Information Access and Retrieval with Semantic Background Knowledge

Anil Srinivasa Chakravarthy

Bachelor of Technology, Computer Science and Engineering
Institute of Technology, Banaras Hindu University, India, 1989
Master of Science
Massachusetts Institute of Technology, 1991

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at the
Massachusetts Institute of Technology

June 1995

Author: [illegible]
Program in Media Arts and Sciences
May 5, 1995

Certified by: [illegible]
Kenneth B. Haase
Assistant Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Advisor

Accepted by: [illegible]
Stephen A. Benton
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Information Access and Retrieval with Semantic Background Knowledge

Anil Srinivasa Chakravarthy

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 5, 1995 in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Abstract

The rapid growth of on-line information on the Internet and other platforms presents a great challenge to information retrieval systems. However, improvements in information retrieval technology have not kept pace with the growth in available information. State-of-the-art information retrieval systems rely exclusively on literal matching of keywords in queries and documents. In order to shoulder a greater load in searching for information and to improve the effectiveness of searches, information retrieval systems need the ability to automatically match user queries and documents without relying on literal equivalence.

This dissertation presents a framework for generalization through the incorporation of semantic knowledge into information retrieval systems. The semantic knowledge is extracted from an online Webster's dictionary and a thesaurus named WordNet. The use of semantic knowledge is guided by constructing structured representations of the queries and the text, which identify the roles being played by the salient words. Generalization is controlled by the use of application-dependent matching rules. The dissertation describes the architecture, implementation and evaluation of two applications, ImEngine and NetSerf, built using this framework. ImEngine retrieves captions of pictures and video clips using natural language queries. NetSerf is a system that enables information access by finding Internet information archives in response to user queries. The performance of ImEngine and NetSerf is compared to that of a standard keyword-based retrieval system, SMART. The dissertation also presents new techniques for the extraction of semantic knowledge from the dictionary, and for word-sense disambiguation using the dictionary and WordNet. Both ImEngine and NetSerf are evaluated with respect to the three main components of the framework: semantic knowledge, structured representations and disambiguation.

Thesis Advisor
Kenneth B. Haase
Assistant Professor of Media Arts and Sciences
Program in Media Arts and Sciences

This work was supported in part by Interval Research Corporation and the News in the Future consortium.

Doctoral Dissertation Committee

[illegible]

Thesis Advisor: ______________________________

Kenneth B. Haase
Assistant Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Massachusetts Institute of Technology

[illegible]

Thesis Reader: ______________________________

Patricia E. Maes
Assistant Professor of Media Technology
Program in Media Arts and Sciences
Massachusetts Institute of Technology

Thesis Reader: ______________________________

Peter Norvig
Computer Scientist
Harlequin, Inc.

Acknowledgements

It is a pleasure to thank my thesis advisor, Ken Haase, for his generous support and guidance over the last four years. When I was a greenhorn, he gave me the freedom to explore; when I discovered what I wanted to do, he helped me narrow my focus. I am just as grateful to my thesis readers, Pattie Maes and Peter Norvig, for their constructive criticism and advice.

I have a lot of good things to say about the Media Lab. The high quality of the free food, for instance. Seriously, the structure of the lab encourages collaboration and "demos." I have learnt much from both. Most importantly, I have had the good fortune to become friends with a number of bright people. Though I cannot list them all here, I am especially indebted to those who helped me with my thesis: Janet Cahn, Marc Davis, Sara Elo, Hideki Hirakawa, Suguru Ishizaki, Yezdi Lashkari, Alan Ruttenberg, Warren Sack, Mike Travers and Louie Weitzman. A special word of thanks to the Media Lab administration, especially Betty Lou McClanahan, Linda Peterson and Santina Tonelli.

Many people in the information retrieval and computational linguistics communities have helped me with encouragement and criticism, particularly, Bill Dolan, Christiane Fellbaum, David Lewis, Phil Resnik, Lucy Vanderwende and Ellen Voorhees. I would like to thank Peter Norvig and Bill Woods for setting up an internship for me at Sun Microsystems two summers ago, and Marc Davis and Meg Withgott for hosting me at Interval last summer.

I am grateful to Marvin Minsky and David Waltz for taking the time to serve on my general exam committee. I am also grateful to Robert Logcher and Duvvuru Sriram for their support during my S.M. The "desi" community at MIT has been a source of sustenance. Again, some names come to mind more readily than others: Ashu, Giri, Koods, Lakshman, Nagi, Nooks, Ranga, Ravi and Satyan.

My family has been the source of my pride and motivation. This work is dedicated to them.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Information is a basic need. Most people spend significant time looking for information that will help them achieve their professional and personal goals. Many authors have written about the emergence of the "information society," in which "information workers" are in the majority. Readers have to sort through a wide variety of information, such as news, trade journals, business reports, advertisements and scientific articles, in order to find all relevant information about a particular topic or to keep abreast of political, technological, social and market developments.

The profusion of information is not a recent problem. In 1945, Vannevar Bush wrote his famous article *As We May Think* calling for new technologies to help in finding information [Bush 1945]. His reasons were clear:

There is a growing mountain of research. But there is increased evidence that we are being bogged down today as specialization extends. The investigator is staggered by the findings and conclusions of thousands of other workers - conclusions which he cannot find time to grasp, much less to remember, as they appear.

Two decades later, Hubert Humphrey said in his message to the Third Annual National Colloquium on Information Retrieval [Schecter 1966]:

Yet, today more than ever before, the deluge of information in every form - printed, written, visual, audio and oral, words, numerals and illustrations, manual and machine - in all sorts of permutations and combinations - represents one of the greatest challenges and opportunities of this dynamic era.

If anything, the problem is more acute now. The advent of desktop publishing has made it much easier to publish multimedia documents. As the number of information workers rises, it has led to more involvement in content creation. And as electronic commerce becomes the norm, there will be an even greater deluge of information. Many industries are making plans for "disintermediation," which is expected to make more information available to the consumer (ultimate user) by bypassing intermediaries like retailers [Hagel and Eisenmann 1994].

However, with the problem comes an opportunity. What we have now is a digital deluge. Almost all text is produced in machine-readable form, and other media, especially video and audio, are rapidly moving in that direction. In addition, since computers are increasingly interlinked using the Internet or on-line services, the physical location of the information is not very important. Hence, the trend toward *automatic information retrieval,* in which a computer program is used to search for information on behalf of the user.

If successful, automatic information retrieval offers three advantages. First and foremost, it could "scale up" to larger information databases. If a database were to grow from a 1000 to a 100000 documents, it would be extremely taxing for a human to search through it. Second, an automatic information retrieval program would be widely affordable. Not everyone can hire a personal librarian, but a program can be made inexpensive enough for common use. Third, customization might be possible so that the program can be designed to tune itself to the individual needs of users.

1.1 Automatic Information Retrieval

The use of computer-based techniques to find relevant information can be traced back to the years following the Second World War. Historically, the problem of finding information had been addressed by a process of classification and cataloguing of all recorded knowledge [Sharp 1965]. But these methods came under strain in the first half of the century as scientific and engineering knowledge grew at an unprecedented rate. When computers became available in the 1940s, an active area of research emerged to develop computer-based techniques for finding information, and the new field was christened "information retrieval" (IR) by Calvin Mooers in 1950. But the name was something of a misnomer, as Lancaster explains [Lancaster 1968]:

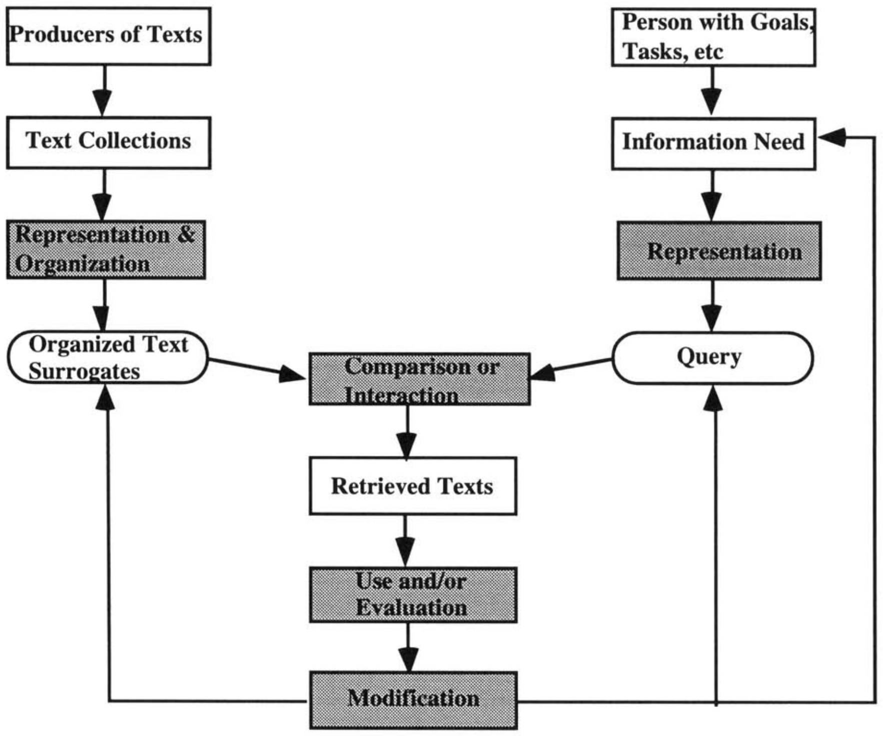
An information retrieval system does not inform (i.e., change the knowledge of) the user on the subject of his inquiry. It merely informs him on the existence (or nonexistence) and whereabouts of documents relating to his request.

An information retrieval system differs from regular database retrieval systems in that the underlying data of an IR system is unstructured, or at best, semi-structured. A payroll database is an example of a structured database. A database of abstracts from scientific journals would be a semi-structured database, in all probability. For each abstract, the structured part would indicate the title, author(s), etc., while the text of the abstract itself would be unstructured.

People use many methods for finding information. Librarians can be helpful in searching through archived knowledge. Knowledgeable friends can be asked for suggestions. Agents can provide guidance for specialized domains like travel or personal finance. For keeping up-to-date with events, one can turn to newspapers, magazines, journals, etc. No automatic information retrieval system today is as comprehensive in the tasks it can perform. Given a query by a user, it is limited to searching through a specified database for information relevant to the query. Figure 1.1 shows a general model of automatic information retrieval. The documents retrieved by the system are called *hits*.

Judging whether a document is relevant to a query is difficult for a computer program. Human experts are capable of "understanding" the content of the document and the query, and using it to judge whether the document and the query are related. In addition, information is relevant only if the user does not already know it. As the information theory slogan goes, "information is the difference that makes a difference."

FIGURE 1.1. A general model of IR (from [Belkin & Croft 1992])[1]



Ideally, an information retrieval system's behavior should mimic that of human experts. To use the vocabulary of Figure 1.1, this would require that the processes of representation and organization be equivalent to human understanding. The process of comparison/interaction would need to be similar to human information searching.

1. Regular boxes denote external data, shaded boxes denote processes, and rounded boxes denote internal representations.

But in practice, this has proved hard to implement. First, human understanding is still opaque, which means that even psychologists cannot describe what cognitive processes are involved in understanding and searching for documents. Second, work in artificial intelligence has revealed that even the simplest understanding requires a large body of "world knowledge," which humans acquire either innately or through learning. It is a safe bet that till computer programs have active access to such knowledge, they will not be capable of any significant understanding. Chapter 2 discusses some of these issues in greater detail.

Widely-used, state of the art retrieval systems have taken a different approach toward representation, organization and comparison [Salton 1989]. Recognizing that understanding documents is very difficult, these systems are built around a simple concept: a query is relevant to a document if they share common *keywords* (e.g., "ethanol" or "justice"). This allows the systems to sidestep another unsolved problem, that of analysis (natural language processing) of the text. Clearly, using keywords as discriminators might lead a query to be matched to many documents. All retrieval systems therefore provide methods for ranking (evaluating) matches between queries and documents.

However, there is evidence from large-scale evaluations of keyword-based IR systems that their performance deteriorates when they are used on sizable collections of text, e.g., [Blair & Maron 1985]. Primarily, there are two problems:

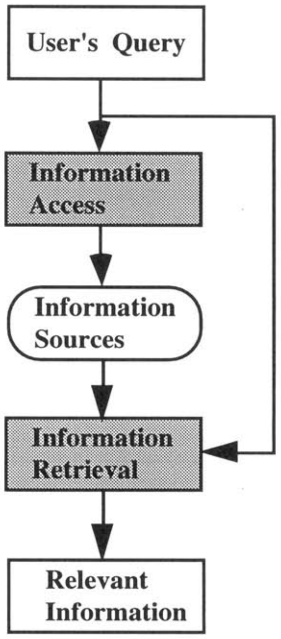
- How can a keyword-based system avoid spurious matches? For example, the word "mortar" has two meanings. The query might use the word in one sense, but the document might use it in the other. Second, even if the words are being used in the same sense, they might be playing entirely different roles, in which case the document might not be relevant to the query.

- Relevant documents might not be phrased in the same vocabulary as the query. How can a keyword-based system find conceptually, but not literally, equivalent documents?

In this thesis, these problems are addressed through the use of semantic knowledge from dictionaries and thesauri. These problems are related to the lack of world knowledge discussed earlier. The use of semantic knowledge from dictionaries and thesauri is not a full substitute for world knowledge, but as the thesis shows, it can lead to significant improvement in performance on selected retrieval problems. Section 1.2 gives an overview of the techniques developed in the thesis.

Information access is another problem addressed in this thesis. In the traditional information retrieval model, the system typically processes the user's query on one, or a few, databases. But that model is not valid for retrieval over the Internet. There are several thousand information archives (sources) on the Internet, which implies that the user needs assistance in deciding which information archives to search on. Let us consider the example of WAIS [Kahle & Medlar 1991]. WAIS allows users to deploy, search and retrieve documents and other types of information from indexed databases throughout the Internet. At the time of retrieval, WAIS expects the user to indicate the databases that should be searched. This is usually done by searching first on a directory of databases maintained by WAIS. Figure 1.2 shows this new model of finding information, where information access locates databases that could contain the information being sought, and information retrieval is, as before, the process of searching the selected databases.

FIGURE 1.2. Information Access and Retrieval



The next section gives an overview of the techniques developed in this thesis for the use of automatically extracted semantic knowledge in information access and retrieval.

1.2 Thesis Overview

The thesis addresses three major issues that arise in the application of semantic knowledge to information retrieval. First, how can semantic knowledge be acquired and represented? Second, what techniques are appropriate for the systematic use of semantic knowledge? Third, what kinds of information access and retrieval applications will benefit from the application of semantic knowledge? These three issues are interlinked. The acquisition and representation of semantic knowledge influences the techniques used to incorporate semantic knowledge. In turn, this influences the kinds of retrieval applications that will see improvements in performance through the use of these techniques. The following three sections offer an overview of the methods developed in the thesis to address each of these issues. These issues are described in depth in the next four chapters.

1.2.1 Semantic Knowledge Acquisition and Representation

In computational systems, knowledge and representation go hand in hand. Representations are needed to store knowledge and to apply it as the situation demands. *Inference* is the application of represented knowledge. For instance, given the right representation, a computer system can represent the knowledge that all men are mortals. Then, given a suitable inference mechanism and the fact that "Socrates is a man," the system can infer that Socrates is a mortal. Therefore, representation schemes have significant influence on what knowledge can be represented and how inference takes place.

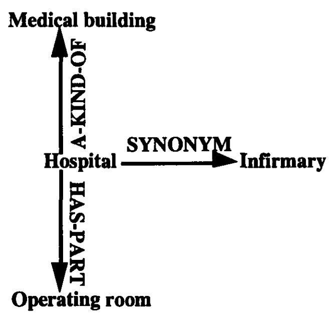
Research in artificial intelligence has resulted in several schemes for the representation of world knowledge. There is a trade-off associated with representation schemes between their expressiveness (the knowledge they can represent) and the efficiency of the inference mechanism. Generally, the more expressive the representation, the less efficient the inference mechanism.

The methods described in this thesis depend on a representation scheme called *semantic networks* [Quillian 1968]. Semantic networks arose out of an attempt to represent the knowledge found in dictionary definitions. A semantic network is composed of *semantic relations*, which is a link of a predefined type between two words or symbols. An example of a semantic relation is ["dog" A-KIND-OF "animal"]. Semantic networks are particularly useful in the context of word-based information retrieval because they offer a basis for generalizing queries and documents. Two kinds of inferences are possible in semantic networks:

- Transitivity; e.g., from ["dog" A-KIND-OF "animal"] and ["animal" A-KIND-OF "organism"], the relation ["dog" A-KIND-OF "organism"] can be inferred.
- Inheritance; e.g., any properties associated with "animal" will be inherited by "dog" unless overruled explicitly.

The algorithms described in the thesis use a manually developed semantic network called WordNet [Miller 1990], and a semantic network extracted automatically by processing definitions from an on-line Webster's dictionary [Chakravarthy 1994a]. A fragment of WordNet is shown in Figure 1.3. WordNet's scope is comparable to that of a mid-size dictionary (Section 2.3.1). The retrieval algorithms described in the thesis are domain-independent by virtue of the domain independence of the sources from which the semantic knowledge is obtained.

FIGURE 1.3. A fragment of WordNet



Section 2.3.2 describes a new method developed in this thesis for extracting semantic knowledge from dictionary definitions. Since dictionaries are carefully edited, the definitions are usually stylized, which makes it possible to develop pattern-based natural language processing programs to extract semantic relations from the definitions. Here is an example from the Webster's on-line dictionary:

> ***Basset hound:*** *any of an old French breed of short-legged, slow-moving, hunting dogs with very long ears and crooked front legs.*

The method described in Section 2.3.2 provides a pattern definition language that can be used to extract semantic relations from single or multiple definitions, including knowledge that is not available through WordNet and other thesauri. The next section introduces some of the problems associated with the use of knowledge from semantic networks in information retrieval, and the solutions advocated in this thesis.

1.2.2 Systematic Use of Semantic Knowledge

How should semantic knowledge be incorporated into an information retrieval system? Ideally, semantic knowledge should be used only through inferences that are valid from two viewpoints: the underlying representations, and the particular retrieval application. Let us consider a simple example of a database of pictures of animals where each picture is captioned with a single term. Figure 1.4 shows one such picture.

FIGURE 1.4. Basset hound

Now if a user is looking for a picture of a dog, using the semantic relation ["basset hound" A-KIND-OF "dog"] makes sense from both viewpoints. From the viewpoint of the application, it can be assumed that the user is looking for a dog, or something that is a dog. From the viewpoint of the representation, since there is only component keyword ("basset hound"), there is no problem in deciding how to use the semantic relation.

But in practice, representations of documents are much harder to get because natural language processing and knowledge representation techniques are not sophisticated enough to construct representations from arbitrary text. Therefore, attempts to use semantic knowledge have focused on *keyword expansion*. In this approach, semantic relations associated with keywords from the query and/or the documents are used to generate other keywords, which then become part of the representation of the query or the document. In effect, retrieval is performed on a larger set of keywords.

Keyword expansion poses three questions, all of which the thesis has tried to address:

- Since not all keywords are significant, what keywords should be expanded?
- Once a keyword is chosen for expansion, what semantic relations should be used to expand it?
- If a keyword chosen for expansion has multiple meanings, i.e., it is ambiguous, which meaning of the keyword should be considered?

The first two problems are addressed through the use of representations, called *case structure representations*, that capture the *roles* being played by the keywords. The roles identify the salient words, and guide the use of semantic knowledge. For example, consider the case structure representation of the sentence "Doctor positioning microscope for microsurgery in an operating room":

AGENT:	doctor
ACTION:	position
PATIENT:	microscope[1]
FOR:	microsurgery
IN:	operating room

Since "operating room" is the location of the action, it can be expanded using the semantic relation ["operating room" PART-OF "hospital"] from WordNet[2], because the container of a location is also a location. This leads

1. The role name "patient" is used to stand for the direct object of the action verb.

to the inference that the action is taking place in a hospital. Therefore, the first problem, selecting salient words, is addressed by choosing only the keywords in the case structure representation for expansion. The second is addressed by using the roles to guide the expansion. Details of the case structure representation are presented in Section 2.2.3.

The third problem is addressed by incorporating a *disambiguator* in building the case structure representation. The disambiguator is described in detail in Chapter 5. Based on the surrounding context of the ambiguous word, the disambiguator uses a set of heuristics to assign one or more senses from the Webster's dictionary or WordNet to the word. For example, one of the heuristics relies on the fact that conjuncted head nouns are likely to refer to objects of the same category. Consider the ambiguous word "snow" in the sentence "Slush and snow filled the roads." In this sentence, a part-of-speech tagger is first used to identify "snow" as a noun. Then, a phrase extractor indicates that "snow" and "slush" are conjuncted head words of a noun phrase. Lastly, the heuristic uses WordNet to identify the senses of "slush" and "snow" that belong to a common category. Therefore, the sense of "snow" as "cocaine" is discarded by this heuristic.

1.2.3 Two Example Applications

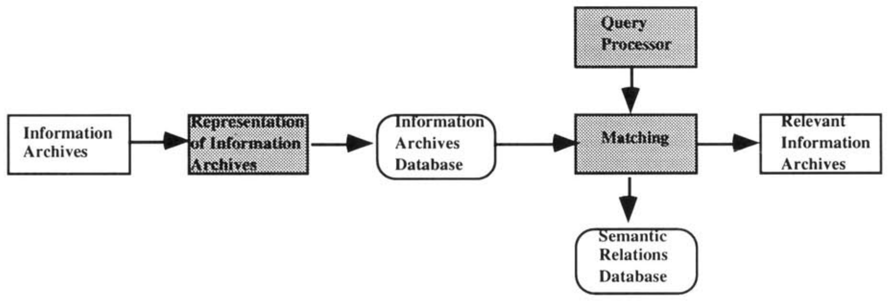
From the discussion in the previous subsections, it is clear that the case structure representation plays a very important role in the targeted application of semantic knowledge. Therefore, this technique is applicable to retrieval applications in which case structure representations can be

2. If one component of a PART-OF relation is a location word, the other component will also likely be a location word.

subjects were asked to describe the photographs and video clips without seeing the captions again. This experiment gathered 280 different descriptions of the photographs and video clips, which were treated as queries. Using this set of queries, ImEngine's performance was compared to that of a standard keyword-based text retrieval system named SMART [Salton 1989].

NetSerf. NetSerf is a program for finding information archives on the Internet using natural language queries (Figure 1.6). NetSerf's query processor extracts case structure representations from the queries. The query representations are matched to hand-coded case structure representations of the archives using semantic knowledge from WordNet and the on-line Webster's dictionary.

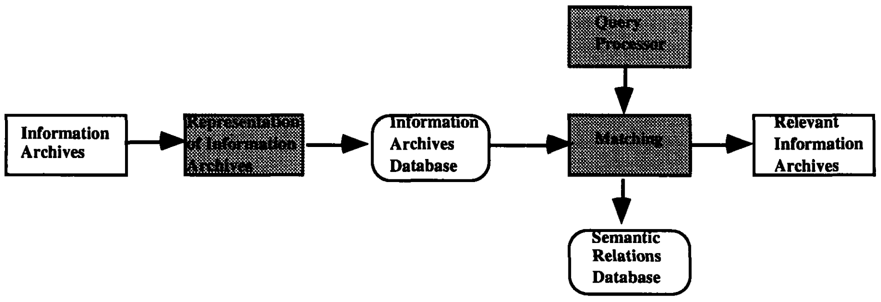
FIGURE 1.6. The Architecture of NetSerf


Query Processor
Information Archives
Representation of Information Archives
Information Archives Database
Matching
Relevant Information Archives
Semantic Relations Database

NetSerf has been evaluated on a set of queries collected from the game *Internet Hunt* [Gates 1992]. Each month, the creator of this game publishes ten questions, which participants are expected to answer using only information available on the Internet. The correct answers are published the following month. Thus, this game provides an independent set of queries with subjects were asked to describe the photographs and video clips without seeing the captions again. This experiment gathered 280 different descriptions of the photographs and video clips, which were treated as queries. Using this set of queries, ImEngine's performance was compared to that of a standard keyword-based text retrieval system named SMART [Salton 1989].

NetSerf. NetSerf is a program for finding information archives on the Internet using natural language queries (Figure 1.6). NetSerf's query processor extracts case structure representations from the queries. The query representations are matched to hand-coded case structure representations of the archives using semantic knowledge from WordNet and the on-line Webster's dictionary.

FIGURE 1.6. The Architecture of NetSerf


Query Processor
Information Archives
Representation of Information Archives
Information Archives Database
Matching
Relevant Information Archives
Semantic Relations Database

NetSerf has been evaluated on a set of queries collected from the game *Internet Hunt* [Gates 1992]. Each month, the creator of this game publishes ten questions, which participants are expected to answer using only information available on the Internet. The correct answers are published the following month. Thus, this game provides an independent set of queries with answers for testing NetSerf. Using this query set, the performance of NetSerf also has been compared to that of SMART.

The three key components of the retrieval framework proposed in this thesis, semantic knowledge, case structure representations and disambiguation, have been tested separately on both ImEngine and NetSerf. The results of these tests are reported in the individual chapters.

The organization of the rest of the thesis is as follows: Chapter 2 deals with the issue of what knowledge is needed for information retrieval and how this knowledge is acquired, represented and applied. Chapter 3 is a detailed description of ImEngine. Chapter 4 presents the architecture, implementation and evaluation of NetSerf. Chapter 5 is a description of the disambiguator, and its influence on the performance of ImEngine and NetSerf. Lastly, Chapter 6 is a summary of the contributions made by the thesis, and a final look at some of the unresolved issues.

CHAPTER 2

Knowledge for Information Retrieval

An information retrieval (IR) system acts as the intermediary between the user and an information database. In order to identify and retrieve relevant information from the database, the system has to have a description of the user's *information need* and a mechanism for searching the database based on the description.

A complete description of the user's information need would include three components: a query that indicates the specific information that the user is looking for, a model of the purpose for which the information is being sought, and a model of the context in which the query is posed. Information retrieval systems typically do not have the wherewithal to get such complete descriptions. Section 2.1 describes in greater detail some methods used currently to obtain descriptions of information need [Belkin and Croft 1992].

Given some description of the user's information need, an IR system uses a retrieval mechanism to match the description to information in the database. This is another hard nut to crack because relevant information in the database is usually not expressed in the same terms as the description of the information need. Therefore, research has yielded a spectrum of methods to search for relevant information, ranging from matching based on keywords to matching based on complex representations of the information. The methods vary primarily along two dimensions: the extent of natural language processing used, and the knowledge that the method brings to bear in matching. These methods, with their attendant pros and cons, are discussed in Section 2.2.

The method proposed in this thesis lies midway on the spectrum along both dimensions. Section 2.2.3 discusses this method in detail with a characterization of the kinds of retrieval applications for which the method is suitable. The section then describes the two semantic knowledge sources, WordNet and the on-line dictionary, which are used by this method.

2.1 Characterization of Information Need

This section discusses the three main characteristics of information need: the format in which the query is presented by the user, the context in which the query is presented, and the purpose for which information is being retrieved. For a general-purpose information retrieval system which does not possess special knowledge of any particular domain, all three aspects pose difficult questions.

General-purpose IR systems support one or more of three formats for constructing queries:

- Boolean queries, which are boolean combinations of words or word compounds provided by the user to characterize relevant information. For example, the query ["President" AND (NOT "Reagan")] indicates that the user is looking for documents that have the word "President" but not the word "Reagan."
- Natural language queries [Salton 1989], where the system expects the query to be a natural language description without any special logical operators.
- Structured natural language queries, e.g. [Harman 1993], where the query consists of two parts: a natural language description and a set of special features such as "AUTHOR," "PUBLISHER," etc. which are used to provide specific information about desired documents.

Boolean querying is the oldest format of the three and it is in wide use in commercial systems such as DIALOG [Dialog 1995]. But user studies indicate that Boolean querying is not intuitive to many naive users. Research also shows that natural language queries are at least as effective for users at retrieving relevant documents while making it easier to frame queries [Turtle 1994]. Structured natural language queries are becoming more popular thanks to the TREC competition [Harman 1993]. Their appeal lies in the fact that the special features enable retrieval to be more precise.

The second aspect of information need is the context in which the query is presented to the system. Ideally, the system's understanding of the context should enable it to judge the user's level of sophistication and knowledge so that the retrieved documents are in accord with the user's expectations. In practice of course, this is very difficult to achieve for a general-purpose IR system that does not have any knowledge of the domain of the information in the database. For a system to be sensitive to the context of retrieval, the best current method is *relevance feedback* [Salton 1989]. In this method, the user is prompted to pick relevant documents from the set retrieved in response to previous queries. From these documents, the system usually gets a greater number of relevant keywords, and is therefore able to do a better job of retrieving documents "similar" to the ones picked by the user.

The final aspect of information need, the ability of the IR system to be informed about the goals or the tasks of the user, is an exercise in cognitive modeling which has proved to be quite difficult, e.g., [Brajnik et al. 1990, Orwant 1991]. In a few cases, depending on the database, the system can make default assumptions about the user's goals; e.g., with reasonable accuracy, it can be assumed that users of the MEDLINE commercial system are health care professionals. Some authors, Winograd and Flores in particular [Winograd & Flores 1986], have argued that a computer system can never be aware of the user's goals and situational motivations.

2.2 Matching Queries to Information

This section examines several methods that have been proposed for matching user queries to information in a database. The section deals with the matching of both structured and unstructured natural language queries. Systems that support boolean queries typically use a straightforward word-matching method that takes the logical operators used into account.

Very rarely, if ever, are queries exactly identical to relevant information. Therefore, in order to match queries to relevant information that happens to be expressed differently, all matching methods employ processing techniques to capture the essence of the text while ignoring the extraneous parts. In other words, matching methods create *representations* (or *surrogates* [Belkin and Croft 1992]) for the queries and the information in the database, which are used as the input for the matching process.

In addition to work in information retrieval, the field of knowledge representation [Brachman & Levesque 1985], a branch of artificial intelligence, has devoted enormous effort to devising representations that enable computer systems to use knowledge intelligently. Some of this work has been applied to information retrieval tasks, e.g. [Jacobs & Rau 1990, Sundheim 1993, Lehnert et al. 1993] (Section 2.2.4).

The overall architecture of a matching method in an IR system therefore consists of three steps:

- Formulating the details of the representation
- Defining a scheme for translating queries and information (text) into the specified representation
- Defining an algorithm for matching query representations to database information representations

The rest of this section describes four different representations, each of which corresponds to one or more architectures. Table 2.1 is a summary of the four representations. The introduction of case structure representations for information retrieval is one of the key contributions of this thesis.

TABLE 2.1. **Summary of representations and matching methods**

Representation Element	Extracting Representations	Matching Representations
Keyword	Morphological analysis	Word Matching
Grammatical Relation	Part-of-speech tagging or Parsing	Grammatical structure matching
Case structure	Part-of-speech tagging or Parsing and Basic Pattern Matching	Case Structure matching with general semantic knowledge
Inference structure	Parsing and Extensive Pattern Matching	Structure matching with specialized knowledge

2.2.1 Keyword-based Retrieval

A keyword can either be a single word like "acetylene" or a collocation (compound) of words like "parallel processing." Retrieval systems usually represent keywords in their basic morphological form, e.g., "process" is used instead of "processing." Keyword representations can be created either manually or automatically. If the keywords are chosen manually, they are usually used to capture the topics of discourse of the document. For instance, a news article might be annotated with the keywords "computer" and "manufacture" to describe its contents. If keywords are chosen automatically, they are typically selected from within the document itself based on statistics pertaining to the relative frequency of word occurence. For example, in the sentence "John tipped the waiter," the word "the" would certainly not be considered a keyword. In addition, many systems use some form of *term weighting,* which is a measure of the relative importance of a keyword in a document or a text unit [Salton 1989]. A common metric is to multiply the *term frequency* by the *inverse document frequency*. Term frequency is a count of how often the keyword is found within a particular text unit, *T*, which is a measure of how well the term characterizes *T*. Inverse document frequency is a measure of how rarely this term occurs in other text units. If it occurs rarely in other text units, it implies that it is highly correlated with *T*.

There are primarily two matching methods for keyword representations: boolean matching and matching based on term weighting. In boolean matching, the logical operators govern the selection of text units for retrieval. In matching methods that use term weighting, given a query, a weight is assigned to every term to reflect the degree of relevance of the text unit. This weight is computed by combining the term weights of the terms that the query and text have in common.

Keyword-based systems such as SMART [Salton 1989] are the most widely used general-purpose information retrieval systems. This is due to three advantages that these systems offer: ease of acquisition, universality, and real-time performance. Keyword representations are easy to acquire because the natural language processing involved is minimal. At most, the system needs a lexicon to construct root forms of the keywords. Universality is perhaps the most important factor. No matter what the text units and the queries are about, the underlying algorithm relies only on matching keywords to keywords and is therefore completely domain-independent. The simplicity of keyword methods also leads to very efficient retrieval performance. Real-time performance is aided by the use of an *inverted index*, which for every keyword compiles a list of occurrences of the keyword in text units.

There are two main drawbacks of keyword-based systems that motivate the use of other representations:

- Keyword-based systems operate on the assumption that if the text unit and the query have some keyword in common, the text unit must be relevant to the query. This is not necessarily true.
- A single concept can be annotated with several distinct sets of keywords, which means that even if the query and the text unit refer to the same concept, they might be expressed quite differently.

Broadly speaking, the two issues affect *precision* and *recall* respectively. Precision is defined as the percentage of relevant text units among all the retrieved text units. Recall, on the other hand, is the percentage of all relevant text units in the database that are actually retrieved by the system. There is a natural trade-off between the two measures. In general, recall can be increased by retrieving more text units from the database, while precision can be increased by retrieving fewer units.

Early on, vocabulary control was used to address these two problems [Lancaster 1972]. Such systems select keywords (typically manually) from a limited, controlled vocabulary to index text units. They also require users to adhere to the controlled vocabulary in constructing queries. The expectation was that the use of a controlled vocabulary would limit the variety of ways in which a concept could be encoded (leading to better recall), and also provide unique descriptors (for higher precision). Despite all the labor invested, the use of controlled vocabularies has not shown any clear benefits over automatic indexing with an unlimited vocabulary [Salton 1989]. The studies indicate that despite the controlled vocabulary, different human indexers treat the same text unit in very different ways. This happens for two reasons: different indexers pay attention to different aspects of the same document, or even within the restricted vocabulary, they find different word choices to index the same document.

Many attempts have been made to improve recall using the technique of *keyword expansion.* In this approach, keywords from the query and/or the text units are used to generate other keywords that also become part of the representation of the query or the text unit. The new keywords are taken from a thesaurus, which can be any of the following:

- A standard thesaurus like Roget's thesaurus, which provides classes or categories of words [Kirkpatrick 1988]
- A specialized thesaurus like WordNet [Miller 1990], which relates words to other words using a predefined vocabulary of semantic relations. For example, a "basset hound" is a kind of "dog." (More about WordNet in Section 2.3.1)
- A thesaurus of frequently co-occuring words collected from a large collection of text units. Such a thesaurus is usually complementary to semantic thesauri [Jing & Croft 1994].
- A specialized, domain-specific thesaurus like MeSH (Medical Subject Headings) [Rada & Bicknell 1989]. Research in this area is described in Section 2.3.3 following the discussion on semantic networks.

The use of semantic thesauri for keyword expansion has not resulted in improvement over vanilla keyword-based retrieval. This is because of the difficulty in deciding which words to expand [Voorhees 1994], and which semantic relations to apply during the expansion, since a given word might have many associated semantic relations. This will be discussed in detail in the section on case structure representations (Section 2.2.3). Nor have co-occurence thesauri led to improved performance, although there are indications in recent work that such improvements might be in the offing [Jing & Croft 1994].

2.2.2 Grammatical Relations as Representations

The purpose of using grammatical relations (instead of keywords) as representations is to improve precision during retrieval. For example, consider the input "Clinton said" which might be part of a larger natural language query. This will match not only text units of Clinton saying something, but also text units of something being said about Clinton or something said in Clinton, Massachusetts [Haase 1994]. But instead of matching keywords, if we were to match keyword relations, this problem can be overcome. For example, in the sentence, "John tipped the waiter," the representation would consist of the relations [John tipped] and [tipped waiter], rather than the three keywords. [Haase 1991] describes a "multi-scale" parser that can extract such representations.

The prominent work in this area has been the Constituent Object Parser-based retrieval system [Metzler & Haas 1989] and multi-scale parsing [Haase 1995]. So far, significant improvement has not been reported over keyword-based systems, but there are promising signs that effective, domain-independent retrieval systems would eventually be possible [Haase 1995].

2.2.3 Case Structure Representations for Retrieval

The case structure representation provides the underlying framework for the information access and retrieval applications described in this thesis. The representation is derived from an extensive body of research in computational linguistics. This section describes the origin of the representation, its applicability to information retrieval, and its limitations.

The case structure representation was pioneered by Fillmore in his work on case grammar [Fillmore 1968]. A case grammar analysis of a sentence results in the assignment of *roles* for salient words in the sentence. Alternatively, these are called *cases* or *thematic roles* or *theta-roles*. Case structure representations therefore help in identifying the salient words in a sentence in addition to identifying the roles they play. For instance, in the sentence "John tipped the waiter," the following assignments would be made:

AGENT:	John
ACTION:	tip
PATIENT:	waiter

From the viewpoint of keyword expansion, the case structure representation offers a solution to the problem of deciding which words to expand and which semantic relations to apply during the expansion. As mentioned earlier [Voorhees 1994], this problem limits the effectiveness of using semantic knowledge with brute-force keyword expansion. [Cohen & Kjeldsen 1987] describe an occurence of this problem in a system that was designed to match grant proposals to descriptions of funding agencies (Section 2.3.3). Their system relied on generalization of all keywords (through a hand-coded semantic network) to find relevant matches. But, when the wrong keyword was chosen for expansion, this method yielded poor results. For example, the system matched the query "economic impact of dandelions on landscaping" to the agency description "reproduction in plants," because the keyword "dandelion" was generalized to match "plant" without regard to their semantic contexts. As [Voorhees 1994, page 68] puts it, the lesson here is that "the challenge now lies in finding an automatic procedure that is able to select appropriate concepts to expand."

By identifying salient words in the text and the roles they play, the case structure representation can help in the incorporation of semantic knowledge. For instance, the case structure representation of the caption "Doctor positioning microscope for microsurgery in an operating room" is:

AGENT:	doctor
ACTION:	position
PATIENT:	microscope
FOR:	microsurgery
IN:	operating room

Given this representation, semantic knowledge can be applied in the following way. Since the action is in an "operating room" and an "operating room" is part of a "hospital," the action is in a "hospital." Therefore, the roles played by the salient words guide the use of semantic relations associated with the words.

The extraction of case structure representations from queries and text units parallels the extraction of grammatical relations. A part-of-speech tagger along with a phrase extractor or a parser identifies the main segments of the input sentence, which in turn leads to the identification of the salient words and their roles. Depending on the application and the role, semantic relations are used to generate new cases (<role, word> pairs). For instance, in the caption example above, we saw the "PART-OF" relation being used to generate the case <IN hospital>. Therefore, expansion here is not keyword expansion but case expansion. Later in this chapter, Section 2.3 describes the two semantic knowledge sources that have been used extensively in this thesis, WordNet and the Webster's on-line dictionary.

Case structure representations are well-defined only over single sentences, and work only for "standard" sentences. For instance, case structure representations are not suitable for imperative sentences like "Just do it" or interrogative sentences like "Why did the chicken cross the road?" Therefore, this method works only for retrieval applications where the query and the text unit are single "standard" sentences. But there are several applications that fit into this framework. Two examples: Chapter 3 shows how case structure representations are effective for working with picture and video clip captions, and Chapter 4 describes the use of such representations on descriptions of Internet information archives.

2.2.4 Retrieval with Inference Structures

Thus far, we have been dealing with representations as surrogates for the queries and the text units. In contrast, a representation in artificial intelligence (AI) is a construction that allows the computer system to *infer* facts that are implied by the represented facts. Knowledge representation plays an important role in all artificial intelligence applications such as natural language understanding, robotics, machine vision, etc. [Brachman & Levesque 1985].

To give an example of two such representations, predicate logic and scripts, let us return to the familiar sentence "John tipped the waiter." In predicate logic [Moore 1982], which is a popular representation language, this sentence would be represented by a predicate "tip" which takes "John" and "waiter" as arguments. If the system has rules that allow inferences to be made from the predicate "tip" (background knowledge about tipping), predicate logic enables the system to adapt the rule to this particular case involving John. For example, using the background knowledge, the system might be able to infer that John gave money to the waiter.

If scripts are used as the underlying representation [Schank 1977], the idea is to tie this particular action of tipping into a larger context. For example, the system might have access to a script that deals with visits to restaurants. Then, the restaurant script can serve as the larger context for this sentence. Again, this would let the system infer other facts, e.g., that John visited a restaurant, gave money to a waiter, and so on.

Clearly, AI representations are more powerful than the other representation schemes that we have seen so far. The hard part lies in constructing such representations for arbitrary text in a domain-independent fashion. As we see from the two examples above, an AI-based retrieval system would need considerable background knowledge about the world before such representations can be built and used accurately.

In extracting representations from text, there are two main problems. First, to be effective, the system needs a vast library of background facts and rules, e.g., scripts or predicates about tipping, restaurants, etc. There is no agreed-upon estimate of the size of this "commonsense knowledge." The second problem is to actually associate text with a particular representation. This is difficult for two reasons: a given concept can be expressed in a variety of textual forms, and a textual form can potentially correspond to different underlying representations because of the ambiguity inherent in language.

Therefore, AI-based retrieval systems have so far concentrated on one or a small number of domains. For example, [Lehnert et al. 1993] deals with episodes of terrorism from the MUC corpora by making use of a large body of heuristics and facts about this domain. The SCISOR system [Jacobs & Rau 1990] handles articles about mergers and acquisitions, and has been extended to some other domains. In their domain areas, these systems have generally exhibited better performance than keyword-based systems. But for an AI-based retrieval system to handle text with the same versatility as a keyword-based retrieval system, there is a need for a knowledge library that captures the "commonsense knowledge" that humans use to process language.

The CYC project [Lenat & Guha 1990] aims to fill this need. CYC aims to capture the common sense shared by most people, i.e., "the factual and heuristic knowledge, much of it usually left unstated, that comprises 'consensus reality': the things we assume everybody already knows" [Lenat & Guha 1990, Page 28]. CYC was built with the aim of removing the "knowledge bottleneck" that all artificial intelligence systems encounter during "scale up." So for example, it was envisioned that CYC would enable expert systems to submit their facts and inferences to real-world constraint checks, and that natural language processing systems could use CYC to attack tough problems like word-sense disambiguation and pronoun resolution.

Since CYC's scope is broader than other previous knowledge representation projects, it has produced novel solutions for issues in representing time, space, number, etc. as they apply to real-world situations. But this also makes it difficult to estimate the extent of CYC's coverage as it cannot be compared with standard benchmarks like dictionaries or encyclopediae. Therefore, it is very hard to predict what impact CYC can have on information retrieval. It should be noted that CYC's goal was never to represent the encyclopedia. An encyclopedia presumes a great deal of common sense on the part of its readers. The intention of CYC's creators was to encode "commonsense knowledge," which in a sense, is the complement of encyclopedic knowledge.

CYC is now being used to support retrieval systems. [Lenat & Guha 1994] reported early results on using CYC in a caption retrieval system, with an example of how it could outdo other methods. When presented with the query, "Photograph of a man not wearing a shirt," the CYC-based system retrieved a photograph of "Pablo Moraes swimming," by using its knowledge base to infer that Pablo Moraes is the name of a man, and that men typically do not wear shirts while swimming. While the example is impressive, it is hard to predict how much commonsense knowledge a CYC-based scheme would need before it can match the universality of keyword-based systems. Additional user studies are also necessary to ensure that retrieval performance does indeed improve (from the user's viewpoint) because of the ability to carry out such detailed inferences.

The viewpoint advanced by this thesis is that the case structure representation is a useful compromise between the universality of keyword-based systems and the inferential power of AI-based systems. For a retrieval system to be generally useful, its mechanisms should not be domain-dependent. The case structure representation is not domain-dependent but *structure-dependent*, i.e., the natural language processing involved expects the input to correspond to a prespecified structural framework. For example, in the work on picture and video clip retrieval discussed in Chapter 3, it is assumed that the captions are structurally predictable but no assumptions are made about their content.

The second element of the compromise is that case structure representations are designed to work with general-purpose sources of semantic knowledge like dictionaries and thesauri. A significant problem in the attempt to encode "commonsense knowledge" (as in CYC) is deciding where to draw the line, i.e., what is and what is not commonsense knowledge. For example, can we consider SCISOR's knowledge of mergers and acquisitions as commonsense knowledge worthy of inclusion in CYC? Or the knowledge of terrorist stories that many MUC programs have? In case structure representations, we skirt this question by using linguistic resources like dictionaries that are designed primarily for human use. The system can make use of whatever knowledge is accessible in the resources, and hence, system-builders do not have to make difficult decisions about what is and is not the "consensus reality" that CYC tries to capture. The next section is a detailed description of the two standard linguistic resources that have been used extensively in this thesis, the Webster's on-line dictionary and WordNet.

2.3 Semantic Knowledge Sources

The section on AI-based retrieval systems (Section 2.2.4) discussed predicate logic and scripts. This section starts with a description of another well-known knowledge representation scheme, *semantic networks*[1]. The two large semantic knowledge sources that are discussed later in the section both fit into the semantic network formalism.

Semantic networks are composed of *semantic relations*, which are structures that provide a means of expressing knowledge about the world. The basis of this idea comes from classical logic, where the goal was to formulate true statements about individual objects in the world and relations between these objects. In semantic relations however, the relation is usually defined between words or terms rather than objects.

TABLE 2.2. Some semantic relations (after [Fox 1980])

Relation	Related Words
Set	Sheep —> Flock
Substance	Tire —> Rubber
Color	Tomato —> Red
Part of	Tusk —> Elephant
Location	Toilet —> Bathroom
Time	Supper —> Evening
Kind of	Traipse —>Walk
Permission	Drop —> Fall

1. [Brachman & Levesque 1985] includes many papers discussing the relationship of semantic networks and scripts to predicate logic.

Semantic networks are generally built around the A-KIND-OF relation, e.g., ["lion" A-KIND-OF "animal"]. The use of this relation dates from antiquity and is the guiding principle for classification and taxonomy. But there are many other semantic relations for words denoting objects, object attributes and actions. Table 2.2 lists some of these relations with examples. Semantic relations try to capture "typical" knowledge about the world, while ignoring the inevitable exceptions.

The first semantic networks were introduced by Quillian [Quillian 1968], who intended them to be representations of words, i.e., word concepts for dictionary entries. Quillian was attempting to construct a computational model of the ability of the human mind to associate words with related words. In his model, a word sense in the dictionary corresponded to a node (vertex) in the semantic network. Therefore, for an ambiguous word like "plant," there would be as many corresponding nodes as dictionary senses. Quillian's work brought up three issues that still continue to play an important role: default inheritance, spreading activation, and representation of ambiguous words.

If we know the two relations ["basset hound" A-KIND-OF "dog"] and ["dog" A-KIND-OF "animal"], transitivity enables the inference that ["basset hound" A-KIND-OF "animal"]. Default inheritance is an inference mechanism that makes it possible to transfer properties through the A-KIND-OF relation. Any properties that are associated with dogs automatically apply to basset hounds, unless they are explicitly overridden. For example, if dogs can play the role of pets, so can basset hounds.

Default inheritance is one way of finding indirect relations between nodes in a semantic network, i.e., it is an inference mechanism. Quillian identified another inference mechanism, spreading activation, in his quest to relate the semantic network model to human associative memory. Spreading activation is an iterative process that begins at a particular node in the semantic network. The first step is to "visit" every node that is linked to the start node. In each subsequent iteration, the process visits every node that is linked to any node visited during the previous iteration. Spreading activation models have been used widely in research on the associative aspects of memory; [Anderson 1990] includes an introductory survey.

Quillian's idea of using a different node for each sense of the word has contributed to research in automatic word-sense disambiguation. The general framework is that given a word in context, each of its nodes can be assigned a weight based on the context. The node that is assigned the greatest weight is the "correct" sense in the given context. Models of disambiguation are covered in Chapter 5.

Quillian's work quickly attracted attention and many research programs were soon using semantic networks for various problems in artificial intelligence. But, as pointed out by [Woods 1975] and [Brachman 1979], much of this work did not have a clear idea of the semantics behind semantic networks. Since much of artificial intelligence depends on the ability to make correct inferences, "free-form" semantic networks were mostly discarded as knowledge representations.

But Quillian's work has always retained its influence in cognitive psychology, especially the branch of cognitive psychology dealing with language (psycholinguistics). As the next two sections show, Quillian's explorations into converting dictionary information into knowledge representations has led to two important sources of semantic knowledge: WordNet [Miller & Fellbaum 1991] and processed on-line dictionaries. While at times the two sources seem opposed to each other, they are really complementary as this quote from WordNet's progenitor, George Miller, indicates [Miller 1985]:

> *There seem to be two approaches to the vocabulary problem. One uses a machine-readable version of some traditional dictionary and tries to adapt it to the needs of a language processing system. Call this the "book" approach. The other writes lexical entries for some fragment of the English lexicon, but formulates those entries in a notation that is convenient for computational manipulation. Call this the "demo" approach. ... The real problem, therefore, is how to combine these two approaches: how to attain the coverage of a traditional dictionary in a computationally convenient form.*

2.3.1 WordNet

WordNet is a large, manually-constructed semantic network built at Princeton University by George Miller and his colleagues [Miller 1990, Miller & Fellbaum 1991]. The goal of the project is to produce a dictionary that can be searched conceptually as well as by regular methods. WordNet is a realization of Quillian's aim of converting the lexical and semantic knowledge in a dictionary into a format suitable for computational processing.

The basic unit of WordNet is a set of synonyms, called a *synset*. A word (or a word collocation like "fountain pen") can occur in any number of synsets, with each synset reflecting a different sense (meaning) of the word. The synsets are organized by part of speech into four divisions: noun, verb, adjective and adverb. Synsets are used by the disambiguation algorithm described in Chapter 5 to assign one or more senses to a word depending on its context.

TABLE 2.3. WordNet semantic relations

Relation	**Related Synsets**
Hypernym	[mortar, howitzer] → [cannon]
Hyponym	[hound, hound_dog] → [basset, basset_hound]
Meronym	[tree_branch] → [tree]
Holonym	[table] → [leg]
Entailment	[watch, look_attentively] → [look, take_a_look]
Causation	[drop, let_fall] → [drop, fall_vertically]
Pertainym	[Indian] → [India, Bharat]
Antonym	[benevolent, good] → [malevolent]

WordNet uses a vocabulary of 12 lexical-semantic relations to link synsets to other synsets. Some of these semantic relations are shown in Table 2.2. Synonyms and antonyms are provided for all kinds of synsets. As in earlier work on semantic relations, the basic relation in WordNet is A-KIND-OF (which is called *hypernymy* here). All hypernym relations in WordNet are bidirectional, the inverse relation being referred to as *hyponymy.* For nouns, other relations provided are *meronymy,* which includes PART-OF, SUB-STANCE-OF and MEMBER-OF, and its inverse, *holonymy.* For verbs, other semantic relations provided are *entailment* and *causation.* Derived adjectives are linked to the nouns they are derived from (which are called *pertainyms*). Adverbs are linked to adjectives they are derived from (if any).

WordNet meshes well with the case structure representation for information retrieval because of its wide coverage and its fixed vocabulary of semantic relations. WordNet 1.3, the version used in this thesis, is quite large, with well over 30,000 words and 60,000 synsets constructed from them. This is of the same order of magnitude as a dictionary. In contrast to CYC, WordNet's coverage is "shallow" but "broad." As the experience with AI-based retrieval systems shows, knowledge sources for information retrieval would ideally be deep and broad, but failing that, a useful source should be shallow and broad, rather than deep and narrow. WordNet however may not be very useful for AI applications since there isn't an explicit model of semantics supporting it.

2.3.2 Semantic Networks from On-line Dictionaries

Apart from WordNet, there is an older thread of research that also builds on Quillian's vision. From the late seventies, regular dictionaries became available on-line to computational linguistics researchers. Recognizing that dictionaries were rigorously edited and structured resources, many researchers began to develop algorithms to extract semantic knowledge using the structure.

Long before on-line dictionaries became available, it had been recognized in computational linguistics that significant semantic knowledge was necessary to tackle such problems as prepositional attachment, word-sense disambiguation, and pronoun resolution. For example, [Katz and Fodor 1963] recommended that the lexical entries of words should provide *semantic features* to distinguish them as animate/inanimate, mobile/immobile, etc.

When dictionaries became available, it was felt that such features could be extracted automatically from them.

The first dictionary that became available on-line was the Longman's Dictionary of Contemporary English (LDOCE), which is aimed at users learning English as a second language [Boguraev & Briscoe 1987]. Most of the entries in LDOCE are written in Basic English, a 800-word subset of English. These two factors have combined to ensure that most work on dictionary processing have been carried out on LDOCE , e.g. [Alshawi 1987, Amsler 1980, Chodorow et al. 1985, Dolan et al. 1993, Fox et al. 1988, Janssen 1990].

In this thesis, however, the Webster's Seventh dictionary has been used as a source of semantic knowledge. Even though the Webster's is larger and more complex linguistically than LDOCE, it is more easily available. But like all dictionaries, the Webster's is quite stylized, making it possible to define fairly simple patterns to extract semantic information. For instance, noun definitions usually consist of a *genus* term identifying the category, followed by *differentiae* that distinguish the noun being defined from its genus category. In the example below, the genus of "basset hound" is "hunting dog" while the differentiae are "very long ears" and "crooked front legs."

> ***Basset hound:*** *any of an old French breed of short-legged, slow-moving, hunting dogs with very long ears and crooked front legs.*

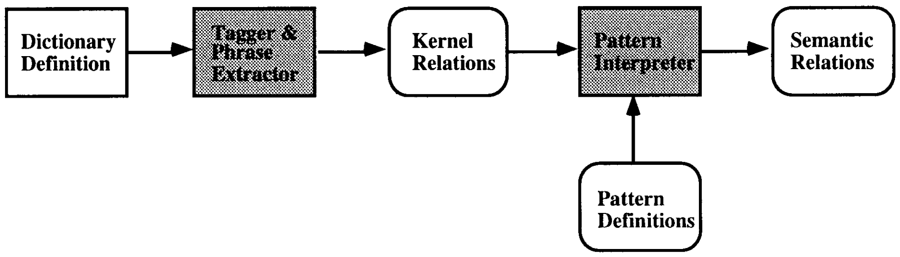
The semantic knowledge extraction program, which extracts a semantic network from the Webster's dictionary, was written as part of a project to retrieve dictionary pictures using semantic knowledge [Chakravarthy

1994a]. For caption-based picture retrieval, dictionary pictures represent the simplest case since the caption is essentially a single word or word compound defined in the dictionary (Figure 2.1).

FIGURE 2.1. Basset hound

The extraction program consists of three main components as shown in Figure 2.2: a part-of-speech tagger, a phrase/clause extractor and a pattern interpreter. The functioning of the extraction program will be illustrated using the definition "**aqueduct**: a conduit for *water.*"

FIGURE 2.2. Extraction of semantic relations from dictionary definitions



For part-of-speech tagging, we used the standard Xerox PARC tagger [Cutting et al. 1992] to segment the definition and assign a part of speech to each word. On the aqueduct definition, the output of the tagger is ((:at "a") (:nn "conduit") (:in "for") (:nn "water")). Here, :at is the tag for article, :nn for singular or mass noun, and :in for preposition.

Based on the output of the tagger, the phrase/clause extractor identifies noun, verb and prepositional phrases and relative clauses. This extractor generates *kernel relations (triplets)* which link words or word compounds in the dictionary definition to other words or compounds in the definition. The triplets are themselves basic semantic relations. For example, in the definition of "aqueduct," the triplets are ["aqueduct" A-KIND-OF "conduit"] and ["conduit" FOR "water"].

The pattern interpreter takes this idea further by providing a *pattern definition language*. Based on the triplets extracted by the phrase/clause extractor, this language enables pattern extractors to be defined for other semantic relations. For example, consider the semantic relation "HAS-PURPOSE." Here is a pattern for this relation defined in the pattern definition language:

```
(($ANNOTATION+ "A-KIND-OF" $OBJECT)
 ($OBJECT "FOR" $QUERY+))
```

When applied to the kernel relations extracted from the definition for "aqueduct," it has the following effect. $ANNOTATION+ is first bound to "aqueduct." Then, using ["aqueduct" A-KIND-OF "conduit"], the pattern interpreter binds $OBJECT to "conduit." Next, this leads to $QUERY+ being bound to "water." Since the pattern has been consistently and successfully matched using the kernel relations, the pattern interpreter generates the semantic relation ["aqueduct" HAS-PURPOSE "water"]. To take a second example, the same pattern would generate the relation ["perch" HAS-PURPOSE "bird"] from the definition "**perch**: a roost for a *bird*."

In addition, the triplets do not have to come from the same definition. Patterns can be defined to "chain" over multiple definitions in looking for particular semantic relations. For example, the relation ["atomizer" OCCURS-WITH "spray"] is extracted by chaining over the following definitions:

- "**atomizer**: an instrument for *atomizing* usually perfume, disinfectant, or medicament"
- "**atomize**: to reduce to minute particles or to a fine *spray*"

TABLE 2.4. Semantic relations for dictionary picture retrieval

Relation	**Related Words**
Plays-role-of	cat → pet
Associated-with	sail → water
Typical-action	arrow → shoot
Collocated-with	violin → bow
Purpose	aqueduct → water

Using this pattern definition language, the dictionary picture retrieval system extracted a number of specialized semantic relations for use in retrieval. Examples of these relations are shown in Table 2.4. One of the advantages of the dictionary approach over WordNet is the ability to extract "non-standard" semantic relations (like HAS-PURPOSE and OCCURS-WITH) relatively easily once the processing infrastructure is in place. Although ImEngine and NetSerf do not use exactly the same relations as the dictionary picture retrieval system, the groundwork laid there has enabled the development of customized knowledge extractors and other processors for these applications.

But the dictionary approach has its drawbacks, e.g. [Veronis & Ide 1991]. The most important problems are the following:

- Different dictionaries provide different information: Dictionaries are written with the view that human readers can "fill in" any gaps. But when the information in the dictionaries is used literally by computers, undesirable distinctions arise across dictionaries.
- Inconsistencies in creating semantic networks: Dictionaries are meant to be read one definition at a time. But when semantic knowledge is extracted, it is composed into networks, which sometimes results in inconsistencies.
- Ambiguity of semantic relations: This again affects chaining of semantic relations. For example, let us turn again to the extraction of the relation ["atomizer" OCCURS-WITH "spray"] discussed previously. This was achieved by chaining over the word "atomize." This word has another definition "**atomize**: to subject to atom bombing" which is not appropriate in this context. But since the extraction program cannot disambiguate the word "atomize," it will chain over both definitions.

Recent research suggests that these problems are not insurmountable. A group at Microsoft [Dolan et al. 1993] has produced a version of LDOCE where all words in the definitions have been successfully disambiguated semi-automatically. One of the conclusions of a recent workshop on the "Future of the Dictionary" was that inconsistencies could be ironed out by the combined efforts of lexicographers[1] and computational linguists [Briscoe 1994]. Finally, there is the hope that the experience gained by the analysis of standard dictionaries will transfer to other definitional resources

1. **Lexicographer**: a writer of dictionaries, a harmless drudge [Samuel Johnson, *Dictionary*, p. 296, 1755]

such as technical dictionaries, encyclopediae, factbooks etc. That would lead to relatively painless knowledge acquisition mechanisms.

2.3.3 Information Retrieval with Semantic Relations

This section is a description of some of the existing body of research on the use of semantic relations and semantic networks for keyword expansion. The work of [Fox 1980], [Rada & Bicknell 1989], [Cohen & Kjeldsen 1987] and [Voorhees 1994] is reviewed.

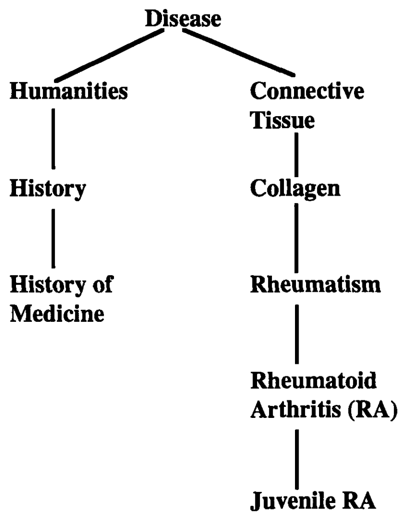
Fox [Fox 1980] investigated the effect of query expansion using a hand-coded set of 103 lexical relations (some of which are shown in Table 2.2) on a database of 82 documents. These relations were organized into 5 groups as shown in Table 2.5, each of which was tested separately. The process of expansion was straightforward. Say group *G* was being tested. Then, in comparing a query to a document, all words that were related by any relation in *G* to some keyword in the query were also considered valid keywords. Fox's study yielded general patterns of retrieval behavior for the different groups as shown in Table 2.5.

TABLE 2.5. Query expansion using groups of relations (from [Fox 1980])

Group	Relation Types	Behavior
Strongly Related	Predicate, Paradigmatic	Improve Precision & Recall
Related	Case, Attribute, Collocational	Improve Recall
Broadening	Taxonomy, Grading, Part-whole	Improve Recall; Reduce Precision
Opposition	Antonyms	Improve Recall
Synonymy	Synonyms	Improve Recall

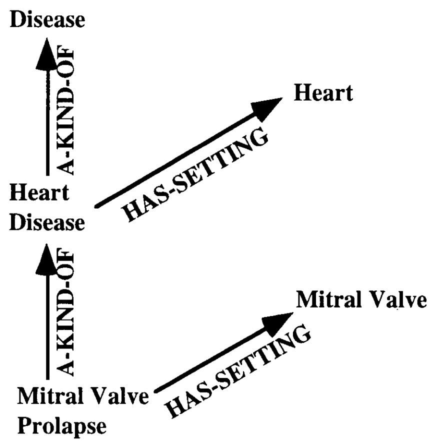
But not all semantic relations are useful across the board in retrieval matching. Rada and Bicknell [Rada & Bicknell 1989] and Cohen and Kjeldsen [Cohen & Kjeldsen 1987] used semantic relations in a much more focused fashion. The first study was conducted with the MeSH (Medical Subject Headings) semantic network of the National Library of Medicine. The MeSH network is organized around a single semantic relation, "broader than." For instance, the fragment shown in Figure 2.3 illustrates that "Rheumatism" is broader than "Rheumatoid Arthritis," which in turn is broader than "Juvenile Rheumatoid Arthritis." In the study, keywords in the query were expanded by the addition of terms that they were broader than. Rada and Bicknell found that the results of this process agreed well with users' intuitions about relevant documents.

FIGURE 2.3. Fragment of MeSH



Cohen and Kjeldsen built a system named GRANT whose objective was to match funding proposals to funding agencies using a constrained spreading activation technique. The authors constructed a semantic network of research topics using a vocabulary of 24 semantic relations and inverses (a fragment is shown in Figure 2.4). Given a query, spreading activation was initiated from each of the keywords. The activation was constrained to traverse only links of the types A-KIND-OF, HAS-SETTING and PART-OF. The rationale for this constraint comes from the nature of the specification of interests provided by funding agencies. If an agency states that it is interested in heart disease research, it is likely to provide funding for mitral valve prolapse research. Using this technique, the authors reported a significant increase in recall over keyword-based systems. But the lack of role or structure information about the text makes it difficult to select the right keywords for expansion, especially if a general-purpose thesaurus is used (as described in Section 2.2.3).

FIGURE 2.4. Fragment of GRANT's network



Voorhees [Voorhees and Hou 1993, Voorhees 1994] pioneered the use of WordNet in keyword expansion for text retrieval. [Voorhees 1994] tackled the problem of selecting the right keywords for expansion by conducting an experiment where keywords (and their senses) were chosen by hand. Therefore, this is a best-case result. Four kinds of expansion were tried: expansion by synonyms only, expansion by synonyms and all hyponyms, expansion by synonyms and all hyponyms and hypernyms, and finally, expansion by all directly related concepts. Voorhees concluded that there was "little difference in retrieval effectiveness if the original queries are relatively complete descriptions of the information being sought even when the concepts to be expanded are selected by hand. Less well developed queries can be significantly improved by expansion of hand-chosen concepts." This result squares well with the applicability of the case structure representation formalism.

2.4 Summary of the Case Structure Framework

This chapter has offered a description of why an information retrieval system needs knowledge of the world and has outlined some methods that have tried to work towards this goal. To date, the most successful have been keyword-based retrieval methods which do not need much knowledge about the world. Instead, they rely on the assumption that a keyword in common between a document and a query indicates that they are related in some fashion that might be of interest to the user. Diametrically opposed to this are knowledge-intensive retrieval methods inspired by work in artificial intelligence. Knowledge-intensive methods attempt to construct comprehensive representations of the text and the query in order to infer relationships between them.

Both methods have advantages and disadvantages. By ignoring the need for knowledge, keyword-based retrieval systems achieve a high level of applicability over different domains and linguistic styles. AI-based retrieval systems offer "intelligent" retrieval, but that is limited to domains and text styles for which the system has enough knowledge to process the text into sophisticated representations. The wide acceptance of keyword-based systems suggests that universality is the greater good and lack of domain/world knowledge the lesser evil.

This thesis advances an approach, the case structure representation approach, that attempts to overcome the knowledge bottleneck that AI-based retrieval systems face. In this approach, knowledge from standard semantic knowledge sources is applied to structured representations extracted from the queries and the text. The knowledge that this approach brings to bear is not domain-dependent but application-dependent (structure-dependent). This knowledge enables the approach to extract structured representations and decide how best to apply semantic knowledge. The use of structured representations also addresses the problems that arise during attempts to augment keyword-based systems with semantic knowledge.

Right now, the lack of robust and domain-independent natural language processing tools limits the applicability of the case structure representation approach to short and structurally predictable text. The two applications that are described in the next two chapters, ImEngine and NetSerf, assume that the input text consists of single sentences. The success of these two systems indicates that the limitation is far from fatal. As progress is made in the development of NLP tools, it should be possible to extend the approach to handle more complex text.

CHAPTER 3

ImEngine

This chapter is a description of ImEngine, the first application of the case structure representation framework described in the previous chapter. ImEngine is a program for retrieving pictures and video clips through caption analysis. The retrieval of pictures and video is attracting quite a bit of attention now because of its great commercial potential. On the one hand, there are huge libraries of annotated pictures and video owned by media and stock footage companies, and by institutions such as the Smithsonian and the Library of Congress. When these libraries are digitized, they will likely be made available over the Internet to users of multimedia authoring or browsing tools. Examples of other sources of pictures and video clips are medical imagery, satellite photographs, maps, architectural and engineering blueprints, and home video and pictures.

At first glance, the automatic retrieval of pictures and video clips does not seem to have much in common with text retrieval. But the link between the two becomes apparent when we consider the fact that currently there are no computational methods that can "understand" pictures or video, i.e., produce high-level descriptions of who, where, when, or what the picture or video is about. Instead, if retrieval is the goal, such descriptions have to be provided manually for pictures and video, as captions for example. Once this is done, there is significant overlap between text retrieval and picture or video retrieval. However, even though there are no computational methods for extracting high-level descriptions, there are robust methods for inferring properties like color, texture, brightness, etc. Since such properties are often not made explicit in captions, the two approaches seem to be complementary. Section 3.1 is a more detailed description of work in this area.

From Section 3.2 on, we turn to the description of ImEngine itself. First, Section 3.2 describes the database of pictures and video that has been used to put ImEngine through its paces, and the caption processor which turns captions into case structure representations. The queries used to test ImEngine were collected through an experiment in which subjects were shown pictures and video clips with captions, and subsequently asked to describe the pictures and video clips in their own words. This experiment is described in Section 3.3. Then, Section 3.4 is a description of the mechanism used to match representations of captions to representations of queries. Section 3.5 presents an evaluation of ImEngine's performance, and compares it to that of SMART [Salton 1989], a standard keyword-based retrieval system. Finally, Section 3.6 explores some of the problems and opportunities that ImEngine presents.

3.1 Approaches to Image Retrieval

This section gives an overview of a variety of image retrieval methods thereby setting the stage for the description of ImEngine from the next section. Many methods are specialized for particular image formats, e.g., photogrammetry techniques for retrieval of remote sensing satellite images [Horn 1986]. Similarly, there are specialized techniques for maps, images of human faces [Turk and Pentland 1991], X-ray images and other medical images. However, the focus of this section is on "general" pictures, i.e., pictures that are mediated representations of scenes or actions.

There are many different ways by which users can retrieve images. We can try to retrieve images based on their physical attributes, e.g., images that have a blue background or images that have a predominantly brick-like texture. On the other hand, we can retrieve images based on features such as objects or actions or time of day, e.g., pictures of Clinton shaking hands. Of course, it possible to combine the two kinds, e.g., a picture of "Clinton shaking hands against a blue background."

Ideally, a retrieval system ought to be able to extract all of an image's features using automatic image processing methods. Unfortunately, the state of the art in image processing and machine vision does not permit that. Fairly robust methods have been developed for extracting the physical features of an image such as color, texture and brightness. But there has not been much progress in the development of robust, domain-independent methods for extracting "high-level" descriptions of images, i.e., information about the who, what, when and where of the image.

Therefore, we can broadly classify existing work in image retrieval into the categories of *physically-based retrieval*, where physical features such as color and texture provide the vocabulary for queries, and *description-based retrieval*, where "higher-level" descriptions of objects, situations and actions are used for retrieval.

Physically-based retrieval methods are usually automatic and domain-independent. QBIC [Niblack et al. 1993] is a good example of this genre[1]. QBIC enables images to be queried using color, texture and shape. Another comprehensive system that provides multiple features is Photobook [Pentland et al. 1994]. [Gudivada and Raghavan 1994] is a detailed survey of picture retrieval systems covering both domain-independent, physically-based methods as well as specialized retrieval techniques.

But there are no completely automatic algorithms for domain-independent description-based retrieval. Instead, the work in this area can be classified into the following types:

- Automatic retrieval limited to a few content domains
- A combination of automatic and manual methods, or purely manual methods for encoding descriptions of images
- Use of auxiliary information such as captions

When retrieval methods are limited to one or a few domains, domain-specific assumptions can be built into them, e.g., [Rabitti and Savino 1991, Bobick 1993]. For example, the system described by [Bobick 1993] has specific knowledge about football plays. The downside is that these

1. It is now marketed by IBM as a commercial product called Ultimedia.

assumptions make it hard to extend the system to other domains, i.e., to "scale it up." Still, it is reasonable to hope that some of the techniques and lessons learnt from domain-specific systems will transfer to more general systems.

A second approach is to use manual annotation to describe the images or video, either in tandem with some automatic extraction or entirely on its own. The Media Streams system built by Marc Davis [Davis 1993] is a good example of the former. Media Streams is an iconic, stream-based video logging and retrieval system. In this system, the annotator (or editor) can provide annotations for any segment of the video stream. These annotations are used to indicate the actors, actions, locations and other items of interest. The system uses automatic techniques such as scene break detection to assist the annotator. On the other hand, purely manual methods have been in use by large archives (like news organizations) for many years. Here, keywords or "keyphrases" are used to indicate the salient features of the images as well as the context from which the image arises. For instance, a description of the famous photograph of Clinton in the background as Rabin and Arafat shake hands would indicate that it was part of the Israeli-PLO peace accord.

The third approach is to use auxiliary information that is sometimes provided along with pictures. For example, news photographs and images in scientific articles are often accompanied by captions because they do not "speak for themselves." This could be considered as a purely manual method of annotation except that what is created is not a description in a format that is convenient for computational processing, but in an intermediate format (natural language) that is convenient for the annotators. [Rowe

1994] describes a research effort aimed at understanding the relationship between pictures and captions. Rowe's work tackles the problem that what is salient in a caption is not necessarily so in the picture and vice versa. [Srihari 1994] describes an application of caption processing to aid in object recognition.

Clearly, physically-based and description-based retrieval are complementary, not competing, approaches. At their best, physically-based approaches facilitate very fine visual detail regarding color, texture, shape, spatial postioning, etc. in querying images. Description-based approaches permit images to be retrieved on the basis of who, what, when, and where, as well as other related information. The rest of this chapter describes ImEngine, which is a description-based approach relying on caption analysis.

3.2 Caption Structure and Processing

Captions can be related to the pictures they accompany in many different ways. Often, a caption is a description of the visual content conveyed by the image or video clip. But that is not necessarily the case. Captions of figures in scientific articles need not be simple descriptions of the figures. Sometimes, when the picture or video clip is considered self-explanatory, the caption is used to provide background information or additional information. Captions can even be used as outlets for humor by clever interplay with the pictures. Here are two examples from the Economist magazine, one about the reluctance of the British government to privatize British Rail, and another about allegations that former Italian Prime Minister Silvio Berlusconi was using his office for personal gains.

FIGURE 3.1. Photographs from the Economist

But the government doesn't

All the way to the bank?

3.2.1 ImEngine's Caption Processor

ImEngine is designed to handle captions of the descriptive kind. It has been tested on captions from two sources, the Associated Press which provides captions with its news photographs, and ImageBank, which is a CD-ROM of stock footage video clips.

ImEngine's caption processor assumes that the caption is a single sentence that describes one or more actions or physical situations; i.e., it does not handle sentences like "Can fish swim?" or "Pat knows the truth." In addition, as discussed in Chapter 1, the structure of the captions need to be predictable. The caption processor assumes that the syntax of the input string is in accord with the following grammar:

Caption	$\rightarrow$	NounPhrase ActionVerb Complement
Caption	$\rightarrow$	ActionVerb Complement

Caption	→	NounPhrase Complement
Complement	→	"as" Caption
Complement	→	"while" Caption
Complement	→	"to" InfinitiveVerb Complement
Complement	→	PrepositionalPhrase Complement
Complement	→	PrepositionalPhrase

The caption processor uses this grammar to extract a case structure representation from the caption. We will illustrate this by means of the following example (from the ImageBank collection).

FIGURE 3.2. Doctor positioning microscope for microsurgery in an operating room.

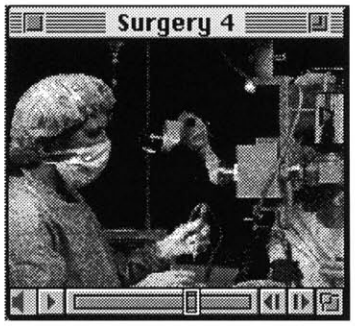
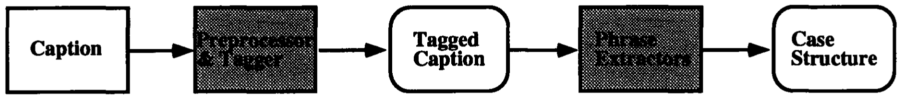


Figure 3.2 shows one frame of a video clip from the ImageBank collection, and the caption used to describe the clip. To extract the representation of this caption, the caption processor assigns roles to various words in the caption, as in case grammar [Fillmore 1968]. We will refer to <role, word> pairs as *cases*. Here is the case structure representation (case representation) of the caption of Figure 3.2:

AGENT:	doctor
ACTION:	position

PATIENT:	microscope
FOR:	microsurgery
IN:	operating room

FIGURE 3.3. Caption processor



The caption processor generates the case representation in three steps as shown in Figure 3.3:

- First, a preprocessor identifies special groups of words like dates, place names, etc., that belong together. It also removes information about camera parameters from captions, e.g., “shots of” from “Shots of a man in a recording studio.”
- Second, a probabilistic part-of-speech tagger, the Xerox PARC tagger, assigns parts of speech to words in the preprocessed caption by weighing their context of occurence [Cutting et al. 1992]. For example, in the caption of Figure 3.2, the word “doctor” gets classified by the tagger as a noun (though it can act as a verb in other contexts).
- Third, the caption processor uses noun phrase and prepositional phrase extractors[1] to parse the tagged caption according to the grammar shown above. This step generates the final case representation. Special rules are included to generate roles such as Agent, Action and Patient. The head preposition of a prepositional phrase acts as the role for the head noun of the phrase.

1. These are the same extractors used by the dictionary processing program described in Section 2.3.2.

From ImEngine's viewpoint, the structure of queries should be the same as that of captions, i.e., in response to a query, ImEngine chooses images (if any) whose captions are conceptually equivalent to the query. In the next section, we will discuss how queries were collected for a subset of the database of Associated Press photographs and ImageBank video clips.

3.3 Query Collection and Processing

This section describes an experiment conducted to collect queries for ImEngine's evaluation. It was found necessary to conduct the experiment because there is, as yet, no standard test collection of pictures and video clips with captions and benchmark queries and answers (unlike CACM, TIME, TREC and other collections for full-text retrieval, e.g., [Harman 1993]). The aim of the experiment is to collect queries that are conceptually, but not literally, equivalent to the captions, because such queries would test ImEngine's abilities fully. The experiment relies on psychological research that indicates that human subjects tend to remember facts conceptually and not literally [Klatzky 1979].

For the experiment, a set of 30 Associated Press news photographs and 20 ImageBank video clips was used. 10 subjects took part in the experiment, which was conducted in two stages as follows:

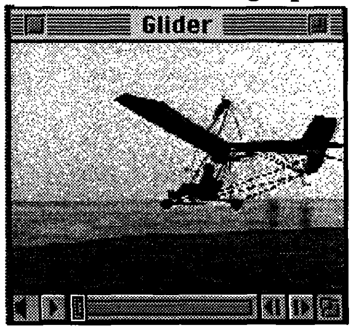
- The 50 pictures and video clips, with captions, were presented in random order to each subject. Each subject was allowed to take as much time as necessary to see the pictures and video clips, and read the captions. Each subject took on average 20 minutes.

- The second stage of the experiment was conducted one day later for five subjects and three days later for the other five. Subjects were asked to describe any pictures or video clips that they could recall, and their descriptions were recorded. Then, each subject was again shown the pictures and video clips, but this time without the captions. For each picture or video clip that the subject could now recognize but had been unable to recall earlier, we recorded the subject's description regardless of accuracy.

The set of recorded descriptions was turned into the query collection by removing duplicate descriptions, incorrect descriptions, and descriptions that were identical to the corresponding caption. This process resulted in 153 queries being collected from the subjects who took part in the second stage of the experiment one day later, and 127 queries from the other subjects (280 in all). More queries were collected from the one-day subjects because duplicates were always removed from the three-day descriptions, and probably because subjects remembered captions more accurately after a one-day, rather than a three-day, interval. The captions used in the experiment and the queries we collected can be found at the URL **http://www.media.mit.edu/people/anil/ImEngine/ImEngine.html**.

This experiment relies on the assumption that subjects remember captions not literally but in a conceptual form. Experimental memory research in cognitive psychology supports this view [Klatzky 1979]. Therefore, when, in the second stage of our experiment, subjects were asked to describe the picture or video clip, they did so in their own words. This yielded a rich collection of semantically equivalent queries after we removed incorrect descriptions and descriptions that were the same as the corresponding captions. Shown below are a caption and 9 other descriptions of the video clip provided by the subjects.

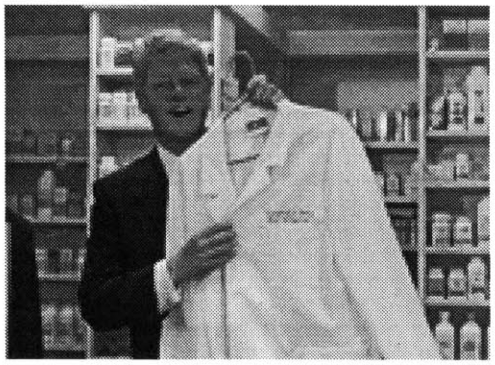
FIGURE 3.4. Ultra-light plane flying low over beach and ocean.



1. Light airplane flying low over the ground on the beach
2. A man flying small plane over beach and ocean
3. Light plane flying across a beach
4. A man flies a man-made flying contraption on a beach
5. Man flying ultralight craft over the sea
6. Ultralight pilot flies low over beach and sea
7. Very light airplane flying low over water
8. Glider going over water
9. Foot-powered glider at the beach

In choosing the pictures and video clips for the experiment, we took into account the phenomenon of memory interference in human subjects [Klatzky 1979]. If subjects are shown a set of pictures and video clips in which each one is not distinctly different from the rest, during recall they are likely to conflate the description of a picture or clip with descriptions of similar pictures and clips. Therefore, we had to choose a set where each of the pictures and video clips were widely different from the others. For example, the photograph of Figure 3.5 is the only one involving "President Clinton." As will be discussed in Section 3.5, this is an important factor to take into account in judging the performance of keyword-based systems on this set.

FIGURE 3.5. President Bill Clinton holds up a pharmacist's coat



Queries are processed in the same way as captions to extract query representations. As made clear by the preceding discussion, queries are simply alternative descriptions of the pictures or video clips. Hence, they obey the linguistic assumptions made by the caption processor, viz., that its input should be a single sentence with a structure corresponding to the context-free grammar given in Section 3.2.

The set of 280 queries was divided into two subsets, one for training (150 queries) and the other for testing and evaluation (130 queries). The training subset was used to develop and refine the query processor as well as the matching framework. To test the accuracy of the query processor, the case structure representations generated for these queries were checked manually. This was done by taking the ratio of the number of correct <role, word> pairs to the total number of pairs. On the training subset, the accuracy was 87.56% (570 correct pairs out of 651). On the testing subset, the figure was 87.48% (454 correct pairs out of 519). The query processor therefore turned out to be quite reliable.

The case structure representation of the query guides the use of semantic knowledge in matching queries to captions. For example, in Figure 3.2, the location of the action is an "operating room." From WordNet, we know that an "operating room" is a part of a "hospital." Putting the two together using a matching rule, ImEngine infers that the location of the action is a "hospital." The next section describes rules used for matching query representations to caption representations.

3.4 Matching Queries to Captions

To match a query to a database of captions, ImEngine first matches the query representation to each caption representation in the database and computes a weight for the resulting match. The matches are then sorted in order of decreasing weight to produce a list of ranked captions. In this section, we will first outline ImEngine's framework for matching a query representation to a caption representation, and then present several matching rules that are used within this framework.

The matching rules that can be applied to a query case depend on its role. Every matching rule has an associated *success weight* that reflects

ImEngine's "confidence" in the rule. The net weight for a query representation, *QR*, when matched to a caption representation, *CR*, is computed as follows:

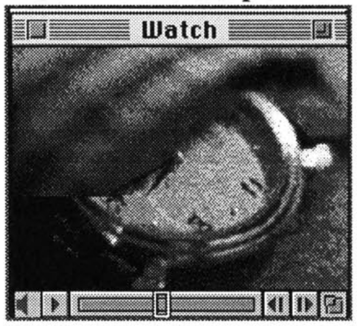
- For each query case, *QC*, in *QR*, the matching rules corresponding to its role are applied in decreasing order of success weight until some rule succeeds, i.e., the rules are applied in the order that maximizes the weight contributed by *QC* to the total weight. If no rule succeeds, *QC* contributes zero to the net weight.
- The total weight for *QR* is calculated by summing the individual weights contributed by its query cases, and by adding additional weights for structural correspondence between *QR* and *CR*. For instance, if adjacent agent and action cases in *QR* match successfully to adjacent agent and action cases in *CR*, an additional weight is added to the total weight of *QR*.

We now turn to the descriptions of the matching rules. The rules with the largest success weights are those that capture the principle of *generalization* that was discussed in the introduction to ImEngine in Chapter 1. There, using the semantic relation A-KIND-OF (hypernymy), we generalized the caption word "basset hound" to the query word "dog." Typically, generalization applies when a query word is a hypernym or a synonym of a caption word playing the same role. In Figure 3.4, we see a video clip and its caption, "Ultra-light plane flying low over beach and ocean." When we match the query, "Glider going over water," to this caption, generalization results in matching the action word "fly" to "go," and the patient word "ocean" to "water." But, generalization is not limited to hypernyms. For roles such as location, generalization takes place through the PART-OF relation (as from "operating room" to "hospital" in the example of Figure 3.2).

In many cases, a query word can be semantically closely related to a caption word even though it may not be possible to generalize the latter to the former. This intuition is captured by a rule that looks for a common hypernym for the two words. For instance, in Figure 3.4, the query agent word "glider" and the caption agent word "plane" have a common hypernym "aircraft." The query and caption words are considered to be conceptually close if the sum of their *distances* (number of intervening links) to the common hypernym is less than a preset threshold. If so, ImEngine considers this a successful match and gives it half the weight of a generalization match.

For matching action words, in addition to rules that use hypernyms, ImEngine has other rules that use the relations ENTAILS and CAUSES provided by WordNet. In WordNet, a verb *V* entails a verb *V'* if *V'* takes place whenever *V* takes place, e.g., the verb "watch" entails the verb "see." Therefore, it makes sense for ImEngine to match a query action word to a caption word that entails it. In using the CAUSES relation, ImEngine takes into account the agent and the patient words of the query and the caption. Figure 3.6 shows an example. The query "Sleeve comes up" matches this caption because the verb "lift" CAUSES "come up," and the patient word of the caption, "sleeve," is the same as the agent word of the query, indicating that the same action is being viewed from a second perspective.

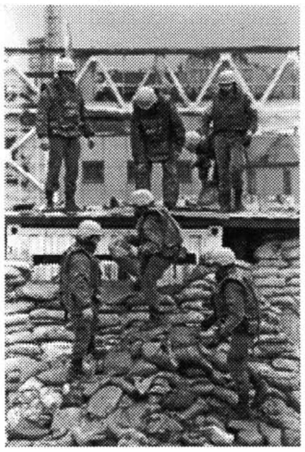
FIGURE 3.6. Closeup of man lifting sleeve to reveal watch.



Some action verbs are typically associated with certain objects, and vice versa, e.g., the verb "sandbag" is related to the noun "fortification"[1] in the Webster's dictionary semantic network (Section 2.3.2). These relations are used to match action words to object or purpose words. For example, consider the query "Soldiers putting up fortifications." Its patient word "fortification" matches the action word "sandbag" in the caption of Figure 3.7. Unlike the rules described earlier that only use canonical relationships, this rule makes assumptions about the typical co-occurence of certain objects (which may not always hold). Therefore, this rule is given less weight than the preceding rules.

1. From the verb definition "**sandbag**: to hit or stun with a sandbag" and the noun definition "**sandbag**: a bag filled with sand and used in fortifications, as ballast, or as a weapon"

FIGURE 3.7. French U.N. troops sandbag their positions in Sarajevo, Friday, March 4, 1994.



ImEngine's last matching rule uses WordNet to find common modifiers between query words and corresponding caption words. For instance, the adjective "American" in the phrase "American soldiers" matches the modifier "U.S." in the phrase "U.S. troops," because WordNet indicates that the adjective "American" pertains to "U.S." When common modifiers are found between the query and caption words of a match, ImEngine increases the weight of the match.

Now that we have discussed the process of extracting caption and query representations and the process of matching the representations using application-specific heuristic rules, we are ready to consider how well ImEngine works on the query collection. That is the subject of the following section.

3.5 ImEngine Performance Evaluation

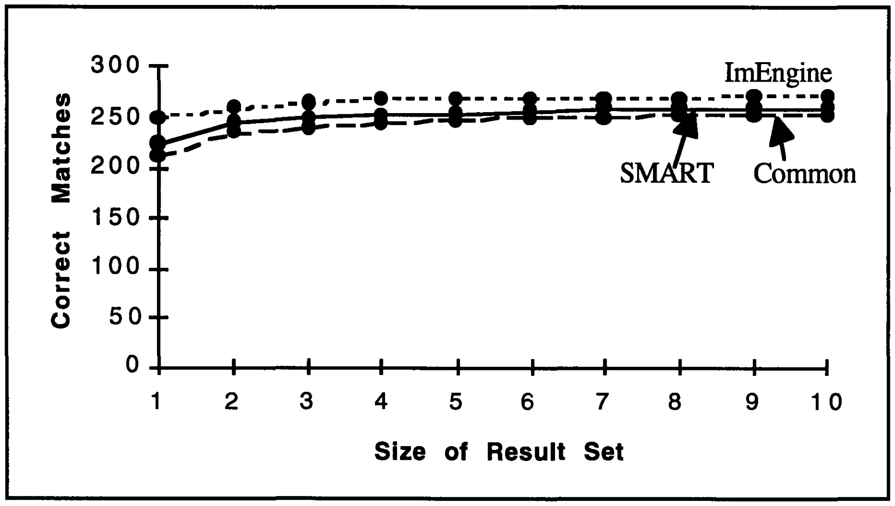
This section presents results from the evaluation of ImEngine on the set of 280 queries obtained from the experiment. As a benchmark, we will evaluate the performance of a well-known text retrieval system named SMART [Salton 1989] on the same set of queries. SMART is a sophisticated keyword-based system that retrieves documents from queries, based not only on keywords they have in common, but also the relative frequency of occurence of the common keywords in the document collection. Finally, we will look at the performance of two versions of ImEngine that do not use semantic knowledge and structured representations respectively, in order to judge the influence of these components.

The strategy used to evaluate ImEngine and SMART is as follows. Each query is derived from a particular picture or video clip (because it was some subject's description of that picture or video clip). We consider that picture or video clip as the only correct answer for the query. Also, for each query, both systems return a list of ranked matches (pictures/video clips). As a measure of each system's success rate, we counted the number of queries for which the correct answer was found in the first *n* top-ranked hits returned by the system. Figure 3.8 shows the results for *n* from 1 to 10. For example, when we consider only the top-ranked hit, ImEngine is successful at matching 250 queries (out of 280), while SMART matches 223. When the top 10 hits are considered, ImEngine matches 270 queries, while SMART matches 259.

In the query collection experiment, subjects were asked only to provide alternative descriptions of the pictures and video clips. They were not asked to rank the descriptions they provided against the entire collection, since that would have been cumbersome. Hence, the evaluation is presented in terms of the number of successful queries, rather than precision and recall, which are the more standard measures.

Since ImEngine and SMART use very different retrieval techniques, we were interested in finding out whether they were complementary, i.e., successful on different subsets of queries in the collection. The line titled "Common" in Figure 3.8 shows the number of queries that both systems were successful on, indicating that ImEngine was successful on almost all the queries on which SMART was successful.

FIGURE 3.8. Performance of ImEngine and SMART

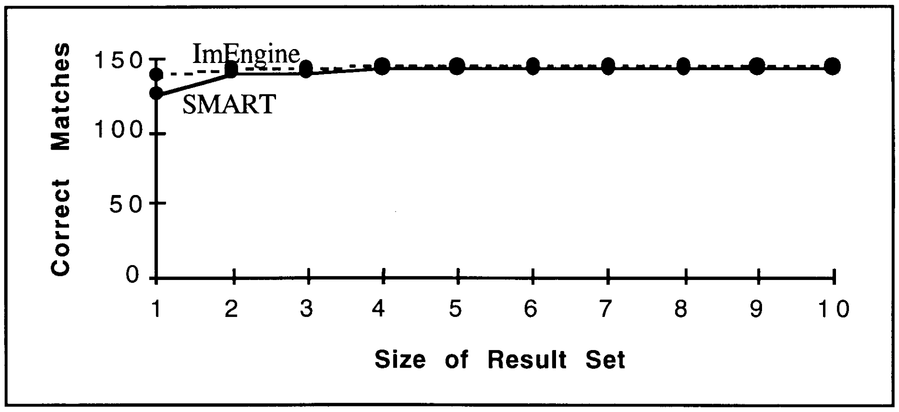


SMART's performance seems to have been helped by the fact that the captions selected for the query collection experiment were widely different from each other. For instance, there is only one picture involving President Clinton in the database. Therefore, SMART is able to retrieve the correct answer for any query that has the word “Clinton.” If the database were to have several captions containing “Clinton,” structured representations would become important in retrieving only the correct answers.

The two examples below give an idea of how ImEngine and SMART differ:

- ImEngine succeeds in matching the query “Glider going over water” to “Ultra-light plane flying over beach and ocean,” but SMART cannot get this one because of the absence of literal matches.
- SMART succeeds in matching the query “Women jumping in a basketball match” to the caption “Connecticut's Kara Wolters (52) pulls down a rebound over the outstretched arms of Seton Hall's Shamona Marable (35) during the first period of the Big East womens basketball tournament championship game at Gampel Pavilion in Storrs Conn., Monday, March 7, 1994” simply based on the common keyword “basketball.” ImEngine misses this as the top hit because the actions cannot be related using the semantic knowledge sources, and the match between the agent words is very weak.

FIGURE 3.9. Performance of ImEngine and SMART on training set



ImEngine was evaluated separately on the training and testing subsets of the 280 queries. Figure 3.9 and Figure 3.10 show these results. As expected, ImEngine performed better on the training subset (accuracy rate in the range of 92.6% to 97.3%). On the testing set, the accuracy rates ranged between 85.4% and 95.4%. ImEngine performed better than SMART on both of these subsets.

FIGURE 3.10. Performance of ImEngine and SMART on testing set

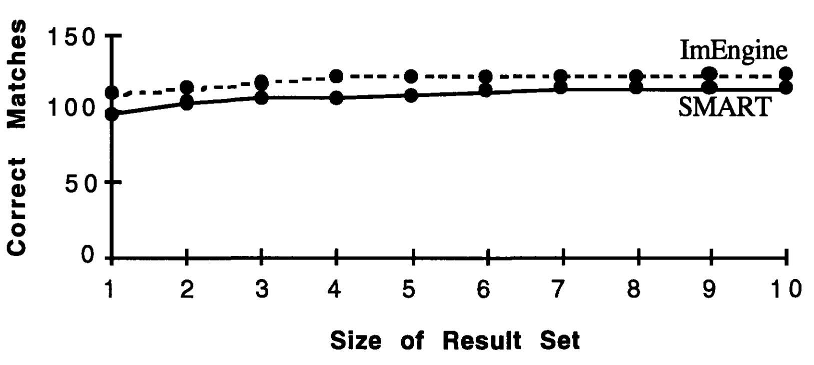
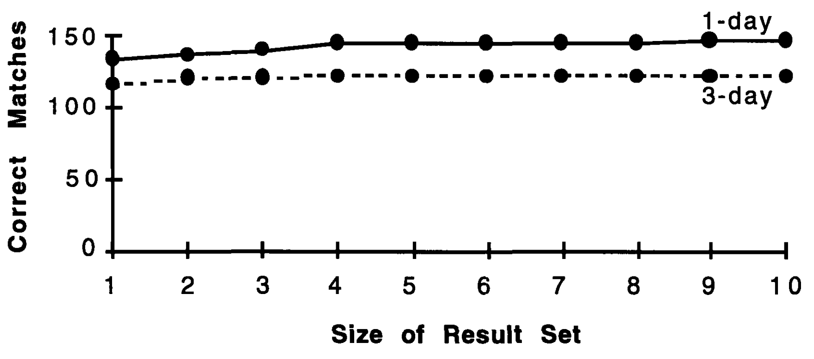


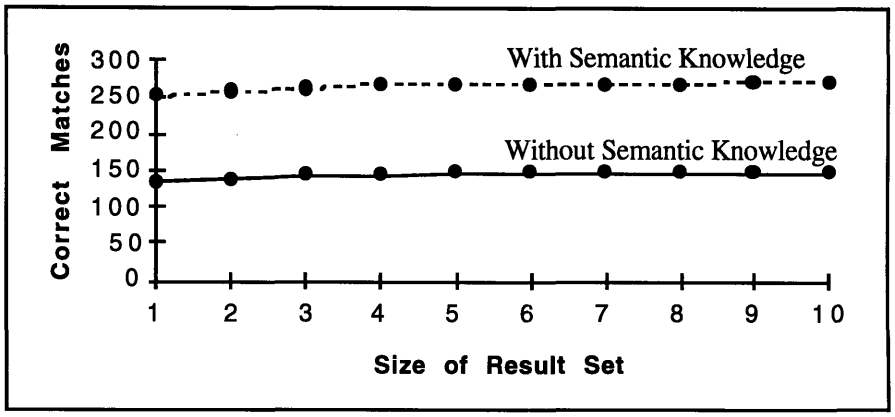
FIGURE 3.11. Performance of ImEngine on one-day and three-day queries



As described in the previous section, the set of 280 queries consists of 153 queries recalled after one day, and 127 queries recalled after 3 days. ImEngine's performance varied only slightly on these two subsets (Figure 3.9). On the 153 one-day queries, ImEngine's success rate ranged from 87.6% to 96.1% for *n* from 1 to 10, while on the 127 three-day queries, the range was 91.3% to 96.8%. The better performance of the 3-day subset is due to the predominance of 3-day queries in the training set.

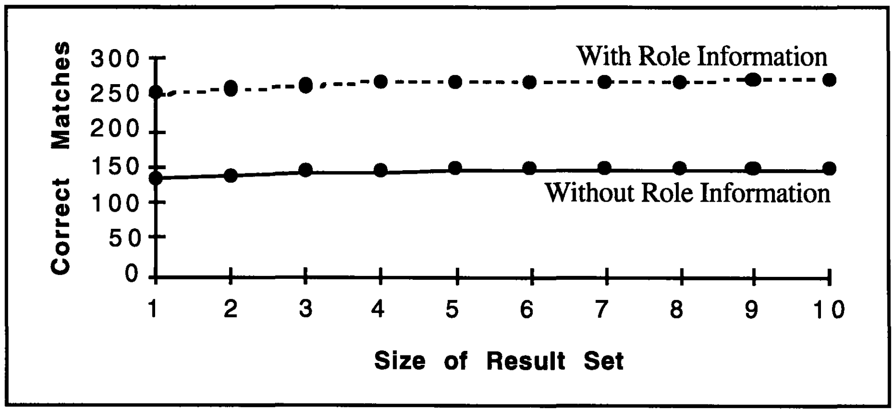
Next, to test the use of semantic knowledge, we built a version of ImEngine which did not use semantic knowledge for matching, but did use structured representations of the captions and queries, i.e., in this version, only keywords and role information were used for matching. Figure 3.10 presents the results of running this version of ImEngine, showing that the use of semantic knowledge leads to very significant improvements. Also, in the regular version, we counted the number of word matches in ImEngine that were based either on semantic relations from WordNet or on semantic relations extracted from the dictionary. When the top-ranked hits for each query are considered, there were 430 semantic matches, i.e., about 1.5 semantic matches for every query. The high number of semantic matches indicates that ImEngine is able to generalize the caption to the query, while maintaining precision by using role information.

FIGURE 3.12. Performance of ImEngine with and without semantic knowledge



Finally, to judge the influence of structured representations on ImEngine's performance, we tested another version of ImEngine which used semantic knowledge for matching, but ignored role information entirely. In this version, all the matching rules of Section 3.4 were applied to all the cases in the query representation. As Figure 3.13 shows, using role information leads to a marked improvement in ImEngine's performance.

FIGURE 3.13. Performance of ImEngine with and without role information



In summary, the performance evaluation of ImEngine shows that on the set of 280 queries collected from subjects, ImEngine achieves better results than SMART. The two main components of ImEngine, structured representations and semantic knowledge, both contribute significantly to ImEngine's success. Suspending the use of either component leads to clear drops in performance. It should be noted that the results presented in this chapter were derived without using the disambiguator. The use of the disambiguator in ImEngine will be described in Section 5.3.

3.6 Problems and Opportunities

As a prototype, ImEngine has served its purpose in evaluating the case structure approach. On the test set of 280 queries collected from the 10 subjects, this prototype has shown significant improvements over the standard vector-space indexing system SMART. However, the prototype can be improved along the following dimensions:

- Caption Processing
- Representation
- Matching

The caption processor ought to include algorithms for pronoun resolution and prepositional phrase attachment (PP attachment). Unfortunately, both are currently unsolved problems in computational linguistics. Pronoun resolution is the process of deciding what a pronoun refers to, e.g., what "it" refers to in the sentence "I saw the cat with the ball before it moved." This is a difficult problem because extensive world knowledge is needed to resolve the ambiguous references. PP attachment refers to the process of deciding which word is modified by a prepositional phrase. For example, in the sentence, "I saw a man with a telescope," does the prepositional phrase modify the action or does it modify the direct object? Again the problem is one of ambiguity. In captions, pronoun ambiguity is not as pervasive a problem as PP attachment ambiguity primarily because pronouns tend to occur over multiple sentences. The problem of disambiguation is taken up again in Chapter 5. Research in computational linguistics indicates that fairly robust, domain-independent algorithms for pronoun resolution and PP attachment might also result from use of semantic knowledge from dictionaries, e.g. [Ravin 1990].

The case structure representation used by ImEngine, which is currently a flat list of <role, word> pairs, can be improved in three ways:

- First, case representations can be made more precise by attaching prepositional phrases to their antecedents. This would result in a nested case structure. As mentioned above, the current version does not attempt prepositional phrase attachment.
- Second, captions of video clips should be organized into temporal representations. Some video clip captions have multiple segments which reflect the temporal structure of the clip. Currently, the roles extracted from all the segments are combined into a single structured representation. Future versions of the caption processor will need temporal representations to index such captions.
- Third, the caption processor should incorporate the "logic" underlying the relationship between captions and pictures [Rowe 1994]. These rules establish visual salience for keywords in a caption. For example, given

the caption "Photovoltaic cell panels for generating power to ultimately operate a radar," the system should be able to infer that cell panels are shown in the picture, but a radar is not. While these are not inviolable rules, they hold often enough that it is worth using them in the absence of robust object recognition methods.

The heart of ImEngine is the matching mechanism. As improvements are made in the caption processing and the representation, the matching mechanism will need changes as well. Here are a few possibilities:

- New matching rules will be required as we get access to new sources of semantic knowledge and as new examples arise. New sources may provide semantic relations that are not used by the existing matching rules. Therefore, to make use of these new relations, new matching rules would be needed.
- In ImEngine now, the success weights of all the rules are hand-coded. As the number of rules increase, success weights should probably be acquired and altered automatically (Section 6.1).
- Since we use WordNet to judge how closely words are related, we need to develop a good measure of semantic distance that corresponds to human intuitions about related words. For example, ImEngine simply counts the number of intervening links between synsets to get their semantic distance, but this measure fails to capture the close relation between the words "crowd" and "people," whose synsets are four links apart in WordNet. A better approach would be to take into account the number of hyponyms (direct and indirect) of a node [Norvig 1994].

- The success weights associated with matching rules take into account only the type of the match, not the words themselves. Analogous to term-weighting indexing systems like SMART, ImEngine needs to be able to judge which words are characteristic of a caption or query, and hence deserve additional weight. This will become particularly important as we try out ImEngine on larger databases.

Lastly, ImEngine's performance has been evaluated on queries provided by subjects who had previously seen the pictures and video clips in the database. These queries may not reflect the queries that would be used in exploratory tasks. ImEngine needs to be evaluated on a collection of such queries as well. This would be made possible by integrating ImEngine with a presentation or authoring tool that is used to conduct exploratory searches.

CHAPTER 4

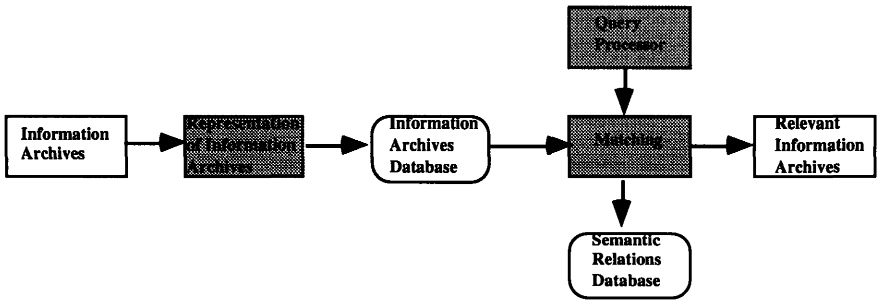
NetSerf

This chapter describes the architecture, implementation and evaluation of NetSerf, the second application of the case structure representation framework described in Chapter 2. NetSerf is a program for finding information archives on the Internet using natural language queries. In Chapter 1, a distinction was drawn between information access and information retrieval. Information access is the task of finding archives that might potentially contain information of interest to the user. Information retrieval is the subsequent task of searching for desired information within a given archive. In this parlance, NetSerf is an access program whereas ImEngine is a retrieval program.

Since a wide variety of news and other information is rapidly becoming available on the Internet, information access has turned into an important problem. Section 4.1 describes some previous approaches to the problem of finding information on the Internet.

From Section 4.2, this chapter will concentrate on NetSerf. The architecture of NetSerf is shown in Figure 4.1 (reproduced from Chapter 1). As described in Chapter 2, the semantic knowledge database is a combination of semantic relations from WordNet [Miller 1990], and semantic relations extracted automatically from an on-line Webster's dictionary. Descriptions of information archives are stored in the form of structured, frame-like representations [Minsky 1975]. The query processor turns natural language queries into structured, disambiguated representations, which are then matched to the archive descriptions using the semantic knowledge database.

FIGURE 4.1. Architecture of NetSerf[1].


Query Processor
Information Archives
Representation of Information Archives
Information Archives Database
Matching
Relevant Information Archives
Semantic Relations Database

NetSerf has been evaluated on a set of queries collected from the game *Internet Hunt* [Gates 1992]. Each month, the creator of this game publishes ten questions, which participants are expected to answer using only information available on the Internet. The correct answers are published the following month. Thus, this game provides an independent set of queries with answers for testing NetSerf. Using this query set, the performance of Net-

1. Regular boxes denote external data, shaded boxes denote processes, and rounded boxes denote internal representations.

Serf has been compared to that of the well-known information retrieval system SMART [Salton 1989].

The organization of the chapter is as follows: Section 4.2 describes the process of creating representations of information archives by hand.
Section 4.3 deals with the query processor and the Internet Hunt questions that are used to evaluate NetSerf. Then, Section 4.4 describes the mechanism used to match query representations to archive representations, and to rank hits. In Section 4.5, the results of running NetSerf and SMART on the set of Internet Hunt queries are presented. Section 4.6 concludes the chapter with a discussion of the problems and opportunities for future work offered by NetSerf.

4.1 Tools for Retrieval of Internet Information

Existing tools for finding information on the Internet can be classified on the basis of their underlying protocols for information organization and transfer [Schwartz et al. 1992]. This section considers five such protocols:

- Directory services
- File transfer protocol (FTP)
- WAIS [Kahle & Medlar 1991]
- Gopher [McCahill 1992]
- World Wide Web [Berners-Lee et al. 1992]

Directory services enable users to find address/location/profession information for people connected to the Internet. Some examples of such services are WHOIS [Harrenstein et al. 1985], X.500 [CCITT/ISO 1988] and NetFind [Schwartz & Tsirigotis 1991]. WHOIS and X.500 require potential addressees to register their whereabouts into a database. User queries are matched against this address database. NetFind operates on a different model. Using clues provided by the users, the UNIX command "finger" is used to check appropriate domain servers. The clues provided could be place names or domain names or type of organization, etc (e.g., "search schwartz colorado edu").

FTP is the standard protocol for transferring files on the Internet. FTP allows files to be made available publicly using a method known as anonymous FTP. Archie [Emtage & Deutsch 1992] is a tool that enables users to search on filenames of anonymous FTP sites worldwide. Archie provides string search and regular expression search.

WAIS [Kahle & Medlar 1991] allows users to deploy, search and retrieve documents and other types of information from indexed databases (servers) throughout the Internet. The databases are indexed using full-text indexing software based on vector-space indexing methods (the software is provided by WAIS). WAIS supports natural language queries and relevance feedback. At the time of retrieval, WAIS expects the user to indicate the databases that should be searched. This is usually done by searching first on a directory of servers maintained by WAIS. In this sense, WAIS subscribes to the two-step information access and retrieval model.

On the Internet, there has long been a tension between the tendency to organize the vast amount of available information and the tendency to let things be. As Rick Gates, the creator of the Internet Hunt, puts it:

*With anything as large, dynamic and unique as the Internet, there are, as one might guess, many different theories about how to get started. These seem to fall into a couple of major camps. The *extremes* of these two camps are:*

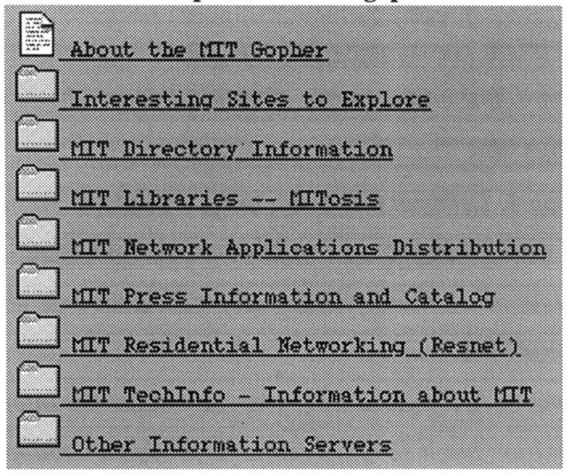
Camp 1: Let's get this damn thing organized. We want a set of indexes to the whole thing so we can find out what we want, when we want, once and for all. Nobody wants to go looking all over creation! You learn only by direction.

Camp 2: Are you kidding, you can't index the Internet. It's too huge! Not only that, but it's too dynamic. Anybody can add to it. Don't try fencing it in. You'll just kill it. What you need to do is jump in with both feet, roll up your sleeves and get your hands dirty. You learn only by doing.

Gopher is one of the more successful attempts to construct hierarchical indexes from Internet information. A Gopher site typically organizes its information in a directory (menu), and provides pointers to other Gopher sites. Figure 4.2 shows the "top level" of **gopher://gopher.mit.edu**

Gopher has proved attractive because it provides a layer of abstraction in browsing and locating information from a number of different sources. In addition, tools such as Veronica [Veronica 1993] maintain an index of titles of all known Gopher menus (the *Gopherspace*), thereby enabling keyword searches of all of Gopherspace.

FIGURE 4.2. Top level of MIT gopher site


About the MIT Gopher
Interesting Sites to Explore
MIT Directory Information
MIT Libraries -- MITosis
MIT Network Applications Distribution
MIT Press Information and Catalog
MIT Residential Networking (Resnet)
MIT TechInfo - Information about MIT
Other Information Servers

The latest and fastest-growing Internet information protocol is the World Wide Web (WWW) [Berners-Lee et al. 1992]. WWW documents have unique identifiers called Uniform/Universal Resource Locators (URLs), and are created in a special markup language called HTML (HyperText Markup Language). HTML allows links to be placed between any two WWW documents, which allows information to be organized and accessed in a hypertext fashion without regard to physical location. A number of keyword-based retrieval tools have been built for the World Wide Web, with entertaining names like worms, spiders, crawlers and robots[1]. These tools include modules for automatically traversing the Web looking for new documents. Documents are typically indexed using the headings, links and the keywords found in the text.

Now that some of the existing Internet information retrieval tools have been described, it is time to address two basic questions:

1. http://home.mcom.com/home/internet-search.html

- Is the two-process model necessary, i.e., should there be a separate information access process preceding actual information retrieval?
- Assuming that the two-process model is of value, is the use of semantic knowledge necessary for information access?

4.1.1 Is the Two-Process Model Necessary?

Essentially, we are questioning whether we should assume that relevant archives need to be located before searching for relevant information items. Let us assume that this is not necessary. Then, the only feasible alternative for finding information on the Internet would be to gather all the information items on the Internet into a single index [Lewis 1994]. In other words, we would be treating the entire Internet as a single gigantic, but "flat," collection. In contrast, organizing information into archives confers two practical advantages:

- It enables the search for relevant information to be progressively narrowed.
- It makes it easier to get a broader picture of the available information.
- It simplifies the task of updating the index when elements are added, deleted or modified.

Progressive narrowing is necessary especially in the context of keyword search with ambiguity. For example, let us assume that a user is looking for information about "mortar" (in the sense of "building material"). Let us also assume that there are databases of documents about weaponry, in which a significant number have the word "mortar" (in its "assault weapon" sense). If a two-process model is being used, the user can weed out all documents that deal with weaponry in one stroke by focusing only on databases that pertain to the desired sense of "mortar." But if the entire Internet is compressed into a single index, documents corresponding to both senses of "mortar" would be treated alike and hence, the user would be flooded with irrelevant documents.

Further, organizing the information by archives makes it easier to browse and navigate through the available information (as seen in the Gopher example of Figure 4.2). The experience of Gopher and WAIS suggests that finding relevant archives is a significant first step in finding information on the Internet.

There are however two disadvantages in this model. First, the archive needs to have a theme that characterizes its contents. If not, it will not be possible to search by generalization. Second, this method will not be able to locate documents that might be relevant to a user's query, but are part of an archive whose theme the user does not consider as relevant.

4.1.2 Is Semantic Knowledge Necessary for Information Access?

In the two-process model, the problem of information access is one of identifying relevant information archives. Typically, the relationship between a query and a relevant archive is that of *generalization*. For instance, if we are looking for information about the yen, an archive that deals with currencies or monetary units would be considered relevant. This process of generalization is important not only for automatic information retrieval, but also for manual searching as the following quote from [Schwartz et al. 1992] indicates:

Library science has developed methods over hundreds of years to construct a model in which the user, with some experience, can navigate through, locate, retrieve and use the desired information. In contrast, in the Internet every user is also a potential "publisher" and "librarian." No one expects the average user to be able to organize his or her information with such skill.

Therefore, generalization is almost always required, except in those cases where there is literal overlap between the query and the archive description. But as the discussion about the Internet Hunt in Section 4.3 shows, queries are typically related not literally but conceptually to archive descriptions.

4.2 Representation of Information Archives

This section describes how representations of information archives are constructed in NetSerf. The representations are constructed manually using a text-based editor, and disambiguated, again manually, using WordNet. This section also gives details of a World Wide Web site where readers can browse through the list of represented archives and other components of NetSerf.

The representation of an information archive is a list of <relation-type, relation-word> pairs. For each relation-word, NetSerf uses WordNet to identify all of its synsets. The user can disambiguate a relation-word by associating it with a subset of the synsets associated with that relation-word. For example, the World Factbook archive, whose text description is "World facts listed by country," is represented as:

TOPIC:	country	
	SYNSET:	[nation, nationality, land, country,a_people]
	SYNSET:	[state, nation, country, land, commonwealth, res_publica, body_politic]
	SYNSET:	[country, state,land,nation]
INFO-TYPE:	facts	

There are two points of note in the representation above. First, the topic of the archive has been separated from the type of information available. This is important because there might be many kinds of information about the same topic, e.g., pictures, text, sound files, etc. about rainforest birds. Second, the relation-word "country" has been associated with three of its four possible synsets. Without manual disambiguation, the representation would be:

TOPIC:	country	
	SYNSET:	[nation, nationality, land, country,a_people]
	SYNSET:	[state, nation, country, land, commonwealth, res_publica, body_politic]
	SYNSET:	[country, state,land,nation]
	SYNSET:	[rural_area,country]
INFO-TYPE:	facts	

In addition to synsets, a relation-word can be associated with other <relation-type, relation-word> pairs, thus creating "parent-child" relationships between pairs. For instance, the archive of "Supreme Court Rulings" would be represented as:

```
OBJECT:     ruling
            AUTHOR:     Supreme Court
                        SYNSET:     [Supreme_Court]
```

The relation-word "ruling" in the representation above has not been associated with any synsets since it has only one noun definition in WordNet, [opinion, ruling][1], i.e., it is unambiguous. A vocabulary of 32 relation-types has been used to construct the archive representations[2].

NetSerf's database currently contains representations of 227 Internet archives. Most of these are taken from two catalogs of information sources, the Whole Internet Catalog [Krol 1992] and the Internet Services List [Yanoff 1993]. These catalogs pair topic descriptions with information sources that contain information pertaining to the topic(s). They are compiled by users who explore the Internet looking for interesting information sources, a process colloquially referred to as *net surfing*.

In addition, some other archives from the Internet Hunt answer database were added to NetSerf. These are the archives considered as "correct" answers for the Internet Hunt questions that were used to evaluate NetSerf (the questions are described in Section 4.3.1). A World Wide Web site, **http://www.media.mit.edu/people/anil/NetSerf/NetSerf.html**, acts as a hypertext appendix to this chapter. It contains the representations of the

1. "Supreme Court" has two senses, the other one being [supreme_court, state_supreme_court, high_court].
2. **http://www.media.mit.edu/people/anil/NetSerf/NetSerf.html**

archives, the Internet Hunt questions, the correct answers to these questions, and the 32 relation-types used in the representations (with examples).

The representation language used by NetSerf is simple and facilitates the representations of most archive descriptions. But it is not rich enough for all cases. For instance, it is not possible to represent negatives, e,g., an archive which contains "all religious books except the Koran." This requires a system that can interpret the logical operation "NOT." Another shortcoming is the lack of representation for time. An archive that dealt with the "History of India before independence," would be an example of a representation which requires temporal reasoning capabilities.

4.3 Internet Hunt

This section describes the Internet Hunt game, and how it has acted as a source of queries to test NetSerf. The first part of the section gives examples from the Hunt, while the second part (Section 4.3.2) describes the query processor used by NetSerf.

4.3.1 Internet Hunt Questions

Here is how the game works. The creator of the Internet Hunt, Rick Gates, publishes ten questions at the beginning of the month. Participants are expected to answer these questions using only information available on the Internet. The correct answers are published the following month.

The questions are intended to test the participants's knowledge of the various archives and tools available on the Internet. Accordingly, there are two kinds of questions. The creator usually tries to contextualize the questions to make them interesting. Here is an example of a question from the Hunt that tests users's knowledge of where to find a specific archive:

A hurricane just blew in! Where can I find satellite photographs of its progress?

The other kind of questions pertains to various tools and facilities offered by the Internet. Here is an example of this kind:

I've heard there's a way to issue mail commands to get files via ftp. Where do I request this service, and what commands should I use to get this file?

The Internet Hunt is therefore a good resource for finding queries to evaluate NetSerf. It is independently motivated, and it is phrased in language that seems "natural" to users. Perhaps most important, independent answers are available for all the questions.

To turn the Internet Hunt questions into a set of queries for NetSerf, two filters were applied:

- First, questions that related to the usage of various Internet tools were not chosen, i.e., only the archive retrieval questions were selected. The reason for this is that NetSerf does not have any special knowledge of Internet tools. Instead, it is an attempt to use semantic knowledge for generalization in information access. Therefore, it does not make sense to test NetSerf on questions about Internet tools.

- Second, the questions were simplified to single sentences without pronouns. This is a requirement of the case structure representation formalism (Section 2.2.3). Therefore, the simplified version of the "hurricane" query above is: *Where can I find satellite photographs of the hurricane's progress?*

A total of 75 questions were used for testing NetSerf. The questions were divided into two subsets, one for training (40 questions) and the other for testing and evaluation (35 questions). The training subset was used to develop and refine the query processor as well as NetSerf's matching framework. The evaluation is described in Section 4.5, but before that, we will see how the query processor and the matching mechanism work.

4.3.2 Processing Internet Hunt Questions

The query processor is a sequential application of the following automatic modules: a part-of-speech tagger, a preprocessor, a representation extractor and optionally, a disambiguator. The first three components are described in this section. The description of the disambiguator is deferred to Chapter 5.

The query is first tagged by the Xerox PARC part-of-speech tagger [Cutting et al. 1992], which segments the query and assigns a part of speech to each token. The preprocessor module then eliminates common query introductions like "Where can I find," "What is" etc. It also extracts leading identifiers of the information type of the archive, like "satellite photographs" (in the "hurricane" query above), or "text" in queries like "Text of technology policy proposed by Bill Clinton." Therefore, after preprocessing, the query is assumed to consist of one or more topic words followed by prepositional phrases and verb clauses that modify either the topic words or preceding modifiers.

Then, the representation extractor module is used to locate the topic word(s) and its (their) modifiers. Topic words and modifiers are cast into <relation-type, relation-word> pairs, with the relation-type being based on whether the modifier is a noun modifier or a phrase/clause. For instance, the queries "Satellite photographs of hurricane's progress" and "What is the primary religion in Somalia?" are translated respectively into:

```
1. TOPIC:      progress
               PERTAINS-TO:       hurricane
   INFO-TYPE:  satellite photographs

2. TOPIC:      religion
               IN:        Somalia
```

The representation extractor uses WordNet to detect word collocations, e.g., in the query "What is the atomic weight of boron?" the relation-word extracted is "atomic weight," not "weight." Also, if the query does not completely fit the structural patterns expected by the processor, processing continues as far as the structural assumptions allow. For instance, the processor extracts "yen," but not "dollar," as a relation-word from the query "How many yen can I get for a dollar?" This is because the input following the word "yen" is neither a prepositional phrase nor a clause, and therefore, the representation extractor's expectations break down.

The final step of the representation extractor is to expand the main topic relation-words using semantic relations from the dictionary. Two examples:

given the topic word "pub," the pair <PERTAINS-TO "alcoholic beverage"> is generated (from the definition "**pub**: an establishment where alcoholic beverages are sold or consumed"), and for the topic word "pollution," the pair <HAS-OBJECT "environment"> is generated from the definitions "**pollution**: the action of polluting" and "**pollute**: to contaminate (an environment) especially with man-made waste."

4.4 Matching Queries to Information Archives

The two preceding sections showed how representations are constructed for information archives and how structured representations are automatically extracted from queries. In this section, we will describe the two steps of matching the query representations to the archive representations, and of ranking the resulting hits.

4.4.1 Matching a Query Representation to an Archive Representation

The matching step is a straightforward implementation of the generalization principle described in Section 4.1.2. A query representation, *QR,* matches an archive representation, *AR,* if some valid synset of some relation-word in *AR* is a hypernym of some valid synset of some relation-word in *QR.* We will call each such match a *hypernym-match.* Within *AR,* a valid synset is one that has been explicitly associated to some relation-word. Within *QR,* if the query has been disambiguated, a valid synset is one explicitly indicated by the disambiguator. If not, any synset of any relation-word in *QR* is valid.

As an example, let AR_1 denote the following archive representation of a monetary statistics archive[1]:

TOPIC:	monetary unit	
	SYNSET:	[monetary_unit]

Let QR_1 denote the following query representation of the currency "yen":

TOPIC:	yen	
	SYNSET:	[yen, Y]

To keep the example simple, in QR_1 the relation-word "yen" has been manually disambiguated to discard the irrelevant sense, [hankering, yen]. QR_1 matches AR_1 because [monetary_unit], which is a valid synset of AR_1, is a hypernym of [yen, Y], which is a valid synset of QR_1. As another example of hypernym-matching, consider the representation of the query "What is the primary religion in Somalia?" (in Section 4.3.2), and the representation of the archive "World Facts listed by country" (in Section 4.2). In this case, NetSerf would find a hypernym-match from the relation-word "Somalia" to the relation-word "country" using WordNet.

4.4.2 Ranking Matches

In the ranking step, a weight is assigned to every hit, i.e., to every *{QR AR}* match obtained from the previous step. The basic component of this weight is the number of hypernym-matches between relation-words in *QR* and *AR*.

1. Accessible via Gopher access to Hermes.Merit.Edu to "UM-Ulibrary" and to the menu "soc-sci," then "ec bulletin board," then "monetary stats."

Other components are added or subtracted for every hypernym-match, *H*, as follows:

- A positive weight is added if the two relation-types of *H* are equivalent, or if there is a hypernym-match between the parents of the two <relation-type, relation-word> pairs of *H,* or if the two relation-words of *H* are both top-level topic words.
- A negative weight is added if the two relation-types of *H* are not equivalent, or if a semantically important child of one pair of *H* does not have a counterpart in the other pair. For instance, consider the example given in the previous section where the query relation-word, "Somalia" is matched to the relation-word "country." The representations of the query and the archive are reproduced here for convenience.

  ```
  Query:    TOPIC:     religion
                       IN:     Somalia
  Archive:  TOPIC:     country
            SYNSET:    [nation, nationality, land, country, a_people]
            SYNSET:    [state, nation, country, land,
                        commonwealth, res_publica, body_politic]
            SYNSET:    [country, state,land,nation]
            INFO-TYPE:facts
  ```

  A negative weight is added in this case since "Somalia" has the relation-type "IN" which is not equivalent to "TOPIC," the relation-type of the word "country." Similarly, a negative weight is added when the query "religion in Somalia" is matched to the archive "religion in India," since the child of the word "religion" in the query (<IN Somalia>) does not have an equivalent counterpart in the archive.

Finally, ties are broken using the average *distance* of all the hypernym-matches between *QR* and *AR*. To get the distance between a synset and its hypernym in WordNet, we simply count the number of intervening links between the two. For example, consider matching the query representation QR_1 (of the query "How many yen can I get for a dollar?") to the following archive representation, AR_2. AR_2 represents an archive containing information about weights and measures[1]:

TOPIC:	unit of measurement	
	SYNSET:	[unit_of_measurement,unit]

Since [unit_of_measurement,unit] is a hypernym of [monetary_unit], it is by default a hypernym of [yen, Y]. Therefore, the distance from [yen, Y] to [unit_of_measurement, unit] is 2, while the distance to [monetary_unit] is 1. So among the hits for QR_1, NetSerf ranks AR_1 higher than AR_2.

Section 4.2 touched upon the lack of the logical operator "NOT" in NetSerf's representation language. The importance of such an operator is made clear by representations such as AR_1 and AR_2. Relation-words in archive representations are implicit generalizations. Therefore, AR_1 is considered relevant for any monetary unit by NetSerf, i.e., archive representations are universally quantified implicitly. So, in addition to the operators NetSerf provides for specializing relation-words, it is important to augment the representation language with operators like "NOT."

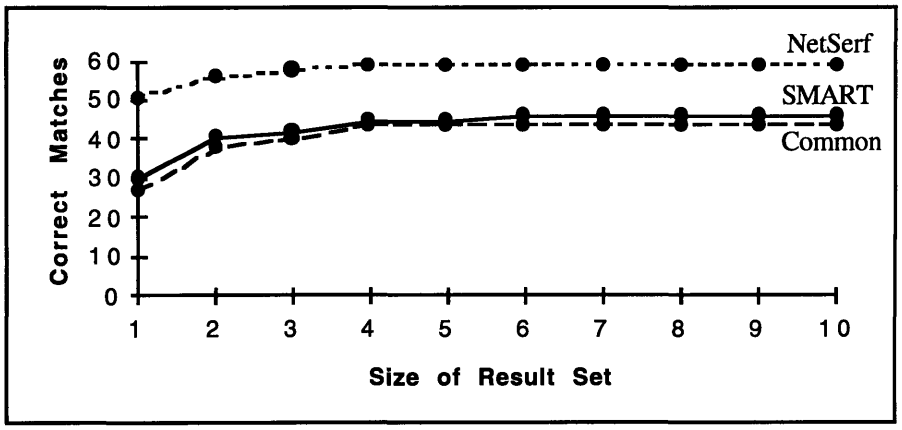
1. gopher scilibx.ucsc.edu 70 $\longrightarrow$ The Library $\longrightarrow$ Electronic Reference Books $\longrightarrow$ CIA World Factbook $\longrightarrow$ Weights+measures

4.5 NetSerf Performance Evaluation

This section reports results from the evaluation of NetSerf on a set of 75 questions chosen from the Internet Hunt collection. The questions were selected based on the criteria described in Section 4.3.1. This section also compares the performance of NetSerf to SMART. To run SMART on this set of questions, the text descriptions of the sources were gathered to form a single document collection. Lastly, the section looks at the performance of two stripped-down versions of NetSerf, one that uses structured representations but not semantic knowledge, and vice versa, in order to judge the influence of these components.

The strategy used to evaluate the two systems is as follows. For each query, the collection provides a set of one or more correct answers (archives). Also, for each query, both systems return a list of ranked matches (archives). The measure of each system's success rate was the number of queries for which any of the correct answers were found in the first *n* top-ranked hits returned by the system. Figure 4.3 shows the results for *n* from 1 to 10. For example, when we consider only the top-ranked hit, NetSerf is successful in matching 51 queries (out of 75), while SMART matches 30.

FIGURE 4.3. Performance of NetSerf and SMART on 75 Internet Hunt queries



Since NetSerf and SMART use very different retrieval techniques, it is interesting to find out if they are complementary, i.e., successful on different subsets of queries in the collection. The line titled "Common" in Figure 4.3 shows the number of queries that both systems were successful on. It indicates that NetSerf was successful on almost all the queries on which SMART was successful.

The two examples below give an idea of how NetSerf and SMART differ. They show cases where one succeeds while the other fails:

- NetSerf succeeds in matching the Hunt query "How many yen can I get for a dollar?" to the archive "Monetary statistics," but SMART cannot get this one because this is not a match based on literal keyword matching. NetSerf uses WordNet correctly to generalize from "yen" to "monetary unit."
- SMART succeeds in matching the query "List of environmental organizations in Denver, Colorado to speak at high school social studies classes" to the archive "Environmental education programs," while Net-

Serf fails to do so. SMART relies on the common word "environmental," but since NetSerf cannot relate the query topic word "organization" to the archive topic word "education," it cannot succeed based only on the common adjective "environmental."

FIGURE 4.4. Performance of NetSerf and SMART on training set

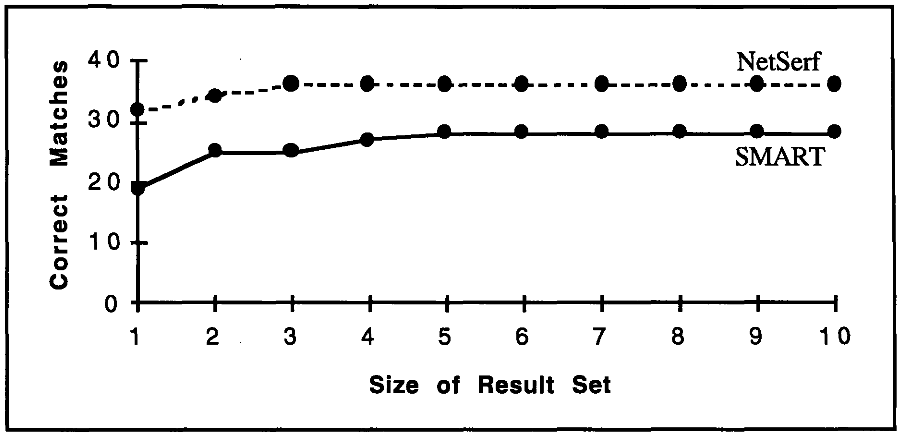
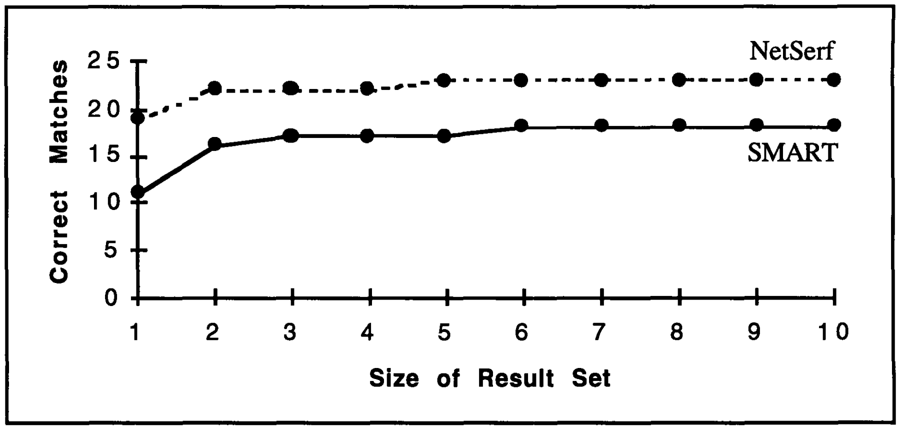


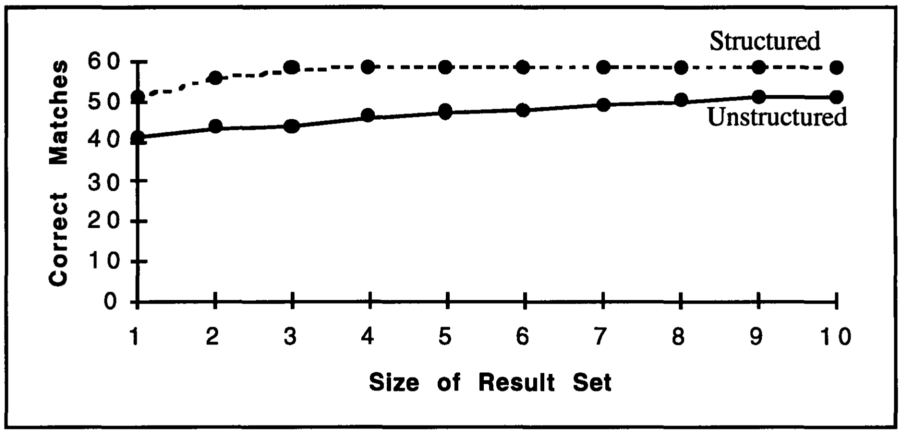
FIGURE 4.5. Performance of NetSerf and SMART on testing set



NetSerf was evaluated separately on the training and testing subsets of the 75 queries. Figure 4.4 and Figure 4.5 show these results. As expected, Net-

Serf performed better on the training subset (accuracy rate > 80%). On the testing set, the accuracy rates ranged between 54.29% and 65.71%. NetSerf performed better than SMART on both of these subsets.

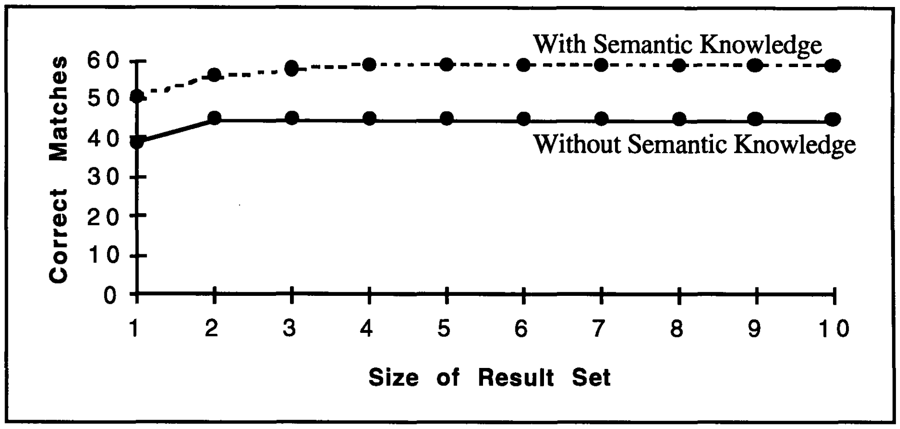
FIGURE 4.6. Performance of NetSerf without structured representations of queries



Described next are the individual influences of two components of NetSerf: structured representations and use of semantic relations. To test the effect of structured representations, a version of NetSerf which did not use structured representations of the queries was implemented. In this version, all the nouns, adjectives and other modifiers in the tagged query were generalized directly to find the hits. The hits were ranked using the method described in Section 4.4.2, excluding the formulae that need information about relation-types. For example, positive weights were not added for hypernym-matches because role information is not available to check whether matching query and archive relation-words have equivalent relation-types (roles).

Figure 4.6 presents the results of running this version of NetSerf, showing that the use of structured representations leads to improvements ranging from 15.7% to 24.4%.

FIGURE 4.7. Performance of NetSerf with/without semantic knowledge



To test the use of semantic knowledge in matching, a version of NetSerf that did not use semantic relations at all in the matching process was implemented. Hits were again ranked using the method described in Section 4.4.2. Clearly, without semantic knowledge, there cannot be any pure hypernym-matches, but there were matches based on synonymy (using synsets). As we see from Figure 4.5, if semantic knowledge is not used, performance drops quite significantly. The use of semantic knowledge leads to a clear improvement in NetSerf's performance (between 30.8% and 31.1%).

In summary, the performance evaluation of NetSerf shows that on the set of 75 Internet Hunt queries, NetSerf achieves better results than SMART. The two main components of NetSerf, structured representations and semantic knowledge, both contribute significantly to NetSerf's success. Suspending the use of either component leads to clear drops in performance. It should be noted that the results presented in this chapter were derived without using the disambiguator. The use of the disambiguator in NetSerf will be described in Section 5.4.

4.6 Problems and Opportunities

This section elaborates on three issues brought up in the preceding discussion: manual creation of archive representations, adequacy of the representation language, and extensions and applications of the NetSerf model.

4.6.1 Creation of Archive Representations

The need to construct archive representations manually might act as a bottleneck in extending NetSerf. It might be possible to automate the extraction of such representations by the development of specialized natural language processing tools. These tools would take as input designated documents such as README files or home pages (on WWW sites) that typically contain information about the contents of archives (meta-information). The tradeoffs between manually-created and automatically-created representations can be characterized in the following way:

- Manually-created representations are typically more accurate and disambiguated representations of the archive's contents. As shown in Section 4.2, if the representation editor is hooked up to a broad-coverage reference (like WordNet), disambiguation becomes simple. But there are two question marks hanging over manually-created representations.

    - First, will archive creators and maintainers take the trouble to create and maintain such representations? This depends on whether there is an incentive to do so. If the archives are made available on a commercial basis where a fee is charged for use, there is a greater likelihood that archive creators will create representations that enable potential users to reach their archives. One such "business" is the Associated Press photograph news service which provided the pictures for ImEngine (Chapter 3 shows some examples).
    - Second, will it be possible to build editors that make it easy and intuitive to create such representations? Here the picture is a little bleak. Researchers who have studied user interfaces for information retrieval have reported that users find even Boolean search interfaces non-intuitive.
- Automatically-created representations are attractive because they address the two drawbacks of manually-created representations. But the drawback here is the lack of NLP tools that are powerful enough. To create representations automatically, significant progress would be needed both in syntactic analysis, in order to deal robustly with text whose structure is unknown, and discourse processing, to identify the relevant meta-information parts of the archives.

4.6.2 Adequacy of Representation Language

As was touched upon in Section 4.2, NetSerf's representation language cannot handle exceptions and negations adequately. This can be remedied by enriching the representation language with logical operators. However, this has to be done with care because inference procedures in logic tend to get out of hand pretty quickly. For instance, there is no sound and complete inference procedure for second-order logic.

A second problem with the representation language is the vocabulary used for the roles (relation-types). For instance, if one user employs the role name "FOR" and another uses the role name "PURPOSE," are they referring to the same role? NetSerf can handle some variations in relation-words through the use of semantic knowledge, but any variations in role-names have to be explicitly indicated to the matching procedure.

4.6.3 Extensions and Indexing

NetSerf's performance could be significantly improved through the incorporation of other semantic knowledge sources. This is especially true of domain-specific sources of knowledge like MeSH (Section 2.3.3). For instance, let us consider an application where a program like NetSerf is being used to find catalogs of products and services on the Internet. Here, depending on the domain, particular semantic knowledge sources might be as useful as the "general" ones. For example, the computer industry maintains a fairly extensive index of its products and services. Other sources of such knowledge would be the Propaedia of Encyclopedia Britannica, the Patent and Trademark office, the Library of Congress and other large libraries.

From its role as a program that can incorporate domain-specific index information, NetSerf can be generalized to actually help a user in finding his/her way in a large index collection. This implies that in the general case, the task of information access can be viewed as helping the user find the appropriate index terms (instead of information archives). Once the right index terms are found, information retrieval can proceed as before.

This view takes us back to a 1960's debate about the proper role of automatic information retrieval systems. When full-text retrieval systems first came into vogue, its proponents argued that, in time, the systems would obviate the need for any indexing whatsoever. There were protests from the library science community (which till then had a monopoly on indexing). A well-known book by Vickery examined the arguments for and against the automation of retrieval [Vickery 1965]. Indexing, Vickery felt, needed only word frequency and thesaurus relation information, and therefore could be automated. Searching for information however, he argued, could not be automated as easily [Vickery 1965, page 151]:

> *Similar argument put against the mechanization of* searching *is perhaps on firmer ground. No matter how many relations we build into our system, we cannot include all the possible word associations of every potential user. Every enquirer has a unique network of associations in his own mind, and in searching a file he makes use of this network as well as of the network embodied in the file structure.*

Thirty years later, the right balance between human and machine functions has still not been struck. Finding the right index terms remains a challenge,

CHAPTER 5

Word-Sense Disambiguation

This chapter presents an algorithm for word-sense disambiguation, and discusses its application to ImEngine and NetSerf. In NetSerf, manual disambiguation was proposed as a solution to the problem of ambiguity in the creation of representations of information archives. In contrast, the algorithm presented in this chapter is completely automatic.

Ambiguity has been recognized as a problem almost since the beginning of computational natural language processing. For example, [Licklider 1965] argued for high-order formal languages to organize the world's body of knowledge in his book "Libraries of the Future" because:

> *The main shortcoming of English, and presumably of any natural language, is its ambiguity. Natural languages are so often used as adjuncts to nonlinguistic processes that natural languages do not have sufficient chance to practice independence and to develop self-sufficiency [Licklider 1965, Page 87].*

There are four basic kinds of ambiguity in natural language:

- Part-of-speech ambiguity
- Word-sense ambiguity
- Phrase attachment ambiguity
- Pronoun resolution ambiguity

Part-of-speech ambiguity arises because different parts of speech can be assigned to a word depending on the context. For instance, the word "left" could act as a noun, a verb or an adjective, and hence the ambiguity of the newspaper headline, "British Left Waffles on Falklands."

Even within a single part of speech, a word can have multiple meanings (senses), which is a second reason for word-sense ambiguity. When the natural language processing system encounters a word with many meanings, how is it to decide which sense is intended in the given context? For example, which meaning should be assigned to the word "shell" in the sentence "The soldiers load a mortar with a shell"?

Phrase attachment is the process of associating a phrase, say a prepositional phrase (PP), with a preceding word. Phrase attachment ambiguity results when the context provides many choices for such an association. For example, given the sentence, "John saw the man with the telescope," there are at least two choices: did John see the man using the telescope, or was the man carrying the telescope?

The fourth kind, pronoun resolution ambiguity, comes from the process of deciding what a pronoun refers to. Consider the sentence, "I opened the bottle with an opener and poured it out." Humans can use their world knowledge to decide that "it" refers to the bottle, because openers cannot be poured out, but a priori, there are two choices to be considered.

In fact, the need to endow computers with world knowledge has vexed many computational linguists. A well-known article by Bar-Hillel, a pioneer in machine translation, lays out the problem [Bar-Hillel 1960]. Bar-Hillel contrasted the sentence "The pen is in the box," with the sentence "The box is in the pen," and claimed that no existing or imaginable program would enable an electronic computer to determine that the word "pen" in the second sentence has the meaning "an enclosure where small children can play" [Bar-Hillel 1960, Barr & Feigenbaum 1981, Page 236]. Bar-Hillel also felt strongly about the need for world knowledge:

> *A translation machine should not only be supplied with a dictionary but also with a universal encyclopedia. This is surely utterly chimerical and hardly deserves any further discussion. ... We know... facts by inferences which we are able to perform ... instantaneously, and it is clear that they are not, in any sense, stored in our memory. Though one could envisage that a machine would be capable of performing the same inferences, there exists so far no serious proposal for a scheme that would make a machine perform such inferences in the same or similar circumstances under which an intelligent human being would perform them.*

Clearly, not everybody agreed with what Bar-Hillel had to say. The dissenters can be classified into two camps: those who agree with Bar-Hillel that significant world knowledge is needed for disambiguation, but do not accept that it cannot be supplied, and those who feel that other approaches to word-sense disambiguation are possible. Section 5.1 is a more detailed descriptions of these approaches.

The focus of this chapter is on resolving word-sense ambiguity. Word-sense disambiguation assigns a word to one or more senses in a reference by taking into account the context in which the word occurs. The reference can be a standard dictionary or thesaurus, or a lexicon constructed specially for some application. The context is provided by the text unit (paragraph. sentence, etc.) in which the word occurs.

Section 5.2 is a description of the disambiguator built as part of this thesis. This disambiguator is based on the two familiar reference sources, the Webster's Seventh Dictionary and the semantic thesaurus WordNet. Before the disambiguator is applied, the text input is processed first by a part-of-speech tagger and then by a phrase extractor which detects phrase boundaries. Therefore, for each ambiguous word, the disambiguator knows the part of speech, and other phrase headwords and modifiers that are adjacent to it. Based on this context information, the disambiguator uses a set of heuristics to assign one or more senses from the Webster's dictionary or WordNet to the word. For example, one of the heuristics relies on the fact that conjuncted head nouns are likely to refer to objects of the same category. Consider the ambiguous word "snow" in the sentence "Slush and snow filled the roads." In this sentence, the tagger first identifies "snow" as a noun. The phrase extractor indicates that "snow" and "slush" are conjuncted head words of a noun phrase. Then, the heuristic uses WordNet to identify the senses of "slush" and "snow" that belong to a common category. Therefore, the sense of "snow" as "cocaine" is discarded by this heuristic.

Section 5.3 is a description of the application of the disambiguator to ImEngine, while Section 5.4 describes its application to NetSerf. The chapter ends with a list of some of the open questions about the disambiguator.

5.1 Previous Work on Disambiguation

This section has three parts. The first part deals with some approaches to disambiguation that use custom-built lexicons. The second part deals with approaches that rely on standard references like thesauri and dictionaries. Finally, the section describes experiments on the role of disambiguation in information retrieval.

McRoy [McRoy 1992] describes a comprehensive disambiguation program developed as part of a system called TRUMP. The knowledge for disambiguation is derived from a custom-built lexicon and concept hierarchy, and a library of collocational patterns specifically designed for use in natural language processing. The disambiguator combines information from syntactic tags, word frequencies, collocations, semantic context, role-related expectations, and syntax restrictions.

Waltz and Boggess [Waltz & Boggess 1979] describe the use of knowledge about the physical world for disambiguation. Using a custom-built lexicon, their system constructs visual analogs for natural language input. This enables the system to distinguish between different senses of prepositions like "on." For example, the system constructs physical models to discriminate between the three different senses of "on" in "The leaf is on the tree," The light is on the ceiling," and "The fly is on the wing." The physical models also enable the system to make spatial inferences correctly.

The CYC knowledge base was described in Section 2.2.4. One of the early motivations for CYC was to provide a knowledge base sufficient for even complex disambiguation tasks [Lenat & Guha 1990, page 24]. They argue, like Bar-Hillel, that a program would need commonsense knowledge to distinguish between the senses of “pen” in the sentences “The ink is in the pen,” and “The pig is in the pen.” No performance results have been reported on the actual use of CYC for disambiguation.

In addition to the three systems described, there have been many disambiguation programs built using custom-built lexicons, e.g., [Small and Rieger 1982, Hirst 1987]. While they often succeed in resolving some hard cases, they face the problem of “scale-up,” i.e., how can their lexicons (vocabularies) be extended to handle unrestricted text (as from a newspaper)? The answer to this question is not obvious and is hotly contested. An alternative (described in the next section) is to try to make use of existing resources so that the resulting system can work on arbitrary text input, even if its methods are not as “commonsensical.”

5.1.1 Disambiguation with Standard References

These approaches can be broadly classified based on the reference from which senses are assigned, and on the method used to take the context of occurrence into account. The references considered here are Roget’s thesaurus, WordNet and on-line dictionaries. To take the context into account, researchers have used a variety of statistical weighting and spreading activation models.

[Yarowsky 1992] describes a disambiguator based on statistical models of Roget’s categories. Given a word, the task of the disambiguator is to assign it to one of the categories in Roget’s thesaurus which contains it, i.e., this disambiguator uses Roget’s categories as conceptual classes. The statistical class models are obtained through training on text from the Grolier's encyclopedia (without any manual intervention). A category is assigned by weighting the evidence given by other words in a 50-word window on either side of the ambiguous word.

Recently, there has been interest in using WordNet for disambiguation and for computing similarity (or "distance") between words (which might also be helpful for disambiguation), e.g., [Voorhees 1993, Resnik 1992]. The disambiguation algorithm described by [Voorhees 1993] partitions WordNet into *hoods*, which are then used as sense categories (like Roget's thesaurus classes). Individual synsets are not chosen as sense categories because that level is too fine-grained. During disambiguation, a single hood is selected for nouns based on the hood overlap with the surrounding text. Hood overlap is calculated according to a formula akin to inverse document frequency in vector-space indexing systems (Section 2.2.1). First, words in the surrounding text are checked to see which hoods they activate, which in turn depends on their WordNet synsets. The disambiguator selects the hood which has the largest difference between its document-induced activation and its collection-induced activation.

There has been considerable interest in the use of on-line (machine-readable) dictionaries for word-sense disambiguation. [Lesk 1986] was one of the earliest attempts to use dictionaries for disambiguation. For an ambiguous word, Lesk's program selected the dictionary sense whose definition had the greatest overlap with the word's surrounding text. For instance, Lesk's program could distinguish between an ice cream "cone"and a pine "cone" depending on the context.

Subsequent attempts to use on-line dictionaries have used not simply the text of dictionary definitions, but rather semantic relations extracted from the definitions, e.g., [Braden-Harder 1992, Guthrie et al. 1991, Janssen 1990, Ravin 1990, Vanderwende 1994, Veronis and Ide 1990]. The work of Braden-Harder, Ravin, and Vanderwende is reviewed here. Braden-Harder's method combined a variety of cues from the dictionary using a strategy for weighting various types of evidence. The cues used were grammatical codes (which provide subcategorization information, e.g., count noun), box codes (which provide semantic features like HUMAN in LDOCE), domain and subject codes, typical arguments, obligatory and optional prepositions and particles, example sentences, synonyms and hypernyms.

[Ravin 1990] also used some of these cues but brought into play more semantic relations (not just synonyms and hypernyms). Ravin addressed the problem of disambiguating the preposition "with" which has the following five senses in verb modification:

- Use ("to fish with a hook")
- Manner ("to attack with words")
- Alteration ("to fill with water")
- Co-Agency/Participation ("to combine with other parts")
- Provision ("to fit with clothes")

Ravin used a variety of heuristics based on semantic relations extracted from a dictionary. For example, cases such as "to fish with a hook" are handled by the INSTRUMENT heuristic. This heuristic succeeds when the genus term (hypernym) of a definition of the head noun is one of the set {"instrument," "implement," "device," "tool," "weapon"}. The relevant definition for the "hook" example is "**hook**: a curved or bent implement for catching, holding or pulling." This heuristic also enables the system to handle cases such as "sever with an ax."

Vanderwende [Vanderwende 1994] has used semantic relations extracted from an on-line dictionary to disambiguate noun sequences (nominal compounds). Disambiguation results from the process of interpretation of the noun sequence, i.e., the discovery of the relation between the two words of the noun sequence. For example, there is a possessive relationship between the words "family" and "estate" in the noun sequence "family estate." Similarly, Vanderwende's algorithm has rules for discovering 12 other relationships: Subject, Object, Locative, Time, Whole-Part, Part-Whole, Equative, Instrument, Purpose, Material, Causes, and Caused-by. If there are multiple interpretations, they are ordered by comparing weights assigned by the rule applications.

5.1.2 A Trade-off in Evidence Calculation

In any word-sense disambiguation algorithm, the basic step is to process the surrounding text in some fashion to generate evidence for the different senses of the ambiguous word. The evidence calculation method has to have two properties:

- It should not generate spurious evidence.
- It should apply to as many cases as possible.

The first criterion is necessary for the disambiguation algorithm to have high accuracy. The second ensures that the algorithm is not restricted to a small range of input. But the trade-off arises because evidence calculation methods that satisfy one criterion typically fail on the other. For example, the evidence calculation methods of the [Ravin 1990] and [Vanderwende 1994] algorithms look for ambiguous words in a particular syntactic combination, and then apply semantic knowledge in a very targeted fashion. Therefore, they easily pass the first criterion but that limits their applicability. Contrariwise, Lesk's algorithm has universal applicability, but uses a simple method to calculate evidence, and hence has a greater potential for generating spurious evidence. The disambiguation algorithm described in Section 5.2 is an attempt to increase applicability while continuing to apply semantic knowledge in a targeted manner.

5.1.3 Disambiguation and Information Retrieval

How important is disambiguation in information retrieval? It a well-known fact that there is considerable lexical ambiguity in documents, even in rather specialized databases [Krovetz and Croft 1992]. This section considers the work of [Krovetz and Croft 1992], [Sanderson 1994] and [Voorhees 1993] which have all shed light on this matter.

At first glance, it would seem that word-sense ambiguity could play havoc with keyword-based retrieval. Both the query and the document could use the word "shell" but in completely different senses. Without disambiguation, they would be identical to a retrieval system. [Krovetz and Croft 1992] examined two standard collections, CACM and TIME, to discover the extent of lexical ambiguity. In the CACM documents, there were on the average 4.4 senses per word (not counting stop words), and in the TIME documents, the average was 3.7.

However, [Krovetz and Croft 1992] found that resolving lexical ambiguity had little impact on retrieval effectiveness for documents that had many words in common with the query. This is because the greater the number of common words, the greater the likelihood that the query and the document are about the same topic. In other words, if a query matches a document only because of a few common, ambiguous words, the match will be overshadowed during ranking by more substantial matches. [Voorhees 1993] provided evidence for the corollary of this result. She showed that the performance of a retrieval system on short queries could be improved through disambiguation. In her experiment, disambiguation was carried out by hand, and hence this should be construed as a best-case result. [Sanderson 1994] also produced results supporting the view of [Krovetz and Croft 1992]. In addition, he discovered that if disambiguation is done poorly (which could happen because of an automatic disambiguation algorithm), the retrieval performance could be worse than with no disambiguation at all. The experiments described in Section 5.3 and Section 5.4 agree with these results.

5.2 Disambiguation with Adjacency Information

The disambiguation algorithm presented in this section builds on the work of [Vanderwende 1994], which deals with noun sequences, and [Ravin 1990], which deals with "with" prepositional phrases. The principal idea behind the algorithm is to increase the applicability of the algorithm by handling more word-to-word relationships, while keeping spurious evidence to a minimum through the use of specific disambiguation heuristics.

TABLE 5.1. The twelve adjacency relationships with examples

Adjacency Relationship	Example
Adjective modifying a noun	Express train
Possessive modifying a noun	Pharmacist's coat
Noun followed by a proper name	Tenor Luciano Pavarotti
Present participle gerund modifying a noun	Training drill
Noun noun	Basketball fan
Conjuncted nouns	A church and a home
Noun modified by a noun at the head of a following "of" PP	Barrel of the rifle
Noun modified by a noun at the head of a following "non-of" PP	A mortar with a shell
Noun that is the subject of an action verb	A monitor displays information
Noun that is the object of an action verb	Write a mystery
Noun that is at the head of a prepositional phrase following a verb	Sentenced to life
Nouns that are subject and object of the same action	The hawk found a perch

The input to the disambiguator is a pair of words, along with the *adjacency relationship* that links them in the input text. The 12 adjacency relationships used by the disambiguator are listed in Table 5.1. The adjacency relationships are obtained from the input text in two stages:

- First, the text is processed by the Xerox PARC part-of-speech tagger [Cutting et al. 1992] which results in the text being segmented and tagged with parts of speech. For example the input “An astronaut drives a lunar buggy” results in the following tags: [(:at “an”) (:nn “astronaut”) (:vbz “drives”) (:at “a”) (:jj “lunar”) (:nn “buggy”)]. The tags, which are from the Brown corpus set [Francis and Kucera 1982], stand for the following parts of speech: :at = article; :nn = singular or mass noun; :vbz = verb, 3rd singular present; :jj = adjective.
- In the second step, a phrase extractor is used on the tagged input to identify the adjacency relationships. For the example above, the following relationships are extracted:
    - Adjective “lunar” modifies the noun “buggy”
    - Noun “astronaut” is the subject of the action verb “drives”
    - Noun “buggy” is the object of the action verb “drives”
    - “astronaut” and “buggy” are subject and object of the same action

Before the disambiguator was implemented, the adjacency relationships were identified through an analysis of the captions of news photographs provided by the Associated Press. About 300 captions were analyzed to get examples of ambiguous words (and the contexts they occur in). The examples also helped in formulating the heuristic rules that could disambiguate the examples using semantic relations from the dictionary and WordNet. Each heuristic rule is associated with a particular adjacency relationship. In all, the disambiguator uses 39 heuristic rules, which are described in the following paragraphs. Some features that are common to many of these heuristics are summarized after these descriptions.

An adjective *A* modifying a noun *N*. There are three heuristic methods associated with this adjacency relationship:

- Use of examples;
  In the Webster's dictionary, examples are associated with particular definitions (senses). Therefore, if a word occurrence is analogous to an example, that is considered evidence for the sense associated with the example. The example can come from either the definition of the adjective *A* or of the noun *N*. For instance, "express train" is disambiguated by analogy to the example given by the definition, "**express**: adapted or suitable for travel at high speed <an express highway>." In contrast, "Japanese descent" is disambiguated using a definition of the noun, "**descent**: BIRTH, LINEAGE <of French descent>."
- Use of an "of" PP that modifies the genus term in the noun definition;
  Consider the combination "Indian flag." Using the definition "**flag**: a usually rectangular piece of fabric of distinctive design that is used as a symbol (as of a nation) or as a signaling device" and the semantic relation that India is a nation, this heuristic succeeds on this combination.
- If *N* has a sense which is a hyponym of "person," and *A* has a sense that denotes a place or region, those senses are selected. For instance, this rule succeeds in disambiguating "Palestinian demonstrator." The product sense of "demonstrator" is discarded.

A possessive *P* modifying a noun *N*. There are two heuristics for this adjacency relationship:

- If *P* has a sense that is a hyponym of "person," and *N* is an artifact, those senses are chosen, e.g., in the combination "pharmacist's coat," the garment sense of "coat" is selected.

- If *N* has a sense which is a meronym of *P* (PART-OF), those senses are selected. This is the case with "elephant's trunk."

A noun *N* followed by a proper name *P*. There is only one heuristic here. If the disambiguator can ascertain that P is a person's name and that N has senses that are hyponyms of "person," those senses are picked. This works for examples like "pitcher Curt Schilling" and "tenor Luciano Pavarotti."

A present participle gerund *V* (verb in "ing" form) modifying a noun *N*. The two heuristics used for this adjacency relationship are:

- Explicit indication of the action *V* in a definition of the noun *N*;
  The combination "training drill" is disambiguated in this way. The definition of the correct sense is "**drill**: the act or exercise of training soldiers in marching and the manual of arms."
- Picking senses which indicate a common context for *V* and *N*;
  For instance, consider "winning entry." The common context word "contest" is obtained from the following definitions: "**entry**: a person, thing, or group entered in a contest" and "**win**: to gain in or as if in battle or contest."

Noun noun sequence N_1 N_2. There are nine heuristics for this adjacency category. Many of the heuristics in this category were also used by [Vanderwende 1994], but not all the heuristics proposed there have been implemented here (and vice versa). The heuristics implemented here are the ones relevant to the examples drawn from the Associated Press captions.

- If N_2 has a sense that is a hypernym of some sense of N_1, choose those senses, e.g., "forehand return."

- If some sense of N_2 and some sense of N_1 have a common hypernym within a fixed threshold of semantic distance, those senses are selected. Examples of the successful use of this heuristic are "computer mouse," "lion cubs," and "ambulance bus."
- If N_2 has a sense that is a meronym of a sense of N_1 (PART-OF), choose those senses, e.g., "mosque compound." This follows from the fact that a mosque is a building, and the definition "**compound**: a fenced or walled-in area containing a group of buildings and especially residences."
- If N_2 has a sense that is a holonym of N_1 (HAS-PART), those senses are selected. An example of this is the combination "train car." This is disambiguated using the WordNet relation [train, railroad_train] HAS-MEMBER [car, railway_car, railroad_car].
- If some sense of N_2 and some sense of N_1 have a common meronym within a fixed threshold of semantic distance, those senses are selected. An examples of the application of this heuristic is in disambiguating the words "floor" and "window" in the phrase "third floor window." This is achieved through the relations [floor, level, storey, story] PART-OF [building, edifice] and [window PART-OF building] from "**window**: an opening especially in the wall of a building for admission of light and air that is usually closed by casements or sashes containing transparent material (as glass) and capable of being opened and shut."
- If N_1 has a sense which has a PURPOSE link to a sense of N_2, choose those senses. For example, "roundtable discussion" is resolved using "**roundtable**: a conference for discussion^1 or deliberation by several participants."

- Using example sentences, noun sequences can be disambiguated. For example, "soccer star" is disambiguated using "**star**: an outstandingly talented performer <a track star>."
- If a definition of N_2 has an "of" relation to N_1 or a hypernym of N_1, choose the corresponding senses. "Basketball fan" is disambiguated using "**fan**: an enthusiastic devotee (as of a sport or a performing art) usually as a spectator."
- By analogy to a word collocation. "Water tap" is disambiguated using the synset [faucet, water_faucet, tap, spigot, hydrant]. "Water tap" is analogous to "water faucet" since "tap" is a synonym of "faucet." Therefore, this is the correct sense of "tap."

Conjuncted nouns. There is only one heuristic to resolve ambiguous conjuncted nouns, but it is a powerful one. It looks for (and selects) senses of the two words that have a common hypernym within a fixed threshold of semantic distance. Examples abound: "slush and snow," "a crane and a ladder truck," "a church and a home."

A noun N_1 modified by a noun N_2 at the head of a following prepositional phrase with the preposition "of". Some examples of this relationship are "barrel of his rifle" and "jack of diamonds." There are six heuristic methods associated with this adjacency relationship. They are:

1. Since the dictionary is not internally disambiguated, this heuristic leads to only the word "roundtable" being disambiguated and not the word "discussion." When the semantic relations come from WordNet, both words are simultaneously disambiguated.

- If some sense of N_2 and some sense of N_1 have a common hypernym within a fixed threshold of semantic distance, those senses are selected. A good example is "jack of diamonds." From WordNet, we get [jack, knave] HYPERNYM [playing_card] and [diamond] HYPERNYM [playing_card].
- If some sense of N_2 and some sense of N_1 have a common meronym within a fixed threshold of semantic distance, those senses are picked. An example of this is "span of the freeway." The synsets [bridge, span] and [expressway, freeway, pike, state_highway, turnpike, superhighway, throughway, thruway] have a common hypernym [transportation_system, transportation, transit].
- If some sense of N_2 and some sense of N_1 have a common PURPOSE word within a fixed threshold of semantic distance, those senses are selected. In the phrase "service of the Zion Christian church," senses of the words "service" and "church" have the common purpose "worship."
- If N_2 has a sense that is a holonym of a sense of N_1 (HAS-PART), choose those senses. "Skin of the animal" is disambiguated from the WordNet relation: [animal, animate_being, beast, brute, creature, fauna] HAS-PART [hide, skin, pelt].
- Analogous dictionary definition involving N_1 and N_2; e.g., "barrel of his rifle" is disambiguated using "**barrel**: the discharging tube of a gun."
- Analogous dictionary example; e.g., "coat of green paint" is resolved using the definition "**coat**: a layer of one substance covering another <a coat of paint>."

A noun N_1 modified by a noun N_2 at the head of a following prepositional phrase with a preposition other than "of". This is an adjacency relationship that casts its net wide, and further work is needed to tease apart the roles played by the various prepositions. The three existing heuristics are:

- If N_2 has a sense that is a holonym of N_1, those senses are selected. An example of this is "branches on a linden tree."
- If N_2 has a sense that is a meronym of a sense of N_1, choose those senses, e.g., "dog with long ears."
- If N_2 has a sense which has a PURPOSE link to a sense of N_1, choose those senses. For example, "mortar with a shell" is resolved using the definition "**shell**: a projectile for cannon containing an explosive bursting charge" and the relation: [mortar, howitzer] HYPERNYM [cannon].

A noun *N* that is the subject of an action verb *V*. Two heuristics are used to disambiguate examples of this kind:

- If the purpose (action) of some sense of *N* is caused by some sense of *V*, choose those senses. For instance, the example words "monitor" and "display" in "monitor displays information" are resolved in this manner. From the definition "**monitor**: a receiver used to view the picture being picked up by a television camera," the disambiguator infers that the PURPOSE of a monitor is to "view" which entails "see." Further, "display" CAUSES "see." The rationale for this rule is that if an action *V* causes another action V_1, a likely sense of *N* is one that indicates that *N* is the agent of V_1. Therefore, this heuristic would not apply to the sentence "The monitor saw the incident."

- By analogy to an example sentence; e.g., "a wave crashes" is disambiguated using the example from the definition "**crash**: to make a smashing noise <thunder crashing overhead>."

A noun *N* that is the object of an action verb *V*. This is a commonly occuring adjacency relationship. Six heuristics apply here:

- If a sense of *N* entails the same action as a sense of *V*, select those senses. This heuristic is used to disambiguate "She delivers a pitch" using the definition "**pitch**: the delivery of a baseball by a pitcher to a batter."
- If the purpose (action) of a sense of *N* is entailed by a sense of *V*, those senses are chosen, e.g., "He watched the monitor."
- By analogy to example sentences, e.g., "write a mystery" is disambiguated using "**write**: COMPOSE <writes poems and essays>." Other examples are "pay royalty," "leave the courthouse."
- If *N* or a hypernym of *N* is the direct object in a definition of *V*, select those senses. e.g., "light a flame" is disambiguated using "**light**: to set fire to." Similarly, "feeds a carrot," "boards his bicycle," etc.
- If a sense of *N* has the same manner as a sense of *V*, select those senses. In verb definitions, manner is usually indicated by adverbs. For example. "hosted a reception" is disambiguated using the common manner word "social" which the heuristic gets from "**host**: to receive or entertain socially" and "**reception**: a social gathering often for the purpose of extending a formal welcome."
- If a sense of *N* has the same "container" as a sense of *V*, select those senses. The container can be any context, such as an event or phenomenon. For instance, "blocked a shot" is disambiguated using the common

container word "game" which comes from the definitions "**block**: to interfere usually legitimately with (as an opponent) in various games or sports," and "**shot**: a stroke or throw in an attempt to score points in a game (as tennis, pool, or basketball)."

A noun *N* that is at the head of a prepositional phrase following a verb *V*. This is another adjacency relationship with broad scope, subsuming for instance the "with" PPs tackled by [Ravin 1990]. Two heuristics are used:

- If either *N* or a hypernym of *N* is the direct object of a sense of *V*, select those senses. For instance, "sentenced to life" is disambiguated using the definitions "**sentence:** to impose a sentence on" and "**life**: a sentence of imprisonment for the remainder of a convict's life."
- If the purpose of some sense of *N* is the direct or indirect object of a sense of *V*, select those senses. This is used to disambiguate "adorn with a hood" using the definitions "**adorn**: to decorate especially with ornaments" and "**hood**: an ornamental scarf worn over an academic gown that indicates by its color the wearer's college or university."

Nouns N_1 and N_2 that are subject and object of the same action. Here again, there are two heuristics:

- If N_2 has a sense which has a PURPOSE link to a sense of N_1, choose those senses. So for example, the words "hawk" and "perch" in the sentence "The hawk found a perch" are disambiguated using "**perch**: a roost for a bird."

- If N_2 has a definition in which N_1 (or a hypernym of N_1) acts as the agent, choose those senses. For instance, the two nouns in "The council adopted a resolution" are disambiguated using the definitions "**resolution**: a formal expression of opinion, will, or intent voted by an official body or assembled group" and "**council**: a usually administrative body."

There are some salient features that are common to heuristics corresponding to different adjacency relationships. Here are some of those features:

- Several heuristics look for a particular semantic relation like hypernymy or purpose linking the two input words, e.g., "return" is a hypernym of "forehand."
- Many heuristics look for particular semantic relations linking the two input words to a common word or synset; e.g., a "church" and a "home" are both buildings.
- Many heuristics look for analogous adjacency patterns either in dictionary definitions or in example sentences, e.g., "write a mystery" is disambiguated by analogy to the example sentence "writes poems and essays."
- Some heuristics look for specific hypernyms such as person or place in the input words; e.g., if a noun is followed by a proper name (as in "tenor Luciano Pavarotti" or "pitcher Curt Schilling"), those senses of the noun that have "person" as a hypernym are chosen.

Here is how the disambiguator puts all the heuristics together. Given a pair of words and the adjacency relationship connecting them, the disambiguator applies all heuristics corresponding to that relationship. Those word senses that are rejected by all heuristics are discarded, i.e., if a word sense is suggested by some heuristic, it is retained as an alternative. The disambiguator uses a very simplistic mechanism for ranking alternatives suggested by different heuristics. Each heuristic rule returns a weight that is the number of semantic relations used to link the input words. For example, the heuristic that disambiguates "barrel of his rifle" from the definition "**barrel**: the discharging tube of a gun," uses the semantic relation [rifle A-KIND-OF gun]. Therefore, this rule returns a weight of 1. The senses are ranked in order of increasing weight. Therefore, if we assume that counting the number of intervening semantic relations is a good measure of semantic distance, we can say that the disambiguator ranks alternatives in order of increasing semantic distance.

We are now ready to check whether the disambiguator, with all its bells and whistles, actually leads to improved performance in ImEngine and NetSerf. To be honest, dear reader, disappointment lies ahead.

5.3 Disambiguation in ImEngine

In Section 3.5, ImEngine (without disambiguation) was evaluated on the set of 280 queries collected from an experiment that used 50 pictures and video clips. This section evaluates the performance of a version of ImEngine with the disambiguator included.

FIGURE 5.1. Performance of ImEngine with and without disambiguation

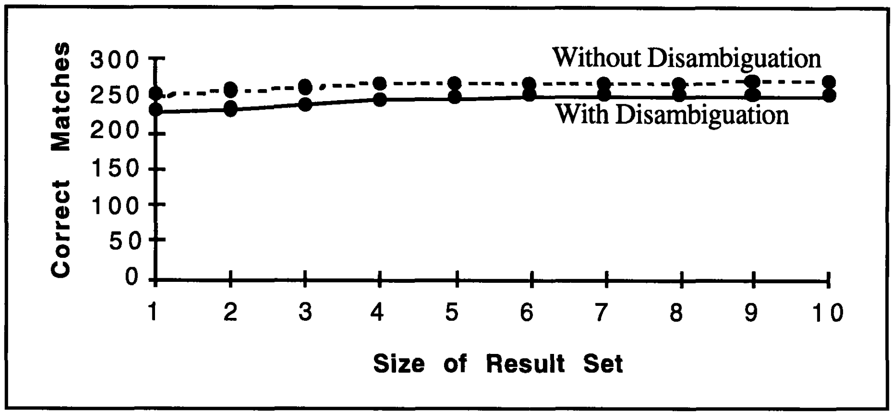


Figure 5.1 shows the performance of ImEngine with the disambiguator. As in Section 3.5, results are shown for *n* from 1 to 10, where *n* is the number of top-ranked hits considered. Figure 5.1 reveals that the disambiguated version performs worse than the undisambiguated version for all *n*. Although disappointing, this is in agreement with the results of [Sanderson 1994] (Section 5.1.3). Also, there were no queries on which only the disambiguated version was successful, i.e., the undisambiguated version retrieved the correct answer for every successful query of the disambiguated version. There are two reasons for the lack of improvement:

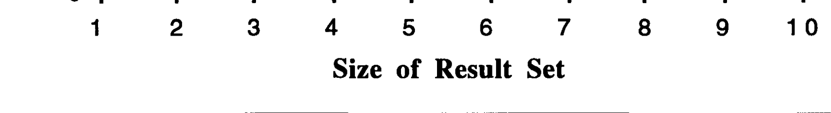
- The disambiguator is not applicable widely enough. The case structure representations of the 50 pictures and video clips have 984 cases (<role, word> pairs). Of these 984 words, 477 have only one sense in WordNet, i.e., they are unambiguous. Of the remaining 507 words, the disambiguator's heuristics were applicable only to 209 (41.2% applicability).
- The disambiguator is not accurate enough. Of the 209 words, the disambiguator picked the correct sense (through one of its heuristics) for 161 words (77% correct). Unfortunately, it seems that this rate of accuracy

was not sufficient for the disambiguated version to perform as well as the original version. Because of inaccurate disambiguation, the new version seems to have missed correct matches between words that were captured by the original version.

5.4 Disambiguation in NetSerf

NetSerf with disambiguation follows the same pattern as ImEngine with disambiguation. Figure 5.2 shows the results for *n* from 1 to 10. Again, the undisambiguated version is more successful than the disambiguated version.

FIGURE 5.2. Performance of NetSerf with and without disambiguation



As with ImEngine, there are two reasons for the poor showing of the disambiguated version. The disambiguator is even less widely applicable on the Internet Hunt queries. The case structure representations of the 75 queries have 463 cases (<role, word> pairs). 277 of these 463 words are unambiguous. The disambiguator's heuristics apply only to 25 of the other 186 (13.4%). The accuracy rate is 64%.

5.5 Problems and Opportunities

There is considerable room for improving the performance of the disambiguator. First, it might be possible to incorporate other disambiguation techniques, especially statistical ones (e.g., [Yarowsky 1992]) in order to increase applicability. Second, instead of using only pairs of words, the heuristic rules could use a "window" of words, or several words at a time. Third, the notion of adjacency relationships can be refined to make finer distinctions when dealing with prepositional phrases. The prepositions at the head of prepositional phrases carry role information that is valuable in disambiguation, as shown by [Ravin 1990]. Fourth, heuristic rules within an adjacency category need to be prioritized based on a weighting scheme, so that heuristics with higher priority can take precedence.

There is also scope for improving the application of the disambiguator in information retrieval. Recognizing that the disambiguator is never perfect, the retrieval system should optimize the use of the disambiguator, i.e., the input to the disambiguator should be carefully selected [Krovetz and Croft 1992]. It is reasonable to expect that a good disambiguator can significantly improve the performance of a retrieval system that uses semantic knowledge, even though it must be admitted that the results of Section 5.3 and Section 5.4 do not support this view. It is ironic that the conclusions of this chapter should be so ambiguous.

CHAPTER 6

Summary and Conclusions

This thesis presents a new framework, the case structure representation framework, for the application of semantic knowledge to information access and retrieval. The thesis deals with the acquisition of semantic knowledge for retrieval, and with issues of salience and ambiguity in the application of semantic knowledge. Two applications that were implemented using this framework, ImEngine and NetSerf, are described in the thesis.

Two sources of semantic knowledge are used in this framework: WordNet, a semantic network built at Princeton University, and an on-line Webster's dictionary. Using a small fixed vocabulary of semantic relation types, these sources provide semantic knowledge for a large number (> 40000) of words and word collocations. Therefore, these semantic knowledge sources can be characterized as "shallow" and "broad," whereas typically, artificial intelligence knowledge bases are relatively "deep" and "narrow."

One of the main attractions of keyword-based retrieval systems is their universal applicability, which they achieve by not relying either on domain or world knowledge. Though the semantic knowledge provided by the two sources is not as "deep" or detailed as in artificial intelligence knowledge bases (such as CYC), their breadth enables the case structure representation framework to be widely applicable. This stands in contrast to AI-based retrieval systems (such as SCISOR), which are applicable only in the domains in which they have special knowledge.

Earlier attempts to use semantic knowledge from dictionaries and thesauri (e.g., [Cohen & Kjeldsen 1987, Voorhees 1994]) have focused on keyword expansion, a technique in which additional keywords are added to the query and/or the document using the semantic relations of the original keywords. Because of the lack of automatic procedures for context-dependent selection of keywords and semantic relations for expansion, these attempts have not showed performance improvements over standard keyword-based systems. In contrast, the case structure representation framework identifies the salient keywords in the text and assigns a role to each keyword. Thus, the problem of choosing the right keywords is tackled by the representation, and the problem of choosing the right semantic relations is tackled by rules that use role information to guide the matching of the query to the documents. In addition, precision is kept high by placing emphasis on matches between words that play the same role.

Chapter 2 (Section 2.3.2) describes a new technique developed in the thesis for extracting semantic relations from on-line dictionaries. In this technique, dictionary definitions are first processed into a set of kernel semantic relations. Second, a pattern definition language is provided, along with an interpreter, to enable the extraction of custom semantic relations. The patterns are matched using the kernel semantic relations of one or more dictionary definitions. This technique has proved very valuable in extracting semantic relations for retrieval applications, and for dictionary-based word-sense disambiguation.

Applications. The case structure representation framework has been applied to two applications, ImEngine and NetSerf. ImEngine is a program for retrieving captions of pictures and video clips through natural language queries. ImEngine has automatic procedures for constructing case structure representations of queries and captions. The representations are matched and ranked using a set of weighted matching rules, the most important of which is generalization from the caption to the query (Section 3.4).

Unlike keyword-based text retrieval methods, ImEngine addresses the problem of matching captions to literally different but conceptually equivalent queries. Owing to the lack of standard benchmarks for testing caption-based image retrieval systems, an experiment was conducted to collect such queries for ImEngine's evaluation (Section 3.3). On this test set, ImEngine's performance was compared to that of SMART, a standard keyword-based retrieval system. ImEngine exhibits performance improvements of 4.2% to 10.4% over SMART (Section 3.5).

ImEngine is complementary to picture retrieval systems that are based on image processing techniques. There are robust techniques for inferring physical properties like color, texture and brightness from images. But as yet, there are no computational methods that can "understand" pictures or video well enough to produce high-level descriptions of who, where, when, or what the picture or video is about. That is the kind of information contained in captions. Indeed, it is the physical properties that are often not made explicit in captions.

NetSerf. NetSerf is the other application of the case structure representation framework. It is an information access program for finding information archives on the Internet using natural language queries. Using the two semantic knowledge sources, the query representations are matched to hand-coded case structure representations of the archives. In contrast to literal pattern-matching tools like Archie, Veronica and WAIS, NetSerf's mechanism is somewhat analogous to the process of locating a book in a library by searching through a more general section.

NetSerf has been evaluated on a set of queries collected from a game called *Internet Hunt* [Gates 1992]. Internet Hunt provides an independent set of queries with answers for testing NetSerf. NetSerf does 28.3% to 70% better than SMART on this query set (Section 4.5).

Component Analysis. There are three components that together distinguish the case structure representation approach: structured representations, semantic knowledge from WordNet and the Webster's dictionary, and disambiguation. The influence of each of these components on ImEngine and NetSerf was evaluated by comparing the programs to versions that did not include that component. For instance, to test the effect of disambiguation on ImEngine, a version of ImEngine with the disambiguator was compared to a version without the disambiguator.

The use of structured representations improves the performance of both ImEngine and NetSerf significantly. ImEngine's performance improved by 85.8% to 108% (Section 3.5), whereas NetSerf's performance improved between 15.6% and 24.4% (Section 4.5). Similarly, the use of semantic knowledge also led to clear improvement. ImEngine saw improvements of 68.9% to 75%, while NetSerf gained by 30.8% to 31.1%. These numbers provide evidence that both structured representations and semantic knowledge are important components of the case structure representation framework.

Disambiguation. The role of disambiguation was evaluated using the disambiguator described in Chapter 5. The disambiguator uses a set of heuristics to assign one or more senses from the Webster's dictionary or WordNet to ambiguous words. The heuristics are grouped according to adjacency relationships between words. These relationships are automatically detected through part-of-speech tagging and phrase extraction. Unlike previous efforts which were aimed at particular adjacency relationships, e.g., [Ravin 1990, Vanderwende 1994], this disambiguator's heuristics are used to resolve 12 adjacency relationships, thus widening its applicability.

But unlike the other two components, the disambiguator did not succeed in improving the performance of either ImEngine or NetSerf. In fact, there was a slight deterioration in performance for the disambiguated versions. Although this seems counterintuitive, the result is in line with other established work, e.g., [Sanderson 1994]. Here is an explanation: The disambiguated version of NetSerf or ImEngine matches not words to words, but word senses to word senses. Since this is a more restrictive criterion, the disambiguated version comes up with at most as many hits as the undisambiguated version (usually, it comes up with fewer hits). Therefore, if the disambiguator is not very accurate, the disambiguated version will miss matches that the undisambiguated one gets (Section 5.3 and Section 5.4).

6.1 Open Questions

In this concluding section, six issues pertaining to ImEngine and NetSerf are discussed: the limitations of the case structure representation, the role of disambiguation, manual representation and indexing, additional knowledge sources, automatic acquisition of the weights used in the matching rules, and possible use of complementary methods of retrieval.

The case structure representation has been enormously useful in ImEngine and NetSerf. But it has limitations. First, it makes sense only for single sentences. Of course, full-text retrieval systems have to deal with documents longer than that. Even for video clip captions, case structure representations are not sufficient to represent temporal changes (Section 3.6). Second, it is not a suitable representation for all kinds of sentences. It works best for sentences describing actions or situations (Section 2.2.3).

In information retrieval, the role of disambiguation is somewhat intriguing. On the one hand, there is clear evidence of extensive lexical ambiguity in texts [Krovetz and Croft 1992]. On the other, studies have shown that if both the query and the document are reasonably long, the influence of ambiguity will be drowned out [Voorhees 1993, Sanderson 1994]. Keyword expansion is the one element in the brew that makes these results inconclusive. If keywords are expanded through inappropriate senses, that should lead to reduced precision. Indeed, this is confirmed by some of the studies on use of synonyms from thesauri for expansion. But as Chapter 5 shows, the use of the disambiguator actually makes ImEngine and NetSerf (which both use expansion) fare worse. It is to be hoped that keyword expansion with better disambiguators will enhance recall without reducing precision.

NetSerf's representations of information archives are coded by hand. Is this a reasonable assumption if we want the system to be able to "scale up"? Ideally, of course, it would be best if the representations were acquired automatically. But, as was argued in Section 4.6, if there is an economic incentive to create high-quality representations by hand, it will be done (the Associated Press photo news service is itself an example). More generally, NetSerf is connected to the larger question of indexing. If manual indexing is used widely, NetSerf or something along the same lines can be used to find appropriate index terms rather than representations of information archives.

WordNet and the on-line dictionaries provide "general" world knowledge. Techniques to garner semantic knowledge from other general knowledge sources, such as encyclopediae and fact books, would significantly increase the retrieval system's semantic knowledge store. There is also reason to believe that special domain knowledge can enhance retrieval (Section 2.2.4). One hope is that the experience gained in processing general, on-line dictionaries will transfer to domain-specific dictionaries, thesauri and glossaries.

The matching rules in both ImEngine and NetSerf both use weights to prioritize rule application and to rank the resulting hits. In both systems, the weights were adjusted manually using the training sets of queries. The intent was to get best-case results for the two systems. However, as pointed out by [Maes 1995], the existing weights could be used as a starting point for automatic acquisition of matching weights. Using the training sets, a hill-climbing algorithm [Shapiro 1992] could modify the weights incrementally to reach a local maximum, i.e., a combination of weights that maximizes the number of training set queries that the system is successful on. Automatic acquisition of weights offers two advantages:

- The resulting assignment of weights would be at least as good as the hand-coded version. Hence, ImEngine and NetSerf might display even better results.
- It would make it easier to reconfigure the matching weights if new rules are introduced or if existing rules are modified.

It would be interesting to combine other methods of retrieval with the methods proposed in this thesis. For instance, Chapter 3 argues that ImEngine and retrieval techniques based on image processing are complementary. Much could be learnt by combining the two into a single system. More generally, other text retrieval methods might also be candidates for combination. Two possible avenues:

- In choosing salient keywords, the case structure representation framework pays no attention to the staple diet of vector-space indexing systems, term frequency and inverse document frequency. This might help refine the framework's notion of salience.
- In expanding keywords, only semantic relations are considered. There are other kinds of thesauri such as co-occurrence thesauri that have something to offer.

Beyond search and retrieval, there are many fields of research that influence the overall process of finding information; e.g., natural language processing [Smeaton 1990], information agents [Lashkari et al. 1994], information presentation [Weitzman 1995], and user modelling [Orwant 1991]. But information retrieval has been enough to keep me busy for some time now.

CHAPTER 7

References

Alshawi, H. 1987. "Processing Dictionary Definitions with Phrasal Pattern Hierarchies," *Computational Linguistics*, **13**(3-4), pp. 195-202.

Amsler, R. 1980. *The Structure of the Merriam Webster Pocket Dictionary*, Ph.D Dissertation, University of Texas, Austin.

Anderson, J. R. 1990. *Cognitive Psychology and its Implications*, W. H. Freeman and Co., New York.

Bar-Hillel, Y. 1960. "The Present Status of Automatic Translation of Languages," in *Advances in Computers*, F. L. Alt, editor, Academic Press, New York.

Barr, A., and Feigenbaum, E. A. 1981. *The Handbook of Artificial Intelligence*, Volume 1, Addison-Wesley Publishing Company, Reading, MA.

Belkin, N. J., and Croft, W. B. 1992. "Information Filtering and Information Retrieval: Two Sides of the Same Coin?," in *Communications of the ACM,* **35**(12).

Berners-Lee, T., Cailiau, R., Groff, J., and Pollermann, B. 1992. "World-Wide Web: The Information Universe," in *Electronic Networking: Research, Applications and Policy,* **2**(1), pp. 52-58.

Bobick, A. F. 1993. "Representational Frames in Video Annotation," MIT Media Laboratory Perceptual Computing Technical Report No. 251.

Boguraev, B. K., and Briscoe, T. 1987. "Large Lexicons for Natural Language Processing: Exploring the Grammar Coding System of LDOCE," in *Computational Linguistics,* **13**, pp. 203-218.

Brachman, R. J. 1989. "On the Epistemological Status of Semantic Networks," in *Associative Networks: Representation and Use of Knowledge by Computers,* N. V. Findler, (editor), Academic Press.

Brachman, R. J., and Levesque, H. J. (eds). 1985. *Readings in Knowledge Representation,* Morgan Kaufman Publishers, Inc.

Braden-Harder, L. 1992. "Sense Disambiguation Using On-line Dictionaries," in *Natural Language Processing: The PLNLP Approach,* Jensen, K., Heidorn, G. E., and Richardson, S. D., editors, Kluwer Academic Publishers.

Brajnik, G., Guida, G., and Tasso, C. 1990. "User Modeling in Expert Man-Machine Interfaces: A Case Study in Intelligent Information Retrieval," in

IEEE Transactions on Systems, Man, and Cybernetics, 1990, **20**, pp. 166-185.

Briscoe, T. 1994. (editor). *Workshop on the Future of the Dictionary,* Uriage-les-Bains, October 1994.

Bush, V. 1945. "As We May Think," in *Atlantic Monthly,* July 1945, reprinted in *Readings in Information Retrieval,* H. S. Sharp, editor, The Scarecrow Press, Inc., New York, 1964.

CCITT/ISO. 1988. *The Directory, Part I: Overview of Concepts, Models, and Services.* Gloucester, England.

Chakravarthy, A. S. 1994a. "Representing Information Need with Semantic Relations," in *Proceedings of COLING-94,* Kyoto, Japan.

Chakravarthy, A. S. 1994b. "Toward Semantic Retrieval of Picture and Video Clips," in *Proceedings of RIAO'94*, New York.

Chakravarthy, A. S. 1995. "Sense Disambiguation Using Semantic Relations and Adjacency Information," submitted to *ACL-95* student session.

Chakravarthy, A. S. and Haase, K. B. 1995. "NetSerf: Using Semantic Knowledge to Find Internet Information Archives," to be published in *Proceedings of ACM SIGIR'95,* Seattle, Washington.

Chodorow, M. S., Byrd, R. J., and Heidorn, G. E. 1985. "Extracting Semantic Hierarchies from a Large On-Line Dictionary," in *Proceedings of the 23rd ACL.*

Cohen, P. R., and Kjeldsen, R. 1987. "Information Retrieval by Constrained Spreading Activation in Semantic Networks," in *Information Processing and Management*, **23**(4), 255-268.

Cutting, D., Kupiec, J., Pedersen, J. and Sibun, P. 1992. "A Practical Part-of-Speech Tagger," in *Proceedings of the 3rd Conf. on Applied NLP.*

Davis, M. E. 1993. "Media Streams: An Iconic Language for Video Annotation," in *Proceedings of the IEEE Workshop on Visual Languages*, Bergen, Norway.

DIALOG. 1995. Internet Manuscript, **http://www.dialog.com/**

Dolan, W. B., Vanderwende, L., and Richardson, S. D. 1993. "Automatically Deriving Structured Knowledge Bases from On-line Dictionaries," in *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, Vancouver.

Emtage, A. and Deutsch, P. 1992. "Archie - An Electronic Directory Service for the Internet," *Proceedings of the USENIX Winter Conference*, pp. 93-110.

Fillmore, C. J. 1968. "The Case for Case," in *Universals of Linguistic Theory*, edited by E. Bach and R. T. Harms, Holt, New York.

Francis, W. N., and Kucera, F. 1982. *Frequency Analysis of English Usage*, Houghton Mifflin.

Fox, E. A., Nutter, J. T., Ahlswede, T., Evens, M., and Markowitz, J. 1988. "Building a Large Thesaurus for Information Retrieval," in *Proceedings of the Second Conf. on Applied NLP.*

Gates R. 1992. *Internet Hunt.* Internet Manuscripts published monthly since September 1992, available through anonymous FTP from ftp.cic.net; directory pub/internet-hunt

Gudivada, V. N., and Raghavan, V. V. 1994. "Picture Retrieval Systems: A Unified Perspective and Research Issues," Technical Report, Department of Computer Science, Ohio University, Athens, Ohio.

Guthrie, J. A., Guthrie, L., Wilks, Y., and Aidinejad, H. 1991. "Subject-Dependent Co-occurrence and Word-Sense Disambiguation," in *Proceedings of ACL-91.*

Haase, K. B. 1991. "Making Clouds from Cement: Building Abstractions around Concrete Examples," in *Proceedings of the US-Japan Workshop on Integrated Comprehension and Generation Systems in Perceptually-Grounded Environments*, Las Cruces, New Mexico.

Haase, K. B. 1994. Personal communication.

Haase, K. B. 1995. "Mapping Texts for Information Extraction," submitted to *SIGIR'95.*

Hagel, J., and Eisenmann, T. R. 1994. "Navigating the Multimedia Landscape," in *The McKinsey Quarterly,* **3**, pp. 39-55.

Harman, D (ed). 1993. *The First Text REtrieval Conference (TREC-1)*, National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, Maryland, 20899.

Harrenstein, K., Stahl, M., and Feinler, E. 1985. "NICName/WHOIS," RFC 954, SRI International.

Hirst, G. 1987. *Semantic Interpretation and the Resolution of Ambiguity*, Cambridge University Press.

Horn, B. K. P. 1986. *Robot Vision.* MIT Press, Cambridge, Massachusetts.

Jacobs, P. S., and Rau, L. 1990. "SCISOR: Extracting Information from On-line News," in *Communications of the Association for Computing Machinery*, **33**(11):88-97.

Janssen, S. 1990. "Automatic Sense Disambiguation with LDOCE: Enriching Syntactically Analyzed Corpora with Semantic Data," in *Theory and Practice in Corpus Linguistics*, J. Aarts and W. Meijs, editors, Rodopi, Amsterdam, pp. 105-135.

Jing, Y., and Croft, B. 1994. "An Association Thesaurus for Information Retrieval," in *Proceedings of RIAO'94: Conference on Intelligent Multimedia Information Retrieval Systems and Management*, New York.

Kahle, B. and Medlar, A. 1991 "An Information System for Corporate Users: Wide Area Information Servers," in *ConneXions - The Interoperability Report*, **5**(11), pp. 2-9.

Katz, J. J., and Fodor, J. A. 1963. “The Structure of a Semantic Theory,” in *Language,* **39**, pp. 170-210.

Kirkpatrick, B. (ed). 1988. *The Authorized Roget’s Thesaurus of English Words and Phrases*, Penguin Books.

Klatzky, R. L. 1979. *Human Memory.* W. H. Freeman & Company, New York.

Krol, E. 1992. *The Whole Internet: User's Guide and Catalog,* O'Reilly and Associates, Inc., Sebastopol, CA.

Krovetz, R., and Croft, W. B. 1992. “Lexical Ambiguity and Information Retrieval,” in *ACM Transactions on Information Systems*, **10**(2), pp. 115-141.

Lancaster, F. W. 1968. *Information Retrieval Systems: Characteristics, Testing and Evaluation*, John Wiley and Sons, Inc.

Lancaster, F. W. 1972. *Vocabulary Control for Information Retrieval*, Information Resources Press, Washington, D. C.

Lashkari, Y., Metral, M., and Maes, P. E. 1994. “Collaborative Interface Agents,” in *Proceedings of AAAI-94.*

Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., and Feng, F. 1993. “Description of the CIRCUS system used for TIPSTER Text Extraction,” in *Proceedings of the TIPSTER Phase 1 Final Meeting.*

Lenat, D. B., and Guha, R. V. 1990. *Building Large Knowledge-based Systems*, Addison-Wesley, Reading, MA.

Lenat, D. B., and Guha, R. V. 1994. "Strongly Semantic Information Retrieval," in *Workshop Notes of the AAAI-94 Workshop on Indexing and Reuse in Multimedia Systems.*

Lesk, M. 1986. "Automatic Sense Disambiguation: How to Tell a Pine Cone from an Ice Cream Cone," in *Proceedings of the SIGDOC Conference.*

Lewis, D. D. 1994. Personal communication.

Licklider, J. C. R. 1965. *Libraries of the Future*, MIT Press, Cambridge, Massachusetts.

McCahill, M. 1992. "The Internet Gopher: A Distributed Server Information System," in *ConneXions - The Interoperability Report*, **6**(7), pp. 10-14.

McRoy, S. 1992. "Using Multiple Knowledge Sources for Word Sense Discrimination," in *Computational Linguistics*, **18**(1).

Metzler, D. P., and Haas, S. W. 1989. "The Constituent Object Parser: Syntactic Structure Matching for Information Retrieval," in *ACM Transactions on Information Systems*, **7**(3).

Miller, G. A. 1985. "Dictionaries of the Mind," in *Proceedings of the 23rd Meeting of the Association of Computational Linguistics.*

Miller, G. A. 1990. "WordNet: An On-line Lexical Database," *International Journal of Lexicography,* **3**(4).

Miller, G. A., and Fellbaum, C. 1991. "Semantic Networks of English," in *Cognition*, **41**, pp. 197-229.

Minsky, M. 1975. "A Framework for Representing Knowledge," in *The Psychology of Computer Vision,* P. H. Winston, editor, McGraw-Hill Company.

Moore, R. C. 1982. "The Role of Logic in Knowledge Representation and Commonsense Reasoning," *Proceedings of AAAI-82*, pp. 428-433.

Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P., Faloutsos, C. and Taubin, G. 1993. "The QBIC Project: Querying Images by Content Using Color, Texture and Shape," IBM Research Report, RJ 9203 (81511).

Norvig, P. 1994. Personal communication.

Orwant, J. 1991. "The Doppelganger User Modeling System," in *Proceedings of the 1991 IJCAI Workshop on Modeling of Intelligent Interaction,* Sydney, Australia.

Pentland, A., Picard, R. W., and Sclaroff, S. 1994. "Photobook: Tools for Content-Based Manipulation of Image Databases," MIT Media Laboratory Perceptual Computing TR No. 255.

Quillian, M. R. 1968. "Semantic Memory," in *Semantic Information Processing*, M. L. Minsky, editor, MIT Press.

Rabitti, F., and Savino, P. 1991. "Automatic Image Indexation and Retrieval," in *Proceedings of the Conference on Intelligent Text and Image Handling (RIAO '91)*, Barcelona.

Ravin, Y. 1990. "Disambiguating and Interpreting Verb Definitions," in *Proceedings of ACL-90.*

Resnik, P. 1992. "A Class-Based Approach to Lexical Discovery," in *Proceedings of ACL-92 student session*, pp. 327-329.

Rowe, N. C. 1994. "Domain-independent Rules Relating Captions and Pictures," in *Proceedings of the AAAI-94 Workshop on Integration of Natural Language and Vision Processing,* Seattle, Washington.

Salton, G. 1989. *Automatic Text Processing*, Addison-Wesley Publishing Company, Reading, MA.

Sanderson, M. 1994. "Word Sense Disambiguation and Information Retrieval," in *Proceedings of ACM SIGIR'94.*

Schank, R. C., and Abelson, R. 1977. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Schecter, G. 1966. (editor) *Information Retrieval: A Critical View; Proceedings of the Third National Colloquium on Information Retrieval*, Philadelphia, Thomson and Co., Washington.

Schwartz, M. F., Emtage, A., Kahle, B., and Neuman, B. C. 1992. "A Comparison of Internet Resource Discovery Approaches," in *Computing Systems*, **5**(4).

Schwartz, M. F. and Tsirigotis, P. G. 1991. "Experience with a Semantically Cognizant Internet White Pages Directory Tool," in *Journal of Internetworking: Research and Experience*, **2**(1), pp. 23-50.

Shapiro, S. C. 1992. (editor) *Encyclopedia of Artificial Intelligence,* John Wiley and Sons, Inc., New York, pp. 1225-1226.

Sharp, J. R. 1965. *Some Fundamentals of Information Retrieval*, London House and Maxwell, New York.

Small, S., and Rieger, C. 1982. "Parsing and Comprehending with Word Experts (A Theory and its Realization)," in *Strategies for Natural Language Processing*, W. Lehnert and M. Ringle (eds), Lawrence Erlbaum Associates, Hillsdale, NJ.

Smeaton, A. F. 1992. "Progress in the Application of Natural Language Processing to Information Retrieval Tasks," in *The Computer Journal*, **35**(3).

Srihari, R. K. 1994. "Photo Understanding using Visual Constraints Generated from Accompanying Text," in *Proceedings of the AAAI-94 Workshop on Integration of Natural Language and Vision Processing,* Seattle, Washington.

Sundheim, B. (ed). 1993. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Morgan Kaufman Publishers, Inc.

Turk, M. and Pentland, A. 1991. "Eigenfaces for Recognition," in *Journal of Cognitive Neuroscience*, **3**(1), pp. 71-86.

Turtle, H. 1994. "Natural Language vs. Boolean Query Evaluation: A Comparison of Retrieval Performance," in *Proceedings of ACM SIGIR'94*, Dublin, Ireland.

Vanderwende, L. 1994. "Algorithm for Automatic Interpretation of Noun Sequences," in *Proceedings of COLING-94,* Kyoto, Japan.

Veronica. 1993. Internet Manuscript, URL **gopher://veronica.scs.unr.edu/11/veronica**

Veronis, J., and Ide, N. M. 1990. "Word-sense Disambiguation using Neural Networks Extracted from Machine Readable Dictionaries," in *Proceedings of COLING-90.*

Veronis, J. and Ide, N. M. 1991. "An Assessment of Semantic Information Automatically Extracted from Machine Readable Dictionaries," in *Proceedings of EACL-91*, pp. 227-232.

Vickery, B. C. 1965. *On Retrieval System Theory,* 2nd edition, Butterworths, Washington.

Voorhees, E. M. and Hou, Y-W. 1991. "Vector Expansion in a Large Collection," in *Proceedings of the First Text Retrieval Conference (TREC-1).*

Voorhees, E. M. 1993. "Using WordNet to Disambiguate Word Senses for Text Retrieval," in *Proceedings of SIGIR'93.*

Voorhees, E. M. 1994. "Query Expansion Using Lexical-Semantic Relations," in *Proceedings of SIGIR'94.*

Waltz, D. L., and Boggess, L. 1979. “Visual Analog Representations for Natural Language Understanding,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 926-934.

Weitzman, L. M. 1995. *The Architecture of Information: Interpretation and Presentation of Information in Dynamic Environments*, Ph.D thesis, MIT Media Laboratory, Cambridge, MA.

Winograd, T., and Flores, F. 1986. *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publication Corporation, Norwood, NJ.

Woods, W. A. 1975. “What’s in a Link: Foundations for Semantic Networks,” in *Representation and Understanding: Studies in Cognitive Science,* D. G. Bobrow and A. M. Collins (editors), Academic Press, New York.

Yanoff, S. 1993. *Internet Services List.* Internet Manuscript available through ‘finger yanoff@csd4.csd.uwm.edu’

Yarowsky, D. 1992. “Word Sense Disambiguation Using Statistical Models of Roget’s Categories Trained on Large Corpora,” in *Proceedings of COLING-92*, Nantes, France.

CHAPTER 8

Glossary

Adjacency Relationship: A relationship between two words that is inferred automatically from text and is used in disambiguation (Section 5.2)

Case Structure Representation: A representation that identifies the salient words in a sentence and their roles and forms <role, word> pairs (Section 2.2.3)

Child Pair: A <role, word> pair in a case structure representation that modifies (in a linguistic sense) a parent pair (Section 4.2)

Disambiguation: A process in which the meaning in which a word is used is inferred from context (Chapter 5)

Distance: A measure of the semantic similarity or dissimilarity of two synsets (Section 3.4 and Section 4.4.2)

Holonym: WordNet terminology for the HAS-PART semantic relation, e.g., [table] has the holonym [leg]. (Section 2.3.1)

Hypernym: WordNet terminology for the A-KIND-OF semantic relation, e.g., [mortar, howitzer] has the hypernym [cannon]. (Section 2.3.1)

Hypernym-match: A <role, word> pair in a query representation that can be generalized to a <role, word> pair in an archive representation (Section 4.4.1)

Hyponym: WordNet terminology for the inverse of the A-KIND-OF semantic relation, e.g., [cannon] has the hyponym [mortar, howitzer]. (Section 2.3.1)

ImEngine: A program built by the author that retrieves pictures and video clips through caption analysis (Chapter 3)

Information Access: The process of identifying Internet information archives that might be relevant to a user's query (Section 1.1)

Internet Hunt: A game that tests players's knowledge of the Internet (Section 4.3)

Meronym: WordNet terminology for the PART-OF semantic relation, e.g., [leg] has the meronym [table]. (Section 2.3.1)

NetSerf: A program built by the author that identifies Internet information archives that might be relevant to a user's query (Chapter 4)

Parent Pair: A <role, word> pair in a case structure representation that is modified (in a linguistic sense) by a child pair (Section 4.2)

Part-of-Speech Tagging: A process in which the part of speech in which a word is used is inferred from context (Section 3.2.1)

Pertainym: A semantic relation in WordNet linking an adjective to its corresponding noun form; e.g., [Indian] $\rightarrow$ [India, Bharat]. (Section 2.3.1)

Precision: In an information retrieval system, the percentage of relevant text units among all the retrieved text units (Section 2.2.1)

Recall: In an information retrieval system, the percentage of all relevant text units in the database that are actually retrieved by the system (Section 2.2.1)

Relation-Type: Another term for the "role" played by a word in a case structure representation (Section 2.2.3)

SMART: A standard, keyword-based text retrieval system (Section 2.2.1)

Success Weight: Weight that is added to a match if a matching rule is successful (Section 3.4)

Synset: A set of synonyms in WordNet, e.g., [mortar, howitzer]. (Section 2.3.1)

Tagging: See Part-of-Speech Tagging.

WordNet: A large, manually-constructed semantic network (Section 2.3.1)