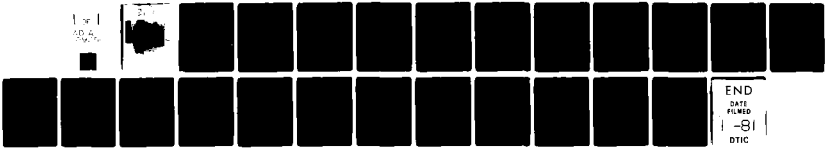
AD-A092 500 DARTMOUTH COLL HANOVER N H DEPT OF MATHEMATICS F/G 5/2
NATURAL LANGUAGE DATA BASE QUERY.(U)
OCT 80 L R HARRIS N00014-75-C-0514
UNCLASSIFIED NL



END
DATE FILMED
-81
DTIC

(12)

(9) FINAL REPORT

Submitted to the Office of Naval Research
for a grant in support of research entitled

(6) Natural Language Data Base Query

(15) N00014-75-C-0514

(11) 1 Oct 80

(12) 24

(10) Larry R. Harris
Principle Investigator
Dartmouth College
Hanover, NH 03755

404 325

Security Classification

DOCUMENT CONTROL DATA · R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) DARTMOUTH COLLEGE HANOVER, NH 03755	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. [redacted] AD-A092 500

3 REPORT TITLE

FINAL REPORT SUBMITTED TO THE OFFICE OF NAVAL RESEARCH FOR A GRANT IN SUPPORT OF RESEARCH ENTITLED NATURAL LANGUAGE DATA BASE QUERY.

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

5 AUTHOR(S) (Last name, first name, initial)

HARRIS, LARRY R.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
1 OCT 1980	20	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-75-~~C-054~~ 0514 ✓ b. PROJECT NO. c. NR049-344 d.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. AVAILABILITY/LIMITATION NOTICES

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	OFFICE OF NAVAL RESEARCH

13. ABSTRACT

This final report is intended to be a summary of the research on Natural Language Data Base Query, ~~performed at Dartmouth College supported by the Office of Naval Research since 1973.~~ It has been the goal of this research to determine a minimal set of techniques sufficient to provide a practical natural language capability for data base query. This report summarizes the basic requirements for such a capability and suggests techniques for meeting these requirements. As such, this report is in effect, a specification of the minimal functionality for a practical natural language data base query capability.

DD FORM 1 JAN 64 1473

UNCLASSIFIED

Security Classification

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
NATURAL LANGUAGE DATA BASE QUERY PARSING						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

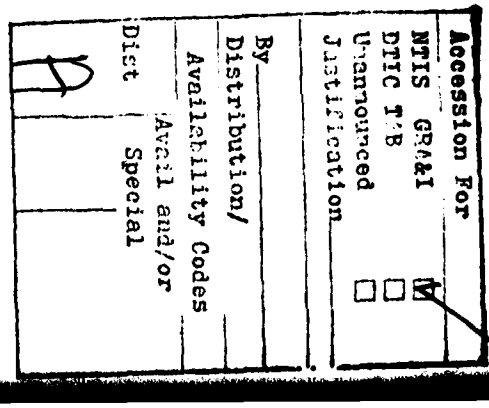
14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

FINAL REPORT
Submitted to the Office of Naval Research for a grant in support of research entitled Natural Language Data Base Query.

Larry R. Harris
Principle Investigator
Dartmouth College
Hanover, NH 03755

Abstract

This final report is intended to be a summary of the research on Natural Language Data Base Query performed at Dartmouth College supported by the Office of Naval Research since 1973. It has been the goal of this research to determine a minimal set of techniques sufficient to provide a practical natural language capability for data base query. This report summarizes the basic requirements for such a capability and suggests techniques for meeting these requirements. As such, this report is in effect, a specification of the minimal functionality for a practical natural language data base query capability.



Introduction

When research under this contract began in 1973, the state of the art in practical natural language data base query was essentially non-existent. All of the then existing research systems (and many of today's systems) were semantically limited to a single domain of discourse. The primary requisite of a "practical" query capability is that it be "application independent". Achieving this application independence requires a fundamental commitment throughout the design of the system. It is encouraging to see that the research community as a whole has started moving in this direction.

This report consists of a summary description of the minimal requirements for a practical query capability. The techniques developed to meet these requirements are organized into the four major components of operation that make up the processing cycle of a request. These components are the lexical analyzer (the scanner), the syntactic analyzer (the parser), the data base structure analyzer (the navigator) and the data processing module.

The Lexical Analyzer -- The Scanner

The basic function of the scanner is to determine what the individual words or tokens are. The scanner breaks the input stream into a sequence of tokens. A modification of the finite automaton scanner used in compilers is sufficient for this task. The modifications are required to deal with phrases and recognition of special numeric tokens. Phrases such as "Vice President" or "New York" must be recognized as single tokens even though they contain a space. Numerical strings such as "80/01/31" must be recognized as a single token (representing a date) whereas "1/3" must be recognized as three tokens representing "one divided by three".

Scanners with these capabilities are commonplace among AI natural language systems. The most common pitfall is to imbed spelling detection, or worse yet, spelling correction within the scanner. Both spelling correction and spelling detection require advance knowledge of all words that can be employed by users. This is prohibative in a domain independent approach, since this set of words must clearly contain all the words in the data base. Therefore, the scanner must accept words about which it knows nothing -- which could be, of course, potentially misspelled words. The detection of a spelling error is best made as part of

the post mortem when a sentence fails to be understood by the system. This approach satisfies the domain independence criterion as well as allowing valid requests with no spelling errors that pertain to data values not actually in the data base to be handled properly.

The Syntactic Analyzer -- The Parser

The parser is the heart of the natural language component of the system. Its role is to syntactically relate the components of the request. This process drives the construction of the semantic structures that represent the meaning of the request. There are several computing paradigms for natural language parsers. In the early part of our research, we successfully employed a top-down context-free parser. Later we switched to an Augmented Transition Network parser (ATN). We feel there are several advantages to the ATN approach that make it far more convenient to use, although successful context-free parsers could be built as well. Recently several new parsing schemes have been reported in the literature, so that it may well be the case that the ATN technology is now dated. However, it seems clear that the choice of parsing algorithm is less important than the mechanism by which the syntax controls the generation of the required semantic structures.

Ambiguity

The definitive problem of this type is that of ambiguity. Several distinct semantic representations must be generated from a single input string. The difficulty here is not how to do it, but how to limit the generation of too many interpretations. Many researchers have chosen to limit the parser to generating only one interpretation and stopping. This approach makes the decision of which parse to pursue a very critical one. It also makes dealing with truly ambiguous requests very difficult.

Our recommendation is to solve the problem from the other end of the spectrum by non-deterministically generating all possible interpretations. This transforms the issue from one of trying to decide on a relative basis which of two partial parses looks more promising, to one of trying to decide on an absolute basis which of two complete parses is more meaningful. There is also a problem of efficiency in coping with the potential exponential number of interpretations. It should be clear that decisions made on an absolute basis after the parse should be more accurate than decisions made on a relative basis during the parse. The effects of the remaining portion of the input have had a chance to impact

the decision in the former case, but not in the latter. However, the problem of exponential growth must be dealt with very carefully. Fortunately it seems that, at least for natural language query, this can be dealt with by tuning the parser to reduce the non-determinism. Fortunately the length of the input is usually very small (less than 20 tokens) and the number of decisions is "reasonably" small.

Binding Values to Fields

One type of ambiguity that arises frequently in data base queries is that of choosing the field to which a given value is related. Most research systems solve this problem with dictionary definitions. This, of course, is clearly a violation of domain independence since it requires enumerating all unique data base values in the lexion. Our approach has been to dynamically determine this from indices maintained by the DBMS. In addition to this, we allow three levels of strength in defining data values in the dictionary. They can be tightly bound, weakly bound or unbound. This allows for sufficient generality at the same time it permits definitions that could limit the non-determinism.

High-Level Semantic Entities

Another important component in the relating of syntax to semantics is the ability to deal with entities that themselves imply a significant semantic structure. This involves both complex definitions in the dictionary, as well as the ability to deal with these definitions in the parser. Examples of this would be terms like "bachelor" or "profit margin". The first specifies a complex description whereas the second specifies a formula for calculating profit margin from other entities that are more directly available. It is of critical importance for a natural language system to be able to deal with such terms directly, rather than forcing the user to continually define them.

Pronouns

Another type of word that implies a substantive body of semantics is the pronoun. The distinction here is that the "meaning" of the pronoun is not to be found in the dictionary but in the context of the dialog. In addition, there may be some ambiguity that arises in determining what the pronoun refers to. A wide variety of solutions appears in the literature for this problem. Our approach is to maintain two context registers

containing the semantic structures of previous queries. One is the previous request, the other is the request that the previous request may have referred to. In addition, intra-sentential pronoun references are also allowed.

We have found this approach to be sufficient for resolving the vast majority of pronoun references in data base queries, including the difficult sequence: "What is the maximum salary in Missouri?" "Who earns it?". Contrary to popular belief, the pronoun "it" does not just refer to the answer of the first request. If it did, the second request would generate all people earning that salary, even those outside Missouri.

Ambiguous pronoun references are dealt with in the same way as other ambiguities. All of the possible referents are considered and non-deterministic interpretations are created for each possibility. These interpretations are then compared on a global basis later along with other interpretations created by other types of ambiguity. We have found a "search logic optimizer" to be invaluable in weeding out undesired interpretations created by improper pronoun references. The optimizer detects the logical contradictions that often get created in this way.

Arithmetic Expressions

One formalism that does creep into natural language data base queries is that of arithmetic expressions. We have found the "recursive descent" form of language approach for parsing arithmetic expressions to be completely consistent with the ATN grammar. However, certain non-formal forms of arithmetic expressions appear in natural language queries. For example: "How much is his salary and his commission?" Here we see "and" used as plus (perhaps) and the pronoun "his" breaking up the operand-operator-operand sequence. Refinements to the basic recursive descent approach are required to deal with these problems.

Response Heuristics

We have found several heuristics necessary in determining the proper response for a given query. These are primarily due to the informal phrasing of queries that occur in natural language. Often users do not explicitly ask to be given information that they quite obviously need to interpret the answers and quite rightfully expect the system to provide. For example, the request "Print the salary of Smith and Lawler" implies the printing of name even though a literal interpretation would not print it.

Without printing the names as well as the salaries, it becomes impossible for the user to determine which salary goes with which name.

Similarly, natural language requests can trigger rather silly responses if they are interpreted too literally. The query "Who is Smith?" illustrates this point. If the system were to mechanically interpret "Who" as an indication to print name, it would respond "Smith" to the request. Clearly, heuristics must be employed to detect such situations and to determine what the proper response should be. The activation of such heuristics must ultimately be under user control, otherwise the system may be perceived as overbearing since the user loses the ability to precisely control the response in those situations where it is important for the system to do exactly what it is told.

The Navigator

After the query is parsed, the semantic structure must be projected onto the database scheme resulting in a strategy for extracting the desired information from the database. The difficulty of the navigation problem can range from trivial to arbitrarily complex depending on data model employed by the DBMS and how well the given data base is organized.

For single flat file organizations, navigation is at its simplest. But even for single flat files, difficulties can arise if the file was created by flattening out a hierarchy or a network. In these cases, the notion of a record corresponding to a real world entity is lost. Hence the navigator must take advantage of the fact that the file was originally non-flat to construct the proper means of access into the file.

For the more structured data models such as relational, network and hierarchical, the navigation process takes on great importance. Determining the proper entry point into the structure as well as the proper linking relationships can critically affect the contents of the final response, not to mention the impact on efficiency. In this regard, the relational model's greatest asset is its closed form expression of a query. This makes it at least

possible to express navigation in a high-level language like SEQUEL. For hierarchies and networks no such closed form expression is used by the DBMS and therefore it is impossible to express navigational choices without resorting to a procedural representation. This is the essence of the reason why no good high-level query languages, even of a formal nature, exist for network and hierarchial DBMS's. Some intermediate level representation is clearly needed here.

Even for the relational systems we have not been uniformly satisfied with SEQUEL as an intermediate language. The navigational linkage is specified in an unduly intricate fashion, and the functionality is incomplete. This latter point is indicative of the fact that much of SEQUEL's power is misdirected in terms of the needs of a naive end user of a natural language system, at least in terms of our experience. Whereas SEQUEL provides no assistance in answering many of the difficult natural language requests we encounter, its power in intricate cyclic navigation is too subtle to be useful in a natural language setting.

Navigational Optimization

Another aspect of navigation is query optimization. The navigational choices made have a profound effect on the efficiency of generating the answer. But even when dealing with a single relation (or a single flat file database), there is a significant amount of query optimization that can be done. For example, questions asking from the maximum, the minimum, or unique listings can be answered directly from the DBMS indices if they are available. These optimizations can change a several minute response into an instantaneous response. These kinds of optimizations require knowledge of how the data is going to be processed after retrieval as well as knowledge of and access to the DBMS indices.

Knowledge of the DBMS indices is also a critical factor in navigation because it determines how long the DBMS will take to respond. Clearly we want to optimize the work done by the DBMS and prefer to generate requests that make use of indices or hash coding rather than file pass searches. But, given the variety of ways in which DBMSs behave on mixtures of keyed and non-keyed searches, it is clear that the interface must have its own ability to perform the functions of searching and sorting. There is a nice meshing of these functions that makes it possible to avoid any

DBMS restrictions in this area and at the same time gives the interface control of the situation so that expensive requests can be trapped. In general, once the DBMS gets control, the user must wait until the DBMS has finished processing the request, which may be quite a while. By selectively sharing some of the searching and sorting work, it is possible to maintain control and warn the user when things get expensive.

This flexibility is achieved only by added complexity. Not only must the interface have the searching and sorting functionality but it must also be prepared to represent the partitioned workload and, of course, compute the partition that will effect the greatest efficiency. With some DBMSs, this capability is only an efficiency option. With other DBMSs that do not support non-keyed searching or sorting at all, this capability becomes critical in terms of being able to answer the request at all.

Security

Another aspect of the navigation problem is how security is taken into account. It is clear that for naive end users of a natural language query system that security from unauthorized

access is a critical issue. Our original hope in this regard was that we could merely rely on the DBMS subschema to provide the necessary security. Unfortunately, we found the granularity of the subschema security to be too large--i.e. access is granted on a field-by-field basis. This is fine for application programs but too restrictive for data base query. We have proposed that security also be defined on a record-by-record basis so that a user might have access to certain fields only for a specific set of records.

The implication of all of this on navigation is that the navigational choices made by the system must be a function of what data is available to the current user. For some users, without access to all relations, this may require less direct paths than would otherwise be required. It is up to the navigator to find the best path to relate all the necessary data without violating any of the security constraints along the way.

The final issue related to navigation is one that is currently unresolved at this time. This issue is the mechanism by which the parser communicates to the navigator the explicit "relationship" information that the user included in the request. In general, the navigator must be prepared to work in the absence of such information making use of predefined "natural paths".

However, in those cases in which the user wishes to override these predefined paths by explicitly mentioning another relationship, the system must be prepared to act accordingly. For example, in a database of professors and students related by both a "teaches" and an "advises" relationship, the two request: "Who are Professor Harris' students?" is different in a navigational sense from "Who are Professor Harris' advisees?" or "Who does Professor Harris advise?" It is clear that for this simple case the use of the word "advises" or "advisees" indicates to the navigator which relationship to employ. But in more complex cases where the same two relations must be joined more than once in a request, it is not clear that all such relationships should be controlled by the explicit use of one relationship words. Of course if not all such relationship choices are impacted, then we must decide which ones are and which ones are not, presumably on the basis of the original syntax. However, at this point, it is not clear that English syntax could (or even should) provide this kind of information. This remains an open issue at this point in time.

The Data Processing and Formatting Module

After a query is processed by the DBMS and additional searching or sorting is performed by the interface itself, we eventually arrive at a single temporary relation that answers the user's request. This data still must be processed to provide the user with the information summarized or formatted in the desired way. On the one hand, this kind of data processing seems very commonplace--computing subtotals, formatting reports, etc. On the other hand, this module would ideally be capable of carrying out an arbitrary sequence of processing. In this case it becomes the automatic programming problem. Some intermediate level of capability is sought.

The placement of various functions such as total, minimum and maximum, etc. can often be in either the DBMS or in the interface. This represents another example of how the workload can be shared. Similarly the data processing module must work intimately with other aspects of the interface that maintain the pronoun context. This is true because it is not until the end of processing the data that we really know all the contexts to which subsequent pronouns might refer. It is certainly conceivable that some processes will restrict even further the set of records displayed for the user on the basis of some arbitrary predicate.

From the user's point of view, a pronoun is likely to refer only to the set of records actually printed. Since the interface has no knowledge of what the arbitrary predicate is, it would be impossible for it to generate a high-level representation of what subsequent pronouns may refer to. The implications of this are quite profound. Since the system no longer has a high-level representation of the pronoun referent, it becomes difficult to "echo" the meaning to the user or to even talk about interpretation in clarification dialogs.

On the positive side, the low-level representation of a pronoun reference that is necessitated by all this can speed up all pronoun references. This is particularly noticeable when the result of a non-keyed search is referenced by a pronoun. If only a high-level pronoun representation is maintained then the costly search must be recomputed. If both a high-level and low-level pronoun representation is maintained, then the system can describe the query to the user with the high-level representation and directly access the desired records without search using the low-level representation. This has the effect of building an index for an arbitrary set on the fly and using it to speed up subsequent accesses to the same set.

With regard to the data processing routines themselves, there is an interesting relationship to syntactic quantification. There is a direct semantic connection between the category specification employed by the processes on certain types of quantification. Consider the request "How many salesmen are over 100 percent of quota in each region?" The quantification "in each region" semantically defines the categories to be employed by the counting process. This gives an interesting simplified representation for this kind of quantification.

Conclusion

We have summarized the results of the ONR supported research on natural language database query. It is interesting to note the change in expectations about database query that have taken place during the life of this research contract. At the outset, people regarded practical natural language systems as a futuristic notion: something that would not be available for 10-20 years. The current atmosphere is one in which a few real world applications are just making it into actual production.

It is fair to say that the issues considered most important at the outset of this research (the natural language analysis) is no longer the limiting factor. As is evident from the discussion given in this report, the navigational and processing functions provide the most fertile ground for future research. As such, the problem of providing practical natural language access to database is no longer to be considered a primarily natural language analysis problem, but also a theoretical database problem with overtones of automatic programming. For this reason, it is not likely that more sophisticated parsing techniques will impact current capabilities as much as more general AI research related to database semantics is likely to do.