(NASA-CR-184525) NATURAL LANGUAGE QUERY SYSTEM DESIGN FOR INTERACTIVE INFORMATION STORAGE AND RETRIEVAL SYSTEMS M.S. Thesis Final Report, 1 (University of Southwestern Louisiana, Lafayette, Center for Advanced

N89-14965

Unclas
G3/82 0183563

USL / DBMS NASA / RECON

WORKING PAPER SERIES

Report Number

DBMS.NASA/RECON-17

The USL/DBMS NASA/RECON Working Paper Series contains a collection of reports representing results of activities being conducted by the Computer Science Department of the University of Southwestern Louisiana pursuant to the specifications of National Aeronautics and Space Administration Contract Number NASW-3846. The work on this contract is being performed jointly by the University of Southwestern Louisiana and Southern University.

For more information, contact:

Wayne D. Dominick

Editor
USL/DBMS NASA/RECON Working Paper Series
Computer Science Department
University of Southwestern Louisiana
P. O. Box 44330
Lafayette, Louisiana 70504
(318) 231-6308

NATURAL LANGUAGE QUERY SYSTEM DESIGN FOR INTERACTIVE INFORMATION STORAGE AND RETRIEVAL SYSTEMS

I-Hsiung Liu

The University of Southwestern Louisiana
Center for Advanced Computer Studies
Lafayette, Louisiana

April 22, 1985

NATURAL LANGUAGE QUERY SYSTEM DESIGN

FOR

INTERACTIVE INFORMATION STORAGE AND RETRIEVAL SYSTEMS

A Thesis

Presented to

The Graduate Faculty of

The University of Southwestern Louisiana

In Partial Fulfillment of the

Requirements for the Degree

Master of Sciences

I-Hsiung Liu

January 1985

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my committee chairman, Dr. Wayne D. Dominick for his guidance and valuable comments during the preparation of this thesis. I would also like to thank Dr. W. Edwards, Jr. and Dr. T. Cousins for their helpful suggestions during the course of this research. Special thanks go to my fellow graduate students M. Granier, C. Pena and S. Triantafyllopoulos for proofreading and encouragement.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF FIGURES. vi

LIST OF EXAMPLES vii

Chapter Page

1 INTRODUCTION. 1

1.1 Casual User-System Interaction 1

1.2 Objectives and Methodology. 4

1.3 Scope of the Project. 5

2 IMPORTANCE OF NATURAL LANGUAGE QUERY SYSTEMS DEVELOPMENT. 10

2.1 Formal Query Approach Versus Natural Language Query Approach. 10

2.2 Dynamics of Information Storage and Retrieval Systems. 18

2.2.1 Tasks of Information Retrieval. . . .20

2.2.2 Databases 27

2.2.3 Users 30

2.2.4 User Interfaces 33

2.3 Formal Query Interfaces and Information Retrieval 39

2.3.1 Formal Query Database Searching . . .40

2.3.2 Information Content 56

2.4 Natural Language Interfaces and Information Retrieval. 62

2.4.1 Natural Language Database Searching .64

2.4.2 Demands on the Database 65

3 NATURAL LANGUAGE QUERY SYSTEMS DEVELOPMENT. . . .70

3.1 Language Capabilities of Natural Language Query Systems. 72

3.2 Phases of Natural Language Processing. . . .78

3.2.1 Syntactic Analysis. 78

3.2.2 Semantic Analysis 90

3.2.3 Execution Phase 99

3.2.4 Natural Language Response Generation.102

4 FRAMEWORK 104

4.1 Design Methodology 106

4.1.1 Redundant Words/Phrase. 113

4.1.2 Ambiguity and Vagueness 113

4.1.3 Error Detection/Correction. 116

4.2 Interfaces of Natural Language Query Systems. 118

4.3 Information Sources. 125

4.3.1 Information About Part of a Document File 125

4.3.2 Information About a Document File . .126

4.4 Features of a Natural Language Query System.131

4.4.1 The ATN Grammar Parser. 131

4.4.2 The Interpreter 137

4.4.3 The Dialogue Control. 140

4.4.4 The Formal Query Generator. 141

4.4.5 Natural Language Response Generators.143

4.5 Overview 147

5 CONCLUSIONS 150

APPENDIX A Hendrix Naltural Language Capability List .153

BIBLIOGRAPHY 155

ABSTRACT 163

BIOGRAPHICAL SKETCH. 164

LIST OF FIGURES

Figure		Page
2.1	Information Retrieval Tasks	21
2.2	Levels of Data Base Design	28
2.3	Interactive Information Retrieval Sequence	42
2.4	Example of Sign-On Operation	44
2.5	Stages of Information Retrieval	60
2.6	Levels of Data Base Design (Revised)	67
3.1	Example of Transformational Grammar	89
3.2	Example of "Pattern --> Action" Rules	94
4.1	Overall Structure of a NLQS	119
4.2	Natural Language Data Base Searching	124
4.3	Portion of Semantic Networks (1)	128
4.4	Semantic Relationships Between TEXT TERM and DOC REF CODES	129
4.5	Portion of Semantic Networks (2)	130

LIST OF EXAMPLES

Examples		Page
1	A Standard Formal Query for Information Retrieval	11
2	A Natural Language Query for Information Retrieval	13
3	An Ambiguous Natural Language Query	17
4	The Formal Counterpart of Example 3	17
5	Search Command and Search Term	46
6	Queries with Relational Operators	48
7	Queries with Adjacency Operators	49
8	Queries with Boolean Operators	49
9	Queries with Related Term Capability	52
10	An Example of the Use of a Context Free Grammar	86

CHAPTER 1

INTRODUCTION

The major function of a computerized information system is to enable its users to retrieve and/or modify a specific subset of the data in the data bases, as well as to provide support to its users in their decision-making activities. In other words, the computerized information system serves its users and satisfies their information needs. Thus, the success or failure of an information system is ultimately decided by the users the system serves.

One of the criteria in evaluating an information system from the user's viewpoint is whether the system allows him to communicate with it conveniently and satisfy his immediate needs for information. Therefore, the interface problem in user/system interaction must be considered seriously while developing an information system.

1.1 Casual User-System Interaction

Recognizing the interface problem, the concept of a multi-level query system has been adopted in developing

different database sublanguages in many information systems to facilitate the communication between the computer and various user groups. In essence, the current approaches of multi-level query systems assume that the human/computer interaction is a one-way communication process. In other words, in the process of information retrieval, a user is responsible for providing queries understandable to the computer system, and, in order to use any query language, the user has to have some understanding of how the computer system represents the data. Thus, prior to performing information retrieval, a user has to learn at least one specific database sublanguage supported by the system, procedural or non-procedural as the case may be [Codd 74; Eason 75; Harris 78; Martin 80]. Consequently these multi-level query systems have been described as language interfaces developed for experienced users and they really cannot cope with the nature of potentially the largest group of the user population, namely, casual users [Codd 74; Martin 82].

Casual users, as defined by Codd [Codd 74], are ones "whose interactions with the system are irregular in time and motivated by (their) jobs or social roles." Such users may not only lack knowledge about computers, programming, formal logic, or relations, but also they are not willing to learn an artificial language. The only query language which they

are willing to use to interact with an information system is their native language.

Because of the above characteristics, casual users are usually categorized as the "indirect" users of an information system, and, for most of them, the fulfillment of their information needs is through an intermediary who has the required knowledge for using database sublanguages and who is able to translate the casual user's request into a specific database sublanguage [Harris 78; Martin 82; Wanger 76].

Smith [Smith 80] studied the development of computerized information systems, and made the following statements. He suggested that, at the early stage of information system development, due to the high cost and special usages of information systems, as well as the knowledge required to operate those systems, casual users rarely used the system directly to obtain information. Thus, these users were categorized as "indirect end-users" and the problem of direct casual user/system interaction was not considered to be the most important issue within information system development. However, along with the increasing diffusion of the utilization of low-cost computers and computerized information systems, the casual user/system interaction problem becomes essential to the development of future information systems. Therefore, the development of natural language query systems which allow casual users to employ

freely their native languages to specify what they want while interacting with an information system has been seriously considered and discussed by many computer scientists (for example, [Codd 74]; [Hendrix 81]; [Kaplan 78]; [Salton 83]).

1.2 Objectives and Methodology

The main objective of this research activity is to propose a framework for exploring the nature, scope and content of the evolving topic of natural language query systems (NLQS). A framework, as defined by Sprague [Sprague 80], "identifies the relationships between parts (of a system), and reveals the areas in which further development will be required." By following this definition, the construction of the framework for this research activity is divided into three phases.

The first phase will describe the rationale for NLQS development. In this phase, the arguments over NLQS development will be critically examined, and the weakness of the conventional formal query interface will be explored.

The second phase will briefly review the development of an NLQS. This phase will identify the language capabilities that should be integrated into an NLQS; It will also examine the alternative concepts and approaches to natural language interface development such that an appropriate approach to

NLQS development can be identified.

The third phase will propose an appropriate approach that can be applied in NLQS development. Then, the overall structure of an NLQS will be presented such that the relationships between components of the entire system, including the natural language interface, formal query interface, and the bibliographic database, can be revealed. Finally, a descriptive model of the NLQS will be presented so that the functions performed by the system in response to natural language input can be examined, and so that the performance and capabilities of the resulting system can be assessed.

It is expected that the framework proposed in this research activity, although it will only specify the high-level structure and general concepts of NLQS development for IS&R systems, will serve as the guideline for the future development of an interactive natural language information system.

1.3 Scope of the Research

"Natural language understanding", which implies the construction of a mapping between a natural language source and its target representation, is one of the major concerns of this research. An ideal NLQS should have the intelligence

to understand any of the user's queries in the form of natural language, as well as the capability of generating responses correctly and promptly.

Restricted by the complexity and non-determinism of the natural language understanding process (1) this research will be limited in its scope, yet will provide general concepts for the development of an NLQS. The scope of this research will be defined in terms of the database environment as follows.

(1) The Information Storage and Retrieval System

The IS&R system, as described by [Wiederhold 83] and [Salton 83], has the following characteristics:

(a) IS&R systems maintain data about collections of publications. They typically have records composed of data items such as authors, keywords, titles, abstracts, and so on.

(b) The entities stored by IS&R systems are complete, independent, and in natural language textual form.

---

(1) The problems of natural language processing have been stated by a number of authors (see [Hendrix 81], [Salton 83], [Rich 83] and [Barr 83]). Also, in Chapter 3, an overview of the development of natural language processing will be presented.

(c) The data stored in IS&R databases is selected, prepared, and classified with the expectation of eventual retrieval and use. Thus, the data stored in such a system is already potential information.

(d) The primary function of the IS&R system is information retrieval.

The databases of IS&R systems are usually refered to as bibliographic databases since the content of such databases, as described above, is document or publication surrogates. Such a bibliographic database environment was selected due to the following considerations.

First, the problem of using natural language in queries in a database environment is much less difficult than the problem of understanding natural language in general. In a formatted database which relates to a specific problem area, such as the bibliographic database environment, the user will operate within a well-defined context. A limited area of discourse makes it possible to parse queries without confusion due to the ambiguity of natural language [Wiederhold 83].

Second, as discussed by Smith [Smith 80] and Hendrix [Hendrix 81], the application which has attracted the most theoretical interest within NLQS has been that of database processing. The reason is that databases are among the few

types of symbolic knowledge representations that are indexed in a computationally efficient manner, are in widespread use, and have well-understood semantics. In the case of bibliographic database searching, the information being retrieved are document items as stored in their natural language textual form. Thus, the semantics are well-defined and well-understood.

Third, an IS&R system is an information system which is used to store items of information that need to be processed, searched, retrieved, and disseminated to various user groups. Thus, the IS&R system shares many of the concerns of other information systems, such as generalized database systems. In particular, it is necessary to choose efficient organizations for the stored records and rapid search procedures capable of effective methods for disseminating the retrieved information, and effective method for interacting with the users [Salton 83].

Finally, the IS&R system, in contrast to generalized data base management systems that process structured data and in contrast to question-answering systems that use complex information organizations and inference procedures, is normally used to handle bibliographic records and textual data. However, in an extended sense, any information system designed to augment the state of human knowledge and to aid human activities does utilize concepts and procedures from

the IS&R system [Salton 83].

Based on the above reasons, although this research restricts the scope of study to the bibliographic database environment, it is believed that the results obtained from this research can be applied to the development of natural language interfaces within other data base environments.

(2) The Task of Information Retrieval

Since this research is intended to construct a framework for NLQS development, the first issue to be considered is "to define precisely what the underlying task is" [Rich 83] such that the functions performed by the system and the type of the user's requests can be identified and satisfied.

Although most information systems allow the user to retrieve, modify, and/or delete data within the database by using some specific database sublanguage, information retrieval is typically the primary function of any information system, and of IS&R systems, in particular. Thus, pursuant to developing a new user interface, the first issue to be considered is how best to facilitate the task of information retrieval, particularly when such a development is aimed at supporting the communication between casual users and the system [Heaps 78].

CHAPTER 2

IMPORTANCE OF NATURAL LANGUAGE QUERY SYSTEMS DEVELOPMENT

2.1 Formal Query Approach Versus Natural Language Query Approach

In order for an information system to attract casual users, interfaces must be provided that approximate the special user terminology and relevant conceptualizations [Lockemann 75]. Based on this consideration, two major approaches toward casual user/system interface development, namely, the formal query approach and the natural language query approach, are to be discussed and compared.

Although both of these two approaches intend to develop user interfaces which allow a user to specify "what" he wants the machine to do instead of supplying his intelligence to instruct the machine with precision exactly "how" to do his job step by step (1) , these two approaches have some major differences.

---

(1) The formal query approach and the natural language query approach represent the trends of casual user/system interface develoment. These trends can be well understood by referencing the What-To-How spectrum proposed by [Feigenbaum 74].

The major distinction between the formal query interface and the natural language interface is that the latter allows many valid English phrasings of a request, whereas the former only allows a single canonical wording, or perhaps a few very similar wordings [Harris 78]. This distinction can be attributed to the underlying concepts of these two approaches.

The formal query approach is based on conventional data models that consist of computer-oriented, syntactic data structures. In using the formal query interface, a user is freed from being concerned with the way in which the data is actually stored and accessed, but he is required to express his query in terms of the logical content of the data. In other words, the user has to have some understanding of how the computer system has represented the data, such as the specific data structure in the schema, the ways relations are used, and the existence of entities. In addition, the user also needs to know the meaning and usages of various Boolean operators in order to express most queries [Martin 82; Harris 78; McLeod 78]. The following example illustrates such requirements.

```
SELECT TITLE EQ "DATA BASE" AND
AUTHOR EQ "MARTIN";
```

Example 1. A Standard Formal Query for Information Retrieval

To submit a query as shown in Example 1, the user is required to have the following knowledge:

(1) To know the command name and usage of the search command of a specific IS&R system, such as "SELECT".

(2) To know the desired search field as defined by a specific IS&R system, such as "TITLE", "AUTHOR" or "ABSTRACT".

(3) To know the usage of various Boolean operators, such as "AND", "OR", and "NOT".

(4) To know the usage of different relational operators, such as "EQ", "GT", "LT", etc.

(5) Most of all, to know the syntax of the search language defined by a specific IS&R system, such as the order of the operations, the use of special punctuation symbols, and so on.

Although the natural language interface is also based on data models, it is user-oriented rather than system-oriented. Via this interface, a user expresses his query of a database in a subset of natural language such as English, and phrases it in the way he perceives the data rather than the way the machine perceives it. Therefore, the user need not be aware of the data structures used in the conceptual schema of the database [McLeod 78; Codd 74].

LIST ALL THE BOOKS WRITTEN BY MARTIN
WITH A TITLE CONTAINING DATA BASE.

Example 2. A Natural Language Query for Information Retrieval

By viewing the above examples, although both the formal query approach and the natural language query approach provide interfaces which allow the user to apply ordinary English terminology to perform information retrieval, the natural language query approach has some distinct advantages over the formal query approach.

First, while using a formal query language, the user has to translate his request, which he formulated in terms of the natural constructs of the application environment, into an artificial language which explicitly identifies the data structures of the data base schema. By examining the above examples, Example 2 may be the user's request constructed in a natural environment, but, in order to interact with the system and retrieve the desired information, he has to translate this request into a syntactic-restricted artificial language such as the query shown in Example 1. In other words, the user is required to transform his natural language request into a query construct utilizing the specific search command, criteria format and operations, and express them in the specific format required by the system. Since this translation process requires some knowledge of the database

architecture which most casual users are not interested in learning, an intermediary is usually introduced as the mechanism for realizing this translation process. Thus, the purpose of the direct casual user/system interaction can not be achieved.

On the other hand, the natural language interface provides the syntactic freedom to its users and automates the process of translating a user's conception of the data into the formalism employed by the machine. Thus, as long as the user knows the information content of the database, he may enter a request as shown in Example 2 without being concerned with the data structures of the database. Therefore, this approach may encourage the casual user to directly communicate with the system for satisfaction of his information needs.

A major obstacle within the natural language approach, as many computer scientists have argued, is that natural language is too ambiguous or imprecise (for example, [Martin 73; 82]). As Rowe [Rowe 82] has discussed, the reason for this perception is that natural languages are too complicated. He suggested that, through the development of artificial intelligence (AI) techniques and linguistic analysis, natural language processing can be more and more regular and rule-base and hence that the stated obstacle can be overcome.

On the other hand, formal language is more precise than natural language due to its syntactic restrictions, but it is important to know that the formulation of a formal query requires a translation process which is often done via a human intermediary. Since the intermediary needs to accept natural language input from casual users and output a formal query understandable to the computer system, ambiguity and/or human errors may occur in this process as well and information retrieval may thus fail. For example, Bailey [Bailey 73] has pointed out that the personal factors during the above process typically account for fifty percent of the reliability problems in computerized information systems. Therefore, some computer scientists suggested that it might be better to have a system that is designed to cope with ambiguity than a system which might produce the above reliability problems [Hendrix 78; 81].

Another issue often raised by some computer scientists is that natural language processing is time-consuming. This argument is usually made by comparing the response time required to handle a natural language query and that required to process a formal query [Martin 73]. The fact is that, prior to using a formal query language, a user has to learn the appropriate skills and user-supports [Eason 75]. For example, Query-By-Example (QBE) has been proved by many behavioral researchers [Greenblatt 78; Zloof 78] to be an

easy-to-use query language for non-programmer users; however, a QBE user still requires about three hours or four sessions of instruction and the knowledge of first-order predicate calculus. On the other hand, while using natural language to perform information retrieval, a user is only required to know the information content of the database and to have the ability to deal with his native language. He does not need to know the logical structure of the database, nor to learn and memorize any artificial language. Therefore, from the casual user's viewpoint, the use of a natural language interface should be more cost-effective than the use of a formal query interface.

Finally, the traditional formal query approach, as Codd [Codd 74] described, has been based on two assumptions:

> 1. Whenever a user conceives a query, he is able to formulate it accurately in English right away - that is, he will be able to convey his intent to the system faithfully and precisely at his first attempt;
>
> 2. If the user's English is beyond the restricted English understood by the system, it is the responsibility of the user alone to re-state his query in system-comprehensible English, whatever that is!

Because of these assumptions, if the user uses one or more words or a sentence structure based on his own conceptualization rather than what the computer expects, the system might pretend to understand the query, but actually

misinterpret the user's intention [Harris 78]. Example 3 will illustrate this point.

GIVE ME THE TITLES OF ALL THE ARTICLES
WRITTEN BY KAPLAN AND HARRIS.

Example 3. An Ambiguous Natural Language Query

While translating the natural language request of Example 3 into a typical formal query (1) , the user may construct the statements of Example 4.

```
FIND ALL TITLES WITH
(PUBLICATION = 'ARTICLE') AND
(AUTHORS = 'KAPLAN' AND 'HARRIS')
```

Example 4. The Formal Counterpart of Exaple 3

Although this query is syntactically correct, the system will probably reply with a 'nil' answer since it might misunderstand the user's intention and not locate any article whose co-authors are Kaplan and Harris. Since the natural language interface is developed to automate the translation process from user's conceptions into computer's conceptions, the system should be able to foresee the possibility of such misinterpretations and to deal with these problems while the

---

(1) The format of the formal query appearing in Example 4 is adopted from ADASCRIPT.

casual user interacts with the system.

As Eason [Eason 75] suggested, while the question of the effective interface between man-machine interaction is raised, the issues of concern are the human factors issues. In summary, although both the formal query approach and the natural language query approach tend to develop interfaces that facilitate the communication between casual users and information systems, the natural language query approach is, and will be, the major trend of the casual user/system interface development activity since this approach recognizes the nature of casual users and is focused on coping with the nature and information needs of casual users.

2.2 The Dynamics of Information Storage and Retrieval Systems

In the previous section, the importance of natural language query systems and the need for developing such systems for casual user/system interaction have been discussed. The purpose of the following three sections are to discuss the dynamic relationships between user interfaces and other components of an IS&R system, and to analyze the changes within such relationships resulting from the development of a natural language interface.

As Martin [Martin 80] claimed, while interacting with an information retrieval system, the principal intention of users is to perform some pre-planned function or set of actions, such as information retrieval or modification. Computers are supposed to offer ways of performing these functions that are more effective than would be the case if they were not available. In this man/computer communication process, four components are involved. (1) They are:

* Tasks
* Users
* User Interfaces
* Data Bases

These four components are not independent; they exist within a common environment and each component influences the other and thereby itself [Smith 80; Wiederhold 83]. In the following sub-sections, these interdependent components will be briefly discussed and examined in the context of the bibliographic information retrieval environment.

---

(1) In a broad sense, [Smith 80] suggested that tasks, users and communication interfaces are the components existing in the human-computer communication environment; [Winderhold 83] claimed that tasks, user interfaces and database organizations are the components which have to be matched while considering information retrieval systems. It is believed that a combination of these two assumptions will be more appropriate in describing the human-computer interaction rather than either of these assumptions individually. Therefore in this research, four interdependent components are applied in the discussion of bibliographic IS&R systems.

2.2.1 Tasks of Information Retrieval

The tasks involved in an information retrieval environment can be described in terms of the amount of knowledge required, the level of difficulty, and the retrieval methodologies applied. In this research, based on Wiederhold's [Wiederhold 83], Martin's [Martin 80], and Minker's [Minker 77] discussions, a taxonomy table is developed and shown in Figure 2.1. In this table, three types of information retrieval tasks are identified. They are document retrieval, generalized data management, and question-answering. In the following context, the three variables which determine the type of the task will be discussed so that the task of bibliographic information retrieval can be identified.

Task	Documental Retrieval	Generalized Data Management	Question-Answering
Retrieval Methodology	Fact Retrieval	Fact Retrieval Statistical Inference	Fact Retrieval Statistical Inference Deductive Inference
Knowledge Required	Domain Specified Knowledge	Domain Specified Knowledge	Domain Specific and World Knowledge
Level of Difficulty	Low	Medium	High

Figure 2.1 Information Retrieval Tasks

(1) Retrieval Methodologies

According to Wiederhold's discussion [Wiederhold 83], "fact retrieval" is the traditional view of the information retrieval operation. This retrieval methodology can be characterized by extensive use of the fetch operation and by the use of indexed or direct access. Whenever a query is entered, the system looks through its database to find the answer which is pre-stored in the database in natural language textual form. Thus, the process of fact retrieval is a simple request-response sequence.

Statistical inference implies the retrieval of large quantities of well-structured data. It is usually applied to provide meaningful data to the user when the size of a response to his request is too large to be understood. The techniques used in this approach, such as cross-tabulations and statistical processing, are mainly aimed at data reduction. During the process of data reduction, intermediate results are obtained from the database and used by the user to formulate further requests until the desired result is generated. Thus, the use of the get-next operation is predominant.

Different from the previous methodologies by which results are directly related to the user query and produced by reference to the content in the database, deductive inference implies that the query processing procedures must

evaluate the possibility of success in following a particular path in order to generate reasonable results. To perform deductive inference, the system must answer WHY-questions. In other words, the query intends to retrieve certain cause-effect relationships existing between data stored in the database. Therefore, relationships of various data in the database have to be constructed and organized to facilitate the deductive query-processing program in searching for the appropriate path. In addition, the process to materialize relationships may be very complex and may require many intermediate facts and relationships stored in the database. Thus, the database needs to be constructed using rules, representing functions, or frames, representing entities, or a combination of both so that all possible relationships and their combinations can be well described.

(2) Levels of Difficulty

Based on Smith's discussion [Smith 80], the ease or difficulty of a task can be determined by its complexity and determinism. Complexity implies "the number of states (number of events or ways of arranging them) in a particular application"; determinism indicates "the extent to which the occurrence of events or their sequence can be predicted in advance" [Smith 80].

Fact retrieval, as discussed above, is the process of a

simple request-response sequence. To process this type of query, the system simply looks through its database and fetches the appropriate textual-form data if it exists, or returns a nil answer to the user if the answer does not exist. Therefore, this task is relatively simple and deterministic.

Statistical inference is more complex since it requires many intermediate results and frequent requests in order to generate final results. As fact retrieval, statistical inference is also deterministic since the answer to a query is directly obtained from the database which contains "structured data" via the use of certain statistical analysis techniques.

Deductive inference is relatively more complex and also non-deterministic. The major reason for its complexity and non-determinism is attributed to the need for the "heuristic searching" of appropriate relationships and paths during query processing. Also, the processing of such queries usually requires the handling of unstructured data as input from a real world environment [Sprague 80].

(3) Knowledge Required

The best way of thinking about information retrieval is perhaps to consider it as the process of communication. As Wilbur Schramm, a pioneer in mass communication research,

defined:

> "when we communicate we are trying to share information, an idea, an attitude. Communication always requires at least three elements -- the source, the message, and the destination." [Schramm 54].

By following this definition, the amount of shared information is the predominant factor in evaluating the success of a communication process, or an information retrieval task. As discussed by Katz [Katz 54], the amount of information shared by two individuals involved in communication process is largely determined by the overlap of their frames of reference; in other words, their knowledge. Therefore, in the case of information retrieval, both the user and the system have to have shared knowledge about the meaning of the query and the answer to the query.

In fact retrieval and statistical inference, all of the data which is directly related to the result exists in the database, and the system produces an answer to the query by using those data. Thus, the database defines the domain of the knowledge required for both users and the system. Hence, the knowledge required for those tasks that apply methodologies such as fact retrieval or statistical inference is domain-specfied.

In deductive inference, although a certain amount of "structured data" exists in the database, the system must

also process "unstructured" data input from the real world, and integrate such data into the relationships existing in the database. Thus, two types of knowledge are required, namely, world-knowledge, representing the facts existing in the real world environment, and domain-specified knowledge, representing the information existing in the database environment.

Since domain-specified knowledge is defined by the database, to process the tasks that require this type of knowledge, the burden is on the user; in other words, the user has to know what kind of information he may obtain from the system and how he must formulate his request based on the formal syntax which the system "knows". World knowledge is defined by the real world. To process queries that are relevant to world knowledge, the system should have the capability of adjusting its database so that the input data can be integrated into the existing relationships, and also the system should have the capability to "learn" new knowledge in order to process users' requests.

2.2.2 Databases

A database, as defined by Abrial [Abrial 74], is the model of a certain reality. The state of such a model, at a given instant, represents the knowledge it has acquired from the real world. In order to allow end-users to obtain knowledge within the database, levels of abstraction, as shown in Figure 2.2, are required in the design of a database [Date 81; Nijessen 74; Ullman 82].

The "levels of abstraction" approach to database design ensures both physical and logical data independence, and allows the retrieved data to be closer to individual users' views rather than toward a machine dependent view of data [Date 81; Tsichritzis 77].

```
Fact                    REALITY

                          |
                          |
                          |
Information        USER'S LOGICAL      Information
Structure               LEVEL          Structure Description
                                       Language (ISDL)

                          |
             (logical data independence)
-------------------------+-------------------------------
                          |
Data               SYSTEM'S LOGICAL    Data Structure
Structure               LEVEL          Description Language
                                       (DDL)

                          |
             (physical data independence)
-------------------------+-------------------------------
                          |
Storage            PHYSICAL STORAGE    Storage Structure
Structure               LEVEL          Description Language
                                       (SDL)
```

Figure 2.2 Levels of Data Base Design

(Adapted from [Nijessen 74])

The most important characteristic of IS&R systems is that the publications or documents are stored in the database in their natural language textual form, including the names of authors, their professional affiliations, the title of the article, a reference to the journal name, an abstract of the article, and other related citation information. With each description, there is also a set of descriptors consisting of "keywords" or phrases, to provide additional indication of the subject matter [Heaps 78].

The extensive use of the fetch operation is another characteristic of IS&R systems. In order to ensure that the desired data can be accessed and retrieved rapidly, usually the design of bibliographic databases utilizes inverted file structures to organize data items. An inverted file stores inverted indexes which contain index terms associated with lists of document reference numbers. Whenever the system receives a search request, it examines the index in the inverted file and determines the items which satisfy the search request. In addition to inverted file structures, most IS&R systems also utilize mechanisms to classify the possible search terms in order to optimize the effectiveness of information retrieval. The classification of individual terms is done by the construction of a "thesaurus". In Section 2.3.2, inverted file structures and thesauri, as well as their functionalities, will be described.

2.2.3 Users

As with many other information systems, IS&R systems have a variety of users with different needs and abilities. Those users can be categorized into three major groups, namely, end-users or non-programmer users, mid-users or programmers, and database or system administrators [Dominick 77; Dattola 77].

(1) End Users

The end user may be defined as "the consumer of computer services provided by the other two groups" [Smith 80]. Conventionally, this user group is further divided into two groups based on the frequencies of interacting with the computer system. The first group is the so-called "parametric users" or "direct end-users". They operate a terminal on a regular basis as part of their job, such as librarians or search specialists in a service center. The other group is "casual users" as defined by Codd [Codd 74] (see Section 1.1). The casual users of IS&R systems have a wide variety of different information needs. For example, they include research scientists seeking articles relating to particular experiments, engineers trying to determine whether a patent covering some new idea has previously been obtained, and so on. Thus, the IS&R system casual users exhibit many different backgrounds or levels of expertise, and many different

reasons may lead them to use the retrieval facilities.

(2) Programmers

As explained by [McLeod 78], programmers are those who build application programs and application systems. The programmers are capable of dealing with conventional programming languages such as COBOL or PL/1, and require the power and flexibility provided by such a language. Also, they need an integrated spectrum of tools, allowing them to rely on the database interaction facilities when appropriate, but also allowing them to use a programming language to manipulate the information in a database, when appropriate.

(3) Data Base Administrators (DBAs)

Data base administrators are the people responsible for managerial control over individual databases. The tasks of a DBA includes designing, maintaining and evaluating an individual database. The major functions performed by a DBA can be described as below:

(a) Data Definition:

A DBA determines the information content of a database, and the interrelationships between data. He also determines data security rules, data access authorizations and data integrity constraints.

(b) Storage Structure and Access Definition:

A DBA determines the physical storage structure of a database, as well as the database access strategies.

(c) Backup/Recovery Process Definition:

A DBA specifies appropriate audit trails in order to collect and maintain historical information about attempted security violations and to assist in security analysis. Also, he specifies appropriate database backup and recovery protocol and procedures.

(d) Monitoring and Evaluating the Database Environment:

A DBA is responsible for defining database performance specifications and measuring database performance. Also, he should make appropriate adjustments based on the results of the performance evaluation in order to optimize database performance.

(e) Specifying the Mapping between the Logical Structure and the Physical Storage Structure of a Database:

A DBA is responsible for specifying the conceptual model of the database by using the appropriate data definition language. This conceptual model provides a logical view of the database. In addition, he also needs to specify the mapping between the conceptual model and

the actual storage structure of the database.

In order to perform the above functions, a DBA must have the knowledge of programming languages, and he has to be capable of dealing with various utility programs, such as loading routines, reorganization routines, journaling routines, and so on [Date 83].

Although all three types of users may perform the task of document retrieval in a bibliographic database environment, the last two user groups, programmers and DBAs, are primarily computer experts. The programmer must have the knowledge of the conceptual model of a database in order to develop application programs for the end users. The DBA must have the entire knowledge of a database environment in order to perform the functions described above. Therefore, while interacting with an IS&R system, the tasks performed by these two groups of users are more sophisticated than simple information retrieval activities.

On the other hand, fact retrieval is the main objective of casual users while they interact with the IS&R system. Hence, how to develop a user interface which allows this user group to express their requests conveniently becomes an important issue.

2.2.4 User Interfaces

The mechanisms of casual user-database interaction that have received the most attention among database and information systems researchers are the user-friendly, self-contained query language and the restricted natural language (for a detailed discussion, see Martin [Martin 82]). The development of such user interfaces can be described by the concept of the hierarchy of user languages [Lockemann 75]. This concept can be defined as follows:

(1) Each interface is defined in terms of a lower interface, and may itself serve as the basis for definition of a higher interface.

(2) There is exactly one interface which can not be defined in terms of another interface and hence serves as the ultimate basis for all other interfaces.

Based on the above definition, [Lockemann 75] introduced some notions for building the hierarchy. They are:

(1) Characteristics of the Root:

The root of the user language hierarchy is the database. A database is considered by Abrial [Abrial 74] and Lockemann [Lockemann 75] as the model of a certain reality. That is, the logical representation of "facts". Hence, a root should be such that it provides concepts so primitive that any reality, be it physical or conceptual, could be adequately covered by it. Abrial [Abrial 74]

has attempted to enumerate certain primitives, such as elementary objects, properties, relations, orderings, types, names, as well as sets of operators for creating, accessing, manipulating and deleting primitives.

(2) Dependencies between Successive Nodes:

The root is of little practical value to the average user. Users are mostly concerned not with all possible facts but with certain classes of facts, and with their models to reflect the corresponding limitations. That is, the modeling tools on level 1 of the hierarchy will differ from those on the root (level 0) by defining certain restrictions on the way the primitives may interact. The same is true for level 2 vis-a-vis level 1, and so on. These restrictions relate mainly to the manner in which objects may be composed into new objects, relations into new relations, and/or operations into new operations.

(3) Characterization of a Node as an Abstract Machine:

The concept of abstract machine is introduced to describe the dependencies between successive nodes more precisely. The following definition of an abstract machine is provided by Lockemann [Lockemann 75]:

"An abstract machine is a set of object types, a set of operators for manipulating objects and defined on object types, together with a control mechanism that allows (one) to construct and execute sequences of

operations."

Based on this definition, each node in the hierarchy of user languages can then be described in terms of an abstract machine.

(4) Dependencies between Abstract Machines:

According to [Lockemann 75], by assigning an abstract machine to each node in the hierarchy, some properties must be hold between two successive nodes A(i) and A(i+1). Those properties are:

(a) The resources and the functions provided by A(i) form the complete basis on which to build A(i+1). There is no way to use properties of A(i-1) in building A(i+1). Hence, every A(i) is a complete interface description in the hierarchy.

(b) Resources of A(i) used in defining new resources of A(i+1) can no longer be present in A(i+1). That is, the resources of A(i) may become resources of A(i+1) only if they are not part of a definition for another resource of A(i+1).

By following the above notions, Lockemann [Lockemann 75] claimed that there is always a new interface that can be defined in terms of its immediate predecessor; and, at any level of the hierarchy, a user can formulate his queries

without having to know the languages existing in the lower levels.

In addition to what Lockemann claimed, the use of the "hierarchy of user language" concept in developing user interfaces also has other advantages. First, the concept suggests that it is always possible to develop a "level of abstraction" on top of the current user-language hierarchy, and such a higher level of abstraction is closer to the user's perception of reality (facts); and, thus, more user-friendly. Therefore, the concept of "hierarchy of user languages" supports the expectation of developing a more user-friendly language interfaces which allow users to formulate their queries in a way which is closer to their own conceptualizations rather than the machine's.

Second, the development of a higher level user language interface can reduce the burden on the end users. In view of the above discussion, the development of a high-level user language hides the lower-level, more machine-oriented languages from the user. Also, such development produces a system which has knowledge that is closer to its user's conceptualization and the capability of understanding the terminologies and "grammar" used by the average user. Therefore. while interacting with such a system, the user does not need to know how information is stored in the database or how to access the information he wants. He only

needs to know what information he wants and know how to express his intent in the high-level language, which is similar to the language he uses in terms of its terminology and meaning. [Feigenbaum 74]

Third, the development of a higher-level user interface also implies the reduction of syntactic restrictions on the use of the language inference. Therefore, such a development can ease the user's effort on memorizing syntactic restrictions of specific user languages.

Because of the above advantages, higher-level user interface development is one of the major concerns within the discipline of Computer Science. Based on the discussion in this section, user interface development is certainly dynamically related to the other components of an information system. Thus, it is important to recognize the dynamic relationships of both major approaches toward user interfaces development, namely, the formal query approach and the natural language approach, with the other components of the IS&R system, such as the nature of information retrieval and databases. In the next two sections, these two major trends of casual user/system interface development will be examined in terms of their dynamic relationships with the other components. It is hoped that, through such a discussion, the necessity and importance of NLQS development can be clarified and the required modifications within an IS&R system

resulting from such a development can be identified.

2.3 Formal Query Interfaces and Information Retrieval

The development of formal query interfaces is aimed at facilitating the task of the casual user in retrieving documents of interest from a bibliographic database. The rationale for such a development is to provide simple, direct, and easy-to-learn language facilities such that the user can be freed of any necessity for dealing with accession or reference numbers, scanning lengthy lists, manually consulting a thesaurus, or performing other repetitious, time consuming actions [Meister 67], as well as dealing with the data structures of the underlying database.

In line with this philosophy, many interactive IS&R systems have developed during last two decades, for example, DIALOG, RECON, STAIRS, MADAM, (1) and so on. These systems are designed to permit anyone to search a suitably prepared

---

(1) DIALOG was developed at Lockheed and was first put into service during 1966. The 1968 version of DIALOG was adapted by Lockheed to NASA's requirements. This versions was named RECON and is now in the public domain. RECON is installed not only at NASA, but also at the Department of Justice, the National Oceanographic and Atomospheric Administration, and other government agencies and academic institutions. STAIRS was developed by International Business Machines (IBM) and was first put into service during July 1972. MADAM was developed by the University of Southwestern Louisiana and was put into service during 1977. MADAM is a bibliographic information storage and retrieval system. This system is currently used as both a research vehicle and a production information system.

body of documents of specific interest by submitting simple search statements expressed as Boolean operations. These operations are applied to "pairs" that consist of a search term and an attribute descriptor that specifies the field in which the term is to be searched.

In this section, the task of bibliographic information retrieval by using formal queries will be examined; then, the organization of the database which ensures the effectiveness and efficiency of the task of information retrieval will be briefly described; and, at the end of this section, the user requirements for using formal queries in the process of information retrieval will be discussed.

2.3.1 Formal Query Database Searching

While interacting with an IS&R system, the use of formal queries allows the user easily to gain access to small numbers of relevant documents buried in immense databases. In response to the user query, instead of answering questions as generalized database management systems and question-answering systems do, the IS&R system provides a quick means for narrowing down the search space so that the odds are improved that the user will find an answer for his question [Meister 67].

In order to ensure the process of retrieving documents

with high degrees of recall and precision (1) , a user is required _to formulate his request 'intelligently'. That is, he needs to try out various search criteria and, by observing how those criteria pare away a database, to examine the distribution of retrieved citations. The user must then reformulate his search strategy until he feels that he can do no better. A typical interactive information retrieval sequence is best described by [Salton 83] and shown in Figure 2.3.

---

(1) According to the definitions provided by [Salton 83],

$$\text{recall} = \frac{\text{number of documents retrieved and relevant}}{\text{total number of relevant documents in the collection}}$$

$$\text{precision} = \frac{\text{number of documents retrieved and relevant}}{\text{total number of documents retrieved}}$$

```
                 +-------------------------+
                 |    Sign-On Operation    |
                 +-------------------------+
                              |
                              V
                 +-------------------------+
                 |   Enter Search Terms    |
+--------------->| ( Number of Retrieved   |
|                |   Items )               |
|                +-------------------------+
|                              |
|                              V
|                +-------------------------+
|<---------------| Display Number of Hits  |
| No Hits        +-------------------------+
|                              |
|                              V
|                +-------------------------+
|                |  Specify Reformulated   |
|                |  Query / Specify        |
|<---------------|  Format for Display     |<---------------+
| New Query      |  of Retrieved Items     |                |
|                +-------------------------+                |
|                              |                            |
|                              V                            |
|                +-------------------------+                |
|                |    Specify Number       |                |
|                |    for Display          |                |
|                +-------------------------+                |
|                              |                            |
|                              V                            |
|                +-------------------------+                |
|<---------------|    Display Retrieved    |                |
|  +------------>|    Information          |                |
|  |             +-------------------------+                |
|  |                            |                           |
|  |More                        V                           |
|  |Display      +-------------------------+                |
|  |             |  Specify Reformulated   |                |
|  +-------------|  Query After Viewing    |----------------+
|                |  Retrieved Items or     |  Reformulated
|<---------------|  Request More Display,  |  Query
| New Query      |  or Stop                |
                 +-------------------------+
                              |
                              V

                             STOP
```

Figure 2.3 Interactive Information Retrieval Sequence

This generic sequence is common to all IS&R systems with slight differences, although IS&R systems may be distinct from each other in terms of storage schemes or databases, their formal query formats, or their implementations. For a detailed feature analysis of IS&R systems, see Meister [Meister 73], Martin [Martin 74] and Salton [Salton 83].

Since the main concern of this subsection is to examine the task of information retrieval by using formal queries in general, the discussion is focused on the common features provided by various IS&R systems for information retrieval rather than on their differences.

(1) Sign-On Operation

The purpose of a sign-on operation is to invoke a specific IS&R system and enter one of its bibliographic database environments, assuming a multi-database system is being accessed.

To perform this operation, a user normally inputs the name of the specific IS&R system, and/or his name and password or account number. Figure 2.4 is extracted from the MADAM Users Manual [Dominick 82] and sufficiently describes the sign-on operation.

```
+-------------------------------------------------------------+
|                                                             |
|  (user) MADAM                                               |
|                                                             |
|                                          MADAM - USL        |
|                                                             |
|                                                             |
| Please supply the following personal data                   |
|  for system evaluation purposes :                           |
|                                                             |
| Please input your name?   (user) DavisCM                    |
|                                                             |
| Please input department name?   (user) DBMS                 |
|                                                             |
| Do you want to see MADAM bulletin data (09/17/80)?          |
| (user) no                                                   |
|                                                             |
| Do you wish operating instructions?                         |
| (user) no                                                   |
|                                                             |
| Please enter desired data base name or a system command?    |
| (user) DBMS                                                 |
|                                                             |
|                                                             |
| DATA BASE NAME: DBMS                CREATION DATE: 01/22/80 |
|                                                             |
| NO. OF RECORDS: 627                 LATEST UPDATE: 09/13/80 |
|                                                             |
|                                                             |
| Do you want to see DBMS bulletin dated (09/17/80)?          |
| (user) no                                                   |
|                                                             |
| Please enter next term or system command?                   |
| 1/                                                          |
|                                                             |
+-------------------------------------------------------------+
```

FIgure 2.4 Example of Sign-On Operation

(2) Query Formulation

After the user has selected the database, most systems expect him to enter a search request. Such an expectation is expressed by a system statement such as "ENTER SEARCH COMMAND OR TYPE HALT" in RIQS (1) or "Please enter next term or system command" in MADAM. After viewing this message, the user is informed that he needs to formulate his search statement according to the features and syntax provided by that specific IS&R system. In general, a query contains two major parts, namely, the search command and search term. A user must indicate his intent by entering a search command such as "select", "search" or "find". The search term to be entered indicates the information items to be searched. In an IS&R system, a search term is a word or a phrase which can be used to pinpoint the topic of a specific area. Example 5 illustrates two queries that are formulated by using the search command and search terms.

---

(1) RIQS was developed by Northwestern University and was first put into service in September 1969. It is intended for maintenance and searching of small-to-medium size databases whether they are bibliographic, textual, or numerical. It can also be used in conjunction with graphics plotting and statistical analysis.

```
Q1:  SELECT  HEART
     SELECT  DISEASE
     COMBINE 1 AND 2

Q2:  SEARCH HEART AND DISEASE
```

Example 5. Search Command and Search Term

In both cases, the system performs the following sequence of operations:

(a) Uses the system dictionary, such as the inverted files, to retrieve the document reference numbers associated with the term "HEART"; call these reference numbers Set 1.

(b) Uses the system dictionary to retrieve the document reference numbers associated with the term "DISEASE"; call these reference numbers Set 2.

(c) Determines which document reference numbers constitute the intersection of Set 1 and Set 2; call these document reference numbers Set 3.

(d) Uses the main document file to retrieve the documents identified by the document reference numbers in Set 3.

In addition to the basic format presented above, most systems also allow the user to enter his query by applying various operators such that more specific search procedures

may be performed.

Relational Operators

The operators such as "greater than", "less than", "between" and "equal", etc. can be used to connect a field name and its value(s) in order to formulate search criteria. The format of queries using such operators is shown as:

<field name> <relational operator> <field value>

The following example shows two queries that apply relational operators to formulate search requests.

Q1: SELECT PY = 1980 : PY = 1984

This is a DIALOG query indicating that documents with a publication year (PY) between 1980 and 1984 are to be retrieved. The colon ':' designates a range of measurable values to be used (i.e., numeric values).

Q2: select title eq "DATA BASE";

This is a MADAM query. In this query, the relational operator "eq" is applied to connect the field name "title" and its value "DATA BASE". In response to this query, the system accepts all the documents with the title containing the phrase

"DATA BASE" as its results.

Example 6. Queries with Relational Operators

Adjacency Operators

Some systems allow the user to specify what terms are to occur in the same field without specifying the exact field. For example the DIALOG system allows the user to specify how many word(s) may separate two terms. The format of such queries is:

<search term1> <#word operator> <search term2>

In Example 7, some simple queries used in DIALOG and STAIRS are presented. Although the operators used in these systems seem different, their usage is the same, namely, to allow the user to formulate queries using words included in the document texts.

Q1: INFORMATION ADJ RETRIEVAL

Q2: INFORMATION (5 W) RETRIEVAL

In Q1, the two search terms INFORMATION and RETRIEVAL must appear next to each other in the text and in a fixed order such that documents on "information retrieval" may be searched. In Q2, the two search terms must appear in the fixed order and

there is a space of up to five words between them.

Example 7. Queries with Adjacency Operators

Boolean Operators

The Boolean operators are most frequently used in formalizing queries. The typical Boolean operators are AND, OR, and NOT. These operations are implemented by using set intersection, set union, and set difference procedures, respectively. The typical format of such queries is shown as:

<search expression1> <Boolean operator> <search expression2>

```
select title eq "DATA BASE"
and (or, not) author eq "MARTIN";
```

In this query, if the Boolean operator AND is used, all the publications written by MARTIN with a title containing DATA BASE are searched; if OR is applied, then the publications either written by MARTIN or with a title containing DATA BASE are retrieved; if NOT is used, then the publications with a title containing DATA BASE but not written by MARTIN are searched.

Example 8. Queries with Boolean Operators

In addition to the above features, most IS&R systems also provide some features such as suffix removal, spelling variations, request sets, and related term capability.

Suffix Removal

Most IS&R systems require the user to indicate suffix removal by entering the root followed by a truncated symbol, such as ? and !. In response to such queries, any term that begins with the root and satisfies the specification of the truncation symbol are acceptable. For example, if the search term in the DIALOG system is "INFO?", all the items associated with

INFORM
INFORMATION
INFORMATIONAL
INFORMATIVE

will be retrieved since they begin with the character string INFO.

Spelling Variations

Some systems allow the user to indicate a search term by using spelling variations. For example, in the MADAM system, words like "January", "JANUARY", "january', "jan" and "Jan" are linked so that the use of any one incorporates the others.

Request Sets

This feature exists in most IS&R systems. Whenever the user inputs a query, the system gives such a request a set number such that later search requests can incorporate earlier sets by simply referencing the set number. Example 5-(Q1) exhibits the function of such a feature and the way the system handles such queries.

Related Term Capability

Some systems, such as DIALOG, RECON and ROBOT, examine their memory alphabetically for the search term and provide an alphabetical listing. In this list, the search term entered by the user is displayed in its correct alphabetical sequence and identified by an assigned index number if this term is one of the index term. Example 9 illustrates the feature of the related term.

```
(user)   S CIRCUIT DESIGN

(system) --SELECT ONE OR MORE BY NUMBER--

            50. CIRCLE
            51. CIRCUIT
            52. CIRCUIT BOARD
            53. CIRCUIT BREAKER
                ---CIRCUIT DESIGN
            54. CIRCUIT PROTECTION
             .        .
             :        :
```

Example 9. Queries with Related Term Capability

In Example 9, since the term CIRCUIT DESIGN is not a NASA/RECON index term, it has no number associated with it. That is, the term "CIRCUIT DESIGN" cannot be selected. Therefore, the user is encouraged to read the alphabetical listing and reformulate his query. Otherwise, a null result would be produced. If the user decides CIRCUIT is his best choice, he will enter "S 51". In response to this reformulated query, the number of documents which reference this term CIRCUIT will be displayed. The reason for the above result is that CIRCUIT is an index term.

Search Profile

Most systems provide a search profile feature so that a user may develop a search strategy, store this strategy, and rerun it in future search sessions. To save a search profile which contains the total requests in one session, the user may simply enter:

(user) ..SAVE <name> in STAIRS; or

(user) END/SAVE <assigned serial number> in DIALOG

If, in some other session, the user wants to rerun this search profile, he can invoke this profile by entering:

(user) ..EXEC <name> in STAIRS; or

(user) .RECALL <assigned serial number> in DIALOG

Display of Results

Almost all commercial and experimental IS&R systems provide a wide variety of formats for the user in displaying the search results. Those formats can be divided into two major categories, namely, system-defaulted formats and user-specified formats.

To obtain a system-default result display, the user may simply enter a display command. In response to this command, the system outputs the most recent result set. For example, to ask for a system-default display, the user may enter:

(user) DISPLAY (for the CRT) or TYPE (for the typewriter)

in RECON or DIALOG; or

(user) "PRINT" in ORBIT

In general, the system displays a limited number or a page of records each time, and then asks the user whether he wishes to continue. The user may answer this question by entering:

(user) PAGE in RECON and DIALOG; or

(user) YES in ORBIT (in answering "CONTINUE PRINTING?")

A user may also control the result display by specifying display formats, order or fields of document records. In the following, some examples of user-specified formats which allow the user to control the result display are shown:

(a) The user may specify that every field of the records is to be displayed by entering:

(user) DISPLAY set# /2 (or 5) in RECON (or DIALOG);

(user) "PRINT FULL" in ORBIT

(b) The user may specify that only certain fields are to be displayed by entering:

(user) "PRINT INCLUDE FN, FN EXCLUDE FN, FN" in ORBIT.

(c) The user may develop formats on-line for later incorporation into record displays by entering:

(user) FORMAT FN, FN;FN in RECON (1)

(d) The user may designate the order in which records are to

---

(1) In the above query statement, the symbol ',' indicates line continuation, and the symbol ';' indicates carriage return.

be sorted by entering:

(user) SORT set#/FN, A or D, field length/... in RECON

In addition to the above examples, many systems also provide some display features to allow a user to change display directions in midstream [Martin 74]. Also, almost all IS&R systems provide features which allow a user to obtain off-line output of his search results.

The query formulation features discussed above are common in most IS&R systems. Rather than comparing the features provided by various IS&R systems, the above discussion is aimed at exploring the capabilities of formal query facilities integrated into most IS&R systems. From this discussion, the query formulation features of most IS&R systems can be summarized as follows:

* Search Field Control
* Suffix Removal
* Relational Operators
* Boolean Operators
* Adjacency Operators
* Spelling Variations
* Related Term Capability
* Search Profile

* Request Sets
* Display of Results

2.3.2 Information Content

Salton [Salton 83] described that any IS&R system consists of a set of information items (DOCS), a set of requests (REQS), and mechanisms for determining which, if any, of the information items meets the requirements of the requests (SIMILAR). Based on this description, while performing information retrieval in a bibliographic database environment, two functions are performed by the system, namely, EXIST and COUNT.

EXIST: After the system receives the user's query, it examines the database to determine whether any information item matches the search criteria.

COUNT: If there is at least one item that meets the search criteria, the system counts the number of items and informs the user such that the user may perform further queries which will broaden (or narrow) the search space, or ask for a display of the content of those items.

To perform the above functions, the primary operation

performed by the system is the "fetch" operation. This operation_ involves searching for and retrieving such information items that satisfy the user's information needs.

As Tanenbaum [Tanenbaum 76] discussed, the fetch operation is a relatively slow operation. Therefore, to optimize system performance, the main concern of IS&R system design is to speed up fetch operations involved in information retrieval, or to ensure quick access to information items.

As described in Section 2.1, the documents stored in the bibliographic database are represented in their natural language textual forms. In order to ensure the efficiency of operations such as database searching, those information item representations must be collected together and well organized. There are many mechanisms for database structuring, such as linear lists, ordered sequence file, and indexed files [Salton 83]. Among them, the most commonly applied structuring mechanism is the inverted file structure [Salton 83; Heap 78]. The main reason is that,

> "an inverted file ensures quick access to the information items because the index alone is examined in order to determine the items which satisfy the search request, rather than the actual file of items. Furthermore, the index is sequentially ordered by the key values. ... One need not examine the individual records to determine their actual key values because that information is already contained in the (inverted) index." [Salton 83]

The other reason for using inverted file structures is that, since the inverted file stores only the indexes which contain the index terms associated with lists of document reference numbers, if the size of the file is not too large, it can reside in main memory, and the system may use the inverted index to retrieve the document reference numbers associated with the search term and find the desired information items very quickly.

Another concern of IS&R system design is to classify the possible search terms in order to ensure the effectiveness of information retrieval. The classification of individual terms is done by the construction of the thesaurus. A thesaurus provides a grouping of the terms used in a given topic area into thesaurus classes, and each thesaurus class is identified by a "concept number" or a "class identifier". A term listed in a thesaurus is chosen from the vocabulary of a database. For each listed term, some relations may be included to indicate other terms that have identical meaning (called "synonyms"), or terms that are narrower or broader in meaning (called "narrower term" and "broader term", respectively).

Since the thesaurus relates the vocabularies in the database, when a document in the database contains the term "superconductivity" while the query term is "cryogenic", a

term match would result in the thesaurus transformation and an appropriate term would be used in searching. Thus, the use of a thesaurus can support the system to handle such search terms as spelling variations, abbreviations, and to expand the indexing vocabulary in various directions.

As discussed in the beginning of Chapter 2, in order to perform search operations and provide desired information items to the user, an IS&R system has to have sufficient knowledge to process the query. The structuring of the thesaurus and inverted files, both defined by the bibliographic database, provides such knowledge. In Figure 2.5, the use of domain-specified knowledge during the process of information retrieval is presented.

```
+-------------+
|    USER     |
| INTEREST    |
+-------------+
       |
       |
       V
+-------------+
|    QUERY    |
| FORMULATION |
+-------------+
       |
       |
       V
+-------------+                    +-----------+
|    QUERY    |<===external===|            |
| EXPANSION   |     indexing       | THESAURUS |
+-------------+                    +-----------+
       |
       |
       V
+-------------+                    +-----------+
|   SEARCH    |<==============|    DATA    |
|             |       A            |    BASE    |
+-------------+       :            +-----------+
       |           internal
       |           indexing         +-----------+
       V              :.........|  INVERTED |
+-------------+                    |    FILE   |
|   OUTPUT    |                    +-----------+
|    ITEMS    |
+-------------+
       |
       |
       V
+-------------+
|    USER     |
| EXAMINATION |
+-------------+
```

Figure 2.5 Stages of Information Retrieval

In line with the concepts of the formal query approach to casual user/system interface development, since the knowledge of the system is limited by the content of the thesaurus and inverted files, the translation of the user interest into a formal query or a sequence of queries becomes essential to the success of the task of information retrieval. That is, while formulating his query, the user has to follow the formal syntax listed in the user manual and submit allowable search terms which exist in the thesaurus and inverted files. Otherwise, an error message, a null result, or even undesired items would be generated by the system and the search fails.

Such user requirements, as discussed previously, can not cope with the nature of casual users. Therefore, to allow the casual user to interact with an IS&R system directly, it is necessary to develop a natural language interface integrated with the IS&R system such that a casual user can perform information retrieval without being limited by his knowledge about the formal syntax and search terms defined by the IS&R system as long as the user knows what information he wants and whether the desired information might exist in the systems knowledge domain (i.e., its database). (1)

---

(1) In the IS&R system environment, the information retrieval functions performed by the system aid the user to narrow or

2.4 Natural Language Interfaces and Information Retrieval

As stated in Section 2.1 and Section 2.3, the formal query approach toward casual user interface development has some disadvantages, such as the requirement of two-step translation (see Section 2.1), or the use of Boolean and/or relational operators. To use a specific formal query, the casual user has to have the knowledge of the logical content of data, such as the terms in the thesaurus and/or the inverted file, and the formal syntax supported by the system. Such requirements have been proven to be the major obstacle of the casual user/system interaction. The intent of natural language interface development is to resolve this obstacle by allowing the user to form his query based on his own conceptualization.

Based on the concept of "the hierarchy of user languages", such an objective can be met by adding a new interface on the existing hierarchy. This new interface can exhibit following advantages:

(1) The user can interact with an IS&R system by specifying what he wants based on his own perception of the

---

broaden the search space. IS&R systems are not developed to answer questions [Salton 83], [Minker 77]. Therefore, prior to interacting with an IS&R system, a user should have knowledge about what he wants and whether the specific IS&R system can help him to perform information retrieval. Thus, he must have knowledge about the domain knowledge of the specific IS&R system and determine whether he might obtain the desired information from that system.

information and his ability to deal with the natural language syntax.

(2) The natural language interface can transform the description of what the user wants into the one of its predecessor language levels. After such a transformation process, the new description can be recognized and used by the computer system in the process of information retrieval.

The ability of the natural language interface to transform the user's concepts and requests into internal, operational representations is the key technical problem of NLQS design. In order to solve this problem, it is important to recognize what impact such a development might bring into the existing IS&R system which applies conventional formal query interfaces. As discussed in Section 2.2, in any man/computer communication process, there are four interdependent components, namely, tasks, users, user interfaces, and databases. The development of a natural language interface, which can facilitate casual user/system interaction, should affect the dynamic relationships of those components, and bring some changes to the nature of bibliographic database searching and database design.

2.4.1 Natural Language Database Searching

The development of a natural language interface causes a major change in the nature of the bibliographic information retrieval task. Conventionally, to perform bibliographic information retrieval, the user inputs a syntax-restricted formal query, as discussed in Section 2.3. In response to this query, the system performs the task of fact retrieval and simply looks through its dictionary and database to fetch the desired data identified by the "search terms or phrases" in the query [Dominick 82]. In such a case, the user must translate his request into the formal query which requires not only the domain-specified knowledge embedded in its thesaurus and inverted file, but also the knowledge about the syntactic restrictions of that specific formal query language. On the other hand, the system only needs the domain-specified knowledge and to invoke appropriate search procedures to perform a sequence of search operations. Thus, the user is responsible for the success of the information retrieval process.

On the other hand, when the input query is in the form of unrestricted natural language, the user only needs to know what he wants and whether his desired information may exist in the database. That is, he only needs to have the domain-specified knowledge. On the other hand, the system should have the capability to process natural language input

and extract the user's concepts. Hence, not only does the system require domain-specified knowledge to locate the desired answer for the request, but also it has to have a certain amount of linguistic knowledge to extract the user's intent from natural language input, which may be ambiguous, vague and unpredictable.

Therefore, the introduction of a natural language interface into the IS&R system will change the nature of bibliographic information retrieval from a simple fact retrieval process to a more difficult deductive inference process. To cope with this change, the system must develop its ability to handle the linguistics of the natural language query. This implies the need to incorporate some modifications into the bibliographic database design.

2.4.2 Demands on the Database

The conventional database design illustrated in Figure 2.2 (see Section 2.2.2), following the concept of "levels of abstraction", establishes a user's logical level which allows the user to see the desired answer in a view closer to his perception. But as discussed before, at this level, the casual user will require some knowledge of the logical structures of data, Boolean operations and other relational operators that may not be familiar to him.

The natural language interface development causes a higher level of abstraction. It builds a level, namely, the user's view level, at which the individual user may not only see the desired data in a view that corresponds to his own concept, but also request information based on his own concept of and ability to use the natural language syntax (see Figure 2.6). The requirements of constructing this "user's view level", as discussed previously, is affected by the nature of the task and the natural language interface.

```
Fact                    REALITY

                          |
                          |
                          |
User's                USER'S VIEW        Natural Language
Perception               LEVEL           (NL)

                          |
                          |
-------------------------+-------------------------------
                          |
Information          USER'S LOGICAL      Information
Structure                LEVEL           Structure Description
                                         Language (ISDL)

                          |
              (logical data independence)
-------------------------+-------------------------------
                          |
Data                 SYSTEM'S LOGICAL    Data Structure
Structure                LEVEL           Description Language
                                         (DDL)

                          |
              (physical data independence)
-------------------------+-------------------------------
                          |
Storage              PHYSICAL STORAGE    Storage Structure
Structure                LEVEL           Description Language
                                         (SDL)
```

Figure 2.6 Levels of Data Base Design (Revised)

Since the natural language interface needs the ability to transform the natural language input into recognizable formal query counterparts used by the system in the process of information retrieval, the system should have some facilities to support a natural language transformation process.

Also, since the nature of the task involves the retrieval method of deductive inference as discussed in Section 2.2, in addition to the domain-specified knowledge defined by the bibliographic database, the system also requires linguistic knowledge in the form of certain built-in rules and frames in order to support the natural language interface to perform both syntactic and semantic analysis on natural language input, and to formulate a formal query or a series of formal queries. To store this linguistic knowledge, it is necessary to design a knowledge database or dictionary (in the following context, KB will be used to refer knowledge base or dictionary). The KB provides certain semantic and syntactic information about the database such that the system can have adequate <u>intelligence</u> to extract the intent of the user's request and translate it into an internal, formal query language. Therefore, the design of the KB is one of the major phases in NLQS development.

In summary, the development of a natural language interface_for IS&R systems can produce major changes in both the nature of bibliographic information retrieval and the demands on database design. To cope with these changes, the system should develop a KB which includes sufficient representations of the linguistic knowledge such that it, along with the domain-specified knowledge, can provide the system with reasonable intelligence to transform natural language input into its formal counterpart lying within the class of formal queries supported as internal interfaces.

CHAPTER 3

NATURAL LANGUAGE QUERY SYSTEMS DEVELOPMENT

The central issue of NLQS development is natural language processing. Natural language processing in the context of bibliographic information retrieval can be defined as the process of translating natural language requests into their formal counterparts. Natural language processing also includes generation of natural language responses to the user's queries. To perform those activities, an NLQS has to make intelligent use of the information within the KB and the bibliographic database. Two important problems relevant to the intelligent use of the system's knowledge are raised:

(1) What types of knowledge the system should have within its KB, and how the knowledge should be structured and integrated into the bibliographic database such that it can provide sufficient intelligence to the system during the process of information retrieval.

(2) How to use the system's knowledge intelligently during the process of natural language translation and response generation. This implies that the system should use appropriate type(s) of knowledge in an appropriate time frame during natural language processing in order to

perform specific functions in response to the user's natural language request.

By examining these two problems, several issues relevant to NLQS development can be identified as follows:

(1) What language capabilities are expected to be needed by the system during the process of natural language translation and response generation?

(2) What operations should an NLQS perform in response to the user's natural language request? Can those operations be grouped into several steps, and what knowledge is required in each step of natural language processing?

(3) What problems are encountered in each step of natural language processing, and how does the system apply its knowledge to solve them?

Within this chapter, these issues will be discussed so that the types of knowledge required in an NLQS and the functions of various types of knowledge can be identified. After reviewing the literature relevant to NLQS development, in Section 3.1, a reasonable amount of natural language capabilities required to be integrated into IS&R systems will be identified and described; in Section 3.2, the phases of natural language processing, including syntactic analysis,

semantic analysis, execution, and response generation, will be identified. The functions and the required knowledge of each phase will also be examined. Finally, the problems encountered at each phase of natural language processing will be identified so that those problems can be relsoved in NLQS development.

3.1 Language Capabilities of Natural Language Query Systems

The major purpose of NLQS development, as discussed in Section 1.1 and Section 2.1, is to remove the burden on the casual user by automating the process of translating his natural language request into the internal, formal representation recognized and used by the system for information retrieval. To automate this translation process, the system should be designed so that it has the capability of handling various natural language inputs which are concerned with specific topic areas or databases. Hendrix [Hendrix 78; 81] examined recent NLQS development and tried to summarize a list of thirteen capabilities exhibited by those systems. However, by examining existing NLQS, rarely does a system exhibit all the capabilities shown in Hendrix's checklist (see Appendix A). Also, Hendrix's checklist is not able to include sufficiently all the capabilities that allow the system to understand various natural language inputs submitted by casual users. These include the capability of

"transportability" proposed by Kaplan [Kaplan 78; 83; 84], that of detecting "typographic errors" suggested by Codd [Codd 74], and the capability of "impertinent responses" proposed by Siklossy [Siklossy 78]. The difficulty of fully identifying all the language capabilities at the initial stage of NLQS development can be attributed to two major reasons:

(1) As many computer scientists suggested, natural language is too complex, and even a simple request may be presented in many different ways [Harris 76; Smith 80; Martin 82; Rowe 82]. In addition, as Kaplan [Kaplan 78] stated, natural language questions require a wider range of potential responses than formal queries do, and they contain cues for selecting among those responses that are generally absent from formal languages. Therefore, to develop a system which can automate the process of natural language understanding is very complex and difficult.

(2) Human errors may happen in the process of natural language man/computer interaction, and the types of those errors are usually unpredictable. Thus, in the initial stage of NLQS design, it is very difficult to identify all possible errors that might be introduced by users [Codd 74; Harris 76].

The above difficulties involved in NLQS design are similar to the problems encountered in software testing and system design/analysis. Thus, by referencing the underlying concepts of system analysis [McClure 81] and software testing techniques [Beizer 83], it is suggested that the intent of finding all human errors and types of natural language queries is impractical, and often impossible. Therefore, a more productive and flexible approach to identifying language capabilities of an NLQS is to conduct a survey and find out the human errors and types of natural language queries which have higher frequencies of occurrence during the man/system interaction process.

Based on the above rationale, most NLQS's are developed with the capabilities of handling limited types of natural language queries, detecting and/or correcting certain types of human errors. By reviewing recent NLQS development, some capabilities that are exhibited in most natural language query systems can be identified as follows:

(1) Answer direct questions:

To answer direct questions, such as "How many articles are there with a title containing 'information retrieval'?" is the fundamental capability of any NLQS. To answer the above question, the system must find all articles with a title containing "information retrieval"

as a search term; and, then, count the number of retrieved articles as its answer to the question.

(2) Handle simple use of pronouns and ellipsis:

Pronoun reference is commonly used in human conversation. In order to simulate human communication behavior, an NLQS has to be able to handle the use of pronouns such as "he", "me" or "it". To handle pronouns, an NLQS needs to remember a history of interaction so that whenever a pronoun reference is encountered, it can review the history and find an appropriate object to replace the pronoun used in the current query.

Like pronoun reference, "ellipsis" is also very common in human communication and needs to be handled by an NLQS. In general, handling the ellipsis is similar to handling a pronoun. That is, the history of interaction must be recorded for future review. Rendezvous II developed by Codd [Codd 78] is an NLQS which aims at handling pronoun reference and ellipsis, including questions such as "How about recently?"

(3) Analyze NULL answers:

Sometimes a user might misperceive the information content of a database. In such a case, although his request may be correct syntactically and semantically,

the system cannot find any answer to this request, i.e., a null answer case. Recent NLQS development recognizes such problems of misconception, and suggests the development of a coorperative NLQS. A coorperative NLQS can detect the reasons for the null answer, and provide the user with an indirect and more informative answer which can help the user in correcting the misperception of the database and rewriting his query.

(4) Restate in English the system's interpretation of inputs:

The system's restatement serves many functions [Codd 74]. For example, it may force the user to consider his intent, and to examine carefully whether the system's interpretation of his request has captured his intent and, thus, to ensure that the result of information retrieval will actually satisfy his intent.

(5) Correct typing or spelling errors:

This capability is also a fundamental requirement of an NLQS. Since typographic and misspelling errors are very common, the system cannot simply reject a user's request because of an unknown word due to a typing or misspelling mistake. An intelligent NLQS should have the ability to detect such an error and find a correct substitute for

the erroneous word.

In addition to the above capabilities relevant to natural language translation and generation, an NLQS should also provide the capabilities exhibited in a conventional formal query system, such as the capability of suffix removal, search expansion, and so on (see Section 2.3.1 for a detailed description). In other words, although the development of an NLQS may complicate the task of language processing, and provide certain additional capabilities relevant to natural language translation, it should also maintain the particular features supported by the formal language interface of an IS&R system.

To exhibit the above language capabilities, an NLQS needs to parse and interpret the natural language query, perform database searching, and generate an appropriate natural language response to the specific query. This process involves several phases of language processing activities. In the next section, the phases of natural language processing adopted by most NLQS development efforts will be discussed.

3.2 Phases of Natural Language Processing

Natural language processing can be divided into four phases referred to as syntactic analysis, semantic analysis, execution, and response generation. In this section, these four phases will be discussed in terms of their functions, knowledge requirements, alternative design approaches, and the output generated by each of these phases.

3.2.1 Syntactic Analysis

In syntactic analysis, linear sequences of words in the input sentences (i.e., a subset of natural language) are transformed into structures that show how the words relate to each other. Some sequences may be rejected by the system at this stage if they violate the language's rules for how words may be syntactically combined [Rich 83; Mylopoulos 76]. For example, the syntactic analyzer will reject the sentence "Give me books the written by Date."

Syntactic analysis can be further divided into two steps, namely, lexical analysis and parsing.

(1) Lexical Analysis

Lexical analysis is responsible for "cleaning up the input" [Waltz 77; Rich 83]. The lexical analyzer reads the

input sentence one word at a time and performs word recognition and word transformation based on knowledge stored in the lexicon [Codd 74,78; Kaplan 78, 84; Salton 83; Aho 79].

[Montgomery 72] suggested that the lexical analyzer can be used to:

(a) Aid in the analysis of the particular subset of a language which constitutes the universe of discourse for the given information system;

(b) Compress voluminous word lists or dictionaries by keeping only the word stems (e.g., write, but not wrote, written or writing);

(c) Automate the expansion of terms in a query or search description to the full paradigm;

(d) Identify grammatical categories in text processing systems employing some form of syntactic analysis (which will be presented in the next portion of this subsection). Words missing from a dictionary may be assigned a syntactic category based on the lexical analysis so that the processing may continue.

The lexical analyzer performs the above operations by referencing a dictionary called a "lexicon". The lexicon resides in the KB and contains entries which reflect

syntactic and/or semantic properties of words. A number of tools have been developed for use in developing the lexicon that describes the syntactic properties of words. For instance, Dolby [Dolby 67] has developed a five-volume compendium, The English Word Speculum, that includes forward and reverse word lists. Other lexicon developments mainly reflect the semantic properties of words. In this type of lexicon, each entry contains two fields, namely, lexicon classes and lexicon values. For instance, Codd [Codd 74] developed a lexicon which contains up to fifty lexical classes for Rendezvous. Each class is identified by its semantic function such as relation, boolean, noise word, location, etc. Kaplan [Kaplan 84] divided the lexicon into three major types of classes, namely, general entries, structural entries, and volatile entries. General entries are those words whose meanings are independent of any particular domain. The verbs such as BE and HAVE, the prepositions such as IN and FROM, and the relative pronouns such as WHICH and THAT are words of this class. Structural entries are terms which make reference to aspects of the database structure, such as AUTHOR, DATE and other field names. Volatile entries are those which refer to specific values in the database or the inverted file, such as author names, keywords, etc. Kaplan [Kaplan 84] explained that the difference between these three lexicon classes is their stability. By applying this method, he also claimed that

lexicon development can support the transportability of knowledge_used by different information systems.

(2) Parsing

Parsing is the "delinearization" of the linguistic input by using grammatical rules and other sources of knowledge. The parser seeks to map a string of input words onto a set of meaningful syntactic patterns which are usually in derivation tree structures [Barr 81].

The design of a parser involves two major steps: (a) to determine what grammar is to be used ; and (b) to determine how to use the grammar to match a word string against patterns of the grammar (i.e., grammar rules). Before describing the approaches to the development of a parser, it is necessary to identify the general considerations of parser design. Barr [Barr 81] specifies four issues that must be considered in parsing:

(a) Uniformity vs. Efficiency

While selecting a scheme for a parser to represent its knowledge about word meanings, grammar, and so on, the tradeoff between efficiency and uniformity must be considered. A parser which has a uniform set of rules and a consistent algorithm for applying rules can usually write and modify the language understanding system with

great simplicity but with less efficiency; in contrast, if rules and processes are based on specialized knowledge of what the input to the parser will contain, then it is possible to do things more quickly and efficiently.

(b) Multiple Sources of Knowledge

Although a parser is developed mainly for syntactic analysis, the design of parsers should also consider the needs of other levels of natural language processing, such as word recognition and use of word meanings. Therefore, Barr [Barr 81] suggested that the method applied in parser design should be aimed at producing intermixed structures rather than purely syntactic structures.

(c) Precision vs. Flexibility

The development of a precise system is highly desired. But since natural language sentences supplied by humans are sometimes meaningful but syntactically wrong, the design of parsers needs to be flexible in order to handle those natural language sentences and to extract the intent of the users. However, a flexible parser usually loses many advantages of the more complete analysis possible with a precise system. Therefore, while designing a parser, it is important to obtain a balance between flexibility and precision.

(d) Types of Structure Returned

The form of the structure assigned to an input sentence during syntactic analysis can be a surface structure or a deep structure of the input sentence. While designing a parser, one should decide which form of structure is expected to be assigned by the parser to the input sentence.

Based on the above considerations, four major approaches to the development of parsers are identified. They are template matching parsers, phrase structure grammar parsers, transformational grammar parsers, and semantic grammar parsers. Each of these approaches requires a different amount of knowledge and applies different grammar rules; therefore, their capability and performance also have significant differences [Barr 81; Rich 83; Jackson 76].

Template Matching Parser

The template matching parser was used in most of the early natural language systems, such as ELIZA [Weizenbaum 66]. This parser performs parsing by matching input words against a finite set of predefined templates. Templates are a set of standard forms of all possible input sentences. To translate a sentence, the template matching parser matches words of the query against words of the template sentence

until there is a complete match between the query and the template sentence. The grammatical rules associated with one template do not apply to other templates. The major advantage of this type of parser is that it can recognize sentences whose grammar is unusual or even sentences that are grammatically incorrect as long as the designer of the parser anticipates such unusual sentences and incorporates appropriate templates. In other words, this parser performs with a great degree of precision. But as [Barr 81] noted, the disadvantage of this type of parser is "knowledge poor". The system that analyzes natural language sentences by using this parser does not really understand input sentences in the sense of mapping them into structures that represent their meanings. Instead, they are mapped directly into an appropriate response and then they are forgotten. Because of the above disadvantage, the template matching parser sometimes provides misleading results to the users. Thus, they are rarely used in recent NLQS development.

Phrase Structure Grammar Parser

The phrase structure grammar parser uses the context-free grammar to perform parsing. The context-free grammar was developed by Chomsky [Chomsky 59]. The context-free grammar is the grammar in which each rewrite rule (or production) must be a single non-terminal symbol on the left-hand side of the production, and either a single terminal symbol or two non-terminal symbols on the right-hand side. Example 10 is adopted from Barr [Barr 81] and illustrates a context-free grammar used in natural language processing.

```
   <SENTENCE> --> <NOUN PHRASE> <VERB PHRASE>
<NOUN PHRASE> --> <DETERMINER> <NOUN>
<NOUN PHRASE> --> <NOUN>
<VERB PHRASE> --> <VERB> <NOUN PHRASE>
 <DETERMINER> --> the
       <NOUN> --> boys
       <NOUN> --> apples
       <VERB> --> eat
```

The lower-case letters are the terminal symbols, and the upper-case letters in brackets are non-terminal symbols. The derivation tree generated by the above grammar rules is present as below:

```
                          <SENTENCE>
                              |
                +-----------+--------------+
                |                          |
                v                          v
          <NOUN PHRASE>              <VERB PHRASE>
                |                          |
      +---------+-------+         +--------+----------+
      |                 |         |                   |
      v                 v         v                   v
<DETERMINER>          <NOUN>    <VERB>          <NOUN PHRASE>
      |                 |         |                   |
      |                 |         |                   |
      v                 v         v                   v
     the               boys      eat                <NOUN>
                                                       |
                                                       |
                                                       v
                                                     apples
```

Example 10. An Example of the Use of a Context-Free Grammar

The advantage of phrase structure grammar parsers is that the structures derived correspond directly to the grammar rules; thus, the subsequent semantic analysis is simplified [Minker 77]. The major problem of the phrase structure grammar is that English is not a context-free language. By using this type of parser, certain common constructions in everyday English cannot be generated [Barr 81]. Therefore, this type of parsers is most often used in the development of a restricted NLQS which has syntax rules for query generation, rather than that of an unrestricted NLQS as discussed in this research.

Transformational Grammar Parser

This parser uses transformational grammar to perform parsing and produces the syntactic components of a sentence. According to Chomsky's Syntactic Structures [Chomsky 59], the parser first applies a phrase structure grammar to generate a set of terminal strings (morphemes), each with an associated description called a generalized phrase marker. A phrase marker is a labelled, rooted, directed tree of a sentence. The phrase marker is termed generalized as the terminals of the tree are not strings of a language, but word classes and conditions on the word that must be satisfied for a word to be used in that position. For example, a verb which is animate and which can be followed by a noun phrase (NP) may be specified. After strings of morphemes and their associated generalized phrase markers are generated, a sequence of transformational rules are applied to rearrange the strings and add (or delete) morphemes to formulate representations of the full varieties of sentences. At this step, the parser accepts as input the generalized phrase marker of the morphemes, referred to as the deep structure, and outputs the derived phrase marker of a new structure, termed the surface structure, by successive application of transformational rules [Barr 81]. Finally, the parser accepts as input the surface structure and produces a phonetic representation of the sentence. For a detailed explanation, see Barr [Barr 81] and Rich [Rich 83].

There have been three approaches developed for recognizing word strings generated by a transformational grammar, namely, "analysis by synthesis" [Matthews 62], "reverse transformations" [MITRE 64] and [Petrick 65], and "augmented transition networks (ATNs)" [Woods 69]. Among them, the ATN grammar is suggested as the most successful approach and is used by many current language processors. The ATN is a finite state transition diagram which has been generalized to a push-down store automation by adding a recursion mechanism and a set of registers that can hold arbitrary pieces of tree structures and arbitrary conditions and actions that can set and test these registers on the arcs of the networks. The importance of the ATN grammar lies in the arbitrary register settings and actions on the arcs. The actions on the arcs permit transformations to take place, making it possible to model transformational grammars. A simple ATN is adopted from Woods [Woods 69] and shown in Figure 3.1.

C-2

```
            +-----+
      +--->| q1  |--v----|                          ..PP..
      | NP +-----+       V                          .    .
+---+ |       | aux  +-----+      +-----+          +-----+
| S |-+       +----->| q3  |- v ->| q4* |-- NP ->| q5* |
+---+ |              +-----+      +-----+          +-----+
      | aux +-----+   NP
      +---->| q2  |------------|
            +-----+


                        ..adj...              .. PP..
                        .      .              .     .
                        ..    ..              ..   ..
+-----+                  +-----+               +-----+
| NP  |---+-- det ----->| q6  |------ n ---->| q7* |
+-----+   |              +-----+               +-----+
          |
          |     NP      +-----+
          +----------->| q8* |
                        +-----+


+-----+                 +-----+                +-----+
| PP  |---- pre ------>| q9  |----- NP ----->| q10*|
+-----+                 +-----+                +-----+
```

S is the start state
q4, q5, q7, q8, q10 are the final states

<u>Notations</u>

qi : 1 <= i <= 10 represents states
qi* : represents the terminal states
S : sentence
NP : Noun Phrase
PP : Prepositional Phrase
aux : auxiliary
det : determiner
pre : preposition
n : noun
v : verb

Figure 3.1 Example of Transformational Grammar

The ATN grammar, since it was developed, has been successfully applied to question-answering in limited domains, especially for the development of natural language interfaces within database systems, such as TORUS [Mylopoulos 76], LUNAR [Woods 72], PLANES [Waltz 77] and CO-OP [Kaplan 78; 83; 84].

Semantic Grammar Parsers

These parsers use the modified phrase structure grammar by changing the conception of grammatical classes from the conventional <NOUN> or <VERB> to classes that are motivated by concepts in the domain being discussed. For example, a semantic grammar for a system that talks about books might have grammatical classes like <BOOK>, <PUBLISHER>, <TITLE>, <AUTHOR>, and so on. The grammar rules used by this parser would describe phrases and clauses in terms of their semantic categories rather than their syntactic categories. The systems which use this type of parser are LIFER [Hendrix 77] and SOPHIE [Burton 76].

3.2.2 Semantic Analysis

Semantic analysis accepts as input the output of the parser, and extracts the meaning of the sentence such that an

internal target representation can be generated and used by the system in the execution phase.

There are several approaches to the development of the semantics of a sentence. They are: syntax-directed approach to semantics, semantic view of information, and heterarchical view of semantics.

Most IS&R systems use the syntax-directed semantic analysis. In these systems, one may obtain phrases by two ways: statistical analysis or semantic analysis. In such systems, words within a certain distance of one another, both of which appear in a phrase dictionary (e.g., thesaurus), may be considered as a phrase in the statistical analysis. If the syntactic analysis of the sentence places the words in the same linguistic structure (e.g., noun phrase), then, in the semantic analysis, the words are considered as a phrase rather than as two individual words. For example, words such as "information retrieval" may be considered as a phrase if the syntactic analysis of the sentence places these two words in the same linguistic structure (e.g., noun phrase); then, in the semantic analysis, these words are considered a phrase rather than two individual words.

The major issue involved in the syntax-directed semantic analysis is string manipulation [Minker 77]. That is, a natural language sentence must be placed into another form,

such as weighted vectors in some IS&R systems or first-order predicate_calculus statements. This form is called "internal target representation" of the input sentence and will be discussed later in this section. To perform such a transformation, the given string of input must be parsed according to the context-free grammar or transformational grammar, and the output string is specified as some function of the parse tree. The translation method applied in the syntax-directed semantic analysis is to associate a grammar rule with each production for permuting the order of the non-terminals on the right-hand side of the production and for introducing output symbols. Given a parse tree with a specific production used at the same node of the tree, it is altered at that node by:

(1) deleting descendants with terminal nodes;

(2) reordering the non-terminal descendants according to the fixed rule; and

(3) introducing descendants labelled by output nodes.

In addition to the above scheme, Woods [Woods 67; 68] described a similar approach. He assumed that the meaning of a sentence is contained in the deep structure of a sentence, and suggested that once the deep structure and phrase marker is found, the semantic analysis of a syntactic construction can be built up from the semantic interpretations of its

semantic rules of the form "pattern --> action". That is, a syntactic_construction generated by matching the sentence against the grammatical patterns leads to a semantic action which specifies an operation to be performed by the system, such as database searching. An example of the "pattern --> action" rule is adopted from [Woods 68] and shown in Figure 3.2.

<u>Sentence</u> "AA-57 flies from Boston to Chicago"

<u>Phrase Marker</u>

```
                 S
                 |
+---------------+---------------+
|                               |
NP                              VP
|                               |
|          +--------------------+------------------+
|          |                    |                  |
NPR        V                   PP                 PP
|          |                    |                  |
|          |              +-----+---+          +--+---+
|          |              |         |          |      |
"AA-57"  "fly"           PREP       NP        PREP    NP
                          |         |          |       |
                        "from"     NPR       "to"     NPR
                                    |                  |
                                "Boston"           "Chicago"
```

<u>Partial Tree Structures</u>

```
G1:       S            G2:       S            G3:       S
          |                      |                      |
   +-----+----+                 VP                     VP
   |          |                 |                      |
   NP         VP          +---+-----+                 PP
   |          |           |         |                  |
  NPR         V           V         NP          +-----+----+
   |          |           |         |           |          |
  (1)        (2)         (1)       (2)        PREP        NP
                                                |          |
                                               (1)        (2)
```

S rules:

1- (G1: FLIGHT ((1)) and (2) = fly) and
2- (G3: (1) = from and PLACE ((2)) and
3- (G3: (1) = to and PLACE ((2)))
==> CONNECT (1-1, 2-2, 3-2)

Figure 3.2 Example of "Pattern --> Action" Rules

The _second approach to the development of the semantics of a sentence is by using the semantic view of information. This approach claims that same conceptual structure underlies an utterance, and the natural language processor needs to fill in the elements of the conceptual structure in cooperation with a world model and an inference mechanism. To develop a program that is a model of human language understanding behavior, predictions must be made at each level in accord with what a human is known to make [Barr 81]. In this approach, rather than perform a complete syntactic analysis and then perform semantic analysis, the syntactic information is used as a pointer to incorporate conceptually with the world model (e.g., domain-specified knowledge).

Following this approach, if it is known that the need is for a certain type of conceptual information, a prediction can be made of the syntactic form and the place it will take. Under this approach, semantic primitives are the smallest units in the process of the semantic analysis, and the emphasis of conceptual representations is on the equivalence of meaning rather than syntax. This approach can be summarized and described as follows:

(1) Words are viewed as shorthand abbreviations for clusters of conceptual primitives, connected by conceptual links into networks;

(2) There are a very small number of primitives which underly all of the language; comprehension occurs in the realm of these primitives, not in the realm of words and phrases;

(3) The primitive concepts are sufficient to characterize any thought which is expressible in a natural language;

(4) Two sentences which have the same meaning, whether expressed in different languages or by two different sentences within one language are represented by the same conceptual graph.

Based on this approach, Schank [Schank 75] developed a system called MARGIE in which the processes of constructing conceptual structures, performing logical inference, and generating surface language are integrated in the stage of parsing.

The last approach to the development of the semantics of a sentence is the hetrearchical approach. This approach is best described by the work of Winograd [Winograd 72; 73]. It assumes that to understand language, one must include a model of the subject being discussed and a context for the discourse individuals, and depend on all sorts of knowledge to fill in any necessary information. The systems based on this approach typically consist of a syntactic parser, a collection of semantic routines that embody the kind of knowledge needed to interpret the meanings of words and

structures, and a cognitive deduction system to explore facts and answering questions. A description of this approach can be stated as follows:

> In designing these pieces, the main emphasis was on the interaction of three domains. The form in which we want to state a syntactic theory or a type of deduction must take into account the fact that is only a part of a larger system. One of the most useful organizing principles was the representation of much of the knowledge as procedures. Many other theories of language state their rules in a form modelled on the equations of mathematics or the rules of symbolic logic. These are static rules that do not explicitly describe the process involved in using them, but are instead manipulated by some sort of uniform deduction procedures. By writing special languages suited to the various types of knowledge, we are able to preserve the simplicity of these systems. This is accomplished by putting the knowledge in a form of programs in which we can explicitly express the connections between the different aspects of the system's knowledge, thus enriching their possibilities for interaction. [Winograd 73]

As stated previously, one of the main objectives of semantic analysis is to generate internal target representations from the natural language input. In general, there are four types of internal target representations, namely, weighted vectors, first-order predicate calculus, semantic networks, and case frames.

Natural language processing in some of the IS&R systems generates weighted vectors as the target representations where the weights represent the importance of the terms. For example, in the SMART system [Salton 68], each document is

represented by a vector of terms. By transforming the query into the_ weighted vectors, each query is identified as a vector. Thus, by using a similarity computation, the similarity between the query vectors and the document vectors is measured such that all documents that exactly contain all the query terms can be retrieved. Therefore, the intermediate language is not complex. For a detailed discussion, see Section 4.1 and Salton [Salton 83].

First order predicate calculus is another form of internal target representations. This form of representation is often used as the intermediate language in many information systems, such as Rendezvous [Codd 74; 78]. In those systems, the natural language processor translates the natural language input into a first-order predicate calculus statement in which the predicate may be a noun or a verb. The major advantage of using the first order predicate calculus as the internal target representation is that it can be implemented by using certain programming languages such as LISP. For a detailed discussion of this internal target representation, see Sandewall [Sandewall 71].

The third form of target representation is semantic networks. The semantic network is a network structure whose nodes are concepts expressed by natural language words and phrases, and whose edges, also called "semantic relations",

are the connections between concept nodes. Quillian [Quillian 68] is the first proponent of the use of semantic networks for natural language understanding. Many recent works on natural language processing also apply this form as the internal target representation, for instance, TORUS and CO-OP. A detailed discussion will be presented in Section 4.3 and 4.4.

The last major form of internal target representations are case frames. The concept of case frames was proposed by Fillmore [Fillmore 68] and originated from that of semantic networks. The central idea of this concept is that "the proposition embodied in a single sentence has a deep structure consisting of a verb (the central component) and one or more noun phrases." [Fillmore 68] Each noun phrase is associated with the verb in a particular relationship. These relationships are called cases and verbs are classified according to the cases that can occur with them. The cases for any particular verb, then, forms an ordered set called a case frame. For a detailed discussion of this form of target representations, see Section 4.4.2, Fillmore [Fillmore 68; 71] and Wilks [Wilks 76].

3.2.3 Execution Phase

At this phase of natural language processing, the system

attempts to extract results to the user's query. To accomplish this objective, the system must have search routines to perform database searching. For example, in some major publicly available IS&R systems, such as SMART, MEDLINE, and FIRST, the internal target representation of a query is conceptually similar to weighted vectors and the system searches for a Boolean combination of search keys; otherwise, one simply matches the possibly weighted query against the vector representing the document. In the latter case, the closest matching vectors represent the result [Salton 83]. The SMART system developed by Salton [Salton 68] represents this type of systems.

The second approach is to retrieve appropriate records by using Boolean conditions. That is, a given Boolean condition must be satisfied by qualifying records in a file. For example, if a Boolean AND condition is used, this condition will be used to search the desired records until it meets a false condition. As [Minker 77] suggested, the advantages of using Boolean conditions in retrieving desired information are time-efficiency and search optimization. For instance, if the query is "How many x's are written by y," the system uses the file management system to retrieve the basic data and then apply necessary functions to obtain the desired information.

The above two approaches basically apply the internal target representations transformed from natural language inputs to perform database searching directly. Therefore, a system developed based on these two approaches is mainly concerned with the problems relevant to natural language interfacing. If a formal query is allowed in such systems, another transformation process is required to translate the formal query into the same internal target representations.

There is another approach to developing the execution phase where the internal target representation is further translated into the formal query representation defined by the system. Thus, the task of information retrieval within such NLQS is still performed by the search procedures used in formal query database searching process. Some of such systems are TORUS and CO-OP. In following this approach, the biggest advantage is that all features supported by the formal query interface are considered in the design of natural language interface; thus, the capability of natural language processing will be at least the same as that of formal query processing. The only disadvantage ofthis approach is that it is time-consuming due to the additional time necessary for the translation process.

3.2.4 Natural Language Response Generation

In an NLQS, in response to natural language inputs, the system may be required to provide natural language replies in a precise, correct, natural language format or other appropriate formats such as tables or graphs. Therefore, the problem of text generation should be solved in the natural language interface design.

By examining existing NLQS's, there are at least four situations under which the system needs to make responses to the user's query. They are:

(1) Responses concerning the detected and/or corrected syntactic, semantic, or spelling errors within the user's query.

(2) Restatements of the logically completed query.

(3) Direct and correct answers to the user's query in the case that the answer exists in the database.

(4) Indirect and cooperative responses to the user's query in case that the answer to the query does not exist in the database due to the user's mis-conception of the database.

To generate any of these responses, the system must not only remember the user's query and construct a text generator in order to transform the system's response from the internal formal language response into a natural language output, but

also have appropriate routines to meet the requirements under the different response situations described above.

This chapter provided an brief overview of the development of natural language understanding. Also, different approaches to natural language processing were briefly described. Based on this overview, the next chapter will present a framework for the development of an NLQS which can perform syntactic and semantic analysis on natural language search requests and generate natural language responses.

CHAPTER 4

FRAMEWORK

This chapter proposes a framework for building a natural language IS&R systems. These systems will have available:

(1) A bibliographic database that stores textual records and is maintained by a database management system (DBMS), or file management system (FMS).

(2) A natural language interface which serves as a front end to the DBMS or FMS, which is given knowledge about the bibliographic database and linguistics, and which can understand and respond to simple natural language requests, and can engage the user in dialogue.

Such systems, as discussed in the previous context, can obviously play a very important role in the future in making bibliographic databases available to casual users who have neither the time nor the interest for learning an artificial language (such as the formal query language described in Chapter 2) before communicating with IS&R systems.

Based on the definition of a "framework" stated in Section 1.3, the goal of the framework presented here will be to identify the necessary components of an NLQS and their functions. Also, the framework will describe the relationships between system components with respect to the task of natural language information retrieval, rather than discuss issues relevant to the detailed system design and implementation. In Section 4.1, the design methodology is described. In this section, the required steps of NLQS design are first discussed; then, the appropriate approaches toward NLQS development are stated; finally, some major problems relevant to natural language processing that must be considered and solved in the design phase are identified. In Section 4.2, the necessary components of an NLQS, as well as their relationships, are identified. In this section, the overall system structure which includes three major interfaces, namely, natural language interface, formal query interface, and database interface, are first presented; then, the components which must be constructed to connect these three interfaces in performing the task of information retrieval are described.

As stated in previous chapters, the major goal of the NLQS is to automate the natural language search operations, the system has to have the domain-specified knowledge which is defined by the bibliographic database, and linguistic

knowledge. Therefore, in Section 4.3, these two types of knowledge are discussed and the approach to representing such knowledge is proposed. Finally, in Section 4.4, the functionalities of the components of an NLQS in the process of natural language information retrieval are discussed.

4.1 The Design Methodology

During the last two decades, a fair number of experimental natural language interfaces have been designed and implemented for some IS&R systems. Notable examples are SMART [Salton 68], BROWSER [Williams 69], LEADERMART [Kasarda 72], SIRE [McGill 76], FIRST [Dattola 79] and MEDLINE [Doszkocs 79]. While communicating with these systems, the online user can enter a search request in free-format form, i.e., English paragraphs, phrases, term lists or a combination of these. Then the retrieval software of the system analyzes the query and translates it into a suitable internal target representation, such as weight vectors. This translation process is similar to a text indexing process and consists of the following steps [Salton 83]:

(1) The individual words that make up a query are first recognized.

(2) A stop list, comprising a few hundred high-frequency function words, such as "of" and "but", is used to

eliminate such words from consideration in the subsequent processing.

(3) The scope of the remaining word occurrences is broadened by reducing each word to word stem form. For example, words such as INFORMATION and INFORMATIONAL are replaced by INFO. This step is done by using relatively simple suffix removal method together with special rules to take care of exceptions.

(4) Following suffix removal, multiple occurrences of a given word stem are combined into a single term for incorporation into the query vectors.

(5) By using a similarity function, all documents whose word stem vectors which are similar to the query vectors are identified and ready to be retrieved.

By examining these steps, it is obvious that only simple syntactic analysis is performed by the system in the process of query translation, and the knowledge used by the system is basically domain-specified. Such an approach has some disadvantages, such as:

(1) The system is unable to provide cooperative answers to the user. The reason is that, to provide cooperative answers, as stated by Kaplan [Kaplan 83], the system has to be able to extract the meaning of the user's query and

detect his misperception. Therefore, it is important for the system to perform a semantic analysis on the natural language query.

(2) Natural language queries to the system are restricted and have to contain key words and/or phrases that exist in the title and/or abstract of the desired documents. This restriction is defined by the indexing language and procedures applied by the system [Salton 83]. Therefore, although the system may retrieve the desired information for the user, it does not really understand the meaning of the user's request.

The framework proposed in this chapter intends to provide a block structure for an interactive IS&R systems so that they have the ability to "understand" and respond to natural language requests. To support such a function, the system would consist of a language processor that can perform not only syntactic analysis but also semantic analysis on the user's request.

To develop an NLQS which is capable of understanding natural language, a developer can referene the seven steps of casual-user language interface development proposed by Codd [Codd 74]. These steps are:

(1) select a simple data model which can describe the data in a relatively simple way, both syntactically and

semantically;

(2) select a high-level internal logic as the internal target representation, for instance, the predicate algebra or predicate calculus;

(3) introduce a strategy by which the system can keep the dialogue closely tied to the database description and the user's intended query;

(4) introduce system restatement of the user's query;

(5) separate query formulation from the database searching;

(6) employ multiple choice interrogation as fall-back; and

(7) provide a definition capability to permit new entries to be "understood" by the system.

These seven steps have been widely followed as the guidelines in the development of several NLQS's, such as TORUS, CO-OP, and PLANES. In addition to Codd's seven steps, Martin [Martin 73] also provided a checklist of possible steps in dialogue design. In his checklist, Martin suggested twenty-one criteria to help the designer in identifying the necessary tasks involved in dialogue design and implementation. In this chapter, the development of the framework will reference some of the seven steps discussed by Codd, some steps such as step 5 will not be followed and the

reasons will be described later in this chapter. This research also suggests that, while designing and implementing an NLQS, the designer and implementor be encouraged to reference the criteria mentioned by Martin.

The second issue that must be considered is how to determine the scope of knowledge needed for an NLQS. As stated previously, in order to have reasonable intelligence to understand and respond to natural language requests, an NLQS has to have both domain-specified and linguistic knowledge. The domain-specified knowledge is concerned with the subject matter and defined by the bibliographic database. This type of knowledge is actually stored in directories, inverted files and thesauri of IS&R systems. The linguistic knowledge required for a system is: (1) a set of grammar rules to determine the syntactic relationships between input words; and (2) a set of semantic rules or procedures to facilitate extraction of meaning and relationships of various concepts and generation of correct target representations for natural language inputs. In Chapter 3, several different approaches toward the development of the linguistic knowledge and target representations have been discussed. In this framework, the ATN grammar is proposed to facilitate the syntactic analysis, and the internal target representation of input sentences is represented in the form of semantic networks. Also, the "case frame" concept is adopted as the

intermediate between the syntactic constituents and the internal target representations of the input sentence. In other words, the system constructs case frames based on the syntactic constituents of a parse tree; then, the filled-in case frame is applied to determine the subset of the semantic network which represents the semantic meaning of the input sentence. A detailed description of the usage of the ATN grammar, semantic networks, and case frames will be presented in Sections 4.3 and 4.4.

After decisions are made with respect to the above issues, this research proposes an experimental approach to NLQS development. The fundamental assumption of this approach is that if an information system supports a natural language interface, this system has to "think" and "speak" as its user, a human being does. Based on this assumption, the designer has to understand the mainstream concepts of behavioral science such as learning psychology and human communication theories, as well as the concepts of "natural language understanding" and "knowledge representation" in the AI field.

Based on concepts of behavioral sciences and the discusssion in Section 3.1, it is reasonable to realize that it is impossible to develop a system whose "experience" and "intelligence" allow it to understand all the queries submitted by the users, and to generate a wide variety of

responses at its early stage of development. On the other hand, an NLQS should have a limited capability of both natural language understanding and natural language response generation, and this capability should grow gradually by the increase of its "experience" and "vocabulary" through its interaction with the users.

Based on the experimental approach proposed here, the development of an NLQS is accomplished by consecutive experiments. In each experiment, the designer should restrict the scope of queries in limited topics and obtain a certain amount of distinct natural language requests on these topics through interviewing sampled casual users; then, the designer should develop a subset of an NLQS which has the ability to understand and respond to those queries. The developed system, then, should be integrated into previously developed subsets of the system. In such a way, the knowledge base of the system, as well as its ability to understand and use the natural language, can be expanded. In Section 3.1, several capabilities of an NLQS have been identified as the basic requirements of an "intelligent" NLQS. In order to perform those capabilities, it is necessary to identify the major problems involved in natural language processing. There are at least three major problems that need to be solved in designing an NLQS. They are: redundant words/phrases, ambiguity and vagueness, and error detection and correction.

In the remainder of this section, the nature of these problems will be briefly examined and discussed.

4.1.1. Redundant Words / Phrases

Redundant means "exceeding what is needed or normal" or "using more words than necessary" [Merriam-Webster 74]. In the process of human communication, the use of redundant words or phrases, such as "please tell me" or "would you please list", is very important since it implicitly comprises certain social or cultural meanings. In the process of automatic natural language processing, these redundant words and phrases are meaningless and create a major problem in the natural language understanding process [Rich 83]. Therefore, to eliminate these words without changing the user's intention and concept is considered as the fundamental phase in natural language interface design.

4.1.2. Ambiguity and Vagueness

Ambiguity and vagueness are two major characteristics of natural languages. Though these two play important roles in the natural language communication between two intelligent information processing units (humans), they complicate the interpretation process, and create difficulties in the NLQS (for a detailed discussion, see Kaplan [Kaplan 78; 84]).

As described by Rich [Rich 83], natural language understanding is the process of mapping a statement from its original form (i.e., natural language) to a more useful one (i.e., formal language or target representation). Ambiguity means "the use of utterances which have multiple interpretations." [Kaplan 83] In other words, an ambiguous natural language query implies a one-to-many mapping between such a query and many of its formal counterparts. For example, saying "Select all books written by Date before 1970 and after 1980." may refer to the books written by Date before 1970 and the books written by Date after 1980, to the books written by Date before 1970 and after 1980, or to the books written by Date before 1970 and the books (written by anyone) after 1980. The interpretation of such a semantically ambiguous natural language expression requires a sensitivity to the context that is not typically required to process artificial languages. That is, the system requires a great deal of nonlinguistic knowledge (e.g., semantic rules or procedures) in order to compute multiple interpretations and make the correct choice among available target representations [Kaplan 83; Rich 83]. Hence, the choice of correct target representations for an ambiguous natural-language query is considered as one major issue in the natural language interpreter design (for example, see Codd [Codd 74; 78], Kaplan [Kaplan 78, 83, 84], Harris [Harris 76; 78], Waltz [Waltz 77] and Moyne [Moyne 77]).

Vagueness means "the absence of detail that would normally be explicit in formal database queries". [Kaplan 84] Kaplan also made the following statement on the nature of vagueness:

> In human question answering, vagueness provides a means for a respondent to actively contribute to the solution of a problem beyond a literal response, by allowing some latitude in the formulation of a response. A vague question may not specify precisely what information an appropriate answer should contain, or how the answer is to be formulated, and so the respondent can exercise some judgment in composing the response. Vagueness therefore provides a mechanism for two "intelligent" processors to both contribute actively to a conversation during the question answering process.

Although vagueness serves as a contributive role in the natural language communication process, like ambiguous natural language, a vague query also may complicate the interpretation process. For example, the use of pronouns and ellipsis, as described in section 3.1, represents the most common type of vague queries. In response to elliptic queries, the system has to fill out incomplete words or phrases using terms already mentioned in past question answering sequences. Therefore, how to record and reference the previous context (i.e., the history of the dialogue) in order to formulate the elliptic natural language query should be solved in the natural language interface design [Kaplan 84; Barr 81; Codd 78].

4.1.3. Error Detection / Correction

The human errors that occurs during man/computer interaction process are mostly the cause of the failure of information retrieval tasks. Codd had tried to categorize various human errors into five categories and to examine the possibility of detecting and/or correcting those errors (for a detailed description, see [Codd 78]). Among those errors, he suggested that an NLQS should be able to detect typographic and spelling errors during the process of language processing. As for other types of errors, he suggested that, although the system may not be able to detect them, by the use of restatements or menu-driven dialogue, the user may review his query and modify his query in case it contains errors [Codd 78].

In Codd's approach to human-error handling, if any human error is detected, the user will be required to read a sequence of restatements and to answer a series of menu-selection questions which are concerned with the detected errors, even if the system has already selected a correct substitute. Such an error handling approach sometimes can not meet the needs of casual users which have been described in Chapter 1.

On the other hand, Kaplan [Kaplan 84] suggested that the natural language processor should be able to detect as many

errors as possible and automatically correct simple typographic and misspelling errors. For some detected errors that the system is unable to correct, such as the user's misconceptions of databases, the system also needs to provide indirect, informative responses to help the user to modify his query (see Section 3.1).

In Kaplan's approach, if a simple human error, such as typing or spelling errors, is detected and the system can find an obviously correct substitute, the system should automatically replace the error and provide simple dialogue to obtain the approval from the user without asking him any further questions. For example, if the query is:

"List all books writtenby Date."

The system should be able to detect the omission of a blank between "written" and "by", and correct "writtenby" automatically rather than ask the user to answer questions like:

I don't know the word "writtenby", is it a typing error?
If it is a typing error, please repeat your query.
Otherwise, . . .

>>(user)

To perform the above error-handling capability, an NLQS should include some routines to detect "detectable" errors

such as typographic or spelling errors, as well as routines to determine whether and how a detected error may be corrected by using the "intelligence" of the system within the database and KB.

4.2 Interfaces of Natural Language Query Systems

The major advantage of an NLQS is that, as shown in Figure 2.7, it provides a user's view level such that the user may enter his search request based on his own concept of information and his ability in the natural language. To support this level, the system needs to extract the user's intention from the natural language request, processes it, and generate its responses to the user in the form of natural language. In this framework, the above processes are performed by the connection of three levels of interfaces, namely, the natural language interface, the internal query interface, and the database interface, as shown in Figure 4.1.

```
        +----------+
        |   USER   |
        +----------+
             |
             | natural language
             | query
             V
  +---------------------+
  |  NATURAL LANGUAGE   |
  |     INTERFACE       |
  +---------------------+
             |
             | internal representation
             | (semantic nets)
             v
  +--------------------+
  |  FORMAL LANGUAGE   |
  |    INTERFACE       |
  +--------------------+
             |
             | formal query
             |
             v
  +--------------------+
  |     DATA BASE      |
  |     INTERFACE      |
  +--------------------+
             |
             V
   +-----------------+
   |  DATA   BASE    |
   |                 |
   +-----------------+
```

Figure 4.1 Overall Structure of an NLQS

Natural Language Interface

This is the interface of the casual user/system interaction. At this level, the user enters his natural language search request that is non-procedural and informal. In order to invoke appropriate search procedures related to this request, the system needs to "understand" the meaning of the query. In other words, the system must perform both syntactic and semantic analysis on the natural language query, and produce an internal target representation of that query. In addition, the system also needs to generate its response which might be a restatement, indicating the system's interpretation of the query; a multiple-choice question for the purpose of clarification; or the result of the database search performed by the system. Therefore, at this level, the system performs three of the four phases of operation of natural language processing, namely, syntactic analysis, semantic analysis, and response generation.

To carry out those operations, the required functional features of this interface are identified to be a parser, which performs syntactic analysis; an interpreter, which carries out semantic analysis; and various response generators, each of which generates appropriate responses to the user. In addition to these features, in order to select a specific type of response as the system's feedback to its user, a dialogue control feature is required. A detailed

discussion of these functional features will be presented in Section 4.4.

Formal Query Interface

This is the interface in which the system accepts a syntactically and semantically unambiguous target representation of the natural language input and translates it into one or more formal queries defined by the specific IS&R system.

Generally speaking, every IS&R system uses a specific type of internal representation for database search, and almost all of the IS&R systems do not perform semantic analysis on the user's request. An early study on the features of different IS&R systems conducted by Meister [Meister 67] found that there was no IS&R system performing any semantic analysis on input queries. (1) Also, the target representation of each IS&R system may be different from those of other systems. In the framework proposed in this

---

(1) Although Meister's study was completed more than ten years ago and many advanced IS&R systems have been developed after that study was published, Meister's finding is still strongly supported by more recent evidence. For example, after examining language processing in some IS&R systems, Jones [Jones 79] claimed that an IS&R system could get good search results with simple terms and weights and without the necessity of developing the representation of meaning of the user's request. Also, Salton [Salton 83] pointed out that a concept similar to Jones' was the one which dominates IS&R system development.

research, the internal target representation generated by the natural language interface is fundamentally different from those of other IS&R systems. First, the target representation discussed in the framework conveys the semantic meaning of the user's query. Second, this target representation is independent of any IS&R system and cannot invoke database search in any IS&R system unless it is further transformed into a form which can be recognized by that IS&R system. Therefore, it is necessary to construct an intermediate level between the natural language interface, which accepts natural language input and generates its internal target representation, and a database interface, which performs database search by using formal language queries. This intermediate level is the formal query interface. This interface carries out a further transformation operation, i.e., to transform the standard target representation produced in the natural language interface into the formal query accepted by a specific IS&R system and to invoke appropriate search programs.

In view of this, the natural language interface developed in this framework is expected to have a high degree of portability. That is, a natural language interface can be built on any IS&R system by simply providing a formal query interface and modifying domain-specific knowledge to cope with the specific nature of that IS&R system.

Data Base Interface

This is the interface that examines the formal query, invokes search procedures, communicates with the bibliographic database, and retrieves the desired information for the user. Although an NLQS accepts natural language requests, after the activities of language processing performed by the previous interfaces, the input to this interface is in the form of formal queries defined by the system. Therefore, the execution of the user's search request can be carried out by invoking search programs. The search results of this interface are transmitted to the natural language interface such that a natural language response can be produced.

Based on the above descriptions of NLQS interfaces, a block diagram of the NLQS which identifies the components of this system is shown in Figure 4.2. In this diagram, the blocks connected by single lines ("|") represent the processes that constitute an NLQS, while the blocks connected only by dots ("...") represent two major information sources, namely the KB and the bibliographic database (the KB consists of semantic networks, lexicon and various dictionaries used). In the remainder of this chapter, this block diagram will be discussed such that the functions of each block and the relationships between blocks can be revealed.

```
-------------------------> USER
A       A                   |
|       |_                  | (NL input)
|       |                   V
|       |             +-------------+
|       |         +---|   PARSER    |<................
|       |         |   +-------------+ ............   :
|       |         |         |                   :   :
|       |     (word)        |(parsing tree)     :   :
|       |         |         V                   V   :
|       |         |   +-------------+      +----------------+
|       |         |   | INTERPRETER |<.....|                |
|       |         |   +-------------+.....>|   KNOWLEDGE    |
|       |         |         |              |      BASE      |
|  +-----------+  |         |              |                |
|  |  ERROR    |  |         |              +----------------+
|  | GENERATOR |  |         |(semantic graph)   :
|  +-----------+  |         V                   :
|       A         |   +-------------+           :
|       |         +-->|   DIALOGUE  |           :
|       +-----------|   CONTROL   |           :
|        (error)    +-------------+           :
|                         |                   :
|           +-------------|                   :
|           |             |(semantic graph)   :
|           |             V                   :
|           |       +-------------+           :
|           |       | FORMAL QUERY|<...........:
|           |       |  GENERATOR  |
|           V       +-------------+
|  +-------------+        |                +-------------+
|  | RESTATEMENT |        |(formal query)  |             |
+--|  GENERATOR  |        |...............>|BIBLIOGRAPHIC|
   +-------------+        |<...............|  DATA BASE  |
                          V                |             |
                    +-------------+        +-------------+
                    |   RESULT    |
                    |  GENERATOR  |
                    +-------------+
                          |
                          |(NL output)
                          |
                          V
                         USER
```

Figure 4.2 Natural Language Data Base Searching

4.3 Information Sources

In this section, the representation of information in an NLQS is described. Information is divided into two general categories and is stored accordingly in the inverted files and bibliographic database available to the NLQS or on the semantic network in the KB.

4.3.1 Information About Part of a Document File

In many IS&R systems, such information is stored in bibliographic databases and utilizes inverted file structures to organize such information. In the inverted file, some fields of the document record or terms in the documents texts within the bibliographic database are used as indices for database search. Each index term points to a set of document reference codes and/or a number which indicates the number of corresponding documents. The following table illustrates such a relation:

Index Term	Doc Ref Code	# Docs
computer	3, 5, 8, 9	4
information	1, 2, 4, 7, 9	5
:	:	:
:	:	:

If the search term in a query is "computer", then, the system can determine that there are four documents containing such a term, the document reference codes of those documents being 3, 5, 8, and 9. These retrieved document reference codes form a set called Set 1. If the user continues his search by entering a new term "information", a new set called Set 2 = {1, 2, 4, 7, 9} is formed. Then, by using the Boolean operators AND, OR or NOT, the user may obtain a new set formed by different combinations of Set 1 and Set 2. By using those sets, the system may retrieve desired documents corresponding to those sets.

The above description also shows the simple semantic relationships between index terms in the inverted file and document records in the bibliographic database.

4.3.2 Information About a Document File

Such information is stored in the KB and consists of the general knowledge the system has about search terms, document reference codes, and the bibliographic database, along with

the system's understanding of the on-going dialogue it may be having with a user.

In this framework, a semantic network is used to represent this category of knowledge. Nodes on the semantic network can be concepts, characteristics and events. Thus, the generic entities AUTHOR, JOURNAL, TEXT TERM, as well as instantiations "Martin, T.", "BYTE", and "computer" are all concepts. Similarly, the generic entities of SELECT (such that every particular "search" action is its instantiation), as well as specific instantiations of the entity, are all events. Finally, ideas which express properties of objects, actions or other properties are characteristics. For example, Doc Ref Code is a property of search terms such as TEXT TERM and their semantic relationship can be represented as Figure 4.3.

Objects (Nodes) in the semantic network are organized in a main-sub hierarchy, and the nodes in the hierarchy are defined by SUB-edges and E-edges. SUB-edges only link generic nodes while E-edges always link a generic node and an instantiation node. The graph defined by SUB-edges and E-edges are acyclic. For example, books and articles are the subconcepts of publications while "Introduction to Data Base Management Systems" is an example (instantiation) of the book concept. Such semantic relationships on the semantic network can be represented as in Figures 4.3 and 4.4.

```
 +---------- S -----------+
 |  +------ D --------+   |
 |  |                 |   |
 |  |                 V   V
 |  |          +----> TEXT TERM ----------->|
 |  |          |                            |
EXPAND          |                            |
               +----> AUTHOR -------------->|
               |                            |
          D    |                            | O
SEARCH------>|----> PUBLISHER ----------->|----> DOC REF CODE
               |                            |
               |                            |
               +----> TIME PERIOD -------->|
               |                            |
               |                            |
               +----> TITLE -------------->|
```

S : Subject
D : Destination
O : Object

Figure 4.3 Portion of Semantic Networks (1)

```
                TEXT TERM --------------> "computer"
                     |            (v)
                     |
                     |
                     |(CH)
                     |
                     |
                     |
                     V
               DOC REF CODES
                     |
                     |
                     |
+----------+----+-----+-----------+
|          |          |           |
|(v)       |(v)       |(v)        |(v)
|          |          |           |
V          V          V           V
3          5          8           9
```

CH: CH-edge (chraracteristic)
v : V-edge (value)

Figure 4.4 Semantic Relationships Between TEXT TERM
DOC REF CODES

```
DOC REF CODE          TEXT TERM              TIME      AFFILIATION
     |                                        |
     |  |                 |                   |              |
     |  |              (D)|                   |(CH)          |(CH)
     |  |    (O)          |                   |              |
 (CH)|  +-------------CONTAIN                 |            AUTHOR
     |                                        |
     |                                        V              |(D)
     |                                     WRITE ---------+
     +--------------> PUBLICATION   (O)     |
                            |        +---+--+
                       (SUB)|        |      |
                +-----------+ ------|-----+|(O)
                |             |       |     ||
                |   (CH)      V       |     VV    (v)
                | +-----> ARTICLE<--+    BOOK ------> title
                | |           |
                | |           |             |(O)
                | |           |             |
                | |      +----+          PUBLISH
                V |      |    |             |
           JOURNAL      |    |(v)          |(D)
                |       |    V             |
                |       |  title           V   (CH)
                |       |               PUBLISHER--> LOCATION
                |       |
           (CH)|        |
                |       |
                |       |              CH:  Characteristics
   +-----+----+---+    |              SUB: Sub-edge
   |     |    |   |    |              D:   Destination
   V     V    V   V    |              O:   Object
 TIME  VOL   NO  PAGE  |              v:   value
                       |(CH)
          |    |   |    |
          |    |   |    |
          +----+---+---+
```

Figure 4.5 Portion of the Semantic Networks (2)

The properties of any object, i.e., the characteristics, are linked to the object by a CH-edge, and the value(s) of a characteristic is linked to that characteristic by V-edges. For example, TEXT TERM and DOC REF CODES are objects in Figure 4.3, since each TEXT TERM may determine a set of DOC REF CODES. Therefore, DOC REF CODES is the property of the object TEXT TERM, although both of them are objects. Figure 4.4 ilustrates the relationship between a TEXT TERM "computer" and its corresponding set of DOC REF CODES {3, 5, 8, 9} by using the CH-edge and V-edges.

Figure 4.3 through Figure 4.5 show portions of the semantic networks that can be constructed for the document file of an IS&R system which applies inverted file structures. In the remainder of this chapter, these figures will be used to describe all the examples presented.

4.4 Features of a Natural Language Query Systems

In this section, natural language processing carried out by the NLQS is described in terms of the functions performed by the processing units of the NLQS (see Figure 4.2).

4.4.1 The ATN Grammar Parser

This parser is responsible for producing a parse tree of

the input sentence. The major function of this process, as stated in Chapter 3, is to identify the syntactic constituents that correspond to most of the semantic units appearing in the representation of the semantic meaning, and to provide a syntactic relation which can guide the later attempt to find semantic relations required in the next stage of language processing.

Based on the above considerations, the parser proposed in this framework applies to an ATN grammar. The reasons for making such a choice are as follows:

(1) Many current natural language processors use ATN-like grammars for syntactic analysis, and most successful natural language systems, such as Woods' LUNAR and Winograd's SHRDLU use an ATN grammar parsers. Thus, in some sense, the ATN grammar parser is considered the state of the art.

(2) By using ATN grammars, Woods [Woods 72] and Winograd [Winograd 72] both found that the parser could deal in a comprehensive way with both syntactic and semantic aspects of language processing. The advantage of ATN's over the more traditional transition networks, as described by Winograd [Winograd 83], is that conditions and actions are associated with the network arcs. Also, Hendrix [Hendrix 81] found that by using the basic

mechanisms of ATN grammars, a natural language system can be developed in a clean and easily programmable way.

In this section, the syntactic analysis carried out by the ATN grammar parser can be divided into two steps, namely, lexical analysis and parse tree generation.

The first step, lexical analysis, is a complete word by word recognition and transformation process. At this step, the parser simply looks up each word of the input sentence in the lexicon. The lexicon contains three categories of lexicon classes, namely, noise words, command words, and thesaurus words. The operations involved in the lexical analysis are to: (1) delete noise (or redundant) words such as "please"; (2) replace input words by their lexicon class entries, for example, "give me" is replaced by "select"; (3) substitute single words for corresponding phrases, for example, "IS&R" is substituted for "information storage and retrieval".

After the word-by-word recognition and transformation process is completed, a phrase construction routine is called to recognize the phrases in the input sentence. For example, two consecutive words "information" and "retrieval" will be recognized as a single phrase "information retrieval" rather than as two individual words.

If an input word cannot be found in the lexicon, a spelling correction routine should be invoked. This routine

will try to find lexicon entries close to the input word. If one entry_(candidate) is found, it is inserted in the place of the misspelled word. If more than one candidate is found, each candidate will be used to generate a parse tree at a time (this process will be further described in the next step). If no candidate is found, the input word is treated as a new lexicon entry and added to the lexicon by finding a synonymous word or phrase already known to the system, and this operation is performed by an "entry addition" routine. By adding this unknown word, the system is able to learn new words and expand its knowledge. In the last two cases (multiple candidates or no candidate found), the ambiguous input word along with all the candidates (if any) are passed to the next step of syntactic analysis.

After the first step is completed, a linked list of lexicon entries, which includes multiple entries for ambiguous words, is generated and passed to the ATN grammar for the second step of syntactic analysis. The second step analysis is non-deterministic, and several parses are possible for the same sentence. However, the parser produces one parse tree at any time. Only if the semantic analysis for that parse tree turns out to be impossible, will the new parse tree be produced. By using this method, the lexicon ambiguity stated previously and some syntactic ambiguity can be resolved. The solution of lexicon ambiguity is, then,

passed to the interpreter for semantic analysis. If no parse tree can be generated from the ambiguous lexicon entries, the ambiguous input word and all the candidates are passed to the dialogue control for generating an error message.

The structure of the ATN grammar parser is a set of subnets [Barr 81]. Each subnet only matches phrases with specific meaning. For example, in the bibliographic database environment, there are subnets for each different semantic object: author, time period, and so on. Some examples of phrases which the subnet for "time period" would match are: "between Jan 1 and March 1 1984", "during Feb and March of 1984", etc. Most subnets match noun phrases (NPs) and prepositional phrases (PPs). The construction of subnets can reference Winograd's analysis of NPs [Winograd 72].

Qualifiers are most commonly used in unrestricted natural language sentences. Examples of qualifiers are the underscored portions of "books <u>written by Martin</u>," and "titles <u>containing computer</u>." To analyze qualifiers, a special subnet is applied.

Since qualifiers are dependent clauses and often appear after the main noun in an NP, they are often introduced by relative pronouns such as "which" (for example,"books <u>which</u> were published by ...") and "that", or by participials (i.e., roots ending in <u>-ed</u> or <u>-ing</u>), for example, "books writt<u>en</u> by

Date"). Since qualifier syntax is fairly restrictive, in many cases, to search for a qualifier, the parser merely examines the single word after the main noun. If a qualifier is identified, the parser performs a sequence of operations to separate the qualifier from the main clause and to analyze the qualifier. Waltz [Waltz 76] suggested some procedures for handling the qualifier analysis. They are: (1) determine the boundaries of the qualifier; (2) if the qualifying clause is not grammatical, a heuristic routine is invoked to bracket the clause; (3) the current history register (will be discussed in the next paragraph) values for the main clause is pushed down such that the processing of the main clause can be suspended; (4) if any relative pronoun is used in the qualifier, the main noun from outside the qualifier is used to substitute it as a phrase element in the qualifier; (5) process the qualifier like a normal request, with the main noun serving as the requested item; (6) after the above operations are completed, the parser should integrate this qualifier subnet with the main clause subnet in order to form an overall query.

Other problems of syntactic analysis are ellipsis and pronoun reference. To solve these problems, a history keeper routine is required [Codd 74]. To develop this history keeper, a set of registers, called history registers, can be organized as a push-down stack. Whenever a subnet matches a

phrase, it sets the value of a corresponding history register. Therefore, if some terms in a history register have been left unspecified or replaced by a pronoun, the values of the history registers from the previous requests can be examined and a suitable value can be used to supply the missing information or the referent of a pronoun.

At end of syntactic analysis, a parse tree, which is in the form of a linked list of subnets, is generated and passed to the interpreter such that the system can carry out semantic analysis on the input sentence.

4.4.2 The Interpreter

The interpreter is responsible for generating a subset of semantic networks for the input sentence as the internal target representation. As stated in Section 4.3, semantic networks consist of concepts, characteristics and events connected through semantic relations (i.e., edges). To translate the meaning of a parse tree into a subset of semantic networks, Simmon [Simmon 72; 73] suggested a case frame approach. According to this approach, each case frame represents a subset of semantic networks and is identified by the verb or characteristic. Also, a case frame consists of a list of NPs which are related to each other with respect to the specific verb or characteristic. Therefore, when the

interpreter receives a parse tree from the parser, it uses the verb_ to search for an appropriate case frame by filling in the NPs in that case frame. If all the NPs match the specifications of a particular case frame, then that case frame can be used to represent the semantic meaning of the request. For example, the case frame for WRITE has this form:

```
WRITE: DESTINATION
       = AUTHOR (SUBJ)
       OBJECT = ARTICLE (OBJ)
```

To fill the syntactic constituents of a parse tree into a specific case frame, the interpreter needs to apply certain case fitting algorithms to search for eligible nodes which match the specification of the case frame. In the above example, to fill in the case frame WRITE, the subject of the syntactic constituents must be an author and the object must be an article.

After the case frame is identified, the values of NPs can be assigned to the corresponding nodes in the semantic network which are linked by V-edges. For example, the portion of the semantic net for the case frame WRITE can be represented as:

```
WRITE ----+-----------> AUTHOR --------------> "Martin, T."
          |    D                       v
          |
          |
          |
          +---------> ARTICLE -----> "Information Retrieval"
             OBJ                v
```

The filled-in case frame along with the command word produced in the lexical analysis is then passed to the next processing unit of the system, namely, the dialogue control, for subsequent language processing.

If the parse tree can not fill in any case frame, then the system is unable to determine the semantic meaning of the input request. In such a case, the failure may be caused by the misuse of verbs or characteristics (while NPs are actually related to each other), or a user attempts to link unrelated NPs together. Therefore, the interpreter needs to re-examine other case frames and find out whether the failure is caused by the misuse of "semantic relations" (i.e., verbs), or the unfilled-in NPs. To make the above decision, a "diagnostic routine" which uses heuristic searching techniques [Jackson 76; Barr 81; Rich 83] to perform error-detecting functions is invoked. After the error type is determined, the parse tree and the NPs which could not fill in any case frame or the misused semantic relations are transmitted to the dialogue control in order to generate appropriate feedback to the user.

If more than one case frames match the parse tree, semantic ambiguity is encountered. To solve this problem, Moyne [Moyne 77] proposed a solution. In his solution, the

interpreter assumes that each of the filled-in case frame may be the user's intent. Therefore, by using a Boolean operator OR, all these case frames are connected. This method is a reasonable solution because of the following reasons:

(1) If all but one case frames generate null answers, then the only non-null answer may fit the user's expectation.
(2) If more than one case frame can be applied to retrieve meaningful answers (i.e., non-null answers), one of them may be the desired answer. Although other meaningful answers might not directly meet the user's expectation, they can be used as references which provide information about documents related to the search term.
(3) Since all candidate case frames will be passed to the dialogue control and be used to generate restatements, even though the system is unable to determine which frame is the desired one, the user may help the system to clarify the semantic ambiguity by selecting a desired interpretation from those candidate case frames.

4.4.3 The Dialogue Control

The dialogue control takes as input the output from both the parser and the interpreter, and examines the input in order to make appropriate decisions on the subsequent step of language processing:

(1) If the input to the dialogue control is a parse tree output from the parser, the dialogue control recognizes that the syntactic analysis failed due to misspelled or unknown words. Thus, it passes the misspelled/unknown words and the candidates chosen by the parser to the "error generator" such that it can generate an error message to the user.

(2) If the input is a parse tree along with NP(s) passed from the interpreter, the dialogue control recognizes that semantic analysis failed due to no appropriate case frame matching the parse tree. In such a case, the NP(s) is passed to the "error generator" so that an error message can be generated.

(3) If one or more case frames are the inputs to the dialogue control, the case frame(s) are passed to the "restatement generator" for generating a restatement which indicates the system's interpretation of the user's request. In addition, a copy of the case frame(s) is transmitted to the formal query generator in order to translate the case frame(s) into corresponding formal query (or queries).

4.4.4 Formal Query Generator

The formal query generator is responsible for translating the filled-in case frame(s) into one or more

formal query expressions for use with a bibliographic database system. This translation involves:

(1) Determining the types of the user's request by checking the command word in order to search for a set of corresponding formal query formats defined by the system. For example, if the command word is "select", the formal query generator recognizes that a search request has been submitted and the expected response is a number indicating the number of documents. Thus the format "<search command> <search term>" is selected.

(2) Selecting the specific search command word(s) defined by that specific IS&R system.

(3) Selecting the search term(s) to examine in order to retrieve the information necessary for answering the user's request. For example, if the case frame is:

```
CONTAIN: DESTINATION
         = TEXT TERM (SUBJ) = computer
         OBJECT = DOC REF CODE (OBJ) = ?
```

then a formal query "SELECT TEXT TERM eq "computer" is formulated.

After a formal query is formulated, it is ready to be used for invoking database search procedures. The search operations are carried out when the user has approved the system's restatement. If the user did not approve the

restatement, that implies that the formulated query does not represent the user's needs. The formulated query is abandoned by the system, and the user's response to the restatement invokes a user/system dialogue for the purpose of further clarifying the query.

4.4.5 Natural Language Response Generators

These response generators are responsible for generating appropriate feedbacks to the user. As stated previously, there are four situations in which the system needs to generate feedbacks. They are

(1) a misspelled/unknown word is detected;

(2) no case frame can be found to match the parse tree;

(3) one or more case frame(s) have been filled in and the user's approval (or his decision on which case frame(s)) is needed;

(4) the search results are generated.

To handle these different types of feedbacks, in this framework, three natural language response generators are required, namely, the error generator, restatement generator, and reply generator. In the remainder of this chapter, the functions of these generators will be described.

Error Generator

The error generator is responsible for generating an error message along with multiple-choice question(s) to the user whenever the system is unable to process the input sentence successfully. There are two major types of error messages generated by this processing unit:

(1) If the input from the dialogue control is the misspelled/unknown word and the candidate entries, the error generator produces an error message indicating that the input word may be misspelled, then, a multiple-choice question is presented. In this question, all candidates for that word are listed (if there are more than one candidate); the unknown word is output and a yes-no question is provided to clarify whether the word is a misspelled word or a correct one.

(2) If the input from the dialogue control is a parse tree and some NPs or a verb, the error generator produces an appropriate error message based on the error term (NP or semantic relation, i.e., verb) and the parse tree.

As Codd [Codd 74] suggested, although casual users are likely to be unwilling to tolerate multiple-choice questions which are concerned with his query, the use of such a type of question is the most effective and efficient way to clarify the doubts or ambiguity the system encountered during

language processing. Therefore, the above interrogations concerning with the user's query are always multple-choice questions.

Restatement Generator

The restatement generator produces a natural language restatement which expresses the system's understanding of the request. The functions of this generator are as follows:

(1) Since the problems involved in the user's request such as ellipsis and pronoun references have been resolved by the system, by producing a restatement, the user may approve or disapprove the system's resolution by checking the substituted or added nouns in the restatement.

(3) If the user does not approve the system's interpretation of his request, he can express his intent via a clarification dialogue with the system. The system may ask whether the user wants his request executed on the bibliographic database, or if he wants to submit a new query which is relevant to the current query. Therefore, the user can terminate the execution of his request, or make minor changes to his request. For example, suppose the user wanted data for "information retrieval" but typed "information" instead, he may correct it by simply typing "no" when asked by the system "shall I execute

this query <...restatement...>, please answer 'yes' or 'no'."; and then typing "information retrieval." The system will recognize this as a modified search term and substitute it for "information".

The restatement is straightforwardly constructed from the case frames. If more than one case frame is used, the system generates a multiple-choice restatement which contains different interpretations of the request. Each interpretation is labelled by a serial number such that the user can select one (or more) restatements as his perception. Then, the response to the restatement can be used to invoke subsequent operations of language processing.

Reply Generator

Once the desired information has been retrieved, the results are passed to the reply generator, which decides on an appropriate output format for the results. If the retrieved information is an integer, such as the number of documents related to the search term(s), the reply generator simply output the number of items. If the results are a list of documents, the system needs to decide how to arrange the output documents. Therefore, a question which is concerned with the output format will be produced. After obtained the user specified output format, the reply generator will generate output based on the specified format.

After the reply generator outputs the search result, it returns control back to the system driver. The system then needs to invite the user to continue the dialogue by providing a simple prompt question (e.g., a "yes" or "no" type question).

If the user wants to continue his database search, the system driver passes control to the parser for processing new requests. If the user wants to terminate his database search operations, the system will ask the user whether he wants to save the search profile, then, automatically sign the user off and terminate the search session.

4.5 Overview

In this chapter, an NLQS has been partitioned into three levels of interfaces. The natural language interface is responsible for translating natural language queries into their internal counterparts. This natural language translation process includes syntactic and semantic analysis, and is performed by the processing units such as the parser and interpreter, respectively. The result of natural language translation is passed to the dialogue control, which examines the input and determines whether a formal query should be generated for database search, or a specific error message should be displayed. The formal query interface is

responsible for accepting a syntactically and semantically correct internal target representation of the natural language input and transform it into its formal counterpart defined by the system. This transformation process is performed by the formal query generator. After the formal query is produced with respect to the user's request, this formal query, in the database interface, invokes appropriate search procedures to retrieve desired document information and return it to the user via the natural language interface.

Since natural language processing requires both linguistic and domain-specified knowledge, the NLQS proposed in this framework contains the above knowledge in its KB and database. The KB contains all the information that the system needs in understanding the bibliographic database as a whole semantically and the on-going dialogue with its users. By reviewing the discussion in this chapter, it becomes clear that the knowledge of the KB consists of a lexicon, ATN grammar rules, case frames, and semantic networks. In addition, the thesauri and inverted files of the conventional IS&R systems can also be integrated into the lexicon to provide necessary information in the lexical analysis. The database contains document records in their natural language textual forms.

Finally, a "linearization" of the natural language understanding process was adopted to explain the functional

units of an NLQS and their relationships, as well as their use of the information resources within the system (i.e., KB and database). Although this research is to propose a framework which is focused on the development of a high-level structure of an NLQS and describe the functionalities of various system processes, it is strongly believed that, by applying this framework as the basis of NLQS design and implementation, the problems encountered at each stage of natural language processing and the mechanisms to solve those problems can be easily identified.

CHAPTER 5

CONCLUSIONS

Ever increasing numbers of casual users are using the computer as an occasional tool in their own area of interest. As a result of this trend, greater emphasis is being placed on well-designed, user-friendly interfaces to make computer tools available to this class of users. NLQS development is an attempt that allows the casual user to directly retrieve desired documents from a computerized IS&R system based on his own concepts and knowledge expressed in natural language.

During the last two decades, several experimental NLQS, such as SMART and FIRST, have been developed to facilitate the casual user/computer interaction in the process of document retrieval. The basic assumption of the above development is that, "in retrieval one needs to render a document retrievable, rather than to convey the exact meaning of the text" [Salton 83]; thus, the processing of natural language requests in those systems requires no semantic analysis.

The underlying assumptions of an NLQS proposed in this

framework —are different from that of the above systems. The first assumption is that the purpose of NLQS development is to build a machine which can simulate human communication behavior. Therefore, NLQS should be able to "understand" human language, i.e., capture the user's intent during the man/machine interaction. Thus, the NLQS proposed in this framework would perform both syntactic and semantic analysis on natural language requests, and generate natural language responses to its user.

The second assumption is that, as discussed in Chapter 2, there are four components of an information system, namely, users, interfaces, databases, and tasks. Each of them is dynamically related to each other. A change in any component will affect other components. Therefore, in constructing a natural language interface to facilitate casual user/system interaction, the NLQS proposed in this framework consists of both domain-specified knowledge defined by the bibliographic database and a knowledge base which provides sufficient information about the semantic relations between the fields of document records, as well as the information about the on-going dialogue (see Sections 4.3 and 4.4). In addition, because of the existence of semantic information related to natural language sentences, this NLQS not only performs fact retrieval operations, but also simple question-answering operations which are concerned with

specific data items.

Based on the above assumptions, a descriptive model of the NLQS has been presented in this research (see Figure 4.2). In this model, the processing units of an NLQS, as well as their functions and relationships with respect to natural language processing, have been identified and discussed. There are a number of issues not discussed. For instance, some auxiliary features such as "help" routines are important since they provide information about the database to the user and assist him to formulate meaningful requests. Also, issues relevant to the detailed system design and implementation are not covered. Therefore, further research and study on NLQS development based on the fundamental features provided in this framework is encouraged.

APPENDIX A

Hendix's Natural Language Capability List

In "A Tutorial on Natural Language Processing" [Hendrix 80], Hendrix and Carbonell examined the development of natural language query systems and identified a list of natural language capabilities which are important for the practical application of a natural language query system.

(1) Access multiple, remote databases.

(2) Answer direct questions.

(3) Coordinate multiple files.

(4) Handle simple uses of pronouns.

(5) Handle restricted ellipsis.

(6) Do basic report generation.

(7) Dynamic extension of linguistic coverage.

(8) Analyze NULL answers.

(9) Restate in English the system's interpretation of inputs.

(10) Correct spelling errors.

(11) Enhance the data in a database with special-purpose functions.

(12) Exploit limited-domain semantic constraints to correct extra-grammatical input, which abounds in human dialog.

(13) Provide usable natural language access to specific databases.

BIBLIOGRAPHY

[Abrial 74]. Abrial, J.R., "Data Semantics", in Klimbie, J.W. and Koffeman, K.L. (eds.), Data Base Management. North-Holland Co., 1974. pp. 1-58.

[Aho 79]. Aho, A.V. and Ullman, J.D., Principles of Compiler Design. Reading, Mass.: Addison-Wesley. 1979.

[Bailey 73]. Bailey, R.W., S.T. Demers, and A.I. Lebowitz, "Human Reliability in Computer-Based Business Information Systems," IEEE Transactions on Reliability, R-22, 3. 1973. pp. 140-148.

[Barr 81]. Barr, A. and Feigenbaum, E.A., The Handbook of Artificial Intelligence, Vol. I, Stanford, Calif.: HeurisTech Press. 1981.

[Beizer 83]. Beizer, B., Software Testing Techniques. N.Y.: Von Nostrand Reinhold Co., 1983.

[Burton 76]. Burton, R.R., "Semantic Grammar: An Engineering Technique for Construction Natural Language Understanding Systems," Technical Report 3453. Cambridge, Mass.: Bolt Beranek and Newman. 1976.

[Chomsky 59]. Chomsky, N., "On Certain Formal Properties of Grammar," Information and Control, Vol. 2, No. 2. 1959. pp. 137-67.

[Codd 71]. Codd, E.F., "A Data Base Sublanguage Founded on the Relational Calculus", Proc. ACM-SGFIDET Workshop on Data Description, Access and Control. Nov. 1971, ACM, N.Y., pp. 35-68.

[Codd 74]. Codd, E.F., "Seven Steps to RENDEZVOUS with the Casual User", in J.W. Klimbie and K.L. Koffemann(eds.), Data Base Management. North-Holland Co., 1974, pp. 179-199.

[Codd 78]. Codd, E.F., "How About Recently?", in B. Shneiderman(ed.), Data Bases: Improving Usability and Responsiveness. London: Academic Press, 1978, pp. 3-28.

[Colby 75]. Colby, K., Artificial Paranoia. N.Y.: Pergamon Press. 1975.

[Date 81]. Date, C.J., An Introduction to Database Systems (3rd ed.). Reading, Mass.: Addison-Wesley. 1981.

[Dattola 77]. Dattola, R.T., "FIRST: An On-Line Document Retrieval System Accepting Natural English Queries." in Information Management In The 1980's: Processings of ASIS Annual Meeting, Vol. 14. 1977. Chicago: ASIS. p. 94.

[Dattola 79]. Dattola, R.T., "FIRST: Flexible Information Retrieval System for Text," JASIS, Vol. 30, No. 1. January 1979. pp. 9-14.

[Dolby 67]. Dolby, J.L. and Resnikoff, H.G., The English Word Speculum, Vol. I, II, III, IV, V. Mouton: The Hague. 1967.

[Dominick 77]. Dominick, W. D., "User Interaction Monitoring as a Multi-Level Process." in Information Management In The 1980's: Processings of ASIS Annual Meeting, Vol. 14. 1977. Chicago: ASIS. p. 63.

[Dominick 82]. Dominick, W.D., C.D. Michelson and M. U. Farooq, MADAM Users Guide, Vol. I. Lafayette: USL Computer Science Department, 1982.

[Dominick 80]. Dominick,W.D., MADAM Users Guide, Vol. II. Lafayette: USL Computer Science Department, 1980.

[Doszkocs 79]. Doszkocs, T.E., "Natural Language Searching in Online Bibliographic Retrieval Systems," 1979.

[Eason 75]. Eason, K.D., L. Damodaran and T.F.M. Steward, "Interface Problems in Man-Computer Interaction", in E. Mumford and Sackman(eds.), Human Choice and Computers. North-Holland Co., 1975, pp.91-105.

[Feigenbaum 74]. Feigenbaum, E.W., "Notes on Structured Programming", in O.J. Dahl, E.W. Dijkstra and C.A.R. Hoare, Structured Programming (eds.). N.Y.: Academic Press. 1974.

[Fillmore 68]. Fillmore, C., "The Case for Case," in Bach, E. and Harms, R. (eds.), Universals in Linguistic Theory. N.Y.: Holt, Rinehart and Winston. 1968. pp. 1-88.

[Fillmore 71]. Fillmore, C., "Some Problems for Case Grammar," in O'Brien, R.J. (ed.), Report of the Twenty-Second Annual Round Table Meeting on Linguistics and Language Studies. Washington, D.C.: Georgetown

University Press. 1971. pp. 35-56.

[Greenblatt 78]. Greenblatt, D. and J. Waxman, "A Study of Three Database Query Languages", in B. Shneiderman, Data Bases: Improving the Usability and Responsiveness. London: Academic Press, 1978, pp. 77-97.

[Harris 76]. Harris, L.R., "User Oriented Data Base Query with the ROBOT Natural Language Query System," in Third Int'l Conf. on Very Large Data Bases Proceedings. 1976. pp. 303-11.

[Harris 78]. Harris, L.R., "The ROBOT System: Natural Language Processing Applied to Data Base Query," ACM '78 Proceedings, Vol. 1. pp. 165-172. 1978.

[Heaps 78]. Heaps, H. S., Information Retrieval: Computational and Theoretical Aspects. N.Y.: Academic Press. 1978.

[Hendrix 77]. Hendrix, G.G., "Human Engineering for Applied Natural Language processing," in The Fifth Int'l Conf. on Artificial Intelligence. 1977. pp. 183-91.

[Hendrix 78]. Hendrix, G.G. and Lewis, W.H., "Transportable Natural Language Interfaces to Databases," Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics. June 1981. pp. 159-65.

[Hendrix 81]. Hendix, G.G. and J.G. Carbonell, "A Tutorial on Natural Language Processing," ACM '81, No. 1. pp. 4-8. 1981.

[Hoover 79]. Hoover, E.D., "A Methodology for Free Format Systems Design," Ph.D. Dissertation. Lafayette, LA: Univ. of S.W. LA, 1979.

[Jackson 76]. Jackson, P.C., Introduction to Artificial Intelligence. N.Y.: Petrocelli/Charter. 1976.

[Jones 79] Jones, K. Sparck, "Problems in the Representation of Meaning in Information Retrieval," in The Analysis of Meaning. M. MacCafferty and K. Gray (eds.), Aslib, London. 1979. pp. 193-201.

[Kaplan 78] Kaplan, J., "On the Difference Between Natural Language and High Level Query Languages," ACM '78 Proceedings. Vol. 1. 1978. pp. 27-38.

[Kaplan 83]. Kaplan, J., "Cooperative Responses from a Portable Natural Language Database Query System," in Brady, M. and Berwick, R.C., Computational Models of

Discourse. Cambridge, Mass.: The MIT Press. 1983. pp. 167-208.

[Kaplan 84]. Kaplan 78]. Kaplan, J., "Designing a Portable Natural Language Database Query System," ACM Transactions on Database Systems. Vol. 9, No. 1. March 1984. pp. 1-19.

[Kasarda 72]. Kasarda, A.J. and Hillamn, D.J., "The LEADERMART System and Service," ACM '72 Proceedings. 1972. pp. 469-77.

[Katz 54]. Katz, E. and Lazarsfeld, P., Personal Influence. Glencoe, Ill.: Free Press. 1954.

[Lindsay 63]. Lindsay, R.K., "Inferential Memory as the Basis of Machines Which Understand Natural Language," in Feigenbaum, E.A. and Feldman, J. (eds.) Computer and Thought. N.Y.: McGraw-Hill. 1963. pp. 217-33.

[Lockemann 75]. Lockemann, Peter C., "Data Base User Language for the Non-Programmer", University Karlsruhe: D-75 Karlsruhe 1, 1975, pp. 183-212.

[Martin 73]. Martin, James, Design of Man-Computer Dialogues. N.Y.: Prentice-Hall. 1973.

[Martin 82]. Martin, James. Application Development Without Programmers. Englewood Cliffs: Prentice-Hall. 1982.

[Martin 80]. Martin, T. "Information Retrieval," in Smith and Green (ed.) Human Interaction With Computers. N.Y.: Academic Press. pp. 161-76. 1980.

[Matthews 62]. Matthews, G.H., "Analysis by Synthesis of Natural Languages," Proc. 1961 Int'l Conf. on Machine Translation and Applied Language Analysis. London: Her Majesty's Stationery Office. 1962.

[McClure 81]. McClure, C.L., Managing Software Development and Maintenance. N.Y.: Von Nostrand Reinhold. 1981.

[McGill 76]. McGill, M.J., et. al., "Syracuse Information Retrieval Experiment (SIRE): Design of an On-Line Bibliographic Retrieval System," ACM SIGIR Forum, Vol. 11, No. 1. Summer 1976. pp. 37-44.

[McLeod 78]. McLeod, Dennis, A Semantic Data Base Model and Its Associated Structured User Interface. Cambridge: MIT Lab. for Computer Science. 1978.

[Meister 67]. Meister, D. and Sullivan, D.J., "Evaluation of User Reactions to a Prototype On-Line Information Retrieval System," NASA Contract Project No. NASw-1369. Canoga Park, Calif.: Bunker-Ramo Co., 1976.

[Merriam-Webster 74]. The Merriam-Webster Dictionary. N.Y.: G & C Merriam Co., 1974.

[Minker 77]. Minker, J. "Information Storage and Retrieval-A Survey and Functional Description," ACM-SIGIR FORUM, Vol. XII, No. 2. pp. XII-108. 1977.

[Moyne 77]. Moyne, J.A., "Simple-English for Database Communication," in Int'l Journal of Computer and Information Sciences, Vol. 6, No. 4. 1977. pp. 327-43.

[Montgomery 72]. Montgomery, C., "Linguistics and Information Science," Journal of the American Society for Information Science, Vol.23, No. 3. May-June 1972. pp. 195-219.

[MITRE 64]. English Preprocessor Manual. Report SR-132. Bedford, Mass.: MITRE Corp., 1964.

[Mylopoulos 76]. Mylopoulos, J., Cohen, B.P., Roussopoulos, N., Tsotsos, J. and Wong, H., "TORUS: A Step Towards Bridging the Gap Between Data Bases and the Casual User," Management Systems, Vol. 2. Pergamon Press. 1976. pp. 49-64.

[Nijessen 74]. Nijessen, G.M., "Data Structuring in the DDL and Relational Model," in Klimbie and Koffemann(eds.), Data Base Management. North-Holland Co., 1974. pp. 369 - 379.

[Petrick 65]. Petrick, S.R., A Recognition Procedure for Transformational Grammars (Ph.D. Thesis). MIT Department of Modern Languages. Cambridge, Mass., 1965.

[Quillian 68]. Quillian, M.R., "Semantic Memory," in Minsky, M. (ed.), Semantic Information Processing. Cambridge, Mass.: The MIT Press. 1968.

[Rich 83]. Rich, E., Artificial Intelligence. N.Y.: McGraw-Hill. 1983.

[Rowe 82]. Rowe, N., "On Some Arguable Claims in B. Shneiderman's Evaluation of Natural Language Interaction with Data Base Systems", in ACM SIGMOD Record, Vol. 13, No. 1. N.Y.: ACM. 1982. pp. 92-97.

[Salton 68]. Salton, G., Automatic Information Organization and Retrieval. N.Y.: McGraw-Hill. 1968.

[Salton 83]. Salton, G. and M.J. McGill, Introduction to Modern Information Retrieval. N.Y.: McGraw-Hill. 1983.

[Sandewall 71]. Sandewall, E.J., "Representing Natural Language Information in Predicate Calculus," in Meltzer, B. and Michie, M. (eds.), Machine Intelligence 6. Edinburgh, U.K.: Edinburgh University Press. 1971.

[Schank 73]. Schank, R.C., "Identification of Conceptualizations Underlying Natural Language," in Schank, R.C. and Colby, K.M. (eds.), Computer Models of Thought and Language. San Francisco: W.H. Freeman and Co., 1973. pp. 187-247.

[Schank 75]. Schank, R.C., Conceptual Information Processing. N.Y.: North-Holland. 1975.

[Schramm 54]. Schramm, W., "How Communication Works," in Schramm, W. (ed.), The Process and Effects of Mass Communication. Urbana: University of Illinois Press. 1954. pp. 3-26.

[Siklossy 78]. Siklossy, L., "Impertinent Question-Answering Systems: Justification and Theory," ACM '78 Proceedings, Vol. 1. 1978. pp. 39-44.

[Simmons 72]. Simmons, R.F. and Slocum, J., "Generating English Discourse from Semantic Networks," CACM, No. 15. 1972. pp. 891-905.

[Simmons 73]. Simmons, R.F., "Semantic Networks: Their Computation and Use for Understanding English Sentences," in Schank, R.C. and Colby, K.M. (eds.), Computer Models of Thought and Language. Hillsdale, N.J.: Lawence Erlbaum. 1973. pp. 63-113.

[Smith 80]. Smith, H.T., "Human-Computer Communication," in Smith and Green (eds.) Human Interaction With Computers. pp. 5-38. 1980.

[Smith & Green 80]. Smith, H.T. and T.R.G. Green,(eds.) Human Interaction With Computers. N.Y.: Academic Press. 1980.

[Sprague 80]. Sprague, R.H., Jr., "A Framework for the Development of Decision Support Systems", MIS Quarterly, December 1980. pp. 1-26.

[Tanenbaum 76]. Tanenbaum, A.S., Structured Computer Organization. Englewood Cliffs, N.J.: Prentice-Hall. 1976.

[Tsichritzis 77]. Tsichritzis, D.C. and Lochovsky, F.H., Data Base Management Systems. N.Y.: Academic Press. 1977.

[Ullman 82]. Ullman, J.D., Principles of Data Base Systems. Rockville, Md.: Computer Science Press.

[Waltz 77]. Waltz, D.L. and B.A. Goodman, "Writing a Natural Language Data Base System", in 5th International Joint Conference On Artificial Intelligence -1977. MIT, 1977, pp. 144-50.

[Wanger 76]. Wanger, J., C.A. Cuadra and M. Fishburn, Impact of On-Line Retrieval Service: A Survey of Users, 1974-75. Santa Monica, Calf.: System Development Corp., 1976. pp. 193 - 206.

[Weizenbaum 66]. Weizenbaum, J., "ELIZA -- A Computer program for the Study of Natural Language Communication Between Man and Machine," CACM, No. 9. pp. 36-45.

[Wiederhold 83]. Wiederhold, Gio, Data Base Design(2nd ed.), N.Y.: McGraw-Hill, 1983.

[Wilks 76]. Wilks, Y.A., "Processing Case," American Journal of Computational Linguistics. Microfiche 56.

[Williams 69]. Williams, J.H., BROWER, An Automatic Indexing On-Line Text Retrieval System. IBM. 1969.

[Winograd 72]. Winograd, T., Understanding Natural Language. N.Y.: Academic Press. 1972.

[Winograd 73]. Winograd, T., "A Procedural Model of Language Understanding," in Schank, R.C. and Colby, K.M. Computer Models of Thought and Language. San Francisco: W.H. Freeman and Co., 1973. pp. 152-86.

[Winograd 83]. Winograd, T., "Language as a Cognitive Process", Reading, MA.: Addison-Wesley Co., 1983. pp. 195-272.

[Winston 79]. Winston, P.H. and R.W. Brown, Artificial Intelligence: An MIT Perspective, Vol. 1, Cambridge: MIT Press, 1979.

[Woods 67]. Woods, W.A., Semantics for a Question-Answering System (Ph.D. Thesis), Division of Engineering and Applied Physics. Cambridge, Mass.: Harvard University. August 1967.

[Woods 68]. Woods, W.A., "Procedural Semantics for a Question-Answering Machine," AFIPS Conf. Proceedings,

Vol. 3, Part One. 1968. pp. 457-71.

[Woods 69]. Woods, W.A., Augmented Transition Network for Natural Language Analysis. Cambridge, Mass.: The Aiken Computation Lab., Harvard University. December 1969. NSF Report No. CS-1.

[Woods 72]. Woods, W.A., Kaplan, R. and Nash-Webber, B., "The Lunar Sciences Natural Language Information System: Final Report," BBN Report, No. 2378. Cambridge, Mass.: Bolt Beranek and Newman, Inc., 1972.

[Zloof 78]. Zloof, Moshe M., "Design Aspects of the Query-By-Example Data Base Management Language", in B. Shneiderman(ed.), Data Bases: Improving Usability and Responsiveness. London: Academic Press. 1978. pp. 29-53.

Liu, I-Hsiung, B.A., Fu-Jen Catholic University, 1974
M.A., National Chen-Chi University, 1976
Master of Science, Spring 1985
Major: Computer Science
Title of Thesis: Nautral Language Query System Design for Interactive Information Storage and Retrieval Systems
Thesis Directed by Professor Wayne D. Dominick
Page in Thesis, 164; Words in Abstract, 88

ABSTRACT

The currently developed multi-level language interfaces of information systems are generally designed for experienced users. These interfaces commonly ignore the nature and needs of the largest user group, namely, casual users. This research identifies the importance of natural language query system research within information storage and retrieval system development; addresses the topics of developing such a query system; and finally, proposes a framework for the development of natural language query systems in order to facilitate the communication between casual users and information storage and retrieval systems.

BIOGRAPHICAL SKETCH

I-Hsiung Liu was born in [redacted] [redacted] on [redacted] [redacted]. He received his B.A. in Mass Communication from Fu-Jen Catholic University (Taipei, Taiwan) in June 1974, and his M.A. in Journalism from National Chen-Chi University (Taipei, Taiwan) in June 1976. He attended the University of Southwestern Louisiana from January 1982 to January 1985, graduating with a Master of Science degree in Computer Science. Upon graduation, he joined Racal Milgo Information Systems in Ft. Lauderdale, Florida, and is currently working in their Communication Network Division as a software engineer.

4.18

1. Report No. IN-82	2. Government Accession No. 183563 ~~447390~~	3. Recipient's Catalog No.
4. Title and Subtitle USL/NGT-19-010-900: NATURAL LANGUAGE QUERY SYSTEM DESIGN FOR INTERACTIVE INFORMATION STORAGE AND RETRIEVAL SYSTEMS		5. Report Date April 22, 1985
		6. Performing Organization Code
7. Author(s) I-HSIUNG LIU		8. Performing Organization Report No.
9. Performing Organization Name and Address University of Southwestern Louisiana The Center for Advanced Computer Studies P.O. Box 44330 Lafayette, LA 70504-4330		10. Work Unit No.
		11. Contract or Grant No. NGT-19-010-900
12. Sponsoring Agency Name and Address		13. Type of Report and Period Covered FINAL; 07/01/85 - 12/31/87
		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The currently developed multi-level language interfaces of information systems are generally designed for experienced users. These interfaces commonly ignore the nature and needs of the largest user group, namely, casual users. This research identifies the importance of natural language query system research within information storage and retrieval system development; addresses the topics of developing such a query system; and finally, proposes a framework for the development of natural language query systems in order to facilitate the communication between casual users and information storage and retrieval systems.

This report represents one of the 72 attachment reports to the University of Southwestern Louisiana's Final Report on NASA Grant NGT-19-010-900. Accordingly, appropriate care should be taken in using this report out of the context of the full Final Report.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Natural Language Query System Design, Information Storage and Retrieval Systems	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	164	

*For sale by the National Technical Information Service, Springfield, Virginia 22161