S. R. Petrick

On Natural Language Based Computer Systems

Abstract: Some of the arguments that have been given both for and against the use of natural languages in question-answering and programming systems are discussed. Several natural language based computer systems are considered in assessing the current level of system development. Finally, certain pervasive difficulties that have arisen in developing natural language based systems are identified, and the approach taken to overcome them in the REQUEST (Restricted English QUESTion-Answering) System is described.

Introduction

The potential use of a natural language to facilitate human interaction with computers has been discussed for over a decade in a series of papers and panels [1-9]; the participants have disagreed about the feasibility and even the desirability of natural language programming and question-answering systems. Although it is beyond the scope of this paper to review in detail all of the arguments that have been marshalled on this topic, the paper identifies some of the key issues and explores the extent to which they have been dealt with in recently developed experimental question-answering systems.

It is not our purpose either to describe in detail current question-answering systems. Instead, we assume some familiarity with the systems considered and concern ourselves principally with the inherent difficulties in man-machine communication and with how they are treated in those systems. In particular, we are interested in estimating the coverage of English that has been achieved in various applications. In discussing these difficulties, special attention is given to the approach taken to overcome them in the REQUEST (Restricted English QUESTion-Answering) system [10-13]. Thus, a rationale and introduction to REQUEST are provided, and the reader is directed to the paper in this issue by Plath for more detailed information about this question-answering system [12].

Natural language pros and cons

The proponents of natural language communication with a computer support their position by some combination of the following claims:

1. A large number of people who are potential computer users are unwilling to learn and then use a formal machine language.
2. For at least some computer applications, natural language provides an ideal medium of communication for everyone.
3. Extrapolating from the capabilities of existing natural language based systems, current technology appears to be sufficiently advanced to support useful computer systems that accept natural language input, or
4. To the extent that current technology is not sufficiently developed for that purpose, the remaining problems can be solved and ought to be considered now.

In reply to these arguments, those who disparage the use of natural language make the following claims:

1. The most difficult aspects of a problem are formulating it precisely, analyzing it, and planning the method of solution in detail. Actual code production is relatively straightforward and easy.
2. Natural language is inherently too loose, vague, and ambiguous to serve as a computer language. For this reason its use would lead to processing inefficiency and possible error due to misunderstanding of intended meaning.
3. Allowing the use of unrestricted natural language is technically unfeasible and likely to remain so in the foreseeable future. Consequently, subsets of natural languages must be used for communicating with computers. These subsets would be harder to learn and use than traditional formal computer languages because of interference with natural language usage habits.
4. Providing a large enough subset of a natural language to be useful is an exceedingly difficult intellectual activity, requiring not only a far greater command of linguistics than is likely to be available for many years, but also requiring capabilities for representing an enormous quantity of information about the world and for efficiently drawing deductive and inductive conclusions from that information.

Pro claim 1, citing the existence of large numbers of people who would use a computer only if this were possible without learning a formal programming or query language, appears to be basically correct. But although this supplies motivation for providing those people with a natural language input system, it does nothing to either confirm or disconfirm the possibility of implementing such a system. And, of course, the assertion assumes the type of formal programming and query languages that presently exist. The use of natural language is not the only possibility to extend the class of computer users. Improved artificial languages are an alternative that must be developed and evaluated. Query by Example [14] is one instance of such an artificial language whose utility for different applications and sets of users must be compared to that of English-based query systems.

Pro claim 2, that natural language is well suited for at least some computer applications, appears to be warranted. In the course of reviewing the information retrieval and data base management requirements and plans of a large company, the author obtained some insight into this situation. Questions were submitted in written English to clerks who translated them to formal query language equivalents, processed the formal queries on the computer, debugged and resubmitted the queries if necessary, and returned the resulting answers to those who submitted the questions. Typical of such questions were:

Who is the buyer on Purchase Order H2394?
What is the total amount of dollars outstanding on purchase orders for Supplier 20035?
Were any pieces received from Supplier 26 in January 1966 discrepant?
What are the quantities ordered and the balances due on all open orders for Part Number 50475?

Clearly, if the questions could have been processed in their original form, much turnaround time as well as human effort might have been saved. Whether or not computer systems can be implemented to accept such questions (with some prompting for clarifying information, perhaps) in the form in which they are naturally formulated, however, is a separate issue to which we turn in a moment.

Even though a few practical applications appear well suited for natural language input, most applications that come to mind offer poor prospects for natural language input. To date there have been few instances of natural language programming, question-answering, or data base management systems where efforts were directed toward a truly practical application. Most natural language question-answering systems have dealt with toy problems and/or data bases for which there exists no body of present or potential computer users who have questions or commands in which they are vitally interested. Notable exceptions include the LSNLIS system [15, 16] and certain applications of the REL system [17, 18]. The first application that we selected for the REQUEST system [10–13] is deficient in this respect; the data base for that application is real—*Fortune* 500-type business statistics, but we know of no body of users who might wish to interrogate information of this type on a continuing basis. A drawback of such applications is that it is difficult to evaluate the question-answering systems to which they give rise. Some of the previously enumerated counterarguments to the use of natural language inputs can only be refuted by implementing systems of demonstrated practical utility.

What then are the characteristics of those applications that appear to be good candidates for consideration at this time? First, the application should be one with a large number of potential users, probably users with a good knowledge of the application but very naive about programming or even algorithmic analysis of their application. A high volume of diverse inputs is necessary to justify the effort of developing a natural language front end; moreover, this volume must be distributed among a large number of users because a small number of people could be economically trained to learn and use a syntactically more restricted formal programming or query language. Of course, for some applications, even if there were a relatively small number of users, their natural reluctance to learn a formal language might only be overcome through the use of a natural language capability.

The observation that potential users are likely to be well versed in their application but computationally naive has some implications for natural language programming systems. For one thing, these systems should be problem-oriented, not procedural. They should make use of application-specific knowledge and algorithms. Furthermore, causal users will not even be aware of all the information that those algorithms require as input, so provisions for computer-prompted dialogue are necessitated. A discourse capability is also desirable in a question-answering system but not as essential as in natural language programming. That is, although the lack of a discourse capability may make a question-answering system clumsier and less convenient, it does not rule out a useful system altogether as it would in the case of natural language programming, where single sentence programs of any complexity are intolerable.

This difference between natural language programming and question-answering brings up the matter of the extent to which they are essentially different. We contend that there is no fundamental difference. The sentence-by-sentence analysis to determine underlying relationships that characterizes question-answering must also be carried out in natural language programming. It is even reasonable to expect that a good interactive programming system will allow users to pose clarifying questions to the system. Question-answering systems often start out simple, but as soon as discourse facilities are added and the ability to manipulate as well as simply retrieve data is provided, the line between question-answering and programming becomes fuzzy. Thus in talking with potential users of the REQUEST system it was clear that, although they would be pleased to have it adapted to their application in order to make use of its present retrieval capabilities, they were even more interested in its extension along lines customarily associated with natural language programming. Some other current systems illustrate the fuzziness of this programming question-answering distinction. Thus SHRDLU [19, 20] not only answers such questions as "How many things are on top of green cubes?" but also provides for commands such as "Put the blue pyramid on the green block," and by a sequence of such commands it can construct complex block structures. Similarly, although REL is often thought of as a question-answering system, it provides for ways of performing computations on selected data and assigning names to the results for subsequent further processing. It meets all of the requisites to be considered a natural language programming system. And finally, the NLP system [21-23], which was designed for use in building and running simulation models in the manner of GPSS or SIMSCRIPT, also allows its users to pose such questions as "Are there three pumps?" and "How often do cars arrive?"

Another requirement that ought to be placed on current application candidates is that they demand relatively little in the way of inferential capability. Inference may turn out to be important, and indeed can now be seen to be vital for certain applications, but there are many applications of practical importance that do not require such capabilities. In a wide range of applications an inferential facility currently appears to offer a very small additional capability at a very high cost.

About a year ago a search was begun for a new application for REQUEST that meets the criteria discussed above. Our motivation was to test the system in a practical application of importance to a number of users and also to determine the level of effort required to adapt REQUEST to a new application. After considering several potential applications, we have concluded that, although many data base applications are inappropriate for REQUEST, there are applications which appear quite well suited. We are currently focusing on data bases of interest to city and county government personnel. In particular, the interrogation of land description, utilization, and taxation records by city planners, assessors, title searchers, and individual property owners looks promising, and we have recently initiated a joint effort with a local municipality aimed at developing such a capability.

Evaluation of some current systems

If we return to the arguments previously cited for and against the use of natural language, a pair of contradictory claims come to our attention. Proponents of natural language systems cite the success of prototype systems and claim the time is ripe to construct large practical natural language systems. However, those who oppose such systems claim that our current knowledge cannot support an undertaking of such difficulty. A perusal of the natural language question-answering literature indicates the reason for these contradictory claims. (See [10-13] and [15-29] for some of the more recent papers on this topic.) With the single exception of the LSNLIS system there have been no attempts to evaluate the capacity of a natural language question-answering system to satisfy the needs of the user community for which the system was designed. Furthermore, there have been few attempts to characterize the extent of the syntactic and semantic coverage of English provided. Lists of sentences successfully processed, syntactic constructions considered, and occasional discussion of problems encountered and acknowledgment of coverage gaps provide the only basis for evaluating system capabilities, and these tend to be either missing or sketchy. Furthermore, the primary source of information about system capability, the grammar and/or program that specifies the set of acceptable inputs, is not usually useful for the interested investigator. Even when it is fully included in a report, it is large, complex, unannotated, and in most cases based on a linguistic theory and formalism that are alien to the would-be investigator.

What is usually provided is a small set of "representative" inputs that were successfully processed at some stage of system development, together with a more or less detailed description of the steps involved in their processing. This is, of course, useful in explaining the algorithms in question because the computer programs that constitute their primary specification are incomprehensible to almost everyone. However, it is not useful in characterizing the syntactic and semantic coverage that a system provides, i.e., the system's capabilities. On the contrary, the consideration of a few examples usually leads to unwarranted extrapolation about system capabilities. This is natural because the reader of a paper who observes sample sentences in which, for example, conjunction, negation, and quantification occur, assumes that these phenomena can be successfully analyzed when they occur in different ways involving no additional lexical items; unfortunately, this assumption may be wrong. This determination, however, can usually not be made from the information provided in question-answering papers. Information about whether a given sentence is acceptable to a question-answering system can only be provided by the system implementor (reliably in at least certain negative cases) or through the use of the system itself. Unfortunately, neither source is readily accessible. Our concern about the coverage provided by most natural language question-answering systems is based on a first hand appreciation (gained from working on REQUEST) of the difficulties involved, the absence of concrete coverage claims, and some knowledge of the coverage provided by a few other representative systems. The author has discussed specifics of coverage with those responsible for the REL, SHRDLU, NLP, and LSNLIS systems and to a limited extent has submitted sentences for consideration and has talked with others who have had the same opportunity. We have not had the chance to systematically explore the coverage of those systems at length, and so there undoubtedly are allowable constructions for each system that we never discovered. It is likely that each system considered specifies some well formed sentence types not provided for by any of the others. In only a few cases do we have computer listings of our attempts to explore the use of those systems. Moreover, our contact with them was spread over a period of several years, and, hence, we have more specific recollections of some than of others.

1. The LSNLIS (Lunar Sciences Natural Language Information System) [15, 16] contains information about the lunar rock and soil samples that were accumulated as a result of the Apollo moon missions. It is an experimental question-answering system that was designed to enable lunar geologists to conveniently access, compare, and evalute these data. Its syntactic component is an augmented transition network grammar [30]. The structures assigned by this grammar, although referred to as "deep structures," are often closer to surface structure than to the deep structures utilized in transformational grammar theory (extended standard theory or generative semantic theory). A case in point is discussed in the penultimate section.

   Although sentences the author formed by combining observed constructions in novel ways were not always processed correctly by LSNLIS, the degree of success was encouraging. LSNLIS appeared to specify an interesting subset of English. This system is, of course, the only one that has been subjected to any degree of evaluative testing with respect to the class of intended users. It has been reported [16, page 5.2] that 78 percent of the initial queries posed by lunar geologists at a 1971 conference were processed correctly, 12 percent were judged to be answerable if simple changes or additions to the system were made, and the remaining 10 percent could not be answered for more deep-seated reasons. It is also interesting to note that in several talks Woods, the principal developer of the system, has cited the results of experiments which allowed lunar geologists to follow up their initial queries with a sequence of subsequent queries. A much larger percentage of these follow-up questions could not be processed.

   Augmented transition network grammars have been used by several research groups in the past few years, for the most part with some success. However, a recent paper by Wolfe [31] describes an unsuccessful attempt to use the LSNLIS grammar for the automatic generation of questions from text taken from a computer programming manual. Wolfe was probably overly optimistic in expecting any existing system to cope with unrestricted text. However, this was not the only problem he encountered. He reported processing times of more than an hour before unsuccessful termination of some sentences and claimed that "incorrect" structures were assigned to 40 percent of the sentences that were parsed. Unfortunately, it is not possible to assess the significance of these reported results without knowing the specific sentences in question and the structures assigned to them. LSNLIS, in common with all existing systems, is not suitable for such applications as Wolfe's, but it appears to be a promising system for extension to more appropriate applications, and we discuss it further in comparison to REQUEST in the sequel.

2. The REL (Rapidly Extensible Language) system [17, 18] provides a framework within which various extensible English-like languages and their associated data bases may be accommodated. It has been applied to such diverse problems as interrogation of anthropological data, class scheduling, and *Fortune* 500 data question-answering. REL's predecessor system, DEACON [32], was among the first to integrate syntactic and semantic components with an associated data base, and from the beginning REL's developers have been much concerned with achieving efficient retrieval by giving careful attention to such matters as novel data structures, tight low level coding, and paging. REL was also the first question-answering system to provide a form of user extensibility, based on string substitution, whereby new terms can be defined in English and subsequently be successfully processed by the system. This definitional capability is an essential feature of the REL approach to question-answering; REL provides a central English core language and relies on user definitions to expand the core to a customized dialect that is appropriate for a particular application.

Perhaps because of this approach to achieving generality and usefulness in a variety of applications, REL English gives the impression of being a somewhat unnatural formal dialect of English. Even sample inputs displayed in papers on the system clearly reflect REL's rather unnatural quality, e.g., "What is the number of 1957 Mazulu sample who are male?," "When did Jill have location New York?," "When each child of Moses Munsaje was born?," and "The clans of how many Mazulu machismos 57 are each clan?." In addition, many of the sentences the author proposed for REL system testing at the International Conference on Computational Linguistics in 1973 were not run in the form submitted but were instead paraphrased into what appeared to be un-English-like equivalents. REL has not been formally evaluated with respect to a group of users, but it has been used by a number of people at the California Institute of Technology, at least one of whom, anthropologist Thayer Scudder, is reported to have achieved useful results over a period of time [18, p. 114]. In all cases, one of the REL system staff members was at least initially available to assist those users with the formulation of their queries, and there are no data on the learnability of REL English under controlled conditions. Consequently, experiments in the learnability and ease of using REL relative to more conventional formal query languages are in order. Similar experiments are needed for other systems as well; only such experiments or the demonstrated satisfaction of a large number of users of some practically oriented natural language question-answering system will provide an acceptable answer to the third objection to such systems cited previously, i.e., that there may be possible interference between a natural language and a query system based on a natural language subset.

3. The SHRDLU system [19, 20] provides facilities for representing, querying, and graphically simulating the manipulation of objects such as toy blocks, pyramids, balls, and a box, all of which are arranged on a table. This system made an important contribution to the development of question-answering systems by demonstrating that it was possible to simultaneously bring together syntactic, semantic, inferential, and graphical capabilities in a single system. SHRDLU also offers a more highly developed English response generator than such systems as LSNLIS, REL, and REQUEST. For example, the question, "When did you pick it up?" was reported to be answered by "While I was stacking up the red cube, a large red block and a large green cube."

   The author presented a list of sentences to T. Winograd, developer of SHRDLU, to determine whether they could be successfully processed. On the basis of our discussion of that list of sentences, the syntactic and semantic coverage provided by SHRDLU appears to be spotty. Although a large number of syntactic constructions occur at least once in sample sentences appearing in published dialogue, our attempts to combine them into different sentences (involving no new words or concepts) produced few sentences that Winograd felt the system could successfully process. Linguistic sophistication was not required in this endeavor. The gaps that were encountered were attributed primarily to syntactic limitations. The actual users of the system with whom the author has spoken reported similar syntactic gaps and also mentioned encountering sentences that, although syntactically acceptable, produced anomalous computer responses. These observations suggest that SHRDLU has not been developed sufficiently to permit testing to determine its habitability [33], even though its block world offers an especially attractive possibility for posing verbal tasks to a user in a nonlinguistic fashion (i.e., by asking a user to issue the instructions necessary to transform the block configuration of a given figure into that of another figure). Winograd has expressed the belief that with work SHRDLU could be developed to the point where linguists could still fool it but where others would have no trouble communicating with it. However, this would appear to require a significant effort.

4. NLP [21–23] is the Natural Language Processing system that was implemented and then used to develop NLPQ, an automatic programming system for queuing systems. Details on both systems can be found in the paper in this issue [22] that compares them to three other automatic programming systems. NLP's syntactic component makes use of rules that are basically phrase structure grammar productions augmented with additional conditions on their applicability and with structure building translation actions to be taken if their corresponding productions are applicable. The conditions are primarily feature-based but may be more general. During a dialogue, information obtained from the user provides the basis for building a semantic network that specifies a flow of mobile entities through a queuing system. This network provides the basis for producing a GPSS program for running a simulation of the queuing system in question. NLPQ incorporates specific knowledge of queuing processes, which allows some information to be inferred and later corrected if necessary. It also features a sentence generating capability that permits the semantic network to be converted back at any point to an English description of the process to be simulated.

The author recently spent an afternoon using the system aided by Heidorn. NLPQ's syntactic coverage was not the central concern of its designer and hence is more limited in its attempted scope than that of the other systems considered here. Thus, no relative clauses are allowed, and noun phrases are restricted in other ways. Gaps of coverage also occur in rather natural simple sentences; thus the system accepted and subsequently generated our input "There are 3 pumps" but rejected "How many pumps are there?" and all the variants of this question that came to mind. Other examples of inputs rejected for syntactic reasons include "between 6 and 12 minutes" in response to the system inquiry "How long do the trucks unload at the dock?" ("from 6 to 12 minutes" was accepted, however), "The trucks go to a pump for service," and "Can I change some information?." The inability to process relative clauses made it difficult to refer to a stage in the queuing process where a change was required because of an erroneous inference made by the system. In addition, the system interpreted the sentence "10 percent of them unload at a dock" to mean that all trucks unload at a dock. NLPQ displayed no more "bugs" than the other system, however, and its English generating capability made possible the detection of several incorrectly understood inputs that would have caused problems if they had gone undetected. The extent to which NLP is suitable for specifying larger subsets of English in support of real applications remains to be shown (as it does for the other systems as well), but that suitability is currently being investigated in conjunction with a business application [34, 35].

5. The REQUEST system is based on a large and growing transformational grammar of English. Initial emphasis has been placed on accessing collections of tabular data. Generality has been achieved by providing both a parser that is valid for a class of transformational grammars and a semantic component which accepts semantic rules that can be tailored to the syntactic rules and to the application in question. Our experience in developing REQUEST indicates a fair degree of success in providing syntactic and semantic coverage of a restricted domain [36]. We have, however, provided fewer capabilities to facilitate dialogue than have a number of the other systems. There are no facilities for inter-sentence pronominal reference, extensible language definition, or complete English sentence response generation. At least some of these capabilities will undoubtedly have to be added in applying REQUEST to the needs of actual users, REQUEST's semantic interpretation component offers fewer problems than do those of other systems because a much larger share of the load of sentence "understanding" is borne by the transformational component. A further consequence of this approach is that our rate of system extension has been limited by the rate at which we can add new grammatical constructions to the repertoire of those that can be processed. As each new construction is implemented, we have taken great care to integrate it with previously allowed constructions to permit all grammatical combinations, thereby maintaining flexibility of usage. The price we have paid for this flexibility, however, is a slower rate of syntactic extension than we initially hoped for.

In spite of all the attention we have given to providing for alternative means of expression, we frequently encounter gaps in coverage. These take the form of words not included in the lexicon, constructions we have not yet considered, and constructions only partially provided for. We try to avoid the latter, but we sometimes succumb to a pressing need for an important construction whose thorough treatment must be deferred until later. Cases in point include comparative constructions, whose use was narrowly restricted for several years but is now being significantly extended, and conjoined constructions, whose use is still very restricted in the latest version of REQUEST.

Remaining arguments against natural language

One of the previously cited objections to natural language that has not yet been discussed is that analysis and planning are more difficult than coding, and the use of natural language is not helpful in this more important area. This argument is more applicable to natural language programming systems than to question-answering systems; but even in programming there are some applications for which we need low-level control and careful planning for reasons of efficiency and others for which we do not. The argument sounds like one that was put forth to support the view that high level programming languages would never displace the universal use of assembly languages. In any case natural language input as noted previously is better suited to problem-oriented input than to procedural input, and efficient low-level control is more of a problem in the latter than in the former case.

The previous discussion is also relevant to that portion of the second objection concerning efficiency. Another point raised in that objection, however, deals with natural language ambiguity and the possibility of misunderstanding. It is always a concern when using a higher-level programming language that the user and the compiler agree on the intended meaning of every line of source code written. The problem is basically the same when the higher-level language is a subset of English. However, unlike programming languages, which are designed and implemented to be unambiguous, natural language sentences (particularly if their context is ignored) are often ambiguous. Ambiguous sentences must be recognized as such and their ambiguity resolved, automatically if possible and otherwise by an appeal to the user for clarification. Unfortunately, most natural language systems have been based on syntactic structure that does not adequately represent meaning and, hence, fails to represent and deal adequately with the meanings of ambiguous sentences in particular. There are, however, some linguistic models that are better suited than others to deal with the ambiguity problem, and we believe that our transformational model is well suited in this respect. In our development of REQUEST we have tried to consider all possible ambiguities of allowable sentences and have attempted to incorporate them in our grammar and deal with them reasonably and correctly. For the most part we think we have been successful in this regard, but in a few important cases (notably sentences involving conjunction and quantification) we are aware of residual problems that must be solved. In addition to these problems of correctly handling certain types of ambiguous sentences, however, there is a more general, recurrent problem that we have not solved. Once we have recognized a sentence as being genuinely ambiguous in a particular context, there remains the task of exchanging information with the user in order to resolve the ambiguity. The logical forms which constitute our computer-interpretable representations of meaning are not suitable vehicles of communication with a casual user. At the same time, the use of semi-canned messages that are fleshed out from alternative logical forms seems to be too ad hoc, and a more general way to go from our internal representations of meaning to appropriate English dialogue is needed.

What can we conclude then as to the significance of results to date in the development of natural language-based computer systems? First of all, no such systems of proven usefulness have yet been produced. (A possible exception is REL, but productive use of this system has depended upon the assistance of REL personnel). Second, most of the systems developed to date have provided sparse syntactic (as well as semantic) coverage of English. This coverage has at least two dimensions—*breadth*, the diversity of syntactic constructions provided for, and *depth*, the degree of flexibility with which constructions may be combined in grammatically allowable ways. Most current systems are lacking in breadth and even more so in depth of syntactic coverage. At least two systems, however, LSNLIS and REQUEST, have provided enough depth to suggest that the third objection to natural language systems raised at the beginning of the paper can be eliminated if more breadth is provided. The question of how much effort is required to provide that breadth remains open. We feel that, given an adequate grammatical model, this effort is large but not prohibitive. Furthermore, we feel strongly that the effort should be directed toward an application that is of importance to some community of users, since only in this way can the practical potential of natural language question-answering systems be meaningfully evaluated.

Natural language system models

Several distinct models have been proposed and used in the syntactic and semantic components of natural language based computer systems. Syntactic models include transformational grammars, context-sensitive grammars, context-free grammars, general rewriting systems, and augmented transition networks. Also, the syntactic components of many systems are defined as syntactic analysis models rather than as more familiar generative models with which various syntactic analysis algorithms may be associated. Many of these analysis-specified systems employ what their developers refer to as "transformations." These are mappings of a string of trees (which exist at some stage of bottom-to-top context-free grammar parsing) into another string of trees, and not linguistic transformations as usually defined. The application of such operations is usually interspersed with phrase structure building. Such systems are thus not variants of transformational grammars but are essentially distinct models that must be independently motivated. The only systems that we know to have been based on generative transformational grammar theory are REQUEST and SAFARI [29].

Models for the semantic interpretation of syntactic structure are equally varied. The most commonly encountered "model" involves the issuance of calls to semantic subroutines interspersed with syntactic analysis, i.e., after the syntactic structure of certain types of phrases is obtained. SHRDLU's semantic component operates in this fashion. Another frequently employed model is that used by Irons in 1961 for ALGOL to assembly language translation [37]. In this model a translation rule is associated with every production of a syntactic component phrase structure grammar. These translation rules are applied from the bottom of a tree up to its root, and each rule assigns a "translation" to a nonterminal node as some function of the translations of its immediate descendent nodes. The Irons translation mechanism is thus seen to be similar to the use of Katz and Fodor's semantic projection rules [38] with respect to the way that both traverse the nodes of a tree, assigning translations (readings) to subtrees in the process. Both REL and NLP use this basic method of tree traversal as well. Note that it is not necessary to first find a syntax tree and then traverse it for the purpose of interpreting its meaning. Interpretation can be carried out on subtrees prior to connecting them in a complete syntactic surface structure. NLP works in the latter fashion, and REL can optionally be operated that way too. The resulting systems resemble SHRDLU with respect to the way in which they interleave syntactic and semantic processing. SHRDLU is less rigid about the order in which the surface tree is implicitly traversed, but it is correspondingly weaker about making any claims concerning the nature of all languages and the way they are understood.

A variation of the Irons model due to Knuth [39] is the approach to semantic interpretation used in REQUEST. It differs from the Irons procedure in providing for more than one "translation" to be associated with a nonterminal node. These "translations" and the names by which they are referenced are called *values* and *attributes*, respectively. Whereas in the Irons procedure a single translation rule is associated with every phrase structure production, in the Knuth procedure a set of translation rules is associated with every production. Each Knuth translation rule assigns a value to a node N by computing some function of the previously determined values of attributes associated with nodes in a *neighborhood* of N (where a node and its immediate descendent nodes are said to be in the same neighborhood). By means of the Knuth translation procedure the values of attributes can be passed down a tree as well as up a tree, and, hence, it is possible to translate a subtree differently depending on the larger context in which it occurs.

Still a different semantic component is used by Woods and Kaplan [15, 16], an adaptation of the semantic interpretation procedure presented by Woods [40]. In this procedure use is made of semantic rules that consist of tree fragment *templates*, which are the basis for pattern matching to determine rule applicability, and *actions*, which specify how semantic interpretation is to be accomplished. These actions take the form of schemata into which the interpretations of embedded constituents are inserted before they are evaluated. The value resulting from such an evaluation constitutes the semantic interpretation of the syntactic structure tree node corresponding to the semantic rule in question.

The better the syntactic component is in assigning structures that reflect underlying meaning, the easier is the task of the semantic component in translating those structures into computer-interpretable form. From this tradeoff in complexity one would expect to find relatively simple semantic components used in systems that expend considerable effort on computing good syntactic structures, and conversely. Surprisingly, however, there are several instances where simple semantic components have been coupled with weak syntactic components, which suggests that many systems are incapable of making up for syntactic shortcomings by means of increased semantic capability. We comment on this further in a later section when we consider the difficulties posed by a specific query.

Intermediate structural and semantic representations

Another way of comparing different language understanding systems is to discuss the different structural representations of input sentences that they employ. Descriptive phrases are not too useful in this regard because, for example, structures referred to as "deep structures" in one system may be closer to the surface structures than to the "deep structures" in another system. We do not discuss this topic in depth here, but we mention some of the intermediate structures for input sentences that are used in the five systems previously discussed.

LSNLIS directly assigns deep structures to sentences by means of an ATN grammar, and then its semantic component maps them to a semantic representation made up of quantified functions and propositions. REQUEST has explicit surface structures that are mapped into underlying structures more abstract (i.e., deeper) than those of LSNLIS by means of its transformational component. These underlying structures are, in turn, mapped by a semantic component into logical forms that consist of quantified propositions together with expressions denoting sets (either extensionally or intensionally represented). In evaluating these logical forms the LISP interpreter calls functions required for such purposes as data base retrieval.

SHRDLU does not explicitly produce either deep or surface syntactic structure, but at least implicitly defines surface structure. By executing semantic routines that are called in tracing through surface structure, PLANNER [41] expressions are built up. These expressions are, in turn, evaluated by a separate PLANNER system. PLANNER is a goal-oriented procedural language that was designed to represent information and make inferences based on it.

REL assigns structure that is basically surface structure but that can be and in some cases is deeper than surface structure. Kay's general rewriting system parser [42] is used to obtain it. Depending upon the user's preference, semantic subroutines that can completely interpret subtrees can be called during the course of parsing or, alternatively, intermediate output can be produced and later evaluated after parsing is complete. This intermediate output takes the form of a sequence of semantic subroutines to be executed and an indication of their arguments.

In the NLP system, surface structure is implicitly traced, and an Irons type of translation mechanism is invoked to produce a deeper structure that takes the form of sets of attribute-value pairs linked together in a semantic network. The network represents internal relations within single sentences and also certain relationships that go beyond single sentences such as the sequence in which actions contained in sentences are to be carried out. The network can be mapped back into a sequence of English sentences or, when problem specification is complete, into an appropriate GPSS program for subsequent execution.

Inherent difficulties of natural language understanding

In order to illustrate some of the difficulties of natural language understanding let us consider the query

(Q) Were GE's earnings greater than IBM's in 1973?

We understand this to refer to the amount of money that the General Electric Company earned in 1973 and to the amount of money that IBM earned in 1973, and we further understand the query to request information as to whether the former amount is larger than the latter amount. Notice, however, that in the query as written, "earnings" is missing from its understood position after "IBM's." In addition, "1973," which logically qualifies the year of both GE's and IBM' earnings, appears in only one place, at the end of the question.

As discussed more fully in Plath [12], question-answering systems whose syntactic component is a phrase structure grammar of any kind are hard put to account for the underlying meaning relationships from the scrambled and incomplete form that natural language input queries frequently take. Phrase structure grammar-based systems must decode intended meaning from surface structures—trees whose debracketizations are input queries and whose structure reflects the hierarchical grouping of the words and phrases of those queries. It is well known that the semantic interpretation of surface structure is extremely difficult. Wild ambiguity of surface structures assigned to unambiguous sentences is simply a special case of the general lack of correspondence between surface structures and intended meaning.

The structural descriptions that a phrase structure grammar would undoubtedly assign to query (Q) would probably analyze "GE's earnings" as a noun phrase, "IBM's" as another noun phrase, and "in 1973" as a prepositional phrase. At least two assigned structures might be expected, one in which the prepositional phrase combines with its adjacent noun phrase to make up the larger noun phrase "IBM's in 1973" and one in which the prepositional phrase is attached to a higher node, perhaps the root of the tree. We note that interpretation of the two noun phrases poses a problem, because information needed to determine their referents is not included in their assigned structures. The structure corresponding to "GE's earnings" is missing information as to the year in question, and the structure corresponding to "IBM's" is not only missing the year but also the fact that it denotes an earnings figure.

Any natural attempt to use the Irons bottom-to-top semantic interpretation procedure on such structures must fail. The noun phrases cannot be interpreted solely on the basis of their surface structure, and the only recourse is to avoid noun phrase interpretation altogether until we are interpreting the top node. This defeats the purpose of the Irons procedure, i.e., to interpret larger phrases as a simple function of the interpretations of their constituent phrases. The resultant top level interpretation rule must be very complex; one noun phrase lacks an essential piece of information and the other lacks two such pieces. Recognizing that this is the case is not easy, and simply providing surface structure does not go very far toward solving the problem of how to interpret this structure.

In view of the above difficulties we would not expect to find a relatively simple Irons-type of semantic component used in conjunction with a surface structure-based syntactic component. Often, however, just such a combination is found. REL and NLP are basically of this type.

Many systems avoid the problem of surface structure interpretation by utilizing syntactic components which assign structures to sentences that more adequately represent their meanings. Thus in REQUEST an underlying (deep) structure is assigned to query (Q) in which a verb "greaterthan" relates two noun phrase structures; the first of these represents "GE's 1973 earnings" in a form that can be paraphrased in English as "the quantity of money X such that GE earned X in the year 1973," and the second represents "IBM's 1973 earnings" similarly.

Although many systems have recognized the necessity of a syntactic component that provides more adequate structure than surface structure, the syntactic structures that they do provide vary widely with respect to the degree of their improvement over surface structure. A majority of these systems employ a few "transformations" to eliminate certain surface structure such as that associated with passive sentences, but their resulting "deep structures" still reflect a large number of phrase structure productions and hence pose the same semantic interpretation problems as the surface structure based systems.

In REQUEST we have placed relatively great emphasis on producing underlying syntactic structure at a level of abstractness that directly reflects meaning. In contrast, even those other systems that attempt to produce structures that represent the successively embedded simple declarative sentence structures advocated by transformational grammarians fall short of the level of abstractness that we believe is called for. Consequently, they must make up for this shortcoming with increased semantic complexity. Often, the semantic corrections for inadequate structure that are made work for simple cases but are not sufficiently general.

A case in LSNLIS that seems to illustrate this situation is Woods and Kaplan's treatment of prepositional phrases. As we have seen, the correct association of prepositional phrases with other structure is both difficult and crucial. In discussing this problem Woods and Kaplan write ([15], page 4.10), "It would be nice if the parser provided a syntax tree in which the various prepositional arguments of a noun phrase were attached directly to the noun phrase where they make sense semantically, and we have experimented elsewhere with a rudimentary facility for using the information in the semantic rules to guide the parser in the placement of prepositional modifiers. In the present system, however, we have taken the opposite tack and provided semantic rules which can locate the necessary prepositional arguments even when the parser has placed them in the wrong place." Woods and Kaplan also refer in several other places to inadequacies of modifier placement in their system. They acknowledge ([15], page 5.6) that the information necessary to achieve correct modifier position "is not in a format which makes it conveniently available to the parser for use in deciding where to put the prepositional phrase," and go on to say, "The parser in our present system, therefore, places the modifier in the syntactic parse tree as if it modified the 'nearest' possible construction . . . . To compensate for this characteristic of the parser, the semantic rules have been made smarter in order to find the modifier . . . even though it appears as a modifier of 'breccias' and not where it should be."

The authors' use of "smarter" in the previous sentence might well be read as "longer and more complicated." There is clearly a tradeoff between syntactic and semantic complexity, and it is overall simplicity and efficiency that are important. This must be kept in mind when critics of transformational grammar based systems claim they are too inefficient for practical consideration. It is very difficult to compare the efficiency of different systems due to disparity of the computers used, coverage achieved, size of data bases involved, etc., but these factors must be taken into account, and the comparison must consider complete systems due to the fact that the relative share of the load borne by the syntactic and semantic components of current systems varies widely.

A very limited exercise in comparing the LSNLIS and REQUEST systems is described in the Appendix. It suggests that the amount of computation required to produce similar logical representations of input sentences in LSNLIS and in REQUEST is comparable.

Future development

In light of our earlier observations concerning the present dearth of natural language based systems of demonstrable practical usefulness, it is reasonable to ask how much time is likely to elapse before one or more such systems have evolved to the point where a more concrete and conclusive assessment can be made of the practical potential of this general line of development. Based on current estimates for REQUEST, our best guess would be that a period of about two to three years would be required—one year to bring the system to the point of readiness for testing with respect to its ability to satisfy user requirements for a single, chosen application, and at least a second year for system testing, modification, and enhancement leading to a stable version that is well-engineered for easy communication with its users. Although the resultant system should be appropriate for extensive testing and evaluation, it will by no means be a production system. Such a possibility, if it materializes, clearly lies still farther in the future.

Appendix: Comparison of LSNLIS and REQUEST processing times

No comparison of processing times for two question-answering systems is meaningful without taking into consideration the coverage provided by those systems. Although differing in certain respects the coverages of LSNLIS and REQUEST are close enough that a comparison of processing times required for sentences belonging to their intersection is not ludicrous. The time for a full-scale comparison was lacking, but we were interested in at least obtaining a rough estimate of relative efficiencies. Four sentences were selected for which Woods has supplied timing information in the appendix of a paper [16]: 1) "How many lunar samples are there?," 2) "How many breccias do not contain Europium?," 3) "How many samples contain chromite?," and 4) "Which rocks contain chromite and ulvospinel?." Each sentence was replaced by a sentence that differed only lexically, and the modified sentences were run under REQUEST. The sentences processed were 1) "How many profitable companies are there?," 2) "How many companies do not produce gas?," 3) "How many companies sell computers?," and 4) "Which people sold IBM and Xerox?." Both parsing time and interpretation time necessary to produce a logical form suitable for evaluation with respect to a data base are included in the times (in seconds) given below. Times required for this evaluation were excluded, however, because they depend on the sizes of the data bases. Although the last three REQUEST sentences refer to information that is not included

Table 1 Sentence processing times required by LSNLIS and REQUEST.

Sentence	*LSNLIS Parsing time*	*LSNLIS Interp. time*	*Total LSNLIS time*	*REQUEST Parsing time*	*REQUEST Interp. time*	*Total REQUEST time*
1	2.039	5.152	7.191	2.814	0.200	3.014
2	6.272	8.593	14.865	2.956	0.279	3.235
3	3.579	8.277	11.856	2.768	0.269	3.037
4	7.743	9.782	17.525	2.189	0.219	2.408

in the REQUEST data base, it could easily be added if such data were available. This is of no consequence because evaluation time had to be excluded anyway for the reason stated above.

The times obtained are shown in Table 1.

Both systems are programmed in LISP and run on paged time sharing systems, but the LISP systems, time sharing systems, and computers in question are different. LSNLIS is coded in BBN-LISP. It runs under the TENEX time sharing system on a DEC PDP-10 Computer. REQUEST is coded for the IBM Research LISP system (version 124-4). It runs under the VM/370 time sharing system on an IBM 370/168 Computer. To meaningfully factor out those differences one must compare the speed of the two LISP systems running on their respective computers under their respective time sharing systems. Although we have estimated the relative speed of the two LISP systems in question on the basis of the best information available, we are not prepared to defend that estimate; thus we leave this normalization to the reader. We would expect, however, that after correcting for the faster IBM system, the LSNLIS and REQUEST total times would be roughly comparable.

Although caution is required because of the small sample, these results cast doubt on claims of greater efficiency which have been made for augmented transition network based systems over transformational grammar based systems. Although current question-answering systems are perhaps too slow for production purposes, their development is not being hindered by their slowness. Those of us who have been working on REQUEST have identified many ways of speeding it up, but we have not done so because we believe that this is less important at this time than developing the system to a stage of proven utility.

References and notes

1. J. E. Sammet, "The Use of English as a Programming Language," *Commun. ACM* **9**, 228 (1966).
2. I. D. Hill, "Wouldn't It Be Nice If We Could Write Computer Programs in English or Would It?," *Comput. Bull.* **16**, 306 (1972).
3. V. E. Guiliano, "In Defense of Natural Language," *Proceedings of the ACM Annual Conference*, New York, 1972, p. 1074.
4. C. A. Montgomery, "Is Natural Language an Unnatural Query Language?," ibid., p. 1075.
5. M. Halpern, "Foundations of the Case for Natural-Language Programming," *AFIPS Conf. Proc. Fall Jt. Comput. Conf.* **29**, 1966, p. 639.
6. W. C. Watt, "Habitability," *J. Am. Soc. Inf. Sci.* **19**, 388 (1968).
7. W. J. Plath, "Computational Linguistics and the Language of Computation," *Research Report 3071*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1970.
8. R. F. Simmons, "Answering English Questions by Computer: A Survey," *Commun. ACM* **8**, 53 (1965).
9. R. F. Simmons, "Natural Language Question-Answering Systems: 1969," *Commun. ACM* **13**, 15 (1970).
10. W. J. Plath, "Transformational Grammar and Transformational Parsing in the REQUEST System," *Research Report 4396*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1973. (to appear in A. Zampolli (ed.), *Computational and Mathematical Linguistics. Proceedings of the International Conference on Computational Linguistics*, Pisa 27 VIII–1/IX 1973, Casa Editrice Olschki, Firenze, Vol. I.)
11. W. J. Plath, "String Transformations in the REQUEST System," *Amer. J. of Computational Linguistics*, Microfiche **8**, 1974.
12. W. J. Plath, "REQUEST: A Natural Language Question-Answering System." *IBM J. Res. Develop.* **20**, 326 (1976, this issue).
13. S. R. Petrick, "Semantic Interpretation in the REQUEST System," *Research Report 4457*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1973. (to appear in A. Zampolli (ed.), *Computational and Mathematical Linguistics. Proceedings of the International Conference on Computational Linguistics*, Pisa 27 VIII–1/IX 1973, Casa Editrice Olschki, Firenze, Vol. II.)
14. M. M. Zloof, "Query by Example: The Invocation and Definition of Tables and Forms," *Research Report 5115*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1975.
15. W. A. Woods and R. M. Kaplan, "The Lunar Sciences Natural Language Information System," *BBN Report 2265*, Bolt Beranek and Newman, Inc., Cambridge, MA, 1971.
16. W. A. Woods, R. M. Kaplan, and B. Nash-Webber, "The Lunar Sciences Natural Language Information System: Final Report," *BBN Report 2378*, Bolt Beranek and Newman, Inc., Cambridge, MA, 1972.
17. F. B. Thompson, P. C. Lockemann, B. Dostert, and R. S. Deverill, "REL: A Rapidly Extensible Language System," *Proceedings of the Twenty-fourth National Conference of the ACM*, New York 1969, p. 399.

18. F. B. Thompson and B. H. Thompson, "Practical Natural Language Processing: The REL System as Prototype," *Advances in Computers* **13**, edited by M. Rubinoff and M. C. Yovits, Academic Press, New York, 1975, p. 109.
19. T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," Project Mac TR-84, MIT, Cambridge, MA, 1971.
20. T. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.
21. G. E. Heidorn, "English as a Very High Level Language for Simulation Programming," *Proceedings of a Symposium on Very High Level Languages, SIGPLAN Notices* **9**, 91 (1974).
22. G. E. Heidorn, "Automatic Programming Through Natural Language Dialogue: A Survey," *IBM J. Res. Develop.* **20**, 302 (1976, this issue).
23. G. E. Heidorn, "Augmented Phrase Structure Grammars," *Theoretical Issues in Natural Language Processing*, edited by R. Schank and B. L. Nash-Webber, June 1975, p. 1.
24. C. Kellogg, "A Natural Language Compiler for Online Data Management," *AFIPS Conf. Proc. Fall Jt. Comput Conf.* **33**, Part I, p. 473.
25. C. Kellog, J. Burger, T. Diller, and K. Fogt, "The Converse Natural Language Data Management System: Current Status and Plans," *Proceedings of the Symposium on Information Storage and Retrieval*, ACM, New York, 1971, p. 33.
26. R. F. Simmons, J. F. Burger, and R. M. Schwarcz, "A Computational Model of Verbal Understanding," *AFIPS Conf. Proc. Fall Jt. Comput. Conf.* **33**, Thompson Book Co., Washington, p. 441.
27. D. B. Loveman, J. A. Moyne, and R. G. Tobey, "Cue: A Preprocessor System for Restricted, Natural English," *Proceedings of the Symposium on Information Storage and Retrieval*, ACM, New York, 1971, p. 47.
28. I. Batori, "LIANA—Ein Deutschsprachiges Frage-Antwort-System," submitted for publication to *Linguistische Berichte*.
29. D. E. Walker, "SAFARI: An On-Line Text Processing System," *MTP-69*, the MITRE Corp., Bedford, MA, 1967.
30. W. A. Woods, "Transition Network Grammars," *Natural Language Processing*, edited by R. Rustin, Algorithmics Press, New York, 1973, p. 111.
31. J. H. Wolfe, "An Aid to Independent Study Through Automatic Question Generation (AUTOQUEST)," *NPRDC TR 76-18*, AD-A017 059, Navy Personnel Research and Development Center, San Diego, CA, 1975.
32. J. A. Craig, S. C. Berezner, H. C. Carney, and C. R. Longyear, "DEACON: Direct English Access and Control," *AFIPS Conf. Proc. Fall Jt. Comput. Conf.* **29**, 1966, p. 365.
33. Habitability of a computer-interpretable language is a term coined by William Watt (see reference [6]) to indicate the ability of users to stay within the limits of that language while expressing themselves productively.
34. W. G. Howe, V. J. Kruskal, and I. Wladawsky, "A New Approach for Customizing Business Applications," *Research Report 5474*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1975.
35. M. Hammer, W. G. Howe, V. J. Kruskal, and I. Wladawsky, "A Very High Level Programming Language for Data Processing Applications," *Research Report 5583*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1975.
36. See references [10–12] for a discussion of the coverage provided by REQUEST. We have not yet achieved enough coverage to consider the habitability of our present English subset.
37. E. T. Irons, "A Syntax Directed Compiler for ALGOL 60," *Commun. ACM* **4**, 51 (1961).
38. J. A. Katz and J. J. Fodor, "The Structure of a Semantic Theory," *The Structure of Language: Readings in the Philosophy of Language*, Prentice-Hall, Englewood Cliffs, NJ, 1964, p. 479.
39. D. E. Knuth, "Semantics of Context-Free Languages," *Math. Sys. Theory* **2**, 127 (1968).
40. W. A. Woods, "Procedural Semantics for a Question-Answering Machine," *AFIPS Conf. Proc. Fall Jt. Comput. Conf.* **33**, 1968, p. 457.
41. C. Hewitt, "PLANNER: A Language for Proving Theorems in Robots," *Proceedings of the International Joint Conference on Artificial Intelligence*, edited by D. E. Walker and L. M. Norton, 1969, p. 295.
42. M. Kay, "Experiments with a Powerful Parser," *Proceedings of the Second International Conference on Computational Linguistics*, Grenoble, France, August, 1967.

Received April 2, 1975; revised March 5, 1976

The author is located at the IBM Thomas J. Watson Reaearch Center, Yorktown Heights, New York 10598.