the advantages of conventional natural language systems, but it solves most of their problems. It also provides some new opportunities that are not possible with conventional natural language systems.

We have accumulated a considerable amount of experience with NLMenu systems. A number of prototype interfaces have been built on LISP machines (approximately 15 such interfaces have been built by our research team). A large natural language system incorporating graphic input and output has been prototyped and is being implemented as part of a large defense contract. The technology also has been applied to the Texas Instruments Professional Computer (TIPC) and there are natural language interface products on the market today, including NaturalLink. NLMenu interfaces on the TI-PC also have been interfaced to speech recognition technology.

PROBLEMS WITH CONVENTIONAL NATURAL LANGUAGE SYSTEMS

Research has been conducted on natural language systems as interactive user interfaces for more than 20 years.[1] Although progress has been made, there are some problems inherent to the technology and the existing implementations. I will cover these for background in discussing the advantages of NLMenu. For the sake of brevity, I will restrict my comments to natural language interfaces to database systems, the most common application for natural language systems.

A conventional natural language system is one in which the user is presented with a blinking cursor and the opportunity to type in whatever question he has. It is then the natural language system's problem to understand what the user wants and return data to him. A number of problems with this have been described, and the discussion below is based primarily on the work on evaluation of natural language interfaces.[2] In this study, users were given problems to solve (data that they were to extract from a database). Their protocols were recorded and analyzed.

First, there were mechanical problems. Most of the users did not know how to type, or at least they could not type well. They all managed to peck out their queries with greater or less facility, but for some, typing was a major obstacle in itself. Users also had considerable difficulties with spelling. The natural language system under test had a spelling corrector, but misspellings still got by, which caused difficulty. Finally, users had a lot of trouble getting started. They seemed to find it difficult to articulate what they wanted to say, in spite of the fact that they had very explicit problems to solve.

Next, there were problems with understanding language itself. It was not uncommon to ask a question in a way that the system could not understand. If properly rephrased, these questions could be understood, but in their present form, they were not. This is called exceeding the linguistic coverage of the system.[3] With lots of hard work, system developers can anticipate every possible synonym, paraphrase, or point of view and prepare the natural language system for them all. So, with enough hard work, the problem of linguistic coverage could be effectively eliminated. Notice, however, that this could be difficult—imagine providing all possible synonyms for all the database values and keeping them current with a dynamically changing database.

A problem related to exceeding the linguistic coverage is exceeding the conceptual coverage of the system. If I were to ask "How many trucks did we ship in January?" I might be told that the system did not understand my query. I would assume that I had exceeded the linguistic coverage and re-

phrase, "How many January truck shipments did we have?" I might again be told to rephrase, and this could go on until I ran out of patience. The problem could be that the system does not know about truck shipments. If so, my questions have exceeded the conceptual coverage of the system.

The limits of coverage, both linguistic and conceptual, are difficult for users to infer. They tend not to learn quickly what is acceptable and what is not. Part of the problem is that natural language systems fail in very different ways from human understanding. If I ask you a question that you do not understand, one likely strategy is to ask again in simpler terms. This tends to have disastrous effects on natural language systems: They tend to be able to accept jargon but not simplified paraphrases.

We find that users tend to retreat to asking simple questions. They tend to use sentence templates that have been found to work through trial and error ("You want to ask for averages? You have to ask it this way . . ."). They also tend not to learn or use the full capabilities that the system has to offer. If they don't happen to stumble on a capability (such as making graphs of data), they may just assume that no such capability exists. Of course, they could read the documentation and find the limits of conceptual coverage and what all the capabilities are, but the whole motivation of natural language systems is to provide an interface to inexperienced users who will not need or have time for reading documentation.

The last major set of problems relates to the implementation of natural language systems. Conventional natural language systems tend to be quite large. Indeed, they must anticipate every likely synonym and paraphrase of questions from users. If they interface to large databases, they must at least be large enough to accept the database values and synonyms for those values. Generally, the dictionaries, grammars, and meaning translations are largely hand-coded. The range of likely synonyms and paraphrases must be determined, at least in part, empirically by observing how users express themselves. The large natural language systems require computers with large memories. Simple systems can be developed very quickly, but for significant applications that make the probability of entering an unacceptable question very small, considerable hand-tuning is generally required.

MENU-BASED NATURAL LANGUAGE INTERFACES

NLMenu solves the problems with conventional natural language systems outlined in the last section. In this section. NLMenu will be illustrated. In the next section, the solutions to the various problems of natural language systems will be covered, and then additional advantages of NLMenu will be discussed.

An example will illustrate the operation of menu-based natural language understanding. The user constructs a natural language query (in this case in English) in window number 1 (see Figure 1) from constituents that he selects from the active menus above. In these figures, the menus with heavy borders are active. The other menus are temporarily inactive.

Choices can be made from the active menus in a variety of ways. These examples come from an implementation on LISP machines and choices are made with a mouse. On the TI-PC, selections are made through the keyboard using arrow keys for positioning; alternatively, selections can be made in sev-

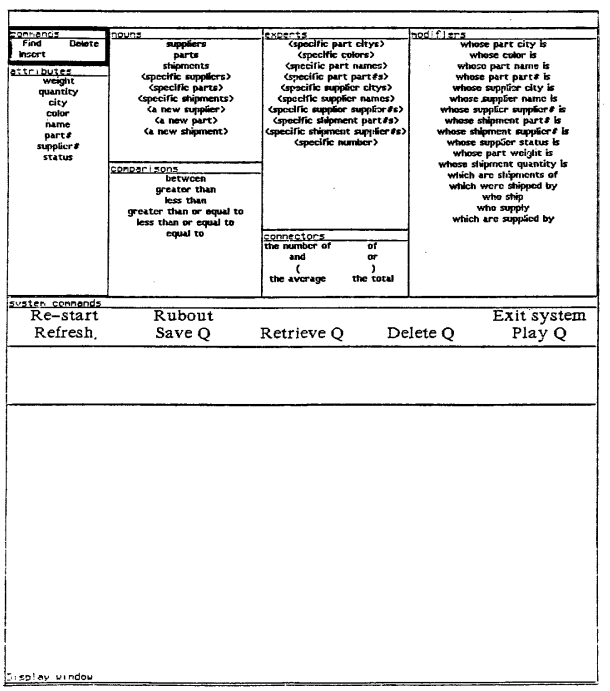

Figure 1—Building an NL menu query

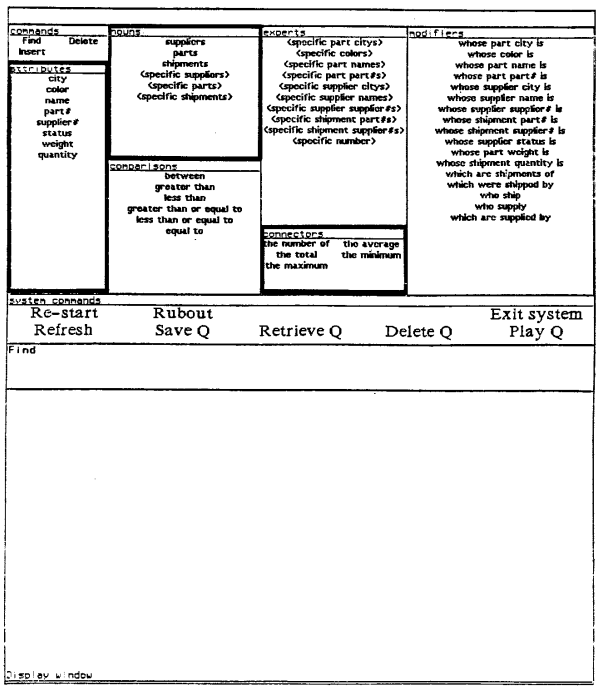

Figure 2—Building an NL menu query

eral other ways, such as dynamic text searching or selection with a mouse or other pointing device.

The user selects "Find" from the active menu in Figure 1. "Find" appears in the query window in Figure 2. He then selects "color," "and," "name," "of," "parts," and "whose color is" from a succession of active menus (Figures 2, 3, and 4, some selections are not illustrated). He then selects "⟨specific colors⟩" to specify actual database values. A special window pops up (Figure 5) with specific colors in it. The user selects "green" and "blue."

These special windows, called experts, are for specific database values. There are several ways to deal with these, depending on the application. One may select from a menu, type in a database value, or use another means. One application that we have implemented pops up a map. The user can input latitude and longitude values by pointing at the area of interest on the map.

The sentence now reads "Find color and name of parts whose color is green or blue." This is a complete sentence, understandable to the system, so the system presents the "Execute" option in the window just above the query window. The user may execute the query as it stands or continue to qualify it. He elects to execute it and the result is shown in Figure 6.

This system will understand any query that the user composes. As the user selects constituents to build his sentence, the system parses the sentence fragment. It then looks ahead in the grammar and presents the user with only those options that make sense given the current context. For example, in Figure 3, the user selected the noun "parts." The modifiers menu has become active. Phrases that did not make sense, such as "who supply" have been eliminated from the modifiers menu. Once "parts" is selected and the modifiers menu becomes active, it is limited to only those constituents that make sense; "who supply" and several others do not appear as options for the user. In this way, the user is prevented from saying anything that will not be understood. However, since the system is built on the same technology as conventional natural language systems (context-free parser, lambda-composition semantics),[5,6] it has the same expressive power as conventional natural language understanding systems.

THE PERFORMANCE OF NLMenu

NLMenu interfaces provide the same expressive power as conventional natural language systems, but the problems of conventional systems are largely eliminated. First, the mechanical problems: typing, spelling, and articulating questions. With an NLMenu interface, there is no typing. Our version on LISP runs completely from mouse selection (a mouse is a pointing device with buttons for selection). The TI-PC version works from positioning the cursor with cursor keys. In either case, the problems of typing and spelling are eliminated.

With conventional natural language systems, users often find it difficult to phrase queries, or to know how to start. With conventional systems they are confronted with nothing more than a blinking cursor, so they must compose their queries entirely by themselves. With an NLMenu interface, on the other hand, the user is presented with words and phrases from which he sees what sorts of questions can be asked.

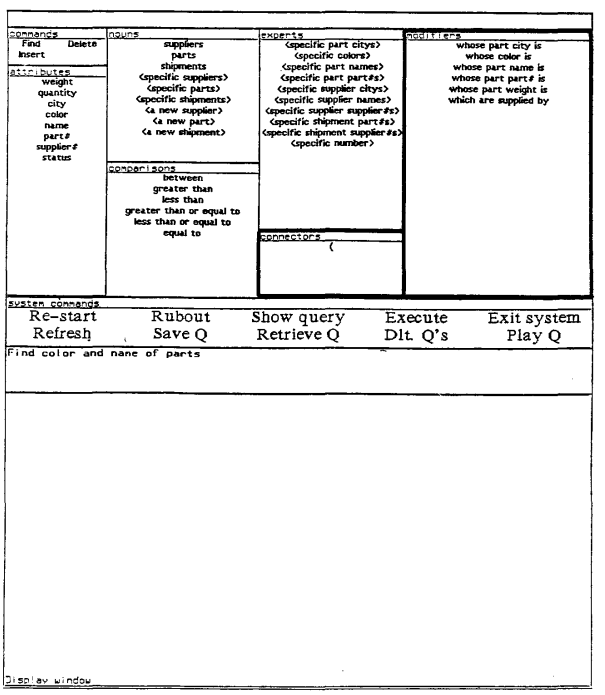

Figure 3—Building an NL menu query

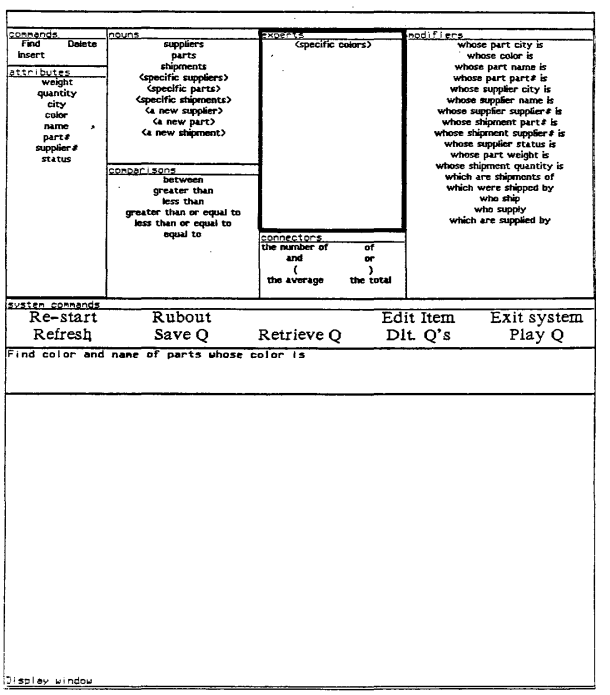

Figure 4—Building an NL menu query

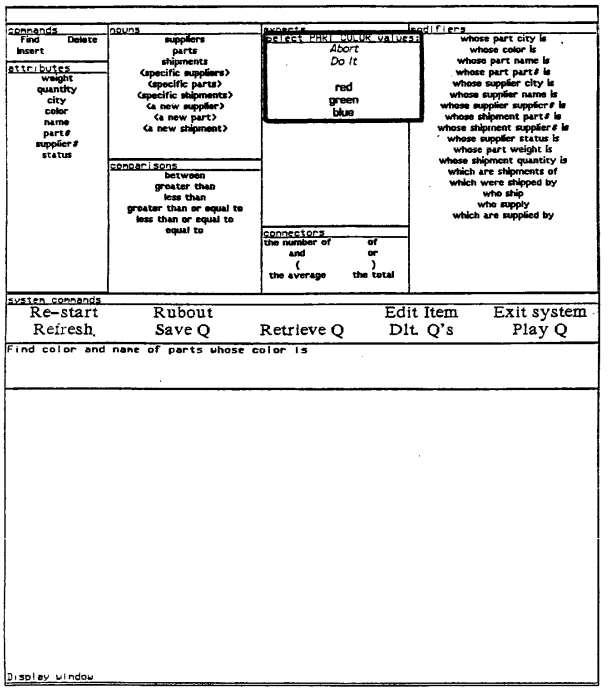

Figure 5—Building an NL menu query

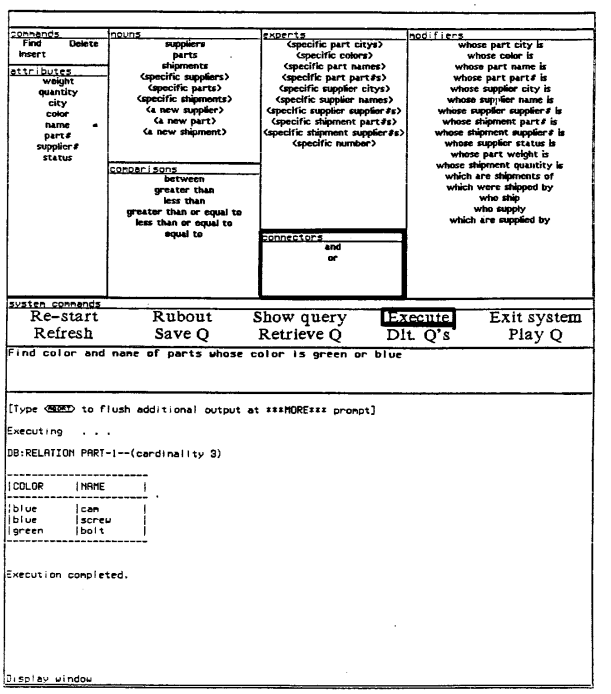

Figure 6—Building an NL menu query

Instead of composing a question, one can think of it as recognizing his question—an easier task—one phrase at a time.

The most dramatic advantage of the NLMenu interface is with language understanding itself. As was noted at the end of the last section, all queries input through the NLMenu interface are accepted—the user gets no opportunity to compose a question that would not be accepted. As a result, the problem of linguistic coverage disappears. Similarly, one cannot exceed the conceptual coverage of the system—that problem disappears as well. Notice that the problems of exceeding linguistic and conceptual coverage have disappeared not because of the massive work of finding all possible paraphrases, but from eliminating the need for paraphrases.

Another problem that is solved by NLMenu is that of revealing the coverage to the user. In one of the interfaces we built in the lab, the user had the option to have objects displayed on a map—an alternative to having coordinate positions output in tabular form. He also had the option of displaying the map with latitude and longitude grid lines. In a conventional natural language system, it is quite possible that a user could query the system for some time, and never happen to discover the graphing option or the grid line option. It might never occur to him. However, with an NLMenu interface, the graphing option and grid option appear in active menus when the context is appropriate for them. In this way, users have a better chance of making full use of the capabilities of the system.

NLMenu interfaces require less memory and processing than do conventional natural language systems. They do not need to sift through large grammars and dictionaries to analyze sentences. We also have found ways of expressing database queries in such a way that the interfaces can largely be generated automatically from a description of the database. In fact, the interface for the sample dialogue was generated from a description of the database.[6] The generation process requires a description of the names of the relations (in this case a relational database was used), their attributes, and the characteristics of the values of the attributes (whether they are numeric, alphabetic, etc.). From this information, an interface is generated.

NLMenu has another advantage that goes beyond conventional natural language technology: It allows for more flexible specification of database values.[7] In a conventional natural language system, input is essentially limited to natural language. This is partly due to the fact that it is not clear how to mix input modes in typewritten natural language. In an NLMenu interface, on the other hand, it is easy to allow the user to input database values in whatever form is most convenient. In one system we built, the user needed to specify the location (latitude and longitude) of airports. The user was given an option: He could either enter the latitude and longitude textually, or he could ask for a map. A map would appear and the user would draw a box around the area of interest. The map would then disappear and coordinates of the box would be inserted textually into the query.

NLMenu has the flexibility to allow the user to specify values in whatever form is most appropriate: graphics, form filling, menu selection, typed input, or any other mode. This seems to allow the user to express himself more "naturally" than limiting him to typed natural language. It also presents the opportunity to input values in appropriate ways that would tend to reduce gross input errors.

CONCLUSIONS

As we have discussed above, NLMenu has many advantages over conventional natural language systems.[8–10] It has the same expressive power as conventional systems, but solves the biggest problems that natural language systems have.

One question that is frequently asked is whether NLMenu understands language. I think there are two answers.

If conventional natural language systems understand language, then NLMenu must also. It uses the same technology as they do, represents and translates questions in the same way that they do. Behind the menus, one cannot tell the difference between these systems. Assuming that one says that conventional systems understand language, the answer is "yes."

The other answer to the question is "Who cares?" This is a technology, and the appropriate forum for evaluating technology is in solving problems. If it provides a flexible, mnemonic, and powerful interface, what difference does it make if we declare that it does or does not understand language?

REFERENCES

1. Tennant, H. R. *Natural Language Processing, An Introduction to An Emerging Technology*. Princeton, N.J.: Petrocelli Books, 1981.
2. Tennant, H. R. Ph.D. Dissertation, Department of Computer Science, University of Illinois, 1980.
3. Tennant, H. R. "Experience with the Evaluation of Natural Language Question Answerers." In *Proceedings of the 6th International Joint Conference on Artificial Intelligence,* Tokyo, 1979, pp. 874–876.
4. Tennant, H. R. et al. "Menu-based Natural Language Understanding." In *Proceedings of the Conference of the Association for Computational Linguistics,* Cambridge, Mass., 1983, pp. 151–158.
5. Tennant, H. R., K. M. Ross, and C. W. Thompson. "Usable Natural Language Interfaces through Menu-based Natural Language Understanding." In *Proceedings of the Conference on Human Factors in Computing Systems,* Cambridge, Mass., 1983.
6. Thompson, C. W. et al. "Building Usable Menu-based Natural Language Interfaces to Databases." In *Proceedings of the 9th International Conference on Very Large Databases,* Florence, Italy, 1983, pp. 43–45.
7. Thompson, C. W. Ph.D. Dissertation, Department of Computer Science, University of Texas at Austin, 1984.
8. Grosz, B. et al. "TEAM: A Transportable Natural Language System." Technical Note 263, Menlo Park, Calif.: SRI International, April, 1982.
9. Harris, L. "Experience with ROBOT in 12 Commercial Natural Language Database Query Applications." In *Proceedings of the 6th International Joint Conference on Artificial Intelligence,* Tokyo, 1979, pp. 365–368.
10. Hendrix, G. and W. Lewis. "Transportable Natural Language Interfaces to Databases." In *Proceedings of the 19th Annual Meeting of the ACL,* Stanford, Calif., 1981, 159–166.