A NATURAL LANGUAGE INTERFACE FOR AN INTELLIGENT DOCUMENT INFORMATION AND RETRIEVAL SYSTEM

BY

G. JAN WILMS

B.A., Katholieke Universiteit Leuven, 1984

M.A., University of Mississippi, 1985

A Thesis

Submitted to the Faculty of

The University of Mississippi

in Partial Fulfillment of the Requirements

for the degree of Master of Science in Engineering Science

in the Department of Computer and Information Science

The University of Mississippi

May, 1988

A NATURAL LANGUAGE INTERFACE FOR AN

INTELLIGENT DOCUMENT INFORMATION AND RETRIEVAL SYSTEM

BY

G. JAN WILMS

Associate Professor of
Computer and Information
Science
(Director of the Thesis)

Associate Professor of
Computer and Information
Science
Chairman of the Department

Associate Professor of
Computer and Information
Science

Professor of Health Care
Administration

ACKNOWLEDGEMENTS

I wish to express my gratitude to those people who have helped to make this thesis a reality.

To my director, Dr. Steven B. Schoenly, who guided my work with patience, understanding, and skill; He was instrumental in my decision to study computer science after I received my Master's Degree in English.

To the other members of my committee, Professor Mickey Smith, Dr. Tobin Maginnis and Dr. Robert Roggio, for their helpful suggestions and generous devotion of time to my problems.

A graduate assistantship and instructorship from the University of Mississippi have made it financially possible for me to perform the research and to write the thesis.

This thesis is dedicated to my wife Wallika. I thank her for her encouragement and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER 1

INTRODUCTION

Information Retrieval (IR) is a technology of representing, storing, organizing and accessing information items [SALTON 83a, 1]. These items consist of the complete or partial natural language text of documents. To satisfy a user's information needs, IR provides a set of relevant references, which the user can consult to find an answer to their question. Hence an IR system is more restricted than a Question Answering System, which can extract meaning from the document collection, and directly ''answer'' the user's problem.

This technology has existed long before the advent of computers. Librarians have used manual mechanisms like call numbers and subject catalogs to impose a physical and logical order on the wealth of information that is stored in the library. However, without more sophisticated technologies, existing retrieval systems seem always to be taxed by the explosion of information, making it increasingly difficult for users to get current and complete information.

While computer-based IR cannot solve the problem of information overload, it has much better tools to deal with it. Like their manual counterparts, computer-based IR systems store and represent part of or whole documents, and output a set of references in response to a user query. Through time sharing access many searches can be conducted at the same time, often involving a long list of terms in complex relationships. Computers are also much more economical in offering

multiple access points to documents. But the main advantage may well be that once the data base is captured in machine readable form, it is easy to duplicate and share among different computing centers.

Unlike data base management systems (another kind of automated information system that deals with tabular data elements), IR systems must be able to process approximate queries. This means that they must attempt to retrieve relevant items even if the user cannot formulate the query exactly. To achieve this purpose, most IR systems employ a controlled vocabulary that is used in the indexing and query negotiating process. This controlled vocabulary implies, however, that trained intermediaries are necessary to help formulate query statements. The casual user may also have problems learning the subtleties of Boolean logic, which is used by most commercial systems to combine search terms into a query statement.

Much experimental research has been done on how to improve the performance of IR systems. While most of it has focused on finding alternative ways of representing and organizing documents (other than by using the classical inverted file), some interesting applications have come from the area of artificial intelligence. Attempts have been made to design ‘‘intelligent’’ information retrieval systems (IIRS) that enhance retrieval efficiency by using natural language processing to allow free-language query submissions and automatic indexing applications.

THESIS OBJECTIVE AND METHODOLOGY

A project has been undertaken, in cooperation with the Department of Health Care Administration, to develop a prototype of an IIRS. The retrieval system emulates a

human information specialist, and assists casual users with their information needs.

The system uses ‘‘determinants of medication’’ as an application field, which is a well defined small interdisciplinary area dealing with the manufacturing, marketing, and consumption of medication. A document collection has been assembled with over three thousand items that are related to the broader domain of pharmacy and medicine. The purpose of the project is to develop an operational information retrieval system, and an experimental software testbed for researching how IR can be enhanced by AI. Consequently the system employs a very modular and flexible design.

The object of this thesis is to design and implement an interface component to this system, which employs natural language techniques to eliminate the need for a formal query syntax. The user interface acts as the scheduler of the retrieval system, and initiates the query process. The front-end allows the user to volunteer their information needs in English and maps the input to a set of frames representing likely query structures. The interface next rephrases the request to display its understanding, and interactively negotiates with the user to fill the missing slots in the frame, which is used to communicate with the processing component.

The natural language interface catches obvious misspellings and employs fuzzy logic techniques to automatically translate user specifications like ‘‘very’’, ‘‘especially,’’ or ‘‘not’’ into weights. The interface also employs a transparent synonym lookup to improve category matching.

As the processing component returns the id numbers of retrieved documents, the interface unit again takes over and interacts with the user to display the references. A history is kept of past queries, which can be referenced by the user to renegotiate

the request. If necessary the interface offers assistance to further reduce this document set (i.e. by committing the user to more specific categorization) or to enlarge the returned group. A history of queries is kept during the session and can be referred to by the user to combine with later queries.

The prototype has been implemented using Turbo Pascal on an AT compatible microcomputer. A hard disk is used to permanently store the document collection and the set of inverted files used by the retrieval component. The IIRS uses a dictionary of 80,000 words (Random House Dictionary) to assist in lemmatizing and spelling checks. The natural language interface has been built separately and independently from the user interface component, but because of the modular design and the common data structures, both units have been compiled into one standalone program. The prototype has been tested using a subset of the document collection and a list of sample queries submitted by an expert in the application domain area.

Chapter two gives an overview of current research in information retrieval, and discusses the potential advantages of combining standard retrieval methods with techniques employed by artificial intelligence. Natural language processing is mentioned in particular, and how it can be of use in an IIRS. Chapter three describes the architecture of the project, and the function of the interface component as part of the system. The next chapter portrays possible techniques used in natural language processing, and a detailed discussion of the actual implementation of this component in the proposed IIRS. The conclusion summarizes the design considerations for the IIRS prototype, and suggests possible further enhancements. The appendix, finally, illustrates a few typical interactions with the system.

CHAPTER 2

INTELLIGENT INFORMATION RETRIEVAL SYSTEMS

INTRODUCTION.

In the early 60s, H.P. Luhn introduced statistical approaches in the analysis and retrieval of documents to increase the efficiency and performance of retrieval systems [LUHN 57]. Since then his technique has been perfected and implemented in commercial retrieval systems, but some fundamental issues still go unanswered. As one reviewer put it,

> We [still] do not know the best way of representing the content of text documents and the users' information needs so that they can be compared and the relevant documents retrieved. We cannot even agree on a definition of relevance [CROFT 87, 249].

Besides this general feeling of dissatisfaction with the current state of affairs, there is a proliferation of computer systems that deal with text and on-line documents, which has lead to an increasing awareness of the importance of IR as an application area.

INTELLIGENT INFORMATION RETRIEVAL SYSTEMS

As a result, there has been a recent shift away from traditional retrieval systems and the statistical approach towards something called ''Intelligent Information Retrieval Systems'' (IIRS). Sparck Jones defined such a system in 1983 as a user's probably ill-defined request to a set of relevant documents [SPARCK 83, 136]. In other words, an IIRS is a system that carries out intelligent retrieval. Using stored knowledge about its documents, the users, and usage patterns, an IIRS infers which documents will help the users satisfy their information needs.

This new concept of an intelligent system shifts much of the attention to the user interface. Most users are unable to specify exactly the information they need, since this involves describing the very thing they do not know. Consequently the computer-human interaction becomes very important in formulating a query, and the need for unrestricted natural language queries becomes evident. A related feature is the dynamic utilization of user feedback in automatic search query reformulation.

A second characteristic of an IIRS is the emphasis on using an inferential process to link queries and users. Van Rijsbergen defines retrieval as a process based on logic, and views the matching function between query and documents as a plausible inference [VAN RIJSBERGEN 86, 195]; thus the retrieval process can be seen as an uncertain implication between a document collection D and a request R (D -> R). In order to do this the system must ‘‘know’’ about its task, and incorporate knowledge of the document collection, of the subject domain, and of the search topic.

The realization of the importance of the interface and of stored knowledge has led to the use of methods and techniques from Artificial Intelligence (AI) in IR. AI systems are systems that emulate human cognitive skills, and have traditionally been concerned with knowledge representation, declarative reasoning (in Expert Systems), and human interaction (through Natural Language Processing). The overlap between AI and IR has been approached from two angles: AI researchers using IR as an application area, and IR investigators blending traditional retrieval techniques with methods developed in AI research.

Expert Systems

One area of AI that may be beneficial is expert systems. These are knowledge based

systems that are considered intelligent because they act in such a way that a human behaving in the same way is considered to be intelligent. Thus, it may be possible to develop an expert intermediary system that assists users with their query formulation, selection of search strategy, and retrieval evaluation.

> I believe that expert systems research is the new frontier and the next area of development in library information science. Expert systems will enable users to make more effective use of the automated systems and on-line databases that were designed and implemented during the past decade, and they will help the libraries to be more productive and efficient in carrying out the many tasks involved in managing an information service center [BORKO 87, 83].

Expert systems work by observing human experts (like trained librarians) and deriving a set of rules and facts based on their expertise, which can guide the casual user and automatically refine his query. The system consists of a knowledge base and a set of facts and rules to traverse the search space.

Unfortunately, as several researchers point out, the success of this approach is disappointing, and the text processing context may not fit the proposed methodology. IR does not appear to be an ideal application domain for expert system development. In order to do any kind of intelligent problem solving of real-world tasks, expert systems require highly specialized domain knowledge and, hence, are restricted to narrow specialist domain areas like diagnosing pulmonary disorders (PUFF) or configuring VAX 11 series computers (XCON). IR, however, is not narrow, nor is it homogeneous or well bounded. There are no obvious human experts, and there is no consensus on the best search technique. One researcher summarizes the situation as follows:

> Human intermediaries have no control over the retrieval algorithms employed by present systems and therefore treat them as a given, designing strategies to make use of their potentials and minimize their drawbacks [BROOKS 87, 375].

Search terms are often ambiguous and have unclear relationships. Rules in

information retrieval are not transparent and have consequents that do not follow unequivocally from the antecedents. Moreover, there is much evidence that the traversal of a hierarchically structured knowledge base shows little resemblance to the actual search strategy used by human experts.

INTERFACE

An important feature of an IIRS, and another major area of overlap between IR and AI, is a flexible and convenient interface which allows powerful interaction between user and retrieval mechanism.

With the advent of interactive computer terminals in the early 1970s, it was expected that the on-line revolution would now involve the end-user directly in the automatic search and retrieval process. This hope was not realized, however [SALTON 87, 3]; many competing retrieval systems were developed (DIALOG, STAIRS, MEDLARS, etc.), each with different access mechanisms and control languages that are incompatible. The standard Boolean search mechanism used by these systems is confusing to the average user because they are untrained in using the operators in their strict logical sense. Users may believe, for example, that ‘‘X AND Y’’ will retrieve more documents than ‘‘X’’ alone and, may forget that ‘‘X OR Y’’ will not perform any ‘‘ranking,’’ and, hence, they will rate documents with one term equal to documents containing both items.

Because of these user-hostile systems, casual users again have to rely on trained search intermediaries to guide them in their query. The real query is in the user’s mind, and most of the understanding process in traditional systems happens off line through interaction with expert searchers. These intermediaries must assume the full burden of understanding the user’s information need, and coming up with the

‘‘right’’ search strategy and query formulation. ‘‘Existing IR systems are basically passive, `dumb’ systems searched by dynamic, `intelligent’ human searchers’’ [DOSZKOCS 86, 193].

Several operational systems have tried to overcome this complexity by using a simplified command language and yes/no menus and scripts. The emergence of user friendly interfaces and gateway systems include such automatic features as dialup and logon procedures, saving search statements, and assistance through help screens and on line tutorials. But all these fixes still offer very little assistance in the formulation of the query.

> The heightened subject diversity and text volume present more vocabulary switching problems and create a need for more advanced NLP capabilities. To meet this need, new releases and versions of operational IR systems ... exhibit even more powerful and versatile functions in the area of proximity searching, automatic multifile query transformation, and multilevel user interfaces, but they still shy away from formal language analysis techniques and tools. [DOSZKOCS 86, 195]

Natural Language Processing

The proposed approach for an IIRS is to allow the user to enter his query using natural language, and to build an interface that will analyze the request and map it to a structure that the retrieval component can process. Many researchers agree to the superiority of using ‘‘high-level query languages which reflect the overall user’s intent rather than the computer operations that may be required to obtain any particular result’’ [SALTON 83a, 258].

Natural language processing (NLP) can play a role in both the retrieval and storage of documents. It can be used to build a friendly user interface that allows free language query submission and hence eliminates the need for mastering a formal query format. For information storage, it can be used to structure the document

database and perform automatic indexing. In addition, NLP can help construct synonym dictionaries and thesauri.

Some researchers have attempted the automatic formulation of Boolean queries from natural language without the use of linguistic input or grammatical theory [SALTON 83b]. Instead of semantic criteria, they use probabilistic tests; the algorithm translates a request by using the estimated number of documents retrievable by a term or term combination until a predetermined threshold has been reached. The estimation is based on the sum of posted document frequencies of individual terms or term combinations. The output has been tested on the MEDLARS system and is clearly competitive with conventional manual Boolean formulations, and even superior when good natural language statements of user need are available.

NLP consists of several levels of conventional processing, some more relevant to IIRS than others. The phonological level analyzes speech sounds, and is of minor importance in IR, except maybe for generating sound alike words in approximate name matching (e.g. if the author's name is misspelled). The morphological layer performs operations on word stems, and can be used to remove suffixes, allow truncation operations, and permit browsing through alphabetically arranged entries (e.g. ELHILL's ''neighbor'' command). Lexical operations include full word processing, and are useful in stopword deletion, spell checking, automatic search key substitution or augmentation, or handling of abbreviations and acronyms through thesauri. The next higher level is syntactic identification of structural units like noun and verb phrases. Sophisticated automatic parsers have been developed, but are rarely used in operational systems, except occasionally in free text searching

(adjacency and pattern matching) and search restriction to certain boundaries (e.g. to Title or Author field). Finally, there is the semantic category which uses contextual knowledge to represent meaning. Here no formal methods exist and there is currently little use for it in existing IRS. The nearest approximation to semantic analysis would be the display of classification schemes like ELHILL's ''tree'' and ''explode'' commands. There is one higher ''pragmatic'' level in NLP which uses knowledge about real-life objects to make meaning unambiguous, but this is experimental at best, even in AI research.

The use of linguistic methods to analyze natural language input of user queries has both opponents and proponents. The latter believe that the analysis of meaning improves the retrieval process, and they consider IR an early stage of more refined question answering [GARDIN 79]. Opponents doubt the usefulness because of the difference between IR and other areas of language processing. Like the critics of the expert system approach to IR, they value much more the use of statistical, probabilistic or vector space techniques [ROBERTSON 79].

> The whole idea of meaning representation is dubiously relevant to document retrieval in anything like its present form. ... One can get quite good results with simple terms and weights [SPARCK 79, 200].

CONCLUSION

This study sides with the researchers who believe that well established linguistic procedures do contribute to retrieval effectiveness. It is true that unrestricted natural language input still poses formidable problems because of inherent semantic ambiguity and absence of a general theory of speech acts and dialog pragmatics. However, an interface based on key word matching and fuzzy set techniques is proposed, which is able to handle relatively unconstrained natural language queries and thus eliminate the need for mastering a formal query syntax.

CHAPTER 3

ARCHITECTURE

The system consists of two integrated components (interface and processor), each embodying some AI techniques to enhance the recall and precision of the retrieval process. The system architecture, which is based on a similar study by BISWAS et al., and it is depicted in Figure 1.

The architecture of the proposed IIRS reflects a knowledge-based system approach; there is a clear separation between the domain specific knowledge source (topical area of medication) and the inference mechanism which does the retrieval and ranking.

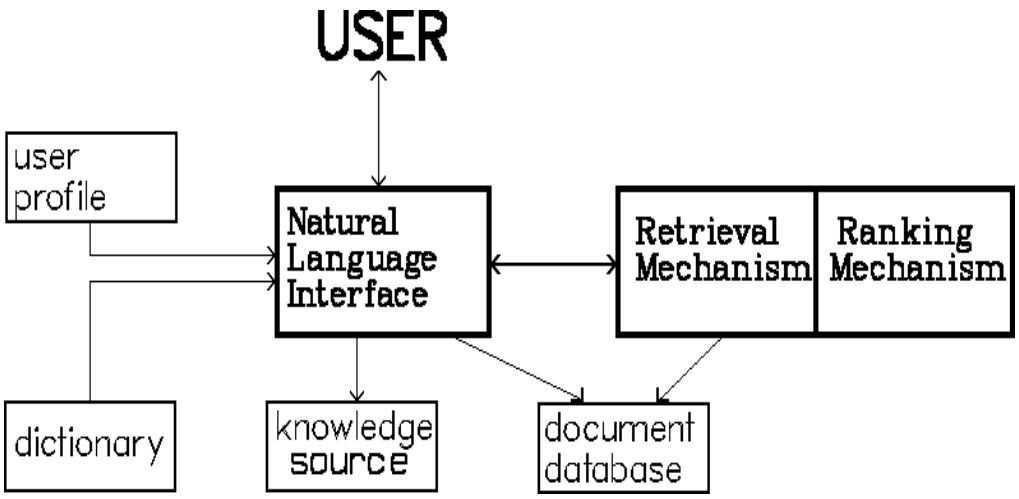

Figure 1: Architecture

$ID 740001
$TI Effects of a Medicaid Program on Prescription Drug Availability and Acquisition
$AU Smith, Mickey C.; Garner, Dewey D.
$SO Medical Case, July 1974, Vol. 12, No. 7, pp. 571-581
$YR 1974
$CC 101.01; 108.28.08; 108.28; 108.08; 108.04; 108.24.08; 108.28.16; 108.24.04; 108.24.12; 108.40.28; 108.56; 108.68.20; 108.28.24; 301.02; 302.08; 601; 603.05; 801.01; 801.07
$AB A study was conducted comparing prescription drug utilization by a sample of patients in similar three-month periods before and after enactment of a Medicaid drug program. Prescription audits and personal interviews were used. After necessary adjustments, total utilization was shown to increase from 5.43 prescriptions per patient pre-Medicaid to 9.48 prescriptions post-Medicaid. The number of different drugs used per patient also increased. The average quantity prescribed increased slightly as did the average cost per dose dispensed. Some changes in prescribing habits might be indicated since the nature of program economics make old habits more costly.

Figure 2: Sample Document

DOCUMENT DATA BASE

Each document in the document data base consists of a template with 7 fields, separated by ‘‘$’’ symbols, as shown in Figure 2. Each document is stored in a separate file to facilitate displaying its content after the relevant references have been ranked and returned to the interface component. Note that the $ID field is a unique number that starts with the year of publication. This same $ID serves as the name of the file that holds the document. Except for the $ID field, all other fields have also been gathered into six inverted files that are used by the inference component in the retrieval process.

KNOWLEDGE SOURCE

The knowledge source with its domain specific terms and relations, is completely separated from the inference mechanism, so the IIRS can easily be used with new domains. It is partitioned into three components: a concept knowledge source (for fields $ABSTRACT and $TITLE), a chronology base (for $YEAR), and a name base (for $AUTHOR and $SOURCE). Each knowledge source entry consists of a list of terms that represent meaningful entities and a set of operators to combine these concepts.

The concept knowledge source consists of a thesaurus of terms related with synonym operators (e.g., ‘‘antacids’’ = ‘‘adsorbents’’) and through the implication relation (e.g., aminoglycosides’’ -> ‘‘antibiotics’’ -> ‘‘anti-infective agents’’). A small sample of the concept knowledge source is listed in Figure 3. This thesaurus has been built by an expert in the field of medication.

The synonym operator allows for more flexibility in the original formulation of the query since the user can use familiar terms, and the system will enhance the query automatically (non obtrusively) as a specialized librarian would do. It does this by matching the search terms against entries in the thesaurus and, if a hit is found, by adding the listed synonyms to the query string.

The implication relation assists the user in reformulating the query if it is unsuccessful. If too few documents were retrieved, then replacing some search terms by a more general category (as indicated by the implication relation) may increase the number of relevant documents found. Conversely, if too many documents were retrieved, then specifying narrower terms will very likely reduce the number of documents relevant to that query. The inference component also aids in identifying

ANALGESIC = ANTIPYRETIC
ANTACID = ADSORBENT
ANTIMUSCARINICS = ANTISPASMODIC
ANTIPRURITICS = LOCAL AND ANESTHETIC
ANXIOLYTIC = SEDATIVE = HYPNOTIC
BRONCHIOLITIS = ACUTE BRONCHITIS
CATHARTIC = LAXATIVE
COAGULATION = BLOOD FORMATION
DDD = DEFINE DAILY DOSE
DEPIGMENTING = PIGMENTING AGENT
ELECTROLYTIC = CALORIC = WATER BALANCE
EMOLLIENT = DEMULCENT = PROTECTANTS
OSTEOPATHY = CHONDROPATHIES = ACQUIRED MUSCULOSKELETAL DEFORMITY
RESPIRATORY = CEREBRAL STIMULANTS
SCABICIDES = PEDICULICIDES
SERUM = TOXOIDS = VACCINE
THROMBOSIS = ARTERIAL EMBOLISM

AMINOGLYCOSIDES -> ANTIBIOTICS -> ANTI-INFECTIVE AGENT
ANTIMUSCARINICS -> ANTICHOLINERGIC AGENT -> AUTONOMIC DRUG
OPIATE AGONISTS -> ANALGESICS -> CENTRAL NERVOUS SYSTEM AGENT
HYDANTOINS -> ANTICONVULSANT -> CENTRAL NERVOUS SYSTEM AGENT

Figure 3: Sample Concept Knowledge Source

which term needs to be more specific / generic by returning a field weight, which indicates how many documents were retrieved by each field (see below). Since all the fields are ANDed together, a low field weight is responsible for the small size of the set of retrieved documents. A series of rules and facts can be used to diagnose the cause of user's dissatisfaction, and to suggest possible remedies. E.g.,

IF (SATISFACTION = low) AND (DOC_RETRIEVED = low) AND FIELD(X)
AND FIELDWEIGHT(X,Y) AND MINIMUM(Y) THEN INVESTIGATE(X)

The chronology base also contains synonyms (''after'' = ''beyond'' = ''past'' = ''since''), and establishes concrete values for fuzzy specifications (''recent'' = after 1986) (see Figure 4). Many of these concrete values are dynamic, and depend on

the current year (recent means different things in 1987 than in 1989) and on the oldest document in the collection (if the oldest document was published in 1957 or in 1976 ‘‘earliest papers’’ takes on quite a different meaning). It may even mean different things to different users (i.e., while ‘‘recent’’ means ‘‘the last two years’’ for one researcher, it may mean ‘‘the last two months’’ for another. The value of ‘‘now’’ (as in ‘‘all papers from 84 till now’’) also depends on the current year, of course. It may even be possible to retrieve ‘‘new’’ documents, if the system keeps track of updates to the document collection since the last interaction with the IIRS. When intensifiers are used in combination with fuzzy specifications (e.g., ‘‘very recent’’), the interface uses a dynamic weighting scheme (e.g., 1986 (0.6) 1987 (0.8) 1988 (1.0)) (See Chapter Four).

Finally there is the name base, which resolves abbreviation and acronym ambiguities, and deals with misspelled names through approximate name matching. The International List of Periodical Word Abbreviations indicates the correct abbreviation of the journals in the document base. For the particular subject area of medication

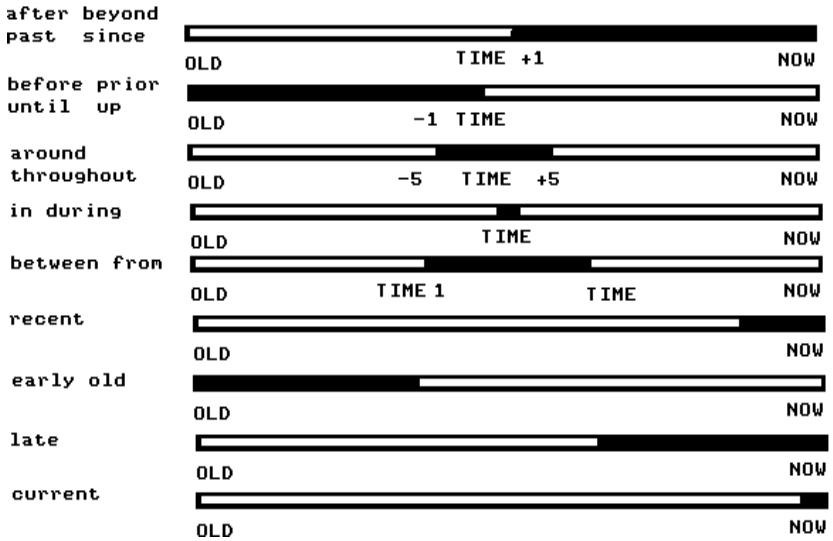

Figure 4: Chronology Base

there also exists a coding scheme which assigns a unique value to each journal, consisting of first letters that make up the name of the journal. This can be used in the $SOURCE field when the user specifies a particular magazine, since it allows using a matching technique without having to check every word of the journal name (see Figure Five for an example of this coding scheme).

Natural Language Interface

The natural language interface uses a mirror image of the frame to analyze the user query (see Figure Six). This image of the user's information need can be conceived as an ideal hypothetical document, perfectly relevant, or as an idealized representation of the items the user wants to retrieve. The input is partitioned into the different

ACFRAO	Agrupacion de Cooperativas Farmaceuticas ACOFAR Agrup. Coop. Farm.
CTOIDG	Cosmetics and Toiletries Cosmet. Toiletries Am. Cosmet. Perfum.
DDIPD8	Drug Development and Industrial Pharmacy Drug Dev. Ind. Pharm. Drug Dev. Commun.
PHMKBZ	Pharmacon Tijdschr. Belg. Hosp. Apoth.
SFPLA3	Studies in Family Planning Stud. Fam. Plann. Curr. Pub. Pop. Fam. Plann.
YHHPAL	Yao Hsueh Hsueh Pao Acta Pharm. Sinica

FIgure 5: Sample Name Base

fields ($ABSTRACT, $AUTHOR, $SOURCE, $TITLE). The $TITLE field will receive the same search terms as the $ABSTRACT field, since the title is often descriptive of the topic of the article. The $CC (Category Code) field is ordinarily transparent to the user, and it is used to automatically enhance the query through thesaurus operations (see above). The $ID field will remain blank, of course; this is how the inference mechanism returns a ranked list of document numbers. After the initial query, the system negotiates with the user to fill in the missing fields. This interaction continues until a certain threshold is reached that indicates enough data is available, or until all fields are filled. A set of expert rules control this interaction process. For example:

```
IF USER(X) AND EXPERT(X) AND SPECIFIED($ABSTRACT) AND
(LENGTH($ABSTRACT) > MIN) THEN PROMPT_MORE(false).
```

Then the frame is sent to the inference component and control is turned over to it.

On-line Dictionary

The natural language processing component employs a dictionary of 80,000 words

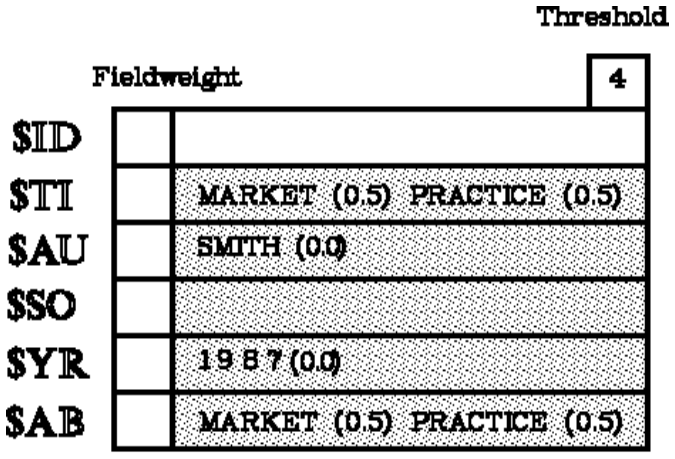

Figure 6: Frame 1, Idealized Document

(Random House Dictionary) to assist in lemmatizing and spell checking search terms (see Chapter Four). This commercial dictionary was chosen for the prototype because of its availability, because it is a BORLAND product, and hence is programmable from within Turbo Pascal, another product from that company.

User Profile

The IIRS keeps a user profile which retains information concerning the user's interaction with the system. Through profiles an IIRS can demonstrate machine learning, and be more sensitive to the user's specification need and manner of query formulation. Profiles delimit the portion of the document space normally searched in response to a query, and they influence the query negotiation process. The use of profiles is especially viable in systems with relatively fixed groups of users, with stable individual interests, but breadth of interest within each group [KORFHAGE 82]. Since the prototype is developed for a document collection of interdisciplinary determinants of drug use (sociological, medical, economical, etc.), it is not uneconomical to maintain a separate database for each user. Expert users, for example, will probably have a relatively complete initial query, so the query negotiation to fill blank fields will rarely be necessary. Hence the profile can be used to dynamically set the template threshold. User topology (expert or beginner) can usually be inferred from the queries themselves (e.g., generality or specifity of concepts). Other uses of the profile include keeping a list of technical words not recognized by the dictionary (see Chapter Four), and to define user preferences like output format, or if ''recent'' means ''last four years'' or ''last 4 months.'' The profile will also keep a dated record of past interactions with the system, so old queries can be referenced by the user and combined into longer queries. If the last date of interaction is recorded, then the system could even report documents that were added to the

document base since that time and that are relevant to past queries.

Retrieval and Ranking Component

The inference mechanism uses a standard inverted file and Boolean operators to retrieve a set of relevant documents, and it ranks them by means of weighted queries and documents. There are six inverted files corresponding to the six fields of the frame. The retrieval component uses a logical OR on the search terms **within** each field, and then it ANDs the fields together. ANDing fields guarantees that when the user specifies a particular year and a topic, documents relevant to the topic, but outside the specified time period, are NOT retrieved. The number of documents retrieved by each field is reported in the field-weight slot of the frame. The ranked list of documents is returned to the interface component via the $ID field of the frame, as shown in Figure 7. It is the responsibility of the interface component to display the references to the user. Hence the interface can be considered the scheduler of the system.

In addition to being ranked by weight, the documents are released to users one by one, each time accompanied by a request for relevance feedback to enable on-the-fly computation of document similarity. Relevance feedback was introduced by Salton in his SMART system [SALTON 83a]. If the user expresses dissatisfaction with a reference (as being not relevant), then the interface component will skip over the next entry if it is very similar on the assumption it will also prove irrelevant. Document similarity can be computed by considering each document as a vector of search terms. Several formulas exist to compute an index of similarity, the most famous index perhaps being the cosine similarity function. It is a ‘‘measure of the angle between two t-dimensional object vectors when the vectors are considered as ordinary vectors in a space of t dimensions’’ [SALTON 83a, 203]:

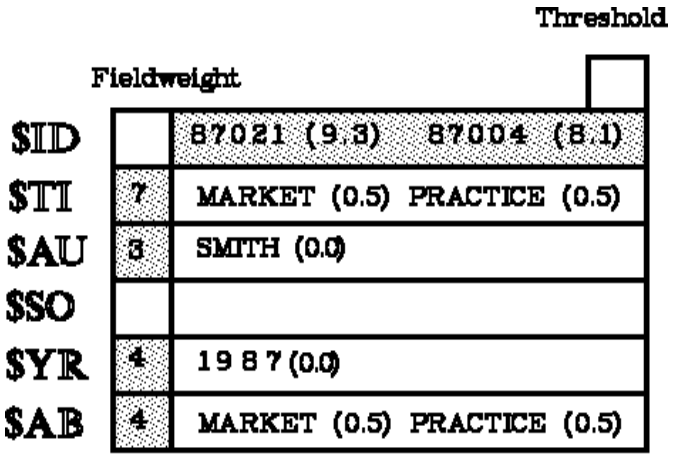

Figure 7: Frame 2, Documents Retrieved

After users have marked the entries they want to print out, they can control the output through a menu, which allows the queries to be output in rank order, chronologically, or sorted by a particular field. This may help answering questions like ‘‘which authors have the most publications in the last 10 years on medication use?’’ See Chapter Four for more information on this feature.

CONCLUSION

The IIRS prototype consists of a retrieval mechanism and a natural language front-end that interacts with the user to obtain the query statement and display the retrieved references. Chapter four discusses the natural language processing performed by the interface component to analyze the user’s request and transform it to an intermediary form used by the retrieval component.

$$SIM(DOC_i, DOC_j) := \frac{\sum_{k=1}^{t} (TERM_{ik} \bullet TERM_{jk})}{\sqrt{\sum_{k=1}^{t} (TERM_{ik})^2 \bullet \sum_{k=1}^{t} (TERM_{jk})^2}}$$

CHAPTER 4

IMPLEMENTATION

Design Considerations

Instead of a ‘‘crisp’’ query language, which requires rigid formats in terms of precise attributes and values, the IIRS recognizes imprecise natural language queries. To develop a sufficiently robust and friendly interface, it is unavoidable that some ad hoc restriction in permissible input be imposed.

> A pragmatic approach to designing NL processors would involve tailoring them to interpret statements and terms only in the context and domain under consideration. This reduces language ambiguities and makes the implementation task easier [BISWAS 87, 87].

In the prototype system under development, only a single imperative sentence is accepted, but this represents the most common query form. A typical example is ‘‘Give me very recent survey articles by M. Smith on the effects of advertising on prescribing.’’ Because keyword matching is used instead of strict syntactical parsing (see below), this restriction is rather flexible, and the IIRS is often able to process ungrammatical queries, ellipses, and simple questions (E.g., ‘‘Has anyone published on prescribing patterns of urologists?’’).

A second restriction is given by the very nature of Information Retrieval; unlike Question Answering Systems (QAS) which actually ‘‘answer’’ a problem by analyzing the ‘‘meaning’’ of the query and the stored document collection, IR returns a set of presumably relevant references which may contain the answer. Hence an IRS, even an ‘‘intelligent’’ IRS, is unable to answer questions like: ‘‘Which authors have

the most publications in the last 10 years on medication use?'' or ''Give two reasons why the use of cephalosporin in hospitals needs to be controlled.'' To answer questions containing relative terms like ''most'' or ''earliest,'' the user should consult a QAS or a Database Management System. But an IRS has no problems displaying a list of article references that discuss drug or medication use, and an intelligent system could assist the user in finding the answer to some of the questions by controlling the way this list is displayed (e.g., in chronological order, or alphabetically by author).

The IIRS prototype relies on users not trying to confuse the system with excessively ambiguous or complex queries. To insure correct interpretation, the NLP unit always displays its analysis so the user can correct any mistakes before the query is processed. One way of allowing the user to modify the query is to explain **why** this particular interpretation was chosen (see the discussion concerning RULES below), so the user can completely resubmit the query after rephrasing the ambiguous part. Since most mistakes will be in the incorrect identification of a substring as belonging to a particular field (abstract, year, source, etc.) an alternative approach to correcting the query would be for the user to mark the substring in question as a block, and to move or assign it to the correct field.

PASS 1: MORPHOLOGICAL AND SYNTACTIC PROCESSING

The first step in the translation process is to parse the sentence into individual words. All punctuation marks are removed except for the quotes (''), which may be useful in identifying the source field. Hyphenated words (e.g., ''cost-benefit, 1978-1985'') are also automatically separated. During the same scanning process all words are capitalized. This eliminates the problem of case sensitivity during the matching process with key words in the inverted files, which are likewise all converted to upper

Please list all very recent papers on marketing practices in hospitals, published in American Pharmacy by Smith.

(1) PLEASE LIST ALL VERY RECENT PAPERS ON MARKETING PRACTICES AND ADVERTISING IN HOSPITALS PUBLISHED IN AMERICAN PHARMACY BY SMITH

(2) PLEASE LIST ALL VERY RECENT PAPER ON MARKET PRACTICE AND ADVERTISE IN HOSPITAL PUBLISH IN AMERICAN PHARMACY BY SMITH

(3) _ LIST _ VERY RECENT PAPER ON MARKET PRACTICE AND ADVERTISE IN HOSPITAL PUBLISH IN AMERICAN PHARMACY BY SMITH

(4) _ (COMMAND) _ (YEAR) VERY RECENT (DOCUMENT) (TOPIC) MARKET PRACTICE _ ADVERTISE (YEAR) HOSPITAL (PUBLISHED) (YEAR) AMERICAN PHARMACY (AUTHOR) SMITH

(5) _ (COMMAND) _ (YEAR) VERY RECENT (DOCUMENT) (TOPIC) MARKET PRACTICE _ADVERTISE _ HOSPITAL (PUBLISHED) (SOURCE) AMERICAN PHARMACY (AUTHOR) SMITH

(6) YEAR => 1986 (0.6) 1987 (0.8) 1988 (1.0)
SOURCE => AMPHDF (0.0)
AUTHOR => SMITH (0.0)
TOPIC => MARKET (0.5) PRACTICE (0.5) ADVERTISE (0.0) HOSPITAL (0.0)

Figure 8: Sample Query

case (see Figure 8).

Morphological processing is the next step in the first analysis pass through the query. Suffixes are removed from the input string, and word stems are generated using an on line dictionary. The same process was applied to the document collection in setting up the inverted files. This makes the retrieval process more effective by increasing the number of retrieved documents which tends to increase the recall factor (proportion of relevant references actually retrieved). Step 2 in Figure 8 illustrates this process on the sample query. A simple algorithm is used to remove plurals, verb conjugations, and superlatives. An on-line dictionary is used

to determine whether a search term has been reduced to its minimal root form, or whether it needs further processing.

The dictionary is also used to flag misspelled words and allows the user to correct them before continuing with the query transformation. In case the word was spelled correctly (but not in the dictionary), the user has the option of forcing acceptance. In this respect the prototype shows some aspects of intelligence in that it ‘‘learns’’ to recognize the word at future times by adding the word to an auxiliary dictionary which is kept in the user’s profile.

The final step during the first pass through the query is a lexical operation; a stop list is employed to delete common words (see step three in Figure 8). This is done to improve the precision performance of the IRS, that is the ability to reject useless material. This list contains high frequency words which have very little ‘‘resolving power’’ (ability to distinguish relevant items) [LUHN 57]. As Figure 9 shows, many of these stop words are short function words. Sometimes the decision whether to include a term in the list results from comparing the recall of retrieved references with and without the term in question. To determine, for example, whether ‘‘USE’’ is a good candidate for inclusion in the stop list, one could analyze the retrieval performance using the following queries: ‘‘I would like a list of non-USA papers on the use of tranquilizers’’ and ‘‘I need a complete list of all papers which use the defined daily dose as a method for measuring utilization.’’ Anyway, the stop list used by the NLP must be a superset of the one used by the retrieval component in setting up the inverted files. If a term belongs to that component’s stoplist, it is excluded from the inverted files and will never contribute in retrieving relevant references, whether the stop list word is included in the query or not.

A	BELOW	NEITHER	TITLE
ABOVE	BESIDE	OF	TO
ACROSS	BOTH	OPPOSITE	TOGETHER
AGAINST	BUT	OR	TOPIC
ALL	COMPLETE	ORDER	TOWARD
ALL	COULD	OUTSIDE	UNDER
ALONG	DO	PLEASE	UPON
AM	ESPECIALLY	REALLY	USE
AMONG	EVERY	SOME	VERY
AN	FOR	SUBJECT	WERE
ANY	HAS	SURVEY	WHAT
ANYONE	HAVE	THAN	WHETHER
ARE	INSIDE	THAT	WHICH
AS	IS	THE	WITH
AT	MAKE	THEIR	WITHOUT
BE	ME	THERE	WORD
BEHIND	MY	TILL	WOULD
			YOU

Figure 9: Stoplist

Because the system will later have to display its understanding to the user, to whom this transformation process must be transparent, stop words are not really deleted from the string, but are marked as TO BE SKIPPED. This will also be necessary later when assigning weights, which depends partially on adjacency information in the query.

PASS 2: PATTERN MATCHING

The second pass over the query consists of syntactic parsing. Grammatical rules are used to build a structure that depicts the relations between words in a sentence. There are several established algorithms to achieve this: context free phrase structure grammars (S -> NP + VP), context sensitive transformation grammars (surface structure mapped to deep structure), and Augmented Transition Networks

[WINOGRAD 83].

The goal of parsing the user's query is to detect relevant search terms and to assign these substrings to the different fields of the frame (author, abstract, year, etc.). The prototype achieves this using keyword matching. Unlike Weizenbaum's ELIZA program, which uses variable pattern matching to achieve sentence recognition and generation, this IIRS employs open pattern matching, augmented by a set of context sensitive conflict resolution rules. During the second and third pass of the query, the NLP detects keywords and flags them for later use. A typical query consists of the following elements: COMMAND (e.g., ''Give me'') DOCUMENT (e.g., ''a complete list'') PUBLISHED (e.g., ''written'') and a combination of elements specifying TOPIC, AUTHOR, YEAR, and SOURCE (see step 4 in Figure 8). Note that all these elements are optional and can occur in any position, with the exception of COMMAND, which if specified, always occurs in the beginning and DOCUMENT which occupies second or third place.

Unlike the syntactic parsing approach, keyword mapping allows the system to deal with grammatically incorrect queries. In fact, given the current contents of the stop word list, some words are deleted from the query that would make it almost impossible to reconstruct the correct underlying grammatical structure. Taking a deep-understanding approach like ATN is not only unnecessary for the purpose of identifying the different fields, it relies heavily on semantic processing to resolve ambiguities through the analysis of meaning.

The keyword mapping approach used by the interface component is called open pattern matching because a word is mapped to an attribute only if it is a keyword (that is, if it is member of a keyword list). A quick look at Figure 10 explains the importance of prepositions hinted at earlier. ''About,'' for example usually flags the beginning of

the TOPIC field (which corresponds to both the $ABSTRACT and $TITLE field of the frame), and ‘‘by’’ most of the time indicates an AUTHOR ($AUTHOR) field. The words usually and most of the time are important, since it is often the case that they are plain prepositions that belong in the stop list instead of field delimiters. In the following query, for example, ‘‘BY’’ does not flag an AUTHOR field: ‘‘List all papers on advertising **by** urologists’’. ‘‘In’’ is especially ambiguous; it may begin a YEAR field, indicate a SOURCE, or act as a plain preposition to be ignored. At this stage, the keywords are mapped to the default attributes as indicated in Figure 10. Again, none of the keywords are actually replaced, but flagged, since it may be (and often is) necessary later to reevaluate this initial assignment. The conflict resolution rules will try to correct mistakes later in the transformation process.

There are two instances where there are no prepositions to flag a field; The YEAR field may occasionally appear without a preposition, namely as an adjective in front of DOCUMENT (e.g., ‘‘recent papers,’’ ‘‘the earliest reference’’). To flag these exceptions, a dummy YEAR flag (not corresponding to an actual word in the query)

(command)	GIVE INTEREST LIKE LIST LOOK NEED RETRIEVE SHOW WANT
(document)	ARTICLE BIBLIOGRAPHY BOOK CITATION DISSERTATION DOCUMENT JOURNAL PAPER PUBLICATION REFERENCE RESEARCH STUDY THESIS WORK
(published)	APPEAR PUBLISH WRITTEN
(topic)	ABOUT CONCERN DEAL DISCUSS ON PERTAIN REGARD RELATE
(author)	BY
(year)	AFTER AROUND BEFORE BETWEEN BEYOND DURING FROM IN PAST PRIOR SINCE THROUGHOUT UNTIL UP

FIgure 10: Pattern Matching

is inserted in front of the adjective to facilitate the conflict resolution process that follows later. Secondly, sometimes DOCUMENT is not followed by any flag, in which case a dummy TOPIC flag is asserted to make up for a lack of a keyword. E.g. ‘‘List all papers which use the DDD as a method of measuring utilization.’’

Pass 3: Boolean Operators and Intensifiers

During the third pass through the query, a different set of key words is flagged. As mentioned in chapter 2, one motivation for building a natural language front end to an IRS is because the casual user is unfamiliar with using Boolean operators in their strict logical sense. This is especially obvious with ‘‘and,’’ which in everyday English usually corresponds to logical OR, as can be seen by the following sample query: ‘‘List all papers from 1960-1969 **and** 1980-1988, written on drug use.’’ (A strict Boolean AND would retrieve an empty set) Ordinarily then, all occurrences of ‘‘and’’ should be substituted by a logical OR. And since the inference component automatically ORs all keywords inside a particular field, as mentioned in Chapter three, any occurrence of or is redundant and can be treated as a stop word. There are a few instances, however, where the user really does mean ‘‘and’’ in the strict logical sense, as for example in: ‘‘Retrieve all papers on medication use written by **both** Haynes **and** Hacket.’’ To flag these exceptions, the parser looks for words like ‘‘both,’’ ‘‘all,’’ and ‘‘neither,’’ which often precede a logical and. The importance of distinguishing between these two interpretations of ‘‘and’’ will become clear later in the processing step that assigns weights to keywords.

During this same pass, the NLP also looks for the third Boolean operator, ‘‘not.’’ It too needs to be flagged for weight assigning purposes. As will be shown later, key words preceded by not receive a negative weight, so documents containing them will be ranked low in the relevance evaluation. Again everyday English is casual about

the range of the NOT operation; ‘‘NOT about A and B,’’ ‘‘NOT about A or B,’’ ‘‘NOT about A and NOT about B,’’ and ‘‘NOT about A or NOT about B’’ are treated as equivalent, although they are obviously not equivalent in the strict logical sense. The algorithm used by the prototype gives all keywords following NOT a negative weight, but not across fields. To facilitate the assignments of weights, a dummy NOT flag is inserted after all ORs and ANDs in the range of the NOT operation. For example, in the query ‘‘Documents not about podiatry or dentistry, but about osteopathy,’’ the first two key words will get a negative weight, but ‘‘osteopathy’’ will not because it is in a different field (following the second ‘‘about’’), and hence is outside the NOT range. Note that at this point in time there are apparently **two** TOPIC fields, as flagged by the preposition ‘‘about.’’ The conflict resolution rules are responsible for combining these two into one large TOPIC field later.

Together with the flagging of Boolean operators, and also for the purpose of assigning weights later, intensifiers are marked. This allows the users to submit their information need and mark specific parts as especially important. Key words flagged by an intensifier will get a higher weight, and thus documents containing them will be ranked higher. Like the NOT operand, of which they are the complement, the intensifiers have a range limited to the current field. If desired, intensifiers could be subdivided into classes of intensity, with corresponding differences in weight (see Figure 11).

PASS 4: CONFLICT RESOLUTION RULES

After the pattern matching process, the query needs to be subdivided into the different fields of the frame ($AUTHOR, $SOURCE, $TITLE, $YEAR). Each field is marked with a flag set up by the first pattern matching process (TOPIC, YEAR, AUTHOR, SOURCE). However, some conflicts may have to be resolved prior to this

parsing; some neighboring fields of the same type may have to be consolidated (e.g., papers by (AUTHOR) Haynes and by (AUTHOR) Sacket -> papers by (AUTHOR) Haynes and by Sacket), while other flags may have to be reevaluated (e.g., papers in (YEAR) American Pharmacy -> papers in (SOURCE) American Pharmacy). For this purpose a set of conflict resolution rules was developed (they are represented in this thesis in paraphrased form). While these rules may sometimes seem arbitrary, and occasionally lead to a misinterpretation of the query, it is important for a working model to fail gracefully under all circumstances, rather than crash. Keep in mind that the next step will be to display the understanding and allow the user to correct the interpretation if necessary.

(1.0) ABSOLUTELY COMPLETELY ENTIRELY EXTREMELY FULLY UTTERLY
(0.9) ALMOST NEARLY VIRTUALLY PRACTICALLY
(0.8) DEEPLY GREATLY HIGHLY REALLY QUITE MOST VERY
(0.4) FAIRLY RATHER PRETTY

Figure 11: Intensifiers

The AUTHOR field poses few problems. It is introduced by the preposition ‘‘by.’’ Of course that preposition may serve other purposes; (e.g., ‘‘List papers on prescriptions by urologists’’). The following rule attempts to solve this type of ambiguity:

IF AUTHOR(X) AND FOLLOWS(AUTHOR(X),TOPIC(_)) THEN SKIP(X). (1)

This rule requires the user to use ‘‘**published** by’’ or a similar phrase when specifying an author after specifying a topic; i.e. while ‘‘books on medication by David Knapp’’ is ambiguous, ‘‘books on medication, written by David Knapp’’ is not. Neither is ‘‘books by David Knapp on medication.’’

A second rule about the AUTHOR field specifies that only one such field may exist in the query. It arbitrarily selects the first occurrence, and consolidates all neighboring AUTHOR fields by making all subsequent occurrences of ‘‘by’’ into plain prepositions to be skipped.

IF AUTHOR(X) AND EXISTS(AUTHOR(_)) THEN SKIP(X). (2)

An example of this is ‘‘papers **by** Haynes and **by** Sacket,’’ which should of course be just one AUTHOR field.

The YEAR field is much more complicated. Any of the prepositions that introduced it may have to be treated as a regular preposition to be skipped. Hence there is a rule similar to rule (1):

IF YEAR(X) AND FOLLOWS(YEAR(X),TOPIC(_)) THEN SKIP(X). (3)

An example of this is ‘‘List any documents about medication **in** 1900.’’ Again, using ‘‘**published**’’ after a TOPIC is required to interpret the query correctly: ‘‘Papers on podiatry, written between 1986 and 88.’’

‘‘In’’ is the most difficult preposition to analyze. It gets a default flag of YEAR, but may just as well be SOURCE, plain preposition, and even TOPIC. The latter is an exception, to be able to process queries like ‘‘List the citations in which the word diabetes appears.’’

IF YEAR(in) AND NEIGHBOR(in,which) THEN TOPIC(in). (4)

Since YEAR involves the specification of a time period, it is usually followed by a number or number range, or by a limited set of keywords (recent, early, late, old, current, now, last). Hence, if neither rule 3 nor rule 4 is fired, and YEAR is followed by a number, or equivalent keyword, then we may assume we made a correct

assumption. Otherwise, YEAR is tentatively reflagged as SOURCE (e.g., ‘‘papers in American Pharmacy’’).

IF YEAR(X) AND NOT RULE(3) AND NOT RULE(4)
AND NEIGHBOR(X,NUMBER(_)) THEN DO_NOTHING. (5)
IF YEAR(X) AND NOT RULE(3) AND NOT RULE(4)
AND NOT RULE(5) THEN SOURCE(in). (6)

Several YEAR flags may qualify for rule (5), but again there is only room for one set in the $YEAR field. Like rule (2), there are cases where YEAR needs to be consolidated, but we can no longer arbitrarily select the first occurrence. Doing that would lead to wrong interpretations of queries like ‘‘**recent** studies of cimetidine, written **in** 1962’’, because ‘‘in 1962’’ would then become part of TOPIC. To prevent this, all instances of YEAR are evaluated against each other, and only the strongest case is preserved (usually the first occurrence of YEAR following the keywords PUBLISHED or DOCUMENT); all other occurrences are skipped as plain prepositions.

IF YEAR(X) AND NEIGHBOR(DOCUMENT|PUBLISHED,X)
THEN WEIGHT(X) = MAX. (7)
IF YEAR(X) AND RULE(5) AND (WEIGHT(X) < MAX) THEN SKIP(X). (8)
IF YEAR(X) AND RULE(5) AND (WEIGHT(X) = MAX)
AND EXISTS(YEAR(_)) THEN SKIP(X). (9)

This will cause the consolidation of queries like ‘‘papers published **after** 87 or **in** 84’’.

Like the two previous flags, TOPIC may only occur once in the query. Like the YEAR flag, all occurrences are evaluated against each other for their appropriateness as TOPIC flag. This evaluation is based on two rules: TOPIC following PUBLISHED or DOCUMENT has a stronger case than those TOPIC flags that do not.

IF TOPIC(X) AND NEIGHBOR(DOCUMENT|PUBLISHED,X)
THEN WEIGHT(X) = MAX. (10)

This rule will solve conflicts like ‘‘documents published **about** the effects of packaging **on** compliance’’. Secondly, TOPIC followed by a search term has a stronger case than those flags followed by a keyword used in the matching process.

IF TOPIC(X) AND NOT NEIGHBOR(X,DOCUMENT|PUBLISHED|COMMAND)
THEN WEIGHT(X) = MAX. (11)

This rule solves possible ambiguities like ‘‘list bibliographies **on** studies **dealing** with cimetidine.’’ After evaluating all flags, only the strongest one will be retained. All others will be skipped as plain prepositions. In case there are several strongest TOPIC flags, the first one is picked arbitrarily (rule (13)).

IF TOPIC(X) AND (WEIGHT(X) < MAX) THEN SKIP(X). (12)
IF TOPIC(X) AND (WEIGHT(X) = MAX) AND EXISTS(YEAR(_))
THEN SKIP(X). (13)

The final category to be evaluated is the SOURCE field. The only possible case is if ‘‘in’’ has failed as YEAR flag, or TOPIC flag (rule(6)). There is little cause for conflict here, except that only one SOURCE may exist in the $SOURCE field. Like rule 2, then, only the first occurrence is arbitrarily retained, the rest is skipped so the field can be consolidated.

IF SOURCE(X) AND EXISTS(SOURCE(_)) THEN SKIP(X). (14)

This will resolve queries like ‘‘papers published **in** American Pharmacy, and **in** Journal of the American Geriatric Society.’’

WEIGHT ASSIGNMENT

After prompting the user for missing fields, the interface displays its understanding of the query, and allows the user to correct any misinterpretations. The different fields have been isolated, and the search terms identified (all words that are not in the stop list, or used in the pattern matching process). These search terms consist of ‘‘crisp’’ items (’’marketing,’’ ‘‘practice’’), imprecise terms (’’recent’’), and fuzzy quantifiers (’’very’’). The last two are considered fuzzy because they convey imprecise information, and do not have sharp distinctions between membership or non-membership. To handle these uncertainties, each concept is given a weight, which is determined by fuzzy logic [ZADEH 81]. These weights range between -1.0 and

1.0, and are used by the retrieval component in addition to weights stored in the inverted files to identify relevant documents (see step 6 in Figure 8).

The retrieval component uses only logical OR operations between each search term within a field. The NOT operation is simulated using negative weights, since this will cause documents containing these terms to be ranked much lower in the relevance evaluation. The second step of pattern matching has evaluated the range of the NOT operation, and has put a dummy NOT flag in front of all the terms covered, so that assigning a negative weight is a simple process of checking whether the term is preceded by a NOT flag.

Weights can also be used to simulate the Boolean AND operand; terms ANDed together get a higher weight so that documents containing both terms will get a higher ranking. The same technique is used for keeping compound terms together (e.g., prescribing pattern,'' ''OTC status,'' etc.). The weight assigned to ANDed and compound terms is calculated by the following formula:

$$1.0 / N$$

where N is the number of ANDed terms and compound words. With the NOT operation the weight is calculated by

$$-1.0 / N$$

For individual search terms, the default weight is 0.0, and -1.0 for NOT operations.

As discussed earlier, the user can apply intensifiers to stress a particular part of the query. Whether the selected items are individual terms or compound words, they will receive a weight of 1.0 when flagged by the intensifier, which causes documents containing them to be ranked higher. The flagging of terms for the intensifier operation is done by the second part of the pattern matching operation.

At this point the user's query has been modified to fit the frame structure through which the interface communicates with the inference component. Search terms have been isolated by field and assigned appropriate weights. At the same time, the query has been enhanced by augmenting the keywords with concepts from the knowledge source. This process is transparent to the user, and has been described in chapter three. After the retrieval component has ranked a list of relevant documents, the interface resumes control, and interacts with the user to display the retrieved references. This can be achieved using a menu structure, or through natural language interaction. The latter is beyond the scope of this thesis, however.

Conclusion

The interface component accepts the user's natural language query statement, and transforms it to a set of weighted search terms, separated into different field categories. The query is transformed in three passes after which any ambiguities are removed through a set of conflict resolution rules and through user intervention. After assigning weights to the search terms, the complete frame is then passed on to the retrieval component.

CONCLUSION

An Intelligent Information Retrieval System provides enhanced performance by integrating conventional IR methods with techniques adapted from Artificial Intelligence. A natural language interface can virtually eliminate the need for a formal query-formatting syntax and the formulation of a request as logical document set manipulations of specific terms.

A prototype of such an IIRS has been built around a document collection of articles on ''determinants of medication,'' with the specific goal of facilitating access to this

information, and to serve as a testbed for enhancing retrieval performance through the use of techniques from AI. The system is intended to simulate a trained human information specialist who assists a user in formulating his information need.

By restricting the user's input to single imperative sentences, the interface can process the user's query statement without requiring infallible detection of syntactic errors or semantic ambiguities. The prototype employs a pattern matching algorithm which is based on delineating fields by flagging prepositions. A set of conflict resolution rules are able to remove most ambiguities in parsing and interpreting the query, but the system relies on the users not trying to confuse it with excessively ambiguous or complex queries. In those few cases where the system does misinterpret a query statement, the user is able to correct the mistake before the frame is sent to the retrieval component.

The system has been tested using a subset of the document collection, and by means of a set of representative queries posed by an expert in the subject domain area. The next step is to expose it to a larger collection of documents, and to a variety of users. By carefully monitoring the understanding of the interface front-end, the performance of the system can be improved by fine-tuning the pattern matching process, the conflict resolution rules, and the weight distribution algorithm.

Since the prototype is designed as a testbed, it has a very modular structure, and lends itself to experimenting with other techniques from AI to enhance the recall and precision. An IIRS can assist the user in reformulating the query when the set of retrieved documents proves irrelevant or unsatisfactory. This resulting interaction with the user could also be achieved through natural language processing.

The natural language processing component has been implemented without using strict syntactic parsing. It would be an interesting experiment to compare the performance of a system based on the latter, and one based on pattern matching. Without the benefit of semantic analysis, it is doubtful that parsing using grammatical rules is better able to resolve ambiguities in the user's request. Both techniques could also be evaluated against a Boolean request from the casual user directly to the retrieval component.

The appendix contains several examples of interactions with the IIRS. To illustrate the way the natural language interface analyzes and converts the query, the prototype displays the query as it is being reformatted. This is normally transparent to the user, and is included here only for demonstration purposes.

SELECTED BIBLIOGRAPHY

Baxendale, P. ‘‘An Emperical Model for Machine Indexing--Progress and Problems. Third Institute on Information Storage and Retrieval. American University, February 1961: pp. 207-218.

Biswas G. et al. ‘‘Knowledge-Assisted Document Retrieval: I. The Natural Language Interface.’’ Journal of the American Society for Information Science. 38 (1987):83-96.

Bolc, L. (ed). Natural Language Communication via Computers. Lecture Notes in Computer Science, Springer Verlag, Berlin: 1978.

Borko, H. ‘‘Getting Started in Expert Library Research.’’ Information Processing and Management. 23 (1987): 81-88.

Brooks, H. M. ‘‘Expert Systems and Intelligent Information Retrieval.’’ Information Processing and Management. 23 (1987): 367-382.

Chiaramella, Y. and Defude, B. ‘‘A Prototype of an Intelligent System for Information Retrieval: IOTA.’’ Information Processing and Management. 23 (1987): 285-303.

Climenson, W. D. Hardwick, N. H. and Jacobson, S.N. ‘‘Automatic Syntax Analysis in Machine Indexing and Abstracting.’’ American Documentation. 12 (July 1961): pp. 178-183.

Cooper, W. ‘‘Exploiting the Maximum Entropy Principle to Increase Retrieval Effectiveness.’’ Journal of the American Society for Information Science. 34 (1983): 31-39.

Croft, W. Bruce. ‘‘Approaches to Intelligent Information Retrieval.’’ Information Processing and Management. 23 (1987): 249-254.

Damerau, F. ‘‘Automatic Parsing for Content Analysis.’’ Communications of the ACM. 13 (June 1970): pp. 356-360.

--------. ‘‘Automated Language Processing.’’ Annual Review of Information Science and Technology. Williams, M.E. (ed). American Society for Information Science, Washington DC. 11, (1976): pp. 107-161.

Doszkocs, Tamas E. ‘‘Natural Language Processing in Information Retrieval.’’ Journal of tthe American Society for Information Science. 37 (1986): 191-196.

Gardin, J. ‘‘On the Relation between Question Answering Systems and Various Theoretical Approaches to the Analysis of Text.’’ The Analysis of Meaning. M. MacCafferty and K. Gray (eds): Aslib: London, 1979: 206-220.

Grishman, Ralph. ‘‘Natural Language Processing.’’ Journal of the American Society for Information Processing 35 (1984): 291-296.

Korfhage, R. and Chavarria-Garza, H. ‘‘Retrieval Improvement by the Interaction of Queries and User Profiles.’’ Department of Computer Science and Engineering. Dallas, TX: Southern Methodist University, 1982.

Lancaster, F. Wilfrid. Vocabulary Control for Information Retrieval. Information Resources Press: Washington D.C., 1972.

--------. Information Retrieval Systems: Characteristics, Testing and Evaluation. New York: John Wiley and Sons, 1979.

Luhn, H. P. ‘‘A Statistical Approach to Mechanized Encoding and Searching of Literary Information.’’ IBM Journal of Research and Development. 1 (1957): 309-317.

Montgomery, C. A. ‘‘Linguistics and Information Science,’’ Journal of the American Society for Information Science. 23 (May-June 1972): pp 195-219.

Robertson, S. ‘‘Between Aboutness and Meaning.’’ The Analysis of Meaning. MacCafferty, M. and Gray, K. (eds). Aslib: London, 1979: pp 202-205.

Rustin, R. (ed). Natural Language Processing. Courant Computer Science Symposium 8, New York: Algorithmics Press, 1973.

Salton, G. ‘‘Automatic Phrase Matching.’’ Readings in Automatic Language Proc-

essing, Hays, D. (ed). New York: American Elsevier Publishing Company, 1966: pp. 169-188.

--------. Introduction to Modern Information Retrieval. New York: McGraw-Hill, 1983a.

--------. ‘‘Expert Systems and Information Retrieval.’’ ACM SIGIR Conference Proceedings. 21 (Spring/Summer 1987): 3-10.

Salton, G.; Buckley, C. and Fox, E. ‘‘Automatic Query Formulations in Information Retrieval.’’ Journal of the American Society for Information Science. 34 (1983b): 262-280.

Schneider, H. J. ‘‘Are Intelligent Information Systems Ever Achievable? Inviting a Discussion.’’ Information Systems. 3 (1978): 1-3.

Sparck Jones, K. ‘‘Problems in the Representation of Meaning.’’ The Analysis of Meaning. MacCafferty, M. and Gray, K. (eds) Aslib: London, 1979: pp 193-201.

--------. ‘‘Intelligent Retrieval.’’ Intelligent Information Retrieval: Informatics 7. Jones, K. P. (ed). London: Aslib, 1983: 136-142.

Sparck Jones, K. and Kay, M. Linguistics and Information Science. Academic Press, NY: 1973.

Van Rijsbergen, C. J. ‘‘A New Theoretical Framework for Information Retrieval.’’ Proceedings of the ACM Conference on Research and Development in Information Retrieval. Pisa, Italy (September 1986): 194-200.

Walker, D. E.; Karlgren, H. and Kay, M. (eds). Natural Language in Information Science, FID Publication 551, Skriptor, Stockholm: 1977.

Winograd, T. Language as a Cognitive Process: Syntax. Reading, MA: Addison-Wesley, 1983.

Woods, W. A. ‘‘Transition Network Grammars for Natural Language Analysis.’’ Communications of the ACM. 13 (1970): 591-606.

Zadeh, L. ‘‘PRUF--A Meaning Representation Language for Natural Languages.’’ in Fuzzy Reasoning and its Applications. Mamdami, E. and Gaines, B. eds. New

York: Academic Press, 1981.

APPENDIX

SAMPLE QUERIES

INPUT THE QUERY:
List some citations on whether being really old makes a difference in compliance.
LIST SOME CITATION ON WHETHER BE REALLY OLD MAKE A DIFFERENCE IN COMPLIANCE
(COMMAND) _ (DOCUMENT) (TOPIC) _ OLD _ DIFFERENCE (YEAR) COMPLIANCE
(COMMAND) _ (DOCUMENT) (TOPIC) _ OLD _ DIFFERENCE _ COMPLIANCE
Author:->
Date: ->
Source:->
Topic: -> WHETHER BE REALLY OLD MAKE A DIFFERENCE IN COMPLIANCE
$AUTHOR =>
$YEAR =>
$SOURCE =>
$TITLE => OLD (0.0) DIFFERENCE (0.0) COMPLIANCE (0.0)
$ABSTRACT => OLD (0.0) DIFFERENCE (0.0) COMPLIANCE (0.0)

INPUT THE QUERY:
Give me a list of papers that deal with cost-benefit analysis.
GIVE ME A LIST OF PAPER THAT DEAL WITH COST BENEFIT ANALYSIS
(COMMAND) _ (DOCUMENT) _ (TOPIC) COST BENEFIT ANALYSIS
(COMMAND) _ (DOCUMENT) _ (TOPIC) COST BENEFIT ANALYSIS
Author:->
Date: ->
Source:->
Topic: -> COST BENEFIT ANALYSIS
$AUTHOR =>
$YEAR =>
$SOURCE =>
$TITLE => COST (0.3) BENEFIT (0.3) ANALYSIS (0.3)
$ABSTRACT => COST (0.3) BENEFIT (0.3) ANALYSIS (0.3)

INPUT THE QUERY:
Give me the earliest paper published on the effect of advertising on prescriptions.
GIVE ME THE EARLY PAPER PUBLISH ON THE EFFECT OF ADVERTISE ON PRESCRIPTION
(COMMAND) _ EARLY (DOCUMENT) (PUBLISHED) (TOPIC) _ EFFECT _ ADVERTISE (TOPIC) PRESCRIPTION
(COMMAND) _ (YEAR) EARLY (DOCUMENT) (PUBLISHED) (TOPIC) _ EFFECT _ ADVERTISE _ PRESCRIPTION
Author:->
Date: -> EARLY
Source:->
Topic: -> THE EFFECT OF ADVERTISE ON PRESCRIPTION
$AUTHOR =>
$YEAR => 1950 (0.0) 1951 (0.0) 1952 (0.0) 1953 (0.0) 1954 (0.0) 1955 (0.0) 1956 (0.0) 1957 (0.0) 1958 (0.0) 1959 (0.0) 1960 (0.0) 1961 (0.0) 1962 (0.0)
$SOURCE =>
$TITLE => EFFECT (0.0) ADVERTISE (0.0) PRESCRIPTION (0.0)
$ABSTRACT => EFFECT (0.0) ADVERTISE (0.0) PRESCRIPTION (0.0)

INPUT THE QUERY:
Give me very recent survey articles written by M. Smith about both predicate medication use and substitution.
GIVE ME VERY RECENT SURVEY ARTICLE WRITTEN BY M SMITH ABOUT BOTH PREDICATE MEDICATION USE AND SUBSTITUTION
(COMMAND) _ VERY RECENT _ (DOCUMENT) (PUBLISHED) (AUTHOR) M SMITH (TOPIC) _ PREDICATE MEDICATION _ AND SUBSTITUTION
(COMMAND) _ (YEAR) VERY RECENT _ (DOCUMENT) (PUBLISHED) (AUTHOR) M SMITH (TOPIC) _ PREDICATE MEDICATION _ AND SUBSTITUTION
Author:-> M SMITH
Date: -> VERY RECENT SURVEY
Source:->
Topic: -> BOTH PREDICATE MEDICATION USE AND SUBSTITUTION
$AUTHOR => M (0.5) SMITH (0.5)
$YEAR => 1986 (0.6) 1987 (0.8) 1988 (1.0)
$SOURCE =>
$TITLE => PREDICATE (0.3) MEDICATION (0.3) SUBSTITUTION (0.3)
$ABSTRACT => PREDICATE (0.3) MEDICATION (0.3) SUBSTITUTION (0.3)

INPUT THE QUERY:
I need a bibliography on the effects of marketing practices on drug use before the 1962 drug amendments.
I NEED A BIBLIOGRAPHY ON THE EFFECT OF MARKET PRACTICE ON DRUG

USE BEFORE THE 1962 DRUG AMENDMENT
_ (COMMAND) _ (DOCUMENT) (TOPIC) _ EFFECT _ MARKET PRACTICE (TOPIC) DRUG _ (YEAR) _ 1962 DRUG AMENDMENT
_ (COMMAND) _ (DOCUMENT) (TOPIC) _ EFFECT _ MARKET PRACTICE _ DRUG _ 1962 DRUG AMENDMENT
Author:->
Date: ->
Source:->
Topic: -> THE EFFECT OF MARKET PRACTICE ON DRUG USE BEFORE THE 1962 DRUG AMENDMENT
$AUTHOR =>
$YEAR =>
$SOURCE =>
$TITLE => EFFECT (0.0) MARKET (0.5) PRACTICE (0.5)
DRUG (0.0) 1962 (0.3) DRUG (0.3) AMENDMENT (0.3)
$ABSTRACT => EFFECT (0.0) MARKET (0.5) PRACTICE (0.5)
DRUG (0.0) 1962 (0.3) DRUG (0.3) AMENDMENT (0.3)

INPUT THE QUERY:
List the papers on drug use which appeared in the ‘‘American Journal of Hospital Pharmacy’’ between 1960-65.
LIST THE PAPER ON DRUG USE WHICH APPEAR IN THE ‘‘ AMERICAN JOURNAL OF HOSPITAL PHARMACY ‘‘ BETWEEN 1960 1965
(COMMAND) _ (DOCUMENT) (TOPIC) DRUG _ (PUBLISHED) (YEAR) _ AMERICAN JOURNAL _ HOSPITAL PHARMACY _ (YEAR) 1960 1965
(COMMAND) _ (DOCUMENT) (TOPIC) DRUG _ (PUBLISHED) (SOURCE) _ AMERICAN JOURNAL _ HOSPITAL PHARMACY _ (YEAR) 1960 1965
Author:->
Date: -> BETWEEN 1960 1965
Source:-> THE ‘‘ AMERICAN JOURNAL OF HOSPITAL PHARMACY ‘‘
Topic: -> DRUG USE WHICH
$AUTHOR =>
$YEAR => 1960 (0.0) 1961 (0.0) 1962 (0.0) 1963 (0.0)
1964 (0.0) 1965 (0.0)
$SOURCE => AJHPA9 (0.0)
$TITLE => DRUG (0.0)
$ABSTRACT => DRUG (0.0)

INPUT THE QUERY:

List all papers on medication use published in the last 10 years. 50
LIST ALL PAPER ON MEDICATION USE PUBLISH IN THE LAST 10 YEAR
(COMMAND) _ (DOCUMENT) (TOPIC) MEDICATION _ (PUBLISHED) (YEAR) _ LAST 10 YEAR
(COMMAND) _ (DOCUMENT) (TOPIC) MEDICATION _ (PUBLISHED) (YEAR) _ LAST 10 YEAR
Author: ->
Date: -> IN THE LAST 10 YEAR
Source ->
Topic: -> MEDICATION USE
$AUTHOR =>
$YEAR => 1978 (0.0) 1979 (0.0) 1980 (0.0) 1981 (0.0) 1982 (0.0) 1983 (0.0)
1984 (0.0) 1985 (0.0) 1986 (0.0) 1987 (0.0) 1988 (0.0)
$SOURCE =>
$TITLE => MEDICATION (0.0)
$ABSTRACT => MEDICATION (0.0)

BIOGRAPHICAL SKETCH OF THE AUTHOR

G. Jan WILMS, son of Mr and Mrs WILMS of Louvain, was born in Belgium on August 20, 1962, and received his B.A. degree in Germanic Philology from the Katholieke Universiteit, Leuven, in 1984, and an M.A. in English from the University of Mississippi in 1985. He married Wallika Sekthira in 1987.

His major interest are the history of the novel, microcomputer applications, and Artificial Intelligence techniques in Natural Language Processing.

Mr WILMS career objective is to teach computer science courses and perform research at a university. His permanent address is Celestijnenlaan 58, B 3030 Leuven, BELGIUM.