1990

An Application Of Case Relations To Document Retrieval

Xin Lu

AN APPLICATION OF CASE RELATIONS TO DOCUMENT RETRIEVAL

by
Xin Lu

School of Library and Information Science

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario
October, 1. Y0

ISBN 0-315-59092-0

Abstract

The purpose of this research is to design a document retrieval model which is a structural model based on case relations and to test how effectively a prototype of this model would perform retrieval on a test database. Case relations are a major component of case grammar proposed by linguistic theorists and developed in computational linguistics and natural language processing.

The design of the structural retrieval model involves case relations and structured document representation, case relation-based natural language parsing and automatic structural indexing, and tree mapping and structural matching. In this model, a document is represented by a set of tree-like case frames in which the components of a natural language clause are assigned to different nodes called cases, and all nodes have pre-defined case relations to the verb of the clause.

To implement such a structural representation by automatic means, an indexing engine was coded (using PROLOG) and developed which consists of a natural language parser and a case frame generator. In response to a natural language query, the prototype of the model 1) processes and converts the query into a set of case frames; 2) measures the structural closeness between the query and every document in a database through tree-mapping; and 3) presents the retrieved documents, according to their closeness to the query, in ranked order.

A number of typical retrieval experiments have been designed to compare the structural model with the vector space model and the Boolean model. All of the model prototypes processed a set of thirty queries on a test database of 534 documents. The retrieval performance was measured using recall-precision graphs, averaged recall and precision, and statistical tests. The experimental results showed that the effectiveness of the structural model was barely comparable to that of the other models. The conclusions

are: 1) the structural model is not more effective than other models, and 2) replications of this study are needed to further prove or disprove the usefulness of case relations in improving retrieval effectiveness.

Acknowledgements

It is a pleasure to express my indebtedness to Dr. Jean M. Tague, who provided her academic guidance throughout my six-year study at Western, especially during my thesis research.

I am also indebted to Dr. B. Gillian Michell and Dr. Mark T. Kinnucan for their help in the areas of linguistics and cognitive science as well as for their constructive criticisms.

The completion of this thesis is, without question, due to the assistance from other faculty members and staff in the school. Their constant assistance is greatly appreciated. The help from the following doctoral students in performing manual parsing is especially appreciated: Lisa Baron, Patricia Burt, Clara Chu, Margaret Ann Wilkinson, and Dietmar Wolfram.

The completion of this thesis is also due to the support from my family. Many thanks go to my wife Wuhua for her invaluable support in spirit that nobody but she can give to me. Many thanks also go to my parents for their high expectation on their son and their constant encouragement.

Finally, I wish to record my gratitude to the people in China and in Canada for their financial assistance.

Table of Contents

Chapter One
INTRODUCTION

In proposing a new theoretical framework for document retrieval, van Rijsbergen [1987, p.23] stated:

> Almost all of the previous work in information retrieval (including my own) has been based on the assumption that a formal notion of meaning is not required to solve information retrieval problems. Typically, researchers have assumed that one could get by, by only considering absence or presence of word tokens in text together with counting information about the distribution of words. Although such an approach has been successful up to a point, it has become clear that further advances in the effectiveness of retrieval by such techniques are not possible.

The development of a document retrieval system with the capability of dealing with the meaning of text is not a new proposal. Although few information scientists have formally addressed the notion of meaning, they have realized that various conceptual relations associated with index terms have to be incorporated into document representation schemes in order to further improve the effectiveness of document retrieval [Sparck Jones, 1973; MacCatterty & Cray, 1979; Belkin & Vickery 1985]. In a summary section on the early unsuccessful syntactic approaches to document retrieval, Sparck Jones wrote [p.119]:

> . . . we thus conclude with some speculation on untried approaches to syntax. For example, it is intriguing to imagine a system in which document descriptions using links and roles (i.e., the various relationships between index terms) would be obtained automatically from deep structures (i.e., the semantic interpretation of the sentences) produced by an advanced transformational analysis technique. This would be a difficult enterprise, but there is no doubt that it would blend theories and methods of interest in both linguistics and information science, and the results could hardly fail to be instructive.

This study looks at one such untried syntactic/semantic approach to document retrieval. But instead of using transformational analysis techniques, this study uses the case relations developed in the theory of case grammar.

In a number of influential papers, Charles Fillmore [1968, 1970, 1971] developed case grammar, the distinguishing feature of which is that a sentence essentially consists of a verb and one or more noun phrases, that are associated with the verb in different case relationships. In other words, the verb has only one basic sense and several possible case relations that may attached to it. As an example, Fillmore [1968] cited the following sentences:

The door opened.
The janitor opened the door.
The key opened the door.
The janitor opened the door with a key.

On the surface, these sentences are very different in their choice of subject and the use or non-use of a preposition. A close look reveals that the case relationships between the verb OPEN and the nouns DOOR, KEY and JANITOR remain constant: the janitor is the *agent* who does the opening, the key is the *instrument* of opening, and the door is the *object* of opening. Fillmore argued that the variety of forms should not be viewed as a special fact about the verb OPEN, but could be applied at least in part to a whole class of verbs, including BREAK, BUY, and SHATTER. For such verbs, he postulated a underlying graph *case frame* that shows the expected case relationships between the verb and its cases and a general set of rules for determining which case relations can be deleted and which cases can appear in what grammatical position in a sentence.

Fillmore proposed a few very general constraints on this model. First, only one representative of any case can appear in a sentence, except where noun phrases are conjoined and bear a same relation to the verb of the sentence. Second, there is a comprehensive but small number of case relations which can be used within a case frame to describe the relationships between noun phrase(s) and the verb in a sentence. Third, the cases within a case frame could be viewed as a simple linear hierarchy in which the deletion of a higher element causes it to be replaced in the structure by a lower ranked

one. For instance, the instrument in the above example sentences can be the subject in the sentence only where the agent is not specified and the object can become the subject only when neither the agent nor the instrument is specified as long as the active form of the verb is used. Fourth, each case defined in the grammar has only one relation to a dominant verb.

Although case grammar captures an important notion of the native speaker's intrinsic competence in natural language, linguists have not agreed upon a definitive list of case relations. About half a dozen case relations are common to most proposed theories. In the meantime, case grammar has had a strong influence on artificial intelligence (AI) because of its convenient set of labels for conceptual relations. For example, the case frame has been frequently used as a framework for parsing sentences and as a storage structure to represent meaning of a parsed sentence. The case frame has also been used in the process of text understanding, inference, and sentence generation.

The possibility of using case relations in document retrieval was suggested by Dee Lewis [1984]. In seeking a more objective method to determine relevance (aboutness) in document retrieval, Lewis analyzed and compared textual characteristics of queries and document abstracts. The linguistic theory which she used to guide the textual analysis and comparison was case grammar. Lewis concluded from her study on relevance judgment that a query and its relevant document abstracts share a set of keywords and a group of the functional relations that connect the set of keywords. On the other hand, the query and its nonrelevant document abstracts might share a set of keywords but the functional relations between the keywords are different. This conclusion suggests that both index terms and the case relationships between those terms could be incorporated into a document retrieval model to improve its effectiveness.

To design such a document retrieval model, there are two major challenges: structural indexing and structural matching. Structural indexing is challenging because it

must provide meaningful indexing results with immature techniques of natural language processing. Structural matching is challenging because the structural nature of the proposed retrieval model prevents from using available mathematical retrieval functions and a new matching function has to be constructed to take into account case relationships between index terms in measuring similarity of document and query.

The purpose of this research is to design, on the basis of case relations, a document retrieval model named the structural model and to test how effectively its prototype would perform retrieval on a test database. The research covers the topics of case relations and structured document (and/or query statement) representation, case relation based natural language parsing and automatic structural indexing, and tree mapping and structural matching. The document retrieval experiments are designed to assess whether retrieval effectiveness could be advanced by explicitly incorporating case relations in a document retrieval model.

The organization of this thesis is as follows: chapter one introduces the study; chapter two reviews related research; chapter three describes the components of the structural document retrieval model; chapter four describes the experimental design; chapter five presents experiment results and the related discussion: finally, chapter six provides the conclusions of this study.

Chapter Two
LITERATURE REVIEW

The review begins with the studies that have investigated the possibility of using case relations in document retrieval. It follows with an examination of applications of case relations in designing question-answering systems, a type of information retrieval system which is studied in cognitive science and which is closely related to document retrieval systems. Then various structural approaches to document retrieval are discussed, as well as the associated matching functions.

2.1. Case Grammar and Document Retrieval

The first work to be reviewed is Lewis's study [1984], since her findings and conclusions are the starting point of this research. The second is Young's document retrieval system [1973]. The system, although it lacks a document representation system and a retrieval function, provides a complete case relation based indexing engine.

Lewis's research

Lewis's main hypothesis is that if the index terms in a query and a relevant document occur in similar case relations there is a significant similarity between the structure of the query and the structure of the document. In other words, if an abstract is judged to be *about* a query statement, the abstract would not only share many identical index terms with the query statement, but would also share similar case relationships between those index terms.

For her case grammar framework, Lewis selected four categories of verb and nine case relations. They are listed in Table 1. With these categories, Lewis represented both document abstracts and query statements in a number of case frames. The graph in Figure 1 is one such case frame, which represents the sentence "What effect does the living

environment and/or the caretaking figure have on language development in the mentally retarded?". In Figure 1, the verb is in capital and its associated cases are enclosed in a pair of square brackets. Each case consists of a relation tag in italics and a set of associated words in a pair of parentheses.

Verb Category	Case Relation
Stative	Agent
Action	Instrument
Process	Experiencer
Action-Process	Object
	Goal
	Source
	Path
	Location
	Time

Table 1: The Verb Categories and the Case relations Used by Lewis

AFFECT [*agent*
(caretaker)
instrument
(living environment)
object
(language development)
experiencer
(mentally retarded)]

Figure 1: An Example of Sentence Representation in Lewis's Study

Lewis collected thirty-two queries from thirty subject specialists in various disciplines. She retrieved document abstracts on DIALOG for each of these queries. The analysis of the case relationships between keyterms in the queries and in the abstracts were carried out by a linguist and Lewis, and the results were then compared to the relevance judgments made by the subject specialists. She applied a Chi-square test of independence to a 2x2 contingency table to determine whether there was any relationship between the predictions of aboutness using both keyterms and their associated case

relations and the aboutness decisions themselves. The strength of the relationship was assessed in terms of the contingency coefficient. The results are encouraging. The agreement between the two sets of decisions is 97% across all queries. Lewis concluded [Lewis, 1984, p.253]:

> the keyterms in relevant abstracts had to have the meaning described in the query;
>
> the keyterms in relevant abstracts had to be functionally related to each other in the ways described in the query; and
>
> the required functional relations between keyterms were indicated in the query.

Other important findings are: 1) the case relations used were adequate for indexing; 2) the descriptions from the subject specialists were adequate for determining the intended topics; and 3) the subject specialists were very consistent in their language behavior during making *about* and *not about* decisions.

These findings and conclusions suggest that it would be justifiable to construct a document retrieval model in which a document and a query are represented in case frames so that structural matching can be carried out on those case frames in such a way that the greater the number of matched or partially matched case frames between the document and the query, the greater the possibility that the document is topically *about* the query.

Lewis did not, however, suggest any method to computerize her indexing procedure, i.e., constructing case frames from a given text. In fact, her indexing procedure would difficult to automate with current techniques of natural language processing because it sometimes requires sophisticated semantic processing for text reduction. Lewis provided a number of examples in her thesis to demonstrate the reduction process. The following is one of the examples [Lewis, 1984, p.140]. The original abstract is given in Figure 2, the result of the reduction is presented in Figure 3, and the final structure of keyterms and their functional relationships is represented as in Figure 4.

ABSTRACT

Surrogate and natural parent comparisons between institutional and noninstitutional children. Rated the surrogate parents of 20 8-13 year old institutionalized educable mental retardates and the natural parents of a matched sample of 20 noninstituionalized EMRs on Hollingshead's Two Factor Index of Social Position. Results indicate that the institutional surrogate parents had significantly higher socioeconomic status. The present and previous findings suggest that this higher social position may have a positive influence on language habilitation among institutional EMRs.

Figure 2: An Abstract To Be Reduced

REDUCTION

Rated surrogate parents of institutionalized mental retardates;

natural parents of noninstitutionalized mental retardates.

(higher social status) influences language habilitation among institutionalized mental retardates.

parents influence language habilitation among mental retardates.

Figure 3: The Result of The Reduction

```
"influence" [ agent
                    (parents)
              experiencer
                    (mental retardates)
              object
                    (language habilitation)   ]
```

Figure 4: Keyterms And Their Functional Relationships

To avoid the complicated semantic processing shown in the above example, the indexing procedure developed for this study incorporates no text reduction processes but constructs one case frame for each natural language clause. The indexing procedure in this study is thus at a lower level of semantic processing than that employed in Lewis's study. Accordingly, this study is also testing whether the simplified indexing is as useful as the elaborated indexing.

Young's study

The problem addressed in Young's study is the possibility of developing computer programs which operate upon English text so as to produce a form of text representation and high quality indexes by automated means. Young proposed a retrieval model which consists of a complete indexing engine and a structural document representation. Young did not, however, provide any retrieval engine in the model.

Three notions developed in Young's research are important to this study: case relation, verb category and case frame. Case relations can be categorized into the two groups, namely essential case relations and peripheral case relations. Essential case relations are the case relations required by each verb category while peripheral case relations are those which are optional to the verb category. A verb category and its essential case relations form a unique case frame. A case frame may also contain peripheral case relations.

Young derived her case system from three major sources: Fillmore's work [1970, 1971], Chafe's work [1970] and Cook's work [1970, 1971, 1972]. Her system has been selected for this study simply because she developed a set of algorithms for automatically creating case frames from texts. In her system, there are four essential case relations, six peripheral case relations, five different verb categories and their case frames. The following are the case relations; the first four are essential and the rest peripheral.

ESSENTIAL CASE RELATIONS

- **Agent** (AGNT): the source of the action.
  Example: The cyclist hit the car.

- **Experiencer** (EXPC): the one who experiences the feeling, sensation, etc.
  Example: The little boy still remembers that traffic accident.

- **Beneficiary** (BNFC): the possessor (in its broadest sense) of some thing, whether the possession be temporary or permanent, positive or negative.
  Example: The professor has a good collection of Japanese poetry.

- **Objective** (OBJ): the receiver of the action.
  Example: He is typing a letter.

PERIPHERAL CASE RELATIONS

- **Location** (LOC): the place where the action occurs.
  Example: He prepared his final examinations in the library.

- **Time** (TIM): the time when the action occurs.
  Example: She will fly to Vancouver tomorrow.

- **Manner** (MANN): the way in which the action is performed.
  Example: He tested the compounds with great skill.

- **Comitative** (CMTV): a subject accompanying the source of the action.
  Example: The professor did the research with a student.

- **Cause** (CAUS): the reason for the action.
  Example: His hands are tough from heavy work.

- **Purpose** (PPS): the purpose of the action.
  Example: This microcomputer is used for class demonstration.

The five verb categories are AGENTIVE, BENEFACTIVE, EXPERIENCER, REFLEXIVE, STATIVE. The case frames of these five verb categories are described in Table 2. Each case frame consists of a verb category in italics and its essential case relations. Each case frame is accompanied with an example sentence whose verb is in italics.

Indexing in Young's system is performed in two steps: analysis of sentence structures and assignment of case relations. During syntactic analysis, the program called MYRA in Young's system scans a sentence for function words (e.g., "the", "for", "that"), using a dictionary of about 400 such words. The function words, which serve as structural markers within the sentence, are from Fries' model of structural classes [1952]. MYRA then classifies all unclassified words between the function words according to a set of 110 parsing rules. At the end, if no verb is found, MYRA looks for a possible verb slot and reassigns the position accordingly. MYRA has been tested on a text of 6,000 words, derived from a scientific essay, Hemingway's novel "The Old Man and the Sea",

CATEGORY	CASE FRAMES
AGENTIVE	AGNT *AGENTIVE* EXPC, OBJ He *gave* me the letter. OBJ *AGENTIVE* EXPC, AGNT The letter *was given* to me by John.
BENEFACTIVE	BNFC *BENEFACTIVE* OBJ I *have* the book.
EXPERIENCER	EXPC *EXPERIENCER* OBJ The little girl *liked* ice cream.
REFLEXIVE	AGNT-OBJ *REFLEXIVE* The bird *flew*.
STATIVE	OBJ *STATIVE* OBJ John *is* a philosopher.

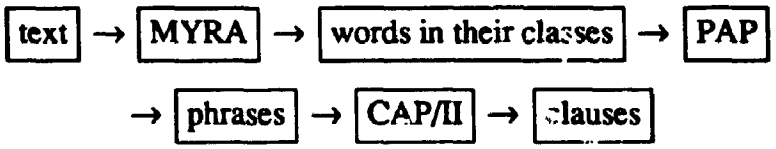
Table 2: The Case Frames

and a general article. The result is average accuracy of 94% in classifying words into classes.

The syntactic analysis proceeds with the output from MYRA. The program CAP/I examines the output, delimiting clause boundaries and identifying the existence of a clause with its 35 rules. For example, if the underlined adjectival clause in the sentence "The girl sitting on the stair won first prize" is not recognized, the clause "The girl won first prize" might be identified but incorrectly delimited by CAP/I. Another program PAP containing 24 rules uses the results from CAP/I to delimit and identify *phrases*. An average accuracy of 62% was attained in delimiting clauses and 90% in delimiting phrases and accuracies of 85% and 92% were found for the identification of clauses and phrases, respectively.

To apply the parsing programs to a logical sequence of words, phrases, and then

clauses, Young proposed the program CAP/II, which incorporates not only the procedures embodied in CAP/I, but also a number of additional procedures that take advantage of the results from PAP. This study uses the following revised parsing sequence:



Once the syntactic parsing is completed, the program CGP is called in to identify the essential cases of agent, experiencer, beneficiary, and object, and the peripheral cases of location, time, manner, comitative, cause, and goal. The program CGP first identifies the dominant verb for each clause by dictionary look-up or by default and then selects a corresponding case frame and instantiates the frame. The test result indicates that the program achieved an accuracy in the range of 75%. Young claimed that, given accurate input from MYRA, CAP and PAP, CGP might achieve greater than 95% accuracy (based on her preliminary studies).

Young also proposed a model for text or sentence representation. She defined one or more graphs for every kind of sentence component such as phrases, phrases related by conjunction, and clauses joined by conjunction. One sentence is thus represented with one or more graphs connected according to their relationships to the verb of the sentence. This representation, according to Young, preserves and explicates the terms and their relations in the sentence and makes it possible to extract terms at various levels of complexity for various purposes. Young did not discuss the representation of inter-sentence relationships nor the representation of text as a whole. She also did not actually test this representation model.

Young's system, or specifically her indexing system, is far from a satisfactory one. Fundamentally, the case relations used in the indexing system were selected with the

concern of practical computation rather than theoretical soundness. Linguistic support has to take into account a number of nonlinguistic matters. For example, Young included the case frame OBJ *STATIVE* OBJ for the complement in the sentence like "that man is a good teacher". This arrangement violates Fillmore's constraint against two instances of the same case in one sentence. However, Young's indexing system is still worth testing since such systems for English are rare. Young's indexing system was used in this research because it is the only one designed exclusively for automatic indexing and it might still have potential for further improvement.

2.2. Case Grammar and Question-Answering Systems

Although case grammar has not been directly used in designing document retrieval systems, it has a history of application in designing question-answering systems. Question-answering systems provide access to factual information in natural language settings. Question-answering systems, like document retrieval systems, have to perform tasks of natural language processing [Lehnert, 1978]. The early applications of case grammar in question-answering systems have been reviewed by Bruce [1975]. The following review, however, concentrates on three studies which were selected because they are closer to document retrieval.

Strong [1974] described an algorithm which generates graph representations of English text. An actual representation is a group of interconnected graphs, with each graph corresponding to one sentence in a text. The shape of the graph corresponds to the syntax of the sentence, the nodes to the phrases of the sentence and the edge types to the case relations associated with the sentence. Different sentence graphs could be interconnected at common nodes and analyzed according to common edges.

Strong's algorithm essentially consists of two parts: syntactic analysis and case frame instantiation. The first part contains 11 rules and the second, 8, including a set of case frames for the verb categories of stative, action, process, action-process,

experiencer, and beneficiary. The algorithm proposed by Strong could be used for automatic indexing and abstracting and question answering. For example, to answer the question "where has popcorn been found?", the system scans the case frames in a database for the verb "found" and the locative case. Strong's algorithm is similar to Young's system but has never been formalized and tested. The suggestion for integrating sentences is interesting; it is an approach to producing coherent representations of a text. However, the effectiveness of the proposed integrating approach is not clear. These problems prohibit any application of Strong's algorithm.

Schank's system of conceptual representation [1975] was developed for common sense reasoning in the field of cognitive science. The system was implemented in the MARGIE system [Schank, Goldman, Rieger, & Riesbeck, 1973]. The system includes a language parser, an inferential memory, and a generation system. At the heart of Schank's system are eleven basic *primitive actions* such as MOVE (move a body part), MBUILD (build a thought), PTRANS (transfer a physical object), etc. Each of these primitive actions in turn determines a conceptual case frame which is the basic component of his semantic network. Other frames can be derived from these eleven basic frames by join and restriction. Schank claimed that these primitive actions, along with a small number of states, are sufficient to represent the meanings of all verbs.

Schank's system was designed for common sense reasoning rather than formal deduction using first-order logic. Given the sentence "Smith gave Mary an aspirin", the system translates it into conceptual dependency graphs, and generates the following plausible inferences:

Smith believes that Mary wants an aspirin.
Mary is sick.
Mary will ingest the aspirin.

Although these inferences are not necessarily true, they do have a high probability of

being true. Schank's system, with its eleven primitive actions, functions at a more abstract level. Schank's system is thus different from those case grammar based systems that depend on the meaning of actual verbs in sentences.

In Schank's question-answering system, the case frame is used as a framework for parsing sentences, as well as a storage structure to represent the meaning of a sentence once it has been parsed. With this storage structure, the process of text understanding, inferential analysis and sentence generation can be conducted conveniently. The usefulness of these techniques to document retrieval will be discussed after reviewing one more recent question-answering system.

Cohen and Kjeldsen [1987] designed an expert system GRANT for searching sources of funding for research proposals. The search method is called "constrained spreading activation", which makes inferences about the information need of a user and searches for information that the user did nc explicitly request but that is likely to be relevant. Unlike most question-answering systems, which concentrate on a tiny knowledge domain and attempt to provide direct answers, GRANT covers many research topics in medical sciences and provides a list of recommended funding agencies for each query.

The semantic network of GRANT contains over 4,500 nodes which represent the various research interests of about 700 funding agencies. The nodes are linked to one another by one of 48 pre-defined relations such as *has-setting* or *isa*. The nodes themselves are represented as frames with slots which in turn represent links to other nodes. There are ten pre-defined frames, including "design", "protect", and "train". Each type of frame is represented by a case frame that includes a set of essential and optional cases or slots. The *study* frame, for instance, contains subject and object slots to represent the topic and a focus slot to indicate which aspect of the topic is to be studied.

There are three constraints in spreading activation. The *distance* constraint ceases activation at a distance of four links from any starting point that is a research topic mentioned in a proposal. The *fan-out* constraint ceases activation at highly connective nodes like "science" or "disease". The third constraint is an inference schema. Within these constraints, activation spreads from the topics of a proposal, through the network, to agencies via their interests. Those activated agencies are regarded as potential funding sources for the proposal and a matching mechanism is called to further compare the number of interests shared by the proposal and the agencies. The results of this matching function are then presented to the author(s) of the proposal in a ranked order.

Question-answering systems like the three systems reviewed are similar to document retrieval systems in their functional configurations, i.e., knowledge representation, query process, database retrieval, and results presentation. The question is whether it is possible to apply techniques such as inference developed for question-answering systems directly to document retrieval. The answer is short: no. Although the two types of information systems share similar functional components, the nature of these components is different in terms of the breadth of subject coverage and the size of text database.

Question-answering systems usually concentrate on specialized subject areas and manage small but well organized knowledge bases. On these knowledge bases, inference processes can be performed in order to instantiate the variables introduced in queries. The queries are usually simple sentences and tend to begin with "who", "what", "how", "where", etc. Document retrieval systems, on the other hand, usually cover broad subject fields and have large text databases and index files. The broad coverage, the large size of the databases, and the limited capacity of language processing make it impossible to design document retrieval systems in the same way as GRANT was designed. Furthermore, information requests in document retrieval systems are generally more complicated than those in question-answering systems and provide few explicit variables

for reasoning. Question-answering systems, as a result, can respond to a request with either a correct answer or an incorrect answer (including no answer), while a document retrieval system can merely recommend either a set of documents or a list of ranked documents to the request without a clear cut-off between relevant and irrelevant answers.

2.3. Structural Approaches to Document Retrieval

The systems to be reviewed in this section attempt to represent documents with both content-bearing terms and the relationships between the terms. The various representations are not directly based upon case relations but are similar to such an approach in various respects.

The SYNTOL System

The SYNTOL system was originally designed as a semi-automatic document retrieval system consisting of a manual indexing subsystem and a computerized retrieval subsystem [Gardin, 1965]. Certain aspects of automatic indexing like syntactic analysis of text were tested on a later version of the system [Bely and others, 1970]. The design of the system aims at the capability of analyzing and retrieving general scientific documents. But the design does not have any theoretical basis.

The structure of the SYNTOL system can be best described by examining its indexing results. Given a document, the corresponding indexing products include a catalog record, an abstract, a source table containing the information of place, date and language, etc., a facet table covering theme, space, time, etc., a number of updated lexicons, and a graph of syntagmata each of which is a triad of term-relation-term. With these index facilities, a user can conduct retrieval at different levels of complexity such as known-item searches on the catalog records, broad subject retrieval on a theme in the facet table, or sophisticated retrieval by selecting broad or specific terms from the lexicons and connecting the terms into syntagmata. The matching procedure in the

SYNTOL system requires that all search entries occurring in a given query are in the retrieved documents, unless they are part of OR or NOT Boolean connectives.

The distinctive feature of the SYNTOL system is the use of syntagmata, pairs of index terms linked by one relation. The four syntactic relations are *coordinative* (e.g., in "differentiation of father ... and mother roles"), *consecutive* (e.g., in "economic crises ... (consecutive to) ... wars"), *associative* (e.g., in "cancerous ... organs"), and *predicative* (e.g., in "increasing ... unemployment"). The selection of these four relations was due to their applicability to a wide variety of semantic contexts. In addition, all index terms are categorized into the four groups, namely, predicates, entities, states and actions. The construction of a syntagma begins with looking up a lexicon and identifying the categories of the two concerned index terms. A relation and the direction of the relation (e.g., from entity to state or from entity to action) are then determined in accordance with certain pre-established rules. Since one index term may appear in different syntagmata and the syntagmata can be connected with one another at common nodes, a document can be represented as one or more directed graphs.

The poor performance of the SYNTOL system in the experiments of the late 60's [Bely and others, 1970] might be attributed to the automatic indexing mechanism, which used a limited syntactic analysis procedure and, more fundamentally, to the syntactic relations employed for constructing syntagmata. These relations are perhaps unnecessarily explicit and have very limited capacity to link index terms into meaningful structures. The procedure of selecting these relations have never been formalized. Probably these problems are the inevitable consequences of the system design which does not have a theoretic basis. In the present study, similar design demerits are reduced to a minimum level.

The SMART Project

Identifying index terms through a syntactic analysis is one of the many document retrieval options investigated in the SMART project [Salton, 1966]. In this project, the major component of the syntactic analysis procedure is a dictionary of criterion phrases or criterion trees represented as dependency trees. There are four categories of criterion trees corresponding to the four syntactic relations: noun phrase, subject-verb, verb-object and subject-object. Each criterion tree consists of concepts, syntactic indicators and the syntactic relationships between the concepts. Since one node of a tree can accommodate different concepts, one criterion tree is capable of representing many different syntactic structures in English. The criterion trees can be easily encoded and stored in a matrix and therefore can be matched by manipulating the matrices.

During syntactic analysis, the sentences of a document are parsed using phrase structure grammar, and each detected tree is matched with the trees in the dictionary to determine if it contains any criterion phrases. The matched or identified criterion phrases, perhaps with a weight, are then merged into a document vector for retrieval. Although this syntactic analysis can process complicated text structures, the use of the dictionary introduces two limitations to this part of the SMART system. First, the limited coverage of the dictionary directly affects the indexing results because some information might be simply not identified during indexing. Related to this is the second that indexing is necessarily limited to a specific subject domain for the sake of comprehensive coverage.

Other problems associated with this criterion tree approach are the weakness of the automatic syntactic analysis, the effectiveness of the syntactic relations and the exact-match method used in comparison For many information systems involving text processing, the unsatisfactory accuracy of the automatic syntactic analysis is a general problem. The use of syntactic relations such as subject and object is not sufficient for representing the contents of documents. Further, purely syntactic structures like criterion

trees are perhaps poor indicators of the semantic structures in a given document. Finally, the exact-match method is a crude measure for comparing structures. The method does not take into account "similarity" between two structures [Karlgren, 1977]. All of these problems might contribute, though each at a different level, to the poor performance of syntactic phrases in the document retrieval experiment [Salton, 1968].

The findings from the development of the SYNTOL system and the experience gathered in the course of the SMART project suggest the following three guidelines to the design of general document retrieval systems:

1. a general document retrieval system should keep its use of context-sensitive dictionaries and/or thesauri to a minimum;

2. purely syntactic structures may not provide adequate representations of document contents for retrieval; and

3. a best-match method would be more appropriate than an exact-match method for measuring structural closeness.

Document Indexing Systems Employing Roles

The two document indexing systems to be reviewed in this section are PRECIS or PREserved Context Index System [Austin, 1974; Austin and Dykstra, 1984] and the Western Reserve University (WRU) indexing system [Perry, 1958; Aitchison & Cleverdon, 1963]. Both indexing systems employ roles or relational indicators to indicate certain conceptual relations between index terms. Although neither system is theoretically based on case grammar, each system is indirectly related to it in the sense that the roles used are similar to case relations.

PRECIS is a string indexing system. The system was designed to increase the effectiveness and efficiency of manual search by providing multiple overlapping index entries for an indexed item and using the computer to generate the descriptive parts of the entries according to a group of explicit syntactic rules. One feature which PRECIS

claimed is its capability of "preserving context" or the relationships between concepts, therefore providing more meaningful index entries than other string indexing systems like KWIC.

To perform the task of categorizing index terms and formatting them, PRECIS uses a system of "role operators". The major operators are KEY SYSTEM, ACTION, AGENT, LOCATION, VIEWPOINT-AS-FORM, SAMPLE POPULATION/STUDY REGION and TARGET/FORM. KEY SYSTEM represents the object of an action given or implied in a statement. ACTION represents an action or the effect of an action. AGENT represents an agent who performs an action. LOCATION represents a geographical environment or setting given in a subject statement to be indexed. VIEWPOINT-AS-FORM represents a viewpoint from which the subject is examined in a document. SAMPLE POPULATION/STUDY REGION represents the scope of a reported work. Finally, TARGET/FORM represents the target at which a document is aimed and the form of the material it contains.

The role operators are represented by a set of codes. One code is assigned to each term in a string by an indexer. The statement "environment planning of new towns in Taiwan" is, for example, coded in the following form so that a computer can be used to actually generate the final index entries:

(0) Taiwan
(1) new towns
(2) environment planning

Here 0 is role operator LOCATION, 1 KEY SYSTEM and 2 ACTION. A computer is then used to generate the following index entries, where the role operators are removed:

Taiwan
new towns. environment planning

New towns. Taiwan
environment planning

Environment planning. new towns. Taiwan

Although it is not difficult to find some parallels between the operators in PRECIS and the case relations in a case grammar system, the PRECIS researchers did not consider case grammar as a theoretical foundation of their system. For example, verbal elements in PRECIS, unlike verbs in case grammar, cannot determine a case frame for an index string. It is interesting, however, that the PRECIS researchers did pursue case grammar when they realized the need of a universal, language independent theory for their multilinguistic indexing system and attempted to claim that PRECIS is essentially a case grammar system [Sorensen & Austin, 1974; Hancox & Smith, 1985]. The claim that the design of PRECIS found support in case grammar is not well-motivated as indicated by Michell [1979].

PRECIS is a string indexing system rather than a complete document retrieval system. Although the performance of PRECIS is not necessarily better than that of other indexing systems such as LCSH [Hunt & others, 1977], PRECIS does suggest and demonstrate two principles that are important for document retrieval. First, a meaningful document representation should include both index terms and the conceptual relationships between the terms. And second, it is possible to use a small set of relational indicators to accomplish the representation. Indeed these are the two basic assumptions of this study.

The goal of the Western Reserve University (WRU) indexing system is to provide access to each document from a large number of specific and generic terms and combinations of these terms. To reach this goal, a document in the WRU system is represented in the form called a "telegraphic abstract", built up by combining terms and

relational or role indicators into subphrases, combining these subphrases into phrases, and combining these phrases, in turn, into sentences and paragraphs. Therefore the key to understand this indexing system is the structure of its subphrase.

A subphrase consists of one term and one or more role indicators. A term is first encoded by means of semantic factors. The semantic factors represent relatively broad concepts and are given a four-digit code consisting of three characters with a space for interpolation of a fourth character. For example, C_RM represents ceramic industry; the space is reserved for a one letter "infix" such as A (is a), E (is made of), and the like to form a factor such as CERM. A numerical suffix can also be added to distinguish terms having identical semantic factors and infix structure. A suffix itself has no semantic significance. The term CERM.2X.METL.001, thus, represents "cermet", PAPR.010 "brittleness", and CIRS.MYTL.RANG.13X.001 "microstructure".

In a subphrase, a term is further related to paired role indicators. The following are some of these role indicators:

KOV, KWV thing:attribute
KEJ, KAM thing:process
KAM, KQJ process:agent
KAP, KAL property:effect

An example of using these roles is "KOV. *cermet*, KWV. *brittleness*", which has two subphrases separated by a comma and expresses the relation that brittleness is a property of cermet. With different subphrases, an indexer translates the theme "the possibility of changing the brittleness of cermet materials by modifying their microstructure" into the following string of symbols:

KOV.CERM.2X.METL.001,
KWV.KAP.PAPR.010,
KAL.CIRS.MYTL.RANG.13X.001.

The WRU indexing system and its procedures are very sophisticated. With this system, both specific and broad searches can be conducted by using or ignoring roles and semantic factors. The most interesting aspect of the system is its capability of expressing very fine shades of meaning. To maintain this capability, the system is forced to limit itself to one compact subject domain, namely, metallurgy. As a result, the WRU indexing system cannot be expanded into a general document retrieval system dealing with many different subjects. In terms of retrieval effectiveness, the performance of the WRU indexing system is no better than that of the Cranfield index system [Aitchison & Cleverdon, 1963], a system based on facet classification and designed to operate below maximum effectiveness for economic acceptance. The design of the WRU indexing system demonstrates again that both index terms and their associated conceptual relations are essential in making a meaningful document representation, and that this representation can be maintained using only a finite number of relational indicators.

Relational Indexing System

Farradane's relational indexing system [1980a, 1980b] is based on Guilford's psychological theory of thinking [Guilford, 1959] which was influential in the 60's [1980a, p.269]:

> The analysis of thinking shows that the mind has basically two main "mechanisms" for interconnecting concepts: association and discrimination. Each mechanism develops into three fairly well-defined stages, ... It is the nine combinations which result from the two sets of three-stage mechanisms which are the basis of the relations between concepts, and these nine relations (together with their possible negations) have been found in practice to be necessary and sufficient to express meaning in all subject fields.

These nine relations are listed in Table 3, copied from Farradane's article [1980a, p.270]. With one of these nine relations, the two concepts in a given document surrogate can be connected. When this indexing procedure is applied to every pair of concepts in the surrogate, the document can be represented in a single conceptual graph by gradually fitting triads of term-relation-term together. The retrieval function of this indexing system

can be computerized since graphs can be converted into connection matrices and a matching process can be conducted on the matrices.

		ASSOCIATIVE MECHANISMS		
		Awareness	Temporary Association	Fixed Association
DISCRIMINATORY MECHANISMS	Concurrent Conceptualization	*concurrence*	*self-activity*	*association*
	Not-distinct Conceptualization	*equivalence*	*dimensional*	*appurtenance*
	Distinct Conceptualization	*distinctness*	*action*	*functional dependence*

Table 3: Relations in Relational Indexing System

The relational indexing system emphasizes "universal representation"; that is, the indexing mechanism is not designed for the vocabulary of a particular subject field. It was claimed that the relational indexing mechanism could be applied to all different fields. The relational indexing system also emphasizes verbs and their relations to nouns. "Meaning depends very greatly on the connectives between nouns and verbs, and these connectives are the means of expressing relations" [1980a, p.268]. This study shares these two emphases but uses case relations to link index words.

The retrieval results from the relational indexing system are not better than the results from other retrieval models such as Boolean logic [Farradane, 1980b]. The major problems include the sufficiency of the nine relations in representing documents of all fields and the manner of utilization of these relations, e.g., accuracy, consistency and determination of the relations.

Document Representation Using Frames

A frame is a data structure for representing a stereotyped situation. If frames like "go-to-restaurant" can serve as a formalism for representing text type structures in memory, it should be possible to detect some sorts of frame in text. Influenced by the studies of text structure in discourse linguistics and in AI, some information scientists investigated whether frames could be derived from scientific abstracts. They hoped that such frames could be used to represent documents and ultimately to propose new retrieval models so that "users can request not only that concepts of interest occur in the retrieved documents but also that these concepts exist in the desired semantic relationships" [Liddy, 1987, p.138].

Meada [1981] investigated a hypothesized functional structure (a frame) for scientific and technical abstracts containing the functional items of *theme, method, result,* and *discussion.* With this formal structure, indexing becomes a procedure of analyzing the sentences in the abstracts in order to fill the slots of these four functional categories. An *ad hoc* procedure was developed to identify one information function for each sentence in a scientific abstract. The procedure begins with transformation of a sentence into a string consisting of function words such as "the" and "that". This string is then matched against a set of "standard" patterns, each of which corresponds to a single information function. An implicit assumption behind the procedure is that each sentence in the abstract provides information for one of these four functional items.

There were no definite conclusions from this research. Although there is preliminary evidence in support of using this frame in the abstract, it is still not clear how to apply such a frame to represent query statements, which do not necessarily share a similar structure with the scientific abstract.

Instead of assuming an existing frame with certain components in scientific abstracts, Liddy [1987] attempted to empirically derive the natural components of a

frame from document abstracts reporting on empirical work. She used the techniques of detecting a structure within a given text type developed in discourse linguistics and the research outcomes of knowledge organization in cognitive science. She designed a series of tasks to uncover various schemas. In task one, she simply asked 14 professional abstractors to list all the text components such as hypothesis and methodology in the abstracts. In the following three tasks, she asked subjects to rank and group the identified components and to label relationships between these components. She found that the abstracts describing empirical work usually contain 15 text components including methodology, findings, hypothesis, results, subjects, purpose, conclusions, etc. She also found various relationships between these components.

The question which has not been answered is whether rule-governed instantiation of abstract frame can be accomplished using lexical clues provided in the abstracts. A positive answer to this question would permit automatic instantiation of frame structure for the abstracts. Future research results might show if the frame is a workable document representation.

The study of frames has generally focused on scientific document abstracts. It is not clear at present whether the detected frame is a workable document representation for certain specific fields or for all different fields, scientific or non-scientific. Even though the document abstracts could be represented in the form of frames, this does not ensure that these frames could be used to represent query statements. A general structure of query statements, if it could be identified, might be different from that of documents. However, current document retrieval systems usually require that the two representations should be in a compatible format for the matching purpose. The investigation of an inherent functional frame in the scientific document abstracts is still in its early stage and the research findings are not adequate to support any application.

Belkin's Structural Document Retrieval Model

In the course of proposing and studying the hypothesis of the anomalous states of knowledge (ASK), Belkin and his colleagues [1982a, 1982b] developed a structural model for document retrieval. The ASK hypothesis is that "an information need arises from a recognized anomaly in the user's state of knowledge concerning some topic or situation and that, in general, the user is unable to specify precisely what is needed to resolve that anomaly" [1982a, p.62]. From the ASK hypothesis, Belkin and his colleagues attempted to seek a retrieval approach that is close to the one used in question-answering systems.

They represented a query statement and/or a document abstract in a network constructed from constrained word associations. They argued that "concepts (represented by words) which are closely associated in an individual's state of knowledge will 1) be recalled close to one another in tasks such as word association; and 2) occur in close proximity to one another in a text by that person on the specific topic" [1982a, p.68]. They evaluated the representation by interviewing the users who submitted query statements and the authors who wrote abstracts to test the acceptance of the representation.

Query statements are classified into several categories such as "well-defined topic and problem" or "topics fairly specific but problems not well defined". Corresponding to each of these categories is a special search strategy. However, both the query classification scheme and the related search strategies have not been systematically defined. The model was further elaborated by Oddy and his colleagues [1986]. But there are as yet no evaluated retrieval results available for assessing the model.

The major problem associated with the model is the assumption of representation. Physical distance between two words, i.e., the number of words between word A and word B which are of concern, is assumed to represent their semantic closeness. There is a

lack of supporting evidence for this assumption either in theory or in practice. The representation, instead of being a semantic network, is no more than a map of the physical connections of words. A structural approach to document retrieval, at least at this stage, could not follow the approach adopted in this study.

2.4. Matching Functions

The matching functions associated with the retrieval models reviewed can be categorized into two groups: exact-match and best-match. The examples of exact-match include the Boolean search in the SYNTOL system [Gardin, 1965] and the graphic and/or subgraphic match in the relational indexing system [Farradane, 1980b]. The examples of best-match include the cosine function in the SMART system [Salton, 1966] and the matching mechanism in the expert system GRANT [Cohen and Kjeldsen, 1987]. The best-match functions are also used as a means to rank retrieved documents.

Exact-match is based on a binary concept of relevance, i.e., documents are relevant or nonrelevant. Only those documents which are logically *true* to a query are retrieved in a Boolean system, and only those documents which share an identical graph or subgraph with a query are retrieved in a structural model. Exact-match is a rigid approach to document retrieval because it ignores a possible ordinal or quantitative scale for the relevance measure.

By contrast, best-match assumes that relevance is a measure that varies from zero to one. Best-match thus introduces the concept of query-document similarity. With a best-match function, it is possible to compute a similarity value between a query and every document in a database and then to rank all documents according to the similarity value. It is assumed that end-users are more likely, with this ranking, to read relevant items in an early stage of searching.

The available best-matching functions in the literature have been designed for simple

document representation schemes such as vector representation and can be expressed by a simple mathematic formula. Typical examples include Dice's coefficient, Jaccard's coefficient, the cosine coefficient, and the overlap coefficient [Rijsbergen, 1979]. Unfortunately, these best-matching functions define no relationships between index terms. These best-matching functions are therefore not suitable for measuring the closeness between two tree structures which contain index words and their case relationships. A new structural matching function which follows the principle of best-match and takes into account case relationships between index terms had to be developed for the proposed structural retrieval model.

Given two tree structures T and T', their structural similarity can be measured by calculating the *editing* cost for transforming T into T'. Editing is carried out using a number of pre-defined editing operations like insertion and deletion of a subtree. Each editing operation is associated with a non-negative cost. The transformation of T into T' is in fact made in a sequence of editing operations, each of which incurs a cost. The distance between T and T' thus is the *minimum* total cost for transforming T into T'.

This technique is called tree-mapping and has been developed in the field of pattern recognition to measure the distance between two trees. Several tree mapping algorithms are available [Selkow, 1977; Lu, 1979; Tai, 1979] and all of them share the same principle proposed in Selkow's paper. The algorithms are different only in their definitions of elementary operations. Selkow's algorithm has been used in developing the retrieval functions for this study; other algorithms are unnecessarily complicated for processing simple trees of two or three levels. The detail of Selkow's algorithm is described in Appendix A.

2.5. Summary

Lewis's study indicates the validity of representing documents and query statements in a series of case frames through human analysis and conducting retrieval by matching case frames. Young's research shows the possibility that those c frame representations could be generated by automatic means.

Case grammar and specifically case relations have a history of application in designing question-answering systems. Although some approaches developed for question-answering systems are potentially useful to document retrieval, especially natural language processing of queries and user interface design, the essential inference approach cannot be directly applied to document retrieval. It is not yet feasible for any system dealing with broad subject areas, large text databases and complicated information requests to implement an approach using inference.

The SYNTOL system and the SMART project suggest a number of guidelines for the structural approach to document retrieval, including the importance of a strong theoretic basis for system design and the ineffectiveness of pure syntactic relations in representing documents and in measuring their structural closeness. The WRU indexing system and the relational indexing system are two complete document retrieval models. Perhaps because of their conceptual problems and their complicated manual indexing procedures, the two models have never been widely accepted. However, these two systems, together with PRECIS, do suggest that a small and finite set of relations could be used to represent documents and ultimately to improve the effectiveness of document retrieval.

The investigation of inherent functional frames in scientific document abstracts is still at an early stage. A potential problem of using the frames is that query statements might not be represented in the same frames. Query statements are not necessarily expressed, for instance, in the sequence of theme, method, result, and discussion or a subset of these.

The matching function of a structural retrieval model should possess two features: it should follow the approach of best-match and it should take into account relationships between index terms. The best-match functions in the literature of document retrieval are too simple to be modified to provide such a function for structural matching. The technique of tree-mapping, however, can be used to develop a structural matching function.

Chapter Three
A STRUCTURAL MODEL FOR DOCUMENT RETRIEVAL

This chapter describes a structural document retrieval model. With this model, documents are represented in case frames, automatic indexing is realized by using the modified version of Young's algorithms, and computerized retrieval is developed using tree-to-tree editing algorithms. The function of automatic indexing has been evaluated by comparing its indexing results with those of manual indexing. In order to reduce the complexity involved in this study of the application of case relations to document retrieval, the processing to be described is at the level of words, or more precisely tokens. Word stems, synonyms, and the like are not considered at this point.

3.1. Document Representation Using Case Relations

The following definitions are used in this study:

Definition 1: A finite verb is a verb that is inflected for tense/agreement (1> He *was* late. 2> They *were* late.).

Definition 2: A nonfinite verb is a verb that is tenseless and agreementless (1> *to attend* a conference 2> *Running* is a good exercise. 3> a *devoted* person).

Definition 3: A clause is a string of words with one and only one verb (1> *He likes mathematics*. 2> *to meet the president*).

Definition 4: A finite clause is a clause that contains a finite verb (*The claim of the councils was a reasonable one.*).

Definition 5: A main clause is a finite clause that is not preceded by a

clause-introducing particle like *that*, or *whether* (*He takes no notice of her appeals* that he should rest more.).

Definition 6: A complement clause is a clause that functions as the complement of a verb, noun, adjective, etc. in a main clause (Mary claimed *that John was innocent.*). The complement clause is also known as the embedded clause.

Definition 7: A finite complement clause is a finite clause that is preceded by a clause-introducing particle (People ask the question *whether euthanasia is ethical.*).

Definition 8: A nonfinite clause is a clause that contains a nonfinite verb. The nonfinite clause functions as a nonfinite complement clause (1> Mary persuated John *to resign in October*. 2> Smith loves *driving motorcycle*. 3> The child came *shouting his name.*).

Definition 9: A sentence contains at least one main clause, which may in turn contain complement clause(s).

Definition 10: A document surrogate (e.g., abstract) or a query (e.g., a description of an information need) consists of one set of sentences.

Definition 11: A case frame is a tree-like structure in which a dominating verb determines a number of case relations R_i and each of these case relations in turn links one or more nouns or noun phrases T_{ij} to the verb:

$$\begin{aligned} \text{VERB} \; [\; & R_1 \\ & \quad (T_{1,1} \; \cdots \; T_{1,n_1}) \\ & R_2 \\ & \quad (T_{2,1} \; \cdots \; T_{2,n_2}) \\ & \cdots\cdots \\ & R_m \\ & \quad (T_{m,1} \; \cdots \; T_{m,n_m}) \;] \end{aligned}$$

The case frame is used to represent a clause, either a main clause or a complement clause. The actual case frames and the case relations to be used in the research were presented in the section of Young's study in the preceding chapter.

With these definitions, a document is represented as a set of case frames F_i each of which shares the same document identification:

$$\text{DOC-ID} \{ F_1, F_2, ..., F_t \}$$

This representation scheme assumes independence among case frames F_1 to F_m to avoid syntactic structure dependency. In reality, many of these case frames will necessarily have various semantic or structural connections with each other. In Figure 5, for example, F_u is structurally similar to F_v and semantically related to F_w. In order to have a logically integrated representation of documents as well as an efficient retrieval implementation (i.e., storage space and processing speed), a merging function MERGE has been designed which, on the basis of simple structural comparison, merges structurally identical and structurally similar case frames in the document representation. More sophisticated semantic connections between case frames are neglected at this time because merging those case frames requires specifically designed inference schemas in addition to simple structural comparison. The proposed document representation, therefore, is not in a semantically integrated form even after the merging process.

The merging process begins with comparing the two verbs of the two instantiated case frames which are under concern. If the two case frames share an identical verb, one frame is merged into another through inserting case relations and the associated noun phrases. When both frames have the same case relations, it is necessary to concatenate the two sets of cases and their noun phrases. For example, in Figure 5, F_w is merged with F_u through inserting the *time* case relation and its noun phrase and concatenating the noun phrases in the *agent* relation and the *object* relation. The result is F_x.

F_u: The janitor opened the door.

OPEN [*agent*
(janitor)
object
(door)]

F_v: The janitor opened the window.

OPEN [*agent*
(janitor)
object
(window)]

F_w: The janitor opened the window yesterday

OPEN [*agent*
(janitor)
object
(window)
time
(yesterday)]

F_x:

OPEN [*agent*
(janitor)
object
(door, window)
time
(yesterday)]

Figure 5: Examples of similar and merged case frames

3.2. Indexing Engine

The indexing engine of this case relation-based retrieval model consists of four algorithms: Classification of Words (CW), Identification of Phrases (IP), Identification of Clauses (IC), and Generation of Case Frames (GCF). CW, IP, and IC have been derived from MYRA, PAP, and CAP/II in Young's system through modifying and reconstructing; GCF is equivalent to Young's CGP. The following is a general description of the four algorithms. The technical details of modification and implementation are provided in Appendix B.

Given an English sentence, CW first scans the sentence to locate function words

(e.g., "the", "of", and "if") by consulting a dictionary of about 1,000 such words. The identified function words serve as the structural markers of the sentence. CW then fills all blank slots between the identified function words by following a list of parsing rules. If no verb is found in the sentence, CW looks for a possible verb slot and reassigns a verb marker to the position. At the end, all of the words in the sentence should have a unique class identification such as noun, verb, preposition, etc.

With these classified words, IP performs the task of identifying noun phrases, prepositional phrases, and adverb phrases. The process is rule-governed to indicate the boundaries of every identified phrase. Once IP completes its processing, the syntactic analysis proceeds to IC, identification of clauses.

IC assumes that an English sentence has a structure of either parallel or nested clauses and that at least one of these clauses is the main clause. IC also assumes that the number of clauses in an English sentence equals the number of verbs or verb phrases occurring in the sentence. With these assumptions, IC identifies clauses by going through several analytical passes. In dealing with the sentence that has the structure of nested clauses, IC first extracts nonfinite complement clauses from the sentence. IC then extracts finite complement clauses. The remaining clause thereafter is taken as the main clause of the sentence. The parallel clause structure is viewed as a special case of the nested structure.

The parsing rules involved in IC rely on the information of the physical location of the certain types of words, such as conjunctions and pronouns, before and after the position of a verb or a verb phrase. IC does not, however, use any semantic information during its processing. The processing results from IC are a set of identified clauses. Each clause has a code, indicating the original document from which it comes. The relationships between a group of clauses which are from the same sentence are kept but not actually used in this research. These relationships might be valuable in the future

experimental study that emphasizes a coherent representation of natural language sentences.

The final indexing task is to transform all identified clauses into an equal number of case frames and it is done by executing GCF, generation of case frame. Given a parsed clause, GCF selects a type of case frame according to the type of the verb in the clause. GCF then decomposes the clause into different cases such as agentive, object, and locative. It actually starts with prepositional phrases to identify the cases of locative, time, manner, and other peripheral cases. Once the peripheral cases have been identified, GCF proceeds to locate essential cases. At the end, GCF instantiates the case frame with the found cases. Unlike CW, IP, and IC, which are modified or reconstructed versions of MYRA, PAP, and CAP/II in Young's system, GCF is equivalent to Young's CGP without any change. It is one objective of this study to test the usefulness or effectiveness of the case relations adopted by Young. It should be pointed out that the title of a document may receive a default verb *vb* during parsing because the title usually lacks a verb and is not the clause defined in this study.

In order to test the capability of the indexing engine or specifically, its language parser, including CW, IP, and IC, the parsing results of the language parser were compared with the results of manual parsing that was done by different people. The capability of the machine parser was assessed by measuring the overall agreement of the two groups of parsing results.

Five doctoral students volunteered to perform a manual parsing on 25 document surrogates. Each doctoral student processed five different document surrogates. In addition, all the doctoral students parsed one particular document surrogate so that the consistency of their parsing could be assessed. Each document surrogate consisted of a title, and in most cases, an abstract. The length of the document surrogates ranged from a few words to about one hundred words. The subjects covered included education, library

science, business, engineering, and medicine. For each parsing there were three passes. In the first pass, all words were classified into the nine pre-defined classes (noun, pronoun, numeral, adjective, adverb, verb, article, conjunction, and preposition) and in the second and third passes, phrases (noun phrases and preposition phrases) and clauses were identified, respectively, and their boundaries indicated. The three passes correspond to the tasks performed by CW, IP, and IC. The consistency among the five doctoral students in their parsing of the same document was found to be close to 90 percent; that is, the number of identical clauses that all five doctoral students identified is about 90 percent of the total number of the identified different clauses. Because of this high level ot consistency, the result from manual parsing was used as the standard for assessing machine parsing. The test results are presented and discussed here. For more detailed information on parsing, refer to Appendix C.

In pass one, an error was recorded when a word was classified differently in manual and machine parsing. A fatal error occurred when a word was incorrectly classified by the machine and this word classification would mislead future parsing. Most fatal verb errors were related to phrasal verbs. For instance, the word "on" in the phrase "focus on" was identified as a preposition and this preposition would guide the parser to find a prepositional phrase following the verb "focus", instead of the expected noun or noun phrase. As to nonfatal verb errors, an instance was the clause "professionals be made acquainted with the total information process" where the word "acquainted" was classified by a doctoral student as the part of the verb but by the machine as a noun. An example of a fatal error of noun identification was classifying "increase" as a verb. This result would make the parser misinterpret the sentence structure. A non-fatal noun error was exemplified by the phrase "Soviet Union". One doctoral student parsed it as "noun noun" while the machine result was "adjective noun". The results of the first pass are presented in Tables 4 and 5.

Word Class	Error Type	Error Frequency
verb	fatal	11
	nonfatal	7
noun/adjective/pronoun	fatal	6
	nonfatal	44
adverb	nonfatal	9
preposition	nonfatal	7
Total		84

Table 4: Error Distribution of Parsing Pass One

	Manual	Overlap	Machine
No. of words classified	2322	2238	2322
Agreement		96%	

Table 5: The Result of Parsing Pass One

In pass two, one noun phrase which was not successfully identified was, for instance, "a 126 feet high concrete-faced rolled rockfill dam". The prepositional phrase "to seismic induced permanent deformations" was incorrectly identified because the word "induced" was not classified as functioning as an adjective. Adverb phrases were not checked because most of them were single word phrases and were treated as function words. The results of pass two are in Table 6 and the results of pass three, i.e., identification of clauses, are in Table 7.

The above parsing results show that the performance of the indexing engine is at an acceptable level. To complete the description of the indexing engine, two indexing

	Manual	Overlap	Machine
No. of noun or noun phrases	299	280	323
Agreement		93%	
No. of preposition phrases	298	269	294
Agreement		90%	

Table 6: The Result of Parsing Pass Two

	Manual	Overlap	Machine
No. of clauses	197	167	205
Agreement		85%	

Table 7: The Result of Parsing Pass Three

examples are shown in Figure 6 and 7. The two examples are from the retrieval experiments to be described in the next chapter. The first example is a query (Figure 6) and the second is a document abstract (Figure 7). Each example contains the original English text and its corresponding indexing results (i.e., a set of case frames). Also included is one frequent parsing mistake.

Query one: Insurance companies in U.S. use direct mail advertising.

```
{Query1, NumOfWords(7), VerbType(agentive),
        USE [ agent
                (insurance, companies)
              object
                (direct, mail, advertising)
              locative
                (U.S.) ] }
```

Figure 6: A Natural Language Query and Its Representation

Document one:

Title: Mass marketing and direct mail advertising

While continuing the methods of advertising via the media of television, radio, and ads, insurers take advantage of direct mail and indulge in mass marketing. Mass marketing can help in achieving maximum distribution of products to all people.

```
{Document1, NumOfWords(6), VerbType(vt*),
    vb* [ object
              (mass, marketing, direct, mail,
               advertising) ] }

{Document1, NumOfWords(7), VerbType(agentive),
    CONTINUING [ object
                     (advertising, methods)
                 manner
                     (television, radio, ads,
                      media) ] }

{Document1, NumOfWords(5), VerbType(agentive),
    TAKE [ agent
               (insurers)
           object
               (direct, mail, advantage) ] }

{Document1, NumOfWords(3), VerbType(reflexive**),
    INDULGE [ locative**
                  (mass, marketing) ] }

{Document1, NumOfWords(6), VerbType(agentive),
    ACHIEVING [ object
                    (products, maximum,
                     distribution)
                locative
                    (all, people) ] }

{Document1, NumOfWords(3), VerbType(reflexive),
    HELP [ agent
               (mass, marketing) ] }


*: default verb
**: a parsing mistake because of a misinterpretation
    of the word in
```

Figure 7: A Document and Its Representation

3.3. Retrieval Function

The heart of a best-match retrieval function is similarity measurement. In order to take into account the hierarchical information in case frames, the similarity measurement in this research is based on the editing cost of transforming a document representation, that is, a group of case frames, into a query representation. It includes a technique named tree mapping [Selkow, 1977].

A (rooted) tree T is either null or a set of nodes with a distinguished node *r* called the root. The remaining nodes in T are partitioned into *m* disjoint subsets, called subtrees of T, each of which is a (rooted) tree.

Two editing operations have been defined for this research, namely, insertion of a tree and deletion of a tree. A non-negative cost is associated with each operation. The transformation of tree T into tree T' is made in a sequence of editing operations, each of which incurs a cost, and the distance $\delta(T, T')$ between T and T' is the *minimum total cost of transforming T into T'*.

In this study, EDIT(T,T') denotes such an editing function, DEL(T) denotes the function that deletes a tree, and INS(T) denotes the function that inserts a tree. EDIT(T,T') may call DEL(T) and INT(T). The cost associated with DEL(T) and INS(T) is equal to the *size* of *T*, i.e., the number of nodes in the tree. The cost is based on the assumption that each word in the case frame is equally important. For example, if C_D and C_I represent the cost of deleting and inserting a tree of n ($n \geq 1$) nodes, respectively, then $C_D=C_I=n$.

Taking a case frame to be a tree and a query representation and a document representation a set of *m* and *n* trees, respectively, the distance between a query representation and a document representation is equal to:

1. when $m = n$,

$$distance = \sum^{m \text{ or } n} \delta(T, T')$$

2. when $m > n$,

$$distance = \sum^{n} \delta(T, T') + \sum^{m-n} C_I$$

3. when $m < n$,

$$distance = \sum^{m} \delta(T, T') + \sum^{n-m} C_D$$

The distance is normalized by dividing it by the sum of the sizes of the two representations so that the normalized distance D_{qd} equals zero when the two are identical and one when the two are absolutely different, that is, the two representations share no single word in any case. D_{qd} can be easily converted into a similarity value by subtracting it from one.

A retrieval function that calculates D_{qd} has been developed using the functions of EDIT(T,T'), DEL(T), and INS(T). In detail, given that a query representation consists of m case frames and a document representation n case frames and that the document representation is to be transformed into the query representation (or vice versa), the retrieval function will:

1. call the function EDIT(T,T') to compute δ(T,T') for every possible pair of case frames between the query and the document as well as the normalized δ(T,T') and store the results in a cost matrix and a corresponding normalized cost matrix of m rows and n columns, respectively.

2. repeat this step minimum[m, n] times: find the minimum entry Normalized_Cost[i,j] ($1 \le i \le m$, $1 \le j \le n$) in the normalized cost matrix, add the corresponding Cost[i,j] in the cost matrix to the Total Editing Cost of transforming the document into the query, and then remove the ith row and the jth column from the two matrices.

3. call the function DEL(T) $|m-n|$ times to delete the remaining case frames in the document representation if $m < n$, or call the function INS(T) $|m-n|$

times to insert the remaining case frames in the query representation to the transformed document representation if $m > n$, and add either the deleting costs or the inserting costs to the Total Editing Cost.

4. Normalize the Total Editing Cost by dividing it by the sum of the sizes of the two representations and subtract the normalized editing cost, that is, the distance between the query and the document from one to convert the distance into a similarity value.

To demonstrate how to use the retrieval function, an example is in order. It is assumed that

$$\text{QUERY} = [\ \text{QCsFrm}_1, \text{QCsFrm}_2\ ], \text{and}$$
$$\text{DOCUMENT} = [\ \text{DCsFrm}_1, \text{DCsFrm}_2, \text{DCsFrm}_3\ ]$$

After step one, the following two matrices are created:

$$\textit{CostMatrix} = \begin{pmatrix} 5 & 6 & 8 \\ 5 & 2 & 3 \end{pmatrix},\ \textit{NormalizedCostMatrix} = \begin{pmatrix} 0.3 & 0.4 & 0.5 \\ 0.4 & 0.1 & 0.2 \end{pmatrix}$$

With these two matrices, step two is repeated twice and the Total Editing Cost is thereafter updated to 2+5=7. In step three the retrieval function calls DEL(T) once to delete the remaining document case frame DCsFrm_3 and adds the deleting cost that is equal to the size of DCsFrm_3 to the Total Editing Cost. Finally, the retrieval function normalizes the Total Editing Cost to get D_{qd} and converts D_{qd} into a similarity value.

From this general retrieval function, a number of variant retrieval functions can be derived for various experimental purposes. First, the EDIT(T, T') function, the major component of the retrieval function that measures the distance $\delta(T,T')$ between two case frames, can be implemented in two different ways. One implementation, called Fixed-Case-Match, compares the pairs of cases which are same type so that an *agent* case is matched only to another *agent* case and an *object* case is matched only to another *object*. The advantage of designing the function EDIT(T, T') using Fixed-Case-Match is to allow further testing of Lewis's conclusion that the documents relevant to a query contain the desired keywords in case relations which match the case relations between the

keywords in the query. Another implementation, called Free-Case-Match, is a more generalized best-match approach. This second implementation allows pairs of cases of different types to be compared provided that the change of one case relation to another incurs a cost of two (one deletion and one insertion). In other words, a case relation is treated as a node in this implementation of EDIT(T, T'). Free-Case-Match is intermediate between Fixed-Case-Match and Vector-Match.

Next, step three in the general retrieval function which deals with those remaining case frames in either document or query representation could be simply removed because the remaining case frames have no value in answering a query or no chance of being answered. This process of discarding useless case frames is called the focusing process and it is expected that the modified retrieval function which incorporates the focusing process will be more effective as well as more efficient. The retrieval function with the focusing process is similar to a Boolean retrieval function in that not all of the index terms of a document are used in response to a query but only a subset of them. The focusing process may also be viewed as a weighting process that may assign a zero weight to certain case frames during retrieval.

The introduction of the focusing process to the general retrieval function does, however, require some adjustments to step four, the last step of the general retrieval function, to reflect the fact that those discarded case frames should not be involved in the final normalization process. The introduction of the focusing process also implies that in answering a query, every document in a database may have a dynamic representation tailored by the retrieval function to the query.

Based on the two different EDIT(T,T') functions and use or no use of the focusing process, four experimental retrieval functions are assembled in Table 8. The corresponding pseudo-code of the four retrieval functions are listed in Appendix D.

	Fixed Case Match	Free Case Match
Without Focusing	Fixed-Match	Free-Match
With Focusing	Fixed-Focusing-Match	Free-Focusing-Match

Table 8: Four Different Matching Functions

Chapter Four
EXPERIMENTAL DESIGN

In order to assess the level of effectiveness of the structural retrieval model developed on the basis of case relations, a comparison of different retrieval models is needed. The retrieval models which were compared in this study with the structural retrieval model are the vector space model and the Boolean model [Salton & McGill, 1983]. As in a typical document retrieval experiment [Tague, 1980], the retrieval prototypes which were programmed to implement the three different retrieval models used the same set of queries and performed indexing and matching work on the same test database of documents. At the end, recall and precision values were calculated and statistical tests were conducted to compare the different models.

4.1. Query Statements and a Test Database

The thirty queries were selected from ERIC Ontap Questions and DIALOG Ontap Questions. The ERIC search queries represent real questions assembled from questions posed to reference librarians and ERIC clearinghouse searchers [Markey & Cochrane, 1981]. All of the twenty "moderate" and "difficult" queries and two "simple" queries in ERIC Ontap were selected. The other eight queries were selected from DIALOG Ontap, which was developed for the master's course in online searching at the School of Library and Information Science, the University of Western Ontario. Nearly two thirds of the queries are incomplete sentences. This characteristic reflects that they are the queries for a Boolean system. All of the thirty queries are listed in Appendix E.

Both the ERIC Ontap file and the DIALOG Ontap file have "answer sets" for every one of their queries. The answer set is a list of documents that have been judged as being relevant to the query. The test database consisted of thirty groups of documents

corresponding to the thirty queries. Each group contained one answer set as well as those nonrelevant documents retrieved with the documents in the answer set, that is, the relevant and nonrelevant documents in each group were obtained from the same search strategy. For the purpose of experimental control, each group always contained the same number of relevant and nonrelevant documents. The document here consisted of a title and an abstract.

Since Lewis just looked at sets of documents where the keywords were the same but some documents were relevant and some were not, an ideal test database for a more objective test of Lewis's findings should contain sets of relevant and nonrelevant documents which share the same keywords. The test database described above is not an ideal one. For example, with the search strategy *(A OR B) AND C* one may retrieve two relevant and three nonrelevant documents that contain keywords *A* and *C* and one relevant document that contains keywords *B* and *C*. To create an ideal test database, only those documents that contain keywords *A* and *C* should be collected. However, in creating the test database for this study all the six retrieved documents might be included. A careful examination reveals that 14 out of 30 groups in the test database have the feature of the same keywords and thus can be used to form an ideal test database. In fact, this small ideal test database was used in a number of model comparisons.

4.2. Retrieval Prototypes

The prototype of the structural retrieval model had two major components: an indexing engine and a retrieval function. The indexing engine consisted of a natural language parser and a case frame generator which utilized parsing results and created a database of case frames for retrieval. In response to a natural language retrieval query, the retrieval function 1) processed and converted the query into a set of case frames; 2) measured the structural closeness between the query and every document in the database; and 3) presented the retrieved documents, according to their closeness to the query, in a ranked list.

Corresponding with the four retrieval functions developed in the preceding chapter, four retrieval prototypes have been programmed using PROLOG for the structural retrieval model. The four prototypes shared the same indexing engine but each maintained its own retrieval function. The four prototypes were named after the retrieval function that they used, i.e., Fixed-Match, Free-Match, Fixed-Focusing-Match, and Free-Focusing-Match.

The two simple prototypes of the vector space model were coded in the C language. The first prototype maintained one vector of nonstop-words for a document or a query without using any statistical information about the words. The second called a weighting function $Weight_{ik}=Freq_{ik}/DocFreq_k$ [Salton & McGill, 1983] to assign a weight to each word in the vector. $Freq_{ik}$ reflects the importance of a given word k in an individual document i, and $1/DocFreq_k$ reflects the usefulness of the word in the test database as a whole. For these two prototypes no vocabulary control process of any kind was employed, so that they possessed comparable information to what the four prototypes of the structural retrieval model did. The two vector prototypes shared the same cosine matching function.

All Boolean searches were conducted directly on the DIALOG system to eliminate the need to program Boolean retrieval prototypes. One search set, which was equivalent to the test database, was created first. The search result for a query was the intersection of the database set and the result set retrieved on DIALOG for that query.

Since Boolean retrieval allowed more than one search strategy to be formulated for a single query, two searches were done for each query: one recall oriented and another precision oriented. The two search strategies used for the two searches are recommended in the ONTAP manuals. The results of the two searches were kept separately and viewed as if they were from two different prototypes of the Boolean model: the Recall-oriented-Boolean and the Precision-oriented-Boolean.

4.3. Experiments

To take into account all of these indexing and retrieval variables, the ten comparisons in Table 9 have been carried out. The purpose of comparison one was to select the better prototype of two to represent the vector space model in the other comparisons. In comparison two, the selected vector prototype was compared with the two prototypes of the structural model. This second comparison was based on ranked retrieval results. Comparison three examined the performance of three different retrieval models: the vector model, the structural model, and the precision-oriented Boolean model. Unlike comparison two, this third comparison was based on nonranked retrieval results. Comparison four was identical to comparison three except that the Boolean model was recall-oriented. Comparisons five, six, and seven were replications of comparisons two, three, and four to evaluate the usefulness of the focusing function that was incorporated in the corresponding structural retrieval prototypes. Similarly, comparisons eight, nine, and ten were replications of comparisons five, six, and seven but using the small ideal test database to more objectively test Lewis's findings.

4.4. Data Analysis

Since both the proposed structural retrieval model and the vector model provided ranked document lists, the retrieval results from the prototypes of these two models were presented and compared by means of recall-precision graphs. Statistical tests were then performed to determine whether one model was superior to another.

Drawing a recall-precision graph began with converting the search results of a query into a table containing similarity values for relevant documents, their ranks and the corresponding recall and precision values. Each of these tables was sorted into descending order according to the precision value. This sorted table was then transformed into a skeletal table by interpolation. That is, the standard recall intervals

Comparison 1:	Vector-Match vs. Weighted-Vector-Match the better prototype of these two is thereafter called VECTOR
Comparison 2:	VECTOR vs. Fixed-Match vs. Free-Match
Comparison 3:	VECTOR vs. Free-Match vs. Precision-oriented-Boolean
Comparison 4:	VECTOR vs. Free-Match vs. Recall-oriented-Boolean
Comparison 5:	VECTOR vs. Fixed-Focusing-Match vs. Free-Focusing-Match
Comparison 6:	VECTOR vs. Fixed-Focusing-Match vs. Precision-oriented-Boolean
Comparison 7:	VECTOR vs. Fixed-Focusing-Match vs. Recall-oriented-Boolean
Comparison 8:	Replication of comparison five using the ideal database
Comparison 9:	Replication of comparison six using the ideal database
Comparison 10:	Replication of comparison seven using the ideal database

Table 9: The Arrangement of Ten Comparisons

(from 0.1 to 1.0 in increments of 0.1) in the skeletal table that had no precision values assigned to them and were less than or equal to the current recall value sequentially taken from the sorted table were assigned to the corresponding precision value. The overall performance of a retrieval prototype was determined by computing the average precision over all of the queries at each standard recall value. Finally, the graph could be constructed by plotting the precision values against the recall values. The position of averaged curves in the graph should indicate which model was more effective.

The Two-Way ANOVA test of repeated measure design was used to analyze ranked retrieval results with precision as dependent variable and standard recall level and

retrieval model or indexing method as independent variables. To normalize the precision values and stabilize the variances, the data were transformed using the ARCSIN function. The parametric test was chosen because of its efficiency.

The ranked retrieval results could not be directly compared to the search results from the Boolean prototypes in the way described in the previous paragraphs because the Boolean prototype provided non-ranked search output. In order to carry out comparisons, the number of retrieved documents should be constant for different retrieval prototypes. This was done by selecting a threshold in similarity values in such a way that the retrieval model which provided ranked retrieval results retrieved the same number of documents as the Boolean prototype did for the same query. The final averaged recall and precision values were compared to determine superiority. An ANOVA test, with both recall and precision as dependent variables and retrieval model as independent variable, was conducted to analyze the sets of retrieval results.

Due to the small size of the ideal test database, no statistical tests were conducted for comparisons eight, nine, and ten. The results of these three comparisons are simply presented in the form of either recall-precision graph or averaged recall and precision tables.

Chapter Five
EXPERIMENT RESULTS AND DISCUSSION

For each of the ten comparisons described in the preceding chapter, the result is first presented and then discussed. A summary table of the statistical analysis is provided at the end of this chapter as well as a general discussion about the experimental results.

5.1. The Ten Comparisons

COMPARISON ONE In comparison one are the two Prototypes of the Vector Space Model, one used word weighting and another did not. The two prototypes processed the same thirty queries on the same test database of 534 document records. The averaged recall precision graph is in Figure 8.

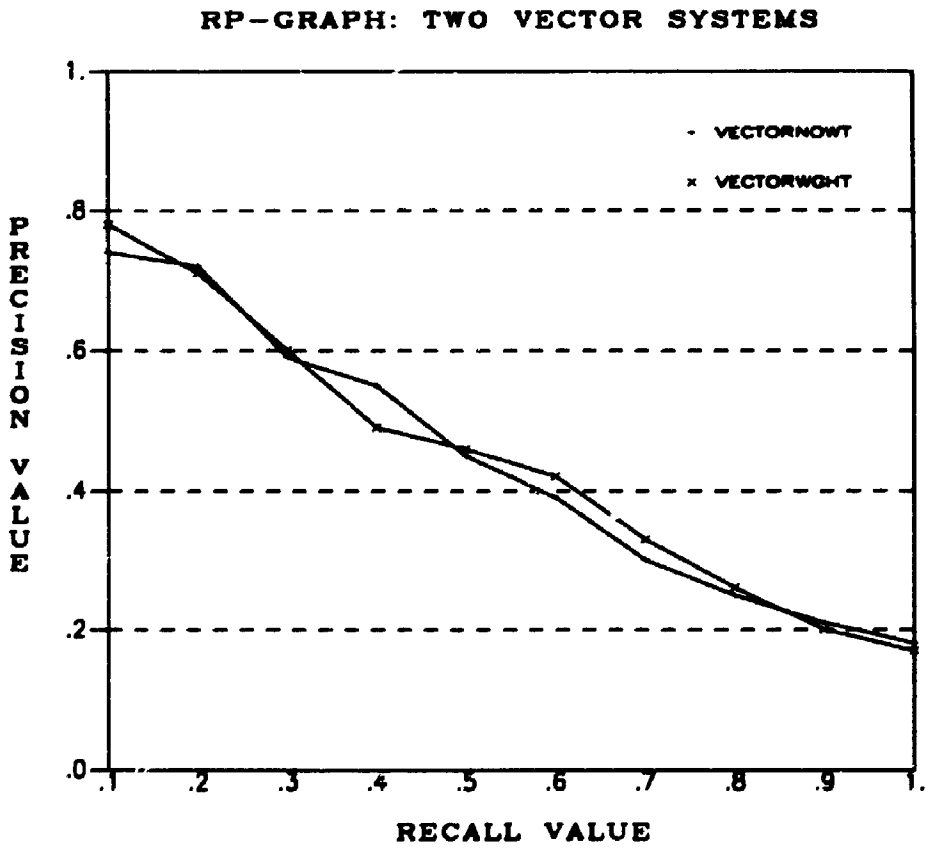


Figure 8: Recall-Precision Graph of the Two Vector Systems

As suggested in the graph, the two prototypes, Vector-Match and Weighted-Vector-

Match, performed similarly. The statistical test confirmed this visual impression: no significant difference between the two prototypes was detected. Either prototype therefore, could be used to represent the vector space model. The final selection was Weighted-Vector-Match and from now on this prototype is called VECTOR.

COMPARISON TWO In comparison two are the three retrieval prototypes: VECTOR, Fixed-Match, and Free-Match. The averaged recall and precision values are summarized in a recall precision graph (Figure 9). Although the ANOVA test of repeated measures indicated that the three prototypes were comparable to each other, the graph suggested that the vector space model was more effective than either implementation of the structural model. The details of the statistical tests are presented in Table 10.

The results obtained in this comparison do not support Lewis's finding that when a relevant document shares words with the query for which the document is retrieved, these words will be related to each other by similar sets of case relations, because the vector model, which incorporates no case relations, looks more effective than the structural model. The next question to be answered is why the vector space model, which incorporates no word relationships of any kind, achieved better retrieval results in this comparison. A number of possible interpretations can be provided in comparing the structural retrieval model with the vector space model.

The two models share the idea of best-matching. Because of this, the structural model in theory should inherit all of the retrieval features provided by the vector space model. However, the structural model also differs from the vector space model in document representation, indexing, and matching. With the vector space model, document representation is simple: one vector or list of words with or without frequency information represents each document. All words have an equal chance of being matched to words in another vector when the two vectors are compared. In the structural model,

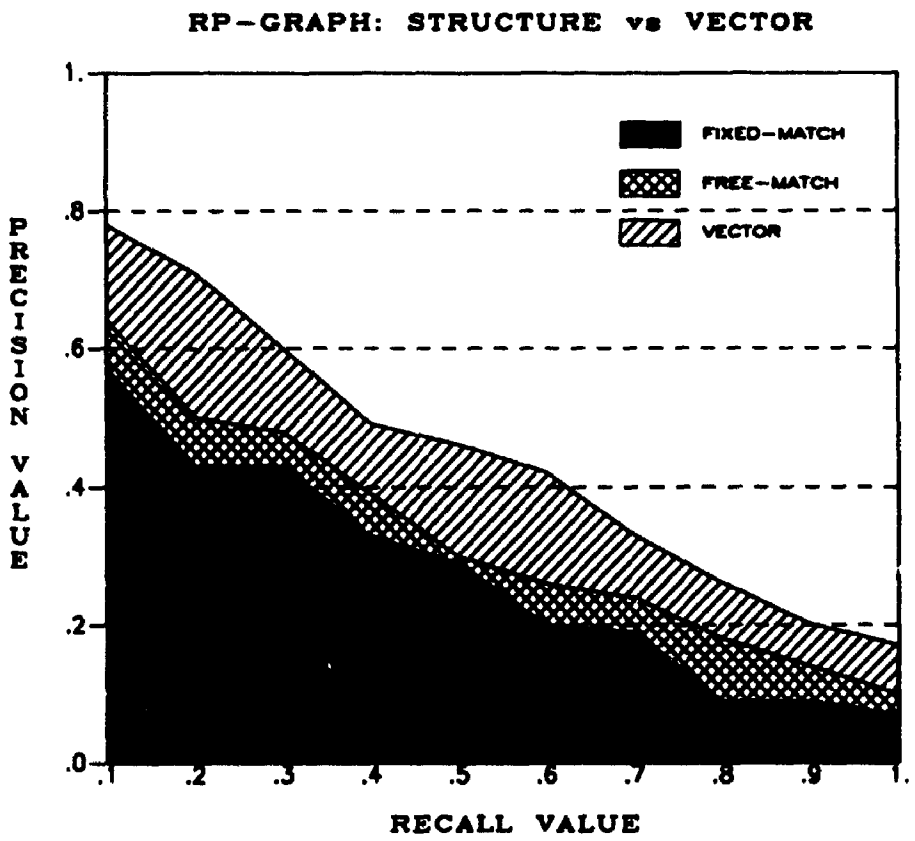


Figure 9: Recall-Precision Graph of VECTOR and Two Structural Models

however, each document is represented by a set of case frames and the words in each case frame are stored in different cases. The representation is a structural hierarchy. This hierarchical system forces a matching function to match words in a case frame differently.

In the vector space model, the unit for matching is the word. The more words a document vector shares with a query vector, the closer the document is topically to the query. In the structural model, the matching unit is the case frame, which represents a natural language clause. Two case frames are compared at a time and their topic closeness depends on the distribution of words over cases. When a word shared by two case frames appears in the same case, its retrieval value is appreciated in both

VECTOR vs. FREE-MATCH vs. FIXED-MATCH

MANOVA Tests involving 'MODEL' Within-Subject Effect

EFFECT .. MODEL
Multivariate Tests of Significance (S=1,M=0,N=13)

TestName	Value	ExactF	Hypoth.DF	ErrDF	Sig.of F
Hotelling	.02584	.36173	2.00	28.00	.700

Note.. F statistics are exact.

Tests involving 'MODEL' Within-Subject Effect.
AVERAGED Tests of Significance for MEAS.1

Source of Variation	SS	DF	MS	F	Sig of F
WITHIN CELLS	1.75	58	.03		
MODEL	.02	2	.01	.38	.683

Table 10: ANOVA Test In Comparison Two

Fixed-Match and Free-Match. When the word shared by the two case frames appears in two different cases, Fixed-Match negatively appreciates its retrieval value and Free-Match matches the word with a penalty. In comparing a query and a document, similarly, the distribution of words over two sets of case frames also affects the measurement of their topic closeness.

The words in the structural retrieval model are evaluated according to their associated case relations since it was expected that the case frame representation of documents would improve retrieval effectiveness. However, this expectation received no support from the current comparison. One explanation would be that many words missed their opportunity of being matched because of the reasons described in the preceding paragraph and thereafter there was no way for the structural prototypes to compete with the vector space model. One remedy for improving the performance of the structural retrieval model is to modify Fixed-Match and Free-Match so that those nonmatched words would have a second chance to be matched; however, this practical approach

deviates from the emphasis of case relations and their potential value in document retrieval. Another suggestion is to introduce the previously described focusing process to Fixed-Match and Free-Match. The usefulness of the focusing process will be discussed in comparison 5.

COMPARISON THREE In comparison three are the three retrieval prototypes of VECTOR, Free-Match, and Precision-Oriented-Boolean. The test results are presented in Table 11 and Table 12. The tests indicated that the Precision-Oriented-Boolean prototype was more effective than the structural prototype in creating a final retrieval set that contained a high percentage of relevant documents. The three prototypes, however, had a similar capability in performing exhaustive retrieval to locate all relevant documents of a query.

	Precision		Recall	
Models	Mean	Std. Dev.	Mean	Std. Dev.
P-O-Boolean*	0.794*	0.534	0.359	0.324
Free-Match*	0.503*	0.482	0.233	0.205
VECTOR	0.563	0.400	0.292	0.221

* the difference between these two is statistically significant

Table 11: Precision and Recall Values in Comparison Three

The better performance of Precision-Oriented-Boolean might not result from the Boolean model itself but from the procedure that converted a ranked list into a set for comparison. Salton [Salton, 1972] has pointed out that this procedure favors the Boolean model. The number of retrieved documents from a Boolean system is not predicable and a threshold determined by this random number has no justification in a retrieval model which provides a list of ranked documents. The procedure becomes more biased when the size of a final retrieval set is either very small or every large. In this comparison, there

1. P-O-Boolean vs. Free-Match vs. Vector
EFFECT .. MODEL
Multivariate Tests of Significance (S=2,M=-1/2,N=42)

TestName	Value	Approx.F	Hypoth.DF	ErrorDF	Sig.ofF
Hotelling	.15238	3.23802	4.00	170.00	.014

Note.. F statistic for WILK'S Lambda is exact.

Univariate F-tests with (2,87) D. F.

Variable	Hypoth.SS	ErrSS	Hypoth.MS	ErrMS	F	Sig.ofF
PRECISION	2.5894	21.4744	1.2947	.2468	5.2453	.007
RECALL	.2614	5.8314	.1307	.0670	1.9504	.148

2. P-O-Boolean vs. Free-Match
EFFECT .. MODEL
Multivariate Tests of Significance (S=1,M=0,N=27.5)

TestName	Value	Exact.F	Hypoth.DF	ErrorDF	Sig.ofF
Hotelling	.17183	4.89710	2.00	57.00	.011

Note.. F statistics are exact.

Univariate F-tests with (1,58) D. F.

Variable	Hypoth.SS	ErrSS	Hypoth.MS	ErrMS	F	Sig.ofF
PRECISION	2.2814	16.5573	2.2814	.2855	7.9918	.006
RECALL	.2609	4.3881	.2609	.0757	3.4484	.068

3. Free-Match vs. Vector
EFFECT .. MODEL
Multivariate Tests of Significance (S=1,M=0,N=27.5)

TestName	Value	Exact.F	Hypoth.DF	ErrorDF	Sig.ofF
Hotelling	.02694	.76777	2.00	57.00	.469

Note.. F statistics are exact.

Univariate F-tests with (1,58) D. F.

Variable	Hypoth.SS	ErrSS	Hypoth.MS	ErrMS	F	Sig.ofF
PRECISION	.0754	12.2662	.0754	.2115	.3564	.553
RECALL	.0551	2.6723	.0551	.0461	1.1950	.279

Table 12: ANOVA Tests In Comparison Three

were three cases in which the final set contained only one relevant document. In these cases, the Boolean model received a perfect precision value while other models suffered. No definite conclusion from comparison three is made at this point.

COMPARISON FOUR In comparison four are the three prototypes of VECTOR, Free-Match, and Recall-oriented-Boolean. As in comparison three, the results are summarized in tables (Table 13 and Table 14). Unlike the previous comparison, no statistically significant difference was found, so the three retrieval prototypes performed at a comparable level of effectiveness.

	Precision		Recall	
Models	Mean	Std. Dev.	Mean	Std. Dev.
R-O-Boolean	0.347	0.397	0.658	0.363
Free-Match	0.213	0.178	0.716	0.453
VECTOR	0.274	0.222	0.810	0.458

Table 13: Precision and Recall Values in Comparison Four

```
EFFECT .. MODEL
Multivariate Tests of Significance (S=2,M=-1/2,N=42)

TestName  Value  Approx.F  Hypoth.DF  ErrDF  Sig.ofF

Hotelling .08674 1.84316   4.00      170.00   .123
Note.. F statistic for WILK'S Lambda is exact.

Univariate F-tests with (2,87) D. F.

Variable Hypoth.SS ErrSS Hypoth.MS ErrMS  F  Sig.ofF

PRECISION .2928  7.2133   .1464   .0829  1.7660 .177
RECALL    .7968 16.9708   .3984   .1950  2.0425 .136
```

Table 14: ANOVA Tests In Comparison Four

COMPARISON FIVE In comparison five are the three retrieval prototypes of VECTOR, Fixed-Focusing-Match, and Free-Focusing-Match. The averaged recall and precision values are summarized in the recall precision graph (Figure 10).

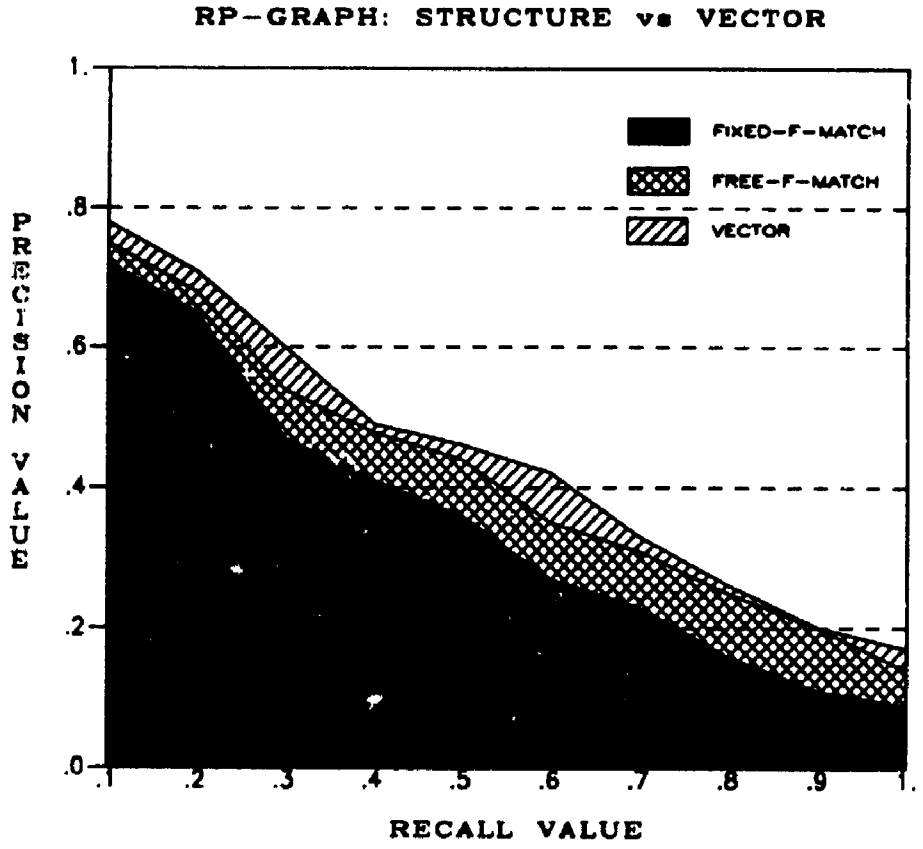


Figure 10: Recall-Precision Graph of VECTOR and The Structural Models With Focusing

Again, an ANOVA test of repeated measures was conducted to complete this comparison. The test suggested that the three retrieval prototypes were comparable to each other in their retrieval effectiveness. Although the introduced focusing process did help Fixed-Focusing-Match and Free-Focusing-Match in achieving their improved retrieval performance, the vector model still seemed more effective than the structural model. A detailed description of the statistical test is provided in Table 15.

```
VECTOR vs. FREE-FOCUS-MATCH vs. FIXED-FOCUS-MATCH

MANOVA Tests involving 'MODEL' Within-Subject Effect

EFFECT .. MODEL
Multivariate Tests of Significance (S=1,M=0,N=13)
TestName  Value  ExactF  Hypoth.DF  ErrorDF  Sig.ofF

Hotelling .06613  .92576    2.00      28.00     .408
Note.. F statistics are exact.

Tests involving 'MODEL' Within-Subject Effect.
AVERAGED Tests of Significance for MEAS.1
Source of Variation  SS  DF  MS    F  Sig of F

WITHIN CELLS       2.22  58  .04
MODEL               .07  2   .03  .91   .407
```

Table 15: ANOVA Test In Comparison Five

COMPARISON SIX In comparison six are the three prototypes of VECTOR, Free Focusing-Match, and Precision-oriented-Boolean. The statistical details are in Table 16 and Table 17. The test results from this comparison indicated that the three retrieval prototypes were functioning at a similar level of effectiveness. In terms of absolute values of the averaged precision and recall, the prototype of precision-oriented-Boolean was the best and the remaining two were comparable to one another.

	Precision		Recall	
Models	Mean	Std. Dev.	Mean	Std. Dev.
P-O-Boolean	0.794*	0.534	0.359	0.324
Free-F-Match	0.565*	0.479	0.328	0.328
VECTOR	0.563	0.400	0.292	0.221

* the statistical difference between the two is marginally significant

Table 16: Precision and Recall Values in Comparison Six

1. P-O-Boolean vs. Free-Focusing-Fatch vs. Vector
EFFECT .. MODEL
Multivariate Tests of Significance (S=2,M=-1/2,N=42)

TestName	Value	Approx.F	Hypoth.DF	ErrDF	Sig.ofF
Hotelling	.09821	2.08688	4.00	170.00	.085

Note.. F statistic for WILK'S Lambda is exact.

Univariate F-tests with (2,87) D. F.

Variable	Hypoth.SS	ErrSS	Hypoth.MS	ErrMS	F	Sig.ofF
PRECISION	2.0170	21.3487	1.0085	.2454	4.10986	.020
RECALL	.0764	7.8288	.0382	.0899	.42429	.656

2. P-O-Boolean vs. Free-Focusing-Match
EFFECT .. MODEL
Multivariate Tests of Significance (S=1,M=0,N=27.5)

TestName	Value	Exact.F	Hypoth.DF	ErrorDF	Sig.ofF
Hotelling	.09172	2.61415	2.00	57.00	.082

Note.. F statistics are exact.

Univariate F-tests with (1,58) D. F.

Variable	Hypoth.SS	ErrSS	Hypoth.MS	ErrMS	F	Sig.ofF
PRECISION	1.4980	16.4316	1.4980	.2833	5.2876	.025
RECALL	.0166	6.3855	.0166	.1101	.1508	.699

3. Free-Focusing-Match vs. Vector
EFFECT .. MODEL
Multivariate Tests of Significance (S=1,M=0,N=27.5)

TestName	Value	Exact.F	Hypoth.DF	ErrorDF	Sig.ofF
Hotelling	.00468	.13345	2.00	57.00	.875

Note.. F statistics are exact.

Univariate F-tests with (1,58) D. F.

Variable	Hypoth.SS	ErrSS	Hypoth.MS	ErrMS	F	Sig.ofF
PRECISION	.0001	12.1405	.0001	.2093	.0007	.979
RECALL	.0217	4.6697	.0217	.0805	.2694	.606

Table 17: ANOVA Tests In Comparison Six

COMPARISON SEVEN In the seventh comparison are the three retrieval prototypes of VECTOR, Free-Focusing-Match, and Recall-oriented-Boolean. The test results in Table 18 and Table 19 again indicated that the three retrieval prototypes were functioning at similar levels of effectiveness. But according to the absolute values of the averaged precision and recall, the three models represented by the three prototypes should be ranked in the following two orders respectively: 1) the recall oriented Boolean model, the structural model and the vector space model, and 2) the vector space model, the structural model, and the recall oriented Boolean model. In either order, the structural model is in the middle.

	Precision		Recall	
Models	Mean	Std. Dev.	Mean	Std. Dev.
R-O-Boolean	0.347	0.397	0.658	0.363
Free-F-Match	0.299	0.332	0.771	0.448
VECTOR	0.274	0.222	0.810	0.458

Table 18: Precision and Recall Values in Comparison Seven

```
EFFECT .. MODEL
Multivariate Tests of Significance (S=2,M=-1/2,N=42)

TestName  Value  Approx.F  Hypoth.DF  ErrDF  Sig.ofF

Hotelling .05881 1.2498     4.00      170.00     .292
Note.. F statistic for WILK'S Lambda is exact.

EFFECT .. MODEL (Cont.)
Univariate F-tests with (2,87) D. F.

Variable Hypoth.SS ErrSS Hypoth.MS ErrMS  F  Sig.ofF

PRECISION .0891   3.6046  .0446  .1104  .40359 .669
RECALL    .8230  16.8131  .4115  .1933 2.12925 .125
```

Table 19: ANOVA Tests In Comparison Seven

COMPARISONS EIGHT, NINE, AND TEN The final three comparisons were replications of comparisons five, six, and seven using a small ideal test database so that Lewis's findings could be more objectively tested. The results of comparison eight are summarized in Figure 11; the results of comparisons nine and ten are presented in Tables 20 and 21, respectively.

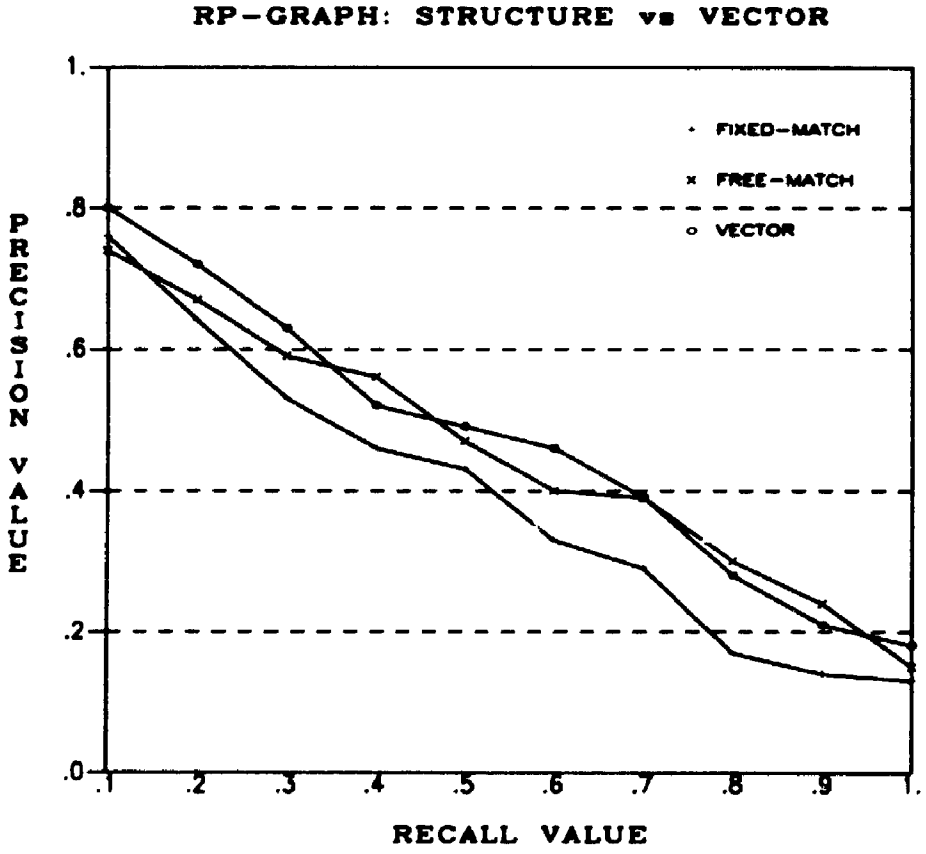


Figure 11: Recall-Precision Graph of VECTOR and The Structural Models With Focusing

The results of comparison eight indicate improved performance of the structual model, especially the performance of the FIXED-MATCH prototype. However, this observed improvement does not look significant. The results of comparisons nine and ten remain relatively similar to those of comparisons six and seven, respectively. Overall, the results of these three comparisons indicate that the three retrieval models are comparable to one another in terms of retrieval effectiveness. The results suggest that the simplified indexing process used in this study might not be as effective as Lewis's indexing process.

	Precision		Recall	
Models	Mean	Std. Dev.	Mean	Std. Dev.
P-O-Boolean	0.781	0.256	0.363	0.293
Free-F-Match	0.556	0.367	0.304	0.284
VECTOR	0.548	0.280	0.239	0.162

Table 20: Precision and Recall Values in Comparison Nine

	Precision		Recall	
Models	Mean	Std. Dev.	Mean	Std. Dev.
R-O-Boolean	0.401	0.343	0.616	0.210
Free-F-Match	0.347	0.301	0.649	0.285
VECTOR	0.336	0.204	0.705	0.267

Table 21: Precision and Recall Values in Comparison Ten

The statistical results of the ten comparisons are summarized in Table 22. In this Table, the symbol ">" stands for "better than".

5.2. A General Discussion

The graphs, the averaged recall and precision values, and the statistical tests in the above ten comparisons have demonstrated that the structural retrieval model does not appear to be more effective than either the vector space model or the Boolean model. The structural model, at its best, can manage a comparable retrieval performance to the other two models. Its unsatisfactory performance perhaps could be attributed to the indexing engine.

The indexing process involved in the vector space model and the Boolean model is an error-free process, as far as words or tokens (not stems, synonyms) are concerned. At

Models	Observation	Statistically significant Difference
1. Vector (no weighting) vs Vector	comparable	No
2. Fixed-Match vs Free-Match vs Vector	Vector > Free-Match Free-Match > Fixed-Match	No
3. P-O-Boolean vs Free-Match vs Vector	P-O-Boolean > Vector Vector > Free-Match	Yes
P-O-Boolean vs Free-Match	P-O-Boolean > Free-Match	Yes
Free-Match vs Vector	Vector > Free-Match	No
4. R-O-Boolean vs Free-Match vs Vector	comparable	No
5. Vector vs Fixed-F-Match vs Free-F-Match	Vector > Free-F-Match Free-F-Match > Fixed-F-Match	No
6. P-O-Boolean vs Free-F-Match vs Vector	P-O-Boolean > Free-F-Match Free-F-Match > Vector	Yes (marginally)
P-O-Boolean vs Free-F-Match	P-O-Boolean > Free-F-Match	Yes (marginally)
Free-F-Match vs Vector	Free-F-Match > Vector	No
7. R-O-Boolean vs Free-F-Match vs Vector	comparable	No
8. Vector vs Fixed-F-Match vs Free-F-Match	comparable	
9. P-O-Boolean vs Free-F-Match vs Vector	comparable	
10. R-O-Boolean vs Free-F-Match vs Vector	comparable	

Table 22: The Results of the Ten Comparisons

the end of indexing, all words that are not in a stop-list are organized into an index file

and will be matched to query words. Associated with the structural retrieval model, however, is a certain level of indexing error because of natural language parsing. No perfect parser is available at this time, though efforts could be made to reduce parsing error to a minimum level. As indicated earlier, the parser developed on the basis of the case relations for this study may mis-identify approximately fifteen percent of processed natural language clauses. About fifteen percent of the case frames in the index file of the structural model thus may have ill-formed structures and incorrect distributions of words. Definitely these errors caused the unsatisfactory performance of the structural retrieval model.

Furthermore, the indexing process used in this study should follow the indexing process used in Lewis's study in order to objectively test her findings. Because of the limitations of natural language processing, the indexing process in this study incorporated no sophisticated semantic processes such as the text reduction found in Lewis's study. In fact, the indexing process did no more than map natural language clauses into case frames, with the hope that this indexing would reach the level of usefulness suggested by Lewis. Although the results of comparisons eight, nine, and ten indicate certain effectiveness of the simplified indexing process, the results of this study as a whole suggest that the simplified indexing process might not be as effective as Lewis's indexing process.

It is also possible that the performance of the structural model partially reflected the system of case relations employed in this study. Since linguistic scholars have not reached any consensus on an ideal case relation system, Young's system is not necessarily the best system. The decision to use Young's system in this research was more for convenience of implementation and the overall coherence of the structural model than for theoretical soundness.

Finally there is perhaps one external factor that might in some way be responsible for

the unsatisfactory performance of the structural model. This is the characteristics of the thirty query statements. Although these queries are from real information users, they are tailored to the Boolean retrieval system. More than fifty percent of the query statements are not complete sentences. The incomplete sentence with no verb forced the indexing engine of the structural model to assign a default case frame structure to it. Obviously the default case frame was not as informative as the case frame instantiated through a normal indexing procedure. The use of default case frame also increased the likelihood of error in case assignment.

Overall, the experimental results suggested that the structural retrieval model was not more effective than the two other retrieval models. Replications of this study with a more reliable indexing engine, however, are needed before one can make the statement that case relations cannot significantly improve document retrieval effectiveness.

Chapter Six
CONCLUSIONS

Document retrieval deals with the capture, storage, and retrieval of natural language texts, which range from short passages and bibliographic records to full text documents. There are two major approaches in this research field: keyword indexing and structural indexing. The first uses only keywords to indicate the subject content of documents; the second considers both keywords and their associated relationships in representing documents. In general, keyword indexing has been characterized as an automatic and domain independent approach to document retrieval and the structural indexing as a manual and domain dependent approach.

Two examples that follow the keyword indexing approach are the Boolean retrieval model and the vector space model. This approach has dominated the research on document retrieval since the SMART project commenced. Many significant research results have been obtained since then. The structural indexing approach, though it may be able to more accurately represent documents, has not yet proved to be better. A bottleneck is how to program a computer to connect document keywords with a list of defined relations, including synonym relations, syntactic relations, and semantic relations. Consequently most of the proposed structural models for document retrieval are domain dependent and incorporate a complicated manual indexing system. In dealing with a large general document database, structural models such as the WRU indexing system and the relational indexing system are in no position to compete with simple computerized keyword systems. However, researchers in the field of document retrieval have never stopped developing different retrieval models involving the structural approach. This study is another attempt to develop a structural model for document retrieval.

This study concerns the use of case relations and structural document representation, case relation-based natural language parsing and computerized structural indexing, and tree mapping techniques and structural matching of documents in a structural document retrieval model. The use of case relations in developing retrieval models was recommended by Lewis [Lewis, 1984] once she found the regularity of language behavior among a group of information users who were scientists and scholars. Specifically, if the keywords in relevant documents are linked with each other through case relations in the ways described in the query for which the documents are retrieved, the likelihood of relevance increases.

In designing such a document retrieval model, there were two major challenges: structural indexing and structrual matching. The structural indexing was developed by modifying and restructuring Young's indexing engine that was designed exclusively for automatic indexing. The developed indexing engine consists of a natural language parser and a case frame generator. The generator utilizes parsing results and creates an index file for retrieval. However, the indexing engine is associated with a certain level of parsing error.

The structural matching function developed on the basis of Selkow's algorithm has been successfully implemented as a means for both matching and ranking documents. For the designed experiments, the structural matching function took into account both index words and their associated case relations. A similar matching function can be developed for other structural retrieval models.

The structural retrieval model has been compared with the vector space model and the Boolean model in terms of retrieval effectiveness. The test results are encouraging but unsatisfactory. It is encouraging because the structural model is, under some of the experimental conditions, statistically comparable to the two other models in locating relevant documents for the queries. It is unsatisfactory because the structural model is no

better than the two other models. In fact, the recall-precision graphs and/or the averaged recall and precision values suggests that the structural model is most frequently ranked as the least effective model among the three compared retrieval models. It is possible, however, that this unsatisfactory performance at this point perhaps could be attributed to the problems associated with the indexing engine.

To improve the indexing engine, its two major components, the syntactic parser and the case frame generator, have to be upgraded. There are two ways to upgrade the parser: enhancement of the present parser or construction of a new parser on a available research parser. The current parser could be improved by expanding its dictionary and modifying and adding related parsing rules. As discussed earlier, most parsing errors and most fatal errors are verb-related. An expanded dictionary with more verbs and verb phrases may prevent the occurrence of certain parsing errors. The different forms of a verb could be treated as different entries in the dictionary so that the various forms of the verb (e.g., tensed, gerund, participle) could be easily determined. Other common phrases such as prepositional phrases and adverb phrases could be incorporated into the dictionary as well. A mechanism of morphological control could be introduced to map words into their stems. Other innovative suggestions should also be considered in future.

The idea of building a new parser is suggested by the available research products developed by computational linguists over many years. Among various research parsers are those based on augmented transition network (ATN) model which are particularly important because an ATN parser is able to decide if an English sentence is grammatical or not and to classify the sentence structures. Once the structural information of the sentence is obtained it could be used by a case frame generator to perform other indexing tasks. The ATN parser is also easy to implement.

To upgrade the case frame generator, a different system of case relations could be explored for a better indexing engine. Two major difficulties are involved in this

approach. First, theoretical criteria for selecting one set of case relations over another are not available. Second, the work of developing rules for case assignment could turn out to be tedious and difficult. Recent developments in case grammar research should be consulted before making any project decision [Starosta, 1988].

The indexing engine, no matter how well it could be refined, is a simplified implementation of the indexing process used in Lewis's study. Because of this, the results of this study alone are not sufficient for either proving or disproving Lewis's conclusion that case relations are helpful in improving retrieval effectiveness. The results of this study merely suggest that the simplified indexing process may not be as effective as the one that Lewis used and that the structural retrieval model based on this simplified indexing process could reach a level of effectiveness no better than that of other document retrieval models. Replications of this study with a more reliable indexing engine are needed to collect additional evidence as to whether case relations are valuable in document retrieval. One thing that is certain, however, is that the price paid for different indexing features that the structural model can incorporate is increased processing cost. Without strong evidence that the structural document retrieval model is significantly more effective than other retrieval models, concerns of cost/effectiveness alone will prevent such a model from being further developed.

Appendix A
DISTANCE BETWEEN TWO TREES

There are several equivalent definitions available for the rooted tree each stressing a different facet of a tree structure. A (rooted) tree T in Selkow's algorithm is either null or a set of nodes with a distinguished node *r* called the root. The remaining nodes in T are partitioned into *m* disjoint subsets, called subtrees of T, each of which is a (rooted) tree.

This definition is recursive: it defines a tree in terms of subtrees which are themselves trees. The definition places emphasis on the nodes in a tree: the edges are implied. For example, (*r*, *n*) is an edge in T if *n* is the root of a subtree of T. A complete (recursive) definition of edges in terms of nodes can easily be developed: the definition of tree is indeed equivalent to the definition of tree based on graph theory, that is, a rooted tree is a connected, directed graph in which each node has one predecessor, except a unique node called the root which has no predecessor.

Denoting by $\lambda(v)$ the label of a general node, with $v \in T$, $\lambda(T)$ the label of the root of T and T<i> the tree obtained by removing the subtrees T_{i+1}, ..., T_m (so that T<m> = T) we say that two trees A, B are equal (written A = B) if

1. $\lambda(A) = \lambda(B)$
2. if A_1, ..., A_n and B_1, ..., B_m are the subtrees of A and B respectively, m=n and $A_i=B_i$ for $1 \le i \le m$

The two operations used for this thesis study on labelled trees are defined as follows.

The operation, written I(A) and called **insertion of subtree**, is applied to T at the index position *i* (1<*i*<m) and transforms T into T' where

$$\lambda(T') = s_j$$

T_1, ..., T_i, A, T_{i+1}, ..., T_m are the subtrees of T'

The operation, written $D(T_i)$ and called **deletion of subtree**, transforms T into T' where

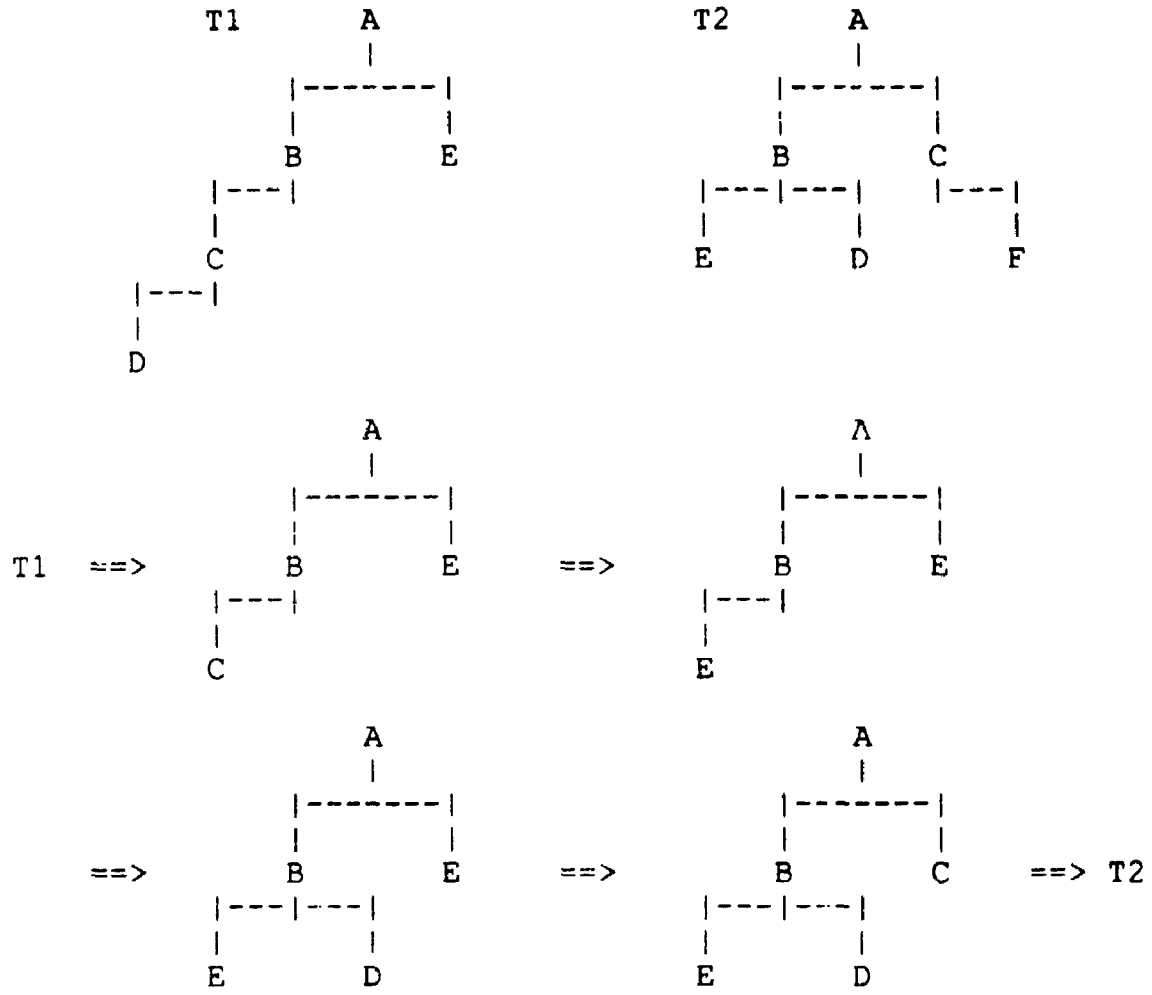
$$\lambda(T') = s_j$$
$$T_1, ..., T_{i-1}, T_{i+1}, ..., T_m \text{ are the subtrees of } T'$$

A non-negative cost is associated with each operation. For the two defined operations it is first necessary to assign the cost of deleting a tree of only a single node labelled by s_i, say $C_D(s_i)$, and that of insertion of this tree, say $C_I(s_i)$. The cost of I(A) and D(A) are then respectively

$$C_I(A) = \sum_{v \in A} C_I(\lambda(v))$$
$$C_D(A) = \sum_{v \in A} C_D(\lambda(v))$$

Thus the transformation of one tree into another is made in a sequence of elementary operations, each of which incurs a cost, and the **distance** $\delta(A, B)$ between two labelled trees A and B is the minimum total cost of transforming one into the other.

An example is given on the next page to illustrate the process of the algorithm. The details of the proof of the theorem, the algorithm itself, and the complexity analysis are given in the original article.



Given $C_I = C_D = 1$, the transformation cost or the distance is 7 for this example. The similarity of these two trees can be calculated by

$$S = 1 - \frac{Total\ \ Cost}{C_D(T_1) + C_I(T_2)} = 0.36$$

Appendix B
IMPLEMENTATION OF THE INDEXING ENGINE

The indexing engine consists of a natural language parser and a case frame generator. The four major components of the natural language parser are the dictionary, classification of words (CW), identification of phrases (IP), and identification of clauses (IC). The dictionary is an expanded version of the dictionary compiled by Young, and CW, IP, and IC are corresponding to the three algorithms MYRA, PAP, and CAP/II in Young's system. CW and IP are the modified versions of MYRA and PAP, respectively. IC is, however, a new algorithm and is almost entirely different from CAP/II in the approach taken to locate individual clauses. The case frame generator is identical to Young's case grammar program (CGP); no change has been made in this program at this point.

This appendix focuses on the implementation of the natural language parser since it has been revised and upgraded for this study. The description is in the sequence of dictionary, CW, IP, and IC. The dictionary is presented as lists of function words. The parsing algorithms are described in the form of pseudo-coded parsing rules. Cross references are provided for those rules which are either copied or modified from Young's system. The description and the related discussion assume a familiarity with Young's indexing system.

The Dictionary

The dictionary consists of 24 different groups of function words. Every group of the function words is listed in this section. All of the function words included are single words; phrasal terms are not covered at this time. This decision affects mostly the two

important function word groups: verb and preposition. In dealing with text within which phrasal verbs (e.g., "focus on") and non-single-word prepositions (e.g., "as to", "from behind", "according to") occur frequently, the capability of the dictionary is limited and the misidentified phrasal terms may cause misinterpretations of sentence structures.

In terms of the relationships between the 24 groups of function words and the conventional classes of English words found in English grammar, there is no simple parallel. In many cases, more than one group of function words can be mapped to a single grammatical word class. These m-to-1 relationships are indicated in the following lists of function words. The group identifications of function words are in boldface and the corresponding names of grammatical classes are in italic. The descriptive definitions of the conventional classes of English words are listed in Appendix C.

1. CCN CCP (*Coordinative Conjunction*)

and, but, or, nor, plus, versus, vs

CCP is the same as CCN. The distinction is made on the basis of sentence structure.

2. EOS (*End of Sentence*)

. ? !

3. PNT (*Punctuation*)

, ; : ' " ( )

4. NEG (*Ordinary Adverb*)

not

5. PRP (*Simple Preposition*)

about, above, according, across, after, against, along, among, around, at, before, behind, below, beneath, between, beyond, by, despite, during, except, for, from, in, inside, instead, into, of, off, on, out, outside, over, regarding, through, throughout, to, toward, under, until, up, upon, via, with, within, without

Note that compound prepositions, double preposition, and phrasal preposition are not included in this list.

6. SCN *(Subordinative Conjunction)*

although, because, however, if, since, than, then, therefore, though, thus, unless, whether, yet

7. THR

there

8. THT# THT

that

The word "that" is initially seen as the left boundary of a clause (THT#); its final class identification is determined during the parsing process. Note that the suffix '#' always indicates a clause boundary.

9. AJN

did, do, does, get, gets, got, keep, keeps, kept, let, lets

10. AUX *(Auxiliary Verb)*

am, are, be, been, being, had, has, have, having, is, was, were

11. MOD *(Modal Verb)*

can, cannot, could, may, might, must, shall, should, will, would

12 VRB *(Verb)*

Irregular verbs: abide, abode, abided, arise, arose, arisen, awake, awoke, awaked, bear, bears, bore, born, beat, beats, beaten, became, become, becomes, befall, befell, befallen, beget, begot, begotten, began, begin, begins, begun, behold, beheld, bend, bent, bereave, bereft, beseech, beset, besought, bespeak, bespoke, bespoken, bespread, bestrew, bestrewn, bestride, bestrode, bestridden, bet, betake, betook, betaken, bethink, bethought, bade, bidden, bide, bode, bind, bound, bit, bitten, bleed, bled, blend, blent, bless, blest, blow, blew, blown, break, breaks, broke, broken, breed, bred, bring, brought, broadcast, browbeat, browbeaten, build, builds, built, burn, burnt,

burst, buy, bought, cast, catch, catches, caught, chide, chid, chidden, choose, chooses, chose, chosen, cling, clung, clothe, clad, come, comes, came, creep, crept, crow, crew, cut, deal, deals, dealt, dig, dug, dispread, done, dropped, draw, drawn, draws, drew, dream, dreamt, drive, drove, driven, dwell, dwelt, eat, ate, eaten, fall, fell, fallen, feed, fed, feel, felt, fight, fought, find, found, flee, fled, fling, flung, fly, flew, flown, follow, follows, forbear, forbore, forborne, forbid, forbad, forbidden, forecast, fordo, fordid, fordone, forego, forewent, foregone, foreknow, foreknew, foreknown, forerun, foreran, foresee, foresaw, foreseen, foreshow, foreshown, foretell, foretold, forget, forgets, forgot, forgotten, forgive, forgave, forgiven, forsake, forsook, forsaken, forsewar, forswore, forsworn, freeze, froze, frozen, gainsay, gainsaid, gild, gilt, gird, girt, gave, give, given, gives, go, goes, going, gone, grave, graven, grow, grew, grown, hamstring, hamstrung, hang, hung, hear, heard, heave, hove, hew, hewn, hide, hid, hidden, hit, hold, held, hurt, inlay, inlaid, kneel, knelt, knit, knew, know, knows, known, lade, laden, lay, laid, lead, leads, led, lean, leant, leap, leapt, learn, learns, learnt, leave, left, lend, lent, lie, lies, lain, lit, lived, lose, lost, made, make, makes, meant, meet, meets, met, melt, melten, misdeal, misdealt, misgive, misgave, misgiven, mislay, mislaid, mislead, misled, mistook, mistaken, misunderstood, mow, mown, outbid, outbade, outbidden, outbreed, outbred, outdo, outdid, outdone, outeat, outate, outeaten, outfight, outfought, outgo, outwent, outgone, outgrow, outgrew, outgrown, outlay, outlaid, outride, outrode, outridden, outrun, outran, outsell, outsold, outshine, outshone, outshoot, outshot, outsit, outsat, outspend, outspent, outspread, outthrow, outthrew, outthrown, outthrust, outwear, outwore, outworn, overbear, overbore, overborne, overbid, overbidden, overblow, overblew, overblown, overbuild, overbuilt, overbuy, overbought, overcast, overcome, overcame, overdo, overdid, overdone, overdraw, overdrew, overdrawn, overdrive, overdrove, overdriven, overeat, overate, overeaten, overfeed, overfed, overfly, overflew, overflown, overgrow, overgrew, overgrown, overhang, overhung, overhear, overheard, overlade, overladed, overladen, overlay, overlaid, overleap, overleapt, overlie, overlain, overpay, overpaid, override, overrode, overridden, overrun, overran, oversee, oversaw, overseen, overset, oversew, oversewn, overshoot, overshot, oversleep, overslept, overspend, overspent, overspread, overtake, overtook, overtaken, overthrow, overthrew, overthrown, overwind, overwound, overwrite, overwrote, overwritten, pay, paid, precast, prechoose, prechose, prechosen, prove, proves, proven, put, quit, raise, raises, ran, read, reave, reflect, reflects, reft, rebuild, rebuilt, recast, reeve, rove, relay, relaid, rend, rent, repay, repaid, reset, retell, retold, rid, ridded, ride, rode, ridden, rang, rung, rose, risen, rise, rises, run, ran, said, sat, saw, say, says, see, seek, sought, seen, send, sent, sold, set, sewed, shake, shook, shaken, shave, shaven, shear, shorn, shed, shine, shone, shoe, shod, shoed, shoot, shot, show, shown, shred, shrink, shrank, shrunk, shrive, shrove, shriven, shut, sing, sung, sink, sank, sunk, sit, sat, slay, slew, slain, sleep, slept, slid, slidden, sling, slung, slink, slunk, slit, smell, smelt, smite, smote, smitten, sow, sown, speak, speaks, spoke, spoken, sped, spell, spelt, spend, spent, spill, spilt, spin, spun, spit, spat, split, spoil, spoilt, spread, sprang, sprung, stand, stood, stave, stove, steal, stole, stolen, stuck, sting, stung, stink, stank, stunk, strew, strewn, stride, strode, strid, stridden, strung, strive, strove, striven, swear, swore, sworn, sweat, sweep, swept, swell, swollen, swim, swam, swum, swing, swung, take, takes, taken, teach, teaches, taught, tear, tore, torn, tell, think, thrive, throve, thriven, throw, threw, thrown, told, took, tried, tread, trod, trodden, unbend, unbent, unbind, unbound, unbuild, unbuilt, underbid, underbidden, underbuy, underbought, undercut, underdo, underdid, underdone, underfeed, underfed, undergo, underwent, undergone, underlay, underlaid, underlet, underlie, underlain, underpay, underpaid, underrun, underran, undersell, undersold, underset, undershoot, undershot, understand, understood, undertake, undertook, undertaken, underwrite, underwrote, underwritten, undo, undid, undone, undraw, undrew, undrawn, unfreeze, unfroze, unfrozen, ungird, ungirt, unhang, unhung, unknit, unlade, unladen, unlay, unlaid, unlearn, unlearnt,

unmake, unmade, unreeve, unrove, unsay, unsaid, unset, unsling, unslung, unspeak, unspoke, unspoken, unstick, unstuck, unstring, unstrung, unswear, unswore, unsworn, unteach, untaught, unthink, unthought, untread, untrod, untrodden, unweave, unwove, unwoven, unwind, unwound, upbuild, upbuilt, uphold, upheld, uppercut, uprise, uprose, uprisen, upsweep, upswept, upswing, upswung, wake, woke, woken, waylay, waylaid, wear, wore, worn, weave, wove, woven, wed, weep, wept, went, win, won, wound, wit, wot, wist, withdraw, withdrew, withdrawn, withhold, withheld, withstand, withstood, worked, wring, wrung, write, written, wrote

Other verbs 1: affect, affects, appear, appears, applies, ask, asks, believe, believes, call, calls, complicate, complicates, consider, considers, contain, contains, continue, continues, contribute, contributes, decide, decides, describe, describes, determine, determines, develop, develops, establish, establishes, expect, expects, follow, follows, happen, happens, include, includes, increase, increases, indicate, indicates, involve, look, looks, obtain, obtains, prepare, prepares, provide, provides, reach, reaches, recommend, recommends, reflect, reflects, receive, receives, relate, relates, remain, remains, remember, remembers, require, requires, seem, seems, serve, serves, suggest, suggests, uses, want

Other verbs 2: abandon, accelerate, accommodate, acquire, accomplish, add, advocates, administer, adopts, advertise, advise, affix, aid, aims, alleviate, allow, allows, appends, apply, argues, arises, arrange, articulates, ascertain, assembles, asserts, assess, assimilate, assist, assure, attract, avoid, bargains, broaden, causes, challenges, charge, clarify, closes, collect, combat, compare, complete, comprehend, comprises, concerns, concludes, condenses, conduct, confirm, confront, connects, connotes, consists, contact, constituent, constitute, contends, converts, convey, coordinate, cope, correlates, cover, covers, create, decline, debate, define, defines, demonstrate, demonstrates, depends, destroy, details, deter, diagnose, differ, discriminate, discuss, discusses, displaces, disseminate, disseminates, distinguish, educate, eliminate, embrace, emphasizes, employs, enable, enables, encompass, encourage, encourages, enforce, engage, engages, estimate, evaluate, evaluates, examine, examines, exclude, exemplifies, exerts, exist, exists, expand, exploits, explore, explores, extend, facilitate, favor, favors, fits, focus, focuses, formulate, furnish, gather, gain, generate, handle, help, highlight, holds, hopes, identify, identifies, ignore, illustrate, improve, incorporate, incorporates, inculcate, individualize, influence, inhibit, initiate, insure, interact, integrate, interpret, intervene, introduce, introduces, investigate, investigates, invokes, isolate, jumps, justify, lends, lists, maintain, manage, mandates, mentions, microfilm, monitor, monitors, moves, motivate, notes, occur, offer, offers, operate, operates, outlines, pass, participate, perceive, perform, permit, play, points, prescribe, present, presents, preserve, prevent, prevents, proceed, proceeds, produce, produces, promises, promote, promotes, proposes, protect, pursue, push, puts, ranks, recognize, reduce, regard, regulate, reject, remedy, replicate, replies, represent, represents, requests, respond, rests, return, returns, reveals, reverse, reviews, revolves, reward, satisfy, select, share, shift, shows, solve, speculates, stands, start, stimulate, submit, summarizes, support, tend, tends, transport, try, turn, unionize, utilize, varies,

15. ADV *(Ordinary Adverb)*

actually, again, ago, ahead, almost, alone, already, also, always, away, apparently, better, certainly, clearly, completely, daily, directly, early, easily, especially, even, exactly, farther, finally, forward, further, generally, hardly, hence, here, immediately, just, later, less, lesser, likely, merely, near, nearly, never, obviously, often, once, only, particularly, perhaps, prior, probably, quickly, ready, really, recently, simply, slowly, sometime, somewhat, soon, still, suddenly, today, together, too, usually, well

14. DTR *(Article)*

a, an, the

15. INT *(Adverb as Intensifier)*

rather, quite, very

16. PRN1 *(Personal Pronoun)*

anyone, he, her, him, I, it, me, none, others, she, them, they, thing, things, us, we, you

17. PRN2 *(Indefinite Pronoun)*

anything, everything, nothing, something

18. PRN3 *(Reflexive Pronoun)*

herself, himself, itself, myself, oneself, ourselves, themselves, yourself, yourselves

19. REL1 *(Interrogative or Relative Pronoun)*

what, whatever, who, whom

20. REL2 *(Interrogative or Relative Pronoun)*

which, whose

21. REL3 *(Relative Adverb)*

how, when, where, while, why

22. AMB *(Indefinite and Possessive Pronoun, Cardinal and Ordinal Numeral)*

all, another, any, billion, both, each, eight, eighteen, eighty, either, eleven, enough, every, few, fifteen, fifty, five, first, four, fourteen, fourth, her, his, hundred, its, many, million, more, most, much, my, neither, nine, nineteen, ninety, non, one, ones, other, our, same. second, seven, seventeen, seventy, several, six, sixteen, sixty, some, ten, their, these, third, thirteen, thirty, this, those, thousand, three, thru, twelve, twenty, two, whole, your

The function words listed in this group are to be assigned a group code AMB (i.e., Ambiguity) at the initial stage of parsing. Their final classes identifications are given on the basis of structural information obtained during parsing.

23. ADJ *(Adjective)*

able, aware, no

Included are only three unique adjectives.

24. ZZZ

as, so, such

Included are the three words which provide certain structural information.

Classification of Words (CW)

It is assumed that a dictionary-consulting program has 1) scanned the text to be analyzed, 2) identified all types of function words and 3) created a parallel symbolic string to store the syntactic information of the text. The symbolic string consists of one symbolic code for each word and punctuation in the text. The symbolic codes for function words are defined in the dictionary; all other words to be classified are given the code "XXX". A code such as "XXX" may have a suffix which indicates the ending of the word, e.g., XXX'ing. Other notation symbols to be used in the rules of CW are summarized in the table below.

XXX	any element of a sentence that has not been classified
YYY	any element of a sentence that has already been classified
$SYMB^n$	*n* consecutive elements in a sentence
XXX'x	x stands for the ending of an unclassified word
=>	yields
( )	used to enclose a series of alternatives
|	logical *or*
¬	logical *not*
'word'	precisely the word enclosed in the quote marks
'WORD'	any inflected form of the word enclosed in the quote marks
NON	noun
PTC	participle
INF	infinitive
DLM#	delimiter to indicate a clause boundary

CW01. Rules for the interrogative sentence, i.e., EOS = "?"

1. REL3 AUX DTR XXX^n XXX'ed ... =>
   REL3 AUX DTR ADJ^{n-1} NON VRB ... ($n \geq 1$)

2. REL3 AUX XXX^n XXX'ed ... =>
   REL3 AUX ADJ^{n-1} NON VRB ... ($n \geq 1$)

3. REL3 MOD DTR XXX^n AUX XXX'ed ... =>
   REL3 MOD DTR ADJ^{n-1} NON AUX VRB ... ($n \geq 1$)

4. REL3 MOD XXX^n AUX XXX'ed ... =>
   REL3 MOD ADJ^{n-1} NON AUX VRB ... ($n \geq 1$)

5. REL3 XXX MOD DTR XXX^n ... =>
   REL3 ADV MOD DTR ADJ^{n-2} NON VRB ... ($n \geq 2$)

6. REL3 XXX MOD XXX^n ... =>
   REL3 ADV MOD ADJ^{n-2} NON VRB ... ($n \geq 2$)

7. (MYRA r46-7) AUX XXX^n [AUX|VRB] ... =>
   AUX ADJ^{n-1} NON [AUX|VRB] ... ($n \geq 1$)

8. (MYRA r49-50) AUX XXX^n ... =>
   AUX ADJ^{n-2} NON VRB ... ($n \geq 2$)

9. (MYRA r48) AUX XXX ... =>
   AUX NON ..

CW02. Rules involving the group AMB

10. ... AMB^n ... =>
    ... DTR XXX^{n-1} ... ($n \geq 2$)

11. ... DTR AMB XXX'ed ... =>
    ... DTR ADV ADJ ...

12. ... DTR AMB ... =>
    ... DTR XXX ...

13. ... AUX AMB XXX'ed ... =>
    ... AUX ADV VRB ...

14. (MYRA r9) ... AMB XXX'($\neg$ ly) ... =>
    ... DTR XXX ...

CW03. Rules involving the group THT

15. (CAP r2 p.114) ... PRP THT# XXX ... =>
... PRP THT XXX ...

16. (CAP r4 p.114) ... THT# CCN ... =>
... THT CCN ...

17. (CAP r5 p.114) ... PNT THT#... =>
... PNT THT...

18. (CAP r1 p.114) THT# ... =>
THT ...

19. (MYRA r83) ... (¬ 'so') THT XXX ... =>
... (¬ 'so') DTR XXX ...

CW04. Rules involving the group DTR

20. ... DTR XXX CCN XXX XXX VRB ... =>
... DTR ADJ CCP ADJ NON VRB ...

21. ... DTR XXX CCN XXX XXX XXX'(edling) ... =>
... DTR ADJ CCP ADJ NON PTC ...

22. ... DTR XXX CCN XXX XXX XXX'ly XXX ... =>
... DTR ADJ CCP ADJ NON ADV VRB ...

23. ... DTR XXX CCN XXX XXX XXX'ly XXX'(edling) ... =>
... DTR ADJ CCP ADJ NON ADV PTC ...

24. ... DTR XXX CCN XXX XXX XXX'(¬ ly) ... =>
... DTR ADJ CCP ADV ADJ NON ...

25. ... DTR XXX CCN XXX XXX'ed ... =>
... DTR NON CCP NON PTC ...

26. ... DTR XXX CCN XXX XXX ... =>
... DTR ADJ CCP ADJ NON ...

27. (MYRA r4) ... DTR XXX XXX'(edling) ... =>
... DTR NON PTC ...

28. (MYRA r5) ... DTR XXX^{n} XXX'(edling) ... =>
... DTR ADJ^{n-1} NON PTC ... $(n \geq 2)$

29. (MYRA r3) ... DTR INT|ADV|AMB XXX^{n} ... =>
.. DTR INT|ADV|ADV ADJ^{n-1} NON ... $(n \geq 2)$

30. ... DTR XXX'ly XXX^n ... =>
... DTR ADV ADJ^{n-1} NON ... ($n \geq 2$)

31. (MYRA r1-2) ... DTR XXX^n ... =>
... DTR ADJ^{n-1} NON ... ($n \geq 1$)

32. ... DTR VRB XXX^n YYY($\neg$ NON) ... =>
... DTR ADJ^n NON YYY ...

33. (MYRA r6) ... DTR VRB ... =>
... DTR ADJ ...

CW05. Rules involving the group AMB

34. ... AMB('her'|'his'|'its'|'my'|'our'|'their'|'your') VRB ... =>
... ADJ NON ...

35. (MYRA r8) ... AMB('her'|'his'|'its'|'my'|'our'|'their'|'your') YYY ... =>
... PRN YYY ...

36. ... AMB XXX'ly ... =>
... AMB ADV ...

37. ... AMB('her'|'his'|'its'|'my'|'our'|'their'|'your') ... =>
... ADJ ...

CW06. Rules involving the groups of PRN1, PRN2, PRN3

38. (MYRA r11) ... PRN1 XXX ... =>
... PRN1 VRB ...

39. (MYRA r12) ... XXX (PRN1|PRN2) ... =>
... VRB (PRN1|PRN2) ...

40. (MYRA r13) ... PRN2 XXX (AUX|VRB) ... =>
... PRN2 ADJ (AUX|VRB) ...

41. (MYRA r14) ... PRN2 XXX YYY ... =>
... PRN2 VRB YYY ...

42. (MYRA r15) ... PRN2 XXX XXX ... =>
... PRN2 ADJ VRB ...

43. (MYRA r16) ... PRN3 XXX XXX ... =>
... PRN3 ADJ VRB ...

44. (MYRA r17) ... PRN3 XXX ... =>
... PRN3 VRB ...

CW07. Rules involving the groups of INT

45. (MYRA r18) ... INT XXX ... =>
... INT ADJ ...

CW08. Rules involving the groups of REL1, REL2, REL3

46. (MYRA r20) ... (¬ PRP) REL1 XXX ... =>
... (¬ PRP) REL1 VRB ...

47. ... (¬ PRP) REL2('which') XXX ... =>
... (¬ PRP) REL2 VRB ...

48. (MYRA r23) ... REL3 XXX'ing ... =>
... REL3 PTC ...

49. (MYRA r27) ... (REL1|REL2|REL3) (EOS|PNT) ... =>
... ADV (EOS|PNT) ...

50. ... (REL1|REL2|REL3) SCN ... =>
... (REL1|REL2|REL3) ADV ...

CW09. Rules involving the groups of AUX, VRB

51. (MYRA r28) ... AUX('BE') (ADV|NEG) AUX('being') ... =>
... AUX (ADV|NEG) PTC ...

52. (MYRA r29) ... AUX('BE') AUX('being') XXX ... =>
... AUX AUX VRB ...

53. ... AUX('being') XXX'ed ... =>
... AUX VRB ...

54. (MYRA r30) ... AUX('being') XXX ... =>
... AUX NON ...

55. (MYRA r31) ... AUX('being') XXX^{n} ... =>
... AUX ADJ^{n-1} NON ... $(n \geq 2)$

56. (MYRA r32) ... AUX('BE') XXX'(edling) ... =>
... AUX VRB ...

57. (MYRA r33) ... AUX('BE') (ADV|NEG) XXX'(edling) ... =>
... AUX (ADV|NEG) VRB ...

58. ... AUX('BE') SCN XXX'ed ... =>
... AUX ADV VRB ...

59. ... AUX('BE') XXX VRB ... =>
... AUX ADV VRB ...

60. ... AUX('BE') XXX XXX'ed ... =>
... AUX ADV VRB ...

61. ... AUX('BE') XXX PRP XXX^n ... =>
... AUX NON PRP ADJ^{n-1} NON ... $(n \geq 1)$

62. (MYRA r35) ... AUX('BE') (ADVIINTINEG) XXX ... =>
... AUX (ADVIINTINEG) ADJ ...

63. (MYRA r36) ... AUX('BE') (ADVIINTINEG) XXX^n ... =>
... AUX (ADVIINTINEG) ADJ^{n-1} NON ... $(n \geq 2)$

64. (MYRA r34,37) ... AUX('BE') XXX^n ... =>
... AUX ADJ^{n-1} NON ... $(n \geq 1)$

65. (MYRA r38) ... AUX('BE') AUX('having') ... =>
... AUX VRB ...

66. (MYRA r39) ... AUX('BE') (ADVINEG) AUX('having') ... =>
... AUX (ADVINEG) VRB ...

67. (MYRA r40) ... AUX('having') XXX'ed ... =>
... PTC PTC ...

68. (MYRA r41) ... AUX('having') AUX('been') XXX'ed ... =>
... PTC PTC PTC ...

69. (MYRA r42) ... AUX('having') ... =>
... PTC ...

70. (MYRA r43) ... AUX('HAVE') XXX'ed ... =>
... AUX VRB ...

71. ... AUX('HAVE') (INTIADVINEG) XXX'ed ... =>
... AUX (INTIADVINEG) VRB ...

72. ... AUX('HAVE') PRP XXX'ed ... =>
... AUX ADV VRB ...

73. ... AUX('HAVE') XXX'ly XXX'ed ... =>
... AUX ADV VRB ...

74. (MYRA r44-5) ... AUX('HAVE') XXX^n ... =>
... AUX ADJ^{n-1} NON ... $(n \geq 1)$

CW10. Rules involving the groups of AJN

75. ... MOD AJN ... =>
... MOD VRB ...

76. ... AJN('GET'|'KEEP') DTR ... =>
... VRB DTR ...

77. (MYRA r56-7) ... (¬ 'to') AJN('GET'|'KEEP') XXX'(edling) ...
... (¬ 'to') AUX VRB ...

78. (MYRA r60-1) ... AJN('let') XXX^n ...
... AUX ADJ^{n-1} NON VRB ... $(n \geq 1)$

79. (MYRA r59) ... AJN('let') XXX PRP ...
... AUX VRB PRP ...

80. (MYRA r58) ... AJN('let') XXX ...
... AUX VRB ...

81. (MYRA r62-3) ... AJN('let') DTR XXX^n ...
... AUX DTR ADJ^{n-2} NON VRB ... $(n \geq 2)$

82. (MYRA r64) ... AJN('let') (PRN1|PRN2|PRN3) XXX ...
... AUX (PRN1|PRN2|PRN3) VRB ...

83. (MYRA r65) ... AJN('DO') NEG XXX ...
... AUX NEG VRB ...

84. (MYRA r67-8) ... AJN('DO') XXX^n ...
... AUX VRB ADJ^{n-2} NON ... $(n \geq 2)$

85. (MYRA r66) ... AJN('DO') XXX ...
... AUX VRB ...

CW11. Rules involving the groups of MOD

86. ... MOD XXX'ly XXX ... =>
... MOD ADV VRB ...

87. (MYRA r51) ... MOD XXX ... =>
... MOD VRB ...

88. (MYRA r52) ... MOD ADV XXX ... =>
... MOD ADV VRB ...

CW12. Rules involving the groups of PRP

89. ... AUX PRP('to') XXX'(¬ ly) ... =>
... AUX PRP INF ...

90. ... PRP('to') XXX'ly XXX ... =>
... AUX ADV INF ...

91. (MYRA r85) ... PRP('to') (ADV|INT) XXX ... =>
... AUX (ADV|INT) INF ...

92. ... PRP('to') XXX'ly VRB ... =>
... AUX ADV INF ...

93. ... PRP('to') AUX VRB ... =>
... PRP AUX INF ...

94. ... PRP('to') AUX('be'|'have') XXX'ed ... =>
... PRP AUX INF ...

95. ... PRP('to') AUX('have') AUX('been') ... =>
... PRP AUX INF ...

96. ... PRP('to') AUX (INT|ADV) VRB ... =>
... PRP AUX (INT|ADV) INF ...

97. ... PRP('to') AUX('be'|'have') (INT|ADV) XXX'ed ... =>
... PRP AUX (INT|ADV) INF ...

98. ... PRP('to') AUX('have') (INT|ADV) AUX('been') ... =>
... PRP AUX (INT|ADV) INF ...

99. ... PRP(¬ 'of') XXX'ing CCN XXX'ing ... =>
... PRP PTC CCN PTC ...

100. (MYRA r87) ... PRP(¬ 'of') XXX'ing ... =>
... PRP PTC ...

101. ... PRP('to') VRB VRB ... =>
... PRP INF NON ...

102. ... PRP('to') (VRB|AJN) ... =>
... PRP INF ...

103. ... PRP VRB'ed ... =>
... ADV VRB ...

104. ... PRP VRB ... =>
... PRP NON ...

105. (MYRA r88-9) ... PRP XXX^n AUX ... =>
... PRP ADJ^{n-1} NON AUX ... $(n \geq 2)$

106. ... PRP XXX^n XXX'(edling) ... =>
... PRP ADJ^{n-1} NON PTC ... $(n \geq 1)$

107. (MYRA r90-1) ... PRP XXX^n ... =>
... PRP ADJ^{n-1} NON ... ($n \geq 1$)

108. (MYRA r92) ... PRP (INT|ADV) XXX^n ... =>
... PRP (INT|ADV) ADJ^{n-1} NON ... ($n \geq 1$)

109. (MYRA r93) ... PRP EOS =>
... ADV EOS

110. ... PRP PRP ... =>
... ADV PRP ...

111. (MYRA r94) ... ($\neg$ XXX) XXX'ing PRP ... =>
... ($\neg$ XXX) PTC PRP ...

CW13. Rules involving the groups of SCN

112. ... AUX SCN AUX ... =>
... AUX ADV AUX ...

113. (MYRA r75) ... SCN XXX'ing ... =>
... SCN PTC ...

114. (MYRA r76) ... SCN EOS =>
... ADV EOS

CW14. Rules involving the groups of THR

115. (MYRA r79) ... THR XXX ... =>
... THR VRB ...

116. (MYRA r80) ... (AUX|VRB) THR ... =>
... (AUX|VRB) ADV ...

117. (MYRA r81) ... THR PRP ... =>
... ADV PRP ...

CW15. Rules involving the groups of THT

118. (MYRA r82) ... THT YYY ... =>
... PRN YYY ...

CW16. Rules involving the groups of NEG

119. (MYRA r96) ... ADV('never') ADV XXX ... =>
... ADV ADV VRB ...

120. (MYRA r97) ... ADV('never') XXX'ing ... =>
... ADV PTC ...

121. (MYRA r95) ... ADV('never') XXX ... =>
... ADV VRB ...

122. (MYRA r98) ... AUX NEG XXX'ed ... =>
... AUX NEG VRB ...

123. ... AUX NEG XXX'ly XXX'ed ... =>
... AUX NEG ADV VRB ...

124. (MYRA r99) ... MOD NEG XXX ... =>
... MOD NEG VRB ...

125. (MYRA r100) ... NEG XXX'ing ... =>
... NEG PTC ...

126. ... NEG SCN ... =>
... NEG ADV ...

CW17. Rules involving the groups of XXX

127. ... XXX^n XXX'ed ... =>
... ADJ^{n-1} NON PTC ... $(n \geq 1)$

128. ... XXX'ly XXX^n ... =>
... ADV ADJ^{n-1} NON ... $(n \geq 2)$

129. (MYRA103-4) ... XXX^n ... =>
... ADJ^{n-1} NON ... $(n \geq 2)$

130. ... XXX'(edling) ... =>
... PTC ...

131. ... XXX'ly ... =>
... ADV ...

132. (MYRA r102) ... XXX ... =>
... NON ...

CW18. Rules involving adjustment

133. ... AMB ADV ... =>
... ADJ ADJ ...

134. ... ADV NON ... =>
... ADJ NON ...

135. ... DTR NON PTC NON ... =>
... DTR ADJ ADJ NON ...

136. ..NON PTC'ing (EOS|PNT|SCN|THR|THT#|AJN|AUX|VRB|PTC)..
=> ..ADJ NON (EOS|PNT|SCN|THR|THT#|AJN|AUX|VRB|PTC)..

137. ... PTC'ing (EOS|PNT|SCN|THR|THT#|AJN|AUX|VRB|PTC) ... =>
... NON (EOS|PNT|SCN|THR|THT#|AJN|AUX|VRB|PTC) ...

138. ... VRB PTC ... =>
... VRB NON ...

139. ... VRB (AUX|VRB) ... =>
... NON (AUX|VRB) ...

140. .. (¬ AJN|AUX|ADV|INT|NEG|MOD) VRB'(¬ ed) PNT(',') .. =>
... (¬ AJN|AUX|ADV|INT|NEG|MOD) NON PNT ...

141. ... NON PRN ... =>
... ADJ NON ...

142. ... (PNT|THT#) NON DTR NON ... =>
... (PNT|THT#) VRB DTR NON ...

143. ... PRP NON'ly PTC (ADJ|NON) ... =>
... PRP ADV ADJ NON ...

144. ... INF PTC ... =>
... INF NON ...

145. ... PTC PTC'ing ... =>
... PTC NON ...

146. ... PTC PTC'ed ... =>
... PTC VRB ...

147. ... NON CCN (DTR|ADJ|NON|PRN|PRN1|PRN2|PRN3) ... =>
... NON CCP (DTR|ADJ|NON|PRN|PRN1|PRN2|PRN3) ...

148. ..(ADJ|ADV|AMB|MOD|PRP) CCN (ADJ|ADV|AMB|MOD|PRP)..
=> .(ADJ|ADV|AMB|MOD|PRP)CCP(ADJ|ADV|AMB|MOD|PRP).

149. ... VRB CCN PTC ... =>
... VRB CCN VRB ...

150. ... PTC CCN VRB ... =>
... VRB CCN VRB ...

151. ... VRB (PRP|ADV) CCN PTC ... =>
... VRB (PRP|ADV) CCN VRB ...

152. ... PTC (PRP|ADV) CCN VRB ... =>
... VRB (PRP|ADV) CCN VRB ...

153. ... INF CCN VRB ... =>
... INF CCP INF ...

154. ... (PRN1|REL1) CCN (PRN2|REL2) ... =>
... (PRN1|REL1) CCP (PRN2|REL2) ...

155. ... PRP ADV INF ADJ NON CCN ADV ADJ ADJ NON ... =>
... PRP ADV INF ADJ NON CCP ADV INF ADJ NON ...

156. ... PRP ADV INF NON CCN ADV ADJ NON ... =>
... PRP ADV INF NON CCP ADV INF NON ...

157. ... PRP INF CCN (ADJ|NON|ADV) ... =>
... PRP INF CCP INF ...

Rule 158-166 assumes that no verb is found in a parsed sentnece

158. ... PTC'ed ... =>
... VRB ...

159. ... NON DTR (ADJ|NON) ... =>
... VRB DTR (ADJ|NON) ...

160. (MYRA r108) ... DTR ADJ^n NON ... =>
... DTR NON VRB ADJ^{n-2} NON ... $(n \geq 2)$

161. ... ADJ^n NON ... =>
... ADJ NON VRB ADJ^{n-3} NON ... $(n \geq 4)$

162. ... ADJ^n NON ... =>
... NON VRB ADJ^{n-2} NON ... $(n \geq 2)$

163. (MYRA r109) ... ADJ NON ... =>
... NON VRB ...

164. ... DTR ADJ^m NON DTR ADJ^n NON... => $(m \geq 1)$
... DTR ADJ^{m-1} NON VRB DTR ADJ^n NON ... $(n \geq 0)$

165. ... DTR ADJ^m NON ADJ^n NON... => $(m \geq 1)$
... DTR ADJ^{m-1} NON VRB ADJ^n NON ... $(n \geq 0)$

166. ... ADJ^m NON DTR ADJ^n NON... => $(m \geq 1)$
... DTR ADJ^{m-1} NON VRB ADJ^n NON ... $(n \geq 0)$

167. ... NON^n ... => ... ADJ^{n-1} NON $(n \geq 2)$

168. ... ZZZ ADV('well') ZZZ ... =>

... ZZZ^3 ...

169. ... ZZZ PRP ... =>
... ZZZ^2 ...

170. ... AUX('HAVE') ... => ... AXH ...

171. ... AUX(¬ 'HAVE') ... => ... AXB ...

172. ... PRN ... => ... PN0 ..

173. ... PRN(1|2|3) ... => ... PN(1|2|3) ...

174. ... REL(1|2|3) ... => ... RL(1|2|3) ...

175. ... PRP('to') ... => ... PRT ...

176. ... PNT(',') ... => ... CMA ...

Identification of Phrases (IP)

IP01. Rules for the phrases related to CCP (PAP, rules on p.120)

1. ... NCN CCP NON ... => ... NON^3 ...

2. ... ADJ CCP ADJ ... => ... ADJ^3 ...

3. ... ADV CCP ADV ... => ... ADV^3 ...

4. ... INF CCP INF ... => ... INF^3 ...

5. ... PRP CCP PRP ... => ... PRP^3 ...

6. ... PN CCP PN ... => ... PN^3 ...
   If PN CCP PN is not one of the following two patterns:
   ... ('I'/'she'/'he'/'we'/'they') CCP ('me'/'her'/'him'/'us'/'them')
   ... ('me'/'her'/'him'/'us'/'them') CCP ('I'/'she'/'he'/'we'/'they')

IP02. Rules for the phrases related to NON (PAP, rules on p.122)

7. ... DTR (ADV/INT) ADJ^n NON ... =>
   ... NON^{n+3} ... $(n \geq 1)$

8. ... DTR ADV^3 ADJ^n NON ... =>
   ... NON^{n+5} ... $(n \geq 1)$

9. ... DTR (ADV/INT) ADJ^n NON^3 ... =>
   ... NON^{n+5} ... $(n \geq 1)$

10. ... DTR ADV^3 ADJ^n NON^3 ... =>
    ... NON^{n+7} ... $(n \geq 1)$

11. ... DTR ADJ^n NON ... =>
    ... NON^{n+2} ... $(n \geq 0)$

12. ... DTR ADJ^n NON^3 ... =>
    ... NON^{n+4} ... $(n \geq 0)$

13. ... (ADV/INT) ADJ^n NON ... =>
    ... NON^{n+2} ... $(n \geq 1)$

14. ... ADV^3 ADJ^n NON ... =>
    ... NON^{n+4} ... $(n \geq 1)$

15. ... (ADV/INT) ADJ^n NON^3 ... =>
    ... NON^{n+4} ... $(n \geq 1)$

16. ... ADV^3 ADJ^n NON^3 ... =>
... NON^{n+6} ... ($n \geq 1$)

17. ... ADJ^n NON ... =>
... NON^{n+1} ... ($n \geq 0$)

18. ... ADJ^n NON^3 ... =>
... NON^{n+3} ... ($n \geq 0$)

19. ... $PN2^m$ ADJ^n ... =>
... NON^{m+n} ... (m,n = 1,3)

20. ... PN^n ... =>
... NON^n ... (n = 1,3)

21. ... ADJ ... => ... NON ...

IP03. Rules for the phrases related to VRB (PAP, rules on pp.122-3)

22. ... MOD NEG AUX VRB ... => ... VRB^4 ...

23. ... MOD (ADV|NEG) VRB ... => ... VRB^3 ...

24. ... MOD AUX^2 ADV VRB ... => ... VRB^5 ...

25. ... MOD AUX^2 VRB ... => ... VRB^4 ...

26. ... MOD AUX (NEG|ADV) VRB ... => ... VRB^4 ...

27. ... MOD AUX (NEG|ADV) VRB ... => ... VRB^4 ...

28. ... MOD AUX^n VRB ... => ... VRB^{n+2} ... (n = 0,1)

29. ... AUX^m NEG VRB^n ... =>
... VRB^{m+n+1} ... (m = 1,2; n = 1,3)

30. ... AUX^m VRB^n NEG ... =>
... VRB^{m+n+1} ... (m = 1,2; n = 1,3)

31. ... AUX^l ($INT|ADV^m$) VRB^n ... =>
... $VRB^{(1,l)+m+n}$ ... (l = 1,2; m,n = 1,3)

32. ... AUX ADV AUX VRB ... =>
... VRB^4 ...

33. ... AUX^m VRB^n ... =>
... VRB^{m+n} ... (m = 0,2; n = 1,3)

34. ... AUX (AXB|AXH) ... =>
... AVRB^{2} ...

35. ... MOD NEG ADV^{n} AXB ... =>
... AVRB^{3+n} ... (n = 1,3)

36. ... MOD (NEG|ADV) (AXB|AXH) ... =>
... AVRB^{3} ...

37. ... MOD (AXB|AXH) ... =>
... AVRB^{2} ...

38. ... (¬ PRT) (AXB|AXH) ... =>
... (¬ PRT) AVRB^{1} ...

39. . . ADV^{m} PTC^{n} ... =>
... PVRB^{m+n} ... (m = 0,1; n = 1,2)

40. ... PRT AUX^{l} (INT|ADV^{m}) INF^{n} ... =>
... IVRB$^{1+l+m+n}$ ... (l = 1,2; m,n = 1,3)

41. ... PRT AUX^{m} INF^{n} ... =>
... IVRB^{1+m+n} ... (m = 1,2; n = 1,3)

42. ... PRT AUX ... =>
... IVRB^{2} ...

43. ... PRT (INT|ADV^{m}) INF^{n} ... =>
... IVRB^{1+m+n} ... (m,n = 1,3)

44. ... PRT INF^{n} ... =>
... IVRB^{1+n} ... (n ≥ 1)

IP04. Rules for the phrases related to ADV (PAP, rules on pp.123)

45. ... (INT|ADV^{n}) ... => ... ADV^{n} ... (n = 1,3)

46. ... AMB ADV ... => ... ADV^{2} ...

IP05. Rules for the phrases related to CCP

47. ... NON^{m} CCP NON^{n} ... =>
... NON^{m+n} ... (m,n ≥ 1)

48. ... ADV^{m} CCP ADV^{n} ... =>
... ADV^{m+n} ... (m,n ≥ 1)

IP06. Rules for the phrases related to PRP (PAP, rules on pp.123)

49. ... PRP^{m} NON^{n} ... =>
... $PRPP^{m+1}$ ... ($m = 1,3$; $n \geq 1$)

50. ... $PRPP^{m}$ (CCN|CCP) $PRPP^{n}$ ... =>
... $PRPP^{m+n+1}$ ... ($m,n \geq 1$)

51. ... NON^{m} ('of')$PRPP^{n}$ ... =>
... NON^{m+n} ... ($m,n \geq 1$)

52. ... $PRPP^{m}$ ('of')$PRPP^{n}$ ... =>
... $PRPP^{m+n}$ ... ($m,n \geq 1$)

IP07. Rules for adjustment

53. ... NON^{l} NON^{m} NON^{n} ... =>
... NON^{l+m+n} ... ($l \geq 0$; $m,n \geq 1$)

Identification of Clauses (IC)

IC01. Rules for identifying clause boundaries (CAP/II, pp.131-3)

1. ... PRP RL(1|2|3) ... => ... PRP# RL(1|2|3) ...

2. ... (¬ ZZZ) ZZZ (¬ ZZZ) ... =>
... (¬ ZZZ) ZZZ# (¬ ZZZ) ...

3. ... (PVRB|IVRB) PRPP PRPP VRB ... =>
... (PVRB|IVRB) PRPP PRPP DLM# VRB ...

4. ... (PVRB|IVRB) PRPP VRB ... =>
... (PVRB|IVRB) PRPP DLM# VRB ...

5. ... PVRB NON PRPP PRPP NON PRPP PRPP VRB ... =>
.. PVRB NON PRPP PRPP NON PRPP PRPP DLM# VRB ...

6. ... PVRB NON PRPP NON PRPP VRB ... =>
... PVRB NON PRPP NON PRPP DLM# VRB ...

7. ... (PVRB|IVRB) NON PRPP PRPP VRB ... =>
... (PVRB|IVRB) NON PRPP PRPP DLM# VRB ...

8. ... (PVRB|IVRB) NON PRPP VRB ... =>
... (PVRB|IVRB) NON PRPP DLM# VRB ...

9. ... PVRB NON PRPP PRPP NON PRPP PRPP NON ... =>
... PVRB NON PRPP PRPP NON PRPP PRPP DLM# NON ...

10. ... PVRB NON PRPP NON PRPP NON ... =>
... PVRB NON PRPP NON PRPP DLM# NON ...

11. ... (PVRB|IVRB) NON VRB ... =>
... (PVRB|IVRB) NON DLM# VRB ...

12. ... IVRB NON PRPP PRPP NON VRB .. =>
... IVRB NON PRPP PRPP DLM# NON VRB ...

13. ... IVRB NON PRPP NON VRB ... =>
... IVRB NON PRPP DLM# NON VRB ...

14. ... IVRB PRPP NON ... =>
... IVRB PRPP DLM# NON ...

15. ... VRB PRPP PRPP VRB ... =>
... VRB PRPP PRPP DLM# VRB ...

16. ... VRB PRPP VRB ... =>

... VRB PRPP DLM# VRB ...

17. ... VRB PRPP PRPP NON VRB ... =>
... PVRB PRPP PRPP DLM# NON VRB ...

18. ... VRB PRPP NON VRB ... =>
... PVRB PRPP DLM# NON VRB ...

19. ... VRB NON PRPP PRPP VRB ... =>
... VRB DLM# NON PRPP PRPP VRB ...

20. ... VRB NON PRPP VRB ... =>
... VRB DLM# NON PRPP VRB ...

If more than one VRB is found between two delimiters "XXX#", the following two rules are invoked to insert additional "DLM#". The process is terminated whe.. every segment between two clause delimiters has one and only one VRB.

21. ... NON \ RB ... =>
... DLM# NON VRB ...

22. ... VRB ... =>
... DLM# VRB ...

23. Once clause delimiters have been located or inserted, clauses are extracted in the following sequence:

a) Nonfinite complement clauses, i.e., infinitive phrases, gerund phrases, and participle phrases;

b) Finite complement clauses;

c) Main clause.

Appendix C
INSTRUCTIONS FOR MANUAL PARSING

There are three passes for each parsing. In pass one, all words are classified into the nine word classes. The descriptive definitions of each word class is on the next page. Make sure that you have read the definitions before performing the parsing. Once you classify a word to, say adjective, you assign a code like "adj" to the word.

In pass two, you identify noun phrases and preposition phrases by simply indicating their boundaries. The two pairs of codes to be used in this pass are:

1) BNP (Beginning of a Noun Phrase) and ENP (End of ...)
2) BPP (Beginning of a Preposition Phrase) and EPP (End of ...)

Following are a few examples of noun phrases:

1) a statistical analysis;
2) automatic indexing;
3) computerized information retrieval;
4) library and information science;
5) humanity, social science, and natural science.

In pass three, you identify clauses, again by indicating their physical boundaries. A clause consists of a subject and a predicate. In addition, infinitive phrases, gerund phrases, and participle phrases are treated as clauses in this study. The codes to be used in this pass are BCS and ECS which represent the Beginning and the End of a clause. Naturally you may encounter nested clauses during your parsing.

Note that the last document surrogate to be parsed is the one for a consistency test.

WORD CLASSES

The nine word classes to be used in parsing are Noun, Pronoun, Numeral, Adjective, Adverb, Verb, Article, Conjunction, and Preposition. To each word class a descriptive definition is given in the form:

Word Class (code) ::= description (examples) | description (examples) | ...

The symbol "::=" is read as "is equivalent to" or "is descriptively defined as"; "code" is a short identification of one word class to be used in parsing; and square brackets may be used to indicate optional components in a phrase or clause.

```
1. NOUN (nun)   ::= Common Noun (library, water,
                                 discussion) |
                    Proper Noun (London, ASIS) |
                    Gerund (reading, training)

   Note: When a gerund leads a gerund phrase, the
         gerund is classified as a verb and the
         phrase as a clause.

2. PRONOUN (prn)  ::= Personal Pronoun
                        (I, s/he, it) |
                      Possessive Pronoun
                        (my, her, his,its) |
                      Reflexive Pronoun
                        (myself, itself) |
                      Demonstrative Pronoun
                        (this, those, same) |
                      Interrogative Pronoun
                        (who, which, what) |
                      Relative Pronoun
                        (who, which, what) |
                      Indefinite Pronoun
                        (some, any, all)
```

3. NUMERAL (num) ::= Cardinal Numeral
(one, twenty) |
Ordinal Numeral
(first, twentieth)

4. ADJECTIVE (adj) ::= Adjective
(red, instructive) |
Participle
(charming, broken) |
Noun (library school)

Note: A participle that leads a participle phrase is treated as a verb and the phrase as a clause.

5. ADVERB (adv) ::= Ordinary Adverb
(slowly, together) |
Interrogative Adverb
(when, why, how) |
Relative Adverb
(when, where) |
Conjunctive Adverb
(then, therefore)

6. VERB (vrb) ::= Notional Verb
(search, retrieve, teach) |
Phrasal Verb
(listen to, take care of) |
Link Verb
(be, seem, become, appear) |
Modal Verb (can, may, must) |
Auxiliary Verb
(shall, will, be, have, do) |
Infinitive
(to + [adverb] + verb + [..]) |
Gerund (gerund + [...]) |
Participle (participle + [...])

Note: Infinitive phrases, gerund phrases, and participle phrases areviewed as clauses in this study.

7. ARTICLE (art) ::= Article (a, an, the)

8. CONJUNCTION (cjn) ::= Coordinative Conjunction
(and, or, but) |
Subordinative Conjunction
(that, if)

9. PREPOSITION (pps) ::= Simple Preposition
(at, in, on, from) |
Compound Preposition
(as to, out of |
Double Preposition
(from behind, until
after) |
Phrasal Preposition
(according to,
in spite of)

Appendix D
THE FOUR STRUCTURAL RETRIEVAL FUNCTIONS

The assumptions of the retrieval functions:

1. a query consists of *m* case frames and a document to be matched consists of *n* case frames, *m* is not necessarily equal to *n*;

2. the size of a case frame *size*[caseframe] is equal or greater than one;

3. the default value of each case is ∅, that is, there is no index word in the case;

4. DEL and INS are the two system functions.

PROGRAM: The Two Retrieval Functions;

Procedure FIXED-CASE-EDIT(Caseframe$_i$, Caseframe$_j$, Cost(i,j), Normalized_Cost(i,j));

```
Begin
 if Verb_i = Verb_j
  then Cost(i,j) := 0
  else Cost(i,j) := 2;
 for k := 1 to 10 do        /* 10 different case relations */
 begin
  if (Case_ik = ∅) and (Case_jk ≠ ∅)
   then Cost(i,j) := Cost(i,j) + DEL(Case_jk);
  if (Case_ik ≠ ∅) and (Case_jk = ∅)
   then Cost(i,j) := Cost(i,j) + INS(Case_ik);
  if (Case_ik ≠ ∅) and (Case_jk ≠ ∅)
   then Cost(i,j) := Cost(i,j) + DEL(Case_jk) + INS(Case_ik)
              - 2 * (number of words shared by Case_ik
                  and Case_jk);
 end;
 Normalized_Cost(i,j) := Cost(i,j) / (size[Caseframe_i] +
                         size[Caseframe_j]);
End;
```

```
Procedure FREE-CASE-EDIT(Caseframe_i, Caseframe_j, Cost(i,j),
                         Normalized_Cost(i,j));

Begin
 if Verb_i = Verb_j
  then Cost(i,j) := 0
  else Cost(i,j) := 2;
 INITIALIZE Case_Cost(s,t) := 0;
 INITIALIZE Normalized_Case_Cost(s,t) := 1;
 for s := 1 to 10 do
 begin
  for t := 1 to 10 do
  begin
   if (Case_s = ∅) and (Case_t ≠ ∅)
    then begin
         Case_Cost(s,t) := DEL(Case_t);
         Normalized_Case_Cost(s,t) := 1;
        end;
   if (Case_s ≠ ∅) and (Case_t = ∅)
    then begin
         Case_Cost(s,t) := INS(Case_s);
         Normalized_Case_Cost(s,t) := 1;
        end;
   if (Case_s ≠ ∅) and (Case_t ≠ ∅)
     and (type(Case_s) = type(Case_t))
    then begin
         Case_Cost(s,t) := INS(Case_s)
                         + DEL(Case_t)
                         - 2 * (number of words shared by
                              Case_s and Case_t)
                         - 2;
         Normalized_Case_Cost(s,t) :=
          Case_Cost(s,t) / (size[Case_s] +
                            size[Case_t] - 2);
        end;
   if (Case_s ≠ ∅) and (Case_t ≠ ∅)
     and (type(Case_s) ≠ type(Case_t))
    then begin
         Case_Cost(s,t) := INS(Case_s)
                         + DEL(Case_t)
                         - 2 * (number of words shared by
                              Case_s and Case_t);
         Normalized_Case_Cost(s,t) :=
          Case_Cost(s,t) / (size[Case_s] + size[Case_t]);
        end;
 end;
end;
```

```
 while Case_Cost(s,t) ≠ ∅ do
 begin
  minimum[Normalized_Case_Cost(s,t)];
  s_minimum := s;
  t_minimum := t;
  Cost(i,j) := Cost(i,j) + Case_Cost(s_minimum, t_minimum);
  remove Case_Cost(s_minimum, _);
  remove Case_Cost(_, t_minimum);
  remove Normalized_Case_Cost(s_minimum, _);
  remove Normalized_Case_Cost(_, t_minimum);
 end;
 Normalized_Cost(i,j) := Cost(i,j)/(size[Caseframe_i] + size[Caseframe_j]);
End;

/* The main procedure of the two functions */

Begin
 Distance := 0;
 for i := 1 to m do
 begin
  for j := 1 to n do
  begin
    /* EDIT stands for either Fixed-Case-EDIT or Free-Case-EDIT */
   EDIT(Caseframe_i, Caseframe_j, Cost(i,j), Normalized_Cost(i,j));
  end;
 end;
 for x := 1 to minimum[m, n] do
 begin
  minimum[Normalized_Cost(i,j)];
  i_minimum := i;
  j_minimum := j;
  Distance := Distance + Cost(i_minimum j_minimum);
  remove Normalized_Cost(i_minimum,_);
  remove Normalized_Cost(_, j_minimum);
  remove Cost(i_minimum,_);
  remove Cost(_, j_minimum);
  remove Caseframe_i_minimum from QUERY;
  remove Caseframe_j_minimum from DOCUMENT;
 end;
 for y := 1 to |m-n| do
 begin
  if m > n
   then Distance := Distance + INS(Caseframe_y);
  if m < n
   then Distance := Distance + DEL(Caseframe_y);
 end;
 Similarity := 1 - Distance / (size[QUERY] + size[DOCUMENT]);
End.
```

```
/* The main procedure which incorporates the focusing process */

Begin
 Distance := 0;
 for i := 1 to m do
 begin
  for j := 1 to n do
  begin
   EDIT(Caseframe_i, Caseframe_j, Cost(i,j),
       Normalized_Cost(i,j));
  end;
 end;
 Normalization := 0;
 for x := 1 to minimum[m, n] do
 begin
  minimum[Normalized_Cost(i,j)];
  i_minimum := i;
  j_minimum := j;
  Distance := Distance + Cost(i_minimum,j_minimum);
  remove Normalized_Cost(i_minimum,_);
  remove Normalized_Cost(_, j_minimum);
  remove Cost(i_minimum,_);
  remove Cost(_, j_minimum);
  Normalization := Normalization + size[Caseframe_i_minimum];
  remove Caseframe_j_minimum from QUERY;
  Normalization := Normalization + size[Caseframe_j_minimum];
  remove Caseframe_j_minimum from DOCUMENT;
 end;
 Similarity := 1 - Distance/Normalization;
End.
```

Appendix E
QUERIES

The following are the thirty queries used in this study. The symbol "@" is used to indicate the end of a title that is not a complete sentence. At the end of each query are the number of relevant documents and the number of nonrelevant documents retrieved for this query; the two numbers are in a pair of brackets.

1. direct charging to users for reference and current awareness service of libraries or other information services@ the articles cover philosophy, policy, practice, or fee charges. libraries are academic, public, special, and others. (8:8)

2. federal aid to day care centers or services in USA@ the articles cover history, philosophy, arguments, pro and con, experiences, funding, evaluation, parent involvement, and attitudes. (10:10)

3. jean piaget's theories, and thought processes or language development of children@ (10:10)

4. libraries and librarians in middle east arab countries including egypt all types of libraries and information centers@ (5:5)

5. library services to physically handicapped@ (13:13)

6. effects of TV violence on children@ (10:10)

7. drug abuse among students of elementary or secondary schools@ drug includes alcohol. the articles describe school education programs and sociological studies. (10:10)

8. school busing and racial integration@ (10:10)

9. recreational use of forest lands@ (10:10)

10. women's sports and title 9, '972 federal education act amendments@ (4:4)

11. white flight to the suburbs@ (5:5)

12. training for supervision and management in libraries and information centers@ the topics include need for training, descriptions of training programs or materials, training of students and professional working librarians, and academic or job training. (10:10)

13. audiovisual aids for orientation or instruction of library users@ (10:10)

14. evaluation of primary school english reading programs, reading materials, and techniques@ (10:10)

15. formal science education programs in universities and secondary schools in the soviet union@ sciences include mathematics and engineering. (3:3)

16. vocational education of american indian@ the articles cover history, data, and programs to provide this education. (10:10)

17. evaluation of bilingual elementary and secondary school programs or techniques for Spanish and English languages@ (10:10)

18. evaluation of indexing and cataloging@ the article cover work, methods, products, languages, and representation and storage of the index information. indexing includes all forms of text searching such as indexing by text words. evaluation is time, cost, error rates, recall/relevance, or evaluation criteria. indexing includ .s citation indexing, title word indexing, or automatic indexing. (10:10)

19. collective bargaining in libraries of institutions of higher education@ (16:16)

20. textbooks or grammars of navaho language useful navaho material for teaching navaho or about navaho linguistics@ (9:9)

21. revision of anglo american cataloging rules@ (3:3)

22. education including library activity in sri lanka@ (12:12)

23. the design, construction, and performance of dams in areas of seismic instability@ (7:7)

24. use solar energy for heating homes. (8:8)

25. the inflationary effects of rising oil prices caused by the OPEC cartel@ (5:5)

26. the transfer of technology to third world or less-developed countries@ (7:7)

27. use behavior modification therapy to treat cerebral palsy victims@ (2:2)

28. insurance companies use direct mail advertising. (15:15)

29. power systems for a space station@ (15:15)

30. adverse side-effects of ibuprofen@ the articles describe human cases or research involving humans. (10:10)

REFERENCES

- Aitchison, J. & Cleverdon, C. W. (1963). *A Report on a Test of the Index of Metallurgical Literature of Western Reserve University.* Cranfield: College of Aeronautics.

- Austin, D. (1974). The development of PRECIS: A theoretical and technical history. *Journal of Documentation, 30,* 47-102.

- Austin, D. & Dykstra, M. (1984). *PRECIS: A Manual of Conceptual Analysis and Subject Indexing* (second ed.). London: British Library Bibliographic Services Division.

- Belkin, N. J., Oddy, R. N., & Brookes, H. M. (1982a). ASK for information retrieval: Part I. Background and theory. *Journal of Documentation, 38,* 61-71.

- Belkin, N. J., Oddy, R. N., & Brookes, H. M. (1982b). ASK for information retrieval: Part II. Results of a design study. *Journal of Documentation, 38,* 145-164.

- Bely, N., Borillo, A., Virbel, J. & Siot-Desauville, N. (1970). *Procedures d'Analyse Semantique Appliquees a la Documentation Scientifique.* Paris: Gauthier-Villars.

- Bruce, B. (1975). Case systems for natural language. *Artificial Intelligence, 6,* 327-360.

- Chafe, W. L. (1970). *Meaning and the Structure of Language.* Chicago: The University of Chicago Press.

- Cohen, P. R. & Kjeldsen, R. (1987). Information retrieval by constrained spreading activation in semantic networks. *Information Processing & Management, 23,* 255-268.

- Cook, W. A. (1970). Case grammar: From roles to rules. *Languages and Linguistics Working Papers* No. 1 (pp. 14-30). Washington, D.C.: Georgetown University Press.

- Cook, W. A. (1971). Case grammar as a deep structure in tagmemic analysis. *Languages and Linguistics Working Papers* No. 2 (pp. 1-9). Washington, D.C.: Georgetown University Press.

- Cook, W. A. (1972). A case grammar matrix. *Languages and Linguistics Working Papers* No. 6 (pp. 15-48). Washington, D.C.: Georgetown University Press.

- Farradane, J. (1980a). Relational indexing. Part I. *Journal of Information Science, 1,* 267-276.

- Farradane, J. (1980b). Relational indexing. Part II. *Journal of Information Science, 1*, 313-324.

- Fillmore, C. J. (1968). Case for case. In E. Back, & R. T. Harms, *Universals in Linguistic Theory* (pp. 1-88). NY: Holt, Rinehart and Winston, Inc.

- Fillmore, C. J. (1970). *Improvement in Case Grammar*, 1970. Presented at the summer institute of the linguistic society of America at The Ohio State University, OH.

- Fillmore, C. J. (1971). Some problems for case grammar. *Working Papers in Linguistics* No. 10. The Ohio State University, Ohio.

- Fries, C. C. (1952). *The Structure of English*. NY: Harcourt Brace & Would, Inc.

- Gardin, J. C. (1965). *SYNTOL*. New Brunswick, New Jersey: The Rutgers University Press.

- Guilford, J.P. (1959). *Personality*. NY: McGraw-Hill.

- Hancox, P. & Smith, F. (1985). A case system for the PRECIS indexing language. *Informatics 8*, Proceedings of a conference jointly sponsored by Aslib, the Aslib Informatics Group, and the Information Retrieval Specialist Group of the British Computer Society, (pp. 120-147). London: Aslib.

- Hunt, R., Horne, C. R., Boone, L., Dennis, L. & Whelan, H. (1977). *PRECIS, LCSH and KWOC: Report of a Research Designed to Examine the Applicability of PRECIS to the Subject Catalogue of an Academic Library. Part 2: Methodology and Results - the Experimental Searches*. University of Wollongong. The Library.

- Karlgren, H. (1977). Homeosemy --- On the linguistics of information retrieval. In D. E. Walker, H. Karlgren & M. Kay, *Natural Language in Information Science*, (pp.167-182). Skriptor, Stockholm, Sweden: FID Publication 551.

- Lehnert, W. G. (1978). *The Process of Question Answering*. NY: John Wiley & Sons.

- Lewis, D. A. (1984). *Case Grammar and Functional Relations in Aboutness Recognition and Relevance Decision-making in the Bibliographic Retrieval Environment*. Unpublished doctoral dissertation, University of Western Ontario, London, Canada.

- Liddy, E. D. (1987). Discourse-level structure in abstracts. In *Proceedings of the 50th Annual Meeting of the American Society for Information Science*. Massachusetts: Boston.

- Lu, S. Y. (1979). A tree to tree distance and its application to cluster

analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence. 1*, 219-229.

- MacCafferty, M. & Cray, K. (EDS.). (1979). The analysis of meaning. *Informatics 5*, Proceedings of a conference held by the Aslib informatics group and the BCS information retrieval specialist group. London: Aslib.

- Markey, K & Cochrane, P. A. (1981). *Online Training and Practice Manual for ERIC Data Base Searchers*, 2nd ed. Syracuse, N.Y.: ERIC Clearinghouse on Information Resources.

- Meada, T. (1981). An approach toward functional text structure analysis of scientific and technical documents. *Information Processing & Management, 17*, 329-339.

- Michell, G. (1979) Does PRECIS have feet of clay? Problems with the universality of the role operators. In *Proceedings of the Seventh Annual Canadian Conference on Information Science*, pp. 123-129. Alberta: Banff.

- Oddy, R. N., Palmquist, R. A., & Crowford, M. A. (1986). Representation of anomalous states of knowledge in information retrieval. In *Proceedings of American Society for Information Science*, 23 (pp. 248-254). Illinois: Chicago.

- Perry, J. W. & Kent, A. (1958). *Tools for Machine Literature Searching*. NY: Interscience Publishers, Ltd.

- Salton, G. (1965). Automatic phrase matching. In David G. Hays, *Reading in Automatic Language Processing*, (pp. 169-188). New York: American Elsevier publishing Company, Inc.

- Salton, G. (1968). *Automatic Information Organization and Retrieval*. NY: McGraw-Hill Book Company.

- Salton, G. (1972). A new comparison between conventional indexing (MEDLARS) and automatic text processing (SMART). *Journal of the American Society for Information Science*, 23, 75-84.

- Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. NY: McGraw-Hill international book company.

- Selkow, S. M. (1977). The tree-to-tree editing problem. *Information Processing Letters, 6*, 184-186.

- Schank, R. C. (1975). *Conceptual Information Processing*. NY: American Elserier.

- Schank, R. C., Goldman, N. M., Rieger, C., & Riesbeck, R. (1973). Margie: Memory, analysis, response, generation and inference on English. In *Proceedings of the 3rd International Joint Conference in Artificial Intelligence*. CA: Stanford.

- Sorensen, J. & Austin, D. (1976). PRECIS in a multilingual context. *Libri, 26*, 108-139.

- Sparck Jones, K. & Kay, M. (1973). *Linguistics and Information Science.* NY: Academic Press.

- Starosta, S. (1988). *The Case For Lexiccse: An Outline Of Lexicase Grammatical Theory.* NY: Pinter Publishers.

- Strong, S. M. (1974). An algorithm for generating structural surrogates of English text. *Journal of American Society for Information Science, 25*, 10-24.

- Tai, K. C. (1979). The tree-to-tree correction problem. *Journal of the Association for Computing Machinery, 26*, 422-433.

- Tague, J. M. (1980). The pragmatics of information retrieval experimentation. In K. Sparck Jones, *Information Retrieval Experiment*, (pp.59-102). London: Butterworths.

- vac Rijsbergen, C. J. (1979). *Information Retrieval.* London: Butterworths.

- van Rijsbergen, C. J. (1987). A new theoretical framework for information retrieval. *SIGIR FORUM, 21*, 23-29.

- Young, C. (1973). *Development of Language Analysis Procedures with Application to Automatic Indexing* (Doctoral dissertation, The Ohio State University). (University Microfilms No. 73-18, 966)