NATURAL LANGUAGE ACCESS TO A LARGE DATA BASE: AN ENGINEERING APPROACH*

David Waltz
Coordinated Science Laboratory and
Electrical Engineering Department
University of Illinois
Urbana, Illinois, U.S.A.

Abstract

An intelligent program which accepts natural language queries can allow a non-technical user to easily obtain information from a large non-uniform data base. This paper discusses the design of a program which will tolerate a wide variety of requests including ones with pronouns and referential phrases. The system embodies a certain amount of common sense, so that for example, it "knows when it does or does not understand a particular request and it can bypass actual data base search in answering unreasonable requests. The system is conceptually simple and could be easily adapted to other data bases.

1. Introduction

The prime obstacle for non-technical people who wish to use computers is the need to learn a special language for communicating with the machine. We feel that the time is ripe for natural language systems which can be used by persons who are not trained in any special computer language. Such a system must embody a degree of "common sense," must have a relatively large and complete vocabulary for the subject matter to be treated, must accept a wide range of grammatical constructions, and of course must be capable of providing the information and computations requested by the user.

In order to design such an ambitious system, it is imperative that the universe of discourse be limited in some way. This paper describes work done on a natural language question-answering system for a data base containing detailed records of U.S. Naval aircraft maintenance and flight information over a period of time. While the subject matter of this system is therefore quite constrained, we feel that the issues we are confronting are nonetheless general. In particular, we will describe the representation of common sense information and procedures, problems of pronoun reference and storage of partial results, the ability to answer vague or poorly defined questions, the ability of the system to recognize when requests are meaningless or too poorly formulated to answer, as well as linguistic issues.

2. The Data Base

We have obtained a data base from the Navy consisting of detailed records of aircraft maintenancy and flight Information extending over a period of time Each time a plane is serviced, a record is made including such information as the time and duration of the maintenance, who performed It, what action was taken, which parts were used or cannibalized, the manufacturers of these parts, and whether or not the service was scheduled or unscheduled. Records on the number of flights and the number of hours In the air are also kept for each plane. There are roughly thirty different record formats which occur in the data base, each containing between ten and twenty separate fields, where each field encodes information like the date of the action, type of aircraft, serial number of the aircraft, type of malfunction, component serviced, the work station performing maintenance and so on.

We are not addressing the problem of restructuring the data base; our system is interfaced to the record formats as they exist, although it would not be difficult to adapt the system to a new data base format

3. Basic System Operation

3.1 Prestored Request Forms.

The system understands requests by matching the requests with prestored request patterns. During this process, standard terms are substituted for synonymous words or phrases, simple misspellings are corrected and appropriate items are inserted for pronouns and referential phrases. If the matching Is successful, a unique prestored request form will be specified, consisting of three structures:

1. an English sentence skeleton which expresses the meanings of the request if fragments of the original request sentence are inserted This will be called the request meaning, skeleton.
2. a search function which can operate on the data base and return the appropriate answer(s). This will be called the search function skeleton.
3. an English sentence skeleton which can be used to answer the request when filled in with retrieved data. Tills will be called the answer skeleton.

Given a request sentence which exactly matches a prestored request form, the computer will type "Is it correct to assume that you want to know [instantiated request meaning skeleton]?" If the user does not object, the instantiated search function skeleton will be executed. (For a description of the data base search functions, see (l).>

If no exact match is possible, the program will record the request sentence for the benefit of the programmers, and try *to* use various techniques to make a partial or fuzzy match with some pattern. If it succeeds in this partial match,

*This work Is supported by the Office of Naval Research under Contract NOOO14-67-A-0305-0026. Auxiliary support was received from the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force under Contract DAAB-07-72-C-0259.

the program types the message "Do you want to know [instantiated request meaning skeleton]?" If the user says yes, then the system will proceed as before; If the user says no, the system will attempt to make weaker matches to patterns. Eventually, If the user does not think that any of the proposed internal meanings for his request are reasonable, the system will type a default request to rephrase the question. In no event will the system ever answer a question unless the user has agreed that the internal form returned for his approval expresses the question he wishes to have answered.

3.2 Processing Simple Requests.

The program operates in one pass; it does not have a separate parsing phase. As words are encountered left-to-right, the program traces through a prestored request network, a structure somewhat reminiscent of an augmented transition network.2 Instead of nodes labeled "Noun," "Adjective," etc., the prestored request network has nodes which correspond either to specific words or to phrase types; some examples of phrase types are *time-period which matches phrases like "during April 1974," "between April 3 1974 and April 30 1974," "In 1974," etc., and *number-planes which matches phrases like "any Phantoms," "more than three Phantoms," "a Phantom or a Skyhawk," etc. Attempts are made to correct spelling to facilitate matches at each node. Each time a node is matched successfully, a phrase type variable is given as a value the matched phrase from the request sentence, transformed into a canonical representation. For example, "during April 1974" is transformed into "time-period April 1 1974 April 30 1974."

Eventually, either there is no match in the network for some word in the request sentence, or all the words In the request are processed. In the former case, various kinds of "fuzzy matching" will be tried to guess at the meaning of the request, In a manner similar to that in PARRY2. (Fuzzy matching involves dropping one or more words from the request to force a match). If all the words have been processed, a check will be made to see if the last node processed is associated with a unique prestored request form (see Section 3.1). If so, this prestored request form is assumed to be the appropriate one for the given request and processing continues as described in Section 3.1. If more than one prestored request form remains as a descendent of the last processed node, an attempt is made, using the context registers, to decide which prestored request form is intended.

3.3 Context Registers and Pronoun Reference.

Once a user has signalled his agreement that the instantiated answer skeleton represents the intended meaning of his request, the program fills certain global program areas called context registers with Information that may aid In anawerlng subsequent questions. There are context registers corresponding to each of the phrase types which can occur as nodes in the prestored request network, as well as registers for the answer to each request and for all the "hits" encountered in generating the answer, unless the number of hits exceeds a set limit. If information required for instantiating a prestored request form is missing, the program will fill In the missing items from the current context register values for the missing variables. So, for example, if the phrase "during April" is encountered with no year specified, the current year In the context registers is assumed to be Intended.

The context registers are also used to handle pronoun reference. If a pronoun or referential phrase (e.g. "the plane") is encountered, it is matched against the appropriate phrase types in the prestored request network (e.g. *specific-planes). The daughters of all nodes which cannot be eliminated are then explored, until a unique phrase type can be chosen to represent the referent. (Often, especially late in a sentence, there will only be one possible continuation in the network, and thus only one possible phrase type to which the referential phrase or pronoun must refer.) The current context register value for this phrase type will then be bound to this phrase type variable.

Certain other phrases have the effect of limiting the scope of the overall data base search. For example, "Of these" followed by a request specifies that the search should be carried out on the data in the context register containing the hits from the last request. Certain phrases which can occur in sentences are not included in the prestored request network. Such phrases include time phrases, like "on April 1 1974," or "between April I and April 15," as well as phrases denoting work centers, bases to which aircraft are assigned, descriptions of malfunctions, ways in which malfunctions were discovered, and others.

These phrases can occur In different places within a sentence, and are handled wherever they occur by the same mechanisms. Such phrases generally act as qualifiers for events, i.e. they require that events satisfy extra criteria such as occurence at a particular location or in a particular manner.

3.4 An Example.

Figure 1 shows a simplified portion of the prestored request network. Note that both the active and passive forms of requests are stored directly, instead of storing Just one form (e.g. the active) and a set of transformations.

Suppose that the request

> (1) Please tell me if Phantom number A49283 had any engine maintenance during April 1974.

is given to the program. This request will match the nodes leading to prestored request form 1, will set context variables *specifIc-planes to ((plane-type F4)(tail-number A49283)), *maintenance-actions to ((type-equipment engine)(type-maintenance any)) and *tlme-period to (time-period April 1 1974 April 30 1974).

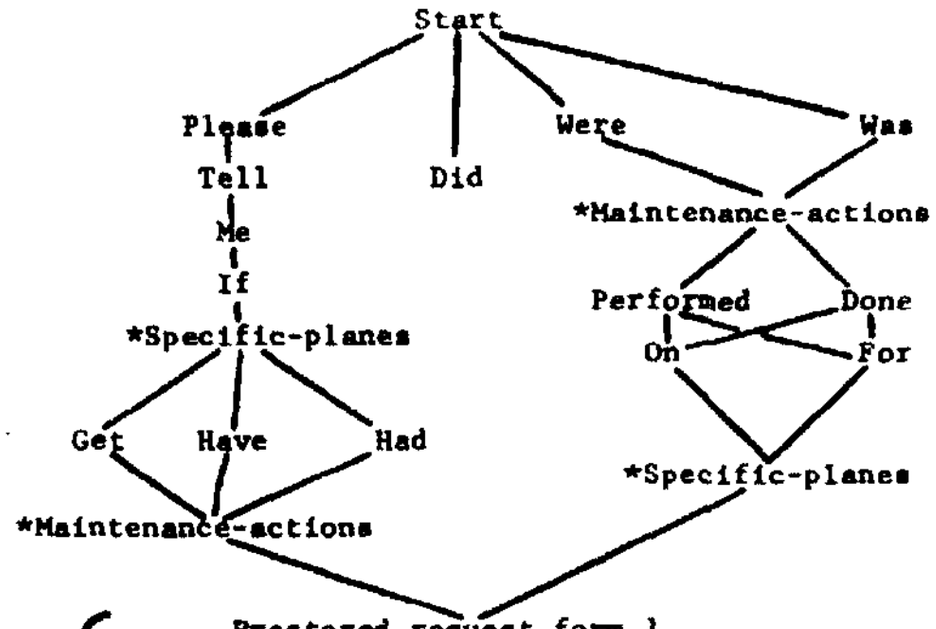
The request meaning skeleton will then be instantiated, and the message

> (2) Is it correct to assume that you want to know if the F4 with tall number A49283 had any engine maintenance between April 1 1974 and April 30 1974?

typed out to the user. If the user types "yes" then the program will execute the instantiated search function skeleton and fill in the spaces

of the answer skeleton to produce something like

(3) Yes, 3 engine maintenances were performed on F4 number A49283, one on April 6, 1974, one on April 10, 1974 and one on April 27, 1974.



Prestored request form 1

1. **request meaning skeleton**
   (if *specific-planes had *maintenance-actions *time-period)
2. **data base search function**
   (FOR ALL X1/(*SPECIFIC-PLANES (*PLANE-TYPE *MAINT *TIME-PERIOD)):
   (MEMBER (MAINT-FIELD) *MAINTEN-ANCE-ACTIONS);
   (RETURN (LIST (TAIL-NUMBER) (MAINT-FIELD) (DATE))))
3. **answer skeleton**
   ((NOT (NULL ANSWER))
   (LIST `YES (LENGTH ANSWER) *MAIN-TENANCE-ACTIONS `WERE `PERFORMED `ON *SPECIFIC-PLANES (ENUMERATE ANSWER `(ONE ON))))
   (T `(NO))

Figure 1. A portion of the prestored request network and a prestored request form.

Suppose that sentence (1) is then followed by

(4) Did it have any unscheduled landing gear maintenance?

This will match the same prestored request form as did sentence (1), but in this case "it" will match *specific-planes, and the contents of the current context register for *speclfIc-planes will be assumed to apply for this sentence also. This system does check for number agreement, so that

(5) Did they have any unscheduled landing gear maintenance?

would fail to match the contents of context register *speclfic-planes which is singular, and would trigger the error message "I do not know what 'they' refers to." Note that in a discourse, (5) may have a valid meaning, referring back to an earlier line of questioning; for now, the program will not look further back than the current contents of context registers. New context Information merely supercedes the old. Thus, once a line of questioning Is dropped, the context for that line of questioning must be explicitly restated If one wishes to resume It so that the system cannot recover the entire context from a sentence like:

(6) Earlier we were talking about Skyhawks.

Of course, there Is no inherent reason why such an ability could not be added later.

To complete the example, when the program attempts to fill out the request meaning skeleton, It will note that a value for *tlme-perlod is needed, and will obtain this from the *time-period *context* register also. Of course, as always the program will check with the user to make sure that it has correctly interpreted his meaning before continuing.

3,5 Processing More Complex Requests.

As described so far, the prestored request network contains no loops. Recursive ability Is available through the recognition of rank-shift constructs, usually signalled by relative pronouns. These cause the request to be broken up Into two or more nested requests. For example

(7) Did any of the Phantoms which had engine maintenance in April also require engine maintenance in May?

can be broken up into two requests, the first to find all Phantoms which had engine maintenance in April, the next to find which if any of these had engine maintenance in May. "flits type of request is handled by general rules, not by exact matching.

Similarly, compound requests can also be broken up in sequences of simple requests, in a manner somewhat reminiscent of that used in Bobrow's STUDENT program.

4. Discussion

4,1 Procedural Problems

There are technical difficulties in enumerating all the sentence patterns that can occur. We have solicited and collected sample questions from a number of persons as a start. Of course, it is likely that when this system is made available to actual users, a large number of new sentence patterns will have to be added to the prestored request network. Our hope is that few new search function skeletons will have to be added. We expect the eventual number of prestored request forms (see Section 3.1) to total between 1000 and 2000.

A learning scheme could be devised to add new items to the network. If a user eventually hit a pattern by rephrasing a request, and agreed that his earlier requests had had the same meaning as the one that ultimately succeeded, then patterns matching his earlier requests could be added In suitable form to the network pointing to the same search, answer and meaning skeletons as the hit pattern. One would have to be quite cautious, though about the potential loss of precision; such a scheme might be most useful for the designers of the system for deliberately adding new patterns that had the same meaning as patterns previously encoded. For further discussion, see Colby et al.^

An effort has been made to combine search

function skeletons. For example (8), (9), (10) and (11) below all point to the same search function skeleton, though the request meaning skeletons and answer skeletons for each are different.

(8) Did any Phantoms have bird strike damage during April?
(9) How many Phantoms had bird strike damage during April?
(10) Did ten or more Phantoms have bird strike damage during April?
(11) List all the Phantoms which had bird strike damage during April.

4.2 Common Sense.

We would like to trap certain types of "unreasonable" requests without actually performing a data base search for answers. For example the requests

(12) Did any Phantom have more than 500 engine maintenances during August 1973? or
(13) Did any Skyhawk crash more than 3 times last month?

are suspicious, Just as

(14) Did you eat more than 500 chickens for dinner last night?

would be. Whenever a request form includes a comparison with a number, a check can be added to the request form, using simple tables giving an order of magnitude estimate for items like the number of man-hours for various types of maintenance, average frequency of failure rates, and Information about the fact that certain events, e.g. crashes, typically happen only once to a particular airplane.

In addition, we would eventually like to include an auxiliary semantic network to allow the system to answer questions like

(15) Does an F4 have propellers?
(16) What components are given periodic maintenance on a Skyhawk?

It is a long range goal of this research to have the system answer vaguely formulated questions like:

(17) Are there any common factors in the maintenance histories of the two Phantoms which crashed last month?

In order to answer such questions, the system must have an understanding of what "common factors" are: similar events or event sequences, servicing by the same shops, the missing of periodic services, failures of the same subsystems, etc.

4.3 HistoricalPerspective.

I have called this an engineering approach because it falls outside the boundaries of traditional linguistics. There is no dictionary entry for many of the words "understood" by the system; In general, words are defined implicitly within each pattern in which they occur In the prestored network. Transformations need not be used; for example both active and passive forms of requests are prestored. Some non-grammatical sentences are accepted as meaningful. For example, the system does not require subject-verb number agreement. This is not because such a mechanism is difficult to program but because we would like the program to be tolerant of common grammatical errors. A number of aspects of the system are, however, in the AI tradition.

Some parsing ability does exist within the matching programs for phrase types. Phrase types are in general either prepositional phrases or noun phrases, and are analyzed in a manner directly drawn from Winograd's noun phrase analysis.4 The rank-shift clause handling mechanisms are also drawn from Winograd's work.

The way in which the meanings of requests (i.e. prestored request forms) are stored is similar to the way meanings are stored in Woods' LUNAR program.[7] Overall Woods' program is the most similar precursor in intent and operation, though the data base we are using and the variety of questions which can be asked in our system is much greater.

The prestored request network can be viewed as a combination of many case-frames. In general, however, the phrase types are much more specific than are cases (e.g. agentlve, instrumental, dative, factlve, etc.) although some are similar (e.g. locative). In fact the phrase types are specific enough so that paths through the prestored request network resemble pieces of a semantic network.'»*" For example, the fact expressed by the network in Figure 1 would be "Specific airplanes can have maintenance performed on them." While it is of course risky to make negative inferences (e.g. since "Pilots can have maintenance performed on them" does not match any prestored form, this sentence does not make sense) if a sentence does match a prestored form, the system can be quite confident that the request is meaningful.

4.4 Relationship to a General Language Understanding System.

It should be obvious that this system does not attempt to model the internal mechanisms of human language understanding; it is an attempt to solve the problem of making a large body of data accessible via natural language, using no more mechanisms than are absolutely necessary. The environment of the program is quite remote from that of a general language understanding system. Questions are ail either In the past tense or present tense, and present tense is used only to refer either to the dialogue or to things which are always true. The number of objects, events and relations In the program's universe of discourse can be easily enumerated. The role of the program with respect to a user is always that of a data retriever.

Nonetheless, there are certain common issues which a general understanding system must face. One particularly pertinent issue concerns the appropriate unit of knowledge. Should wordB be defined explicitly, as In dictionary "definitions," or implicitly, as in this system? Should "world knowledge" and linguistic knowledge be stored separately or should they be merged as in this system? I believe that a general language understanding system can benefit from some of the ideas in this paper.

In order to present the type of knowledge contained in the prestored request network, something akin to this network must be constructed. Certainly, such facts as "Airplanes require main-

tenance" do not emerge from concatenating isolated word definitions of "airplane," "require" and "maintenance," unless these facts are somehow made part of these or related definitions. (I.e. machines require maintenance, and an airplane is a machine).

Similarly, one cannot write a program in which data base search functions can be incrementally constructed, (as in Wlnograd's SHRDLU[4]) and in which at the same time meaningless requests can be trapped without writing a program much larger and slower than ours.

Whether or not one should transform all input sentences into a canonical form to save on world knowledge storage space, or whether all surface forms should be matched directly is much more debatable, but the storing of surface forms does allow non-grammatical input sentences to be tolerated easily.

4.5 Advantages for Data Base Question Answering Systems.

There are some distinct advantages to the type of system described in this paper.

1. Idioms can be handled very neatly and easily.
2. The system can operate very rapidly and need not have backtracking ability.
3. The prestored request network allows pronoun reference to be handled in a way that seems intuitively clear and which is simple to implement.
4. Perhaps most Importantly, unlike the cases of most other language systems, implementing a system like ours Is a simple and straightforward process, (although the writing may indeed be time consuming and each application to a new data base does require a substantial amount of rewriting.)

References

(1) Rutter, P. "Navy Data Base Read Package," Coordinated Science Laboratory A.I. Working Paper 3, University of Illinois, 1974.

(2) Woods, W. A. "Transition Network Crammers for Natural Language Analysis," Comm. ACM 13, 1970, pp. 591-602,

(3) Colby, K. M., Faught, B., and Parklson, R. "Pattern-Matching Rules for the Recognition of Natural Language Dialogue Expressions," Stanford Artificial Intelligence Laboratory Memo AIM-234, 1974.

(4) Wlnograd, T. Understanding Natural Language, Academic Press, New York, 1972.

(5) Bobrow, D. G. "Natural Language Input for a Computer Problem-Solving System," In Mlnsky (ed.) Semantic Information Processing, M.I.T. Press, Cambridge, Mass., 1968.

(6) Gabriel, R. P. and Waltz, D. L. "Natural Language Based Information Retrieval," Proc. of the 12th Annual Allerton Conference, University of Illinois, 1974,

(7) Woods, W. A., Kaplan, R. M., Nash-Webber, R., "The Lunar Science Natural Language Information Systems Final Report," BBN Report No. 2378, Bolt, Beranek and Newman, Inc., Cambridge, Mass., 1972.

(8) Fillmore, C. "The Case for Case" in Bach and Harms (eds.) Universals in Linguistic Theory. Holt, Rinehart *and* Winston, 1968.

(9) Qulllian, R. "The Teachable Language Comprehender: a Simulation Program and Theory of Language," Conn. ACM 12, 1969, pp. 459-476.

(10) Norman, D. A. and Rumelhart, D. E. "Active Semantic Networks as a Model of Human Memory," Proc. of Third International Joint Conference on Artificial Intelligence. Stanford Univ., 1973.

(11) Collins, A., Warnock, E. H., Alello, N. and Miller, M. L. "Reasoning from Incomplete Knowledge," unpublished draft, Bolt, Beranek and Newman, Cambridge, Mass., June 1974.