KNOWLEDGE-BASED ENGLISH LANGUAGE SYSTEMS FOR MANAGEMENT SUPPORT: AN ANALYSIS OF REQUIREMENTS

Ashok Malhotra
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Abstract

This paper investigates the utility and feasibility of a computer-based system that supports managers by allowing them to ask questions and issue commands in English. To determine managers' actual requirements an experiment was conducted in which subjects were asked to solve a problem using a Perfect System that could respond to any request made to it. Their behavior was analyzed to determine the acceptability of the system and the facilities it would have to provide as well as the power of the parser and the vocabulary that would be required. These requirements were modified in light of available technology to design a system that is both useful and feasible.

1- Introduction

Two of the factors that have mitigated against better management use of computers are the necessity to communicate with them in a special language and to specify the details of the processing required. Apart from the investment required in learning how to use computers, these factors lead to delays and neccessitate a significant, special effort whenever the manager has an unusual request. To try to overcome these limitations we decided to investigate the utility and feasibility of a computer-based management-support system that would allow the manager to phrase his requests in English and contain enough domain-specific knowledge to analyze them and respond to them. The investigation is described in detail in Malhotra (4]. This paper summarizes our methodology and main findings.

Preliminary conversations with managers indicated that such a system should serve as a front-end to a corporate data base to support problem analysis and decision-making. It should provide facilities for data retrieval and manipulation as well as be able to answer questions about its contents and capabilities. The system should also provide facilities for building and using management models. If a software error occurs during the processing of a request the user should not be asked to take any special action. It should be trapped by the system and the user merely asked to rephrase his request.

2. The Prototype System

To come to grips with the substantive problems involved in building such a system we decided to implement a prototype as a front-end to the corporate data base of the operations of a manufacturer of lead batteries. Figure I. is a schematic diagram of the prototype system. Functionally, the system can be divided into two parts, the parser and the processor. These two operating subsystems rely upon a knowledge base that contains a model of the world, a model of the problem area and knowledge of the structure and the contents of the data base.

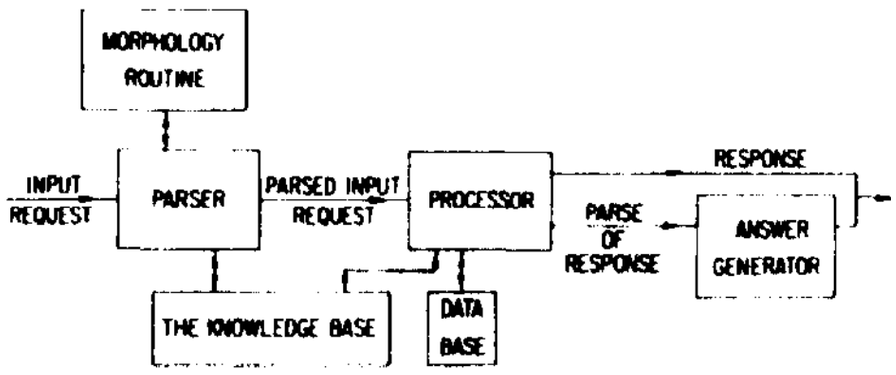

Figure 1. Schematic Diagram of the Prototype System.

The parser undertakes the initial analysis of input to the system. We shall not describe the parser since it is the knowledge base and the processor that are central to this paper

The morphology routine acts as a preprocessor for the parser. It examines each word in the input request and checks if it is known to the system. Unknown words are analyzed to determine whether they belong to general classes of words for which the system has knowledge or are variants of known words. If a word cannot be recognized by the morphology routine a message is printed out indicating the offending work and the user is asked to retype his request.

Once the complete sentence is accepted, the "case-oriented" parser attempts to find the main verb and to arrange the noun phrases in the sentence as "cases" of the main verb. (See Fillmore [3] for the theory of case grammar and Celce-Murcia [2] for an early implemtation of a case-oriented parser.) The processor examines the output of the parser and attempts to identify the request as one of the classes of requests it knows about. The classification is along the general lines described in Section 4.4. Further processing towards response generation is based on special knowledge about the request type. A request for data, for example, is processed very differently from a request for a model value or a yes-no question.

3. The Experiment

To test whether such a system would be natural and useful to managers we conducted an experiment in which 23 subjects, chosen to have some acquaintance with the concepts and vocabulary of management, were asked to explore a realistic problem situation. Analysis of their problem-solving protocols was used to determine the facilities that would be required for a system of this type. The sentences used by them in framing their requests were analyzed to determine the vocabulary and the parsing capability required for such a system.

The situation placed the subject in the role of the president of a company that manufactures lead batteries and confronted him with the operating results for the last year which show that although

sales increased by 20 percent profits decreased. He was asked to reach an understanding of the situation sufficient to recommend a course of action with the assistance of a Perfect System that would respond to any request he made. By allowing unrestricted English input we attempted to create conditions in which the subjects could behave as naturally as possible, unhampered by technological restrictions.

The Perfect System was implemented by interconnecting two consoles logged into a time-sharing system. Requests typed at the subject's console appeared at the experimenter's console and he was able to create responses to them by calling on the prototype system as well as on a set of prepared programs. One of these allowed him to create responses by typing them in directly. The subject saw only the output of the programs invoked by the experimenter. The system was perfect in the sense that it could respond to any request that the experimenter could understand and for which he could create answers.

4. The Results

The behavior of the subjects and the requests made by them were analyzed to provide the following results:

1. Behavioral reactions to the system and the setting.
2. Vocabulary requirements for an English language management-support system.
3. Parsing requirements for an English language management-support system.
4. Facilities required to support management problem-solving in a specific domain.
5. Knowledge requirements for a domain-specific English language management-support system. In other words, the knowledge required to provide the facilities described in 4.
6. Conceptual structures and strategies used by the subjects to solve the problem.

These results are summarized in the following six sub-sections.

4 1 Behavioral Reactions to the System and the Setting

In every case, the subject read the problem scenario and the instructions to use the system and went readily to work. In a few cases the mechanics of editing and sending a request had to be explained. This was done quite rapidly, however, and the subject was at work within a few minutes after reading the documentation.

Some subjects started out with very simple requests for single items of data. As they gained confidence in the system, they asked more demanding questions requesting blocks of data, invoking models and performing complex computations on the data. They would then go on to ask "what if" questions, define models and ask for underlying causes. Thus, the subjects explored the capabilities of the system and gained confidence in it while solving the problem. They did this by gradually increasing the complexity of the questions and by asking direct questions about system capability:

"Can you format reports?"

"Do you perform mathematical computations?".

One of the initial, fuzzy notions we had was that managers "should be able to talk to the system like a human being". And indeed, after a few questions, the subjects began to treat it like one. Their English was informal, they were cavalier about sentence forms and style and tended to ignore inessentials like punctuation. The formality of having to type in the requests and the knowledge that they were interacting with a computer system did seem to have some effect on the input, however. Their sentences were short and simple and for the most part coherent and unambiguous.

A few subjects expressed their impatience at having to precede all requests for data with "what is" by leaving it out. Others attempted to set specifications to be obeyed over the next set of questions. Yet another form of economizing on input was to define simple models and then merely specify parameters in subsequent questions. Thus, some subjects seemed to feel that English was a little cumbersome for routine data retrieval. It may be desirable, therefore, to build a command language on top of the English system.

In summary, all the subjects took quite naturally to the system and were able to work comfortably with it without significant problems. After the experiment, most of them commented that the system "would be very useful if it could be implemented". A high-level manager for a retail food chain felt it would be very useful to train store managers and also to manage individual profit centers like a bakery.

4.2 The Vocabulary

The 496 sentences used by the subjects were formed from 358 basic words. Further, the probability of encountering new words in subsequent sentences decreased rapidly with the number of sentences. Analysis of the rate at which new words occur seems to suggest that a vocabulary of 1000 to 1500 words may be sufficient for an English language system to support a particular business application. (See Malhotra [4].)

Analysis of the words used in the subjects' requests allows us to develop the requirements for the morphological analysis program that attempts to associate each word of the input with appropriate pieces of knowledge contained in the system. If a word is not contained in the dictionary the program should check to see if it belongs to a class of words it knows about. If so, it can create the required knowledge from general knowledge about the class and the special characteristics of the word. In this way it can recognize inflected forms of known words (ran and running from run), noun idioms (cost of goods sold), numerical nomilizations (products 1, 2 and 3), contractions (what's, I'm) and abbreviations ($, info, OH, mfg). The program must also be able to make allowances for common misspellings and for run in words such as "whatis". (See Tietelman [6].)

It seems desirable to allow the user to define new words conversationally as part of his interaction. The problem is, however, that, except in special cases such as names of defined models, each work in a knowledge-based system has a significant amount of knowledge attached to it. Without this knowledge it cannot be processed correctly, if at all. Since it is unrealistic to expect the user to be able to supply this knowledge (in the proper format) it seems best not to allow words to be defined conversationally.

4.3 The Parsing Requirements

A basic parser that analyzes sentences syntactically to match ten known sentence types and uses semantic knowledge to put together a canonical representation of the sentence was able to parse 78 percent of the sentences obtained from the users. The parser also possesses the capability to analyze simple conjunction forms and initial preposition groups and to ignore the noise word "please". A detailed description of the parser is given by Malhotra [4]. Figure 2. shows the ten sentence types and their relative frequency of occurrence in the parsed sentences.

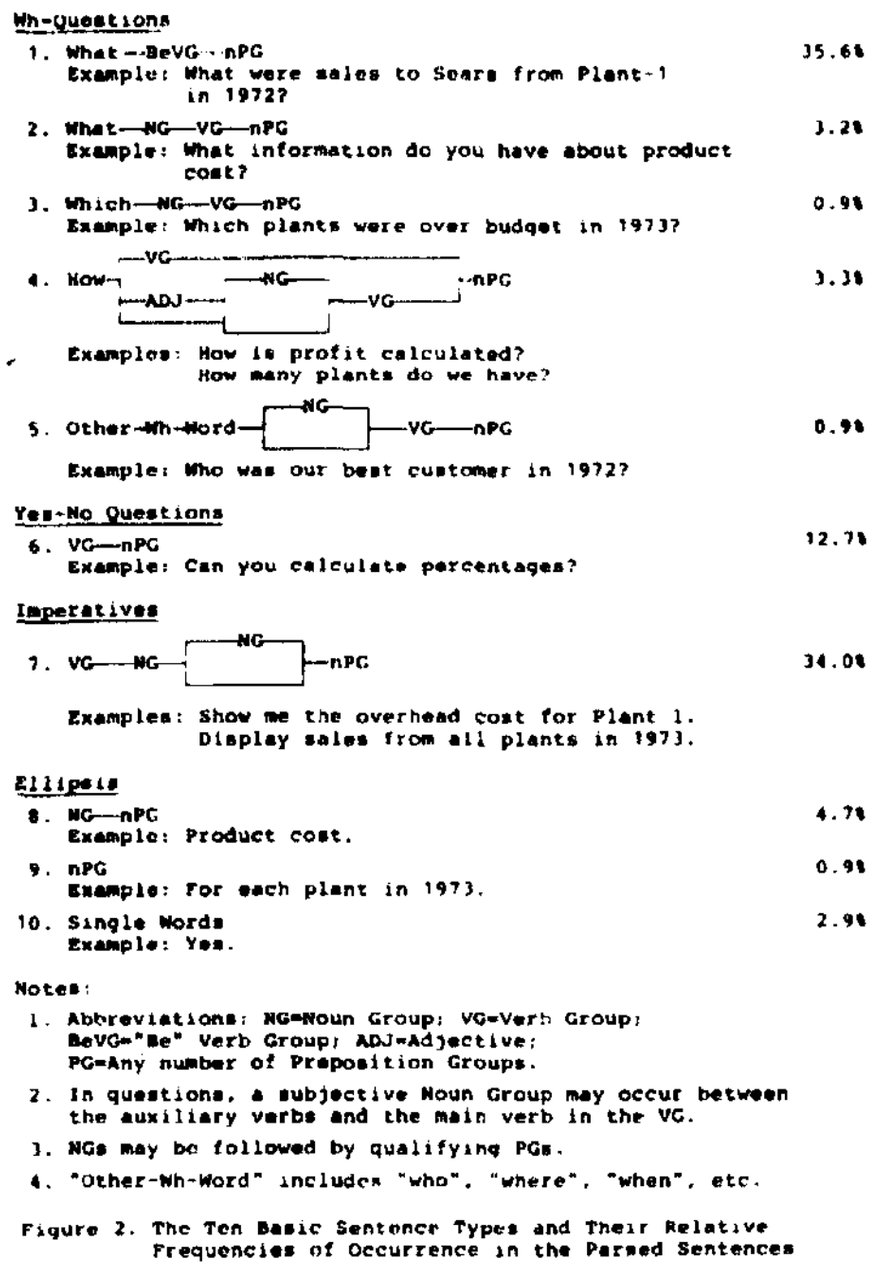

Figure 2. The Ten Basic Sentence Types and Their Relative Frequencies of Occurrence in the Parsed Sentences

The frequency of sentences classified by sentence type seems to follow the well known Pareto distribution [5]. This often appears in analyzing occurrence frequencies by class; be they sales by item or the amount of damage by fire. Typically, a few classes account for a large percentage of the occurrences. Thus, the majority of the sentences fall into a few types but, if the tail is to be covered, a number of additional sentence types must be added. We estimate that a parser capable of analyzing some twenty sentence types and other syntactic conventions will be able to provide adequate fluency and completeness. It will not be able to accept anything the user wishes to say but will accept a subset of English that is "habitable" in the sense of Watt {7].

4 4 The Facilities Required

The requests obtained from the subjects, which can be considered to be typical of those that will be made to a management-support system of this kind, can be divided into two major classes: requests for information about the problem situation and requests for information about the contents and capabilities of the system. The following are typical examples extracted from the user protocols.

REQUESTS ABOUT THE PROBLEM SITUATION

Data Retrieval

What was production by plant by product?

How much did we sell to Sears in '72?

Functions of Data

What is the ratio of overhead cost to sales for the last 2 years?

What is the percentage increase in sales of each product in 1973?

Models and What~If Questions

What was contribution margin for each plant?

What would profits have been if there was no deviation between selling price and list price?

Would sales have decreased the price if product 5 was raised to give a margin of $2?

Properties of Entities and Identity Questions

How many plants do we have?

Which products are made by plant 4?

Yes-No Questions

1. About the Corporation

   Do we have any repeat customers?

   Was any equipment purchased for long term depreciation?

2. Asking *if* a Sub-Problem Exists

   Did the product mix change for any plant whose profitability had decreased from last year?

   Were profit margins maintained in 1973?

Model Definitions

Define p-cost to be the sum of overhead and production cost.

Define chcost =

((Cost in 1973 - Cost in 1972)/(Cost in 1972))

REQUESTS ABOUT THE SYSTEM

Regarding Capability

1. Computational Capability

   Can you calculate percentages?

   List all the functions you can perform.

2. Content Capability

   Can you produce a profit figure for each product at a specific plant?

   Can you give me data on product mix from each plant?

Regarding Contents

How far back does your information go?

Do you have a forecasting model for demand?

Do you have any information on customer satisfaction?

Regarding Composition of Data Items

Do overhead costs vary with volume?

Where does transportation cost get included?

What makes up operating costs?

Definition of Data Items and Models

Define the terms "unit cost" and "unit price".

How is profit calculated?

What is the definition of profit for a product?

The above examples indicate the kind of facilities requested by the subjects. Requests inquiring about the causes and motivation of various states and events cannot, of course, be answered by such a system. Similarly, it cannot accept information and respond to requests for data or facilities it does not possess.

4.5 The Knowledge Base

A variety of different kinds of knowledge is required to analyze and respond to the requests obtained from the subjects. The system needs to have knowledge about data, about models, and about functions of data and model values. For each of these it requires different kinds of information. The system also needs to know the properties of entities and deduction rules that can be used to relate questioned properties to stored properties. In addition to knowledge about the problem situation and the environment the system also needs to know how to respond to different types of requests including those that are ambiguous, incorrect or cannot be analyzed by the system. If a request cannot be processed the system should ask the user to rephrase it and provide as much information as possible to assist him in doing so.

The total amount of knowledge required to respond to the 496 requests made by the subjects is presented in Appendix III of Malhotra (4]. Although the amount of knowledge is large, it is not intractable and it seems feasible to incorporate it into a management-support system.

4.6 Analysis of Problem Solving Behavior

A paradigm of coming to grips with problem situations is described in Malhotra [4]. This states that managers attempt to understand a gross problem by checking lists of sub-problems that may contribute to it. This hierarchical process stops with the isolation of a set of sub-problems that can either be alleviated directly by actions or decisions or for which more information or expertise is required for further analysis.

In cases where the set of potential sub-problems does not yield an existing problem the manager follows one or more of three strategies: he goes back over the list of sub-problems and rechecks each one, perhaps using different data and different functions to test if it exists; he attempts to generate additional potential sub-problems; or he reverts to more basic concepts and uses these to attack the problem.

The paradigm was supported by the problem-solving protocols of the subjects. Its validation indicates that managers use a few basic processes to try to understand problem situations. Thus, if the system provides capabilities to support these basic processes it will be useful for a wide range of management problems. Moreover, if problem-solving processes are found to be stable over a wide range of managers or if they can be identified for a set of managers then the design of the system can be based upon them.

This brief description does not do justice to the paradigm. It is included here to support the generality of our results.

5. Preferred Answering Strategies

Our basic contention underlying answer generation was that people appreciate brevity and tire of repetition. If they have faith in the system and it analyzes their requests carefully there should be tittle need to specify the question in the response. If data is asked for, it should be presented without any explanation. If the question is "Who is our largest customer?" the answer should be "Sears", not "Our largest customer is Sears". Defaults and assumptions made by the system should, however, be stated along with the answer on the principle that the user should know all the information used in generating the answer.

In some cases, a request can be answered in a number of ways. Some of these are preferable to others because they lead to system efficiency or because they support the user's problem-solving process better. The following sub-sections provide examples of prefered answering strategies.

5.1 Yes-No Questions

The system should respond to questions of the type:

"Do you have sales figures?"

"Can you show me overhead cost?"

by attempting to provide the data mentioned.

Questions of the type:

"Is transportation cost included in overhead?"

should be replied to with either a "yes" or with information about where transportation cost is really included. In general* the system should try to indicate the correct state of affairs rather than respond to such questions with merely a "no". (See also Winograd [9].)

In some cases, additional information should also be included with a "yes" answer. For example:

Was actual expense in plant 4 higher than budget?"

If it was, the system should anticipate the following "By how much?" and provide the variance.

5.2 Identity Questions After Yes-No Questions

Yes-no questions asking whether entities with given properties exist are often followed by questions asking for their identities.

"Did any plants exceed their production budget in 1973?"

"Which ones?"

Since this is a common sequence, also reported by Woods [8], it seems desirable to check the properties of all the relevant entities in answering the yes-no questions, not stopping after the first positive instance, and to maintain the list of positive instances in a special register to answer the identity question.

5.3 Fuzzy Discriminating Functions

Some subjects asked yes-no questions, testing the existence of a

sub-problem, that required the system to make a judgement:

"Were profit margin* maintained in 1973?"

"Did unit coats increase significantly last year?"

Such questions are identified by "fuzzy" words such as "maintained", "changed" and "same". Since the system cannot provide the judgement needed to answer these questions, it should provide the data and ask the subject to draw his own conclusions.

5.4 Free Standing Noun Groups

The SOPHIE system [1] has a default that if s user types in a noun group which is a "measurement" he is assumed to want its value. Some of the subjects tended to drop the "what is" before a request for a data item and type just the noun group, optionally followed by preposition groups. The default does, therefore, seem to make good sense.

5.5 Definitions

.Every entity known to the system should have a prepared definition and description that should be printed out if it is asked for or if the user makes an incorrect request related to It. In fact, there probably should be a definition and special messages to respond to different ways in which a request regarding that entity can be misphrased.

6. System Characteristics

It is time now to reconsider the system requirements described above in light of our assessment of the state of the art. In general, most of the facilities required can be provided with sufficient power and generality. Some of them are difficult to provide but, on the balance, there seems to be adequate capability to build a system that will be very useful.

Of all the facilities requested by the subjects data retrieval was the most popular. This is relatively straightforward to provide. The major difficulty is the matching of noun groups used to specify data with data names known to the system. This is discussed in detail in Mslhotra [4].

Formatting the answers and aligning the figures in tables with the decimal points one below the other and with commas after every three positions to indicate significance was found to be very important to the subjects' problem-solving process. Some subjects also wanted to change the number of significant digits in the answer. These facilities were neglected in the prototype system but are not difficult to provide.

Some subjects wanted to examine sets of data such as the profit and loss statement and the balance sheet and could, as an extreme example, ask for the general ledger. Retrieving and presenting these named sets of data also does not present any significant technical problems.

Some subjects wished to set a series of specifications to be used for all the succeeding requests until reset. For example, "Provide the following data for plant 2." Such a facility can be implemented by storing the specifications in special registers that are checked during the process of creating specifications for data retrieval. It seems desirable, however, to print out these specifications each time they are used since the user may forget he has set them and misinterpret the answers. As mentioned earlier, there seems to be a need for a simple command language for users who would use the system for a few, fixed types of data retrieval. Such s facility can be provided easily and efficiently and would considerably reduce the burden of typing in long requests.

The system should provide at least the following functions: "percentage", "average", "maximum", "minimum", "increase", "sum", "difference", "change", "variance" (both accounting and statistical), and "distribution". Functional capabilities are fairly straightforward to provide and the system design should lean towards prolixity rather than parsimony.

One of the more significant results of the experiment was the importance of models to the problem-solving process. Not only did subjects ask for models as naturally as they asked for data, but most of them wanted to define new models and ask what-if questions that require models to answer them. It is very difficult, however to provide conversational motel-building facilities. The ability to describe them in English sentences and have the system set up appropriate internal structures is related to the general problem of having computer systems learn from information presented to them. Besides, the knowledge required *to* build models is very complex and it is difficult to describe models in single sentences. Learning from natural language input and the comprehension of a number of connected sentences is somewhat beyond the current state of the art.

Since it does not seem feasible to provide model building facilities in natural English, the system should attempt to provide them in some other manner. We suggest that whenever the user attempts to define a model the system should invoke a special modelling sub-system. This sub-system would initiate s structured interaction with the user during which it would ask questions and the user would supply the information needed to build the model. The sub-system would, of course, make extensive use of system knowledge to frame the questions. In this manner, the user would have access to a fairly powerful model-building facility rather than a rudimentary, conversational model-building system.

What-if questions ask for the value of a target variable given hypothetical values for contributing parameters and states of nature. Such questions can only be answered if a model exists with the target variable as output and the specified parameters and states of nature as inputs. The response generator to what-if questions should, therefore, start by looking for an appropriate model. If such a model can be found, the inputs should be picked up from the sentence or supplied by defaults and the answer created. If, however, a model cannot be found, the user should be told so and, if he wishes, led into the mode) building sub-system.

The ability to answer yes-no questions and identity questions is extremely important to the success of a managerial question-answering system. Indeed, yes-no questions were the third most popular syntactic type in the sentences obtained from users. Such questions are difficult to answer because special pieces of knowledge are required to understand them. In the sentences:

"Who is our largest customer?"

"Is Sears our largest customer?"

The word "largest" acquires a special meaning, namely "the one who bought the most from us". The utility of such a piece of

knowledge is restricted to a narrow range of requests and a number of such pieces are required. Nevertheless, it seems possible and necessary to provide adequate facilities in these areas.

The analysis presented above and in more detail in Malhotra [4] shows that it is now possible to implement a system that mirrors the complexity of the managerial problem-solving process and allows both new and experienced users to work easily and naturally with it. Powerful technology in natural language processing and knowledge representation and processing now exists and is being strengthened further. The next logical step seems to be to implement such a system for a real situation and learn from an analysis of the use that managers make of it. This is probably the most effective way to make progress in responsive support systems for managers.

7. Implementation Issues

Since our main result is that an English language support system for managers is feasible and one of the obvious directions for future research is to implement such a system, we should touch briefly on implementation issues. First, since the amount of knowledge required, although tractable, is large such a system should be built for specific, limited problem domains. There may, thus, be a support system for budgeting, another for controlling production costs and so on.

Second, such a system would resemble a service rather than a product It would have to be brought up especially for each problem area and it would change and grow with the managers and their jobs and their understanding of the problem domain. A knowledge-based system implies continuous modification. It seems best, at this stage, however, to relegate the function of adding knowledge to the system to an intermittent, background, system maintenance phase. The process of adding to the system will be extremely important to its success, however.

Third, our investigation assumed a simple data base structured in the form of arrays. Real world data bases are, however, very complex, consisting of sequential, indexed sequential, random, inverted and chained files. The retrieval mechanisms from such fields will need to be very sophisticated and use knowledge about the structure of the files. Furthermore, certain kinds of questions cannot be answered from an inappropriately structured data base without a record by record search of the entire data base. These questions must be considered inappropriate for the data base and should receive an "error" response.

Bibliography

1. Burton, R R, "Semantically Centered Parsing System for Mixed Initiative CAI Systems", presented at the Twelfth Annual Meeting of the Association for Computational Linguistics, Amherst, Mass.. July 26-27, 1974.

2. Celce-Murcia, M., "Paradigms for Sentence Recognition.", in System Development Corporation final report No. HRT-15092/7909.

3. Fillmore, C. T., "The Case for Case.", in "Universals in Linguistic Theory", Bach, E., and Harms R., (ed). Holt, Rinehart and Winston, 1968.

4. Malhotra, A., "Design Requirements for a Knowledge-Based English Language System for Management: An Experimental Analysis.", Ph.D. Thesis, Sloan School of Management, M I T , Cambridge, Mass., February 1975.

5. Parzen, E., "Modern Probability Theory and its Applications", John Wiley. New York, N Y ., 1960.

6. Teitelman, W.$_t$ "Automatic Programmering: The Programmer's Assistant.", FJCC, December 1972.

7. Watt W. C, "Habiubtlity", American Documentation, Volume 19, No. 3, July 1968.

8. Woods, W. A., Kaplan, R. M , and Nash-Webber B., "The Lunar Sciences Natural Language System: Final Report", Bolt, Beranek and Newman, Cambridge, Mass., June 15, 1972.

9. Winograd, T.. "Understanding Natural Language.", Academic Press, New York. 1972.