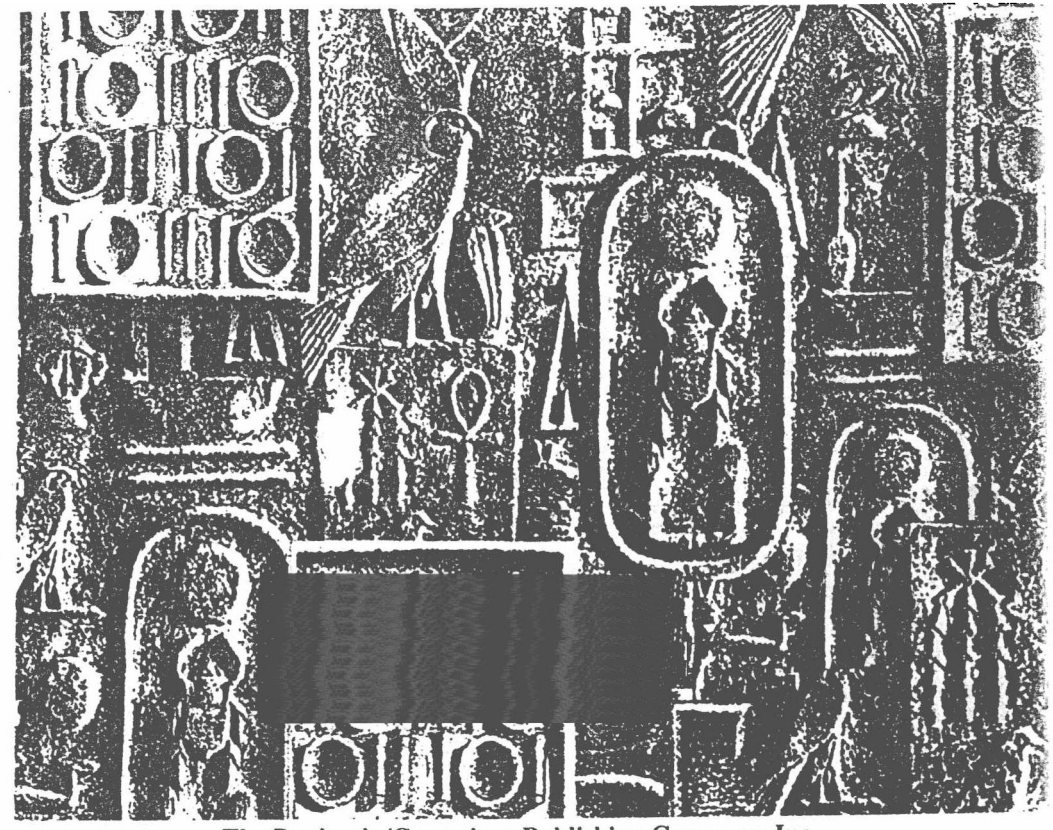
NATURAL LANGUAGE UNDERSTANDING

JAMES ALLEN

SECOND EDITION

Natural Language Understanding

James Allen
University of Rochester



The Benjamin/Cummings Publishing Company, Inc.
Redwood City, California • Menlo Park, California • Reading, Massachusetts
New York · Don Mills, Ontarto Wokingham, U.K. · Amsterdam · Bonn
Sydney · Singapore · Tokyo · Madrid · San Juan

Brief Contents

Acquisitions Editor: J. Carter Shanklin
Executive Editor: Dan Joraanstad
Editorial Assistant: Melissa Standen
Cover Designer: Yvo Riezebos Design
Technical Assistant: Peg Meeker
Production Editor: Ray Kanarr
Copy Editor: Barbara Conway
Proofreader: Joe Ruddick
Design Consultant: Michael Rogondino

Macintosh is a trademark of Apple Computer, Inc. Word is a trademark of Microsoft, Inc. Canvas is a trademark of Deneba Software.

Camera-ready copy for this book was prepared on a Macintosh with Microsoft Word and Canvas.

Library of Congress Cataloging-in-Publication Data

```
Allen, James.
    Natural language understanding / James Allen. -- 2nd ed.
       p.   cm.
    Includes bibliographical references and index.
    ISBN 0-8053-0334-0
    1. Programming languages (Electronic computers)--Semantics.
  2. Language and logic.  3. Artificial intelligence.   I. Title.
  QA76.7.A44  1994
  006.3'6--dc20                                                  94-18218
                                                                   CIP
```

2 3 4 5 6 7 8 9 10 – MA – 98 97 96 95

The Benjamin/Cummings Publishing Company, Inc.
390 Bridge Parkway
Redwood City, CA 94065

Contents

(◦—optional topic)

Preface

The primary goal of this book is to provide a comprehensive, in-depth description of the theories and techniques used in the field of natural language understanding. To do this in a single book requires eliminating the idiosyncratic complexities of particular approaches and identifying the underlying concepts of the field as a whole. It also requires developing a small number of notations that can be used to describe a wide range of techniques. It is not possible to capture the range of specific notations that are used in the literature, and attempting to do so would mask the important ideas rather than illuminate them. This book also attempts to make as few assumptions about the background of the reader as possible. All issues and techniques are described in plain English whenever possible. As a result, any reader having some familiarity with the basic notions of programming will be able to understand the principal ideas and techniques used. There is enough detail in this book, however, to allow a sophisticated programmer to produce working systems for natural language understanding.

The book is intended both as textbook and as a general reference source for those interested in natural language processing. As a text, it can be used both in undergraduate and graduate courses in computer science or computational linguistics. As a reference source, it can be used by researchers in areas related to natural language, by developers building natural language systems, and by individuals who are simply interested in how computers can process language.

Work in natural language understanding requires background in a wide range of areas, most importantly computer science, linguistics, logic, psycholinguistics, and the philosophy of language. As very few people have all this background, this book introduces whatever background material is required, as needed. For those who have no familiarity with symbolic programming or logical formalisms, two appendices are provided that describe enough of the basic ideas to enable nonprogrammers to read the book. Background in linguistics and other areas is introduced as needed throughout the book.

The book is organized by problem area rather than by technique. This allows the overlap in different approaches to be presented once, and facilitates the comparison of the different techniques. For example, rather than having a chapter on context-free grammars, one on transition network grammars, and another on logic programming techniques, these three techniques are discussed in many chapters, each focusing on a specific set of problems. There is a chapter on basic parsing techniques, then another on using features, and another on handling movement phenomena in language. Each chapter lays out its problem, and then discusses how the problem is addressed in each approach. This way, the similarities and differences between the approaches can be clearly identified.

Why a Second Edition?

I have taught a course based on the first edition regularly since it first appeared. Given developments in the field, I began to find it lacking in some areas. In other areas the book had the right content, but not the right emphasis. Another motivation came from what I saw as a potential split in the field with the introduction of statistically-based techniques and corpus-based research. Claims have been made as to how the new techniques make the old obsolete. But on examining the issues carefully, it was clear to me that the old and new approaches are complementary, and neither is a substitute for the other. This second edition attempts to demonstrate this by including new material on statistical methods, with an emphasis on how they interact with traditional schemes.

Another motivation was to improve the pedagogical aspects of the book. While the original edition was intended for use by students and researchers at different levels of background and with different needs, its organization sometimes made this difficult. The second edition addresses this problem in several ways, as described below in the section on using this book. As a result, it should be accessible to students with little background—say, undergraduates in a computer science program with little linguistics background, or undergraduates in a linguistics program with only a basic course in programming—yet also comprehensive enough for more advanced graduate-level courses, and for use as a general reference text for researchers in the field.

There are many other changes throughout the book, reflecting how the field has changed since the first edition, or at least reflecting how my views have changed. Much of this is a change in emphasis. The sections on syntax, for instance, now use feature-based context-free grammars as the primary formalism for developing the ideas, rather than augmented transition networks. This better reflects current work in the field, and also allows for a better integration of insights from linguistics. Transition network formalisms are still covered in detail, however, as they underlie much work in the field. The sections on semantics have also changed substantially, focusing more on underlying principles and issues rather than specific computational techniques. The logical form language was updated to cover more complex sentences, and the primary method of semantic interpretation is a compositional rule-by-rule analysis. This allows for both a better discussion of the underlying problems in semantic interpretation and a better integration of insights from linguistics. New material has also been added on domain-specific semantic interpretation techniques that have been an active focus in practical systems in the last few years. The material on contextual processing and discourse has generally been rewritten and updated to reflect new work and my better understanding of the principles underlying the techniques in the literature.

The other significant change is the development of software to accompany the book. Most of the principal examples and algorithms have been implemented and tested. This code is available so that students can run the examples to get a better understanding of the algorithms. It also provides a strong base from which students can extend algorithms and develop new grammars for assignments. The code also helps to clarify and simplify the discussion of the algorithms in the book, both because they can be more precise, being based on running programs, and because implementation details can be omitted from the text and still be available to the student in the code.

How to Use This Book

To cover the entire book in detail would require a two-semester sequence at the undergraduate level. As a graduate seminar, it is possible to move more quickly and cover most of the book in a single semester. The most common use of this book, however, is as an undergraduate-level course for a single semester. The book is designed to make this possible, and to allow the instructor to choose the areas on which to focus. There are several mechanisms to help customize the book to your needs. First, each chapter is divided into core material and optional material. Optional sections are marked with an open dot (o). In general, if you are using a chapter, all of the core material should be covered. The optional material can be selected as best fits the design of your course. Additional optional material is included in boxes. Material in boxes is never required, but can be used to push a little deeper in some areas.

The chapters themselves have the general dependencies shown in the figure on the next page. This means that the student should cover the core material in the earlier chapters to best understand the later chapters. In certain cases, material in a section may require knowledge of a technique discussed earlier but not indicated in the dependencies shown here. For example, one of the techniques discussed in Chapter 11 uses some partial parsing techniques described in Chapter 6. In such a case, the earlier section will need to be covered even though the chapter as a whole is skipped.

Readers familiar with the first edition will notice that the two final chapters, dealing with question answering and generation, are not present in the second edition. In keeping with the organization by topic rather than technique, the material from these chapters has been integrated into other chapters. Sections dealing with generation appear as each topic area is discussed. For example, a head-driven realization algorithm is described in Chapter 9 on linking syntax and semantics, and the procedural semantics technique used in database query systems appears in Chapter 13 on knowledge representation techniques.

Software

The software accompanying this book is available via anonymous ftp from bc.aw.com in the subdirectory "bc/allen." All software is written in standard COMMON LISP, and should run with any COMMON LISP system. To retrieve the software, ftp to bc.aw.com by typing

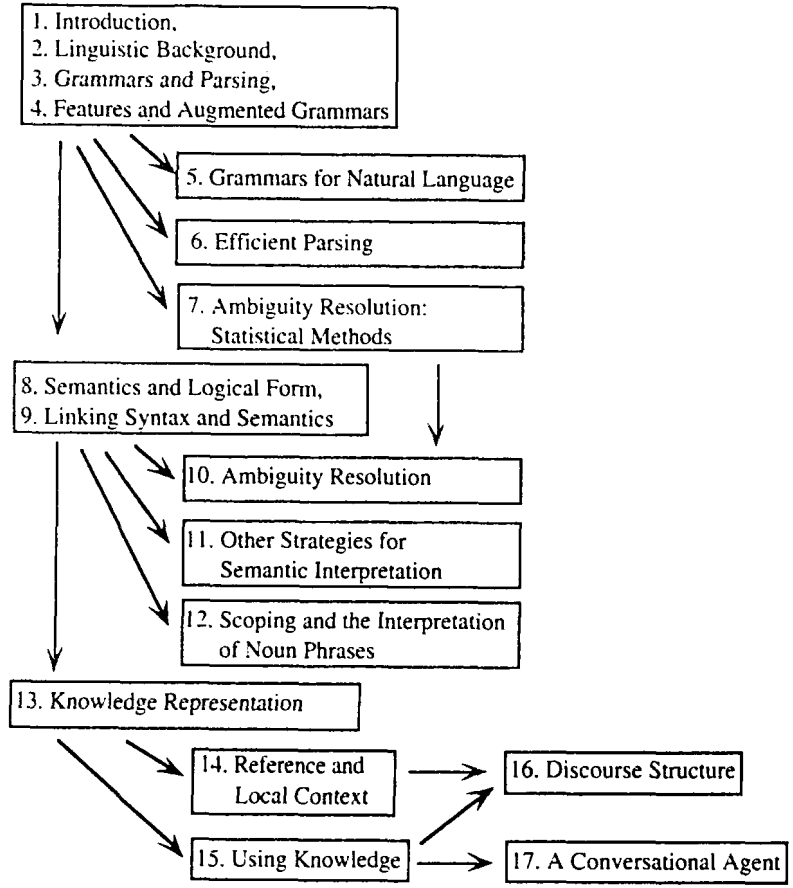
ftp bc.aw.com

and log in as "anonymous." Change to the directory for this book by typing

cd bc/allen

Before retrieving the files, it is a good idea to look at the "readme" file to see if changes have been made since this book went to press. To retrieve this file, type

get README



Quit ftp to log off and read the file. (Although the file can be read online, it is courteous not to tie up the login for reading.) Log back on when you are ready to download. You can also get a listing of filenames available using either the conventional UNIX "ls" command or the DOS "dir" command. Assuming that you are in the proper subdirectory, these commands require no arguments:

dir or ls

Using ftp and then de-archiving files can get complicated. Instructions vary as to whether you are downloading Macintosh, DOS, or UNIX files. More instructions are included in the README file. If you are new to using ftp, it is best to consult your favorite UNIX guide or wizard.

References

This book contains extensive references to literature in this field, so that the particular details of the techniques described here can easily be found. In addition, references are provided to other key papers that deal with issues beyond the scope of the book to help identify sources for research projects. References are chosen based on their general availability. Thus I have tried to confine citations to journal articles, books, and major conferences, avoiding technical reports and unpublished notes that are hard to find ten years after the fact. In general, most work in the field is published first in technical reports; more accessible publications often do not appear until many years after the ideas were introduced. Thus the year of publication in the references is not a reliable indicator of the time when the work was done, or when it started to influence other work in the field. While I've tried to provide a wide range of sources and to give credit to the ideas described in this book, I know I have omitted key papers that I will regret immensely when I remember them. To these authors, I extend my apologies.

Acknowledgments

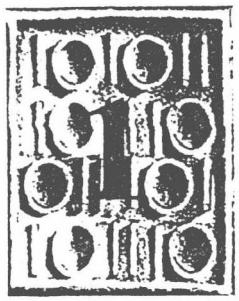
This book would not have been completed except for a sabbatical from the University of Rochester, and for the continual support of the Computer Science Department over the last three years. I was provided with the extra resources needed to complete the project without question, and I am very grateful. I also must thank Peggy Meeker for her efforts in editing and preparing endless drafts as well as the final pages you see here. Peggy has had amazing patience and good humor through countless revisions, and has enforced a consistency of notation and grammatical style that appears to be foreign to my natural way of writing!

I have been fortunate to have a wide range of reviewers on various drafts, both those working for the publisher and many other friends who have taken much time to read and comment on drafts. The list is long, and each one has devoted such effort that listing their names hardly seems sufficient. But here goes. Many thanks to Glenn Blank, Eugene Charniak, Michelle Degraff, George Ferguson, Mary Harper, Peter Heeman, Chung-Hee Hwang, Susan McRoy, Brad Miller, Johanna Moore, Lynn Peterson, Victor Raskin, Len Schubert, Mark Steedman, Mike Tanenhaus, David Traum, and the students in CSC 447 who used the book in the Spring of 1994. The book is significantly different and improved from the early drafts because of all the honest and critical feedback I received.

Finally, I must thank my wife, Judith Hook, for putting up with all this for a second time after experiencing the trauma of doing the first edition. Pushing the book to completion became an obsession that occupied me day and night for nearly a year. Thanks to Judi, and to Jeffrey, Daniel, and Michael, for putting up with me during this time and keeping an appropriate sense of humor about it.

James F. Allen
Rochester, New York

CHAPTER



Introduction to Natural Language Understanding

This chapter describes the field of natural language understanding and introduces some basic distinctions. Section 1.1 discusses how natural language understanding research fits into the study of language in general. Section 1.2 discusses some applications of natural language understanding systems and considers what it means for a system to understand language. Section 1.3 describes how you might evaluate whether a system understands language. Section 1.4 introduces a few basic distinctions that are made when studying language, and Section 1.5 discusses how computational systems often realize these distinctions. Finally, Section 1.6 discusses how natural language systems are generally organized, and introduces the particular organization assumed throughout this book.

The Study of Language

Language is one of the fundamental aspects of human behavior and is a crucial component of our lives. In written form it serves as a long-term record of knowledge from one generation to the next. In spoken form it serves as our primary means of coordinating our day-to-day behavior with others. This book describes research about how language comprehension and production work. The goal of this research is to create computational models of language in enough detail that you could write computer programs to perform various tasks involving natural language. The ultimate goal is to be able to specify models that approach human performance in the linguistic tasks of reading, writing, hearing, and speaking. This book, however, is not concerned with problems related to the specific medium used, whether handwriting, keyboard input, or speech. Rather, it is concerned with the processes of comprehending and using language once the words are recognized. Computational models are useful both for scientific purposes—for exploring the nature of linguistic communication—and for practical purposes—for enabling effective human-machine communication.

Language is studied in several different academic disciplines. Each discipline defines its own set of problems and has its own methods for addressing them. The linguist, for instance, studies the structure of language itself, considering questions such as why certain combinations of words form sentences but others do not, and why a sentence can have some meanings but not others. The psycholinguist, on the other hand, studies the processes of human language production and comprehension, considering questions such as how people identify the appropriate structure of a sentence and when they decide on the appropriate meaning for words. The philosopher considers how words can mean anything at all and how they identify objects in the world. Philosophers also consider what it means to have beliefs, goals, and intentions, and how these cognitive capabilities relate to language. The goal of the computational linguist is to develop a computational theory of language, using the notions of algorithms and data structures from computer science. Of course, to build a computational model, you must take advantage of what is known from all the other disciplines. Figure 1.1 summarizes these different approaches to studying language.

Discipline	Typical Problems	Tools
Linguists	How do words form phrases and sentences? What constrains the possible meanings for a sentence?	Intuitions about well-formedness and meaning; mathematical models of structure (for example, formal language theory, model theoretic semantics)
Psycholinguists	How do people identify the structure of sentences? How are word meanings identified? When does understanding take place?	Experimental techniques based on measuring human performance; statistical analysis of observations
Philosophers	What is meaning, and how do words and sentences acquire it? How do words identify objects in the world?	Natural language argumentation using intuition about counter-examples; mathematical models (for example, logic and model theory)
Computational Linguists	How is the structure of sentences identified? How can knowledge and reasoning be modeled? How can language be used to accomplish specific tasks?	Algorithms, data structures; formal models of representation and reasoning; AI techniques (search and representation methods)

Figure 1.1 The major disciplines studying language

As previously mentioned, there are two motivations for developing computational models. The scientific motivation is to obtain a better understanding of how language works. It recognizes that any one of the other traditional disciplines does not have the tools to completely address the problem of how language comprehension and production work. Even if you combine all the approaches, a comprehensive theory would be too complex to be studied using traditional methods. But we may be able to realize such complex theories as computer programs and then test them by observing how well they perform. By seeing where they fail, we can incrementally improve them. Computational models may provide very specific predictions about human behavior that can then be explored by the psycholinguist. By continuing in this process, we may eventually acquire a deep understanding of how human language processing occurs. To realize such a dream will take the combined efforts of linguists, psycholinguists, philosophers, and computer scientists. This common goal has motivated a new area of interdisciplinary research often called cognitive science.

The practical, or technological, motivation is that natural language processing capabilities would revolutionize the way computers are used. Since most of human knowledge is recorded in linguistic form, computers that could understand natural language could access all this information. In addition, natural language interfaces to computers would allow complex systems to be accessible to

BOX 1.1 Boxes and Optional Sections

This book uses several techniques to allow you to identify what material is central and what is optional. In addition, optional material is sometimes classified as advanced, indicating that you may need additional background not covered in this book to fully appreciate the text. Boxes, like this one, always contain optional material, either providing more detail on a particular approach discussed in the main text or discussing additional issues that are related to the text. Sections and subsections may be marked as optional by means of an open dot (◦) before the heading. Optional sections provide more breadth and depth to chapters, but are not necessary for understanding material in later chapters. Depending on your interests and focus, you can choose among the optional sections to fill out the core material presented in the regular sections. In addition, there are dependencies between the chapters, so that entire chapters can be skipped if the material does not address your interests. The chapter dependencies are not marked explicitly in the text, but a chart of dependencies is given in the preface.

everyone. Such systems would be considerably more flexible and intelligent than is possible with current computer technology. For technological purposes it does not matter if the model used reflects the way humans process language. It only matters that it works.

This book takes a middle ground between the scientific and technological goals. On the one hand, this reflects a belief that natural language is so complex that an *ad hoc* approach without a well-specified underlying theory will not be successful. Thus the technological goal cannot be realized without using sophisticated underlying theories on the level of those being developed by linguists, psycholinguists, and philosophers. On the other hand, the present state of knowledge about natural language processing is so preliminary that attempting to build a cognitively correct model is not feasible. Rather, we are still attempting to construct any model that appears to work.

The goal of this book is to describe work that aims to produce linguistically motivated computational models of language understanding and production that can be shown to perform well in specific example domains. While the book focuses on computational aspects of language processing, considerable space is spent introducing the relevant background knowledge from the other disciplines that motivates and justifies the computational approaches taken. It assumes only a basic knowledge of programming, although the student with some background in linguistics, artificial intelligence (AI), and logic will appreciate additional subtleties in the development.

.2 Applications of Natural Language Understanding

A good way to define natural language research is to consider the different applications that researchers work on. As you consider these examples, it will

also be a good opportunity to consider what it would mean to say that a computer system understands natural language. The applications can be divided into two major classes: text-based applications and dialogue-based applications.

Text-based applications involve the processing of written text, such as books, newspapers, reports, manuals, e-mail messages, and so on. These are all reading-based tasks. Text-based natural language research is ongoing in applications such as

- finding appropriate documents on certain topics from a database of texts (for example, finding relevant books in a library)
- extracting information from messages or articles on certain topics (for example, building a database of all stock transactions described in the news on a given day)
- translating documents from one language to another (for example, producing automobile repair manuals in many different languages)
- summarizing texts for certain purposes (for example, producing a 3-page summary of a 1000-page government report)

Not all systems that perform such tasks must be using natural language understanding techniques in the way we mean in this book. For example, consider the task of finding newspaper articles on a certain topic in a large database. Many techniques have been developed that classify documents by the presence of certain keywords in the text. You can then retrieve articles on a certain topic by looking for articles that contain the keywords associated with that topic. Articles on law, for instance, might contain the words *lawyer, court, sue, affidavit*, and so on, while articles on stock transactions might contain words such as *stocks, takeover, leveraged buyout, options*, and so on. Such a system could retrieve articles on any topic that has been predefined by a set of keywords. Clearly, we would not say that this system is understanding the text; rather, it is using a simple matching technique. While such techniques may produce useful applications, they are inherently limited. It is very unlikely, for example, that they could be extended to handle complex retrieval tasks that are easily expressed in natural language, such as the query *Find me all articles on leveraged buyouts involving more than 100 million dollars that were attempted but failed during 1986 and 1990.* To handle such queries, the system would have to be able to extract enough information from each article in the database to determine whether the article meets the criteria defined by the query; that is, it would have to build a representation of the information in the articles and then use the representation to do the retrievals. This identifies a crucial characteristic of an understanding system: it must compute some representation of the information that can be used for later inference.

Consider another example. Some machine translation systems have been built that are based on pattern matching; that is, a sequence of words in one language is associated with a sequence of words in another language. The

translation is accomplished by finding the best set of patterns that match the input and producing the associated output in the other language. This technique can produce reasonable results in some cases but sometimes produces completely wrong translations because of its inability to use an understanding of content to disambiguate word senses and sentence meanings appropriately. In contrast, other machine translation systems operate by producing a representation of the meaning of each sentence in one language, and then producing a sentence in the other language that realizes the same meaning. This latter approach, because it involves the computation of a representation of meaning, is using natural language understanding techniques.

One very attractive domain for text-based research is story understanding. In this task the system processes a story and then must answer questions about it. This is similar to the type of reading comprehension tests used in schools and provides a very rich method for evaluating the depth of understanding the system is able to achieve.

Dialogue-based applications involve human-machine communication. Most naturally this involves spoken language, but it also includes interaction using keyboards. Typical potential applications include

- question-answering systems, where natural language is used to query a database (for example, a query system to a personnel database)
- automated customer service over the telephone (for example, to perform banking transactions or order items from a catalogue)
- tutoring systems, where the machine interacts with a student (for example, an automated mathematics tutoring system)
- spoken language control of a machine (for example, voice control of a VCR or computer)
- general cooperative problem-solving systems (for example, a system that helps a person plan and schedule freight shipments)

Some of the problems faced by dialogue systems are quite different than in text-based systems. First, the language used is very different, and the system needs to participate actively in order to maintain a natural, smooth-flowing dialogue. Dialogue requires the use of acknowledgments to verify that things are understood, and an ability to both recognize and generate clarification subdialogues when something is not clearly understood. Even with these differences, however, the basic processing techniques are fundamentally the same.

It is important to distinguish the problems of speech recognition from the problems of language understanding. A speech recognition system need not involve any language understanding. For instance, voice-controlled computers and VCRs are entering the market now. These do not involve natural language understanding in any general way. Rather, the words recognized are used as commands, much like the commands you send to a VCR using a remote control. Speech recognition is concerned only with identifying the words spoken from a given speech signal, not with understanding how words are used to communicate. To be an understanding system, the speech recognizer would need to feed its input to a natural language understanding system, producing what is often called a spoken language understanding system.

With few exceptions, all the techniques discussed in this book are equally relevant for text-based and dialogue-based language understanding, and apply equally well whether the input is text, keyboard, or speech. The key characteristic of any understanding system is that it represents the meaning of sentences in some representation language that can be used later for further processing.

1.3 Evaluating Language Understanding Systems

As you can see, what counts as understanding might vary from application to application. If this is so, how can you tell if a system works? One obvious way to evaluate a system is to run the program and see how well it performs the task it was designed to do. If the program is meant to answer questions about a database of facts, you might ask it questions to see how good it is at producing the correct answers. If the system is designed to participate in simple conversations on a certain topic, you might try conversing with it. This is called **black box evaluation** because it evaluates system performance without looking inside to see how it works. While ultimately this method of evaluation may be the best test of a system's capabilities, it is problematic in the early stages of research because early evaluation results can be misleading. Sometimes the techniques that produce the best results in the short term will not lead to the best results in the long term. For instance, if the overall performance of all known systems in a given application is uniformly low, few conclusions can be drawn. The fact that one system was correct 50 percent of the time while another was correct only 40 percent of the time says nothing about the long-term viability of either approach. Only when the success rates become high, making a practical application feasible, can much significance be given to overall system performance measures.

An alternative method of evaluation is to identify various subcomponents of a system and then evaluate each one with appropriate tests. This is called **glass box evaluation** because you look inside at the structure of the system. The problem with glass box evaluation is that it requires some consensus on what the various components of a natural language system should be. Achieving such a consensus is an area of considerable activity at the present.

Despite the difficulties, evaluations are of crucial importance to the field, as very little can be concluded from examining a few examples that a particular system can handle. If you cannot interact freely with the program yourself but must depend on transcripts of others' interactions, you can easily be mislead into overestimating a program's abilities. It is very hard for people not to attribute much more intelligence to the program than is actually present.

Consider a concrete example—the ELIZA program that was developed in the mid-1960s at MIT. This program must be one of the most popular AI

User:	ELIZA:
Men are all alike.	In what way?
They're always bugging us about something or other.	Can you think of a specific example?
Well, my boyfriend made me come here.	Your boyfriend made you come here.
He says I'm depressed much of the time.	I am sorry to hear you are depressed.
It's true, I am unhappy.	Do you think coming here will help you not to be unhappy?
I need some help; that much seems certain.	What would it mean to you if you got some help?
Perhaps I could learn to get along with my mother.	Tell me more about your family.
My mother takes care of me.	Who else in your family takes care of you?
My father.	Your father.
You are like my father in some ways.	What resemblance do you see?

Figure 1.2 A dialogue with ELIZA

programs of all time, and versions of it exist for most machines, including most personal computers. ELIZA was never claimed to embody a theory of language comprehension and production, but it serves as an excellent example as its behavior initially seems impressive. The system plays the role of a therapist and, to obtain the best results, the user should correspondingly play the role of a patient. Figure 1.2 presents a transcript of this system in operation. Given this transcript, or even playing with the system yourself for a few minutes, ELIZA's performance certainly seems impressive.

Here is a simple description of how ELIZA works. There is a database of particular words that are called **keywords**. For each keyword, the system stores an integer, a pattern to match against the input, and a specification of the output. The algorithm is as follows: Given a sentence S, find a keyword in S whose pattern matches S. If there is more than one keyword, pick the one with the highest integer value. Use the output specification that is associated with this keyword to generate the next sentence. If there are no keywords, generate an innocuous continuation statement, such as *Tell me more* or *Go on.*

Figure 1.3 shows a fragment of a database of keywords. In this database a pattern consists of words and variables. The prefix ? before a letter indicates a variable, which can match any sequence of words. For example, the pattern

?X are you ?Y

would match the sentence *Why are you looking at me?*, where the variable ?X matches *Why* and ?Y matches *looking at me.* The output specification may also use the same variables. In this case, ELIZA inserts the words that match the variables in the input into the output after making some minor changes in the

Word	Rank	Pattern	Outputs
alike	10	?X	In what way? What resemblance do you see?
are	3	?X are you ?Y	Would you prefer it if I weren't ?Y?
	3	?X are ?Y	What if they were not ?Y?
always	5	?X	Can you think of a specific example? When? Really, always?
what	2	?X	Why do you ask? Does that interest you?

Figure 1.3 Sample data from ELIZA

pronouns (for example, replacing *me* with *you*). Thus, for the pattern above, if the output specification is

Would you prefer it if I weren't ?Y?

the rule would generate a response *Would you prefer it if I weren't looking at you?* When the database lists multiple output specifications for a given pattern, ELIZA selects a different one each time a keyword rule is used, thereby preventing unnatural repetition in the conversation. Using these rules, you can see how ELIZA produced the first two exchanges in the conversation in Figure 1.2. ELIZA generated the first response from the first output of the keyword *alike* and the second response from the first output of the keyword *always.*

This description covers all of the essential points of the program. You will probably agree that the program does not understand the conversation it is participating in. Rather, it is a collection of tricks. Why then does ELIZA appear to function so well? There are several reasons. Perhaps the most important reason is that, when people hear or read a sequence of words that they understand as a sentence, they attribute meaning to the sentence and assume that the person (or machine) that produced the sentence actually intended that meaning. People are extremely good at distinguishing word meanings and interpreting sentences to fit the context. Thus ELIZA appears to be intelligent because you use your own intelligence to make sense of what it says.

Other crucial characteristics of the conversational setting also aid in sustaining the illusion of intelligence. For instance, the system does not need any world knowledge because it never has to make a claim, support an argument, or answer a question. Rather, it simply asks a series of questions. Except in a patient-therapist situation, this would be unacceptable. ELIZA evades all direct questions by responding with another question, such as *Why do you ask?* There is no way to force the program to say something concrete about any topic.

Even in such a restricted situation, however, it is relatively easy to demonstrate that the program does not understand. It sometimes produces completely off-the-wall responses. For instance, if you say *Necessity is the mother of invention*, it might respond with *Tell me more about your family*, based on its pattern for the word *mother*. In addition, since ELIZA has no knowledge about the structure of language, it accepts gibberish just as readily as valid sentences. If you enter *Green the adzabak are the a ran four*, ELIZA will respond with something like *What if they were not the a ran four?* Also, as a conversation progresses, it becomes obvious that the program does not retain any of the content in the conversation. It begins to ask questions that are inappropriate in light of earlier exchanges, and its responses in general begin to show a lack of focus. Of course, if you are not able to play with the program and must depend only on transcripts of conversations by others, you would have no way of detecting these flaws, unless they are explicitly mentioned.

Suppose you need to build a natural language program for a certain application in only six months. If you start to construct a general model of language understanding, it will not be completed in that time frame and so will perform miserably on the tests. An ELIZA-like system, however, could easily produce behavior like that previously discussed with less than a few months of programming and will appear to far outperform the other system in testing. The differences will be especially marked if the test data only includes typical domain interactions that are not designed to test the limits of the system. Thus, if we take short-term performance as our only criteria of progress, everyone will build and fine-tune ELIZA-style systems, and the field will not progress past the limitations of the simple approach.

To avoid this problem, either we have to accept certain theoretical assumptions about the architecture of natural language systems and develop specific evaluation measures for different components, or we have to discount overall evaluation results until some reasonably high level of performance is obtained. Only then will cross-system comparisons begin to reflect the potential for long-term success in the field.

1.4 The Different Levels of Language Analysis

A natural language system must use considerable knowledge about the structure of the language itself, including what the words are, how words combine to form sentences, what the words mean, how word meanings contribute to sentence meanings, and so on. However, we cannot completely account for linguistic behavior without also taking into account another aspect of what makes humans intelligent—their general world knowledge and their reasoning abilities. For example, to answer questions or to participate in a conversation, a person not only must know a lot about the structure of the language being used, but also must know about the world in general and the conversational setting in particular.

The following are some of the different forms of knowledge relevant for natural language understanding:

Phonetic and phonological knowledge—concerns how words are related to the sounds that realize them. Such knowledge is crucial for speech-based systems and is discussed in more detail in Appendix C.

Morphological knowledge—concerns how words are constructed from more basic meaning units called **morphemes**. A morpheme is the primitive unit of meaning in a language (for example, the meaning of the word *friendly* is derivable from the meaning of the noun *friend* and the suffix *-ly*, which transforms a noun into an adjective).

Syntactic knowledge—concerns how words can be put together to form correct sentences and determines what structural role each word plays in the sentence and what phrases are subparts of what other phrases.

Semantic knowledge—concerns what words mean and how these meanings combine in sentences to form sentence meanings. This is the study of context-independent meaning—the meaning a sentence has regardless of the context in which it is used.

Pragmatic knowledge—concerns how sentences are used in different situations and how use affects the interpretation of the sentence.

Discourse knowledge—concerns how the immediately preceding sentences affect the interpretation of the next sentence. This information is especially important for interpreting pronouns and for interpreting the temporal aspects of the information conveyed.

World knowledge—includes the general knowledge about the structure of the world that language users must have in order to, for example, maintain a conversation. It includes what each language user must know about the other user's beliefs and goals.

These definitions are imprecise and are more characteristics of knowledge than actual distinct classes of knowledge. Any particular fact might include aspects from several different levels, and an algorithm might need to draw from several different levels simultaneously. For teaching purposes, however, this book is organized into three parts, each describing a set of techniques that naturally cluster together. Part I focuses on syntactic and morphological processing, Part II focuses on semantic processing, and Part III focuses on contextual effects in general, including pragmatics, discourse, and world knowledge.

BOX 1.2 Syntax, Semantics, and Pragmatics

The following examples may help you understand the distinction between syntax, semantics, and pragmatics. Consider each example as a candidate for the initial sentence of this book, which you know discusses natural language processing:

1. Language is one of the fundamental aspects of human behavior and is a crucial component of our lives.
2. Green frogs have large noses.
3. Green ideas have large noses.
4. Large have green ideas nose.

Sentence 1 appears to be a reasonable start (I hope!). It agrees with all that is known about syntax, semantics, and pragmatics. Each of the other sentences violates one or more of these levels. Sentence 2 is well-formed syntactically and semantically, but not pragmatically. It fares poorly as the first sentence of the book because the reader would find no reason for using it. But however bad sentence 2 would be as a start, sentence 3 is much worse. Not only is it obviously pragmatically ill-formed, it is also semantically ill-formed. To see this, consider that you and I could argue about whether sentence 2 is true or not, but we cannot do so with sentence 3. I cannot affirm or deny sentence 3 in coherent conversation. However, the sentence does have some structure, for we can discuss what is wrong with it: Ideas cannot be green and, even if they could, they certainly cannot have large noses. Sentence 4 is even worse. In fact, it is unintelligible, even though it contains the same words as sentence 3. It does not even have enough structure to allow you to say what is wrong with it. Thus it is syntactically ill-formed. Incidentally, there are cases in which a sentence may be pragmatically well-formed but not syntactically well-formed. For example, if I ask you where you are going and you reply "I go store," the response would be understandable even though it is syntactically ill-formed. Thus it is at least pragmatically well-formed and may even be semantically well-formed.

.5 Representations and Understanding

As previously stated, a crucial component of understanding involves computing a representation of the meaning of sentences and texts. Without defining the notion of representation, however, this assertion has little content. For instance, why not simply use the sentence itself as a representation of its meaning? One reason is that most words have multiple meanings, which we will call **senses**. The word *cook*, for example, has a sense as a verb and a sense as a noun; *dish* has multiple senses as a noun as well as a sense as a verb; and *still* has senses as a noun, verb, adjective, and adverb. This ambiguity would inhibit the system from making the appropriate inferences needed to model understanding. The disambiguation problem appears much easier than it actually is because people do not generally notice ambiguity. While a person does not seem to consider each of the possible

senses of a word when understanding a sentence, a program must explicitly consider them one by one.

To represent meaning, we must have a more precise language. The tools to do this come from mathematics and logic and involve the use of formally specified representation languages. Formal languages are specified from very simple building blocks. The most fundamental is the notion of an atomic symbol, which is distinguishable from any other atomic symbol simply based on how it is written. Useful representation languages have the following two properties:

- The representation must be precise and unambiguous. You should be able to express every distinct reading of a sentence as a distinct formula in the representation.
- The representation should capture the intuitive structure of the natural language sentences that it represents. For example, sentences that appear to be structurally similar should have similar structural representations, and the meanings of two sentences that are paraphrases of each other should be closely related to each other.

Several different representations will be used that correspond to some of the levels of analysis discussed in the last section. In particular, we will develop formal languages for expressing syntactic structure, for context-independent word and sentence meanings, and for expressing general world knowledge.

Syntax: Representing Sentence Structure

The syntactic structure of a sentence indicates the way that words in the sentence are related to each other. This structure indicates how the words are grouped together into phrases, what words modify what other words, and what words are of central importance in the sentence. In addition, this structure may identify the types of relationships that exist between phrases and can store other information about the particular sentence structure that may be needed for later processing. For example, consider the following sentences:

1. John sold the book to Mary.
2. The book was sold to Mary by John.

These sentences share certain structural properties. In each, the noun phrases are *John*, *Mary*, and *the book*, and the act described is some selling action. In other respects, these sentences are significantly different. For instance, even though both sentences are always either true or false in the exact same situations, you could only give sentence 1 as an answer to the question *What did John do for Mary?* Sentence 2 is a much better continuation of a sentence beginning with the phrase *After it fell in the river*, as sentences 3 and 4 show. Following the standard convention in linguistics, this book will use an asterisk (*) before any example of an ill-formed or questionable sentence.

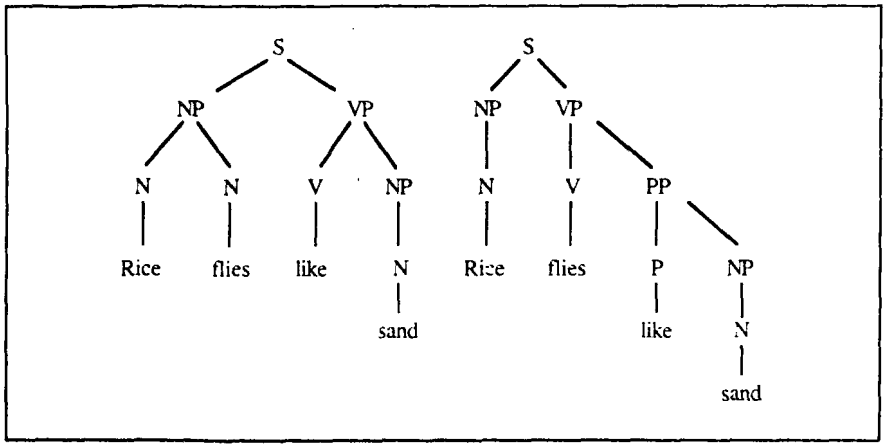


Figure 1.4 Two structural representations of *Rice flies like sand.*

3. *After it fell in the river, John sold Mary the book.
4. After it fell in the river, the book was sold to Mary by John.

Many other structural properties can be revealed by considering sentences that are not well-formed. Sentence 5 is ill-formed because the subject and the verb do not agree in number (the subject is singular and the verb is plural), while 6 is ill-formed because the verb *put* requires some modifier that describes where John put the object.

5. *John are in the corner.
6. *John put the book.

Making judgments on grammaticality is not a goal in natural language understanding. In fact, a robust system should be able to understand ill-formed sentences whenever possible. This might suggest that agreement checks can be ignored, but this is not so. Agreement checks are essential for eliminating potential ambiguities. Consider sentences 7 and 8, which are identical except for the number feature of the main verb, yet represent two quite distinct interpretations.

7. Flying planes are dangerous.
8. Flying planes is dangerous.

If you did not check subject-verb agreement, these two sentences would be indistinguishable and ambiguous. You could find similar examples for every syntactic feature that this book introduces and uses.

Most syntactic representations of language are based on the notion of context-free grammars, which represent sentence structure in terms of what phrases are subparts of other phrases. This information is often presented in a tree form, such as the one shown in Figure 1.4, which shows two different structures

for the sentence *Rice flies like sand.* In the first reading, the sentence is formed from a noun phrase (NP) describing a type of fly, rice flies, and a verb phrase (VP) that asserts that these flies like sand. In the second structure, the sentence is formed from a noun phrase describing a type of substance, rice, and a verb phrase stating that this substance flies like sand (say, if you throw it). The two structures also give further details on the structure of the noun phrase and verb phrase and identify the part of speech for each word. In particular, the word *like* is a verb (V) in the first reading and a preposition (P) in the second.

The Logical Form

The structure of a sentence doesn't reflect its meaning, however. For example, the NP *the catch* can have different meanings depending on whether the speaker is talking about a baseball game or a fishing expedition. Both these interpretations have the same syntactic structure, and the different meanings arise from an ambiguity concerning the sense of the word *catch*. Once the correct sense is identified, say the fishing sense, there still is a problem in determining what fish are being referred to. The intended meaning of a sentence depends on the situation in which the sentence is produced. Rather than combining all these problems, this book will consider each one separately. The division is between context-independent meaning and context-dependent meaning. The fact that *catch* may refer to a baseball move or the results of a fishing expedition is knowledge about English and is independent of the situation in which the word is used. On the other hand, the fact that a particular noun phrase *the catch* refers to what Jack caught when fishing yesterday is contextually dependent. The representation of the context-independent meaning of a sentence is called its **logical form.**

The logical form encodes possible word senses and identifies the semantic relationships between the words and phrases. Many of these relationships are often captured using an abstract set of semantic relationships between the verb and its NPs. In particular, in both sentences 1 and 2 previously given, the action described is a selling event, where *John* is the seller, *the book* is the object being sold, and *Mary* is the buyer. These roles are instances of the abstract semantic roles AGENT, THEME, and TO-POSS (for final possessor), respectively.

Once the semantic relationships are determined, some word senses may be impossible and thus eliminated from consideration. Consider the sentence

9. Jack invited Mary to the Halloween ball.

The word *ball*, which by itself is ambiguous between the plaything that bounces and the formal dance event, can only take the latter sense in sentence 9, because the verb *invite* only makes sense with this interpretation. One of the key tasks in semantic interpretation is to consider what combinations of the individual word meanings can combine to create coherent sentence meanings. Exploiting such