1. Natural-Language Interface

Gary G. Hendrix, Chairperson

SRI International
Menlo Park, CA 94025

Panelists

Jaime G. Carbonnell, Carnegie Mellon University
Jeff Hill, Artificial Intelligence Corporation
Aravind Joshi, University of Pennsylvania
Jerry Kaplan, Stanford University
Earl Sacerdoti, Machine Intelligence Corporation
David L. Waltz, University of Illinois
Stan Petrick, IBM Corporation

1.1 The Interface Problem

A major problem faced by would-be users of computer systems is that computers generally make use of special-purpose languages familiar only to those trained in computer science. For a large number of applications requiring interaction between humans and computer systems, it would be highly desirable for machines to converse in English or other natural languages familiar to their human users.

Over the last decade, in laboratories around the world, several computer systems have been developed that support at least elementary levels of natural-language interaction. Among these are such systems as those described in the several references at the end of this paper.

1.2 Proven Capabilities

Natural-language (NL) interfaces built so far have primarily addressed the problem of accessing information stored in conventional data base systems. Among the proven capabilities exhibited by these systems are those that:

- Provide reasonably good NL access to specific data bases
- Access multiple, remote data bases.
- Answer direct questions.
  ("What is Smith's salary?")
- Coordinate multiple files.
  ("What is Smith's location?" translates into "What is the location of the department of Smith?.")
- Handle simple uses of pronouns.
- Handle many elliptical inputs.
  ("Where is John? Sam?")
- Do basic report generation.
  ("By sex and age, list the salary and title of employees in New York.")
- Extend linguistic coverage at run time.
  (Define "JD" as "Jefferson Davis Jones" Let "Q1 Smith salary" be like "What is the salary of employee Smith?" "Q1 JD AGE?")
- Analyze NULL answers.
  ("How many Japanese carriers have inoperative air search radar?" Response: "There are no Japanese carriers.")
- Restate in English the system's interpretation of inputs.
- Correct spelling errors.
- Enhance the data in a data base with special-purpose functions.
  (E.g., calculate distances between cities.)

1.3 Prominent Potential Applications

Among the many promising prospects for NL interfaces, in increasing order of perceived difficulty, are interfaces to structured data bases, simulation models (e.g., VISICALC), operating systems, expert systems, transaction systems (e.g., airline reservations), and text data bases.

1.4 Factors for Suitable Applications

For NL-interface methodology to be suitable for a given application, the construction of such an NL interface must be technologically practicable. Moreover, it should make a positive contribution to the achievement of pragmatic goals in the area of application.

For NL-interface methodology to be technologically practicable now in a specific area of application:

- There must be a solid system to interface with. (Garbage accessed is garbage retrieved.)
- The application domain must be conceptually well-bounded, i.e., there must be a limited number of objects in the domain and a limited number of relationships among them.
- The domain's objects and relationships must be well-behaved. It is relatively straightforward to deal with concrete objects such as ships and employees, but such intangibles as human goals, beliefs, plans, and wants present very serious problems.
- It is desirable that truth regarding the domain be determinable through evaluation. Current techniques falter, for example, if the system must deal

with the fact that "either P or Q is true," without knowing which is true.

- System users must have reasonable expectations of what the system can do. This is largely a question of the level of abstraction at which a user wishes to interact with the system. For example, a marketing data base may easily be built to answer the specific question "What were our sales in May?" It is far harder to build a system with the abstract reasoning needed to handle "Why are sales slumping?", which is perfectly natural to ask in exactly the same domain as the previous question.
- Users must be able to type (at least until speech technology in combination with NL-interface technology makes it unnecessary).

For NL-interface methodology to be useful in a given area, the application must require flexibility and diversity. If the same report is to be produced every month, there is no particular advantage to requesting that it be printed by giving instructions in English. However, if there are hundreds or thousands of possible independent operations that a system might be called upon to perform, it may then become a very difficult task to indicate precisely which operation is desired. English is much better suited for a task of such complexity than menu selection systems, and it is easier to learn and remember than a sophisticated formal language.

Natural language may be of value even if there are only a few dozen operations to be discriminated, provided the operations are not performed very often. For example, a travel reservation system might have over 100 kinds of operations – too many for function keys alone. For the person who uses the more obscure operation only occasionally, natural language could be helpful.

The cost of creating an NL interface must be justified either by the system's volume of usage, or because utilization of the interface expedites access to data or other computer-based resources when time is a critical factor.

In summary, NL-interfaces are suitable for use in applications in which the personal or financial cost of learning a special purpose formal language may exceed the value of the information retrieved. This is most likely to occur in situations where the typical user has only one (or a few) queries at irregular intervals, needs to use the system only infrequently, is unfamiliar or uncomfortable with formal languages, or has only a partial understanding of the structure or content of the underlying system.

1.5 Advantages of NL Interfaces

Among the major advantages of NL interfaces are the following:

- Natural language is flexible.
- People need little training to use it in interfacing with a computer system, and they have very little difficulty remembering it (especially as compared with remembering the syntax or reserved function terms of formal languages).
- Natural language pronouns, quantification, and contextual conventions make it easy to perform a number of operations in natural language that are difficult or impossible in other languages. For many applications, the use of natural language is faster than using a menu system, composing formal queries, or writing computer programs.
- Natural languages allow follow-up questions to build on the linguistic context established by previous dialogue.

It is important to recognize that natural-language interfaces typically solve two problems simultaneously in providing users with access to computational resources.

- They can deal with a natural language articulation of what the user wants.
- They can transform a description of what the user wants into a computer program that specifies how to accomplish it.

The aspect of automatic programming provided by the second function is, for many potential applications, at least as important and useful as the ability to deal with natural-language syntax.

The automatic-programming aspect of many NL-interface systems is a key benefit of the interface technology, in that it provides a means for reducing the high labor cost of using humans to program computer algorithms to grapple with the inevitable ad hoc problems that arise in conjunction with any application system.

In general, the primary utility of NL interfaces is that they support the user's view of the domain and the application system. In other words, they transform the user's concepts into those actually used by the application system – and do so in a matter of milliseconds. NL syntax provides helpful support, but it is not necessarily the crucial feature that makes these systems useful. Other types of interface systems can also transform the user's conceptualization – NL systems do it by virtue of their essential nature.

1.6 Disadvantages of NL Interfaces

Natural language is unsuitable for some applications because it provides flexibility at the cost of verbosity. Text editors exemplify a type of system in which the commands are limited in number, used very frequently, and, conforming to user preference and convenience, are kept as short as possible. However, one can well imagine wanting to use "plain" English for the less common commands or to ask the system for assistance, e.g. "how do I change the margins?"

A system that uses natural language does not "wear its constraints on its sleeves", i.e., the system's capability is not reflected in the input language. This means that users can easily pose questions or give commands that are beyond the ability of the system to interpret. This is in contrast with a menu system, in which the system – always in control of the conversation – constrains the user to select from a limited number of choices. Whether or not explicit constraints are useful depends largely on the particular application.

Perhaps the main disadvantage of NL systems is that people tend to assume they are "smart." For example, if a system can provide NL access to a data base of information, users will tend to believe it can deduce other facts from that information – facts that, although not explicitly encoded, would be obvious to anyone with common sense. More formal systems are not expected to perform common sense reasoning, because their functionality, and therefore their inherent limitations, is readily apparent to the user.

1.7 Problems and New Techniques

1.7.1 Three Lines of Research and Three Levels of Systems

There are three major lines of research on natural-language interfaces:

- To make interfaces more easily transportable to new applications.
- To increase the linguistic coverage of systems.
- To increase the conceptual coverage of systems.

These three lines are so tightly intertwined that research focusing on one almost inevitably involves research on all three.

The systems already created and the diverse facets of ongoing research can be divided into three levels of complexity. To elucidate the various extensions and new techniques that have been developed in NL interfaces, let us define and discuss each of these levels in turn.

1.7.2 Level 1 Systems

Primary Characteristics

The primary characteristic of a Level 1 system is that the interface per se incorporates only an extremely limited theory of the domain of application.[1] In particular, the interface may have access to taxonomic information about the sorts of objects in the application domain, and may have information about the names of relationships in which those objects may participate – but it will not have knowledge of specific instances of relationships among objects. (That is, it may know that an employee is a person and that "salary" is the name of an attribute relating employees, but it will not explicitly encode the fact that John's salary is $30,000.) As a rule, it will rely on a conventional, external data base as its sole source of such information.

Translation in a Level 1 system is most often made directly into a data base query. Seldom is there an explicit representation of what the user actually said.

Most NL systems created to date are of the Level 1 category. Moreover, systems at this level are the only ones currently available that are sufficiently fast and robust to be considered for serious applications. In particular, the INTELLECT system (A.I.Corp., 1981), which is typical of a Level 1 configuration, is the only system currently available commercially.

Some Level 1 Extensions

The paragraphs below describe some relatively simple and inexpensive extensions currently under development that should enhance the utility of Level 1 systems.

Transportability. There is general agreement that the greatest problem now facing Level 1 systems is how to make effective use of existing techniques on new sets of data. Work is under way on methods to ease the transport problem, including work on using interactive dialogues with data base administrators (Hendrix and Lewis 1981) for automatic acquisition of the information needed to create new interfaces.

Database Enhancement Tools. As illustrated by the following examples, it is often desirable to extend the conceptual coverage of an interface to include access not only to primary data, but also to functions that can compute information derivable from those primary data.

Where is the Fox?
(Can be looked up directly in the data base.)
How far is she from LA?
(The locations of Fox and LA can be looked up. But the distance between them must be computed.)
How soon could she get there?
(The time needed to travel the distance must be computed, taking into account that ships cannot

[1] The word "theory" is used here to mean a description of a domain represented in some formal language. Such descriptions are sometimes called models. However, the word model is perhaps more properly used to refer to complete descriptions of a domain, so that an object or relationship exists in the domain if AND ONLY IF it is included in the model. A theory of a domain may be less precise. For example, it could specify that either P or Q holds in the domain, without specifying which.

follow straight courses if they pass over land masses, ice, or shallows.)

Developers and users of interfaces need special tools to make it easier to create enhancement functions and integrate them with the language-processing capability.

Database Update. Many users would find Level 1 systems more serviceable if they could be employed not only for querying data bases, but also for updating them (Kaplan and Davidson 1981). Admittedly, this can introduce problems. Suppose the user says

CHANGE BOB DAY'S LOCATION TO BLDG. 7.

If the data base is constructed in such a way that an employee is associated with a department, a department is associated with a location, and an employee's location is assumed to be that of his department, then processing the user's request will either result in a change in the location of the department (and all its employees) or force a reorganization of the data base.

Multilevel Systems. For a number of applications, users would like to be able to access data bases by means of any one of several query systems at various levels of relative convenience, precision, and efficiency. For example, we might imagine a natural-language system that:

- Accepts English questions and translates them into predicate calculus.
- Translates predicate calculus into data base queries in a formal language that does not require the user to know the structure of the data base.
- Translates such queries into a formal language that specifies particular joins between generic files.
- Translates those queries into specific codes that prescribe the order in which joins are to be made on particular physical files.

Users could interact with the underlying data by asking questions at any one of these levels. This feature is available in Waltz 1978 and Hendrix et al. 1978.

Context Setting. Users would find it convenient to restrict the context of evaluation. For example, after saying "Consider only US ships," the question "What ships are in the Med" would retrieve only US ships in the Med. This feature is available in Thompson and Thompson 1975.

Graceful Failure. With little effort, the response of most Level 1 systems to failure could be greatly improved. Ideas for graceful failure include Codd's 1974 notion of rendezvous, flexible parsing (as in Carbonell and Hayes 1981), and intelligent responses to null answers (as in Kaplan 1979 and Mays 1980). Work on ungrammatical and unparsable sentences as in Kwasny and Sondheimer 1981 is also relevant here.

Concise Responses. The answers provided by Level 1 systems could be made more intelligent in some special cases. For example, if asked, "Who has a company car?" a smart system might answer "The president and the VPs," rather than produce a list of the names of the president and vice-presidents. (An "outsmarted" system might answer "Employees in the ADM Building with ID-NUMBERs less than 1072 whose SPOUSE-MIDDLE-NAME is Jane.")

Metaquestions. It has become obvious over the last few years that users of natural-language interfaces to data bases desire far more than mere access to the data therein. There are a number of types of "metaquestions" they would like to pose as well. Among them are the following:

What information is in the db?

What are the allowable values for employee job titles?

How timely are the sales data?

How were they acquired?

How reliable is the source?

Can you handle relative clauses?

Can you handle a sentence that has a relative clause?

Why might the ship not be ready (causal relationships)?

Some steps in this direction have been taken in McKeown 1980 and Hendrix et al. 1978.

1.7.3 Level 2 Systems

Primary Characteristics

Level 2 systems must include an explicit theory of their domain of application (or be able to acquire one "on the fly," as in Haas and Hendrix 1980!). That is, they incorporate internal representations of some of the objects in the domain, as well as explicit knowledge about particular instances of relationships among those objects. The general paradigm of these systems is to:

- Use an explicit theory of the application domain to control all processing.
- Translate into an intermediate logical form, rather than into a db query.
- Provide access to multiple knowledge sources.
- Use deduction techniques to aid translation and fact retrieval.
- Provide discourse models for noun-phrase resolution.
- Allow explicit descriptions of events with complex histories.
- Follow the course of processes to determine the changing physical context against which noun-phrases must be resolved.
- Provide for dynamic data bases.
- Use constraints to check the validity of data.

A key facet of Level 2 systems is that they use knowledge about particular individuals and their specific properties to help resolve definitely determined noun phrases. Level 2 systems may also have discourse models that draw upon a source of knowledge as to which objects have been mentioned recently or are otherwise in focus (see Grosz 1977) because of their particular properties and the structure of the discourse.

To contrast Levels 1 and 2, consider an NL interface to a railroad's information management system. In a Level 1 system, a question such as "Where are the boxcars?" will always mean "Tell me the locations of ALL the boxcars in the data base." In a Level 2 system, if we have just indicated that we want to make up a train from the rolling stock in Yard 3, only boxcars in Yard 3 will be retrieved. In a Level 1 system, if we ask "What are the numbers of the cars?" we will get numbers for all cars of all types. In a Level 2 system, only the boxcars in Yard 3 would be selected.

Systems of the Level 2 type are becoming more common in laboratories and are likely to provide the basis for more sophisticated interface systems in the future. Level 2 ideas are also being developed independently by researchers concerned with data base systems per se (such as Wiederhold, Chen and McLeod).

The Intermediate-Language Problem

It is generally easier to produce a Level 1 system that can cope in at least an elementary manner with some arbitrary phenomenon than it is to produce the corresponding Level 2 system. This has led to the belief in some quarters that ad hoc systems are actually more flexible than linguistically motivated systems.

This is probably a distortion of the true situation. Level 1 systems translate directly into calls on software, whereas Level 2 systems force all inputs into a uniform, intermediate logical form. The result is that Level 2 systems deal with a linguistic phenomenon relatively well, once they deal with it at all, but they are forced to confront a general case before they cope with any specific instance. Level 1 systems can quickly accommodate simple new extensions, but tend to have very uneven linguistic coverage. As more extensions are added, an unwieldy "house of cards" is created which soon collapses under its own weight.

Beyond Conventional Data Bases

Level 2 systems are well equipped to move beyond the relatively simple problems of interfacing with conventional data bases composed of atomic facts. In particular, Level 2 systems may be easily interfaced with data bases capable of storing any well-formed formula in first-order logic – namely, with a much richer body of information than is available in conventional data bases.

We may also envision Level 2 systems that use multimedia communication, e.g., the combination of natural language with graphics and pointing.

Limitations of Level 2 Systems

Although Level 2 systems contain explicit theories of objects and relationships in their application domains, they do not contain explicit theories of the knowledge, goals, or plans of external systems, such as external data bases and the user. Consequently, Level 2 systems are incapable of reasoning about the intention of user inputs or about how to use external data bases. If external data bases (or other knowledge sources) are used by such systems, the attachment to them is provided through procedures whose knowledge of the external systems is implicit in the computer codes themselves and is thus unavailable for meaningful examination by the system.

Basic Stumbling Blocks

Even for Level 2 systems, many fundamental problems remain unanswered. Much of the deficiency centers upon the current inability of computers to represent and reason about time and space, substances, collective entities, events, actions, processes, nonstandard quantifiers, propositional attitudes, and modalities. These are thorny problems that philosophers, linguists, mathematicians, and computer scientists have been wrestling with for many years. Their solution is not likely to come easily.

1.7.4 Level 3 Systems

Principal Characteristics

Level 3 systems contain explicit theories of external agents, including information about their knowledge, goals, and plans. Some possible agents are the user, various data bases, human experts, and other software systems. Level 3 systems always translate into an explicit representation of what a user has said; that analysis then becomes the starting point for reasoning about:

- What the user meant.
- How to use internal and external resources to acquire information needed to respond to the user.
- How to communicate with the user's implied goals.

Eventually, we may see Level 3 interfaces emerging as a kind of information broker. Such a broker would:

- Model multiple external entities (both human and mechanical).
- Communicate with each in his (or its) own language.
- Use and explicitly understand goal-directed behavior.

What we have at present is only a start toward building systems of this level of sophistication. If we can build them at all, they will no doubt be many times more expensive computationally than the Level 1 and 2 systems now available.

1.7.5 The NL-Interface Problem in Perspective

If we are ultimately to achieve our long-range objective of producing genuinely fluent natural-language interfaces to computer systems, we must recognize and pay special attention to the fundamental problems of language understanding. It has recently become evident that the use of natural language must be studied as one facet of a general theory of goal-directed behavior. In particular, to use natural language fluently, a system must understand how the communication process itself is affected by language users' goals, plans, and beliefs.

So that the field of Computational Linguistics may benefit our society in both the short and the long term, it is important to continue work at all three levels of systems described above. Because it has only recently become technically feasible to consider the actual construction of a Level 3 system, special consideration should be given to launching a research program at that level.

1.8 Recommendations

The Natural-Language Interface Panel of the Workshop on Applied Computational Linguistics in Perspective made the following recommendations to the sponsors:

- Identify a specific DoD data base amenable to Level 1 technology and, using proven techniques, support the construction of an interface to it.
- Support AI core research on knowledge representation, acquisition, and use.
- Support basic work on the use of natural language as goal-directed behavior.