DOMAIN-SPECIFIC KNOWLEDGE ACQUISITION FOR CONCEPTUAL SENTENCE ANALYSIS

A Dissertation Presented

by

CLAIRE CARDIE

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1994

Department of Computer Science

DOMAIN-SPECIFIC KNOWLEDGE ACQUISITION
FOR CONCEPTUAL SENTENCE ANALYSIS

A Dissertation Presented

by

CLAIRE CARDIE

Approved as to style and content by:

____
Wendy G. Lehnert, Chair

____
Edwina L. Rissland, Member

____
Paul E. Utgoff, Member

____
Lyn Frazier, Member

____
W. Richards Adrion, Department Chair
Computer Science

ACKNOWLEDGMENTS

There are many people to whom I am grateful and without whom the thesis would have been almost impossible to write (much less finish):

First, Wendy Lehnert provided much inspiration both through her astounding energy and her insightful and informed views of natural language processing and artificial intelligence. I would also like to thank the other members of my dissertation committee for their help and guidance. Edwina Rissland helped me to view my work from a different perspective and (possibly unknowingly) offered encouragement and unbiased advice at critical points in my graduate school career. Paul Utgoff asked probing questions regarding some of the underlying assumptions of the work and left no "which" or "that" unturned. Illuminating discussions with Lyn Frazier improved the thesis tremendously. Special thanks also to Priscilla Coe (who actually volunteered to be on the committee). I am also indebted to Bob Futrelle (Northeastern University). His energetic and enthusiastic support of my work helped to increase my confidence in my own abilities.

Thanks too to my officemates in A249 for putting up with the humor vortex that hovered over Ren and that threatened the entire office: Jody Daniels, Fang-fang Feng, David Fisher, Timur Friedman, Joe McCarthy, Jonathan Peterson, and Stephen Soderland. Ellen "Betty" Riloff is missing from the above list only because she deserves a line of her own. Someday we'll really have to put together that slide show...

For countless softball games and practices, bicycle rides from hell, ice hockey on Cranberry pond, year-round volleyball, food snob dinners, fireworks, and all kinds of fun, I thank: Westy, Teri, Brian, Josh, Alan, Hildum, Jack, Keith, Bart, Dann, Jody, Ruth, Zack, Alice, Steve, Glo, Molly, Tommy, Kevin, Christy, Clarence, and Cleo. I hope I haven't forgotten anyone.

I would also like to acknowledge my parents for their support; my brothers and sisters — Bobby, David, Paul, Joanne, Meg, Peter, and Suzanne — for their lighthearted encouragement; and especially Aunt Claire, for always being so nice.

Most of all, I would like to thank David Bingham Skalak.

This research was supported by the National Science Foundation, the Advanced Research Projects Agency of the Department of Defense, and the Office of Naval Research.

ABSTRACT

DOMAIN-SPECIFIC KNOWLEDGE ACQUISITION FOR CONCEPTUAL SENTENCE ANALYSIS

SEPTEMBER 1994

CLAIRE CARDIE, B.S., YALE UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wendy G. Lehnert

The availability of on-line corpora is rapidly changing the field of natural language processing (NLP) from one dominated by theoretical models of often very specific linguistic phenomena to one guided by computational models that simultaneously account for a wide variety of phenomena that occur in real-world text. Thus far, among the best-performing and most robust systems for reading and summarizing large amounts of real-world text are knowledge-based natural language systems. These systems rely heavily on domain-specific, handcrafted knowledge to handle the myriad syntactic, semantic, and pragmatic ambiguities that pervade virtually all aspects of sentence analysis. Not surprisingly, however, generating this knowledge for new domains is time-consuming, difficult, and error-prone, and requires the expertise of computational linguists familiar with the underlying NLP system. This thesis presents Kenmore, a general framework for domain-specific knowledge acquisition for conceptual sentence analysis. To ease the acquisition of knowledge in new domains, Kenmore exploits an on-line corpus using symbolic machine learning techniques and robust sentence analysis while requiring only minimal human intervention. Unlike most approaches to knowledge acquisition for natural language systems, the framework uniformly addresses a range of subproblems in sentence analysis, each of which traditionally had required a separate computational mechanism. The thesis presents the results of using Kenmore with corpora from two real-world domains (1) to perform part-of-speech tagging, semantic feature tagging, and concept tagging of all open-class words in the corpus; (2) to acquire heuristics for part-of-speech disambiguation, semantic feature disambiguation, and concept activation; and (3) to find the antecedents of relative pronouns.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS iv

ABSTRACT v

LIST OF TABLES x

LIST OF FIGURES xiii

CHAPTER

1. INTRODUCTION 1

1.1 Ambiguity Resolution in Language Understanding 3

1.1.1 Part-of-Speech Ambiguity 3
1.1.2 Word Sense Ambiguity 3
1.1.3 Dealing with Unknown Words 4
1.1.4 Prepositional Phrase Attachment 4
1.1.5 Pronoun Resolution 5
1.1.6 Understanding Conjunctions and Appositives 6
1.1.7 Uniform Treatment of Ambiguity 6

1.2 A Framework for Knowledge Acquisition 7
1.3 An Example 9
1.4 Advantages of the Framework 11
1.5 Claims and Contributions of the Thesis 13
1.6 What's to Follow 14

2. RELATED WORK 16

2.1 Hand-crafted Knowledge Acquisition 16
2.2 Intelligent Interfaces 17
2.3 Statistical Approaches 18

2.3.1 Acquisition of lexical knowledge 18
2.3.2 Acquisition of structural knowledge 21
2.3.3 Comprehensive approaches to ambiguity resolution 22

2.4 Knowledge-Based Approaches 23
2.5 Connectionist Approaches 23

2.5.1 Distributed Approaches 23
2.5.2 Localist Approaches 24

2.6 Machine Learning Approaches 24

2.6.1 A Comparison with SOAR 25

2.7 Case-Based Approaches 25
2.8 Summary 28

3. KNOWLEDGE ACQUISITION FOR CONCEPTUAL SENTENCE ANALYSIS 30
3.1 The CIRCUS Conceptual Sentence Analyzer 30
3.1.1 Syntactic Processing in CIRCUS 31
3.1.2 Predictive Semantics in CIRCUS 33
3.1.3 Handling Embedded Clauses in CIRCUS 35
3.1.3.1 Understanding Wh-Constructions 36
3.1.3.2 LICK Caveats and Conclusions 38
3.2 Examples of “Context” in CIRCUS 39
3.3 Knowledge Needed for Conceptual Analysis 42
3.3.1 Syntactic Processing 42
3.3.2 Predictive Semantics Module 43
3.3.3 LICK Processing 43
3.3.4 Knowledge Acquisition Tasks for Kenmore 44
4. DOMAIN-SPECIFIC KNOWLEDGE ACQUISITION 45
4.1 The Task 45
4.2 Constraints on the Knowledge Acquisition Process 46
4.3 The MUC and TIPSTER Performance Evaluations 47
4.3.1 MUC Corpus of Latin American Terrorism 48
4.3.2 TIPSTER Corpus of Business Joint Ventures 52
5. LEARNING LEXICAL KNOWLEDGE 56
5.1 The Problem 57
5.2 Instantiating the Kenmore Framework for MayTag 59
5.3 MayTag’s Training Phase 61
5.3.1 The Taxonomies 62
5.3.2 Case Representation 66
5.3.3 Case Base Construction 68
5.4 MayTag’s Application Phase 68
5.4.1 Recognizing Incomplete Word Definitions 70
5.4.2 Modified Case Retrieval Algorithm 70
5.5 Evaluation of MayTag 72
5.5.1 Handling Occasional Unknown Words 73
5.5.2 Tagging All Open-Class Words From Scratch — Acquiring a Domain-Specific Lexicon 74
5.5.2.1 MayTag in the MUC-5/TIPSTER Evaluation 74
5.5.3 Creating Explicit System Lexicons 76
5.5.4 Extending MayTag for Domain-Independent Lexical Disambiguation 77
5.6 Summary 78

6. USING DECISION TREES TO DISCARD IRRELEVANT FEATURES . . . 79
- 6.1 A Decision Tree Approach to Feature Set Selection . . . 80
- 6.2 Evaluation of the Approach . . . 82
  - 6.2.1 Case Retrieval Using All Context Features . . . 83
  - 6.2.2 Case Retrieval Using Human-Generated Relevant Features . . . 85
  - 6.2.3 Hybrid Case Retrieval Algorithm . . . 87
- 6.3 Analysis of Approach . . . 90
  - 6.3.1 Decision Trees for Part-of-Speech Prediction . . . 90
  - 6.3.2 Decision Trees for Semantic Feature Prediction . . . 90
  - 6.3.3 Decision Trees for Concept Type Prediction . . . 92
- 6.4 Problems with the Hybrid Case Retrieval Algorithm . . . 95
- 6.5 Related Work in Automated Feature Set Selection . . . 99

7. MAYTAG PERFORMANCE ANALYSIS . . . 100
- 7.1 Part-of-Speech Disambiguation . . . 100
  - 7.1.1 Types of Errors . . . 102
  - 7.1.2 Reducing False Hits on Majority Classes . . . 103
- 7.2 Semantic Feature Disambiguation . . . 106
  - 7.2.1 Types of Errors . . . 110
  - 7.2.2 Domain-Dependent vs. Domain-Independent Features . . . 113
- 7.3 Domain-Specific Concept Disambiguation . . . 115
  - 7.3.1 Types of Errors . . . 116
  - 7.3.2 Expanding the Concept Type Taxonomy . . . 117
- 7.4 Generalization Issues . . . 122
- 7.5 Summary . . . 122

8. FINDING ANTECEDENTS OF RELATIVE PRONOUNS . . . 124
- 8.1 The Problem . . . 124
  - 8.1.1 Current Approaches to Relative Pronoun Disambiguation . . . 127
- 8.2 Instantiating the Kenmore Framework for WHirlpool . . . 127
- 8.3 WHirlpool's Training Phase . . . 129
  - 8.3.1 Case Representation . . . 130
  - 8.3.2 Case Base Construction . . . 132
- 8.4 WHirlpool's Application Phase . . . 133
  - 8.4.1 Case Retrieval in WHirlpool . . . 134
- 8.5 Evaluation of WHirlpool Using the Baseline Case Representation . . . 136
- 8.6 Using Cognitive Biases to Improve a Baseline Case Representation . . . 137
  - 8.6.1 Incorporating the Recency Bias . . . 138
  - 8.6.2 Incorporating the Restricted Memory Bias . . . 140
  - 8.6.3 Incorporating the Subject Accessibility Bias . . . 142
  - 8.6.4 Discussion . . . 143
- 8.7 Summary . . . 145

9. ESTABLISHING CONSTRAINTS ON THE KENMORE FRAMEWORK 147
9.1 The Corpus 148
9.2 The Sentence Analyzer 149
9.3 The Taxonomies 149
9.4 The Parsing Task To Be Learned 150
9.5 The Case Representation: Representing Context 151
9.5.1 Using a Uniform Case Representation 152
9.5.2 Righthand Context 153
9.5.3 Improving the Case Representation Automatically 154
9.5.3.1 Making Finer Distinctions 155
9.5.3.2 Derived Features and the Issue of Transfer 155
9.6 The Inductive Learning Component 156
9.7 Training Issues 156
9.8 Putting It All Together 157
10. CONCLUSIONS 160
10.1 Contributions of the Research 160
10.2 Future Directions 162
10.2.1 Broader Domains 162
10.2.2 Unsupervised Training Methods 163
10.2.3 Smarter Supervised Training 164
10.2.4 Problems That Cross Clause and Sentence Boundaries 164
10.2.5 Dynamic Modification of Case Representation 165
10.2.6 Environment for Natural Language System Development 165
10.2.7 Higher Level Knowledge Structures 165
BIBLIOGRAPHY 166

LIST OF TABLES

Table Page

5.1 Semantic Feature Taxonomy (Part 1). 63

5.2 Semantic Feature Taxonomy (Part 2). 64

5.3 Concept Type Taxonomy. 65

5.4 Part-of-Speech Taxonomy. 65

5.5 Handling Occasional Unknown Words (% correct for prediction of word definition features). 73

5.6 Tagging from Scratch (% correct for prediction of word definition features). 75

5.7 Using MayTag with Larger Case Base for the Tagging-From-Scratch Task (% correct). 75

5.8 Creating an Explicit Lexicon. 78

6.1 Case Retrieval Using All Context Features (% correct). 85

6.2 Case Retrieval Using Human-Generated Relevant Features (Table shows % correct for k=10; ** indicates significance with respect to the "all features" variation; p = .01). 86

6.3 Case Retrieval Using C4.5-Generated Relevant Features (Table shows % correct for $k = 10$; * and ** indicate the significance of the decision tree features variation with respect to **all** other variations. $** \rightarrow p = .01$ and $* \rightarrow p = .05$. See accompanying text for explanation of **/*.). 89

7.1 Part-of-Speech Frequencies for Unknown Words (2056 cases). 102

7.2 Part-of-Speech Error Matrix for Occasional-Unknown-Word Tagging. (*'s indicate verb/non-verb errors.) 104

7.3 Unknown Words and Parts of Speech for Retrieved Cases. 104

7.4 Using Smaller Values of k for Part-of-Speech Prediction in the Occasional-Unknown-Word Task (Results shown are for open-class words only using the 2056-case case base). 105

7.5 General Semantic Feature Frequencies for Unknown Words (2056 cases). . 108

7.6 Specific Semantic Feature Frequencies for Unknown Words (2056 cases). . . 109

7.7 General Semantic Feature Error Matrix for Occasional-Unknown-Word Tagging. 111

7.8 Specific Semantic Feature Error Matrix for Occasional-Unknown-Word Tagging. 112

7.9 Domain-Dependent and Domain-Independent Semantic Features. (General semantic features are shown in UPPER CASE; specific semantic features are shown in lower-case.) 113

7.10 Results for Domain-Independent vs. Domain-Dependent General Semantic Features (% correct). 114

7.11 Results for Domain-Independent vs. Domain-Dependent Specific Semantic Features (% correct). (Values in parentheses indicate percentage correct if *nil* included.) 115

7.12 Concept Type Frequencies for Unknown Words (2056 cases). 116

7.13 Concept Type Error Matrix for Occasional-Unknown-Word Tagging. 118

7.14 Effect of Smaller Values of k for Prediction of Non-*nil* Concept Types. (Results shown are for open-class words using the 2056-case case base for the occasional-unknown-word task). 118

7.15 Expanded Concept Type Taxonomy. 119

7.16 Prediction of Concept Types from the Expanded Taxonomy (occasional-unknown-word task, k=10). 120

7.17 Concept Type Frequencies for Unknown Words Using the Expanded Concept Taxonomy (3060 cases). 121

8.1 WHirlpool Results (% correct). 137

8.2 Incorporating the Recency Bias by Modifying the Weight Vector. 139

8.3 Results for the Recency Bias Representations (% correct). 140

8.4 Results for the Restricted Memory Bias Representation (% correct). (*'s indicate significance with respect to the original baseline result shown in **boldface**, $* \rightarrow p = 0.05$). 141

8.5 Results for the Subject Accessibility Bias Representation (% correct). 142

8.6 Additional Results for the Subject Accessibility Bias Representation (% correct). (*'s indicate significance with respect to the original baseline result shown in **boldface**, $* \rightarrow p = 0.05$, $** \rightarrow p = 0.01$; RM refers to the memory limit). 143

8.7 Cognitive Bias Modifications. 144

8.8 Summary of Cognitive Bias Results. 146

LIST OF FIGURES

Figure Page

1.1 Simple Summarization Task for Natural Language Processing Systems. . . . 1

1.2 More Complicated Tasks for Natural Language Processing Systems. 2

1.3 Prepositional Phrase Attachment Ambiguities. 5

1.4 Kenmore Training/Acquisition Phase. 8

1.5 Kenmore Application Phase. 9

1.6 Training Case for "Japan". 11

1.7 Problem Case for "tote". 12

1.8 Retrieved Case for "tote" and the Training Sentence that Spawned It. . . . 12

3.1 CIRCUS Status After "John brought..." 32

3.2 PSM Status After "John brought..." 34

3.3 Instantiated Semantic Case Frame. 34

3.4 State of CIRCUS after "The policeman saw the boy who..." 37

3.5 Child LICK Status after "The policeman saw the boy who the crowd at the party accused..." 38

3.6 Child LICK Status after "The policeman saw the boy who the crowd at the party accused of the crime." 38

4.1 Terrorism Corpus Answer Key (Part 1). 50

4.2 Terrorism Corpus Answer Key (Part 2). 51

4.3 Simplified JV Answer Key. 53

4.4 Actual JV Answer Key. 54

5.1 MURDER Semantic Case Frame. 59

5.2 Instantiating Kenmore for MayTag. 60

5.3 Components of MayTag's Case-Based Inductive Learning Algorithm. . . . 60

5.4 MayTag Training Phase. 61

5.5 Case for "venture." 66

5.6 MayTag Application Phase. 69

5.7 Black Box View of Feature Selection Algorithm. 71

6.1 Simple Decision Tree for Classifying Balls. 81

6.2 Decision Tree Feature Selection. 81

6.3 Training Case for "venture" as it is Used in the Sentence "Toyota Motor Corp. has set up a joint venture firm with Yokagawa Electric Corp..." . 84

6.4 Selecting Relevant Features for the Lexical Acquisition Task. 88

6.5 Hybrid Case Retrieval. 88

6.6 Sample Decision Tree for Part-of-Speech Prediction. 91

6.7 Histogram of Features From rel_{p-o-s} Lists (part-of-speech prediction). . . . 91

6.8 Sample Decision Tree for General Semantic Feature Prediction (partial tree). 93

6.9 Sample Decision Tree for Specific Semantic Feature Prediction (partial tree, part 1). 94

6.10 Sample Decision Tree for Specific Semantic Feature Prediction (partial tree, part 2). 95

6.11 Histogram of Features From $rel_{gen-sem}$ Lists (general semantic feature prediction). 96

6.12 Histogram of Features From $rel_{spec-sem}$ Lists (specific semantic feature prediction). 96

6.13 Sample Decision Tree for Concept Type Prediction (part 1). 97

6.14 Sample Decision Tree for Concept Type Prediction (part 2). 98

6.15 Histogram of Features From $rel_{concept}$ Lists (domain-specific concept prediction). 98

7.1 Part-of-Speech Prediction (% correct). 101

7.2 Performance on Individual Parts of Speech for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting part of speech, p; $*$'s indicate % of instances that should have been assigned p; line is best linear approximation to x's.) 103

7.3 General Semantic Feature Prediction (% correct). 106

7.4 Specific Semantic Feature Prediction (% correct). 107

7.5 Performance on Individual General Semantic Features for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting general semantic feature, g; $*$'s indicate % of instances that should have been assigned g; line is best linear approximation to x's.) 108

7.6 Performance on Individual Specific Semantic Features for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting specific semantic feature, s; $*$'s indicate % of instances that should have been assigned s; line is best linear approximation to x's.) 110

7.7 Concept Type Prediction (% correct). 115

7.8 Performance on Individual Concept Types for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting concept type, c; $*$'s indicate % of instances that should have been assigned c; line is best linear approximation to x's.) 117

8.1 Understanding Relative Clauses. 125

8.2 Relative Pronoun Antecedents. 125

8.3 Instantiating Kenmore for WHirlpool. 128

8.4 Components of WHirlpool's Case-Based Inductive Learning Algorithm. . . 128

8.5 WHirlpool Training Phase. 129

8.6 Relative Pronoun Disambiguation Training Cases. 130

8.7 WHirlpool Application Phase. 133

8.8 WHirlpool Case Retrieval. 134

8.9 Incorporating the Recency Bias Using a Right-to-Left Labeling. 138

9.1 Modified Kenmore Framework. 153

CHAPTER 1

INTRODUCTION

It is a particularly exciting time for the field of natural language processing (NLP). The availability of on-line corpora is rapidly changing the field from one dominated by theoretical models of often very specific linguistic phenomena to one guided by computational models that simultaneously account for a wide variety of phenomena that occur in real-world text. As a result, NLP systems that process only a handful of sentences are no longer taken seriously, and there is increasing emphasis on building programs that can handle large amounts of real-world text.

Although current natural language processing systems cannot yet perform in-depth text understanding, they *can* read an arbitrary text and summarize its major events provided that those events fall within a particular domain of interest (e.g., stories about natural disasters or terrorist events) [Chinchor *et al.*, 1993]. This scenario is illustrated in Figure 1.1. Thus far, among the very best-performing and most robust language processing

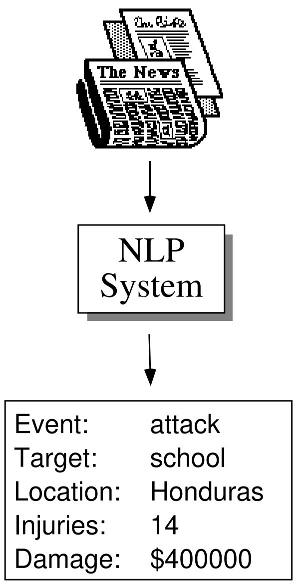

Figure 1.1: Simple Summarization Task for Natural Language Processing Systems.

systems for this type of limited summarization task have been knowledge-based natural language systems — NLP systems that understand an input text by relying heavily on handcrafted knowledge about the domain and about the world in general. Not surprisingly, however, generating this background knowledge for new domains is time consuming, difficult, and error prone, and requires the expertise of computational linguists familiar with the underlying NLP system. This is an example of the *knowledge engineering bottleneck* for natural language processing systems. It is one of the biggest problems in designing and building natural language systems and promises only to become worse as natural language systems attempt to understand a wider variety of texts, to produce more complex summaries of the text, and to derive knowledge structures directly from text (see Figure 1.2). On the other hand, much of human knowledge is described in written documents and, as mentioned above, NLP systems can now perform limited understanding of complicated texts. Machine learning techniques for inductive learning have also become increasingly available and offer powerful mechanisms for simplifying the knowledge acquisition process.

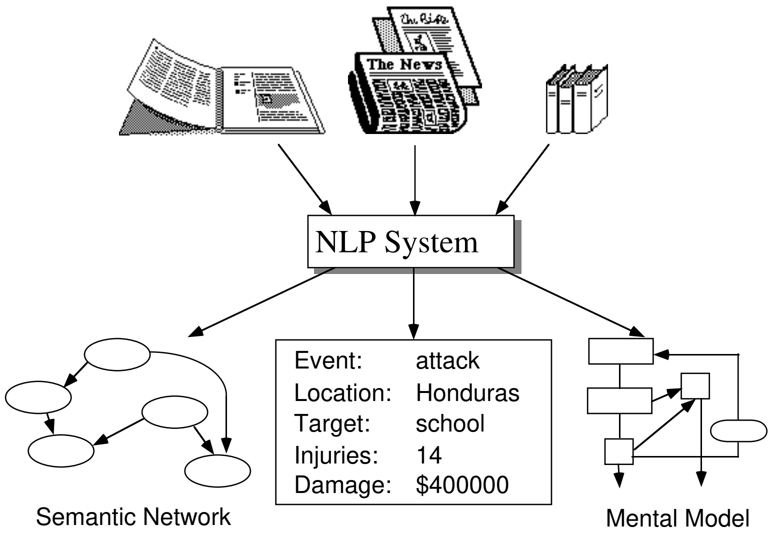

Figure 1.2: More Complicated Tasks for Natural Language Processing Systems.

As a result of these advances, we believe that it is feasible for NLP systems to begin to bootstrap their own knowledge bases and this thesis presents a framework within which this bootstrapping process can occur. More specifically, *the thesis presents a general framework for tackling the knowledge-engineering bottleneck for natural language processing systems at the level of sentence analysis*. The knowledge acquisition framework, called **Kenmore**[1], exploits an on-line corpus using a robust parsing strategy and symbolic machine learning techniques, and requires minimal human intervention. Moreover, the framework uniformly addresses a range of subproblems in sentence analysis, each of which traditionally had required a separate computational mechanism. In particular, it supports the *acquisition* and *application* of heuristics for *semantic* and *syntactic* ambiguity resolution at both the *lexical* and *structural* levels.

[1]**ken** (ken), *vi.* 1 [Scot.] to know

1.1 Ambiguity Resolution in Language Understanding

Ascertaining what is intended in a text when more than one interpretation is possible has always been a central issue in natural language processing: ambiguity resolution is required whenever the system must choose among two or more distinct representations of the input. Ambiguity pervades virtually all aspects of language analysis, and sentence analysis in particular exhibits a large number of syntactic, semantic, and pragmatic ambiguities that demand adequate resolution before the sentence can be understood. The Kenmore knowledge acquisition framework is designed to acquire solutions to virtually all lexical and structural ambiguity problems encountered during sentence analysis. The sections below describe just a few. Unless otherwise noted, all examples in this section are taken directly or derived from sentences the TIPSTER/MUC business joint ventures corpus.[2]

1.1.1 Part-of-Speech Ambiguity

Knowing the part of speech of a word in a particular context, for example, often supplies important hints for determining the word's function in the sentence. Consider the word "plans" in the sentences below:

1. BMW **plans** to build a plant for large machine tools in Eisenach.
2. Ambitious BMW **plans** to build a plant for large machine tools in Eisenach were not approved by management.

Despite nearly identical local contexts, "plans" is a verb in sentence 1, but a noun in sentence 2, and making this distinction is crucial to determining meaning of each sentence. The main event in the first sentence is a planning event in which BMW is the actor. Sentence 2, on the other hand, describes a state change — the canceling of a set of plans.

1.1.2 Word Sense Ambiguity

Even if a word's part of speech is known, the intended meaning of the word in a particular context often requires disambiguation. The word "vehicles," for example, is a noun in both sentences below. In each sentence, however, the word takes on a very different meaning:

1. Suzuki Motor Co. will produce 240,000 passenger and commercial **vehicles** annually at a new factory in South Korea.
2. Eight industries have emerged as **vehicles** to transform Indonesia into a nation of technology.

In sentence 1, "vehicles" refers to the product of a company while in the second sentence it is used metaphorically to mean "instruments" or "means." Even when it seems as if a word is unquestionably unambiguous, it can be used in contexts that confer a novel meaning. In general, one would probably say that the word "Japan" is unambiguous, for example. It refers to a country in eastern Asia. Consider the following sentences, however:

[2] This corpus will be described in detail in Chapter 4.

1. The company reported additional purchases of implanters by unnamed chip makers in **Japan**.

2. Pearle has formed a joint venture with **Japan** Optical.

3. The Soviet Union has proposed a joint venture with **Japan** to build a salmon hatchery in the Soviet Union.

The "country in eastern Asia" meaning of "Japan" is, in fact, the one intended in sentence 1. But in the second sentence, "Japan" is part of a company name and that company may or may not be located in Japan. In the last sentence, "Japan" refers more precisely to the government of Japan rather than the body of land that is Japan.

1.1.3 Dealing with Unknown Words

The problem of ambiguity in sentence analysis is also clearly pronounced in the case of unknown words. When encountering a word for which it has no definition, a robust NLP system makes a series of decisions that together shape the meaning of the word as it functions in the current context. Each decision is essentially a separate, but related, ambiguity resolution task. What is the word's part of speech in the current context? What is its meaning? How is it related to other items in the sentence or paragraph or text? Is the word of special importance with respect to the goals of the text processor?

A related problem for natural language processing systems is knowing when the system's knowledge of a word is incomplete. Assume, for example, that an NLP system had a definition for the word "market" that was syntactically and semantically compatible with its use in sentence 1 below, but then encountered "market" in sentence 2:

1. The Asian beef **market** generally is starting to open up to exporters.

2. Two companies plan to **market** a new chip with ceramic circuits.

A robust system should (1) note that its current definition is inadequate, (2) infer the appropriate syntactic and semantic features of "market," (3) incorporate the new definition into the system's lexicon, and (4) determine how the system will distinguish the two uses of "market" in the future.

1.1.4 Prepositional Phrase Attachment

In addition to instances of *lexical* ambiguity (i.e., ambiguity at the word level), ambiguity resolution is required at the constituent, or *structural*, level as well. Consider the prepositional phrases in the following very similar sentences:

1. Taiyo Oil Co. said Wednesday it plans to open the oil refinery **in** Bintulu **on** the island of Kalimantan **in** Malaysia.

2. Taiyo Oil Co. said Wednesday it plans to open an oil refinery **in** a joint venture **on** the island of Kalimantan **in** June.

Although the low-level syntactic structure of the sentences is identical, the prepositional phrase attachment decisions vary substantially. As shown in Figure 1.3, the "in" prepositional phrase that follows "refinery" (in1-pp) modifies "refinery" in the first sentence (i.e., the refinery will be in Bintulu), but modifies the verb in the second sentence. In addition, the second prepositional phrase that begins with "in" in sentence 1 (in2-pp) modifies the preceding noun (i.e., Kalimantan is in Malaysia), while the prepositional phrase in the same position in sentence 2 modifies the verb (i.e., the refinery will open in June). Again, each prepositional phrase attachment decision will affect the semantic representation derived for the sentence by the natural language processing system.

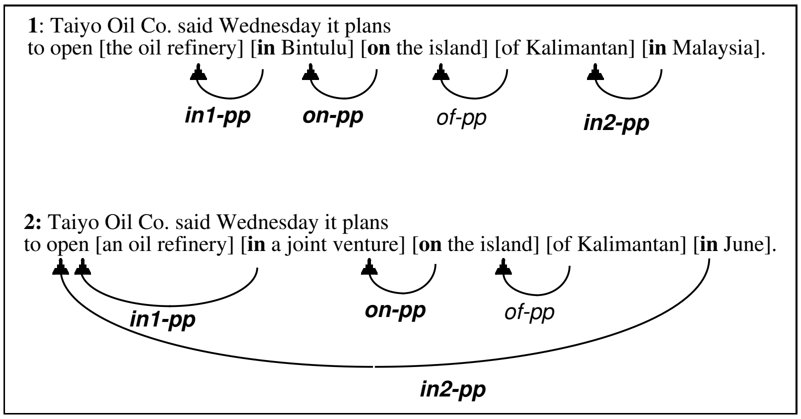

Figure 1.3: Prepositional Phrase Attachment Ambiguities.

1.1.5 Pronoun Resolution

A natural language system also requires a mechanism for accurately determining the phrase referenced by a pronoun. Many times, the gender or number of the pronoun can limit the number of possible referents, but often the ambiguity must be resolved by other means. Consider the following sentences:

1. Merck & Co. formed a joint venture with Ache Group, of Brazil. **It** will be called Prodome Ltd.

2. Merck & Co. formed a joint venture with Ache Group, of Brazil. **It** will own 50% of the new company to be called Prodome Ltd.

3. Merck & Co. formed a joint venture with Ache Group, of Brazil. **It** had previously teamed up with Merck in two unsuccessful pharmaceutical ventures.

In each sentence, the pronoun is resolved differently by the righthand context that follows. In sentence 1, "it" refers to the joint venture company; in sentence 2, "it" refers to Merck & Co.; and in sentence 3, "it" refers to Ache Group.

1.1.6 Understanding Conjunctions and Appositives

Understanding complicated noun phrases that involve conjunctions and appositives is also a notoriously difficult structural ambiguity resolution task.

- The decision to abandon production of [[Honda's Legend model at Rover's Cowley Plant] and [its Accord model at Strathmere]] was made 18 months after the project began.
- The decision to abandon production of [[Honda's Legend model at Rover's [Cowley and Birmingham] Plants] and [its Accord model at Strathmere]] was made 18 months after the project began.
- [[The decision to abandon production of Honda's Legend model at Rover's Cowley Plant] and [the subsequent legal suit]] were both items of interest in yesterday's Guardian.

As seen in the above examples, even slight changes in the wording of a sentence can produce radical changes in the attachment decisions for conjunctions. The problem only becomes worse when appositives appear within the constructions:

- The decision to abandon production of [[Honda's Legend, its high-end luxury model] and [its Accord model, the top-selling import in the U.S.]], was made 18 months after the project began.

1.1.7 Uniform Treatment of Ambiguity

The sections above identified a number of ambiguities faced by natural language systems during sentence analysis. Typically, NLP systems treat each of these ambiguities separately along a number of dimensions. First, an NLP system views each ambiguity as a completely different problem. Second, each type of ambiguity is handled by a separate module of the system. There is often a part-of-speech tagger, for example, that is solely responsible for handling part-of-speech ambiguities. Third, each ambiguity resolution module has limited access to the system's knowledge. The part-of-speech tagger, for example, may only have access to part-of-speech information for each word in the sentence. Fourth, computationally, the NLP system handles each ambiguity using a different mechanism (e.g., rule-based processing, context free parsers, regular expression recognizers, Markov models, constraint satisfaction, relaxation networks), each one of which may require very different sorts of background knowledge.

This dissertation presents a knowledge acquisition framework that instead *uniformly* addresses the problem of ambiguity in sentence analysis. In Kenmore,

- All types of ambiguity are viewed as instances of the same general problem.
- As a result, a single ambiguity resolution module handles all types of ambiguity.
- That module has access to all of the system's knowledge for resolving each type of ambiguity.
- Computationally, a single mechanism is responsible for handling each type of ambiguity. Moreover, the same mechanism is responsible both for the acquisition and the application of the ambiguity resolution heuristics.

As a result of its uniform view of ambiguity, Kenmore can handle lexical, structural, syntactic, and semantic ambiguity problems within a single architecture. As described below, solutions to these problems can be acquired with minimal human intervention, thus avoiding the knowledge acquisition bottleneck in natural language system design.

1.2 A Framework for Knowledge Acquisition

The Kenmore knowledge acquisition framework for natural language processing systems is motivated by the following observations:

1. *The problem of ambiguity resolution can be recast as a classification problem.* In classification problems, the system is presented with the description of an object, situation, or set of observations, and must label it with one of a number of *classes* or categories.[3] Given a description of the context of a lexical item, for example, a classification system will tag the word with the appropriate syntactic class (e.g., noun) and semantic class (e.g., a human); given a prepositional phrase and a description of the surrounding context, a classification system will choose among available attachment points. By treating ambiguity resolution as a classification problem, we can immediately apply any of a number of inductive machine learning techniques to *learn* solutions to these problems rather than handcrafting a set of disambiguation rules. We expect the inductive learning algorithms to capture automatically those regularities in the use of language that permit the resolution of ambiguity in sentence analysis.

2. *Some aspects of language processing can be viewed as memory-based.* By this we mean that a sentence analyzer can resolve an ambiguity by "remembering" how it resolved a similar ambiguity in the past. Here, the necessary system behavior is best provided by case-based, or instance-based, learning algorithms that store individual episodes in their entirety rather than incorporating them into a global concept description. In addition, case-based reasoning is useful in domains for which no strong domain theory exists. This is the case in natural language processing where the mechanisms that underly language understanding and language acquisition are not well understood. Finally, there is evidence from psychology that some language acquisition tasks, like learning word meanings, proceed through instance-based stages (e.g., Keil and Kelly [1987]).

3. *Knowing the context in which an ambiguity occurs is crucial for resolving it.* Text understanding in general, and sentence analysis in particular, require a series of context-sensitive mappings from one representation into another — the system often maps the words of a sentence into parts of speech, the part-of-speech sequences into low-level constituents, and the low-level constituents into predicate-argument relations, for example. As a result, the solutions acquired by Kenmore are context-sensitive solutions.

---

[3]Assume, for example, that there are four classes of balls — baseballs, volleyballs, kickballs, and tennis balls — that will be described in terms of their color, type of covering, and diameter. Given a description of a ball that is fluorescent green in color, with a fuzzy covering, and three inches in diameter, a classification system presumably should label the ball a tennis ball.

4. *People are still by far the very best language processors.* In spite of the difficulties it creates for machines, ambiguity resolution is generally a trivial problem for people. For this reason, a human supplies supervision during Kenmore's training phase.

Kenmore relies on three major components. First, it requires a **corpus of texts** — a collection of on-line documents. Second, it requires a **robust sentence analyzer**, or parser. The specific parser used throughout is the CIRCUS conceptual sentence analyzer [Lehnert, 1990]. This is not a requirement of the framework, however, and a variety of sentence analyzers could be used in its place. Finally, the framework requires a **case-based reasoning (CBR) module** [Riesbeck and Schank, 1989, Kolodner, 1993]. Very generally, CBR systems solve problems by first creating a case base of previous problem-solving episodes. Then, when a new problem enters the system, the "most similar" case is retrieved from the case base and used to solve the novel problem. The retrieved case can either be used directly or after one or more modifications to adapt it to the current problem-solving situation.

There are two phases to the framework: (1) a partially automated training phase, or **acquisition phase**, in which the solution to a particular problem in sentence analysis is learned, and (2) an **application phase**, in which the learned solution can be applied in novel situations. More specifically, the goal of Kenmore's training phase (Figure 1.4) is to create a *case base*, or memory, of ambiguity resolution episodes for a particular type of ambiguity (e.g., prepositional phrase attachment or word sense disambiguation). To do this, a small set of sentences is first selected randomly from the corpus. Next, the sentence analyzer processes the training sentences and, with a human supervisor, creates a case every time an instance of the ambiguity occurs. To learn a set of heuristics to handle prepositional phrase attachment, for example, the parser would create a case whenever it recognizes a prepositional phrase. As shown in Figure 1.4, each case has two parts. The *context* portion of the case encodes the context in which the ambiguity was encountered — this is essentially a representation of the state of the parser at the point of the ambiguity. The context part of the case is supplied automatically by the sentence analyzer in both the training and application phases. The *solution* portion of the case describes how the ambiguity was resolved in the current example. In the training phase,

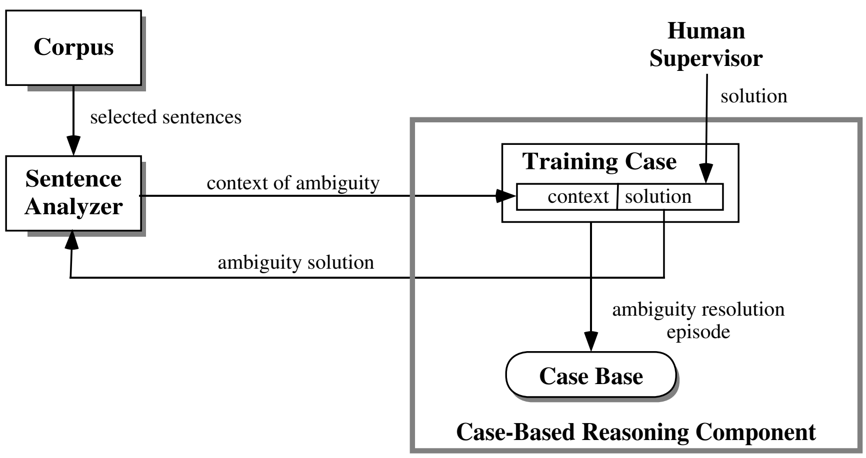

Figure 1.4: Kenmore Training/Acquisition Phase.

a person supplies this part of the case using a menu-driven interface. For prepositional phrase attachment, for example, the human supervisor simply chooses the phrase that the current prepositional phrase modifies. Together, the context and solution portions of the case represent a single ambiguity resolution episode. Once the case is created, it is stored in the case base. In addition, the solution to the ambiguity is forwarded to the parser so that it can resolve the current ambiguity, update its state, and continue processing the training sentences. At the end of the training phase, the case base will contain one case for every instance of the ambiguity that appears in the training sentences. When acquiring heuristics for prepositional phrase attachment, for example, there will be one case for every prepositional phrase attachment decision in the training sentences.

After training, we can use the case base *without human intervention* to resolve occurrences of the ambiguity in novel sentences from the corpus (Figure 1.5). Whenever the sentence analyzer encounters an instance of the ambiguity, it creates a problem case, automatically filling in its context portion based on the state of the system at the point of the ambiguity. The structure of a problem case is identical to that of a training case except that the "solution" part of the case is missing. To resolve the current ambiguity, Kenmore next compares the problem case to each case in the case base, retrieves the most similar training case, and sends the solution stored there back to the parser to be used as the solution in the current situation.

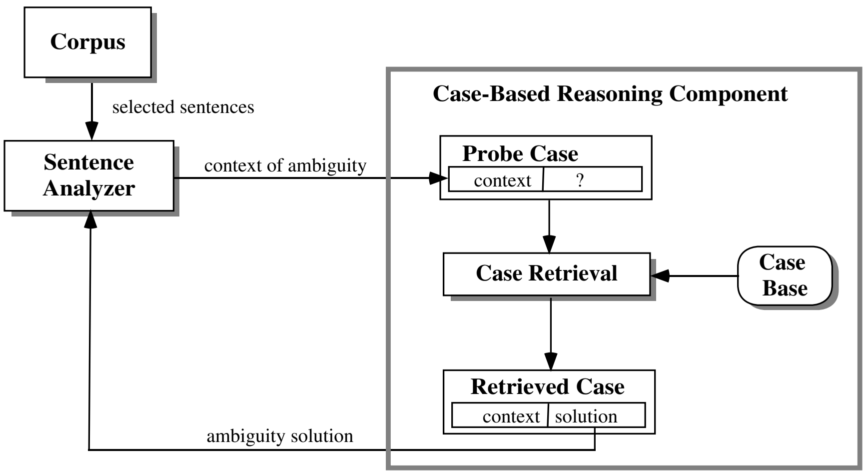

Figure 1.5: Kenmore Application Phase.

Of course, many issues must be addressed in order to apply the Kenmore framework to a particular problem in sentence analysis. These include: the choice of an appropriate learning algorithm and similarity metric, the choice of a corpus and sentence analyzer, and the representation of the "context" and "solution" in a case. We will address these and other issues in detail in later chapters and proceed next with an example of Kenmore in action in an attempt to make the workings of the framework more concrete.

1.3 An Example

This section shows how the Kenmore framework for knowledge acquisition can be used to learn word sense disambiguation heuristics. To start, we will assume that the goal of the NLP system is to process texts from a particular corpus — the TIPSTER business joint ventures corpus (henceforth, the JV corpus). Briefly, this corpus contains over 1000 actual newswire accounts of world-wide activity in the area of joint ventures or "tie-ups" between businesses. We also assume that word meanings will be represented in terms of one or more semantic features from a predefined taxonomy designed for use in the joint ventures domain.[4] Finally, we assume that Kenmore has been instantiated with the CIRCUS [Lehnert, 1990] sentence analyzer.

To generate word sense disambiguation heuristics for use with the JV corpus, Kenmore first randomly selects a subset of sentences from the corpus and presents them as input to CIRCUS. Without describing its details, CIRCUS processes the training sentences one word at a time, looking up each word in its domain-specific lexicon for part-of-speech and word sense information. Whenever more than one word sense is available, Kenmore consults CIRCUS and a human supervisor to create a word sense disambiguation training case for the ambiguous word. For example, because "Japan" can refer to either a *country*, a *company name*, or a *government* in this domain (see Section 1.1.2 above for examples), Kenmore creates a training case when it reaches "Japan" in the sentence:

IBM opened a factory in **Japan**.

Cases in Kenmore are represented as a list of attribute-value pairs, also called features. A portion of the training case for "Japan" is shown in Figure 1.6. All but one of the displayed features describe the context in which the ambiguity occurred. These are supplied automatically by CIRCUS and represent CIRCUS's state when the ambiguous word was recognized. More generally, Kenmore's context features are meant to encode any information available to the sentence analyzer that may be of help in disambiguating the current word. The exact type and form of knowledge included in the context features, however, is parser-dependent. In addition, a human supervisor uses a menu-driven interface to indicate which of the three plausible semantic features is appropriate in the given context (i.e., *country*). The selection is encoded as the solution portion of the case and represents the class information to associate with the training case. Next, Kenmore stores the training case in the case base and notifies CIRCUS of the correct word sense. Kenmore creates training cases like the one shown in Figure 1.6 every time a word sense ambiguity occurs in the training sentences.

After training, the natural language system can use the resulting case base to resolve semantic feature ambiguities in novel sentences. Consider, for example, the word "tote" in the following sentence:

> AAAI has set up a joint venture with ACL for production of canvas **tote** bags for distribution at annual conferences.[5]

In the joint ventures domain, "tote" has three plausible meanings associated with the noun form of the word — it can refer to a *company name*, a *product*, or a generic *thing*. (The *thing* semantic feature denotes a word meaning that is unimportant in the joint ventures domain.) To determine which of the three is appropriate in the current context, Kenmore

---

[4]Details regarding the taxonomies are not important for the following example; they will be described in greater detail in subsequent chapters. In addition, the relationship between word senses and semantic features will be discussed in Chapter 7.2.

[5]This sentence is not from the JV corpus.

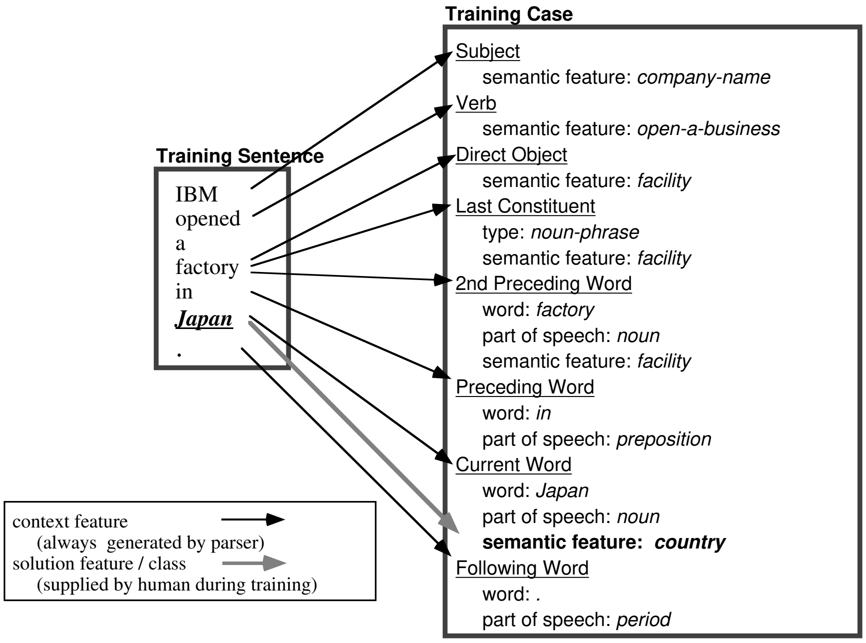

Figure 1.6: Training Case for “Japan”.

creates a problem case (Figure 1.7). Like the training cases, the problem case includes a set of context features that describe the context in which the ambiguous word occurred. Unlike the training cases, however, the problem case contains no solution (i.e., it contains no class information). To resolve the ambiguity, the problem case is compared to all of the cases in the case base and the most similar one is retrieved. For now, we will ignore the important issue of how to measure similarity and select the best case. In short, the case retrieved in response to the “tote” case after training on 120 sentences from the JV corpus (2056 cases) is shown in Figure 1.8 along with the training sentence that generated it. The retrieved case indicates (correctly) that “tote” refers to a *product* in the current context.

1.4 Advantages of the Framework

Kenmore’s approach to knowledge engineering for natural language processing systems has a number of advantages over traditional methods for acquiring the background knowledge needed for sentence analysis. Although more detailed comparisons to existing approaches will be discussed in Chapter 2, the major advantages are summarized here:

- *In Kenmore, the same case-based method is used to acquire solutions to syntactic and semantic problems at both the lexical and structural levels.* This simplifies the system design. As mentioned earlier, different methods are traditionally used to handle each class of problem encountered during sentence analysis.

- *The case base and case retrieval algorithm together implicitly define a set of disambiguation heuristics.* This obviates the need for generating and maintaining explicit, hand-coded disambiguation heuristics.

- *No hand-coded heuristics are required to drive the acquisition process.* Again, this greatly simplifies system design and maintenance.

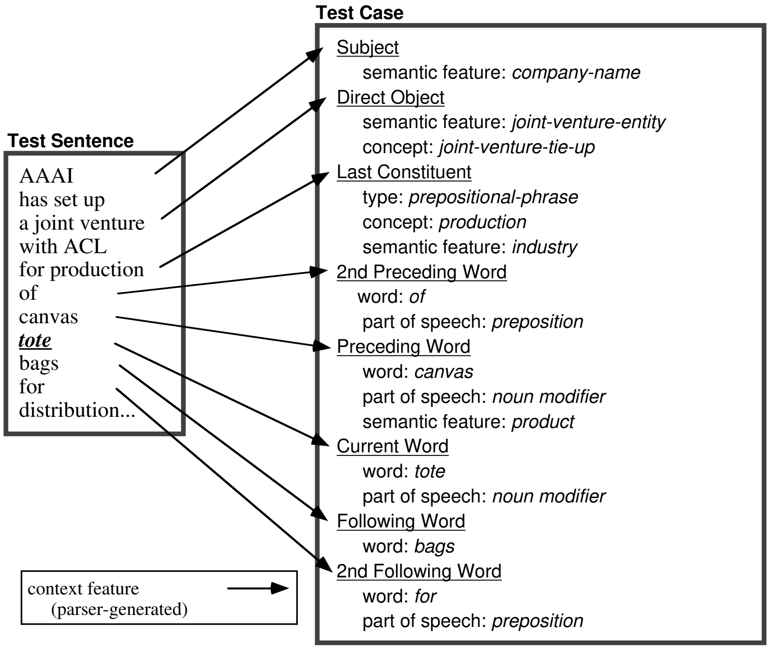

Figure 1.7: Problem Case for "tote".

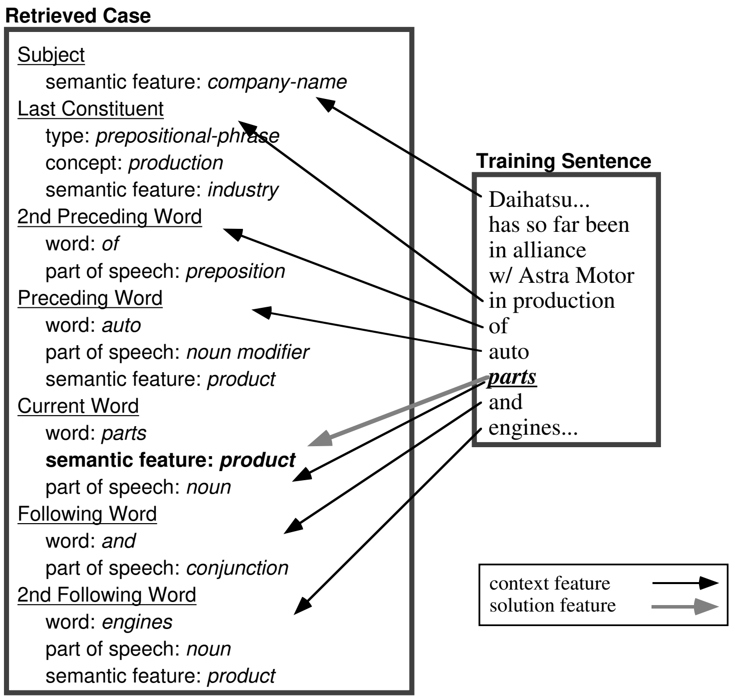

Figure 1.8: Retrieved Case for "tote" and the Training Sentence that Spawned It.

- *Kenmore's approach requires relatively little training.* In addition to noting the presence of individual words, cases in Kenmore encode relatively high-level information describing the state of the parser as well as the syntactic and semantic classes associated with both words and constituents. As a result, we obtain promising results with Kenmore after training on a relatively small number of sentences. This makes the method especially appropriate for use with small corpora where word-based statistical approaches fail due to lack of data.

- *Training does not require the expertise of computational linguists.* With the possible exception of part-of-speech tagging, supervisory information can be provided by anyone who can understand texts from the training corpus.

- *The case-based reasoning paradigm allows for incremental learning of disambiguation heuristics.* This feature allows Kenmore's training phase to become progressively easier for the human supervisor because Kenmore can access the existing case base to suggest solutions to each ambiguity and rely on the supervisor only to override incorrect predictions.

- *The machine learning approach to knowledge acquisition provides a flexible control structure for combining multiple sources of knowledge.* Thus, Kenmore fosters exploration of the usefulness of different types of knowledge for solving a particular problem in sentence analysis.

1.5 Claims and Contributions of the Thesis

The goal of this thesis is to address the knowledge-engineering bottleneck for natural language processing systems. To this end, it will present the Kenmore knowledge acquisition framework for natural language processing systems and make the following claims:

1. *The Kenmore framework is able to address uniformly a range of problems in sentence analysis each of which traditionally had required a separate computational mechanism.* In particular, a single architecture:

    - handles both syntactic and semantic ambiguities,
    - handles ambiguity at both the lexical and structural levels, and
    - accommodates both the acquisition and application of disambiguation heuristics.

2. *Kenmore demonstrates that symbolic inductive machine learning and case-based techniques can be used to learn solutions to significant problems in sentence analysis and can be incorporated into a working NLP system that has demonstrated success in processing real-world text.* As such, it demonstrates the feasibility of truly trainable, portable, customized sentence analyzers.

To demonstrate support for these claims, we have used the Kenmore knowledge acquisition framework to learn lexical and structural parsing knowledge for corpora from two real-world domains. At the lexical level, Kenmore has been used to acquire part-of-speech, semantic feature, and concept activation knowledge for all open-class words (e.g., nouns, adjectives, verbs) in the JV corpus. More specifically, the system learns for this domain:

- the parts of speech, semantic features, and concepts that should be associated with each word, and
- a set of lexical disambiguation heuristics that discern which part of speech, semantic feature, and concept are appropriate for a word in a given context.

Kenmore's performance in these lexical acquisition tasks has been evaluated in a series of experiments. In particular, we first assume the existence of a nearly complete domain-specific dictionary and use Kenmore to infer the features of occasional unknown words that occur in input texts. Next, we present Kenmore with a much more ambitious task. We assume only a small dictionary of function words (e.g., prepositions, auxiliaries, determiners) and use Kenmore to determine the definition of *all* non-function words in an incoming text from scratch.

At the structural level, Kenmore has been used to learn heuristics for locating the antecedents of relative pronouns. Here we used texts that describe Latin American terrorist events and found that the learned heuristics outperform a set of hand-coded heuristics that had been developed for use in the terrorism domain.

Finally, because Kenmore relies heavily on machine learning components, the thesis also addresses the issue of knowledge representation for machine learning systems. In particular, it is well known that the performance of any machine learning system depends critically on the representation of the training cases it receives as input. Unfortunately, finding a good case representation is a notoriously difficult and time-consuming task and presents another potential knowledge-engineering bottleneck during system development [Quinlan, 1983]. As a result, this thesis also presents two automated techniques for improving a baseline case representation. The first approach uses decision trees [Quinlan, 1986] to discard irrelevant features from a case representation. The second technique explicitly encodes any of a handful of well-known cognitive biases into the case representation.

1.6 What's to Follow

Related Work (Chapter 2) This chapter discusses prior work in the area of knowledge acquisition for natural language processing and distinguishes it from the work presented here.

Knowledge Acquisition for Conceptual Sentence Analysis (Chapter 3) The Kenmore framework for knowledge acquisition is parser-independent — the sentence analyzer component of the system can be replaced by any language processing system that provides an on-line model of sentence analysis and that can examine its state at any point in the parse.[6] Only the specifics of the case representation (i.e., the context and solution features) need to change in response to the types and forms of knowledge available to the new sentence analyzer. This thesis, however, emphasizes the acquisition of knowledge for the task of *conceptual sentence analysis*[7] [Riesbeck, 1975]. This chapter describes CIRCUS [Lehnert, 1990], the conceptual sentence analyzer used throughout

[6]Section 9.2 discusses the requirements of the sentence analyzer in more detail.

[7]Conceptual sentence analysis focuses on building a representation of the meaning of a sentence rather than a representation of the syntactic structure of a sentence.

this work. The chapter also specifies the state information maintained by CIRCUS during sentence analysis; this information ultimately comprises the context portion of Kenmore's case representation. Finally, the chapter discusses a number of knowledge acquisition tasks that are important for conceptual sentence analysis in general, and for CIRCUS in particular.

Domain-Specific Knowledge Acquisition (Chapter 4) Although the Kenmore framework for knowledge acquisition for natural language systems is also domain-independent, it requires a corpus from which training sentences are selected. As a result, the case base created during the acquisition phase implicitly encodes heuristics tuned to the training corpus. In short, Kenmore automates the acquisition of domain-specific ambiguity resolution heuristics. This chapter first discusses the notion of *domain-specific knowledge acquisition*. It then describes the domains of the two corpora used in all Kenmore experiments as well as the information extraction tasks for which the corpora were designed.

Learning Lexical Knowledge (Chapter 5) This chapter describes and evaluates **MayTag**, an instantiation of Kenmore for the acquisition of lexical knowledge.

Using Decision Trees to Discard Irrelevant Features (Chapter 6) This chapter presents and evaluates MayTag's decision tree approach to improving a baseline case representation. The technique uses decision trees [Quinlan, 1986] to locate the relevant features in a representation so that the irrelevant ones can be discarded.

MayTag Performance Analysis (Chapter 7) This chapter provides a deeper analysis of MayTag's ability to handle lexical ambiguity.

Finding Antecedents of Relative Pronouns (Chapter 8) This chapter describes **WHirlpool**, an instantiation of Kenmore that learns heuristics for finding relative pronoun antecedents. It also presents and evaluates WHirlpool's approach to finding an appropriate case representation — an automated method that explicitly encodes cognitive biases into a baseline case representation.

Establishing Constraints on the Kenmore Framework (Chapter 9) This section discusses constraints on the Kenmore framework that must be satisfied before successful instantiation of the framework on a new problem.

Conclusions (Chapter 10) This section summarizes the general conclusions of the work and outlines directions for future work.

The chapters have been organized and written so that different paths through the thesis are possible, depending on the reader's interests:

- For an *overview* of the work, read Chapters 1, 3, and one of 5 or 8.1-8.5.
- To focus on *lexical ambiguity tasks*, read Chapters 1, 3, 5, and 7.
- To focus on *structural ambiguity tasks*, read Chapters 1, 3, and 8.
- To concentrate on the thesis's contributions in *machine learning*, read Chapters 1, 6, and 8.6.

CHAPTER 2

RELATED WORK

To address the knowledge engineering bottleneck for NLP systems, this thesis presents Kenmore, a knowledge acquisition framework within which solutions to problems in sentence analysis can be learned. More specifically, however, Kenmore addresses the problem of *domain-specific* knowledge acquisition for *conceptual sentence analysis*. These constraints on the knowledge acquisition task lead to a number of implicit design goals that may not apply to the general language acquisition task. Namely, (1) semantic disambiguation at both the lexical and structural levels will be of particular importance, (2) the domain-specific nature of the task precludes the use of general lexicons[1], (3) any heuristics acquired by the system should be tailored for use with the corpus of interest, (4) it is better if domain experts can train the system rather than computational linguists or system designers, (5) the training phase should become easier for the human supervisor over time, and (6) the acquired knowledge bases and heuristics should have some capability for explaining their decisions.

Many of these design goals were mentioned briefly in Section 1.4, a summary of Kenmore's advantages over existing approaches to the knowledge engineering bottleneck for NLP systems. The purpose of this chapter is to provide a more detailed comparison of the Kenmore framework with existing work in knowledge acquisition for natural language systems. In particular, we have divided related work into seven separate areas: (1) hand-crafted knowledge acquisition, (2) intelligent interfaces to natural language systems, (3) statistical approaches, (4) knowledge-based approaches, (5) connectionist approaches, (6) machine learning approaches, and (7) case-based approaches. We treat each area in turn in the following sections.

2.1 Hand-crafted Knowledge Acquisition

Generating lexical knowledge, lexical disambiguation heuristics, and structural disambiguation heuristics by hand is clearly a time-consuming and tedious task. The ability to acquire knowledge about words and to encode this knowledge in the lexicon has

[1]On-line sources of linguistic and commonsense knowledge usually will not have detailed enough coverage in the area of interest.

been labeled by some as the major bottleneck for building natural language processing systems (e.g., Zernik [1991a]). In addition, the manual encoding of lexical and structural disambiguation rules — especially those that merge syntactic and semantic constraints — is particularly difficult and prone to error. Maintenance of rule sets becomes increasingly difficult over time. Moreover, hand-coded heuristics and lexicons are often incomplete and perform poorly in new domains comprised of specialized vocabularies or a different genre of text.

Until fairly recently, natural language processing systems relied exclusively on hand-crafted lexicons and knowledge bases. Early systems for conceptual sentence analysis (e.g., ELI [Riesbeck and Schank, 1978] and QA [Lehnert, 1978]) embedded much of their background knowledge in hand-coded lexicons. Other text analyzers relied on detailed, hand-coded knowledge of scripts, plans, goals, and even political beliefs, in addition to hand-coded lexicons (e.g., SAM [Schank and Riesbeck, 1981], FRUMP [DeJong, 1979], PAM [Wilensky, 1978], and POLITICS [Carbonell, 1978]). The Boris system [Dyer, 1983] included all of these knowledge structures (and more) and integrated them directly into the parser. Martin's DMAP system [Riesbeck and Martin, 1985], which processed texts using a memory-based approach to language understanding, relied on a hand-coded representation of event memory. In general, reliance on hand-crafted background knowledge was possible in these systems because the domains and tasks for which they were designed were very narrow in scope, and the texts processed by the systems for the most part contained short, simple sentences of limited syntactic structure. Conversely, the need for extensive hand-coded background knowledge *demanded* that the domains and tasks of these systems remain limited and narrowly focused. More recently, NLP systems have begun to tackle relatively ambitious language-processing tasks, for which reliance on hand-coded lexicons and hand-coded background knowledge is impractical.

It is partly in response to such scalability issues that Kenmore was created. Kenmore provides an automated mechanism for acquiring knowledge for sentence analysis and tunes that knowledge to a particular domain. It requires human intervention only during the training phase and thus avoids many problems associated with human-generated heuristics. There is no explicit rule base to maintain. In addition, algorithms used within Kenmore's architecture provide access to a simple knowledge representation and control structure that together naturally handle the assimilation of syntactic and semantic knowledge. Most importantly, the same basic framework accommodates the acquisition of a wide variety of knowledge needed for conceptual analysis of texts.

2.2 Intelligent Interfaces

One approach to the problem of lexical knowledge acquisition is to design intelligent interfaces to help computational linguists assign syntactic and semantic tags to individual words in a corpus. Interfaces increase the speed with which knowledge can be acquired and also help avoid some problems associated with the manual encoding of such knowledge. The Kenmore framework for knowledge acquisition, however, requires much less human intervention than interfaces that demand human perusal of possibly an entire corpus — Kenmore is able to generalize from the relatively small set of examples it is given during training. In addition, both structural and lexical ambiguity resolution are handled within the Kenmore framework. Smart interfaces, on the other hand, assist the user in tagging or bracketing a corpus (see Marcus [1991]), but do not generally assist in the acquisition and maintenance of disambiguation heuristics. Notable exceptions are interfaces that help the user define semantic case frames and their associated selectional restrictions [Ayuso *et al.*, 1987, Grishman *et al.*, 1986] — structures that implicitly perform some lexical and structural disambiguation. Traditionally, this type of knowledge is crafted exclusively by hand.

An intelligent interface with some similarities to Kenmore is the Simmons and Yu system [Simmons and Yu, 1992]. They present a simple user interface for acquiring context-dependent part-of-speech tags and context-dependent grammars as well as an algorithm for applying the acquired rules. Like the Simmons and Yu system, Kenmore also acquires context-dependent parsing knowledge; however, Kenmore is designed more for the acquisition of disambiguation heuristics than for the acquisition of grammars. In addition, we rely on machine learning methods for retrieval of similar training cases during the application phase while the Simmons and Yu system indexes all acquired rules based on the top two items of the current stack of a shift-reduce parser. A nearest-neighbor algorithm is then used to locate the best of the retrieved cases. Finally, Kenmore allows the use of semantic as well as syntactic knowledge (if that knowledge is available to the parser), while Simmons and Yu allow only syntactic knowledge in the acquired rules. Like Kenmore, the Simmons and Yu system uses relatively few training examples. In their evaluation, a case base of over 8000 cases was created in response to the manual tagging of 345 sentences. The system correctly predicts the part of speech 99.52% of the time when testing on the training set; choosing the most frequent part of speech achieves an accuracy of 97.52%. No results were posted for novel test sets.

2.3 Statistical Approaches

2.3.1 Acquisition of lexical knowledge

Statistical methods that acquire lexical knowledge have enjoyed recent success. For the most part, work in this area has concentrated on automated part-of-speech tagging using Markov model-based stochastic taggers (e.g., Jelinek [1985], Church [1988], DeMarcken [1990], Charniak [1993]). These systems predict the part of speech of the current word, given the part of speech assigned to the preceding n words.[2] They essentially choose the tag sequence that maximizes the following equation across all words of the sentence:

$$Probability(word|tag) * Probability(tag|preceding\ n\ tags)$$

[2]If $n = 2$, then the model is a bigram model; if $n = 3$, then a trigram model, etc.

In general, Markov-based tagging algorithms achieve accuracies in the 95% correct range for part-of-speech tagging of unrestricted text [Church and Mercer, 1993, Charniak, 1993]. It has been postulated that these methods succeed where traditional rule-based approaches fail because they manage to account for lexical preferences (e.g., "bird" is generally used as a noun rather than a verb).

Unfortunately, statistical methods require the existence of extremely large, often hand-tagged training corpora. For example, one corpus that is often used to construct statistical models for natural language acquisition tasks is the 4.5 million-word Penn Treebank [Marcus *et al.*, 1993]; the smaller Tagged Brown Corpus (one million words) is considered too small for many tagging tasks [Church *et al.*, 1991]. However, corpora of even 150 million words have been found to be inadequate for some statistical approaches to ambiguity resolution [Dagan and Itai, 1991]. Lexical ambiguity resolution in Kenmore, on the other hand, incorporates lexical preferences into the learned heuristics (whenever word-level information is included as part of a case) and also includes higher-level information regarding the semantic classes of surrounding words and the (syntactic and semantic) state of the parser. We believe that it is this access to higher-level knowledge sources that allows Kenmore to succeed with relatively little training.[3] As a result, Kenmore is particularly well-suited to learning lexical knowledge in domains where large amounts of annotated text are unavailable. In addition, Kenmore allows evidence from multiple knowledge sources to be combined without the complex modeling of dependencies that is often required in statistical approaches.

Another advantage of Kenmore over statistical approaches to lexical ambiguity resolution is its explanation capabilities: Kenmore captures information about lexical ambiguity resolution in the individual cases rather than in very large tables of statistics. This allows the system to provide a more compelling explanation of its decisions than purely statistical methods. The rule-based part-of-speech tagger presented in Brill [1992] also has this advantage. His system acquires a set of ordered, non-stochastic rules using transformation-based error-driven learning [Brill, 1993]. To learn a set of rules, the system requires an annotated corpus, a method for assigning a word its most likely tag, and a set of allowable transformations.[4] The system tries every possible transformation, counts the errors caused by each, chooses the transformation that best reduces the overall error rate on the corpus, and adds it to the end of the ordered list. This cycle continues until the error rate falls below some preset threshold. To tag a new sentence, the most likely tags are assigned to each word and all rules are applied to each word, in turn. Unlike most corpus-based

[3] Evidence that incorporation of such knowledge can reduce sparse data problems can be found in Fisher and Riloff [1992].

[4] Examples of transformations are: (1) If the current word has tag a and the preceding (following) word has tag b, then change a to z; (2) If the current word has tag a and the preceding word has tag b and the following word has tag c, then change a to z.

approaches to part-of-speech tagging, the acquired rule set directly and explicitly captures relevant linguistic knowledge while performing as well as stochastic taggers.

Like Kenmore, Brill's approach has also been applied to a number of language learning tasks — part-of-speech tagging, grammar acquisition, and prepositional phrase attachment. There are a number of differences in the two systems, however. Brill assumes that a new a set of transformations will be specified for each new ambiguity task. The equivalent task in Kenmore is the specification of the attributes that should be part of each case. For Kenmore, these are parser-specific features rather than task-specific and need not be redefined for each new type of ambiguity. As a result, Kenmore's uniform view of ambiguity resolution gives the learning component access to the same knowledge sources for all types of ambiguity. In addition, the rules acquired using Brill's transformation-based approach access only lexical information for one or two words in a six-word window surrounding the unknown word. Because Kenmore's learning component is embedded within the parser, heuristics learned by the system can be more varied and flexible — they can access lexical information for any or all words in a small window centered on the unknown word and can also test features of the global state of sentence analysis. Finally, modifications to Brill's original system were required to incorporate lexical preferences and to handle unknown words or words that did not appear during training [Brill, 1994]. These situations are handled naturally within the Kenmore framework for knowledge acquisition.

Most statistical approaches to lexical ambiguity resolution have focused primarily on the acquisition of syntactic knowledge. However, a few systems have been designed to learn semantic knowledge for sentence analysis — or at least knowledge that is on the boundary of syntax and semantics. Brent has focused on the acquisition of semantic knowledge for verbs: (1) distinguishing stative from non-stative verbs[5] [Brent, 1990], and (2) detecting verbs with one of five subcategorization frames [Brent, 1991]. His systems are particularly interesting because they avoid the need for human supervision by searching for a predefined set of unambiguous syntactic contexts. Other work on verbs includes a method for inferring predicate argument relations using mutual information statistics [Church and Hanks, 1990]. Some progress has also been made in the area of word sense disambiguation. Brown *et al.* present a statistical method to label word senses of a word with the context in which they appear. In their work, the "context" consists of the words in a window surrounding the ambiguous word, but includes no higher level knowledge. In addition, their system assigns at most two senses to each word. Yarowsky [1992] presents a class-based variation of a Bayesian classifier for word sense disambiguation. Resnik [1993] uses large corpora to discover lexical relationships for use in selectional constraints.

Another general approach to knowledge acquisition for lexical ambiguity tasks that bears some similarity to Kenmore is a method for acquiring decision lists for lexical

---

[5]Stative verbs are verbs whose actions are assumed to hold at all times after their assertion (e.g., know, believe, love); the actions of non-stative verbs are not always assumed to hold after assertion (e.g., fix, walk).

ambiguity resolution [Yarowsky, 1994]. Yarowsky's decision lists are ordered lists of single-antecedent rules — each entry in the list tests one feature and produces a classification. Unlike Brill's transformation-based rule set, only the first applicable pattern in the list is applied to a novel instance. The decision lists are task-specific and are created by (1) measuring collocational distributions across a corpus for a predefined set of hand-crafted lexically-based collocations (e.g., the word to the right (left) of the current word or pair of words at offsets -2 and -1), and (2) sorting them by their log-likelihoods. As in most statistical approaches, the method has been tested on a very large annotated corpus, but not on the smaller, domain-specific corpora that Kenmore handles. In addition, the approach has been applied only to fairly low-level lexical ambiguity tasks (e.g., restoring accents in Spanish and French texts), although the approach could be extended to handle structural ambiguity tasks as well. The output of the system is a set of heuristics, each of which classifies a new instance by checking at most one piece of the available evidence. A surprising result is that this approach performs as well as more complex approaches that try to combine all available evidence. Kenmore's heuristics, on the other hand, fall somewhere in between, testing some, but not necessarily all, sources of syntactic and semantic knowledge.

All of the above approaches to lexical acquisition require supervision of some sort. However, a number of systems have begun to look toward unsupervised methods for finding clusters of related words (e.g., Brown *et al.* [1992], Grefenstette [1992]). While preliminary results are very promising, problems with ambiguous words and inconsistent clusters (see Brown *et al.* [1992]) make it difficult to incorporate the learned lexical knowledge into existing NLP systems, especially those that use specialized vocabularies and require domain-specific distinctions.

In summary, Kenmore differs from statistical approaches to lexical knowledge acquisition in a number of ways. First, syntactic and semantic lexical knowledge can be learned simultaneously within the same framework. Next, Kenmore requires a very small training corpus by allowing access to class-level and structural knowledge in addition to individual words. This makes Kenmore especially attractive for acquiring lexical knowledge for texts in specialized domains where only small, untagged corpora are available. Finally, Kenmore learns knowledge in a form directly usable by the sentence analyzer.

2.3.2 Acquisition of structural knowledge

Statistical approaches have been used for at least one structural disambiguation task — prepositional phrase attachment [Hindle and Rooth, 1993, Resnik and Hearst, 1993, Brill, 1993]. Like the statistical methods employed for lexical disambiguation, the systems that handle structural ambiguity also use very large corpora for training. Kenmore handles structural ambiguity using much less text by creating training cases based on parser output rather than only lexical cooccurrences. Kenmore also provides a better explanation of its ambiguity resolution decisions than most statistical approaches because it knows which training case is responsible for each ambiguity decision. This provides useful feedback for the system designer and, as discussed in Section 10.2.3, could be used to make Kenmore's training phase an active, rather than a relatively passive one. Statistical approaches to knowledge acquisition generally offer no such credit assignment mechanism.

2.3.3 Comprehensive approaches to ambiguity resolution

In addition to Brill's transformation-based system for learning solutions to a variety of problems in natural language understanding (see above), BBN's PLUM system offers another statistically-based, comprehensive approach to knowledge acquisition for sentence analysis [Weischedel *et al.*, 1993]. Like Kenmore, PLUM handles lexical ambiguity, unknown words, and structural ambiguity for real-world text. It also espouses a hybrid approach to knowledge acquisition and text processing by integrating probabilistic models with knowledge-based parsing methods. Nevertheless, the two systems treat ambiguity resolution very differently. At the lexical level PLUM finds all part-of-speech tags for an entire sequence of words simultaneously.[6] At the structural level, the system deals with ambiguity *after* the parser and semantic interpreter have computed all plausible parses for the sentence. Only then is a context-free probabilistic model applied to choose among the available options. Kenmore, on the other hand, employs an on-line model of ambiguity resolution — it resolves both lexical and structural ambiguity dynamically as the parse proceeds. This on-line model makes Kenmore better suited to incorporate results from psycholinguistic experiments directly into its sentence processing strategies.

Both PLUM and Kenmore include a non-trivial supervised training phase. However, Kenmore's reliance on symbolic machine learning methods, and case-based methods in particular, allows learning to proceed incrementally — each decision made by the human supervisor is incorporated directly into the case base and, hence, is available for retrieval immediately. This means that supervision becomes progressively easier over time because the system learns with each training instance. Incremental learning is not feasible within PLUM's current architecture or, for that matter, within many of the statistical approaches described above. The incremental nature of Kenmore's learning algorithms also provides a built-in mechanism for knowing when to stop training — training can end when the current case base has begun to provide supervisory information at an acceptable level. Weischedel *et al.* [1993] presents guidelines for when to stop training based on corpus statistics, rather than overall system performance.

Like the statistical approaches described above, PLUM also requires the derivation of different probability models for dealing with each new class of ambiguity as well as mechanisms for efficiently and accurately estimating the probabilities for those equations. Kenmore, on the other hand, handles all ambiguity uniformly without similar modifications.

---

[6] Although PLUM could be extended to handle lexical semantic ambiguity, it currently ignores that class of ambiguity.

2.4 Knowledge-Based Approaches

There exist a number of knowledge-intensive methods for the acquisition of syntactic and semantic lexical knowledge. These systems rely heavily on either hand-coded background knowledge (e.g., Berwick [1983], Granger [1977], Hastings *et al.* [1991], Lytinen and Roberts [1989], Martin [1992], Selfridge [1986]) or detailed hand-coded heuristics that describe how and when to acquire new word definitions (e.g., Jacobs and Zernik [1988], Wilensky [1991]). Unfortunately, generating either world knowledge or acquisition heuristics is more difficult and time-consuming than generating the lexical knowledge itself. In addition, it is very likely that the background knowledge and rules for acquisition will be domain- or corpus-dependent and will have to be regenerated or at least customized for use with texts in new domains. Finally, we know of no knowledge-based approaches for the automatic acquisition of structural disambiguation heuristics.

Kenmore, on the other hand, acquires lexical knowledge without the use of either world knowledge or explicit lexical acquisition heuristics and also handles the acquisition of structural disambiguation heuristics.

2.5 Connectionist Approaches

2.5.1 Distributed Approaches

Connectionist techniques have been applied to natural language processing for a number of years without much success in terms of real-world progress. Many have addressed the problem of full sentence analysis or grammar learning using distributed connectionist methods (e.g., Elman [1990], Jain [1989], McClelland and Kawamoto [1986], Miikkulainen and Dyer [1987], Miikkulainen and Dyer [1989], St. John [1992]), but each of these systems only processes sentences of limited syntactic structure in very narrow, toy domains, and scaling the systems to real-world domains does not seem imminent. Like the statistical approaches to knowledge acquisition and the localist approaches described below, these connectionist systems tend to have poor explanation capabilities.

A number of systems that combine traditional parsing techniques and distributed connectionist modules have also been developed for sentence analysis (e.g., Wermter [1990], Wermter and Lehnert [1989], Chen *et al.* [1993]). Like Kenmore, these systems integrate a traditional parser with separate learning components, each of which specializes in the resolution of a single class of ambiguity. In theory, these systems could be extended to handle real-world text and additional structural ambiguities, but they currently tackle only noun phrase analysis within limited syntactic constructs. Work in this area has also focused purely on learning heuristics for structural rather than lexical disambiguation. In addition, none of the connectionist systems described above provides a capability for incremental learning. Incorporating a single training instance requires retraining the entire network, which can take hours or even days. In comparison, the symbolic machine learning systems used in Kenmore, particularly the case-based and instance-based algorithms, can incorporate new instances with very little computation.

2.5.2 Localist Approaches

Like Kenmore, localist connectionist approaches to word sense disambiguation (e.g., Cottrell [1986]), sentence analysis (e.g., Waltz and Pollack [1985]), and high-level inferencing (e.g., Lange and Dyer [1989]) provide a unified approach to problems of disambiguation. Again, however, these approaches either handle only extremely simple sentence structures or assume large amounts of world knowledge (or both). More importantly, they provide no mechanisms for learning this background knowledge, for acquiring knowledge about how to structure the underlying networks, or for remembering decisions from one run to the next. Kenmore provides a framework for both learning and applying knowledge required for sentence analysis.

2.6 Machine Learning Approaches

There have been relatively few systems that use standard symbolic machine learning techniques to learn knowledge for natural language processing systems. Prior work in this area has concentrated mainly in the application of explanation-based learning (EBL) methods to grammar learning (e.g., Samuelsson and Rayner [1991], Samuelsson [1991]). The work presented here, on the other hand, focuses on learning knowledge for natural language systems that process text *without* the use of formal grammars. Although EBL has also been used for low-level language tasks like learning phonological representations [Stetham, 1991] and higher level tasks like learning causal relationships [Bozsahin and Findler, 1992, Pazzani, 1991], it has not been used to learn solutions to the lexical and structural disambiguation tasks handled within the Kenmore architecture. In addition, the EBL systems mentioned above generally require extensive background knowledge in the form of commonsense knowledge, detailed semantic hierarchies, or causal theories. The Kenmore framework presumes no such background knowledge.

Inductive logic programming (ILP) techniques [Muggleton, 1992] have only recently been employed to learn case-role mappings (e.g., Zelle and Mooney [1993]) and to learn rules from parsed text (e.g., Delannoy *et al.* [1993] and Matwin and Szpakowicz [1993]). While learning within the realm of logic programming is appropriate for certain NLP systems, few conceptual sentence analyzers translate text into the formal logic format required for some ILP techniques. This translation often precludes the kind of robust parsing needed to process real-world text. In very recent work, however, Zelle and Mooney [1994] have used ILP techniques to generate Prolog parsers from real-world text, making ILP an exciting new area of resesarch in machine learning and natural language processing.

Finally, Riloff [1993] has used one-shot learning in her AutoSlog system to learn one type of knowledge structure for conceptual sentence analysis — semantic case frame definitions. AutoSlog differs from all of the above systems in that it relies on a small set of hand-coded rules that are, for the most part, domain-independent, to propose semantic case frame definitions for perusal by a human supervisor, who then decides which definitions to keep and which to discard.

2.6.1 A Comparison with SOAR

Kenmore has been presented as an architecture in which solutions to a number of problems in sentence analysis can be acquired. As such, it may seem similar to SOAR [Laird, 1987, Newell, 1990] — a general architecture for building intelligent systems. Both systems, for example, are able to handle seemingly very different problems within a single framework; both systems incorporate a learning component. Nevertheless, the two systems vary tremendously. First, SOAR provides a general problem-solving architecture. Kenmore is not nearly as general an architecture — it was designed solely for the acquisition of knowledge needed for natural language processing. Natural language understanding, for instance, is just one of the tasks that can be handled within SOAR [Lehman *et al.*, 1993]. Second, all aspects of the SOAR architecture are based on a set of assumptions regarding the cognitive nature of problem solving; cognitive biases and cognitive limitations are built directly into the system architecture. Kenmore, on the other hand, incorporates all cognitive biases and constraints into its case representation rather than in the case-based architecture itself (see Chapter 8). Third, Kenmore assumes that a person provides supervisory information. When needed, SOAR receives supervisory information through interaction with the world (or the system's model of it). Finally, SOAR learns solely via *chunking*, or remembering useful sequences of rule firings and combining them into a single rule. Kenmore learns by storing new cases in its case memory — it remembers individual parsing actions rather than a sequence of actions. Also, SOAR always generalizes chunks before adding them to long-term memory, but Kenmore's cases are never generalized prior to storing them in the case base.

2.7 Case-Based Approaches

Kenmore views knowledge acquisition for natural language processing systems from a case-based reasoning perspective. As a result, Kenmore can be compared to existing CBR systems along a number of dimensions including:

Case Representation: Like HYPO [Ashley, 1990, Rissland and Ashley, 1986], Kenmore uses a simple attribute-value pair case representation rather than a more structured case representation (e.g., the semantic network case representation of GREBE [Branting and Porter, 1991, Branting, 1991]).

Case Indexing: In handling lexical ambiguities, Kenmore indexes cases using a decision tree subsystem. In particular, this aspect of the system is related to HYPO's *dimensions* [Rissland *et al.*, 1984], ReMind's prototypes and inductive indexing [Cognitive Systems, 1992], and CYRUS's E-MOPs [Kolodner, 1983]. (See Chapter 6.)

Case Retrieval: Kenmore's case retrieval system is based on a nearest-neighbor similarity metric and, hence, is much simpler than the domain-dependent and knowledge-based case retrieval algorithms typically employed in CBR systems (e.g., CHEF [Hammond, 1989]).

Best Case Selection: Kenmore uses a simple voting scheme to derive a solution from the retrieved cases rather than using more sophisticated case selection methods (e.g., HYPO [Ashley, 1990, Rissland and Ashley, 1986] and GREBE [Branting and Porter, 1991, Branting, 1991]).

Case Adaptation: Unlike most CBR systems, Kenmore uses only a very weak form of case adaptation to modify the retrieved cases to better fit the context of the problem case. (See Section 8.4.)

The components of Kenmore's case-based inductive learning component reflect the goals of our approach — to acquire the knowledge needed for sentence analysis **without** relying heavily on handcrafted heuristics and handcrafted background knowledge: (1) Kenmore's simple case representation facilitates the automatic generation of cases by the parser; (2) case indexes are learned rather than handcoded; (3) no domain-specific knowledge is used (or available) for case selection or case adaptation.

In general, Kenmore's knowledge acquisition task is one of interpretation: Given an ambiguous word or phrase, Kenmore must provide a syntactic and semantic interpretation of the word or phrase that is consistent with the current context. This makes Kenmore an *interpretive CBR system* — a system that analyzes a new case by comparing and analogizing it with previous interpretation episodes. Interpretive CBR systems exist in a number of domains: (1) The HYPO system, for example, [Ashley, 1990, Rissland and Ashley, 1986] creates legal arguments for a problem case in the domain of trade secret disputes; (2) GREBE [Branting and Porter, 1991, Branting, 1991] and CABARET [Rissland and Skalak, 1991] analyze problems in the domain of legal reasoning; (3) ANAPRON's [Golding and Rosenbloom, 1991] interpretive task is word pronunciation. The remainder of this section of the thesis, however, focuses specifically on related work from CBR in the area of knowledge acquisition for NLP systems. Additional comparisons to particular components of existing CBR systems will be provided in the chapters that describe the details of Kenmore's case-based inductive learning algorithm (Chapters 5 and 6).

Although others have addressed language learning within the case-based paradigm, prior research in the area has focused on learning fairly low-level knowledge rather than the higher level parsing knowledge acquired within the Kenmore framework. Stanfill and Waltz [1986], for example, used a case-based approach in their MBRtalk system that learns how to pronounce English words. More recently, ANAPRON [Golding and Rosenbloom, 1991] tackled the problem of surname pronunciation using an architecture that combines rule-based and case-based reasoning. Additional work in low-level language acquisition has been done by researchers from the ATILA project — a joint effort between Antwerp and Tilburg Universities to study the process of human language acquisition by applying machine learning algorithms to a variety of language learning tasks. They have found that instance-based approaches to language acquisition work well for low-level language tasks like stress acquisition [Daelemans *et al.*, 1994] and grapheme-to-phoneme conversion

[Bosch and Daelemans, 1993] and have hypothesized that they will also perform well for higher level tasks. In addition, they argue that language acquisition is a behavior-based, data-driven process rather than one guided by knowledge-based principles and parameter setting. Like the work presented here, the authors view many problems in NLP as categorization problems, to which case-based learning techniques can be applied [Daelemans, to appear].

A commercial system that uses a case-based architecture to acquire parsing knowledge is Parse-O-Matic [Goodman, 1991]. Parse-O-Matic is a conceptual sentence analyzer that builds semantic case frame representations directly from simple requests. The sentence analyzer used in Parse-O-Matic is a rule-based system that builds a semantic representation of the input by creating, linking, and modifying (partial) case frames in response to words in the input stream. The rules appear to be domain-dependent, however, and a new set is required when the system is moved from one domain to the next. The goal of Parse-O-Matic was to develop a case memory that effectively replaces the rules of the system. In short, when used to design a new NLP system, Parse-O-Matic was able to cut knowledge engineering time in half while losing none of the performance of the rule-based system.

Cases in Parse-O-Matic are much more complicated than those used in Kenmore. The "context portion" of each case consists of the current word, any previous cases generated for earlier parts of the sentence, and all active semantic representations for the sentence. The "solution" portion of a case contains a case frame representation of the actions to apply to the active semantic representations in response to the current word. In effect, each case represents a rule. Cases are created in a partially automated fashion. The system supplies the context portion and the user helps the system to create the solution portion by modifying existing knowledge structures. Some of the hand-crafted knowledge structures that may be modified include (1) case adaptation routines, (2) the lexicon, (3) a set of conceptual hierarchies, (4) case frame definitions. In one sense, the system is an intelligent interface for designing natural language processing systems (although the final system is entirely automatic).

Because cases are so complex, indexing them for retrieval becomes a problem. Parse-O-Matic relies on an automated, inductive, statistical indexing scheme based on Hartigan's interaction detection algorithm [Hartigan, 1975]. In addition, it relies on a set of hand-crafted heuristics that decide which cases and which features of the current case to present to the clustering algorithm. It is the indexing scheme that generalizes the cases/rules for more flexible retrieval.

Kenmore differs from Parse-O-Matic in a number of ways. First, Kenmore currently relies on case-based approaches only to handle ambiguities, not to perform all functions for sentence analysis. Second, Parse-O-Matic requires extensive hand-coding of all background knowledge required by the CBR component and the parser. It is not at all clear how much, if any, of this knowledge is reusable when Parse-O-Matic must tackle a new language processing task in a new domain. Finally, Kenmore has been shown to work with real-world corpora that contain long, complicated sentences rather than the short, highly structured sentences handled in Parse-O-Matic.

Another CBR system that is similar to Kenmore in its focus on knowledge acquisition is PROTOS [Bareiss, 1989, Bareiss *et al.*, 1989]. PROTOS, however, is not designed specifically for knowledge acquisition in the domain of natural language processing. Instead, it provides a general architecture for knowledge acquisition and has been employed in a number of domains including the diagnosis of hearing disorders. Like Kenmore, however, PROTOS relies on case-based reasoning techniques to perform classification as well as knowledge acquisition. Given a new set of symptoms and laboratory test results, PROTOS retrieves the best-matching training case and uses its classification as the diagnosis in the new situation. During training, a human expert monitors the decisions of PROTOS and intervenes when errors occur. Unlike Kenmore, however, the human expert is responsible for supplying the missing knowledge that caused the error as well as the correct classification. This step would be difficult to implement in a natural language context where (1) the supervisor may not know what knowledge was missing, or (2) the problem may not have been due to missing knowledge in the first place. PROTOS also differs from Kenmore in that it relies extensively on a highly connected case library and on background knowledge in the form of a domain model. The domain model includes important functional, correlational, causal, and taxonomic relationships and specifies how and when the features of two cases should be considered a match. In addition, knowledge acquisition in PROTOS is driven solely by its mistakes.

2.8 Summary

In summary, Kenmore's machine learning and case-based approach to knowledge acquisition is especially useful for addressing the problem of domain-specific knowledge acquisition for conceptual sentence analysis. When the natural language system must process complex, real-world text, it is not feasible to generate the necessary background knowledge by hand or even with the help of an intelligent interface: the process is too slow and subsequent maintenance of the knowledge bases is difficult and generally requires the expertise of the NLP system designers. In addition, knowledge-based approaches to language acquisition do not necessarily avoid the knowledge acquisition bottleneck because the design and maintenance of the necessary background knowledge and of procedures that specify when and how to acquire new background knowledge is extremely difficult. Statistical approaches to knowledge acquisition, nevertheless, appear to work well for many natural language tasks. However, we have concentrated on knowledge acquisition for domain-specific tasks where the large, tagged corpora generally required for stastistical approaches are not readily available. To avoid the need for huge amounts of manually tagged text, Kenmore instead incorporates its inductive language-learning component within the parser, giving the learning algorithms access to higher level knowledge regarding the progress of sentence analysis. One final advantage of the Kenmore framework is its ability to handle many subproblems in sentence analysis uniformly and within the same architecture.

CHAPTER 3

KNOWLEDGE ACQUISITION FOR CONCEPTUAL SENTENCE ANALYSIS

As described briefly in the introduction, the Kenmore framework for knowledge acquisition is compatible with a variety of language processing theories and parsing frameworks. It requires only that the natural language system used as Kenmore's sentence analyzer component have the ability to examine its current state at the point of an ambiguity. Whenever the sentence analyzer component is replaced by a new one, only the underlying case representation needs to change to conform to the type and form of knowledge available in the new sentence analyzer. Despite the general applicability of the framework, however, this work emphasizes the acquisition of knowledge only for the task of *conceptual sentence analysis* [Riesbeck, 1975].

This chapter first describes the conceptual sentence analyzer called CIRCUS [Lehnert, 1990] that will be used in all experiments throughout this work. It is important to understand generally how CIRCUS processes sentences for two reasons: (1) because the *context* portion of each case in Kenmore's case-based reasoning component depends on the state of the CIRCUS parser, and (2) because CIRCUS's approach to sentence analysis constrains the types of knowledge that must be acquired before CIRCUS can process a text. The chapter begins by describing how CIRCUS understands simple and complex sentences (Section 3.1) and then provides examples of CIRCUS's state representation at different points in a sentence (Section 3.2). The chapter concludes with a summary of the kinds of knowledge necessary for parsing within the CIRCUS conceptual sentence analysis framework (Section 3.3) and a list of the specific knowledge acquisition tasks that will be addressed in later chapters (Section 3.3.4).

3.1 The CIRCUS Conceptual Sentence Analyzer

The goal of conceptual analysis is to produce a representation of the meaning of a text rather than a representation of its syntactic structure. Because this goal differs substantially from that of syntactically-oriented parsers, the relative role of syntactic and semantic analysis in conceptual sentence analyzers has been disputed (see Lytinen [1984]).

Traditionally, however, conceptual analyzers employ no distinct syntactic grammar and reflect a completely integrated theory of natural language interpretation (e.g., Birnbaum and Selfridge [1981], Cullingford [1986], Dyer [1983], Lebowitz [1980], Riesbeck [1975]). They not only allow syntactic and semantic processing to happen at the same time, but intertwine the parser's morphological, syntactic, semantic, and pragmatic knowledge in a monolithic and largely procedural representation.

CIRCUS [Lehnert, 1990] is a conceptual sentence analyzer that synthesizes three distinct processing techniques to produce a semantic case frame representation of an input sentence. It uses (1) a stack-oriented control for syntactic analysis, (2) a marker-passing design for predictive preference semantics, and (3) lexically-indexed control kernels [Cardie and Lehnert, 1991] for handling embedded clauses.[1] Although CIRCUS makes no commitment to a particular style of semantic representation, we use "deep" semantic case frames of the sort found in *conceptual dependency* [Schank, 1975]. In general, CIRCUS recognizes low-level syntactic constituents like noun phrases, verbs, and prepositional phrases, and consults semantic knowledge prior to making any attachment decisions. In addition, CIRCUS explicitly attaches constituents only to the semantic representation of the sentence. Syntactic attachments occur implicitly as a side effect of the semantically-driven modifications. The CIRCUS system has been used successfully to provide natural language processing capabilities for a variety of projects including summarization of wire service texts [Lehnert *et al.*, 1993a, Lehnert *et al.*, 1992, Lehnert *et al.*, 1991], text classification [Riloff and Lehnert, 1992], and the analysis of citation sentences in research papers [Lehnert *et al.*, 1990]. Its components will be described in the next three sections.

3.1.1 Syntactic Processing in CIRCUS

CIRCUS's stack-oriented syntactic analyzer segments incoming text into constituent phrases. In the tradition of conceptual analyzers, this component produces no parse tree of the input and employs no global syntactic grammar. It is based on the McEli parser [Schank and Riesbeck, 1981] and uses lexically-indexed, local syntactic knowledge to recognize noun phrases, prepositional phrases, and verb phrases. These constituents are stored in global buffers that track the subject, verb, direct object, indirect object, and prepositional phrases of a sentence. Like lexical-functional grammars [Bresnan, 1982], CIRCUS stores the syntactic predictions associated with each word in the lexicon and retrieves them during sentence analysis. Because we restrict the buffer contents to simple syntactic structures with a strongly "local" sense of the sentence, larger constituents like clauses are not explicitly recognized by the syntactic component.

To process the sentence that begins,

---

[1] CIRCUS also employs a numerical relaxation algorithm to perform bottom-up insertion of unpredicted slots into case frames. This module is not important for the purposes of this work, however, and will not be discussed here.

John brought...

for example, CIRCUS scans the sentence from left to right, using its stack-oriented control and the lexically-indexed syntactic predictions to assign words and phrases to syntactic constituents. Initially, the stack contains a single prediction — the hypothesis for a subject of the sentence. When CIRCUS sees the word "John," it accesses its part-of-speech lexicon, finds that "John" is a proper noun, loads the standard set of syntactic predictions associated with proper nouns onto the stack, and recognizes "John" as a noun phrase (NP). Because the presence of a noun phrase satisfies the initial prediction for a subject, CIRCUS then places "John" in the subject buffer (*S*) and pops the satisfied syntactic prediction from the stack. Next, CIRCUS processes the word "brought," finds that it is a verb, and assigns it to the verb buffer (*V*) as shown in Figure 3.1. In addition, the current stack contains

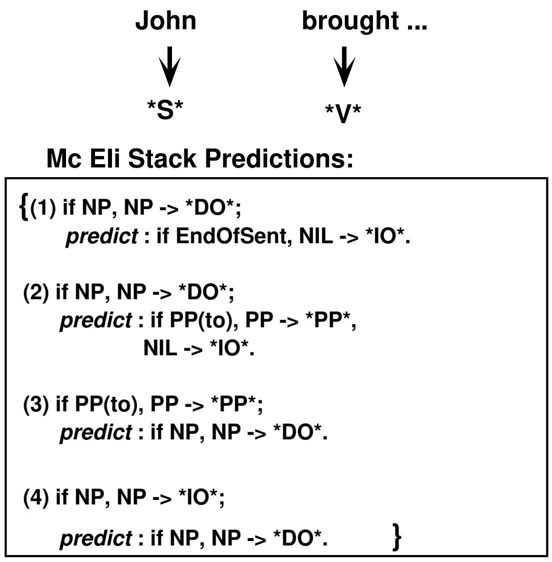

Figure 3.1: CIRCUS Status After "John brought..."

a single packet encoding the syntactic expectations associated with "brought."[2] This verb predicts (1) a direct object (DO), (2) a direct object followed by a "to" prepositional phrase (PP), (3) a "to" prepositional phrase followed by a direct object,[3] or (4) an indirect object (IO) followed by a direct object. The following sentences illustrate each possibility:

1. John brought a cake.

[2]Each prediction in a packet is called a request. Whenever one request in the topmost packet on the stack is satisfied, the entire packet containing the request is popped from the stack and all subsequent predictions associated with the request are pushed onto the stack in a new packet.

[3]Although this sentence is ungrammatical, it is still possible to ascertain its intended meaning. As a result, we let CIRCUS derive a meaning representation for the sentence.

2. John brought a cake to the party.
3. *John brought to the party a cake.[4]
4. John brought Mary a cake.

If the next word in the sentence were the noun phrase "Mary," for example, CIRCUS would assign "Mary" to *both* the direct object and the indirect object buffers and update its stack of syntactic expectations. The new predictions resolve the momentary syntactic ambiguity by overwriting the contents of either *DO* or *IO* depending on the phrase that follows "Mary" in the sentence.

3.1.2 Predictive Semantics in CIRCUS

As soon as CIRCUS recognizes a syntactic constituent, that constituent is made available to the mechanisms performing predictive semantics. The predictive semantics module (PSM) is responsible for making case role (or thematic role) assignments. In CIRCUS, this consists of top-down slot-filling for any active semantic case frames.[5] Whenever a syntactic constituent becomes available in one of the global buffers, PSM examines any active case frame that expects a slot filler from that buffer. PSM then fills the slot if the constituent satisfies the slot's semantic constraints. CIRCUS allows both hard and soft slot constraints. A hard constraint is a predicate that *must* be satisfied. In contrast, a soft constraint defines a preference for a slot filler rather than a predicate that blocks slot-filling when it is not satisfied. Consider, for example, the semantic case frame for a PTRANS event triggered by the word "brought" in the phrase "John brought" (Figure 3.2).[6] The case frame definition indicates the mapping between surface constituents and case frame slots: subject → Actor; direct object → Object; prepositional phrase or indirect object → Destination. In addition, it specifies a set of *enabling conditions* that describe the linguistic context in which the case frame should be triggered. In this case, the PTRANS case frame should be triggered by "brought" only when the verb occurs in an active construction. As in lexical-functional grammar (LFG), a different case frame definition (with a different set of enabling conditions) would be needed to handle a passive sentence construction.

Figure 3.2 also depicts the hard and soft constraints associated with each slot. Namely, the Actor should be animate, the Object should be a physical object (phys-obj), the

[4]A "*" will be used to mark those examples that either are difficult for a person to understand or display ungrammatical or atypical usage of English.

[5]Semantic case frames in CIRCUS represent the meaning, or "deep" structure of the sentence rather than its surface structure; slots in the case frames represent conceptual roles (e.g., actor and object) rather than syntactic positions (e.g., subject, direct object).

[6]PTRANS is a primitive act in conceptual dependency describing a physical transfer (see Schank [1975]). The PTRANS case frame actually has a fourth slot — the original location or *Source* of the object. For the purposes of this example, however, we will ignore this slot.

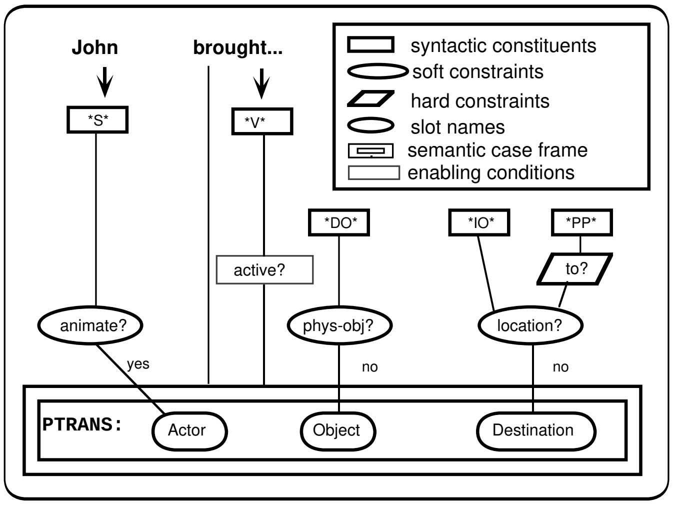

Figure 3.2: PSM Status After "John brought..."

Destination should be a location, and the prepositional phrase filling the Destination slot must begin with the preposition "to."[7] At this point in the parse, PSM successfully fills the Actor slot with "John" because "John" is the subject of the sentence and its entry in the lexicon indicates that "John" is animate. All of the other slots in the PTRANS frame remain empty.

When a frame satisfies certain instantiation criteria, PSM "freezes" the case frame with its assigned slot fillers. Any instantiated case frames then become part of the semantic representation CIRCUS derives for the sentence. Figure 3.3, for example, shows the PTRANS case frame instantiation returned by CIRCUS after parsing "John brought Mary to Manhattan."

Sentence: John brought Mary to Manhattan.

PTRANS
Actor = "John"
Object = "Mary"
Destination = "Manhattan"

Figure 3.3: Instantiated Semantic Case Frame.

[7]This is a hard constraint.

3.1.3 Handling Embedded Clauses in CIRCUS

When sentences become more complicated, CIRCUS has to "partition" the stack processing in a way that recognizes embedded syntactic structures as well as conceptual dependencies. Consider, for example, the embedded clauses in the following sentences:

(a) John asked Bill *to eat the leftovers*.

(b) That's the gentleman *that the woman invited to go to the show*.

(c) That's the gentleman *that the woman declined to go to the show with*.

Understanding these constructions requires that the parser infer the existence of an invisible or phonetically null constituent in the embedded clause and then associate the missing constituent with an antecedent phrase that may be arbitrarily distant from it. In (a), for example, the parser should infer that "Bill" is the subject of "eat"; in (b), "gentleman" is the direct object of "invited" as well as the subject of "go"; and in (c), "woman" is the subject of "go" while "gentleman" is the prepositional object of "with." This is accomplished in CIRCUS with lexically-indexed control kernels (LICKs) [Cardie and Lehnert, 1991].

We view the stack of syntactic predictions as a single control kernel whose expectations and binding instructions change in response to specific lexical items as we move through the sentence. When we come to a subordinate clause, the top-level kernel creates a subkernel that takes over to process the interior clause. In other words, when a subordinate clause is first encountered, the parent LICK spawns a child LICK, passes control over to the child, and later recovers control from the child when the subordinate clause is completed.

Each control kernel essentially creates a new parsing environment with its own set of bindings for the syntactic buffers, its own copy of the main stack, and its own predictive semantics module. The spawning of LICKs is deterministic — the current LICK can spawn exactly one child LICK at a time. Once control is returned to the parent from a child, however, the parent is free to spawn another child if necessary as the sentence progresses. To understand the behavior of multiple LICKs, we need only specify rules for passing control among LICKs and rules for passing variable bindings across LICKs:

Inter-LICK Control Rules:

1. An existing LICK can create a new LICK at which time control moves from the parent LICK to the child LICK.
2. When a child LICK relinquishes control, control reverts back to the parent LICK.

Inter-LICK Communication Rules:

1. When moving from a parent LICK to a child LICK, all syntactic buffers in the child LICK are initialized by the parent LICK. The buffers are initialized to *nil* unless otherwise specified by the parent.

2. When moving from a child LICK to a parent LICK, the only buffer that can be initialized or reassigned in the parent LICK is the *LB* buffer.

LB (lick buffer) is a special syntactic buffer used only for inter-LICK communication. Typically, the conceptual representation for an entire subordinate clause is stored in *LB* until it can be incorporated into the representation being constructed by a parent control kernel.

LICKs, then, embody the basic control mechanism of ATN's [Woods, 1970] but enforce a much stricter set of communication rules. The ATN framework, for example, provides at least three different mechanisms for handling embedded clauses [Winograd, 1983] — one possibility is to use a global *hold register* to store a constituent until its gap can be found. None of these mechanisms, however, allows the parent to instantiate more than a single syntactic buffer[8] in the embedded clause with the antecedent. Instead, ATN's rely on nondeterminism to allow an antecedent to fill one of a number of possible gap sites. In addition, CIRCUS's use of LICKs differs tremendously from the pervasive recursion of ATN's — CIRCUS employs the LICK mechanism only at the clause level and selectively triggers the mechanism via lexically-indexed signals. Unlike ATN's, the parsing of constituents within a clause remains deterministic and strictly bottom-up.

The next sections walk through specific examples that use LICKs to process relative clauses (e.g., I saw the man *who ate the ice cream cone*) and infinitival complements (e.g., John asked Bill *to go to the restaurant*).

3.1.3.1 Understanding Wh-Constructions

This section shows how sentences containing embedded wh-phrases (i.e., phrases introduced by a relative pronoun like "who," "whom," "that," "which") are handled by local syntactic predictions and interactions between cooperating LICKs. Consider the following sentence:

The policeman saw the boy who the crowd at the party accused of the crime.[9]

Figure 3.4 shows the state of CIRCUS after the word "who." The LICK processing the main clause has triggered a semantic case frame for SAW and has successfully filled its Actor and Object slots. In addition, the lexicon entry for "who" indicates that processing of the main clause should be temporarily suspended and a child LICK spawned. Because

[8]The ATN equivalents to CIRCUS's syntactic buffers in this case are called *role registers*.

[9]For most English dialects, "whom" is actually the grammatically correct relative pronoun to use in this example. We use an ungrammatical example for two main reasons. First, we want to emphasize the fact that CIRCUS should have the ability to understand sentences that people can understand, regardless of grammaticality. Second, we want to show that the LICK solution to understanding relative clauses handles ungrammatical input without sacrificing the ability to handle grammatical constructions.

the antecedent for "who" can bind to one of four possible syntactic constituents within the subordinate clause, CIRCUS initializes each of the child *S*, *DO*, *IO*, and *PP* buffers with "boy." (The issue of how CIRCUS can learn that "boy" is the correct antecedent of "who" will be addressed in Chapter 8.) When the child completes a semantic case frame

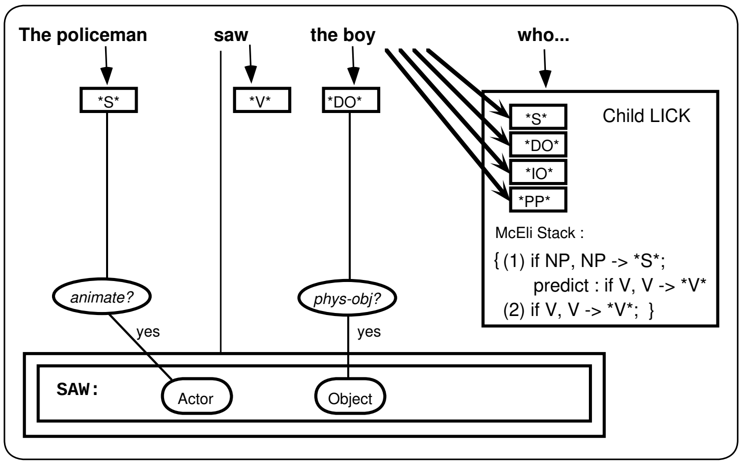

Figure 3.4: State of CIRCUS after "The policeman saw the boy who..."

instantiation, only the buffer associated with the *gap* (i.e., the missing or phonetically null constituent) should hold the *filler* (i.e., the antecedent). At the end of the example sentence, the filler "boy" should appear only in the direct object buffer — "the crowd accused [the boy] of the crime." The other buffers initialized with the antecedent will either be overwritten with actual phrases from the embedded clause or eliminated as possible gaps by syntactic information associated with the verb. In any case, few case frame definitions will access all four buffers. As indicated in the McEli stack of Figure 3.4, "who" also sets up syntactic predictions for either a verb phrase or a subject-verb sequence before passing control to the embedded clause LICK.

Figure 3.5 shows the state of the child LICK just after processing "accused." "Crowd" has overwritten *S* and "party" has overwritten *PP*.[10] In addition, "accused" activates a case frame and makes initial slot assignments based on the case frame definition: Actor = crowd and Patient = boy. The Accusation slot remains empty even though we have a prepositional phrase because the hard constraint that the preposition be "of" is violated.[11] Note that although both *IO* and *DO* contain the antecedent "boy," *IO* does not interfere

[10]Currently CIRCUS has only one prepositional phrase buffer (*PP*). The implication is that the parser only has access to the most recent prepositional phrase. Clearly, we should be using multiple buffers or a stack of *PP* buffers.

[11]As described above, a hard constraint is a predicate that must be satisfied before the filler is allowed to fill a slot.

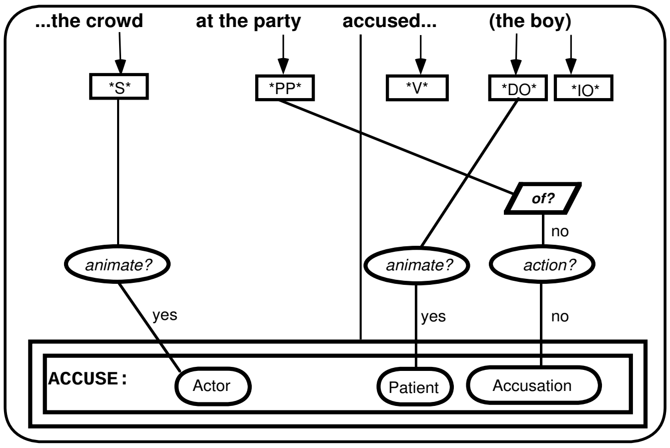

Figure 3.5: Child LICK Status after "The policeman saw the boy who the crowd at the party accused..."

with the semantic representation because the ACCUSE case frame does not access that buffer.

Figure 3.6 shows the state of the child LICK at the end of the embedded clause: Actor = crowd, Patient = boy, Accusation = crime. At this point, CIRCUS freezes the ACCUSE case frame, assigns the instantiated representation to the *LB* buffer, exits the child LICK, and returns control to the main clause where *LB* is attached to the antecedent "boy."

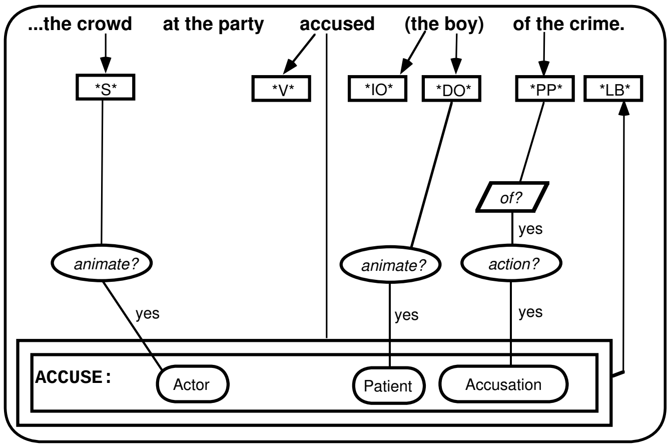

Figure 3.6: Child LICK Status after "The policeman saw the boy who the crowd at the party accused of the crime."

3.1.3.2 LICK Caveats and Conclusions

The last section illustrated the use of LICKs to process only wh-phrases. However, CIRCUS currently handles many types of embedded clauses using a small number of LICKs. For example, the LICK mechanism is responsible for handling some sentential complements (e.g., The peasants thought the president had been assassinated) and compound verb phrases (e.g., Castellar was kidnapped in Achi and taken by car to El Salvador). In particular, verbs that optionally predict an infinitival complement (e.g., "ask," "promise," "want")[12], are handled by LICKs that initialize the child subject buffer with the subject or direct object of the main clause depending on the constituent structure of the main clause. Consider the following sentences:

1. John asked $Bill_i$ $?_i$ to eat the leftovers.
2. $John_i$ asked $?_i$ to eat the leftovers.
3. $John_i$ asked Bill the question $?_i$ to shock the crowd.

In the first sentence, the presence of the verb "ask" followed by an indirect object and the word "to" causes a child LICK to be spawned that inherits the indirect object of the main clause as its subject. This *IO* inheritance makes "Bill" the Actor of "eat" in the embedded clause. If, on the other hand, there is no indirect object between "ask" and the infinitive (sentence 2), a LICK is triggered that clears all child buffers except *S*, which is inherited from the parental *S* buffer. In this case, the *S* inheritance makes "John" the Actor of "eat".[13] Finally, if the verb "ask" is followed by an indirect object and a direct object (sentence 3), no LICK is triggered by "ask" at all.

However, many embedded-clause problems cannot be resolved by the simple inter-LICK control rules and communication rules described here. Infinitives in the subject position (e.g., To win would be a welcome change) and infinitival indirect questions (e.g., It is unknown what to do in this case) are two examples. In both of these cases, the actor of the infinitive is implied. In addition, a reduced relative clause presents an ambiguity that must be resolved by either a parent LICK (in the case of an active past tense verb form) or a child LICK (in the case of a passive past participle verb form). The control kernel formalism encourages us to view this disambiguation problem in terms of competition for control, but does not suggest how that competition should be resolved. This approach to syntactic-semantic interactions simply recasts the problems of embedded constructions as issues concerning communication across scoping environments.

3.2 Examples of "Context" in CIRCUS

As described in Chapter 1, Kenmore relies on a case-based approach to knowledge acquisition where each "case" encodes a description of the context of an ambiguity as well as its solution in that context. This section provides concrete examples of the "context" available to CIRCUS as it processes a sentence. In general, the context is a representation of the state of CIRCUS at any given point in time. In practice, however, there are many

---

[12]These verbs are often called *control* verbs.

[13]For control verbs, it is the soft constraints of the associated semantic case frame that are actually responsible for triggering the appropriate LICK.

viable representations of that state. The approach that we have taken is to choose a representation that is natural for CIRCUS and then to rely on automated techniques to modify this representation as needed.[14] The state representation used throughout this work incorporates the following basic processing characteristics of CIRCUS:

- CIRCUS recognizes phrases as it finds them in its left-to-right traversal of a sentence.
- CIRCUS recognizes major constituents like the subject, verb, and direct object.
- CIRCUS makes no immediate decisions on structural attachment. In particular, it does not handle conjunctions or appositives, or perform prepositional phrase attachment.
- CIRCUS has access to constituent information for the current LICK only.
- CIRCUS uses one or more semantic features to describe every noun and adjective in its lexicon. For instance, in the terrorist domain, "mayor" is tagged as human, "ELN" as an organization, and the noun "murder" as an attack.
- CIRCUS treats punctuation marks as individual words.[15]
- CIRCUS has access to all of the words in an input sentence.

Consider the following sentence from the TIPSTER business joint ventures corpus:

> *Japan Airlines Co. (JAL) and Toyo Real Estate Co., a subsidiary of Sanwa Bank, bought a six million U.S. dollar luxury hotel Tuesday, Malaysia's first hotel to be wholly owned by Japanese.*

If we assume that CIRCUS has access to semantic feature information for each word and that each verb will trigger the appropriate semantic case frame, CIRCUS should have determined the following by the time it reaches the word following "bought"[16]:

Japan Airlines Co.	*S*, company name
JAL	noun phrase, company alias
Toyo Real Estate Co.	noun phrase, company name
a subsidiary	noun phrase, joint venture entity
of Sanwa Bank	*PP*, company name
bought	*V*, triggers a BOUGHT case frame
a	**current word**

[14] Two methods for automatically modifying a baseline representation are described in Chapters 6 and 8.6.

[15] This lets us handle all tokens in an input sentence using the same general processing routines.

[16] The Actor slot of the BOUGHT case frame will be filled by "Japan Airlines Co.," but not the full conjunction since no such attachments are made by CIRCUS at this time.

All of this information is part of CIRCUS's "context" at this point in the parse. Not included in the above representation is information about the positions of nearby words. As noted above, CIRCUS also knows all of the words of the current sentence. However, we arbitrarily include only a window of four words surrounding the current word as part of the context representation. As a result, the following becomes part of the current context:

,	comma, 2nd preceding word
bought	verb, preceding word
six	following word
million	2nd following word

Just after "Tuesday," the state representation should look like the following:

Japan Airlines Co.	*S*, company name
JAL	noun phrase, company alias
Toyo Real Estate Co.	noun phrase, company name
a subsidiary	noun phrase, joint venture entity
of Sanwa Bank	*PP*, company name
,	
bought	*V*, triggers a BOUGHT case frame
a six million U.S. dollar luxury hotel	*DO*, money/facility
hotel	noun, facility, 2nd preceding word
Tuesday	adverb, time, preceding word
,	**current word**
Malaysia's	following word
first	2nd following word

At the end of the sentence, however, the state representation includes only information from the embedded clause because LICKs maintain state information for one clause at a time:

Malaysia's first hotel	*S*, facility
to	
be	*AUX*
wholly	adverb
owned	*V*, triggers OWN case frame
by Japanese	*PP*, nationality
by	preposition, 2nd preceding word
Japanese	noun, nationality, preceding word
.	**period, current word**

The following sentence fragment (also from the joint ventures corpus) shows how information that is carried from one clause to another via one of the lexically-indexed control kernels can affect CIRCUS's state:

When you wanted to make a copy of something, you had to...

At "wanted," CIRCUS's state looks like the following:

when	connective, 2nd preceding word
you	*S*, human, pronoun, preceding word
wanted	**current word**
to	following word
make	2nd following word

By the word "of," however, a child LICK has been spawned and the subject of the embedded clause instantiated with the subject from the main clause:

you	*S*, human
to	
make	*V*, triggers MAKE case frame
a copy	*DO*, entity
a	determiner, entity, 2nd preceding word
copy	noun, entity, preceding word
of	**current word**
something	following word
,	2nd following word

The above examples should give a general idea of the kinds of information that will be included in the "context" portion of Kenmore's cases. In practice, however, this state information first will have to be mapped into a representation that can be used with the particular inductive learning algorithm employed. The details of how this is done will be described in later chapters (Chapter 5 for learning lexical disambiguation heuristics; Chapter 8 for learning relative pronoun disambiguation heuristics).

3.3 Knowledge Needed for Conceptual Analysis

Given the general description of CIRCUS presented above, this section simply lists the types of knowledge needed for each phase of conceptual sentence analysis within the CIRCUS system. Section 3.3.4 indicates the subset of these that we will attempt to learn within the Kenmore framework in subsequent chapters.

3.3.1 Syntactic Processing

1. For each part of speech distinguished by the system, CIRCUS requires a set syntactic predictions to drive the McEli syntactic analyzer.

2. For each word in the lexicon, CIRCUS needs to know which parts of speech are associated with the word.

3. CIRCUS also needs a set of disambiguation routines to handle part-of-speech ambiguities.

3.3.2 Predictive Semantics Module

1. CIRCUS requires a set of semantic case frame definitions to pick up or extract information from a sentence. Each case frame definition includes:
   (a) *enabling conditions* that specify the contexts in which the case frame should be activated (e.g., the PTRANS case frame in Figure 3.3 should only be activated for "brought" when the verb appears in an active construct);
   (b) a *mapping* from syntactic buffers to slots;
   (c) *hard slot constraints*;
   (d) *soft slot constraints* in the form of semantic features (e.g., the Actor of a PTRANS should be animate).
2. Case frame definitions have to be explicitly linked to the lexical items that "trigger" the case frame. (The PTRANS case frame in Figure 3.3 might be linked to "brought" and "took.")
3. Each noun and adjective in the lexicon (and optionally adverbs) has to be described in terms of one or more semantic features so that CIRCUS can test whether the word satisfies a slot's hard and soft constraints. (We assumed, for example, that "John" was tagged as *animate* in the example in Section 3.1.2.)
4. CIRCUS requires a mechanism for word sense disambiguation whenever a word has more than one semantic feature assigned to it. (For instance, the word "ball" may be tagged with both the *toy* semantic feature and the *social event* semantic feature.)

3.3.3 LICK Processing

Each LICK describes how to handle a particular class of embedded clause and requires the following pieces of information:

1. a list of the Child LICK syntactic buffers that should be instantiated with the antecedent by the Parent;
2. a specification of the syntactic construction to expect in the Child clause; and, for some LICKs,
3. a set of heuristics that can locate an antecedent (or notice that there is none) so that it can be used to fill a gap in a subsequent clause. For example, "boy" should be recognized as the antecedent of "who" in "I saw the boy who..." but there is no antecedent for "who" in "I don't know who..."

3.3.4 Knowledge Acquisition Tasks for Kenmore

The last three subsections enumerated the types of knowledge required by each component of CIRCUS for a minimal semantic analysis of a sentence. Later chapters present automated solutions to a subset of these knowledge acquisition tasks. Chapter 5 describes the learning of lexical knowledge required by CIRCUS including both syntactic lexical knowledge (Section 3.3.1 items 2 and 3 above) and lexical knowledge used by the predictive semantics module (Section 3.3.2 items 2, 3, and 4 above). In addition, that chapter shows how Kenmore can learn the enabling conditions associated with each case frame definition (Section 3.3.2, item 1a above). Chapter 8 focuses on learning one class of structural disambiguation knowledge. It describes the learning of heuristics that find relative pronoun antecedents for LICK processing.

CHAPTER 4

DOMAIN-SPECIFIC KNOWLEDGE ACQUISITION

This thesis addresses the problem of knowledge acquisition for natural language processing systems. However, we have narrowed that broad topic along two dimensions. First, the thesis concentrates on learning knowledge for conceptual sentence analysis. This was discussed in the preceding chapter. Second, the thesis concentrates on domain-specific knowledge acquisition, the topic of this chapter. The goals of the chapter are to (1) describe the task of domain-specific knowledge acquisition and distinguish it from knowledge acquisition for open-ended texts (Section 4.1), (2) discuss the constraints that domain-specific text processing imposes on the knowledge acquisition task (Section 4.2), and (3) briefly describe the two corpora that will be used throughout this work as well as the associated text processing tasks (Section 4.3).

4.1 The Task

Domain-specific knowledge acquisition for natural language systems is the process of acquiring the knowledge needed to process texts from a particular domain of interest. The requirements of domain-specific text processing are generally less stringent than open-ended text processing because the NLP system only needs to understand those portions of the text that pertain to the area of interest. Any texts or portions of text that fall outside of the domain effectively can be ignored. As a result, system designers can concentrate on encoding knowledge from a single sphere of expertise. An NLP system designed to understand only texts about bicycle repair, for example, can ignore descriptions of bicycle routes or training regimens that appear in the text and can ignore entire documents if they describe something outside the realm of bicycle repair and bicycle maintenance.

On the other hand, a domain-specific natural language system needs some method for distinguishing relevant texts or sections of text from irrelevant ones. This distinction may be less of a problem for open-ended text processing where the option to ignore text may not be available. Usually, however, there is an implicit relationship between domain-specific text processing and the level at which entire texts must be understood. Analysis of domain-specific documents promotes the use of *variable-depth text processing* — the NLP system can spend a lot of time on sections of the text that contain information relevant to the domain of interest while just skimming the rest.[1] Conversely, open-ended text understanding often implies in-depth analysis of all input documents.

Although even open-ended text processing does not make sense without the existence of global goals to drive the understanding process, domain-specific text processing systems may have to handle very detailed constraints that address the specialized information-processing needs of a particular audience. A system designed specifically to summarize texts about *recent* terrorist acts in Great Britain, for example, will need to operationalize the idea of recency and pay particular attention to the location of events. Normally, summarization of open-ended texts will not have to handle constraints of this type.

4.2 Constraints on the Knowledge Acquisition Process

Domain-specific text processing imposes the following two constraints on the knowledge acquisition process:

> **1. The acquired knowledge must be able to support variable-depth text processing.** To meet this constraint, we will use variable-depth representations of semantic knowledge.
>
> **2. The acquired knowledge must be adequate for enforcing domain-specific constraints.** In particular, any semantic feature taxonomies must be able to distinguish relevant objects from irrelevant ones.

For CIRCUS, the first constraint is handled by (1) creating semantic feature taxonomies that provide deep coverage of categories pertaining to the domain of interest and cursory coverage of objects and events unconnected to the domain, (2) defining semantic case frames that organize topics relevant to the domain and, (3) ensuring that these case frames are activated only in relevant contexts. In the business joint ventures domain, for example, we might want to recognize nouns that describe company names, governments, business persons, products, factory names, and dates, but all other nouns can be merged into a single "don't care" semantic class. In addition, we might define an INDUSTRY-PRODUCT case frame that extracts from a sentence a company name and a description of the product that the company plans to produce. The case frame would be triggered by phrases like "will produce" in a sentence like:

IBM Japan will produce 2000 laptop computers a month.

Here, the INDUSTRY-PRODUCT case frame would recognize "IBM Japan" as the company and "2000 laptop computers" as the product. However, it is important that the INDUSTRY-PRODUCT case frame not be activated in a non-business context, such as:

---

[1] Ultimately, variable-depth text processing should recognize and make use of discourse structures and cues that indicate a change of focus or a change of topic because these changes can help the system identify important passages in a text. However, *focus* is generally considered a discourse problem, that is, a problem that spans multiple sentences[Sidner, 1983, Grosz and Sidner, 1986]. Because this work concentrates on the analysis of individual sentences, issues of focus will not be explicitly addressed.

The new recipe will produce approximately five dozen cookies.

The second constraint imposed on the knowledge acquisition process states that the acquired knowledge must distinguish differences that matter for the domain. In the terrorism domain, for example, the NLP system must distinguish civilian victims from military ones. Therefore, CIRCUS's semantic feature hierarchy must make that distinction — a single "human" category would not be adequate. In the business joint ventures domain, however, the civilian vs. military distinction is unimportant and a single "human" category might be appropriate.

The use of variable-depth semantic representations that encode important domain-specific distinctions transforms CIRCUS from a sentence analyzer that is, in theory, capable of in-depth processing of all types of text, into a sentence analyzer designed expressly for domain-specific text processing. The section below describes the two corpora used in all experiments in the context of the domain-specific text-processing tasks in which they were used.

4.3 The MUC and TIPSTER Performance Evaluations

TIPSTER and MUC[2] are ARPA-sponsored performance evaluations of natural language systems. The general task in the evaluations is one of domain-specific *information extraction* [Lehnert *et al.*, 1994] — the NLP system processes a collection of texts, finds all information of interest, and produces a summary of that information in a rigid template format. In the TIPSTER and MUC performance evaluations, each participating site initially is given a corpus of texts and the corresponding "answer keys" to use for system development. The answer keys are manually encoded templates that capture all information from the corresponding source text that is relevant to the domain, as specified in a set of written guidelines.[3] After a six-to-nine month development phase, the NLP systems are evaluated by comparing the summaries each produces with the summaries generated by human experts for the same test set of previously unseen texts. The comparison is performed using an automated scoring program that rates the system according to measures of recall and precision.[4]

---

[2]MUC is an acronym for Message Understanding Conference.

[3]In spite of the guidelines, the notion of "relevance" remains somewhat vague and human encoders often disagree as to the contents of an individual answer key.

[4]Each answer key template and system output template is a series of slots and possibly empty slot fillers. *Recall* = (# correct slot-fillers in output template) / (# slot-fillers in key). *Precision* = (# correct slot-fillers in output template) / (# slot-fillers in output template). Therefore, if a system correctly filled 600 of a possible 1000 template slots correctly, its recall is 60%. If it only attempted to fill 800 of the 1000 slots, however, its precision is 75%.

In the 1991 MUC-3 performance evaluation, 15 industry and university research labs participated; 19 labs took part in the 1992 MUC-4 evaluation. Both evaluations used a corpus of texts describing terrorist activity in Latin America that was provided by the Foreign Broadcast Information Service. The 1993 MUC-5 and TIPSTER performance evaluations ran concurrently and together included 19 sites. This time, however, information was extracted for texts in two new domains — newswire accounts of business joint ventures and advances in microelectronics.

The work presented here draws from both the terrorism and joint ventures corpora. The sections below present examples of the information extraction task in each of those domains and describe each corpus in a little more detail.

4.3.1 MUC Corpus of Latin American Terrorism

The MUC-3/MUC-4 development corpus contains 1300 documents and the associated answer keys. These texts were selected as relevant to the Latin American terrorism domain from a number of on-line sources using keyword-based algorithms. Slightly over half of the texts actually describe terrorist acts given the more specific definition of terrorism provided in the MUC-3 guidelines. The remaining texts are considered irrelevant and should generate empty summary templates when presented as input to a MUC NLP system. Also associated with the MUC corpus are four sets of 100 texts and answer keys originally used for evaluation purposes only.

Documents in this corpus vary in length from a paragraph to a page or two. Average sentence length in the corpus is 27 words; the average length of each article is 12 sentences. There are newswire stories, speeches, radio and TV broadcasts, interviews, and even rebel communiques. Texts contain both well-formed and ungrammatical sentences. All texts are entirely in upper case.

The MUC-3/MUC-4 task is to extract information about terrorist incidents from each text and store that information in a template format, one template per event. Although the details of the extraction task are not important for the work described here, it is useful to look at a sample text and its associated answer key. Below is a text taken from the development corpus.

```
DEV-MUC3-0008 (NOSC)

   BOGOTA, 9 JAN 90 (EFE) -- [TEXT] RICARDO ALFONSO CASTELLAR, MAYOR
OF ACHI, IN THE NORTHERN DEPARTMENT OF BOLIVAR, WHO WAS KIDNAPPED ON 5
JANUARY, APPARENTLY BY ARMY OF NATIONAL LIBERATION (ELN) GUERRILLAS,
WAS FOUND DEAD TODAY, ACCORDING TO AUTHORITIES.

   CASTELLAR WAS KIDNAPPED ON 5 JANUARY ON THE OUTSKIRTS OF ACHI,
ABOUT 850 KM NORTH OF BOGOTA, BY A GROUP OF ARMED MEN, WHO FORCED HIM
TO ACCOMPANY THEM TO AN UNDISCLOSED LOCATION.

   POLICE SOURCES IN CARTAGENA REPORTED THAT CASTELLAR'S BODY SHOWED
SIGNS OF TORTURE AND SEVERAL BULLET WOUNDS.
```

```
   CASTELLAR WAS KIDNAPPED BY ELN GUERRILLAS WHILE HE WAS TRAVELING IN
A BOAT DOWN THE CAUCA RIVER TO THE TENCHE AREA, A REGION WITHIN HIS
JURISDICTION.

   IN CARTAGENA IT WAS REPORTED THAT CASTELLAR FACED A ''REVOLUTIONARY
TRIAL'' BY THE ELN AND THAT HE WAS FOUND GUILTY AND EXECUTED.

   CASTELLAR IS THE SECOND MAYOR THAT HAS BEEN MURDERED IN COLOMBIA IN
THE LAST 3 DAYS.

   ON 5 JANUARY, CARLOS JULIO TORRADO, MAYOR OF ABREGO IN THE
NORTHEASTERN DEPARTMENT OF SANTANDER, WAS KILLED APPARENTLY BY ANOTHER
GUERILLA COLUMN, ALSO BELONGING TO THE ELN.

   TORRADO'S SON, WILLIAM; GUSTAVO JACOME QUINTERO, THE DEPARTMENTAL
GOVERNMENT SECRETARY; AND BODYGUARD JAIRO ORTEGA, WERE ALSO KILLED.

   THE GROUP WAS TRAVELING IN A 4-WHEEL DRIVE VEHICLE BETWEEN CUCUTA
AND THE RURAL AREA KNOWN AS CAMPANARIO WHEN THEIR VEHICLE WAS BLOWN UP
BY FOUR EXPLOSIVE CHARGES THAT DETONATED ON THE HIGHWAY.
```

The text includes descriptions of two terrorist events — the kidnapping of Castellar and the bombing of Torrado's four-wheel drive vehicle. This means that there should be two output templates generated for the text. For the terrorism domain, each template includes the same 24 slots describing the date and location of the incident, the type of incident (one of 24 possible types), the perpetrators and victims, and any physical objects targetted in the attack. The answer key templates for the above story are shown in Figures 4.1 and 4.2.[5] Slots that have more than one legal filler will list each option individually, separated by a "/." The output templates generated for each story by the NLP system, however, may only have one filler per slot. The slot "matches" the corresponding slot in the answer key if its filler matches any of the allowable fillers for that slot in the answer key.

There were a number of constraints for the domain that complicated this information extraction task. For example,

1. Only terrorist acts that occurred in one of nine Latin American countries are considered relevant.
2. Only "recent" terrorist acts (i.e., those that occurred within three months of the wire date) or texts that provide new information regarding old terrorist events are considered relevant.
3. Generic descriptions of terrorist events are not relevant. The text must provide information regarding either the specific type of terrorist attack or the specific targets of the attack. For example, a newswire noting a "terrorist attack on peasants in El

[5]Some of the slots in a template are to be filled with phrases taken directly from the text (e.g., the "string fills" denoting names of victims). Others hold information converted into a canonical form (e.g., dates and locations), and the rest are "set fills" or fillers taken from a fixed set of options (e.g., effects on physical targets can be one of SOME DAMAGE, DESTROYED, NO DAMAGE).

1. message: template	1
2. incident: date	05 JAN 90
3. incident: location	COLOMBIA: BOLIVAR (DEPARTMENT): ACHI (TOWN)
4. incident: type	KIDNAPPING
5. incident: stage of execution	ACCOMPLISHED
6. incident: instrument id	*
7. incident: instrument type	*
8. perp: incident category	TERRORIST ACT
9. perp: individual id	"GUERRILLAS" / "GROUP OF ARMED MEN" / "ARMED MEN"
10. perp: organization id	"ARMY OF NATIONAL LIBERATION" / "ELN"
11. perp: organization confidence	SUSPECTED OR ACCUSED / REPORTED AS FACT: "ARMY OF NATIONAL LIBERATION" / "ELN"
12. phys tgt: id	*
13. phys tgt: type	*
14. phys tgt: number	*
15. phys tgt: foreign nation	*
16. phys tgt: effect of incident	*
17. phys tgt: total number	*
18. hum tgt: name	"RICARDO ALFONSO CASTELLAR"
19. hum tgt: description	"MAYOR OF ACHI": "RICARDO ALFONSO CASTELLAR"
20. hum tgt: type	GOVERNMENT OFFICIAL: "RICARDO ALFONSO CASTELLAR"
21. hum tgt: number	1: "RICARDO ALFONSO CASTELLAR"
22. hum tgt: foreign nation	-
23. hum tgt: effect of incident	DEATH: "RICARDO ALFONSO CASTELLAR"
24. hum tgt: total number	-

Figure 4.1: Terrorism Corpus Answer Key (Part 1).

0. message: id	DEV-MUC3-0008 (NCCOSC)
1. message: template	2
2. incident: date	05 JAN 90
3. incident: location	COLOMBIA: CUCUTA (CITY) - CAMPANARIO (RURAL AREA)
4. incident: type	BOMBING
5. incident: stage of execution	ACCOMPLISHED
6. incident: instrument id	"FOUR EXPLOSIVE CHARGES" / "EXPLOSIVE CHARGES"
7. incident: instrument type	EXPLOSIVE: "FOUR EXPLOSIVE CHARGES" / "EXPLOSIVE CHARGES"
8. perp: incident category	TERRORIST ACT
9. perp: individual id	"GUERILLA COLUMN"
10. perp: organization id	"ARMY OF NATIONAL LIBERATION" / "ELN"
11. perp: organization confidence	SUSPECTED OR ACCUSED: "ARMY OF NATIONAL LIBERATION" / "ELN"
12. phys tgt: id	"VEHICLE" / "4-WHEEL DRIVE VEHICLE"
13. phys tgt: type	TRANSPORT VEHICLE: "VEHICLE" / "4-WHEEL DRIVE VEHICLE"
14. phys tgt: number	1: "VEHICLE" / "4-WHEEL DRIVE VEHICLE"
15. phys tgt: foreign nation	-
16. phys tgt: effect of incident	DESTROYED: "VEHICLE" / "4-WHEEL DRIVE VEHICLE"
17. phys tgt: total number	-
18. hum tgt: name	"CARLOS JULIO TORRADO" "TORRADO'S SON, WILLIAM" / "WILLIAM" "GUSTAVO JACOME QUINTERO" "JAIRO ORTEGA"
19. hum tgt: description	"MAYOR OF ABREGO": "CARLOS JULIO TORRADO" "SON": "TORRADO'S SON, WILLIAM" / "WILLIAM" "DEPARTMENTAL GOVERNMENT SECRETARY": "GUSTAVO JACOME QUINTERO" "BODYGUARD": "JAIRO ORTEGA"
20. hum tgt: type	GOVERNMENT OFFICIAL: "CARLOS JULIO TORRADO" GOVERNMENT OFFICIAL: "GUSTAVO JACOME QUINTERO" CIVILIAN: "TORRADO'S SON, WILLIAM" / "WILLIAM" SECURITY GUARD: "JAIRO ORTEGA"
21. hum tgt: number	1: "CARLOS JULIO TORRADO" 1: "TORRADO'S SON, WILLIAM" / "WILLIAM" 1: "GUSTAVO JACOME QUINTERO" 1: "JAIRO ORTEGA"
22. hum tgt: foreign nation	-
23. hum tgt: effect of incident	DEATH: "JAIRO ORTEGA" DEATH: "GUSTAVO JACOME QUINTERO" DEATH: "TORRADO'S SON, WILLIAM" / "WILLIAM" DEATH: "CARLOS JULIO TORRADO"
24. hum tgt: total number	-

Figure 4.2: Terrorism Corpus Answer Key (Part 2).

Salvador" is specific enough to warrant an output template, but a "terrorist attack in El Salvador" is not.

4. Terrorist acts against military targets or military personnel are not relevant, unless a civilian target (human or otherwise) is injured or damaged as a result.

It will be important that the knowledge acquired by Kenmore aid the NLP system in meeting such constraints. The same will be true for the joint ventures domain described below.

4.3.2 TIPSTER Corpus of Business Joint Ventures

The goal of the extraction task for the MUC-5/TIPSTER Joint Ventures (JV) domain is to find information relevant to all business joint ventures mentioned in a text. A joint venture (also known as a "tie-up") is defined as "a cooperative association between two or more parties to own and/or develop a project together." Like the terrorism domain, all relevant information is to be captured in template format.

The JV development corpus contains approximately 1000 documents, mainly newswire accounts. Some documents include tables, graphs, or charts. Some texts are mixed case, but most are all upper case. Documents vary considerably in length, ranging from a single paragraph to about 18 pages. Like the terrorism corpus, this corpus also currently has four additional sets of texts used for evaluation purposes. Unlike the terrorism corpus, however, most (approximately 90%) of the texts in the JV corpus are relevant to the goals of the domain and should generate non-empty output templates.

A sample text from the development is included below:

```
<DOC>

<DOCNO>  0024 </DOCNO>

<DD>     FEBRUARY 18, 1991, MONDAY  </DD>

<SO>     Copyright (c) 1991 Jiji Press Ltd.;  </SO>

<TXT>

   DAIWA SECURITIES CO. SAID MONDAY ITS HUNGARIAN JOINT VENTURE,
DAIWA-MKB, HAS OBTAINED A SECURITIES BUSINESS LICENSE FROM AUTHORITIES
IN THAT COUNTRY.  DAIWA-MKB IS THE FIRST JAPANESE -AFFILIATED
BROKERAGE TO RECEIVE THE LICENSE FROM THE STATE SECURITIES SUPERVISION
OF HUNGARY.  THE JOINT VENTURE BETWEEN DAIWA AND MAGYAR
KUELKERESKEDELMI BANK WILL START BROKERAGE, DEALING AND UNDERWRITING
SERVICES IN EARLY MARCH, TAKING OVER THE HUNGARIAN FOREIGN TRADE
BANK'S MEMBERSHIP OF THE BUDAPEST STOCK EXCHANGE, DAIWA OFFICIALS
SAID.

</TXT>
```

Given this text, an NLP system should note that there is a single joint venture event that involves three entities. The system must also distinguish parent companies in the joint venture from child companies. In this text, Daiwa Securities Co. and Magyar Kuelkereskedelmi Bank are parent companies of the Daiwa-MKB joint venture company. The system also must determine the locations of the entities, if possible, and find any aliases used to describe them. In addition, the NLP system should determine the reason that the companies are collaborating. Figure 4.3 depicts a simplified version of the answer key associated with this text. The actual answer key is shown in Figure 4.4 — its structure is more complicated than that used in the terrorism domain because slot fillers are often references or pointers to other slots. TIPSTER systems must generate output templates in this latter format.

```
TEMPLATE-0024-1 :=
     DOC NR: 0024
     DOC DATE: 180291
     DOCUMENT SOURCE: "Jiji Press Ltd."

TIE_UP_RELATIONSHIP-0024-1 :=
     TIE-UP STATUS: EXISTING

ENTITY-1:
     NAME: DAIWA SECURITIES CO
     ALIASES: "DAIWA"
     TYPE: COMPANY
     ENTITY RELATIONSHIP: PARENT

ENTITY-2:
     NAME: MAGYAR KUELKERESKEDELMI BANK
     TYPE: COMPANY
     ENTITY RELATIONSHIP: PARENT

JOINT VENTURE CO:
     NAME: DAIWA-MKB
     LOCATION: HUNGARY (COUNTRY)
     TYPE: COMPANY
     ENTITY RELATIONSHIP: CHILD

ACTIVITY:
     INDUSTRY-TYPE: FINANCE
     PRODUCT/SERVICE:
        ("BROKERAGE, DEALING AND [UNDERWRITING SERVICES]")
```

Figure 4.3: Simplified JV Answer Key.

Finally, like the terrorism domain, vague guidelines that distinguish a relevant joint venture event from an irrelevant one complicate the extraction task:

```
{TEMPLATE-0024-1} :=
    DOC NR: 0024
    DOC DATE: 180291
    DOCUMENT SOURCE: “Jiji Press Ltd.”
    CONTENT: {TIE_UP_RELATIONSHIP-0024-1}
{TIE_UP_RELATIONSHIP-0024-1} :=
    TIE-UP STATUS: EXISTING
    ENTITY:     {ENTITY-0024-1}
                {ENTITY-0024-2}
    JOINT VENTURE CO: {ENTITY-0024-3}
    ACTIVITY: {ACTIVITY-0024-1}
{ENTITY-0024-1} :=
    NAME: DAIWA SECURITIES CO
    ALIASES: “DAIWA”
    TYPE: COMPANY
    ENTITY RELATIONSHIP: {ENTITY_RELATIONSHIP-0024-1}
{ENTITY-0024-2} :=
    NAME: MAGYAR KUELKERESKEDELMI BANK
    TYPE: COMPANY
    ENTITY RELATIONSHIP: {ENTITY_RELATIONSHIP-0024-1}
{ENTITY-0024-3} :=
    NAME: DAIWA-MKB
    LOCATION: Hungary (COUNTRY)
    TYPE: COMPANY
    ENTITY RELATIONSHIP: {ENTITY_RELATIONSHIP-0024-1}
{INDUSTRY-0024-1} :=
    INDUSTRY-TYPE: FINANCE
    PRODUCT/SERVICE:
      (62 “BROKERAGE, DEALING AND [UNDERWRITING SERVICES]”)
{ENTITY_RELATIONSHIP-0024-1} :=
    ENTITY1:    {ENTITY-0024-1}
                {ENTITY-0024-2}
    ENTITY2:    {ENTITY-0024-3}
    REL OF ENTITY2 TO ENTITY1: CHILD
    STATUS: CURRENT
{ACTIVITY-0024-1} :=
    INDUSTRY: {INDUSTRY-0024-1}
    ACTIVITY-SITE: (- {ENTITY-0024-3})
    START TIME: {TIME-0024-1}
{TIME-0024-1} :=
    DURING: EA0391
```

Figure 4.4: Actual JV Answer Key.

1. Long-term business relationships are reportable as joint ventures as long as the relationship is not purely a standard practice for the business.
2. Joint ownership of a company does not necessarily count as a joint venture. The owners must be pursuing some common activity.
3. Agreements between governments are not reportable as joint ventures unless some business activity is mentioned.

Whenever possible, the lexical and structural knowledge acquired by Kenmore for use in this domain should help the NLP system adhere to and satisfy these guidelines.

CHAPTER 5

LEARNING LEXICAL KNOWLEDGE

This chapter demonstrates Kenmore's ability to acquire lexical knowledge in a system called MayTag.[1] MayTag simultaneously learns the part of speech, semantic feature(s), and domain-specific concept(s) to be activated for all open-class words in a corpus using a case-based reasoning algorithm. It first creates a case base of context-sensitive word interpretations during a human-supervised training phase. Each interpretation encodes the context in which the word was encountered as well as the "definition" of the word as it would be specified in the lexicon of the NLP system. For CIRCUS, this "definition" consists of the word's part of speech, semantic class, and activated concepts. After training, given an unknown word and the context in which it occurs, MayTag retrieves similar past cases of interpretation from the case base to infer the word's syntactic and semantic class information in the new context. As such, MayTag is one of a number of interpretive CBR systems including HYPO [Ashley, 1990, Rissland and Ashley, 1986], GREBE [Branting and Porter, 1991, Branting, 1991], CABARET [Rissland and Skalak, 1991], and ANAPRON [Golding and Rosenbloom, 1991]. Very generally, case similarity in MayTag is assessed using a hybrid case retrieval algorithm that combines a k-nearest neighbors matching routine with a decision tree approach for finding relevant features. All retrieved cases then vote on the syntactic and semantic interpretation of the current word. The details of the case retrieval and case selection algorithms will be described in Section 5.4 as well as in the next chapter.

By encoding context as part of a word interpretation case, the meaning of a word can change dynamically in response to surrounding phrases without the need for explicit lexical disambiguation heuristics. Moreover, as an instantiation of the Kenmore framework, MayTag acquires all three classes of knowledge using the same case representation and requires relatively little training and no hand-coded knowledge acquisition heuristics.

The chapter first briefly introduces the problem of lexical acquisition (Section 5.1) and then presents the specifics of MayTag (Sections 5.2 - 5.4). We then evaluate the system in experiments that explore two of many practical applications of the technique (Sections 5.5.1 and 5.5.2). In the first application, we assume the existence of a nearly complete domain-specific dictionary and use MayTag to infer the features of occasional unknown words. In the second, more ambitious application, we assume only a small

[1]Some of the material from this chapter appeared originally in Cardie [1993a].

dictionary of function words and use MayTag to determine the definition of *all* open-class words in an input text. Results of these experiments indicate that MayTag provides a promising approach to automated dictionary construction and knowledge acquisition for sentence analysis in limited domains. Sections 5.5.3 and 5.5.4 briefly explore MayTag's ability to create static lexicons and to handle domain-independent lexical disambiguation.

5.1 The Problem

The ability or inability of a natural language processing (NLP) system to handle gaps in lexicon coverage ultimately affects the system's performance on novel texts. Suppose, for example, that a natural language system is processing a text and unexpectedly encounters an unknown word. Rather than stop and wait for a knowledge engineer to enter the missing lexical information, or skip the offending word altogether, a robust sentence analyzer should infer the necessary syntactic and semantic knowledge for the unknown word and then continue processing the text. Although the exact type and form of the knowledge required by a parser varies from system to system, knowledge-based domain-specific language processing systems typically rely on at least the following information: for each word encountered in a text, the system must (1) know which parts of speech, word senses, and concepts are plausible in the given domain, and (2) determine which part of speech, word sense, and concepts apply, given the particular context in which the word occurs.

For example, consider the following sentences within the context of the MUC domain of Latin American terrorism[2]:

1. The terrorists **killed** *General* Bustillo.
2. The *general* concern was that children might be **killed**.

It is clear that in this domain the word "general" has at least two plausible parts of speech (noun and adjective) and two plausible word senses (the military officer sense and the universal entity sense). A sentence analyzer has to know that these options exist[3] and then choose the noun/military officer form of "general" for sentence 1 and the adjective/universal entity form in sentence 2.

Thus far, we have used the term *word sense ambiguity* to refer to ambiguities with respect to the meaning of a word; however, this type of ambiguity is often more accurately described as a *semantic feature ambiguity* because many NLP systems (including CIRCUS) represent word meanings using one or more semantic features drawn from a predefined

---

[2]See Chapter 4 for a description of this domain and the associated corpus.

[3]In addition, the system should know when its existing knowledge will not handle the current usage of the word. This aspect of lexical ambiguity will be discussed in detail in Section 5.4.1.

taxonomy (e.g., *military officer* and *universal entity*).[4] Together these semantic features denote the meaning of the word in the current context.[5] Sometimes the same semantic feature will be used to describe two words with distinct meanings. For example, an NLP system may assign the *human* semantic feature to both “child” and “policeman” if a more specific distinction is not represented in the system’s semantic feature taxonomy. On the other hand, a word considered to have a single word sense may be represented by one of a number of distinct semantic features depending on the context in which it is used. In the business joint ventures domain, for example, “Japan” can refer to a *location*, a *government*, or part of a *company name* even though it has only one real word sense.

In addition to part-of-speech and semantic feature ambiguity, sentences 1 and 2 also illustrate a form of concept ambiguity with respect to the domain of terrorism. Sentence 1, for example, clearly describes an instance of the *terrorist act* concept — the word “killed” implies that a murder took place and the perpetrators of the crime were “terrorists.” This is not the case for sentence 2 — the verb “killed” appears, but no murder has occurred and there is no implication of terrorist activity. This distinction is important in the terrorism domain where the goal is to extract from texts only information concerning eight classes of terrorist events including murders, bombings, attacks, and kidnappings. All other information effectively should be ignored. To be successful in this *selective concept extraction* task [Lehnert *et al.*, 1991], a sentence analyzer not only needs access to word-concept pairings (e.g., the word “killed” is linked to the TERRORIST MURDER concept), but must also accurately distinguish legitimate concept activation contexts from bogus ones (e.g., the phrase “terrorists killed” implies that a TERRORIST MURDER occurred, but “children might be killed” probably does not).

Domain-specific concept activation is akin to the more general problem of recognizing and handling *open-textured concepts* — concepts for which necessary and sufficient conditions of membership are difficult or impossible to describe. Open-textured concepts feature prominently in tasks other than natural language processing and have been studied from a number of perspectives: philosophy (e.g., Wittgenstein [1953]), law (e.g., Hart [1961]), psychology (e.g., Rosch [1973], Rosch and Mervis [1975]), cognitive science (e.g., Lakoff [1987]), and artificial intelligence and law (e.g., Rissland [1990] and Rissland and Skalak [1991]). Although the problems of representing and recognizing open-textured concepts remain largely unsolved, a number of CBR systems have offered promising computational framework for the interpretation of open-textured concepts. In particular, mixed-paradigm reasoners like ANAPRON [Golding and Rosenbloom, 1991] and CABARET [Rissland and

---

[4]We will see shortly that this taxonomy will be a component of the concept description language made available to MayTag’s learning algorithm. As a result, the taxonomy limits the range of concepts that can be learned by the inductive learning component.

[5]Some systems, however, restrict semantic feature tagging to a subset of syntactic classes. This subset usually includes nouns and adjectives and sometimes includes adverbs. CIRCUS (Chapter 3), for example, does not assign semantic features to verbs.

Skalak, 1991] use a combination of rules and cases to handle concept ambiguity within their domains: heuristic rules are used to recognize unambiguous instances of a concept and cases are used to handle exceptions to the general rules. As will be described shortly, MayTag relies entirely on case-based reasoning for concept disambiguation — it gathers examples of instances and non-instances of the concepts during training and then draws from those examples to interpret concepts in the application phase.

The relationship between semantic features and concept activation requires some discussion. In conceptual sentence analyzers, and in most NLP systems, "concept activation" implies the triggering of a semantic case frame whose slots eventually will contain the information to be extracted from the sentence. The word "killed" in sentence 1, for example might trigger a MURDER semantic case frame that would be instantiated as shown in Figure 5.1 by the end of the sentence. Unlike semantic features, which may

MURDER
Actor: terrorists
Victim: General Bustillo

Figure 5.1: MURDER Semantic Case Frame.

be assigned to all words in a sentence to represent word meanings, only words that indicate objects or events that are important in the current domain and that organize the surrounding information should activate a domain-specific semantic case frame. Not all words assigned the *attack* semantic feature, for example, may activate the ATTACK concept. In addition, as shown in the "killed" examples above, a word that activates a concept in one context may not activate the concept in a different context.

The remainder of this chapter describes MayTag, an instantiation of the Kenmore architecture that performs lexical knowledge acquisition. MayTag begins with an empty lexicon and learns (1) to tag each word in an incoming text with the appropriate part of speech and semantic feature(s), and (2) to recognize which lexical items should activate domain-specific concepts. Thus, MayTag learns part-of-speech, semantic feature, and concept activation knowledge for all words in the corpus. No explicit system lexicon is constructed, however. Instead, the lexical knowledge associated with all words in the corpus is stored implicitly in the case base.

5.2 Instantiating the Kenmore Framework for MayTag

To learn domain-specific lexical knowledge we instantiate the three components of the Kenmore architecture as illustrated in Figure 5.2. As described in Chapter 4.3.1, the business joint ventures (JV) corpus contains over 1000 documents that describe world-wide activity in the area of joint ventures or "tie-ups" between businesses. CIRCUS [Lehnert,

Corpus:	Business Joint Ventures
Sentence Analyzer:	CIRCUS
Inductive Learning Algorithm:	CBR (k-nearest neighbor)

Figure 5.2: Instantiating Kenmore for MayTag.

1990] is a conceptual sentence analyzer that produces a semantic case frame representation of the meaning of an input sentence. Details regarding CIRCUS can be found in Chapter 3.

The heart of the learning component employed by MayTag is a nearest-neighbor case-based reasoning (CBR) algorithm. In this algorithm, the case base is effectively a set of training examples, each of which describes a single episode in lexical disambiguation. After training, each word in an incoming text is tagged using a case retrieval algorithm that compares the current context to those stored in the case base, finds the most similar cases, and then uses them to assign a syntactic and semantic interpretation to the current word. As described in the introduction, we chose a case-based reasoning algorithm as the inductive learning component because of the lack of a strong domain theory for natural language learning tasks. In addition, there is evidence from psychology that some language acquisition tasks, including the learning of word meanings, proceed through instance-based stages (e.g., Keil and Kelly [1987]). Finally, we chose the nearest-neighbor similarity metric for its simplicity: it allows us to test the basic case-based approach on lexical acquisition tasks. For reference, the components of MayTag's case-based inductive learning algorithm are summarized in Figure 5.3; the specifics of the CBR module and comparisons to related work will be described in Section 5.4.

Case Indexing:	performed automatically by decision trees
Similarity Metric:	10-nearest neighbors
Case Selection:	prefer cases that match unknown word
Solution Policy:	majority vote
Automated Improvements to Baseline Case Rep:	via decision tree approach

Figure 5.3: Components of MayTag's Case-Based Inductive Learning Algorithm.

Very generally, MayTag acquires lexical knowledge in its training phase with the help of a human supervisor and then uses the acquired knowledge to handle lexical ambiguities during the application phase without any human intervention. The next two sections and 5.4) describe MayTag's acquisition and application phases in detail.

5.3 MayTag's Training Phase

The goal of MayTag's training phase (Figure 5.4) is to create a case base of episodes in lexical ambiguity resolution. Since each case encodes the part of speech, semantic features, and activated concepts for a single word in a specific context, cases also can be seen as context-sensitive word interpretations. To create the case base, MayTag randomly selects sentences from the JV corpus and provides them as input to CIRCUS, which processes each sentence, one word at a time, and, together with a human supervisor, creates a case as each word is encountered. As described in Chapter 1 and shown in the training case of Figure 5.4, cases have two parts. The *context* portion describes the context in which the current word occurred. This is just a representation of the state of the CIRCUS parser when the current word is recognized and is supplied automatically by CIRCUS. (See Section 3.2

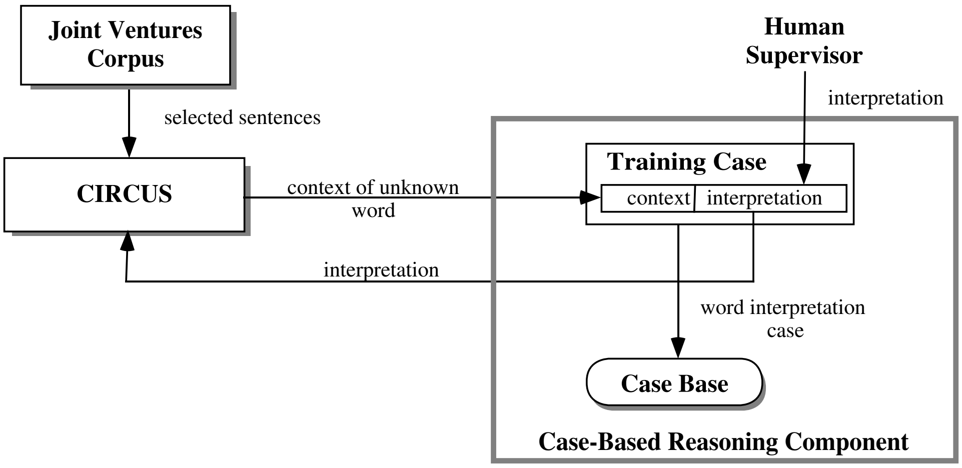

Figure 5.4: MayTag Training Phase.

for examples of the kinds of knowledge that are maintained by CIRCUS.) The second part of the case (called the *solution* in Chapter 1) encodes the syntactic and semantic *interpretation* of the current word and is supplied by a person using a menu-driven interface. The supervisor simply chooses the part of speech, semantic feature(s), and domain-specific concepts to associate with the current word from a menu of predefined options. The choices are encoded into the solution part of the case, and MayTag stores the case in the case base. In addition, the part-of-speech, semantic feature, and concept information are directed back to CIRCUS so that CIRCUS will know how to interpret the current word, can update its state, and continue with the next word in the training sentence.

At the end of training, MayTag will have created one case for each word that occurred in the training sentences. In particular, if a word appears n times in the training sentences, there will be n cases associated with it in the case base. As of yet, we have not addressed the specifics of MayTag's case representation. Because the same representation is used both in the training and application phases, we will describe the case representation in detail in the next subsections before discussing its use during the application phase (Section 5.4).

5.3.1 The Taxonomies

Cases in MayTag are represented as a set of attribute-value pairs. Although some case-based reasoning systems use a richer case representation (e.g., CHEF [Hammond, 1989], GREBE [Branting and Porter, 1991, Branting, 1991]), the simple attribute-value pair knowledge representation is a very general representation scheme that can be used in conjunction with case-based algorithms (see HYPO [Ashley, 1990]) as well as the vast majority of symbolic inductive machine learning algorithms.[6] More specifically, MayTag cases consist of 37 attribute-value pairs: 33 features represent the *context* in which the current word was encountered and 4 represent the syntactic and semantic *interpretation* of the current word in this context. The case representation relies on three predefined taxonomies, one for each class of knowledge that we are trying to learn. These will be described briefly before presenting the solution and context parts of the case.

MayTag first requires a taxonomy of semantic features. This taxonomy was designed for use with the business joint ventures corpus and is shown in Tables 5.1 and 5.2. It is a two-level taxonomy that includes 14 general semantic features and 42 more specific semantic features. As discussed in Section 4.2, this taxonomy should reflect any distinctions in word meanings that are important for processing texts from the joint ventures domain. For example, in this domain it is important to know when a word is describing a party involved in a joint venture. Any word used in this fashion should be tagged with the *jv-entity* general semantic feature. In addition, it is also important to note whether that word describes a more specific joint venture entity like a *company name*, a *government*, or a *person*.

Next, MayTag requires a set of domain-specific concepts for use with texts from the JV corpus. MayTag's current taxonomy contains 11 domain-specific concept types (Table 5.3). These represent a subset of the concepts that are important in the joint ventures domain and act as semantic case frame types for CIRCUS. (See Chapter 3 for more information about the use of semantic case frames in the CIRCUS parser.)

Finally, MayTag uses a taxonomy of 18 parts of speech (Table 5.4). The taxonomy specifies 7 parts of speech for *open-class words* and reserves the remaining 11 parts of speech for *closed-class words*. Closed-class words are function words like prepositions, auxiliaries, and connectives, whose meanings vary little from one domain to another. All other words (e.g., nouns, verbs, adjectives) are open-class words. Open-class words are generally considered content words. Although the semantic feature and concept taxonomies are clearly domain-specific, the part-of-speech taxonomy is parser-dependent rather than domain-dependent.

---

[6] Although we concentrate on case-based classification methods, we will also investigate the use of decision trees as MayTag's inductive learning component in Chapter 6.

Table 5.1: Semantic Feature Taxonomy (Part 1).

General Semantic Features **Specific Semantic Features**	**Description**
jv-entity	party involved in a tie-up
company-name	name of company
company-alias	alias for company
generic-company	e.g., "Co." in "Plastics Co."
government	government-affiliated entity
person	an individual
industry	type of business or industry
research	research and development
production	manufacturing, production
sales	sales, marketing, trade
service	consumer services, service industry
finance	e.g., banking, finance, real estate
capitalization	total cash capitalization
person-position	position of a person within company
ceo	chief executive officer
cob	chairman of the board
pres	head of company
offic	any officer of company
srexec	senior executive
exec	other management
owner	owner, partner
gov	government official
spoke	spokesperson
prof	titled professional
emp	employee
other-position	any position not included above

Table 5.2: Semantic Feature Taxonomy (Part 2).

General Semantic Features **Specific Semantic Features**	**Description**
facility	building or site
communications	e.g., radio, television, satellite
site	e.g., development sites, industrial buildings
factory	e.g., manufacturing building, computing facilities
farm	e.g., agricultural and forestry facilities, orchards
office	e.g., auxiliary offices, departments, sales offices
mine	e.g., coal or mineral mines, quarries, gas wells
store	e.g., stores, shops, dealers, amusement parks
transportation	land, air, water transportation, pipelines
utilities	e.g., electric power, heat, lighting
warehouse	general and specialized storage
facility-name	name of facility
other-facility	any facility not included above
ownership-percentage	share in company
entity	generic entity/thing, irrelevant object w.r.t. JV domain
money	any monetary reference
nationality	any nationality
human	person not a jv-entity or person-position
human-name	person name
human-title	person title
time	any reference to time or date
location	reference to a specific place
country	country name
city	city name
province	province or state
continent	name of continent
other-loc	any other specific place name
generic-location	general spatial area, e.g., north, Midwest
nil	no general semantic feature applies
nil	no specific semantic feature applies; this feature can be used in conjunction with ANY general semantic feature

Table 5.3: Concept Type Taxonomy.

Concept Types	**Description**
tie-up-relationship (tie-up)	indicates a tie-up activity
tie-up-relationship-secondary (tie-up-secondary)	weak indicator of a tie-up
tie-up-relationship-jv-entity-only (tie-up-jv)	indicates a tie-up activity but only mentions the joint venture company
total-capitalization	total cash capitalization
ownership-%	percentage share in the tie-up
industry	type of industry performed within the scope of the tie-up
industry-research	research and development
industry-production	manufacturing, production
industry-sales	sales, marketing, trade
industry-service	consumer services, service industry
industry-finance	e.g., banking, finance, real estate
nil	no concept should be triggered

Table 5.4: Part-of-Speech Taxonomy.

Open-Class	**Closed-Class**
noun	preposition (prep)
noun modifier (nm)	auxiliary (aux)
adverb (adv)	copular (cop)
verb	article (art)
past participle (pasp)	relative pronoun (rel)
present participle (pres)	modal
gerund (ger)	infinitive (inf)
	conjunction (conj)
	negation (neg)
	connective (conn)
	particle (ptcl)

5.3.2 Case Representation

As mentioned above, each case in MayTag represents the syntactic and semantic interpretation of a single word as well as the context in which it occurs in the corpus. It is a list of 37 attribute-value pairs in which 4 features encode the solution part of the case and the remaining 33 features encode the context. The best way to describe MayTag's case representation is to look at an example. Throughout this section, we will examine the case that would be generated for the word "venture" in the following sentence taken directly from the JV corpus:

> Toyota Motor Corp. has set up a joint **venture** firm with Yokogawa Electric Corp.

Figure 5.5 shows the training case for "venture" as it is used in the sentence above. The 4 features that encode the syntactic and semantic interpretation of "venture" are depicted at the top of the figure and are labeled "Word definition features." These are the "solution" features of the case and are supplied by the human supervisor. In the current context, the supervisor has specified that (1) the part of speech (**p-o-s**) for "venture" is a *noun modifier (nm)*,[7], (2) its general semantic feature (**gen-sem**) is *entity*, (3) no specific semantic feature (**spec-sem**) applies, and (4) "venture" activates the domain-specific *tie-up* concept.

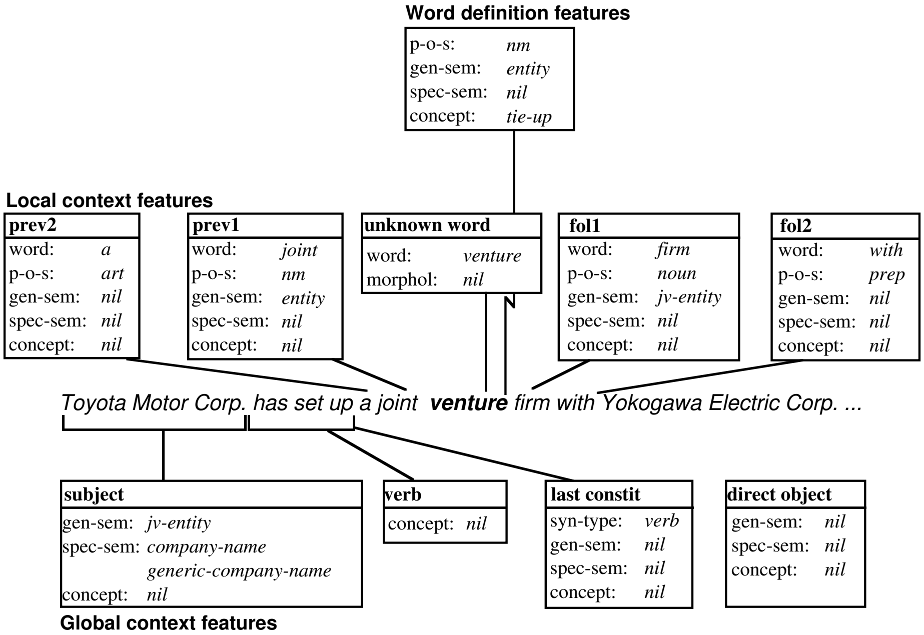

Figure 5.5: Case for "venture."

The case's 33 "context" features are divided conceptually into two sets:

---

[7]The *noun modifier* category covers both adjectives and nouns that act as modifiers. We reserve the *noun* category for head nouns only.

- **local context features** (22) that represent semantic and syntactic knowledge for the words within a five-word window centered on the unknown word; and
- **global context features** (11) that encode information for any major syntactic constituents that have been recognized in the current clause at the point of the unknown word.

The general idea behind the representation of context is to include any information that the parser has that might be useful for inferring the definition of the current word. This representation of context is necessarily dependent on the kinds of information available to and maintained by the sentence analyzer employed by MayTag. Therefore, the context features described below are consistent with the CIRCUS parser. If MayTag uses a different parser, the representation of context will change.

The 22 local context features include two attribute-value pairs that describe surface features for the **unknown word**. First, CIRCUS knows that the unknown word is "venture." Second, the **morphol** feature indicates morphological information for the word based on its suffix. The *nil* value used here means that no such information was available for "venture." The remaining 20 local context features describe information that CIRCUS maintains for the two words that precede and the two words that follow the unknown word (labeled **prev1**, **prev2**, **fol1**, and **fol2** in Figure 5.5). For each of these positions, CIRCUS should know (1) the word in that position, (2) its part of speech, (3) semantic features, and (4) activated concepts. The word immediately preceding "venture," for example, is the *noun modifier* "joint." It has been recognized as an *entity* and activates no domain-specific concept in this context.

Finally, the 13 global context features encode information about any major syntactic constituents that CIRCUS has recognized at the point of the unknown word. When the parser reaches "venture," for example, it has recognized two major constituents — the subject and verb phrase. Neither activates any domain-specific concepts, but the subject does have general and specific semantic features. These are acquired by taking the union of the semantic features of each word in the noun phrase. As described above (Section 5.3.1), verbs in CIRCUS are assigned no general or specific semantic features, but can activate domain-specific concepts although none was activated in the example sentence. Because CIRCUS has not yet recognized the direct object (this will not happen until after the word "firm"), all of the direct object features in the case are empty. In addition to specifying information about each of the main constituents, the global context features also include syntactic and semantic knowledge for the most recent low-level constituent (**last constit**). A low-level constituent can be either a noun phrase, verb, or prepositional phrase, and sometimes coincides with one of the major constituents — the subject, verb phrase, or direct object. This is the case in Figure 5.5 where the low-level constituent preceding "venture" is the verb.

Because CIRCUS generally maintains syntactic information for at most one clause at a time, the global context features are limited to constituents recognized in the current clause. An exception to this is when the LICK mechanism instantiates constituent buffers in the current clause with phrases from the preceding clause (see Section 3.1.3). When this happens, the global context features may refer to portions of the preceding clause. In addition, the **last constit** feature takes on the value *clause* at the beginning of a new clause, indicating that the last item recognized by CIRCUS was a clause. Clause boundaries are only implicitly noted in the local context features, via the presence of punctuation marks or connectives, for example.

5.3.3 Case Base Construction

Using the case representation described in the last section, MayTag creates a case base of context-dependent word interpretations from a small, randomly selected subset of the sentences in the JV corpus. As each word in the training sentences is encountered, MayTag creates a case by alternately consulting a human supervisor and the CIRCUS parser. The human supervisor supplies the current word's part-of-speech, semantic features, and concept activation information. These values are stored in the **p-o-s**, **gen-sem**, **spec-sem**, and **concept** word definition features and are used by the parser to process the current word. CIRCUS automatically supplies the global and local context features of the case after examining its state. The **prev1** and **prev2** features are immediately available because a person has already provided CIRCUS with the necessary syntactic and semantic information for those words. The local context features associated with the words in the **fol1** and **fol2** positions, however, are added to the current case after those words have been tagged by the supervisor in CIRCUS's left-to-right traversal of the training sentence. After training, MayTag will have produced one case for every word in the training sentences. In particular, if "venture" appears n times during training, there will be n cases associated with it in the case base.

5.4 MayTag's Application Phase

Once the case base has been constructed, MayTag can use it to determine the interpretation of new words in the corpus (Figure 5.6). Assume, for example, that CIRCUS needs to know the part of speech, semantic features, and activated concepts for "Toyo's" in the sentence:

Yasui said this is **Toyo's** and JAL's third hotel joint venture.

Based on the state of the CIRCUS parser at the word "Toyo's," MayTag creates a problem case for "Toyo's" filling in its global and local context features just as it would during training.[8] The only difference between a problem case and a training case is the word definition features — the **gen-sem**, **spec-sem**, **p-o-s**, and **concept** features for the unknown word. During training, the human supervisor specifies values for these features. During the application phase, it is the job of the case retrieval algorithm to find the training cases

[8]There is a bootstrapping problem in that the **fol1** and **fol2** features are needed to specify the problem case for "Toyo's." This problem will be addressed in the second experiment. For now, assume that the parser has access to all **fol1** and **fol2** features at the position of the unknown word.

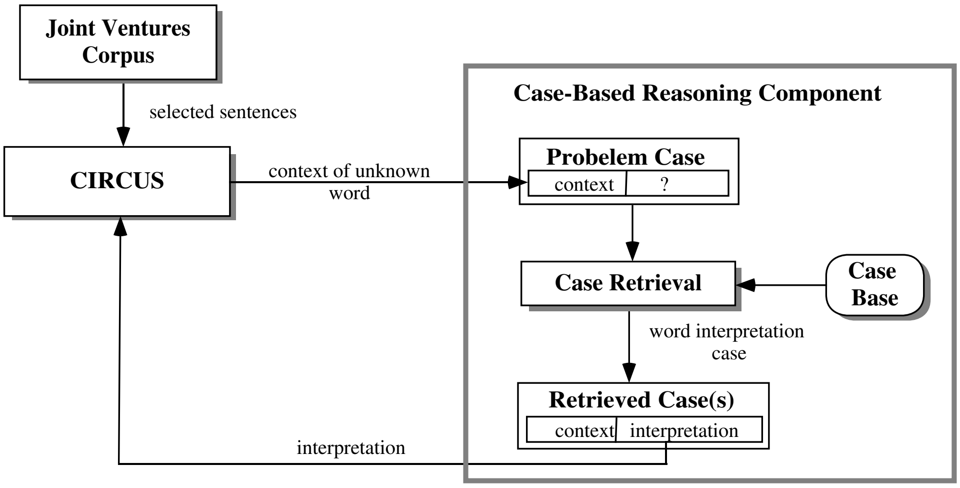

Figure 5.6: MayTag Application Phase.

that are most similar to the problem case and then use them to determine values for the missing word definition features of the unknown word. We use the following algorithm for this task:

1. Compare the problem case to each case in the case base, counting the number of context features that match (i.e., match = 1, mismatch = 0). Only give partial credit (.5) for matches on *nil*'s and for partial matches.[9]
2. Keep the ten highest-scoring cases.
3. Of these, return those case(s) whose **word** feature matches the unknown word, if any exist. Otherwise, return all ten cases.[10]
4. Let the retrieved cases vote on the values for the four word definition features of the problem case. Ties are broken randomly.

The case retrieval algorithm is essentially a k-nearest neighbors algorithm (k = 10) with a bias toward cases whose **word** matches the unknown word.[11] Because MayTag uses the retrieved cases to derive an appropriate syntactic and semantic interpretation for the current word, it is considered an interpretive CBR system. (For examples of other interpretive CBR systmes, see HYPO [Ashley and Rissland, 1988, Ashley, 1990], GREBE [Branting and Porter, 1991, Branting, 1991], CABARET [Rissland and Skalak, 1991],

[9]Partial matches occur when two features have overlapping, but not identical values. A partial match occurs, for example, when the subject of the problem case is *(company-name product)* and the subject of the training case is *(company-name)*.

[10]More than ten cases will be returned if there are ties.

[11]Other values of k were tested, but setting k to ten produced the best results overall given the corpus, taxonomies, and training sets used in the experiments.

ANAPRON [Golding and Rosenbloom, 1991].[12]) MayTag, however, differs from many traditional case-based problem solvers in that it employs no case adaptation phase — the retrieved solution is used directly rather than being modified to fit the new situation. In addition, MayTag's case representation contains none of the deep or derived features that typically comprise a case in a CBR system.[13] MayTag's emphasis on case-based classification using simple feature vectors and without case adaptation makes it more similar to *instance-based* machine learning algorithms [Aha *et al.*, 1991].

5.4.1 Recognizing Incomplete Word Definitions

As a result of its case-based approach to lexical disambiguation, MayTag allows a word to take on an interpretation different from any it received during the training phase. This will happen if the word appears in a context that is different from any in which it occurred during training. The case retrieval algorithm delivers this behavior by first looking for cases that are the best overall matches for the problem case (step 2) and then searching among these for cases generated during training in response to the current unknown word (step 3). In a preliminary training phase, for example, the word "lettuce" appeared once across all of the training sentences where it was tagged as a *noun/industry/product*:

> Guam United Agricultural Management...has developed a small computerized water culture system which can grow vegetables like tomatoes, **lettuce**, spring onions, melons...

When running MayTag in test/application mode, however, "lettuce" was recognized as a *nm/jv-entity/company-name*. Although it initially seemed that this definition must be incorrect, the opposite was actually the case. "Lettuce" had been seen in the following context:

> Ogden Allied Services Corp. had formed a joint venture with Richard Melman's **Lettuce** Entertain You Enterprises.

This type of adaptive behavior is generally very difficult to achieve in natural language processing systems. It is much easier for an NLP system to assume that it knows either all or none of the definitions of a word rather than deal with the complications that accompany partial definitions. The ability to recognize novel uses of words is a natural side effect of MayTag's case-based approach to lexical acquisition.

5.4.2 Modified Case Retrieval Algorithm

One problem with the case retrieval algorithm presented above is that it assumes that all context features are equally important for learning part-of-speech, semantic feature,

[12]See Section 2.7 for a brief description of the interpretive tasks performed by these systems.

[13]It could be argued, however, that MayTag's global context features are actually derived features since they represent higher level knowledge obtained by the sentence analyzer from one or more words.

and concept activation knowledge. Intuitively, it seems that accurate prediction of each part of the definition may rely on very different subsets of the features. Unfortunately, it is difficult to know which combinations of features will best predict each class of knowledge without trying many (or all) of them. To avoid this additional knowledge engineering bottleneck, we developed an automated approach for locating the relevant features in a baseline instance representation and have incorporated the approach into the original case retrieval algorithm.[14] The modified algorithm uses the C4.5 decision tree system [Quinlan, 1992] and will be described in detail in Chapter 6. (Related work will also be discussed in that chapter.) For the remainder of this chapter, however, it will suffice simply to think of this feature selection algorithm as a black box (Figure 5.7) that takes as input a set of training cases for a particular classification task. Each training case is described in terms of n features, f_1 through f_n, and the class value c to be associated with the case. For MayTag, f_1 through f_n are the local and global context features and c is one of the **p-o-s**, **gen-sem**, **spec-sem**, or **concept** word definition features. As output, the algorithm produces a list, rel_c, of the context features that are useful or relevant for prediction of c.[15]

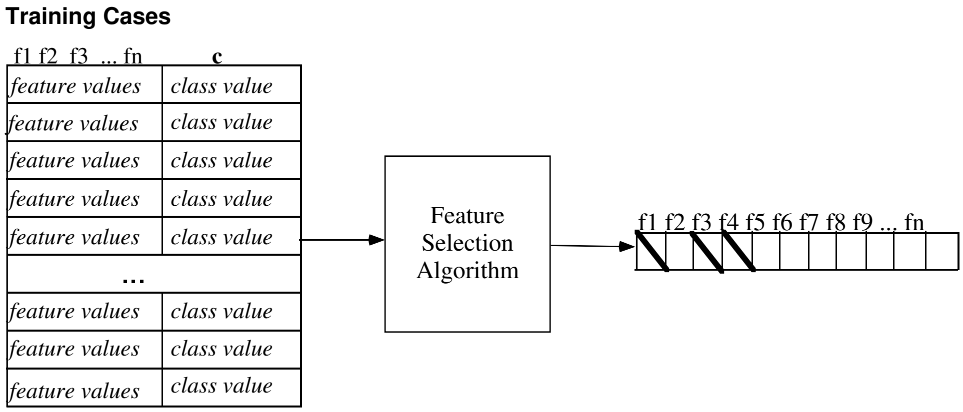

Figure 5.7: Black Box View of Feature Selection Algorithm.

All experiments described in this chapter rely on a case retrieval algorithm that uses this automated feature selection algorithm to select the subset of context features that should take part in the nearest-neighbor calculations. To use the feature selection algorithm in this manner, the original case retrieval algorithm must be altered in two ways. First, the training phase must include an additional offline step:

[14]It should be noted that the problem of automated feature selection is not new: it was first addressed by Samuel [1963,1967] in the context of evaluation function learning for game tree search. MayTag, however, encounters the problem in the context of learning an appropriate similarity function.

[15]One can also view the feature selection algorithm as a feature weighting algorithm in which relevant features are assigned a weight of one and irrelevant features a weight of zero.

- Given MayTag's case base of training cases as input, use the feature selection algorithm to create a list of relevant features, rel, for each word definition feature to be predicted.

This produces four feature lists: rel_{p-o-s}, $rel_{gen-sem}$, $rel_{spec-sem}$, and $rel_{concept}$. Second, we alter the original case retrieval algorithm to use these lists as follows:

- Instead of invoking the case retrieval algorithm once for each problem case, run it four times, once for each class value to be predicted.[16] In the retrieval for any particular class attribute c, however, include only the features in rel_c in the nearest-neighbors calculations.

By using the feature selection algorithm to discard irrelevant features, we automatically tune the case retrieval algorithm for independent prediction of part of speech, semantic features, and domain-specific concepts. Chapter 6 describes the series of experiments that compare this modified algorithm for feature specification to the original case retrieval algorithm and to knowledge-based methods for feature specification. It also discusses related work in the area of automated feature set specification. The next section describes experiments that evaluate the performance of MayTag on a number of lexical acquisition tasks. As stated above, all experiments use the modified case retrieval algorithm.

5.5 Evaluation of MayTag

We ran a number of experiments to test MayTag's ability to acquire lexical knowledge from a corpus using the modified case retrieval algorithm. Each experiment drew its training and test cases from the same set of 120 randomly selected sentences from relevant texts in the JV corpus. For each experiment, we supplied CIRCUS with a small lexicon of function words (e.g., the, a, an, is, are, was, and, or) and used MayTag to supply syntactic and semantic information all remaining (open-class) words in the test sentences. The function word lexicon maintains the part of speech and semantic features (if any apply) for 129 words. None of the function words has any associated domain-specific concepts.

All experiments report average results using 10-fold cross validation. We first randomly partition the 120 sentences into ten segments. Altogether, the 120 sentences generate 2056 cases, one context-sensitive word interpretation case for each occurrence of an open-class word. In each of ten experiments, we use the sentences in nine of the segments to generate the case base (approximately 2050 cases) while reserving the sentences in the remaining segment for testing (approximately 205 cases). Results are averaged across the ten runs, which progress through all training/testing combinations. The same training/test set combinations are used for each experiment.

We evaluate MayTag with respect to two main tasks. In the first set of experiments, we demonstrate MayTag's ability to provide the part of speech, semantic features, and

---

[16] There is a single case base, but each case has four class labels associated with it, values for the four solution features.

domain-specific concept for occasional unknown words in an input text (Section 5.5.1). Next, we demonstrate MayTag's ability to tag all open-class words in an input text from scratch (Section 5.5.2). Finally, we explore the possibility of using MayTag to generate explicit domain-specific lexicons for a corpus (Section 5.5.3). Chapter 7 provides a more qualitative analysis of MayTag's ability to handle each type of lexical ambiguity.

5.5.1 *Handling Occasional Unknown Words*

In this section we describe experiments that use MayTag to determine the part of speech, semantic features, and domain-specific concepts for occasional unknown words in an input text. This task is important when an NLP system is using a nearly complete domain-specific dictionary, but still occasionally encounters words for which the lexicon has no entry. We simulate this situation by assuming that MayTag has perfect knowledge of all words in a test sentence except the word that it currently is trying to tag. This means that the representation of context that becomes part of the problem case is relatively accurate. Table 5.5 shows MayTag's average percentage correct for prediction of each of the four word definition features and compares them to two baselines.[17] The first

Table 5.5: Handling Occasional Unknown Words (% correct for prediction of word definition features).

Word Definition Feature	**Random Selection**	**Default**	**MayTag (open-class words)**	**MayTag (all words)**
p-o-s	34.3	81.5	93.0	96.1
gen-sem	17.0	25.6	78.0	87.7
spec-sem	37.3	58.1	80.4	89.0
concept	84.2	91.7	95.1	97.2

baseline indicates the expected accuracy of a system that randomly guesses a legal value for each missing feature based on the distribution of values across the training set. The second baseline shows the performance of a system that always chooses the most frequent value across the training set as a default. The third column of results indicates MayTag's performance on the open-class words in the test sentences and the final column indicates MayTag's performance when function words are included. Chi-square significance tests on the associated frequencies show that MayTag performs significantly better than both baselines ($p = .01$).

[17]Note that we allow mismatches between the noun and noun modifier part-of-speech categories for all experiments and baselines because the parser can accurately fix these errors.

5.5.2 *Tagging All Open-Class Words From Scratch — Acquiring a Domain-Specific Lexicon*

In the second, more ambitious, task, we use MayTag to acquire definitions of *all* open-class words in the test sentences from scratch. Unlike the experiments above, we make no assumptions about the availability of definitions for words surrounding the current unknown word. Instead, MayTag has to rely on its own predictions for preceding words even though some of those may have been incorrect. As a result, the representation of context that comprises each problem case may contain many errors. More specifically, MayTag calls on CIRCUS to parse each test sentence and to create a problem case each time an open-class word is encountered, filling in its global context features and the local context features for the preceding two words. Determining values for the **fol1** and **fol2** features is more difficult because CIRCUS has not processed those words yet in its left-to-right traversal of the input sentence. If the following two words are both function words, then the **fol1** and **fol2** features can also easily be specified by looking up the words in the function word lexicon. Most of the time, however, one or both of **fol1** and **fol2** are open-class words for which the system has no definition. In these cases, the MayTag makes an educated guess based on the training cases:

1. If the word appeared during training, let each **fol1** and **fol2** feature be the union of the values that occurred in the word's training phase definitions.
2. If the word did not appear during training, fill in the **word** features, but use *nil* as the value for the remaining **fol1** and **fol2** attributes.

We also relax the case retrieval matching algorithm and allow a non-empty intersection on any **fol1** or **fol2** feature to count as a full match during the nearest-neighbor matching.[18] Results for this tagging-from-scratch task are shown in Table 5.6 along with the same baseline comparisons from the first experiment. Not surprisingly, all of the results have dropped somewhat; however, chi-square analysis still shows that MayTag's performance is significantly better than the baselines ($p = .01$).

5.5.2.1 *MayTag in the MUC-5/TIPSTER Evaluation*

MayTag handled the semantic feature tagging task for the UMass/Hughes CIRCUS system that participated in the MUC-5 and TIPSTER evaluations [Lehnert *et al.*, 1993a, Lehnert *et al.*, 1993b]. Very generally, the goal for each system in this evaluation was to read a set of previously unseen texts from the business joint ventures domain and to produce a template summary for any joint venture event described in the texts. Chapter 4 provides a more detailed description of this information extraction task. In the UMass/Hughes

[18] Matches on *nil* continue to receive only half credit. In the original case retrieval algorithm, both partial matches (i.e., matches where two cases have overlapping, but not identical values) and matches on *nil* received half credit.

Table 5.6: Tagging from Scratch (% correct for prediction of word definition features).

Word Definition Feature	**Random Selection**	**Default**	**MayTag** (open-class words)	**MayTag** (all words)
p-o-s	34.3	81.5	91.0	95.0
gen-sem	17.0	25.6	65.3	80.6
spec-sem	37.3	58.1	74.0	85.5
concept	84.2	91.7	94.3	96.8

system, the OTB tagger [Lehnert *et al.*, 1993a] supplied parts of speech for each word in the texts and domain-specific concept triggering information was generated by the AutoSlog system [Riloff, 1993], which was described briefly in Section 2.6. In addition, the CIRCUS system included three "specialists" that recognized date expressions, money expressions, and some location expressions, and converted them into canonical forms. Instead of the case base used in the above experiments (2056 cases based on 120 sentences), MayTag relied on a new, larger case base for this task. The new case base contained 3060 cases and was created from 174 training sentences from the JV corpus.

Table 5.7 shows the performance of this version of MayTag on the base set of 174 sentences from the JV corpus using 10-fold cross validation. Experiments tested the system's ability to perform semantic feature tagging from scratch. Results from the original semantic feature tagging experiments are included for comparison.

Table 5.7: Using MayTag with Larger Case Base for the Tagging-From-Scratch Task (% correct).

Word Definition Feature	**Original Results** open-class words (2056 cases)	**Larger Case Base** open-class words (3060 cases)	**Larger Case Base** incl. function words (3060 cases)
gen-sem	65.3	74.0	85.7
spec-sem	74.0	75.0	86.3

In an attempt to determine the impact of MayTag on the overall performance of the UMass/Hughes MUC-5/TIPSTER system, we ran two variations of the system on the three test sets from the final TIPSTER evaluation.[19] In the first run, the system used MayTag (and the usual lexicon of 129 function words) to assign semantic features to all open-class words in the test texts. In the second run, the system had access to the function word lexicon, but MayTag was turned off and all open-class words were assigned the *nil* general

[19]Thanks to Jonathan Peterson for running these experiments.

and specific semantic features. In this run, the system was forced to rely on its default heuristics for determining object types rather than relying on MayTag for that information.

Without MayTag, precision on the test sets increases by 1%, but recall falls by 41%, resulting in a 35% drop in F-measure.[20] These results can be explained by the fact that the default heuristics handle only clearcut cases (e.g., "Co." indicates the name of a company), but were not designed to handle novel or ambiguous situations. In addition, the module of the system that is responsible for generating output templates uses decision trees that rely on semantic feature information. It should be emphasized that this experiment was meant only to give an indication of the extent to which MayTag contributed to the UMass/Hughes MUC-5/TIPSTER system. More rigorous testing would be required to determine the exact role played by MayTag.

Training MayTag for use with the UMass/Hughes MUC-5/TIPSTER system took approximately 14 hours. The case base acquired during training acts both as the lexicon for all open-class words and as the set of accompanying disambiguation heuristics that would be required. The 14-hour training time is minimal in comparison to the task of building the lexicon and disambiguation heuristics for the approximately 32,000 distinct words in the joint ventures corpus by hand. Specifying semantic features for only the 5,000 most frequently occurring words would take 1,250 hours at 15 seconds per word. Building the disambiguation heuristics would take an additional day or two at the least and the resulting system would require additional routines to handle words that were missing from the lexicon. It is doubtful that a lexicon created by hand would produce significant improvements over MayTag's automated approach for a number of reasons: (1) people can introduce unnecessary ambiguity in lexical entries by listing semantic features that are seldom (or never) used, (2) conversely, people can omit important semantic features for a word, (3) novel uses of words will be impossible for the resulting system to recognize without the addition of special routines, and (4) the disambiguation heuristics and heuristics for handling unknown words are difficult for people to specify accurately.

5.5.3 Creating Explicit System Lexicons

MayTag's approach to lexical acquisition and lexical ambiguity does not create any explicit lexicons. The cases and the case retrieval algorithm together represent an implicit domain-specific lexicon tuned expressly for the JV corpus. However, MayTag can also be used to construct **explicit** domain-specific lexicons. To do this, one first creates the case base by training MayTag on a subset of sentences from the selected corpus. Then MayTag is run on the entire corpus in application mode, keeping track of the interpretations assigned to each lexical item. By examining the frequency with which a word was assigned various "definitions" across the corpus and by setting thresholds appropriately, a static dictionary

---

[20] The F-measure combines recall (R) and precision (P) scores: $(P * R)/(P + R)$. This version of the F-measure weights recall and precision equally and gives a higher score to systems with a high recall and precision sum *and* with relatively close recall and precision values.

that lists the allowable parts of speech, semantic features, and domain-specific concepts for all words in the corpus can be generated automatically. At the very least, this method can be used to create a static dictionary of words that are unambiguous with respect to the current domain. MayTag might then rely on the static lexicon entry whenever one of these unambiguous words is encountered rather than perform the relatively expensive case retrieval operation to infer its definition.

As an exploratory investigation, we trained MayTag on 174 sentences from the JV corpus (as in Section 5.5.2.1) and then ran MayTag in application mode on 100 novel texts from the corpus.[21] As in the above experiments, the OTB tagger supplied parts of speech for each word in the texts and domain-specific concept triggering information was generated by the AutoSlog system. Whenever a word in the sample was assigned a *noun*, *nm*, *gerund*, or *adverb* part-of-speech tag by the OTB tagger, we gathered information regarding the semantic features assigned to the word by MayTag. Table 5.8 lists the semantic features assigned to a handful of the 5546 unique words appearing in that sample along with their associated frequencies. If this pattern of usage continued as more texts in the corpus were tagged, one might want to conclude that: "also" should always be assigned the *nil/nil* tags; "largest" and "proposed" should always be assigned the *entity/nil* tags; "Subaru," "Sugetsu," and "Sumitomo" should always be considered *jv-entity/company-names*; "years" is a *time/nil*; and "still" and "technology" are too ambiguous to assign any default semantic feature tags.

5.5.4 Extending MayTag for Domain-Independent Lexical Disambiguation

Kenmore has been presented as a framework for domain-specific knowledge acquisition for conceptual sentence analysis. Hence, MayTag has been presented as a system that can acquire domain-specific lexical knowledge. An important question, however, is how this approach can be extended to handle lexical ambiguities in unrestricted domains. One solution would be to build up a collection of case bases, each one of which was designed to cover texts in a single specialized domain. A more feasible solution might be to start with a general, on-line dictionary and use the case-based approach to decide which of the definitions listed for a word is appropriate in a given context. Each case in the case base includes the usual context and solution portions, but the context part could be expanded to include a description of the options available for the current word in the on-line dictionary. The solution part of the case specifies the definition that is correct in the current context or, if none were correct, describes the novel definition. This approach allows one to start with a general lexicon (rather than creating one from scratch) and make implicit domain-specific or parser-specific additions to it via the case base. Unfortunately, broadening the domain of the texts without limiting the scope of the natural language task may make MayTag's training phase prohibitively long — both proposed solutions still

[21] This included every other text of the 200 TIPS2 evaluation texts.

Table 5.8: Creating an Explicit Lexicon.

Word	**Semantic Feature Combination**	**Frequency**
ALSO	nil/nil	86
LARGEST	entity/nil	23
	jv-entity/company-name	1
PROPOSED	entity/nil	10
STILL	nil/nil	11
	time/nil	9
SUBARU	jv-entity/company-name	8
	jv-entity/government	2
SUGETSU	jv-entity/company-name	5
SUMITOMO	jv-entity/company-name	8
TECHNOLOGY	entity/nil	10
	industry-product/nil	5
	industry-product/research	4
	industry-product/product	3
	jv-entity/company-name	1
YEARS	time/nil	28
	entity/nil	5

require supervised training. We will return to these issues again in Chapter 10 following a more detailed analysis of the results presented in this chapter.

5.6 Summary

MayTag illustrates Kenmore's ability to acquire lexical knowledge. It simultaneously learns three classes of knowledge using the same case representation, and requires no hand-coded acquisition heuristics and relatively little training. By encoding context as part of a word's representation, MayTag also allows the syntactic and semantic interpretation of a word to change dynamically in response to new contexts *without* the use of lexical disambiguation heuristics. In addition, MayTag's context-sensitive interpretations provide a simple mechanism for detecting when an existing "definition" for a word is inappropriate or incomplete. MayTag has been tested in two practical applications — (1) assigning interpretations for occasional unknown words and (2) tagging all words in an incoming text from scratch. In these tasks, MayTag performs statistically significantly better than baselines that randomly guess or choose default values for the features of the unknown word. In addition, the system was used to provide semantic feature tagging for the UMass/Hughes CIRCUS system, which competed in the MUC-5 and TIPSTER national performance evaluations of text processing systems. Finally, we have demonstrated that MayTag provides a uniform computational mechanism for handling ambiguous words and unambiguous words, as well as completely unknown words.

CHAPTER 6

USING DECISION TREES TO DISCARD IRRELEVANT FEATURES

The Kenmore framework uses an inductive learning component to acquire knowledge for conceptual sentence analysis.[1] In the last chapter, we saw how MayTag successfully used a case-based reasoning algorithm as its inductive learning component to acquire lexical disambiguation knowledge for the CIRCUS sentence analyzer. In general, however, the performance of any learning algorithm depends to a large extent on the representation used to encode the training cases. Unfortunately, choosing an appropriate case representation is both a time-intensive and knowledge-intensive task and is another potential knowledge engineering bottleneck during system development. As a result, we take the following approach to designing an adequate case representation within the Kenmore framework. First, because Kenmore's cases encode the progress of sentence analysis, we choose a case representation that is natural for the sentence analyzer. This means that the representation should adhere to the sentence analyzer's major language processing assumptions. CIRCUS, for example, recognizes major syntactic constituents like the subject, verb, and direct object of a sentence, so any case representation used with the CIRCUS sentence analyzer should include those constituents. Second, once this baseline case representation has been created, we will rely on entirely automated approaches to improve the representation by discarding irrelevant features, associating appropriate weights with the features, and adding new features.

This thesis presents two approaches for improving a baseline case representation. One method has been used in conjunction with the WHirlpool system (Chapter 8) and will be described in Chapter 8.6. The second method has been used in conjunction with MayTag and was described at a high level in the last chapter. It is a domain-independent approach that uses decision trees to find the relevant features in the representation, those features that aid a particular classification task. Any features that are not deemed relevant to the task can then be discarded from the representation. The main goal of this chapter is to describe and evaluate this decision tree approach to feature selection. Section 6.1 first presents the approach; Section 6.2 evaluates it using the lexical acquisition task described in the last chapter. We compare three case-based solutions to that problem: (1) a solution that accesses all context features during case retrieval (i.e., MayTag's original case-based

[1] Some of the material from this chapter appeared originally in Cardie [1993b].

reasoning component), (2) a solution that uses only the “relevant” context features during case retrieval as determined by the decision tree subsystem, and (3) a solution that uses a set of human-generated “relevant” features during case retrieval. In addition, all of the case-based solutions are compared to a solution that uses the decision tree directly. Experiments show that the solution that includes the automated approach to feature set selection performs the best of the case-based solutions and also outperforms the pure decision tree approach. Moreover, a related result emerges in the experiments of this chapter: MayTag consistently achieves better results when the context made available to the learning component is appropriately limited, indicating that lexical ambiguity resolution in CIRCUS can proceed without the use of all available syntactic and semantic cues.

6.1 A Decision Tree Approach to Feature Set Selection

In Section 5.4.2, we described the decision tree approach to feature set selection at a high level as a black box that takes as input a set of training cases for a particular classification task described in terms of attribute-value pairs. It produces as output a list of those attributes that are useful for the task. This section provides the details of the feature selection algorithm.

A decision tree is a data structure or knowledge structure that represents a set of rules for classification. Figure 6.1 shows a simple decision tree that might be used to classify balls into one of five classes — a tennis ball, basketball, baseball, kickball, or football. Each internal node in a decision tree represents a single feature of a ball (e.g., its **diameter**, **color**, or **covering**) and edges from a parent node to its children represent various values of the parent node feature (e.g., the covering of a ball can be one of *leather* or *fuzzy*). Leaves in the tree provide the class information, in this case the type of ball. Once the tree is created, a new instance of a ball is classified by starting at the root, testing the specified feature at each node, and branching to the appropriate child until a leaf node is reached. The decision tree in Figure 6.1, for example, would classify a brown, fuzzy ball, four inches in diameter as a football.

Although there are a number of methods for building a decision tree that are consistent with a set of training examples, we use the C4.5 decision tree system [Quinlan, 1992]. In building a decision tree, C4.5 employs a metric from information theory (i.e., *information gain ratio*) to decide which feature to test at each branching point. The information gain ratio metric finds the feature that best discriminates among the remaining training cases assigned different class values. Features that become part of the decision tree, therefore, are features that C4.5 has found to be important for the current classification task; features omitted from the tree are regarded as irrelevant.

Figure 6.2 illustrates the decision tree approach to finding the relevant features in a case representation. The algorithm begins with a set of training cases, each of which is described in terms of n features, f_1 through f_n, and the class value to be associated with the case. For MayTag’s lexical acquisition task, features f_1 through f_n are the local and

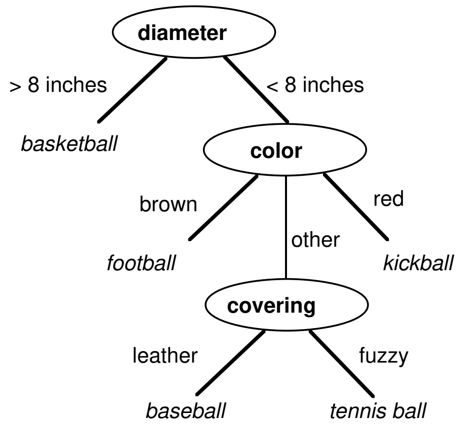

Figure 6.1: Simple Decision Tree for Classifying Balls.

global context features that encode the state of the parser at an ambiguity point and the class c to be predicted is one of the word definition features for the current word — the part of speech, semantic features, or concept to associate with the word. C4.5 then uses the training cases to create a decision tree for predicting the value of class c. Finally, the algorithm notes which features occur in the tree.[2] These are the features that C4.5 found useful for predicting the value of c for the training cases; they are returned as the "relevant" features in the representation. Features that are not referenced in the decision tree can be discarded from the representation as irrelevant to the task.

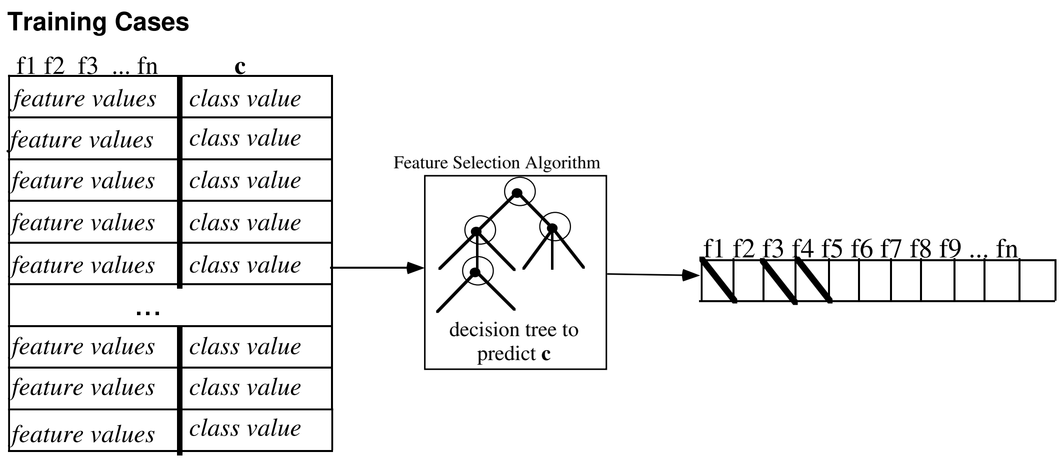

Figure 6.2: Decision Tree Feature Selection.

[2]C4.5 produces a full decision tree and a pruned tree. Unless otherwise noted, we use the pruned tree in all experiments.

6.2 Evaluation of the Approach

In the next sections, we evaluate this decision tree method for feature set selection. In all experiments we use the following simplification of the problem tackled by the MayTag system:

> Given the context in which an unknown word occurs as a set of attribute-value pairs, predict the word's part of speech, general semantic feature(s), and specific semantic feature(s).

It is a simplification of the original problem in that no concept activation information will be predicted. To evaluate the decision tree feature selection algorithm on this task, we compare three case-based solutions to the problem:

1. The first solution employs all context features during case retrieval (Section 6.2.1). It is equivalent to MayTag's original case-based reasoning algorithm (see Section 5.4).
2. Next, we use the same general case-based algorithm as above, but rely on a human-generated set of "relevant" features during case retrieval (Section 6.2.2).
3. Finally, we use the decision tree feature selection algorithm to find the "relevant" context features and then access just these features during case retrieval (Section 6.2.3). This variation is equivalent to MayTag's modified case retrieval algorithm and was employed in all experiments in Chapter 5.

From a case-based reasoning perspective, the human- and decision-tree-generated "relevant" feature sets are being used to handle the *case indexing problem* — the problem of assigning labels to cases so that only applicable ones are retrieved at the appropriate times. Most CBR systems rely on indexing schemes specified by the system designers that are based on expert knowledge of a particular domain. The HYPO system [Rissland *et al.*, 1984, Ashley, 1990], for example, retrieves relevant cases in the trade secrets legal reasoning domain based on a predetermined set of *dimensions* that influence the outcome of cases in this domain. Examples of dimensions in that domain are the **competitive-advantage** dimension (i.e., the extent to which the defendant's access to the plaintiff's trade secrets gave the defendant a competitive advantage), and the **disclose-secrets** dimension, (i.e., the extent to which the plaintiff had already disclosed the trade secrets to people outside the company). Given a set of dimensions for the domain, HYPO restricts case retrieval to those cases that address dimensions that are present in the problem case. Similarly, the ReMind case-based reasoning tool [Cognitive Systems, 1992] provides two mechanisms for specifying features that will be important during case retrieval. First, the user can assign weights to each feature in the case representation or can specify that some features should be ignored entirely when ReMind's nearest-neighbor case retrieval algorithm is used. Second, the user can define *prototypes* — a handcrafted combination of specific attribute-value pairs that will always be associated with a particular outcome or solution feature. An example of a prototype in the "balls" domain might be the following: If *diameter* $>$ *8 inches* and *covering* = *leather*, then the ball is a basketball. When prototypes are used as the case retrieval mechanism and an incoming case matches one of the predefined prototypes, ReMind retrieves all training cases indexed under the prototype. MayTag's decision tree approach to case indexing differs from HYPO's dimensions and ReMind's feature weights and prototypes in that it *automatically* determines the features that are relevant for the current retrieval task. ReMind, however, also has a mechanism for using decision trees to create indexes automatically.[3] It will be discussed in Section 6.2.3 along with other automated papproaches to case-indexing after the details of MayTag's hybrid case retrieval algorithm have been presented.

In the experiments below that compare three case-based solutions to MayTag's lexical acquisition problem, we draw the training and test instances from MayTag's base set of 2056 cases, one case for each occurrence of an open-class word in 120 sentences of the JV corpus. In addition, all experiments use 10-fold cross validation: we randomly partition the case base into ten segments and, in each of ten runs, use nine segments as the case base and keep the remaining segment for testing. The runs progress through all ten training/testing combinations. In particular, we emphasize that the same ten training and test set combinations were used in the 10-fold cross validation of each experiment. All experiments simulate the occasional-unknown-word task introduced in the last chapter.

The baseline case representation used in the experiments below predates the one used in the MayTag experiments of Chapter 5. Nevertheless, it is nearly identical to the MayTag case representation described in detail in Section 5.3. The only difference is that the representation used here includes "semantic" features for verbs. These are used to provide information about the use and tense of the verb. This difference is highlighted in Figure 6.3, which depicts the training case for "venture" as it is used in a sentence from the JV corpus.

6.2.1 Case Retrieval Using All Context Features

MayTag's original case-based reasoning algorithm for learning lexical disambiguation knowledge was described in Section 5. During the training phase a flat case base of training examples is created. During the application phase, cases most similar to the current context are retrieved and used to resolve the current lexical ambiguity. MayTag's original algorithm for case retrieval as described in Section 5.4 employs all context features in its similarity metric and is essentially a k-nearest neighbors (*k-nn*) algorithm with a bias toward examples of the unknown word encountered during training:

1. Compare the test case to each case in the case base, counting the number of context features that match (i.e., match = 1, mismatch = 0). Only give partial credit (.5) for partial matches and matches on *nil*'s.

2. Keep the k highest-scoring cases.[4]

---

[3]Prototypes are actually used in conjunction with ReMind's decision tree indexing scheme.

[4]In the original MayTag case retrieval algorithm, $k = 10$. Here we will test three values of k.

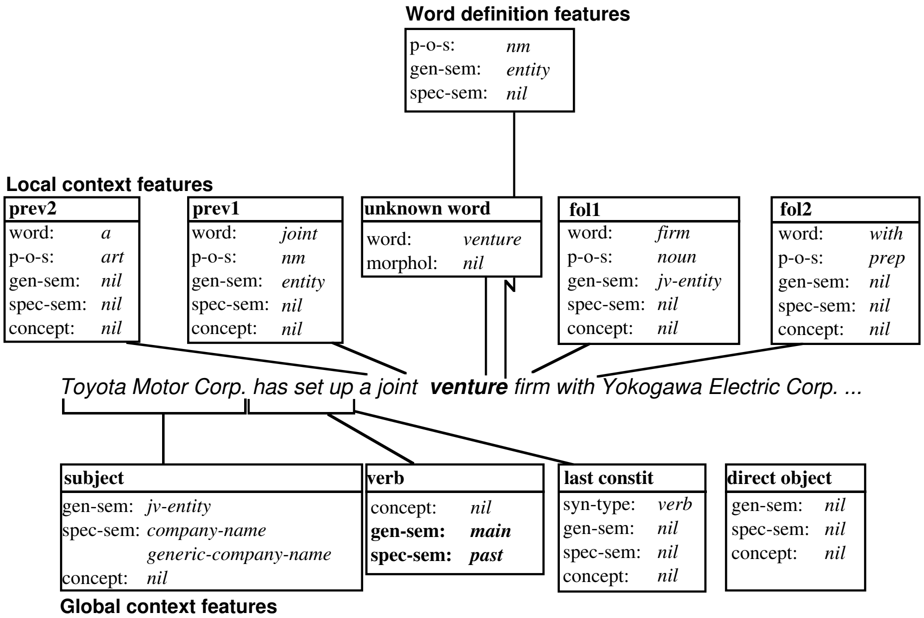

Figure 6.3: Training Case for "venture" as it is Used in the Sentence "Toyota Motor Corp. has set up a joint venture firm with Yokagawa Electric Corp..."

3. Of these, return the case(s) whose **word** matches the unknown word, if any exist. Otherwise, return all k cases.
4. Let the retrieved cases vote on the definition of the current unknown word. Break ties randomly.

The case retrieval algorithm first finds cases that best match the context of the problem case and then restricts retrieval, if possible, to cases that were generated during training in response to the same unknown word. As discussed in the previous chapter, this allows the system to generate novel interpretations for words that were seen during training, but encountered in very different contexts than the current one.

Table 6.1 shows the accuracy of the above algorithm in predicting the part of speech and semantic features of occasional unknown words in the test sentences for k= 1, 5, and 10.[5] The results are compared to two baselines. The first baseline (Random Selection) indicates the expected accuracy of a system that randomly guesses a legal value for each missing word definition feature based on the distribution of values across the training set. The second baseline (Default) shows the performance of a system that always chooses the most frequent value as a default. Chi-square significance tests indicate that the case-based approaches always outperform the baselines at the 99% significance level.

[5]As in Chapter 5, we allow mismatches between the noun and noun modifier parts of speech because the parser can accurately fix these errors.

Table 6.1: Case Retrieval Using All Context Features (% correct).

Word Definition Feature	CBR (k=1)	CBR (k=5)	CBR (k=10)	Random Selection	Default
p-o-s	86.6	88.8	89.4	34.3	81.5
gen-sem	58.5	66.2	69.1	15.9	25.6
spec-sem	62.9	70.4	72.2	24.7	45.3

6.2.2 Case Retrieval Using Human-Generated Relevant Features

The problem with the original MayTag case retrieval algorithm as described above is that it assumes that all context features are equally important for predicting each of the word definition features for an unknown word. Intuitively, however, it seems that accurate prediction of each class of missing information may rely on very different subsets of the feature set. Indeed, it has been shown that nearest-neighbor algorithms perform poorly in the presence of irrelevant features [Aha *et al.*, 1991, Aha, 1989].

One general method for optimizing our case retrieval algorithm for a particular interpretation task is to include in the k-nn calculations only those features deemed relevant to the task according to some collection of expert knowledge. As described at the beginning of Section 6.2, HYPO's dimensions and ReMind's prototypes allow expert domain knowledge to restrict case retrieval to relevant portions of the case base.

In the same way, we can optimize MayTag's case retrieval algorithm for prediction of each word definition feature by incorporating into the similarity metric informed intuitions about the nature of each class of knowledge to be predicted. Some successful part-of-speech taggers, for example, make decisions based only on knowledge of the words in a window to either side of the unknown word. This implies that the k-nn routine should only include the local context features in its calculations — the global context features may be irrelevant to the part-of-speech tagging task. On the other hand, the semantic features of an unknown word seem to depend partially on local context and partially on knowledge about the global state of the parse. For example, the semantic class of a noun in the direct object position may depend on the semantic class of the clause's subject. The direct object is much more likely a company if it follows "IBM bought..." than "John bought..." Therefore, when predicting the semantic class of a lexical item (e.g., *human*, *company-name*), it might be better first to find the most similar cases using the local context features and then to choose from these the cases that match best along both the local and global context dimensions.

We incorporated these observations into two variations of the original MayTag case retrieval algorithm. The first variation, labeled the "p-o-s" variation, was designed to improve part-of-speech (**p-o-s**) prediction and uses only the local context features in its k-nn comparisons. The second variation, labeled the "semantic feature" variation, was designed to improve prediction of the **gen-sem** and **spec-sem** word definition features. It submits those cases initially selected using just local context features to an additional k-nn filter that includes the global context features as well. Like the original case retrieval algorithm, both intuitive variations also prefer cases whose **word** feature matches the unknown word.

Table 6.2 shows the results of using the human-generated relevant feature sets for case retrieval and compares them to MayTag's original case retrieval algorithm that accessed all context features in the nearest-neighbor calculations. Only the results for k=10 are shown, but runs using k=1 and 5 exhibited similar behavior. Also shown in the table are annotations for statistical significance. (*'s indicate performance of human-generated feature sets as compared to the "all features" variation.) As expected, focusing on local

Table 6.2: Case Retrieval Using Human-Generated Relevant Features (Table shows % correct for k=10; ** indicates significance with respect to the "all features" variation; $p = .01$).

Word Definition Feature	P-O-S Variation	Semantic Feature Variation	All Features Variation
p-o-s	91.4**	91.0**	89.4
gen-sem	70.3	69.4	69.1
spec-sem	73.6	72.2	72.2

features improved part-of-speech prediction when compared to the original case retrieval algorithm. This result is consistent with other results that have shown part-of-speech tagging to rely mainly on local, lexical information (e.g., Brill [1994]).[6] As shown in Table 6.2, the semantic feature variation unfortunately did not improve performance of semantic feature prediction. Instead, it unexpectedly improved performance of part-of-speech prediction. This seems to indicate that semantic feature disambiguation does not require access to global context features often enough to warrant their inclusion in the nearest-neighbor matching.

In more general terms, the above experiments indicate that it is possible to use informed intuitions and expert domain knowledge to discard irrelevant attributes from a case representation, but that the results are not always predictable.

[6]It should also be noted, however, that a global context feature can only contribute fully to the similarity score when (1) the same syntactic constituent (e.g., subject, direct object) exists in both the problem case and the training case, and (2) their semantic classes match. As a result, the system is biased toward using the global context features only for tasks that rely on semantic rather than syntactic cues, and suggests that accuracy in part-of-speech assignment does not rely on the kind of semantic information encoded in MayTag's global context features.

6.2.3 *Hybrid Case Retrieval Algorithm*

The experiments in the last section relied on handcrafted feature sets to improve the performance of MayTag's k-nn case retrieval algorithm. Given that feature set specification is a notoriously time-consuming and knowledge-intensive task [Quinlan, 1983], however, it would be better if the feature set could be chosen systematically and automatically. This is exactly what the decision tree algorithm for feature selection described in Section 6.1 was designed to do. The experiments below use a hybrid case retrieval algorithm that combines the decision tree feature selection algorithm with the usual nearest-neighbor case retrieval algorithm — the decision tree selects the features to be included in the k-nn calculations. This is MayTag's modified case retrieval algorithm that was described briefly in Section 5.4.2 and used in all experiments of that chapter.

As described in Section 5.4.2, the hybrid case retrieval algorithm requires two modifications to MayTag's original case-based reasoning algorithm. First, after the case base has been created, we use the decision tree feature selection algorithm to find three sets of relevant features — one set for each of the three classes of lexical knowledge to be predicted for the unknown word (rel_{p-o-s}, $rel_{gen-sem}$, and $rel_{spec-sem}$). This is performed once, prior to the first case retrieval operation, and appears in Figure 6.4.[7] Then, instead of invoking the case retrieval algorithm once for each test case, we run it three times, once for each of the three word definition features to be predicted. In the retrieval for attribute a, however, only the features in rel_a should be included in the k-nn calculations.[8] The hybrid case retrieval algorithm is depicted in Figure 6.5.

Table 6.3 shows the average performance of the hybrid case retrieval algorithm across ten runs and compares it to the "all features" and "p-o-s" variations as well as to two additional baselines.[9] The first of these baselines (Random Features) is a system that randomly chooses the features to be used in the k-nn calculations while controlling for feature set size (i.e., we use the same number of features that were used in the corresponding run for the hybrid approach).[10] The second baseline (Pure Decision Tree) shows the accuracy of a system that uses the C4.5-generated decision tree directly to predict each word definition feature. Again, only results for k=10 are shown although results for k=1

---

[7]Note that as part of the 10-fold cross validation scheme, we actually create ten decision trees for each word definition feature — one for each group of training cases.

[8]We actually only compare each test case to the entire case base once (not three times) and use the results of that comparison for each of the three k-nn calculations.

[9]We omitted the "semantic features" variation from the table because it performed poorly.

[10]This baseline was included to ensure that the performance of the decision tree approach to feature set selection is not simply a result of choosing a smaller set of features.

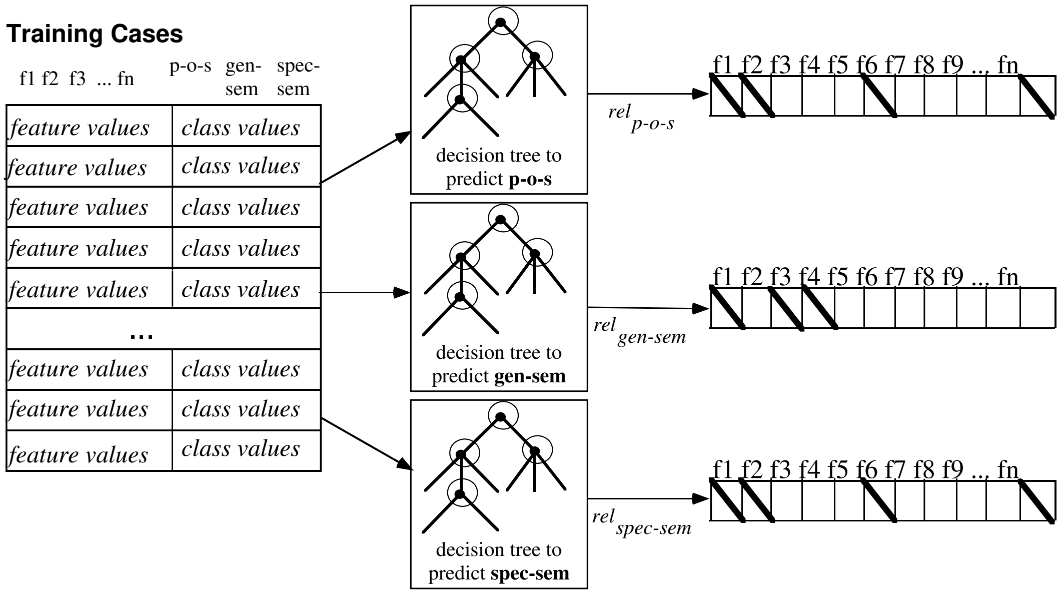

Figure 6.4: Selecting Relevant Features for the Lexical Acquisition Task.

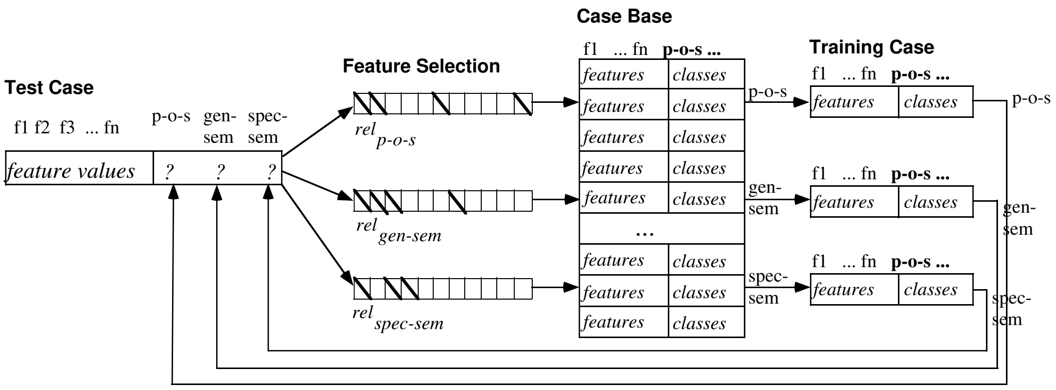

Figure 6.5: Hybrid Case Retrieval.

and 5 were much the same.[11] In all but one instance in Table 6.3, the case retrieval algorithm that used the decision tree features significantly outperforms the other approaches at the 99% level ($p = .01$). The only exception was for part-of-speech prediction, in which the decision tree features variation outperformed the "p-o-s" variation at the 95% significance level and the remaining approaches at the 99% level.

Table 6.3: Case Retrieval Using C4.5-Generated Relevant Features (Table shows % correct for $k = 10$; * and ** indicate the significance of the decision tree features variation with respect to **all** other variations. $** \rightarrow p = .01$ and $* \rightarrow p = .05$. See accompanying text for explanation of **/*.).

Word Definition Feature	Decision Tree Features Variation	All Features Variation	P-O-S Variation	Random Features	Pure Decision Tree
p-o-s	92.5**/*	89.4	91.4	89.7	89.0
gen-sem	73.4**	69.1	70.3	62.9	66.0
spec-sem	76.7**	72.2	73.6	71.1	69.9

The use of decision trees to create indexes for the case base may initially appear similar to a number of automated approaches to case indexing. For example, the CYRUS system [Kolodner, 1983] and many follow-on CBR systems (e.g., CHEF [Hammond, 1989]) use E-MOPs to organize cases hierarchically in a dynamic memory [Schank, 1982]. The resulting discrimination network is then traversed when processing new problem cases. The E-MOPs indexing scheme, however, explicitly encodes generalizations of individual cases as higher level nodes in the network. In addition, E-MOPs index a new case in terms of all of its non-"normative" features — that is, in terms of any features that are not part of the generalized event description. MayTag's use of decision trees as a case-indexing method is very different than E-MOPs: no generalizations are calculated or represented and an incoming case is indexed in terms of all attributes deemed relevant by the decision tree rather than in terms of its individual differences from a norm.

The ReMind case-based reasoning shell [Cognitive Systems, 1992] also uses decision trees as one of its available case-indexing methods. Like MayTag, the system creates a decision tree from the case base during training by determining which context features correspond to each solution feature. ReMind's decision tree generator, however, keeps track of the training cases affected by decisions at each node in the tree. After training, when a new case enters the system, ReMind presents the case to the decision tree and returns all cases indexed at the leaf. ReMind's use of decision trees, then, is equivalent to

[11] In general, we did not expect the decision tree to perform as well as the case-based approaches because attribute values in MayTag can be a list of values (e.g., the subject of the sentence may contain both a *company-name* and an *industry-product*). The case-based approaches allow partial matches on features of this type; the decision tree forces the use of a single, combined attribute value (e.g., *company-name-industry-product*).

our experiments that use the decision tree directly to determine the missing word definition features. MayTag, on the other hand, uses the decision tree to for feature set selection and then indexes each case with respect to this set of relevant features.

6.3 Analysis of Approach

The sections below take a closer look at the decision tree approach to feature set selection. In each section we provide concrete examples of the decision trees used in MayTag's hybrid case retrieval algorithm. In addition, we look more generally at the classes of features deemed relevant by the decision tree algorithm for each of MayTag's lexical acquisition tasks. We find that, for CIRCUS, the solution to each lexical ambiguity task accesses a limited subset of the available context features.

6.3.1 Decision Trees for Part-of-Speech Prediction

Figure 6.6 shows one of the decision trees created by C4.5 for part-of-speech prediction. (There were ten such trees created for the experiments above and in Chapter 5 as part of the 10-fold cross validation evaluation scheme.) The tree indicates that the part of speech of the current word should be determined by first checking the general semantic feature associated with the following word (**fol1-gen-sem**). For most values of **fol1-gen-sem**, the tree dictates that the current word is a noun modifier. However, if the general semantic feature for the following word is *industry-product*, *time*, or *nil*, then the part of speech of the preceding word (**prev1-pos**) plays a role in determining the current word's part of speech. For part-of-speech prediction, **fol1-gen-sem** was the root node feature in all ten decision trees. Although this one word lookahead seems reasonable, it may cause problems for the tagging-from-scratch task where information regarding the next word is unavailable unless the word appeared during training.

Figure 6.7 shows the frequency with which the non-"word" context features occurred in the ten rel_{p-o-s} relevant features lists for the MayTag part-of-speech experiments of Chapter 5 (Section 5.5).[12] Preceding and following context features appear to be equally important for part-of-speech prediction, especially those for the immediately preceding and immediately following words. In addition, 100% of the context features associated with syntactic knowledge appeared in the relevant features lists. The same was not true for context features associated with semantic knowledge. Finally, very few global context features appear to play a part in part-of-speech prediction although the semantic features associated with the subject of the current clause are sometimes important.

6.3.2 Decision Trees for Semantic Feature Prediction

Figure 6.8 shows one of the ten decision trees for general semantic feature prediction that was generated during the MayTag experiments of Chapter 5. The tree indicates that

[12] Recall that all "word" features were omitted from the cases used to train the decision trees.

fol1-gen-sem = generic-loc, location, person-position, ownership-percentage,
capitalization, human, facility, money, nationality, entity, jv-entity -> ***nm***
fol1-gen-sem = industry-product ->
| **prev1-pos** = conjunction ->
| | **prev2-gen-sem** = industry-product -> ***nm***
| | **prev2-gen-sem** = [others] -> ***verb***
| **prev1-pos** = [others] -> ***nm, verb***
fol1-gen-sem = time ->
| **prev1-pos** = noun ->
| | **prev1-gen-sem** = human -> ***verb***
| | **prev1-gen-sem** = jv-entity -> ***verb***
| | **prev1-gen-sem** = [others] -> ***adv***
| **prev1-pos** = nm ->
| | **fol1-pos** = noun -> ***nm***
| | **fol1-pos** = [others] -> ***noun***
| **prev1-pos** = [others] -> ***adv, verb, nm***
fol1-gen-sem = nil ->
| **prev1-pos** = connective -> ***noun, adv***
| **prev1-pos** = adv -> ***adv, verb***
| **prev1-pos** = aux, verb, comma -> ***pasp, verb, adv, noun***
| **prev1-pos** = noun ->
| | **morphol** = ed -> ***pasp, verb***
| | **morphol** = nil ->***verb, adv, pasp, modl***
| | **morphol** = [others]-> ***verb, conn, pres, adv noun***
| **prev1-pos** = determiner -> ***noun, adv, nm***
| **prev1-pos** = [others]-> ***noun, verb***

Figure 6.6: Sample Decision Tree for Part-of-Speech Prediction.

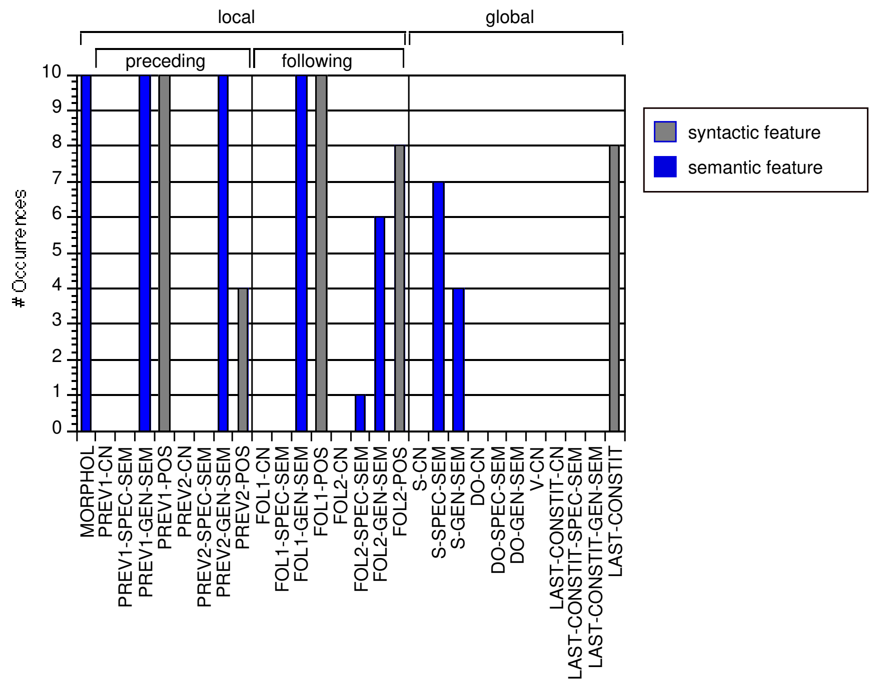

Figure 6.7: Histogram of Features From rel_{p-o-s} Lists (part-of-speech prediction).

the general semantic feature of the current word should be determined by first checking the specific semantic feature of the preceding word (**prev1-spec-sem**). If this information is missing (i.e., *nil*), then the specific semantic feature of the following word (**fol1-spec-sem**) should be noted. If we look at the decision trees associated with all ten $rel_{gen-sem}$ feature lists, we find that **prev1-spec-sem** is always the top-ranked feature. A sample decision tree for specific semantic feature prediction is shown in two parts in Figures 6.9 and 6.10. Not surprisingly, many of the same context features are tested at the top levels of both the general and specific semantic feature decision trees. Like general semantic feature prediction, **prev1-spec-sem** is always the root node feature in the C4.5 decision trees for specific semantic feature prediction.

Figures 6.11 and 6.12 show the frequency with which the non-“word” context features occurred in the ten $rel_{gen-sem}$ and $rel_{spec-sem}$ relevant feature lists for the MayTag experiments of Chapter 5 (Section 5.5). As was the case for part-of-speech prediction, 100% of the syntactic context features are referenced by at least one decision tree for general and specific semantic feature prediction. For general semantic feature prediction, the “following” local context features appear to be slightly more important than the “preceding” local context features. In addition, three semantic global context features appear to play a role — the domain-specific concepts associated with the verb and the last constituent, and the specific semantic features associated with the subject. For specific semantic feature prediction, the “preceding” and “following” local context features are equally important. Although a greater variety of global context features are referenced for specific semantic feature prediction than for general semantic feature prediction, the specific semantic feature rel lists access the global context features less often (8 vs. 23 times).

6.3.3 Decision Trees for Concept Type Prediction

Figures 6.13 and 6.14 show a sample decision tree for predicting the domain-specific concept associated with the current word. Unlike the decision trees for part-of-speech and semantic feature prediction, however, the concept type tree is the full C4.5 decision tree rather than the pruned version. The full trees were used to derive the $rel_{concept}$ lists because the pruned trees test no context features at all. They simply state that the *nil* concept type should be returned.[13] In any case, the decision trees indicate that the first feature to check for concept type prediction of an unknown word is the specific semantic feature of the second following word, i.e., **fol2-spec-sem**. **Fol2-spec-sem** is the root node feature in all ten $rel_{concept}$ feature lists. This implies that the concept associated with the current word can be determined by looking for the semantic type of potential slot fillers for the concept. If this information is missing (i.e., *nil*), then the concept associated with the second following word (**fol2-cn**) should be noted.

Figure 6.15 shows the frequency with which the non-“word” context features occurred

[13]The pruned trees were of this form because relatively few words actually activate concepts in this domain and returning the default value of *nil* works quite well overall.

```
prev1-spec-sem =  ceo, exec, sr-exec -> person-position
prev1-spec-sem =  generic-company, company-name -> jv-entity
prev1-spec-sem =  factory, mine -> entity
prev1-spec-sem =  research, government -> industry-product
prev1-spec-sem =  product+site, facility+production, other-facility,
                  facility-name -> facility
prev1-spec-sem =  sales+store, service+store -> facility+industry-product
prev1-spec-sem =  person, human-title, human-name -> human
prev1-spec-sem =  other-loc, continent -> location
prev1-spec-sem =  sales ->
|       last-constit = clause, prep-phrase, noun-phrase, nil -> jv-entity
|       last-constit = verb -> entity
prev1-spec-sem =  country ->
|       prev2-gen-sem = location, jv-entity -> jv-entity
|       prev2-gen-sem = entity -> money
|       prev2-gen-sem = [others] -> location
prev1-spec-sem =  finance ->
|       prev2-gen-sem = nationality -> jv-entity
|       prev2-gen-sem = [others] -> industry-product
prev1-spec-sem =  city ->
|       fol2-pos = infinitive -> jv-entity
|       fol2-pos = adv -> facility
|       fol2-pos = verb -> entity
|       fol2-pos = [others] -> location
prev1-spec-sem =  nil ->
|       fol1-spec-sem =  ceo, sr-exec -> person-position
|       fol1-spec-sem =  mine, transportation -> entity
|       fol1-spec-sem =  production+site -> money
|       fol1-spec-sem =  site, factory, production, store -> industry-product
|       fol1-spec-sem =  person, finance, government -> nationality
|       fol1-spec-sem =  other-loc, other-facility, city -> location
|       fol1-spec-sem =  continent, exec -> generic-loc
|       fol1-spec-sem =  service -> time
|       fol1-spec-sem =  spoke, generic-company, company-name -> jv-entity
|       fol1-spec-sem =  human-name -> human
|       fol1-spec-sem =  facility-name -> facility
|       fol1-spec-sem = research ->
|       |       prev1-pos ...
|       |       fol2-gen-sem ...
|       fol1-spec-sem = nil ->
|       |       morphol ...
|       |       |       prev2-pos ...
|       |       |       fol2-cn ...
|       |       |       |       fol1-pos ...
|       |       |       |       |       prev1-pos ...
|       |       |       last-constit-cn ...
|       |       |       fol1-gen-sem ...
  ...
```

Figure 6.8: Sample Decision Tree for General Semantic Feature Prediction (partial tree).

```
prev1-spec-sem =  company-name ->
|       fol1-gen-sem = person-position -> exec
|       fol1-gen-sem = product-service -> product
|       fol1-gen-sem = time -> generic-company
|       fol1-gen-sem = [others] -> company-name
|       fol1-gen-sem = nil ->
|       |       morphol = ing, proper -> company-name
|       |       morphol = [others] -> generic-company
|       |       morphol = s ->
|       |       |       fol1-pos = prep -> sales
|       |       |       fol1-pos = aux, conj -> company-name
|       |       |       fol1-pos = [others] -> spoke
prev1-spec-sem =  ceo -> ceo
prev1-spec-sem =  generic-company -> generic-company
prev1-spec-sem =  exec -> exec
prev1-spec-sem =  factory, mine, sales, service, city, finance -> nil
prev1-spec-sem =  research -> research
prev1-spec-sem =  sr-exec -> sr-exec
prev1-spec-sem =  production+site -> site
prev1-spec-sem =  sales+store -> sales+store
prev1-spec-sem =  person -> person
prev1-spec-sem =  other-loc -> other-loc
prev1-spec-sem =  human-title, human-name -> human-name
prev1-spec-sem =  government -> production
prev1-spec-sem =  factory+production -> factory
prev1-spec-sem =  other-facility -> other-facility
prev1-spec-sem =  facility-name -> facility-name
prev1-spec-sem =  service+store -> service+store
prev1-spec-sem =  country ->
|       prev2-gen-sem = location, entity, jv-entity -> nil
|       prev2-gen-sem = [others] -> country
```

Figure 6.9: Sample Decision Tree for Specific Semantic Feature Prediction (partial tree, part 1).

in the ten $rel_{concept}$ relevant features lists for the MayTag experiments of Chapter 5 (Section 5.5). In general, the "following" local context features are accessed more often than the "preceding" local context features. In addition, it is interesting that the decision tree accessed many syntactic features (i.e., the **last-constit** attribute and **p-o-s** features) in spite of the fact that concept activation would seem to be a purely semantic question. One explanation for this is that accurate concept activation requires clause boundary recognition and some knowledge of the type of syntactic structure that the parser is processing. Although neither of these pieces of information is included explicitly in the case representation[14], both can be inferred by examining the **last-constit** and **p-o-s** features. Finally, many global context features appear to play a role in concept activation — only the semantic features associated with the last constituent are not accessed by any of the decision trees for concept type prediction. It should be noted, however, that many more context features appear in this chart than in the previous charts simply because we are using the full C4.5 decision tree rather than the pruned version.

[14]The **last-constit** feature does note when clause boundaries have been crossed, but clause boundaries to the right of the current word are not recognized.

```
prev1-spec-sem = nil ->
|       fol1-spec-sem = ceo -> ceo
|       fol1-spec-sem = mine -> nil
|       fol1-spec-sem = sr-exec -> sr-exec
|       fol1-spec-sem = production+site, continent, exec, sales, finance, service, government,
                            transportation, store, service+store -> nil
|       fol1-spec-sem = site -> production+site
|       fol1-spec-sem = sales+store -> sales+store
|       fol1-spec-sem = person -> person
|       fol1-spec-sem = other-loc -> other-loc
|       fol1-spec-sem = factory -> factory+production
|       fol1-spec-sem = spoke, generic-company, company-name -> company-name
|       fol1-spec-sem = other-facility -> city
|       fol1-spec-sem = facility-name -> facility-name
|       fol1-spec-sem = research ->
|       |       prev1-pos = determiner -> nil
|       |       prev1-pos = [others] -> research
|       fol1-spec-sem = country ->
|       |       fol2-gen-sem = money -> nil
|       |       fol2-gen-sem = [others] -> country
|       fol1-spec-sem = production ->
|       |       prev2-gen-sem = time -> company-name
|       |       prev2-gen-sem = entity ->
|       |       |       fol1-gen-sem ...
|       |       prev2-gen-sem = nil ->
|       |       |       prev1-cn = nil -> nil
|       |       |       prev1-cn = [others] -> production
|       |       prev2-gen-sem = [others] -> production
|       fol1-spec-sem = human-name ->
|       |       prev1-pos ...
|       fol1-spec-sem = nil ->
|       |       prev2-spec-sem ...
|       |       |       fol1-pos ...
|       |       |       morphol ...
|       |       |       prev1-pos ...
|       |       |       |       morphol ...
```

Figure 6.10: Sample Decision Tree for Specific Semantic Feature Prediction (partial tree, part 2).

6.4 Problems with the Hybrid Case Retrieval Algorithm

MayTag's modified case retrieval algorithm (Section 6.2.3) combines the decision tree approach to feature set selection with a nearest-neighbor matching routine. In spite of the promising performance of this hybrid case-based reasoning algorithm on the lexical acquisition task, there are problems with the current approach. In general, the speed of case retrieval degrades linearly with the size of the case base. This is a problem with the nearest-neighbors component rather than the decision tree component, however. On the other hand, introducing decision trees for feature selection transforms MayTag from an incremental learning algorithm to a nonincremental one. This is a disadvantage if we want to employ the case-based learning algorithm during the training phase to propose definitions for the current word. One solution to this problem is to replace C4.5 with an incremental decision tree algorithm [Utgoff, 1989, Utgoff, 1994]. This would allow the system to determine the relevant attributes associated with each word definition feature as training cases arrive *without* reprocessing all preceding instances. Alternatively, one might wait until the case base was relatively stable before employing the decision tree approach to feature selection. Another option is to recompute the relevant feature sets

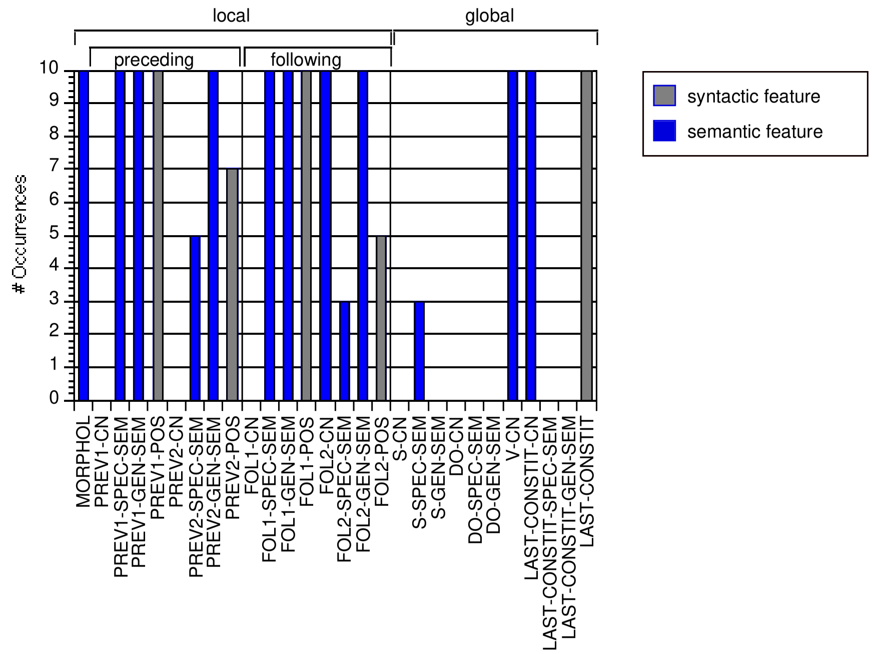

Figure 6.11: Histogram of Features From $rel_{gen-sem}$ Lists (general semantic feature prediction).

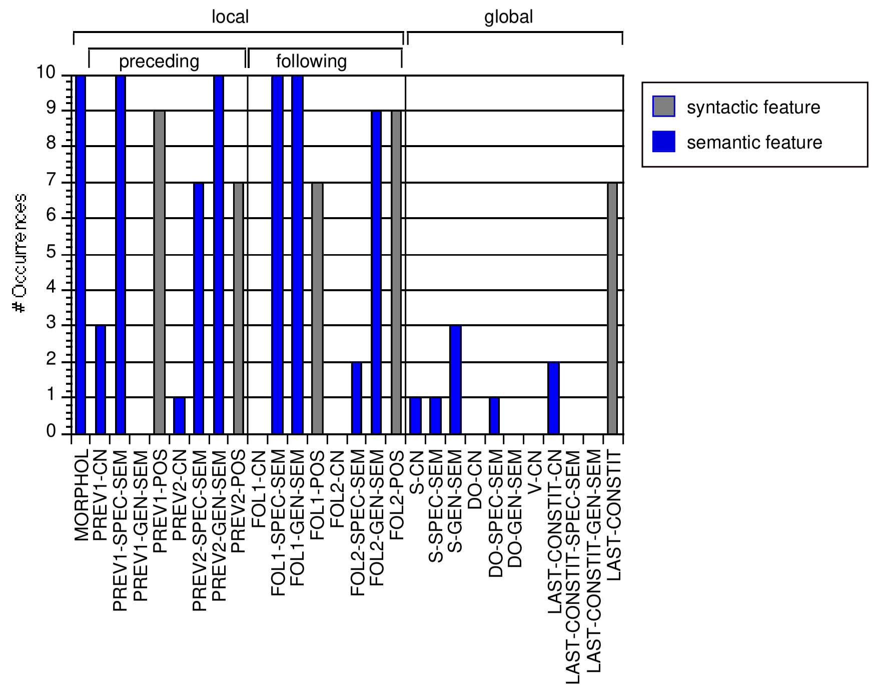

Figure 6.12: Histogram of Features From $rel_{spec-sem}$ Lists (specific semantic feature prediction).

```
fol2-spec-sem = government -> tie-up-relationship
fol2-spec-sem = research ->
|       last-constit = verb -> industry-research
|       last-constit = [others] -> nil
fol2-spec-sem = finance ->
|       fol1-pos = determiner -> industry
|       fol1-pos = [others] -> nil
fol2-spec-sem = production ->
|       do-cn = industry-research -> industry-production
|       do-cn = nil ->
|       |       prev2-cn = industry-research -> industry-research
|       |       prev2-cn = industry-production ->
|       |       |       prev1-pos = nm -> nil
|       |       |       prev1-pos = [others] -> industry-sales
|       |       prev2-cn = nil ->
|       |       |       prev1-pos = adverb, noun-phrase -> industry-production
|       |       |       prev1-pos = verb ->
|       |       |       |       s-gen-sem = entity+nationality -> industry-sales
|       |       |       |       s-gen-sem = entity+jv-entity, jv-entity -> industry-production
|       |       |       |       |       prev1-cn ...
|       |       |       prev1-pos = aux ->
|       |       |       |       s-cn ...
|       |       |       prev1-pos = [others] -> nil
|       |       prev2-cn = [others] -> nil
|       do-cn = [others] -> nil
fol2-spec-sem = country ->
|       last-constit = verb ->
|       |       prev2-pos = determiner -> tie-up-relationship
|       |       prev2-pos = [others] -> nil
|       last-constit = [others] -> nil
fol2-spec-sem = company-name ->
|       prev1-gen-sem = ownership-percentage -> ownership-%
|       prev1-gen-sem = product-service ->
|       |       fol1-pos = preposition -> tie-up-relationship
|       |       fol1-pos = [others] -> nil
|       prev1-gen-sem = entity ->
|       |       morphol = er -> tie-up-relationship
|       |       morphol = nil ->
|       |       |       fol1-pos = connective -> ownership-%
|       |       |       fol1-pos = conjunction -> nil
|       |       |       fol1-pos = comma, nm -> tie-up-relationship-jv-entity-only
|       |       morphol = [others] -> nil
|       prev1-gen-sem = nil ->
|       |       prev2-gen-sem = jv-entity ->
|       |       |       prev1-pos = aux -> tie-up-relationship-jv-entity-only
|       |       |       prev1-pos = [others] -> nil
|       |       prev2-gen-sem = [others] -> nil
|       prev1-gen-sem = [others] -> nil
```

Figure 6.13: Sample Decision Tree for Concept Type Prediction (part 1).

fol2-spec-sem = nil ->
| **fol2-cn** = industry-production ->
| | **do-gen-sem** = jv-entity, entity+jv-entity -> ***industry-research***
| | **do-gen-sem** = [others] -> ***nil***
| **fol2-cn** = industry-sales ->
| | **fol1-pos** = infinitive, aux, comma -> ***nil***
| | **fol1-pos** = [others] -> ***industry-production***
| **fol2-cn** = tie-up-relationship ->
| | **morphol** = number -> ***ownership-%***
| | **morphol** = ed -> ***tie-up-relationship-secondary***
| | **morphol** = nil ->
| | | **fol2-pos** = nm -> ***industry***
| | | **fol2-pos** = [others] -> ***nil***
| | **morphol** = [others] -> ***nil***
| **fol2-cn** = nil ->
| | **fol2-gen-sem** = ownership-percentage ->
| | | **s-gen-sem** = nil -> ***nil***
| | | **s-gen-sem** = [others] -> ***ownership-%***
| | **fol2-gen-sem** = jv-entity ->
| | | **last-constit-cn** = tie-up-relationship-secondary -> ***tie-up-relationship***
| | | **last-constit-cn** = [others] -> ***nil***
| | **fol2-gen-sem** = [others] -> ***nil***
| | **fol2-gen-sem** = nationality ->
| | | **v-cn** ...
| | | | **prev1-gen-sem** ...
| | | | | **fol1-pos** ...
| | **fol2-gen-sem** = entity ->
| | | **prev1-pos** ...
| | | | **do-cn** ...
| | | | | **fol1-spec-sem** ...
| | | | **morphol** ...
| **fol2-cn** = [others] -> ***nil***
fol2-spec-sem = [others] -> ***nil***

Figure 6.14: Sample Decision Tree for Concept Type Prediction (part 2).

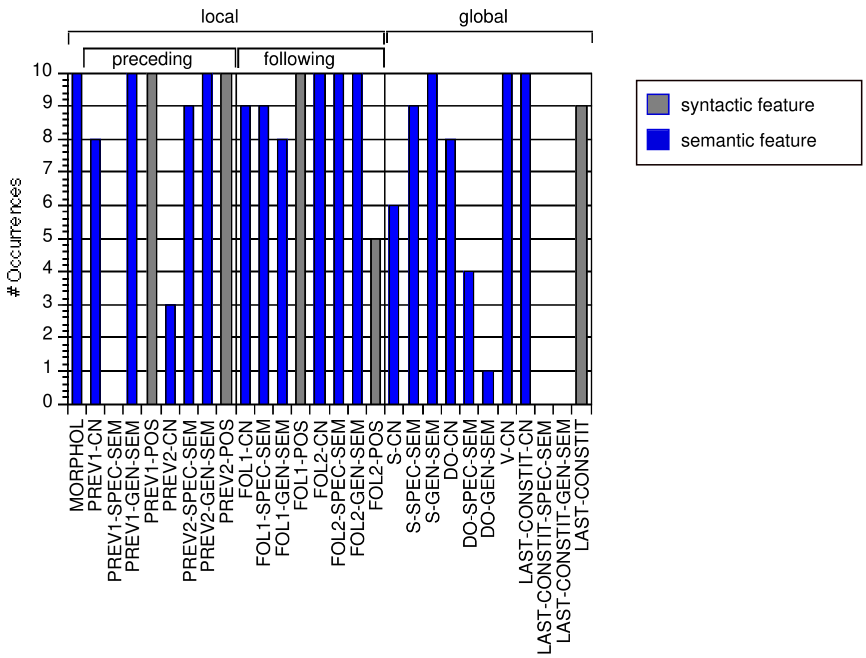

Figure 6.15: Histogram of Features From $rel_{concept}$ Lists (domain-specific concept prediction).

(rel_{p-o-s}, etc.) only occasionally. In the last two solutions, however, we might lose some of the inherent advantages associated with the incremental nature of case-based reasoning algorithms. Finally, our decision tree approach to feature set selection should be tested on additional data sets to determine its generality and to determine the class of problems that will respond favorably to the technique.

6.5 Related Work in Automated Feature Set Selection

We are not the first to use decision trees to find a set of relevant features in a case representation. Aha and Kibler [1987] used decision trees to specify the features to be included in k-nearest neighbor case retrieval. They evaluated their approach using two datasets from a medical domain task. However, they obtained mixed results — the hybrid case retrieval approach marginally improved overall classification accuracy for one dataset (from 96% to 97% correct) but degraded accuracy for the other (from 89% to 84%). In related work, Aha [1989] presents a method for learning concept-dependent attribute relevancies in a case-based paradigm. He dynamically updates the similarity function for each concept by modifying its attribute weight vector in response to classification performance. Here we use decision trees essentially to create an attribute weight vector for each concept where the weights are either zero or one. However, one possibility that we have not yet explored is to use the information gain metric directly to derive attribute weights. This approach has been used in Daelemans *et al.*[1993] and Daelemans [in press] to find attribute weights for a number of low-level language learning tasks.

The results of the experiments in this chapter indicate that the decision tree approach to feature set selection is a viable method for improving a baseline case representation. This result has important implications for work in case-based paradigms because it clearly indicates that decision tree algorithms can be used to improve the performance of some CBR systems without reliance on potentially expensive expert knowledge. On one hand, these results may not seem surprising since previous research has found the converse to be true — Skalak and Rissland [1990] show that a case-based reasoning system can successfully perform the feature selection task for a decision tree classification system. However, Almuallim and Dietterich show that the ID3 decision tree system [Quinlan, 1986] is not particularly good at selecting a minimum set of features from an original set containing possibly many irrelevant attributes [Almuallim and Dietterich, 1991]. While their results may hold in general, we claim that there is at least one important class of problem for which decision tree algorithms can perform feature specification reasonably well. In particular, this approach may succeed when the baseline case representation contains many irrelevant features.

Chapter 7

MayTag Performance Analysis

The last two chapters described MayTag, an instantiation of the Kenmore framework that acquires lexical knowledge, as well as its decision tree approach to feature set selection. This chapter examines more closely MayTag's ability to handle lexical ambiguities. In it, we discuss when and why our case-based approach to lexical disambiguation succeeds and when and why it fails. In addition, we outline possible solutions to some of the problems that were discovered.

7.1 Part-of-Speech Disambiguation

Figure 7.1 summarizes MayTag's performance for part-of-speech prediction in two different tasks: (1) predicting the part of speech for **occasional unknown words** in the test sentences and (2) predicting the part of speech for all words in a test sentence — **tagging from scratch**. These results were presented in Chapter 5 where a detailed description of the experiments can be found. Briefly, the experiments of that chapter draw from a base set of 2056 cases created during a human-supervised training phase. Each case represents the context-dependent interpretation of a single occurrence of an open-class word from 120 sentences in the JV corpus. In each experiment, MayTag relied on a hand-coded lexicon of 129 function words to provide part-of-speech information for closed-class words and drew from the interpretations stored in the case base to predict the part of speech for all other (open-class) words in each test sentence. The results in Figure 7.1 show MayTag's average percentage correct using 10-fold cross validation and are compared to two baselines. The first baseline (Random Selection) indicates the expected accuracy of a system that randomly guesses a legal part of speech based on the distribution of parts of speech (for the open-class words) across the training set. The second baseline (Default) shows the performance of a system that always chooses the most frequent part of speech as a default. MayTag's performance on the open-class words is shown in the third column of each block of results and MayTag's performance across all words in the test sentences is shown in the final column of each block. In general, MayTag's overall performance for part-of-speech prediction is comparable to the performance of statistical part-of-speech taggers, which generally achieve accuracies in the 96% range [Charniak, 1993].

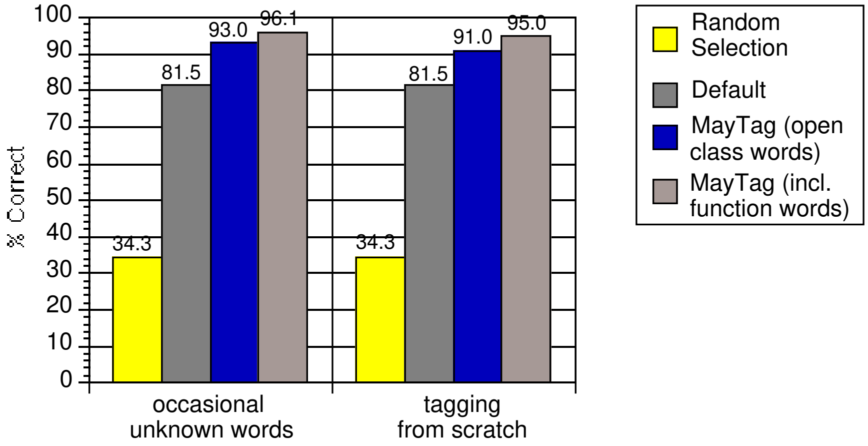

Figure 7.1: Part-of-Speech Prediction (% correct).

As described in Chapter 5, the part-of-speech taxonomy used by MayTag contains 18 parts of speech (see Table 5.4). Seventeen of 18 part-of-speech tags occur across the **prev1-pos**, **prev2-pos**, **fol1-pos**, and **fol2-pos** features of the training cases. (Only copulars do not occur.) However, only 11 parts of speech were selected by the human supervisor as values for the part of speech of the unknown word represented by each training case, i.e., for the part-of-speech word definition feature. The distribution of these values is shown in Table 7.1. Although most closed-class words are defined in the function word lexicon, the table shows that some are represented in the case base (e.g., *modl*, *conn*, *prep*). This happens in one of two situations: (1) a closed-class word is missing from the function word dictionary (e.g., the connective "otherwise" appeared in the training sentences, but was not in the function word dictionary), or (2) a particular closed-class word is very ambiguous for the domain and, hence, was removed from the function word dictionary so that MayTag might perform the part-of-speech disambiguation. For example, in the joint ventures domain the word "may" is sometimes a modal (e.g., "IBM *may* begin building...") and sometimes a noun (e.g., "The company will open in *May*"). Rather than design a complicated procedural definition for "may" that can determine when the noun form is preferred over a modal (and vice versa), we simply exclude the word from the function word dictionary altogether and let MayTag provide the disambiguation.

In any case, it is clear from Table 7.1 that the vast majority of open-class words are either nouns or noun modifiers.[1] Because MayTag receives more training on words of these classes, one would expect MayTag's accuracy for noun and noun modifier prediction to be quite good. This is, in fact, the case. Figure 7.2 shows the breakdown of MayTag's performance on individual parts of speech for the occasional-unknown-word task using

[1]The number of nouns and noun modifiers is high for two reasons: (1) the texts contain many company names and product descriptions comprised of noun modifiers and nouns, and (2) the part-of-speech taxonomy does not have separate categories for proper nouns and pronouns — both will be labeled as nouns or noun modifiers depending on their position within the noun phrase.

Table 7.1: Part-of-Speech Frequencies for Unknown Words (2056 cases).

Part of Speech	Frequency
modl (modal)	1
conn (connective)	2
prep (preposition)	3
ger (gerund)	5
pres (present participle)	15
ptcl (particle)	24
pasp (past participle)	73
adv (adverb)	83
verb (verb)	175
noun (head noun)	792
nm (noun modifier)	883

the 2056-instance case base.[2] The graph shows two values for each part of speech, p. The $*$'s indicate the percentage of unknown words that should have been tagged with p and the corresponding x's show MayTag's accuracy in predicting p. The graph clearly shows that MayTag performs well overall because it performs well on the frequently occurring parts of speech. Taken together, the noun and noun modifier categories account for 81.5% of the test cases of which MayTag correctly tags 98.6% correctly. In general, the graph indicates that accuracy on individual part-of-speech prediction improves as representation of that part of speech in the training set increases.

7.1.1 Types of Errors

It is informative to examine the kinds of errors made by MayTag during part-of-speech prediction. MayTag's errors for the occasional-unknown-word task are shown in Table 7.2. Each row of the table corresponds to a part of speech, p, while the columns indicate how words that should have been tagged with p were misclassified by MayTag. Verb particles (PTCL), for example, were mistaken once for a preposition (PREP) and twice for noun modifiers (NM). In sentence analysis, some errors cause more trouble than others. Recognizing an adverb (ADV) as a verb particle (PTCL), for example, will have little effect on the meaning representation derived by CIRCUS for a sentence, for example. The most grievous part-of-speech tagging errors are those that mistake non-verbs for verbs and vice versa.[3] All such errors are marked with *'s in Table 7.2. Approximately 46% of MayTag's errors were of the verb/non-verb variety. Of these, 66% were due to false hits on nouns

[2] The average accuracy for part-of-speech prediction of the open-class words in this 10-fold cross validation run was 92.8%.

[3] Here we use "verb" to denote any of MODL, PASP, PRES, or VERB.

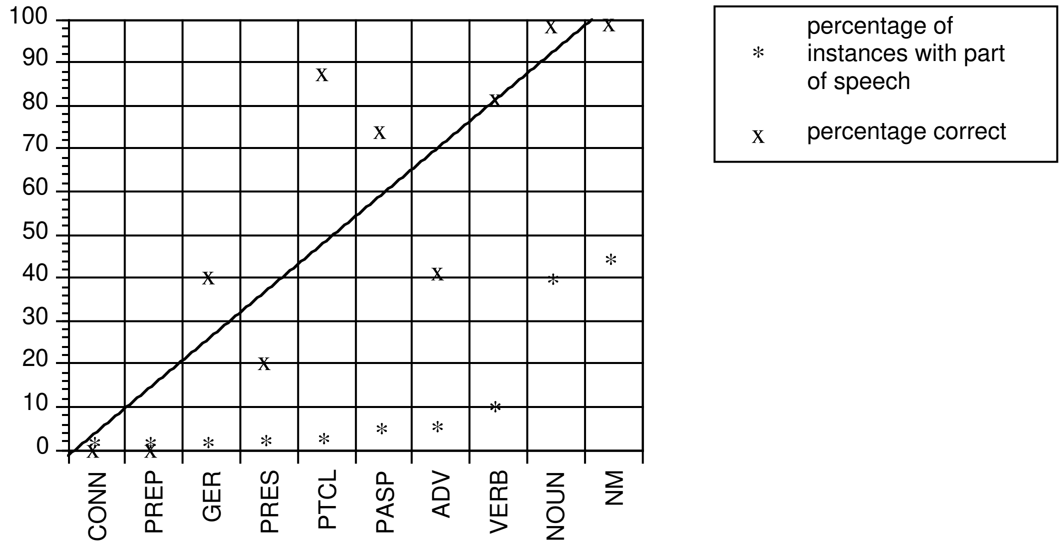

Figure 7.2: Performance on Individual Parts of Speech for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting part of speech, p; $*$'s indicate % of instances that should have been assigned p; line is best linear approximation to x's.)

or noun modifiers. For the tagging-from-scratch task, 55% of MayTag's errors were of the verb/non-verb variety and 79% of these were due to false hits on nouns and noun modifiers.

There are a number of possible reasons for the large number of false hits. First, the case retrieval algorithm uses a rather high value for k (i.e., ten) in its nearest-neighbor retrieval algorithm and all retrieved cases vote on the value of the unknown word's part of speech. In fact, more than ten cases are sometimes retrieved when there are ties. Because there are many more noun/nm cases in the case base, the majority vote tends to be one of noun or noun modifier even though some of the retrieved cases cast (correct) dissenting votes. For example, MayTag retrieved 15 cases in determining the part of speech for "decided" in the sentence,

Tamotsu Goto, JAL senior vice president, said the airline **decided** to take part in the project...

The words associated with these 15 cases are shown in rank order in Table 7.3 with their corresponding part of speech. Of the 15 cases, five vote that "decided" should be a *noun*, four vote that it should be a *verb*, four vote for *pasp*, and two vote for *noun modifier*. In spite of the fact that together the *verb* and *pasp* votes account for the majority of retrieved cases (and indicate at the very least that "decided" should be one of the verb classes), MayTag's simple voting scheme assigns "decided" the *noun* part of speech.

7.1.2 Reducing False Hits on Majority Classes

One method for reducing the number of verb/non-verb errors is to tackle the general problem of reducing the number of false hits on nouns and noun modifiers. One way

Table 7.2: Part-of-Speech Error Matrix for Occasional-Unknown-Word Tagging. (*'s indicate verb/non-verb errors.)

	MODL	CONN	PTCL	PREP	PASP	NOUN	PRES	ADV	NM	GER	VERB
MODL											1
CONN						2					
PTCL				1					2		
PREP			1						1		1*
PASP						7*			6*		6
NOUN			4					3			5*
PRES					2	1*			4*		5
ADV			2		1*	18	1*		21		6*
NM			2		2*			1			6*
GER									3		
VERB			1*		5	7*			20*		

Table 7.3: Unknown Words and Parts of Speech for Retrieved Cases.

Unknown Word	Part of Speech
plan	verb
wanted	verb
expected	pasp
involved	pasp
held	pasp
due	nm
firm	noun
venture	noun
said	verb
venture	noun
construction	noun
intends	verb
scheduled	pasp
project	noun
established	nm

to do this might be to use smaller values for k in the case retrieval algorithm. In all, 62% of MayTag's errors were false hits on nouns and noun modifiers for the occasional-unknown-word task when k was ten. As shown in Table 7.4, the false hit rate drops to 57% by lowering k to five, and to 45% when k is set to one. The second row of the table shows that MayTag's overall accuracy for part-of-speech prediction is unaffected when k is set to five and declines only slightly when k is set to one.[4] A bigger question is whether smaller values of k will reduce the number of verb/non-verb errors. In the "decided" sentence from above, for example, retrieving just the top-ranked case would result in the correct part-of-speech assignment and retrieving the top five cases would at least allow MayTag to choose one of the verb parts of speech (i.e., *pasp*) rather than a non-verb. The results shown in the bottom row of Table 7.4, however, do not support this hypothesis. As described above, 46% of MayTag's errors in the occasional-unknown-word task are of the verb/non-verb variety when k is ten. When k is five, 50% of MayTag's errors in the occasional-unknown-word task are verb/non-verb errors. When k is one, this percentage drops back to 46%.

Table 7.4: Using Smaller Values of k for Part-of-Speech Prediction in the Occasional-Unknown-Word Task (Results shown are for open-class words only using the 2056-case case base).

MayTag Results	$k = 10$	$k = 5$	$k = 1$
% false hits	62	57	45
% correct	92	92	91
% verb/non-verb errors	46	50	46

There are additional methods available for reducing the number of false hits and, thus, possibly reducing the number of verb/non-verb errors. One option is to modify the case retrieval voting scheme to assign greater weight to cases with low-frequency part-of-speech values. Alternatively, each vote might be weighted by the degree of similarity between the retrieved case and the problem case, or additional cases for under-represented parts of speech (and for the verb classes in particular) could be included in the training set. Finally, we might modify the case retrieval algorithm to use information regarding the verb/non-verb distinction explicitly. The retrieved cases could first vote as to whether the unknown word was a verb or a non-verb and then only those cases of the appropriate general class would participate in the final vote for a particular part of speech.

A final source of the false hit problem might be lurking in the hybrid similarity metric itself. As described in Section 5.4, MayTag's nearest-neighbor calculations include only those features associated with nodes in the part-of-speech decision tree — the decision tree

[4] Again, note that we are using results from a different 10-fold cross validation run than those presented in Chapter 5.

generated to predict the part of speech for a word given its context. However, 81.5% of the instances used to train the decision tree fall into either the noun or noun modifier class. As a result, the features that appear as nodes in the corresponding decision tree more than likely will be those that aid the recognition of nouns and noun modifiers rather than one of the minority parts of speech. One potential solution to this problem is to use the decision tree to find *case-specific* relevant feature sets rather than a single set of relevant features that applies to all problem cases. Given a problem case, for example, we would first send the instance through the decision tree and note the features that were tested during the tree traversal. Then just those features would be included in the nearest-neighbors calculations. This approach is different from the current case retrieval algorithm which includes **all** features that appear in the decision tree in all nearest-neighbors calculations.

7.2 Semantic Feature Disambiguation

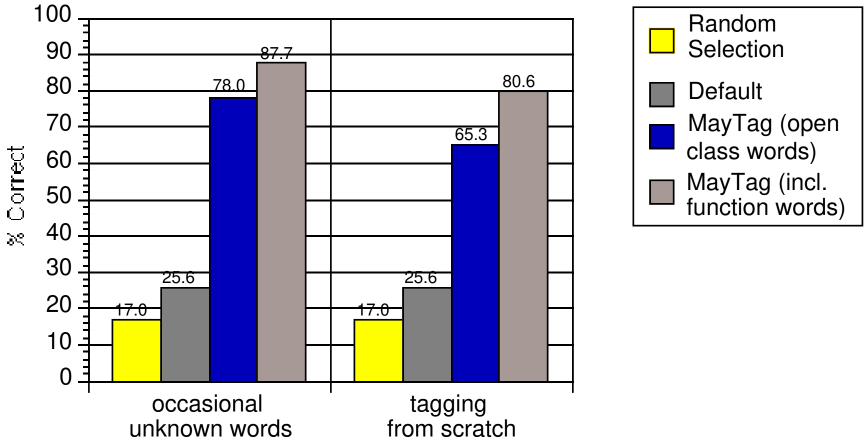

Figure 7.3: General Semantic Feature Prediction (% correct).

Figures 7.3 and 7.4 summarize MayTag's performance for semantic feature prediction for tagging occasional unknown words and for the tagging-from-scratch task using a case base with 2056 cases. These results and the associated experiments were presented in Chapter 5. As was the case in analyzing MayTag's performance on part-of-speech prediction, MayTag's ability to predict the semantic features of an open-class word depends on the quality of its training cases, that is, on how well the training cases cover the semantic feature disambiguation concepts to be learned.

As described in Chapter 5, the semantic feature taxonomy (see Tables 5.1 and 5.2) includes 14 general semantic features and 42 specific semantic features and reflects domain-specific distinctions that are important for an NLP system processing texts from the JV corpus. All 14 general semantic features occur across the **prev1**, **prev2**, **fol1**, and **fol2** features in the 2056 training cases. However, 12 of the 42 specific semantic features did not occur across those local context features in the training cases: *province*, *other-position*, *emp*, *prof*, *gov*, *owner*, *offic*, *warehouse*, *utilities*, *office*, *farm*, and *communications*. The

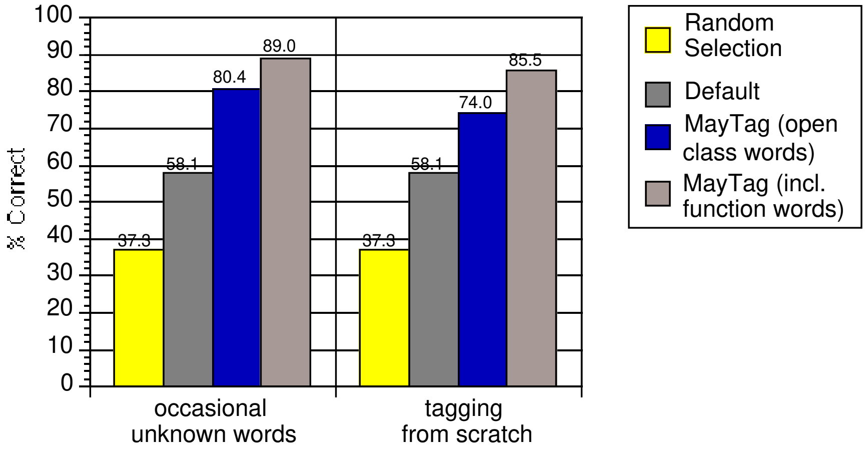

Figure 7.4: Specific Semantic Feature Prediction (% correct).

unknown open-class word represented by each training case assumed one of 18 general semantic feature combinations and one of 35 specific semantic feature combinations.[5] The distribution of these values is shown in Tables 7.5 and 7.6.

As was the case with part-of-speech prediction, a small number of semantic features are used to tag a large percentage of the unknown open-class words in the training sentences. The *jv-entity* and *entity* general semantic features, for example, were used to tag over 50% of the open-class words in the training sentences and the *nil* specific semantic feature was assigned to 58% of them. As shown in Figures 7.5 and 7.6, MayTag attains greater predictive accuracy for a semantic feature as the number of training cases with that feature increases. These graphs show the breakdown of MayTag's performance on individual general and specific semantic features, respectively. Both graphs are based on a 10-fold cross-validation run in which MayTag simulates semantic feature prediction for occasional unknown words using the usual 2056-instance case base.[6] For each semantic feature, s, the graphs show two values. The $*$ indicates the percentage of unknown words that should have been tagged with s and the corresponding x shows MayTag's performance on s. Like the part-of-speech performance analysis, the graphs indicate that MayTag performs best on frequently occurring semantic features. For general semantic feature prediction, the three most frequent tags (i.e., *jv-entity*, *entity*, *nil*) account for 66.3% of the test instances and MayTag attains 88.0% accuracy across these features. For specific semantic feature prediction, two features (i.e., *nil*, *company-name*) account for 74.6% of the instances and MayTag achieves an average of 91.1% correct for these features.

[5]All 14 general semantic features and 30 of the possible 42 specific semantic features were supplied as **gen-sem** and **spec-sem** values for the open-class word defined in each training case. In addition, some words were assigned more than one semantic feature in certain contexts.

[6]The average accuracy for open-class words across the ten runs for was 77.4% for general semantic features and 80.1% for specific semantic features.

Table 7.5: General Semantic Feature Frequencies for Unknown Words (2056 cases).

General Semantic Feature	Frequency
NATIONALITY, JV-ENTITY	1
LOCATION, JV-ENTITY	1
CAPITALIZATION	5
INDUSTRY, JV-ENTITY	6
FACILITY, INDUSTRY	9
MONEY	24
GENERIC-LOC	25
PERSON-POSITION	25
OWNERSHIP-PERCENTAGE	31
HUMAN	41
FACILITY	43
NATIONALITY	65
TIME	95
LOCATION	131
INDUSTRY	191
NIL	322
ENTITY	515
JV-ENTITY	526

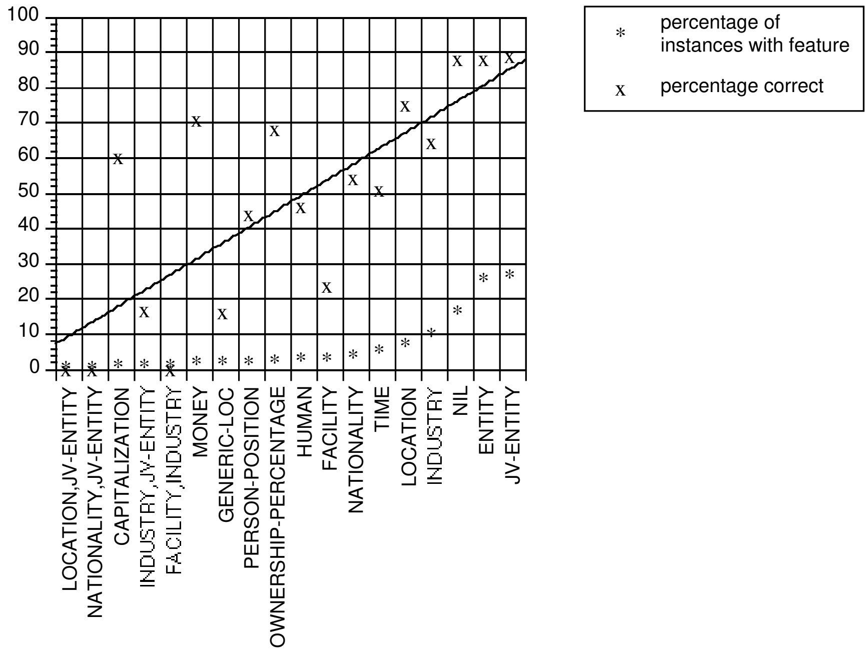

Figure 7.5: Performance on Individual General Semantic Features for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting general semantic feature, g; $*$'s indicate % of instances that should have been assigned g; line is best linear approximation to x's.)

Table 7.6: Specific Semantic Feature Frequencies for Unknown Words (2056 cases).

Specific Semantic Feature	Frequency
PRES	1
COB	1
FACTORY, PRODUCTION	1
SITE, PRODUCTION	1
MINE	1
COUNTRY, GOVERNMENT	1
CEO	2
SITE	2
STORE, SALES	2
OTHER-FACILITY	2
TRANSPORTATION	3
PERSON	3
STORE, SERVICE	5
GOVERNMENT	5
EXEC	5
HUMAN-TITLE	6
SREXEC	7
COMPANY-ALIAS	7
STORE	7
SERVICE	8
RESEARCH	8
CONTINENT	9
SPOKE	9
FACILITY-NAME	11
FINANCE	12
FACTORY	13
SALES	17
OTHER-LOC	22
CITY	25
HUMAN-NAME	28
COUNTRY	75
GENERIC-COMPANY	78
PRODUCTION	146
COMPANY-NAME	338
NIL	1195

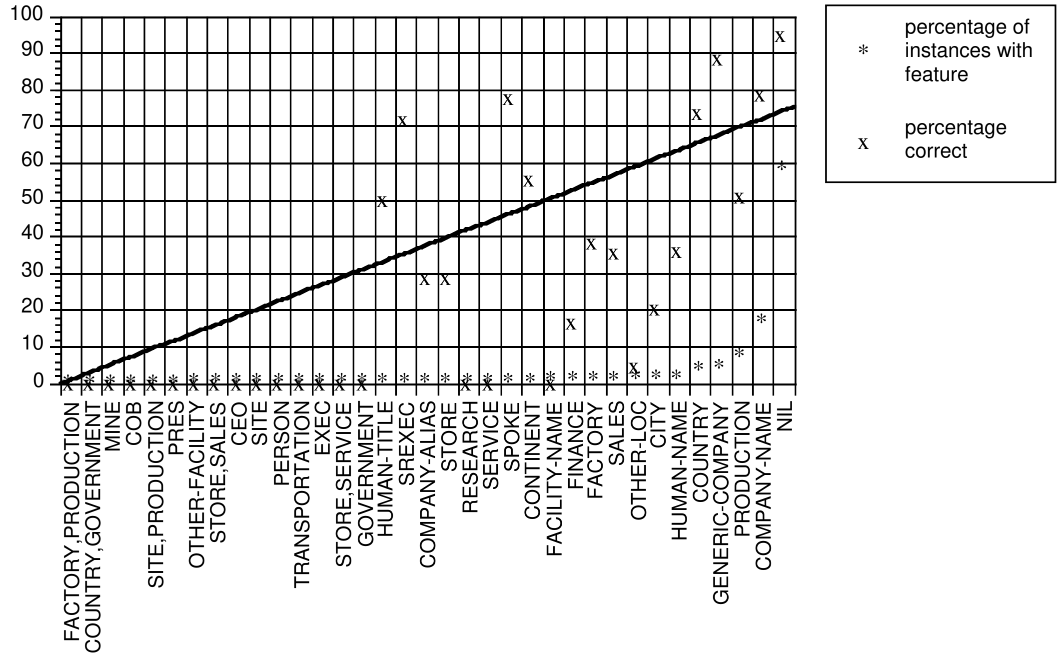

Figure 7.6: Performance on Individual Specific Semantic Features for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting specific semantic feature, s; $*$'s indicate % of instances that should have been assigned s; line is best linear approximation to x's.)

7.2.1 Types of Errors

The error matrix of Table 7.7 displays MayTag's errors for general semantic feature prediction. Of seven errors in tagging *money* words, for example, one was mistakenly given the *jv-entity* feature, five were presumed to be *entity*'s, and one, an *industry*. A portion of the error matrix for specific semantic feature prediction is shown in Table 7.8.[7] Neither table shows errors for words tagged with more than one semantic feature by the human supervisor (for a single occurrence of the word). There were 16 such cases for general semantic feature tagging in the test sentences, 44% of which MayTag correctly assigns at least one of the required features. For example, in one test sentence the word "China" should have been assigned both the *location* and *jv-entity* general semantic features, but MayTag thought that only the *location* feature applied.

It is clear from Table 7.7 that a majority of general semantic feature tagging errors (67%) are due to false hits on the *jv-entity* and *entity* features. A similar situation holds for specific semantic feature tagging, where 77% of the errors are caused by false hits on either the *nil* or *company-name* semantic features. General solutions to the problem of false hits were presented in Section 7.1.2 and would apply here as well.

[7]Only those features used to tag more than nine open-class words in the test sentences are included.

Table 7.7: General Semantic Feature Error Matrix for Occasional-Unknown-Word Tagging.

A	CAPITALIZATION
B	GENERIC-LOC
C	LOCATION
D	HUMAN
E	PERSON-POSITION
F	OWNERSHIP-PERCENTAGE
G	NIL
H	MONEY
I	JV-ENTITY
J	ENTITY
K	FACILITY
L	INDUSTRY
M	NATIONALITY
N	TIME

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A										2				
B			6						1	12		1		1
C							1		11	18		1	2	
D			2				3		14	3				
E			1						8	5				
F			2				1			7				
G			4						15	16		3		2
H									1	5		1		
I			12	1		1	11	1		24		6	1	2
J		1	5		1	3	12	2	21			12	2	5
K			2						8	19		2	1	
L			6				2		20	38	1			
M	1		2			2			9	15		1		
N		1	3			1	11		6	25				

Table 7.8: Specific Semantic Feature Error Matrix for Occasional-Unknown-Word Tagging.

A	SPOKE
B	CONTINENT
C	FACILITY-NAME
D	FINANCE
E	FACTORY
F	PRODUCTION
G	SALES
H	OTHER-LOC
I	CITY
J	HUMAN-NAME
K	COUNTRY
L	GENERIC-COMPANY
M	GOVERNMENT
N	COMPANY-NAME
O	NIL

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A												1			1
B								1							3
C											2	1		2	4
D														1	9
E						1									7
F											1			5	66
G	1					2									7
H											4			2	15
I											7	1		2	10
J														5	13
K			1											2	16
L														3	6
M														1	4
N									1		2	21			47
O				1		13					5	9		32	

7.2.2 Domain-Dependent vs. Domain-Independent Features

The set of semantic features used with the JV corpus can be divided into two major groups — domain-dependent features and domain-independent features (see Table 7.9). Eight general semantic features, for example, represent distinctions that must be made in the domain of business joint ventures but that may not pertain outside the domain. The remaining entries in the general semantic feature hierarchy, however, might be of use in other domains. The domain-independent features of Table 7.9 were, in fact, also part of the semantic feature taxonomy for the terrorism domain [Lehnert *et al.*, 1991].

Table 7.9: Domain-Dependent and Domain-Independent Semantic Features. (General semantic features are shown in UPPER CASE; specific semantic features are shown in lower-case.)

Domain-Dependent Features		**Domain-Independent Features**	
JV-ENTITY		MONEY	
company-name	company-alias	NATIONALITY	
generic-company	government	HUMAN	
person		human-name	human-title
INDUSTRY		TIME	
research	production	LOCATION	
sales	service	country	city
finance		province	continent
CAPITALIZATION		other-loc	
PERSON-POSITION		GENERIC-LOCATION	
ceo	cob	NIL	
pres	offic	nil	
srexec	exec		
owner	gov		
spoke	prof		
emp	other-position		
FACILITY			
communications	site		
factory	farm		
office	mine		
store	transportation		
utilities	warehouse		
facility-name	other-facility		
OWNERSHIP-PERCENTAGE			
ENTITY			

It is useful to evaluate MayTag's performance on domain-dependent vs. domain-independent semantic feature prediction because it is directly related to MayTag's ability to acquire variable-depth knowledge representations.[8] Given that the corpus is domain-specific, one may hypothesize that the texts naturally will provide contexts that make domain-specific distinctions easier to predict. On the other hand, the domain-independent semantic features seem to represent more general semantic categories, which may be easier to predict. The results in Tables 7.10 and 7.11 indicate that MayTag is better at assigning domain-dependent semantic features than assigning domain-independent semantic features.[9] For the smaller case base (2056 cases), 34.2% of the open-class words were given domain-independent semantic features by the human supervisor. In the occasional-unknown-word task, MayTag provides the correct general semantic feature for 71.5% of these cases (see Table 7.10). Words with domain-dependent features, on the other hand, account for 65.3% of the open-class words in the training sentences and MayTag correctly assigns the general semantic feature for these words 80.4% of the time. For specific semantic features, there is a question as to whether the value *nil* should be considered domain-dependent or domain-independent because it can be used to indicate either that no specific semantic feature from the taxonomy applies or that a word is of a type for which no semantic features should be assigned (e.g., a verb). The first use of *nil* seems to be a very general, but domain-dependent one, while the second use of *nil* seems to be a domain-independent one. Excluding *nil*'s, MayTag correctly tags 47.9% of the unknown words that require domain-independent features (see Table 7.11). This rate increases to 63.5% for domain-dependent features. When *nil* is considered both a domain-independent and domain-dependent semantic feature, the results are much closer — 89.1% for domain-independent semantic features and 83.5% for domain-dependent features. Tables 7.10 and 7.11 indicate that these trends continue as the case base grows and hold for the more difficult tagging-from-scratch task.

Table 7.10: Results for Domain-Independent vs. Domain-Dependent General Semantic Features (% correct).

General Semantic Feature Type	**Occasional Unknown Word** 2056 cases	**Occasional Unknown Word** 3060 cases	**Tagging From Scratch** 2056 cases
domain-dependent	80.4	80.7	71.2
domain-independent	71.5	69.3	64.2

[8]These were proposed in Section 4.2 to support variable-depth text processing, which, in turn, is needed for most domain-specific text summarization tasks.

[9]Note that the results do not include cases (two) in which the human supervisor supplied a domain-dependent as well as a domain-independent semantic feature for the unknown word.

Table 7.11: Results for Domain-Independent vs. Domain-Dependent Specific Semantic Features (% correct). (Values in parentheses indicate percentage correct if *nil* included.)

Specific Semantic Feature Type	**Occasional Unknown Word** 2056 cases	**Occasional Unknown Word** 3060 cases	**Tagging From Scratch** 2056 cases
domain-dependent	63.5 (83.5)	70.2 (83.8)	45.2 (77.2)
domain-independent	47.9 (89.1)	51.2 (91.6)	37.9 (89.1)

7.3 Domain-Specific Concept Disambiguation

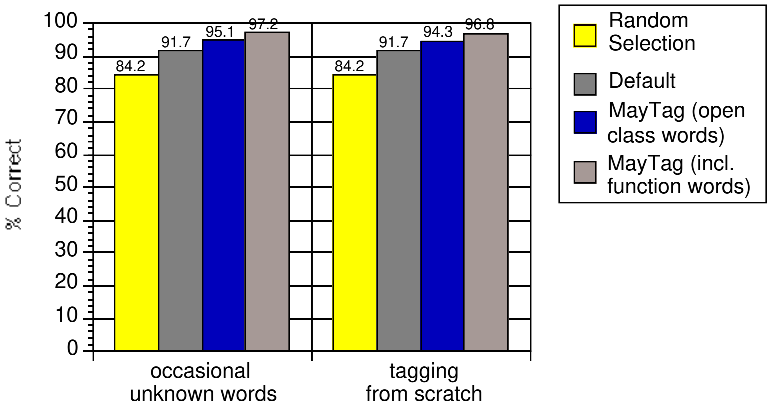

Figure 7.7: Concept Type Prediction (% correct).

Concept activation in CIRCUS was described in Chapter 3. It is a complex task that amounts to selectively triggering a semantic case frame, and requires that MayTag implicitly learn three pieces of information. First, MayTag must decide which *word* should trigger the case frame. Next, it must determine the *type* of case frame to trigger (i.e., the concept type). Finally, it must learn to recognize the context in which the case frame should be activated: in CIRCUS's terminology, MayTag must learn a set of *enabling conditions*. Figure 7.7 summarizes MayTag's ability to predict domain-specific concept activation for both the occasional-unknown-word task and the tagging-from-scratch task using the 2056-case case base. These results were presented in Chapter 5. Possibly the most striking result from this chart is the exceptional performance of the default baseline, a system that always decides that a word activates **no** domain-specific concepts (i.e., activates the *nil* concept). This strategy achieves an accuracy of 91.7% correct across open-class words in the test sets. Although chi-square significance tests indicate that MayTag performs significantly better than this baseline, a closer examination of these results is provided below.

MayTag's concept taxonomy is shown in Table 5.3 of Chapter 5. It represents a subset of the events that are important to recognize in the joint ventures domain. Of the 11 concept types in the taxonomy (12 including *nil*), 9 occur across the open-class words in the training sentences. Only the *industry-sales* and *industry-finance* concepts are not represented. Table 7.12 shows the distribution of concept types across the 2056 open-class words in the training sentences and Figure 7.8 displays MayTag's ability to predict individual concept types.

Table 7.12: Concept Type Frequencies for Unknown Words (2056 cases).

Concept Type	Frequency
INDUSTRY-RESEARCH	4
TOTAL-CAPITALIZATION	9
TIE-UP-RELATIONSHIP-SECONDARY	11
TIE-UP-RELATIONSHIP-JV-ENTITY-ONLY	12
INDUSTRY-SALES	16
INDUSTRY	18
INDUSTRY-PRODUCTION	24
OWNERSHIP-TIE-UP-RELATIONSHIP	48
NIL	1885

As was the case for part-of-speech and semantic feature prediction, the graph indicates that MayTag performs best on frequently occurring concept types. The *nil* concept type that was used in the default baseline, for example, accounts for 91.7% of the test cases and MayTag achieves 98.8% accuracy for this value. If we discard all words associated with the *nil* concept type, however, MayTag only predicts 49% of the concept types of open-class words correctly. In the tagging-from-scratch task, this value drops to 43%. Not surprisingly, false hits on *nil* account for 68% of MayTag's errors in the occasional-unknown-word task and for 75% of MayTag's errors in the tagging-from-scratch task.

7.3.1 Types of Errors

Table 7.13 shows MayTag's concept activation errors in the occasional-unknown-word task. For this task, 37% of the errors that were not false hits on *nil* were near misses in the sense that main event type (e.g., tie-up, industry) was predicted correctly. In one case, for example, MayTag recognized an *industry-product* concept as an *industry-sales* concept. In another case, a *tie-up-relationship* event was recognized as a *tie-up-relationship-secondary*. For the tagging-from-scratch task, 34% of the errors were near misses. If we count near misses as correct, MayTag's accuracy for prediction of non-nil concept activations increases from 49% to 52% correct (occasional-unknown-word task) and from 43% to 49% correct (tagging from scratch).

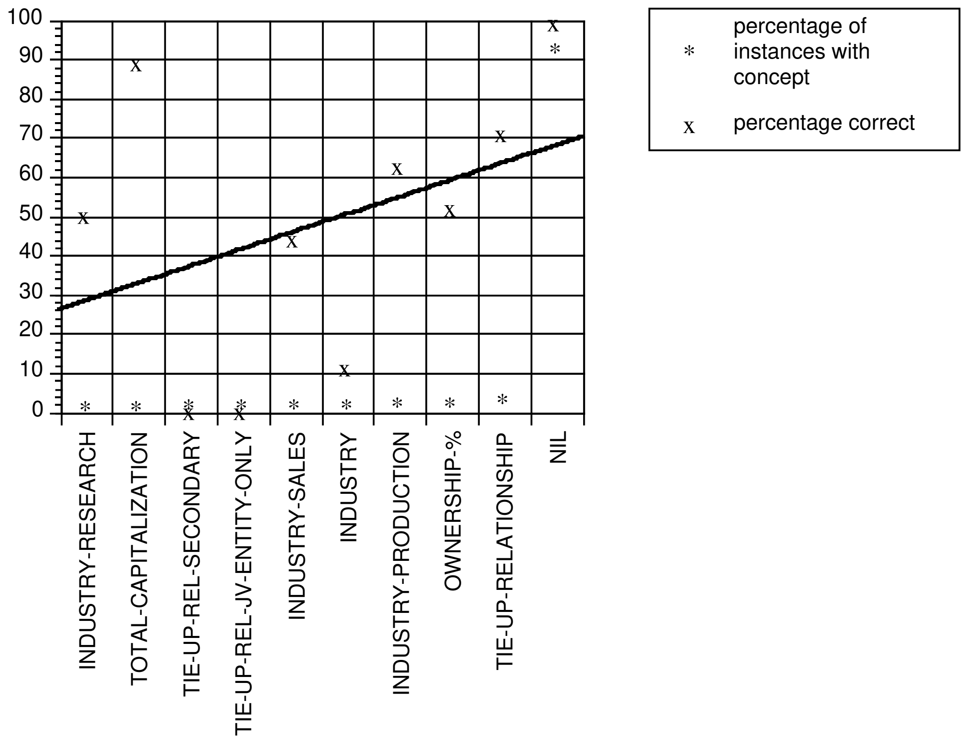

Figure 7.8: Performance on Individual Concept Types for Occasional-Unknown-Word Tagging. (x's indicate % correct for predicting concept type, c; $*$'s indicate % of instances that should have been assigned c; line is best linear approximation to x's.)

As discussed in Section 7.1.2, using a smaller value of k may help reduce the number of false hits on majority class values like the *nil* concept types. Table 7.14 summarizes the effects of changing the value of k on domain-specific concept activation and shows that lowering k does reduce the percentage of false hits. While this seems encouraging, the reduction in the number of false hits on *nil* does not increase MayTag's accuracy in predicting non-nil concept types. To keep the false hit rate low and the accuracy for non-nil concept prediction high, we might use small values of k in conjunction with one or more methods for increasing the representation of minority concept types in the training set (described in Section 7.1.2).

7.3.2 Expanding the Concept Type Taxonomy

Thus far, all concept activation experiments drew from a taxonomy that represented just a subset of the concepts that should be recognized in the joint ventures domain. To process texts from the JV corpus to the degree required in the official MUC and TIPSTER performance evaluations, however, an NLP system requires a more detailed and expanded taxonomy of concept types. The taxonomy used by the UMass/Hughes CIRCUS system for the MUC-5 evaluation contained sixteen concept types. Table 7.15 lists all concept types from this expanded taxonomy that appeared in the 174 training sentences used to

Table 7.13: Concept Type Error Matrix for Occasional-Unknown-Word Tagging.

A	INDUSTRY-RESEARCH
B	INDUSTRY-PRODUCTION
C	TIE-UP-RELATIONSHIP-JV-ENTITY-ONLY
D	NIL
E	INDUSTRY
F	TIE-UP-RELATIONSHIP
G	TIE-UP-RELATIONSHIP-SECONDARY
H	TOTAL-CAPITALIZATION
I	OWNERSHIP-%
J	INDUSTRY-SALES

	A	B	C	D	E	F	G	H	I	J
A				2						
B				8						1
C				4		8				
D			2		3	11		1	5	
E				16						
F			1	12			1			
G				9		2				
H				1						
I				14						
J				9						

Table 7.14: Effect of Smaller Values of k for Prediction of Non-*nil* Concept Types. (Results shown are for open-class words using the 2056-case case base for the occasional-unknown-word task).

Domain-Specific Concept Prediction	$k = 10$	$k = 5$	$k = 1$
% false hits	68	61	36
% non-nil correct	49	49	47
% non-nil correct (incl. near misses)	56	56	58

generate MayTag's larger case base of 3060 cases along with its corresponding concept type from our original taxonomy. The MUC-5 taxonomy includes three entirely new concept types and expands two of the original concepts into six more specialized ones. For the

Table 7.15: Expanded Concept Type Taxonomy.

MUC-5 Concept Types	Corresponding MayTag Concept
jv-entity-person-not-jv	tie-up-relationship
jv-entity-company-not-jv	tie-up-relationship
jv-person	tie-up-relationship
jv-entity-not-jv	tie-up-relationship
jv-entity-company-jv	tie-up-relationship-jv-entity-only
jv-entity-jv	tie-up-relationship-jv-entity-only
industry	industry
industry-production	industry-production
industry-sales	industry-sales
industry-service	industry-service
facility	none
ownership	none
own_percent	ownership-%
revenue	none
nil	nil

MUC-5 performance evaluation, the UMass/Hughes system used the AutoSlog [Riloff, 1993] system to acquire concept activation knowledge for the JV corpus in the form of concept activation rules. For example, consider the following sentence:

IBM recently formed a subsidiary for production of PowerPC chips.

Given this sentence, the CIRCUS parser, a small set of rules, and the fact that the phrase "IBM" denotes a parent company involved in a joint venture, AutoSlog would suggest the following rule:

If the current word is "subsidiary" and the verb is "formed", used in active voice, then activate a JV-ENTITY-COMPANY-NOT-JV concept.

AutoSlog was used to generate concept activation rules like the above for the entire JV corpus. However, because many of AutoSlog's concept activation rules are too general, too specific, or just incorrect, a person first must peruse the proposed rule set and discard bogus entries.

We use the AutoSlog-generated concept activation rules in experiments that investigate MayTag's ability to perform concept activation for the expanded concept taxonomy. During the training phase in these experiments, we use the human-filtered AutoSlog concept activation rules to indicate the domain-specific concept that should be activated for each open-class word rather than asking the human supervisor to supply that information.[10] In the application/test phase, therefore, MayTag is trying to predict when the AutoSlog concept activation rules should fire without explicitly knowing the rules. Table 7.16 shows the results for the occasional-unknown-word task using 10-fold cross validation for a case base of 3060 cases.[11] The overall accuracy for concept prediction declines from 95% correct to 92% correct using the expanded taxonomy, this in spite of the fact that more training is provided. On the other hand, the new task is much more difficult because there are many more non-nil concepts to predict across the test sentences — only 83% of the open-class words are associated with the *nil* concept as compared to 92% in the original experiments. This means that the use of the expanded concept taxonomy and a larger set of training sentences produce a case base that is better balanced than the original and this may at least partially explain the improved performance on words that activate a non-nil concept type (see the last three rows of the table).

The new task is more difficult for another reason: many words activate more than one domain-specific concept, but MayTag receives credit only when it predicts *all* of the expected concepts. The frequency with which different concept type combinations are used across the training set is shown in Table 7.17. The final column in the Table shows that, in general, MayTag performs better as the representation of the concept type in the training sentences increases.

Table 7.16: Prediction of Concept Types from the Expanded Taxonomy (occasional-unknown-word task, k=10).

MayTag Results	**Original Concept Taxonomy** 2056 cases	**Expanded Concept Taxonomy** 3060 cases
% correct	95	92
% *nil* concepts	92	83
% false hits on *nil*	68	61
% non-nil correct	49	63
% non-nil correct incl. near misses	56	68

One problem with the concept activation rules acquired by AutoSlog is their limited notion of the linguistic context that is adequate for activating the concept. In the terrorism domain, for example, whenever "killed" is used as a verb in the passive voice, the concept

[10] In addition, OTB [Lehnert *et al.*, 1993b] was used for part-of-speech tagging.

[11] This is the same case base and experimental design as was used in Section 5.5.2.1 that tested MayTag's ability to predict semantic features for the MUC-5/TIPSTER evaluation. In particular, the current experiments use the same subset of relevant features in its nearest-neighbor calculations as were used for those experiments.

Table 7.17: Concept Type Frequencies for Unknown Words Using the Expanded Concept Taxonomy (3060 cases).

Concept Type	Frequency	% Correct
jv-entity-company-not-jv, facility, industry	1	0
jv-entity-jv, jv-person	1	0
jv-entity-jv, industry	1	0
jv-entity-company-not-jv, industry-production, revenue	1	0
jv-entity-company-jv, jv-entity-jv, own_percent	1	0
jv-entity-company-not-jv, jv-entity-not-jv, own_percent	1	0
industry-service	2	0
jv-entity-company-not-jv, jv-entity-person-not-jv	2	0
jv-entity-company-not-jv, industry, revenue	2	0
jv-entity-company-not-jv, industry-sales, revenue	2	0
jv-entity-company-not-jv, facility, industry-production	3	67
jv-entity-company-not-jv, jv-entity-not-jv, industry, revenue	3	0
jv-entity-not-jv	5	0
revenue	5	40
jv-entity-company-jv, ownership	6	40
jv-person	6	50
industry-sales	6	0
jv-entity-company-not-jv, industry-sales	6	50
jv-entity-company-not-jv, own_percent	7	29
ownership	8	63
jv-entity-not-jv, jv-person	9	100
jv-entity-company-not-jv, industry-production	9	22
facility, industry-production	10	70
jv-entity-company-jv, own_percent	10	63
industry-sales, revenue	11	64
own_percent	11	9
facility	12	58
jv-entity-company-not-jv, industry	15	46
facility, industry	15	73
jv-entity-jv, jv-entity-not-jv	17	94
jv-entity-company-jv, jv-entity-company-not-jv, own_percent	17	88
jv-entity-company-jv jv-entity-jv	24	71
jv-entity-company-jv	28	71
jv-entity-jv	32	34
industry	34	52
jv-entity-company-not-jv	50	53
jv-entity-company-jv, jv-entity-jv, own_percent, revenue	65	98
industry-production	78	68
nil	2545	98

activation rule for TERRORIST-MURDER is satisfied. This means that the TERRORIST-MURDER concept would be activated in the following sentence:

Gun law legislation was killed in Congress by powerful NRA lobbyists.

Because MayTag activates concepts based on a more flexible set of (implicit) enabling conditions, it can recognize these inappropriate contexts more readily. The current experiments, however, do not test MayTag's ability in this regard because no human supervisor was used to discard erroneous AutoSlog concept activations during the training phase.

7.4 Generalization Issues

One question that arises for MayTag is whether the system learns any useful generalizations. For example, after training on the sentences from the joint ventures corpus, does MayTag know that "in" implies "location?" It is difficult to address generalization issues because the case-based approach does not encode any explicit generalizations. However, given that MayTag learns to predict that "locations" follow "in" in certain contexts and "time" phrases follow "in" in other contexts, we would argue that MayTag is learning to make implicit generalizations based on the training cases. There is a further complication in determining the extent of MayTag's generalizations — some features are discarded from the case representation by the decision tree subsystem. If the features that represent "in" in a test sentence are discarded from the case representation, for example, then the case retrieval algorithm will not be able to use "in" as a cue that a location will follow.

7.5 Summary

MayTag provides a data-driven approach to lexical acquisition (see Chapter 5). It should not be surprising, therefore, that its success depends to a large degree on the examples it receives for training and how well they cover the lexical ambiguity resolution concepts to be learned. In general, the preceding sections determined that MayTag's performance on a particular class value (e.g., a particular part of speech, semantic feature, or concept) increases as the number of training cases with that value increases. In particular, if those classes that are most important to recognize for the natural language processing task are well-represented in the case base, then MayTag's approach will be successful for that NLP task. In the business joint ventures domain, for example, it is very important to recognize company names and, at the very least, to distinguish them from products and people. Since the training cases for the JV corpus provide adequate representation of these classes, MayTag's approach is sufficient. If it were more important to recognize facilities, however, our machine learning approach to lexical disambiguation will fail. Happily, it is usually the case that classes that are important for a domain or task occur frequently in the corpus, and, hence, occur frequently in the case base.

One major problem in MayTag is that cases associated with majority classes overwhelm the case base and make it difficult for the system to perform well on minority classes.

Section 7.1.2 outlined a number of modifications to the case retrieval algorithm (and to the training phase) that may help to solve this problem. In addition, MayTag could also be improved if the training and/or case retrieval algorithm were tailored to reduce specific errors that are especially detrimental to system performance (e.g., mistaking verbs for non-verbs during part-of-speech disambiguation). These modifications, however, require expert domain knowledge that is often difficult, expensive, and time-consuming to obtain.

CHAPTER 8

FINDING ANTECEDENTS OF RELATIVE PRONOUNS

This chapter demonstrates Kenmore's ability to acquire heuristics for one class of structural disambiguation problem — finding the antecedents of relative pronouns.[1] We instantiate Kenmore in a system called **WHirlpool** that uses a case-based reasoning algorithm to predict the antecedent of a relative pronoun like "who" or "whom" given a description of the clause that precedes it. During a human-supervised training phase, WHirlpool creates a case base of context-sensitive heuristics for relative pronoun disambiguation. Because context is encoded as part of each case, no explicit disambiguation rules need to be formulated. After training, WHirlpool draws from the case base to find the antecedent of a relative pronoun without any human intervention. Our experiments show that the heuristics learned within the Kenmore framework equal the performance of a set of hand-coded rules in terms of prediction accuracy.

The chapter first introduces the problem (Section 8.1) and then presents WHirlpool, the instantiation of Kenmore that learns to find relative pronoun antecedents (Section 8.2). We describe WHirlpool's training phase, application phase, and initial case representation in Sections 8.3-8.4, and then evaluate WHirlpool using this initial case representation in Section 8.5. Next, we describe and evaluate WHirlpool's approach to improving a baseline case representation, an approach in which cognitive biases are explicitly incorporated into an existing case representation (Section 8.6). Experiments indicate that WHirlpool performs as well as a set of hand-coded heuristics for relative pronoun disambiguation and performs significantly better than a default rule that always chooses the most recent constituent as the antecedent. In addition, we find that WHirlpool's prediction accuracy increases monotonically as each of three cognitive biases is added to the baseline case representation.

8.1 The Problem

Relative clauses consistently create problems for language processing systems. Consider, for example, the sentence in Figure 8.1. Its semantic interpretation should include

---

[1] Some of the material from this chapter appeared originally in Cardie [1992a, 1992b, 1992c].

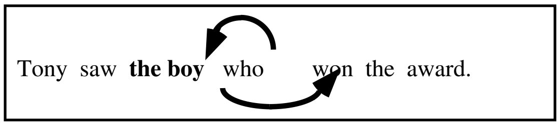

Figure 8.1: Understanding Relative Clauses.

the fact that "the boy" is the actor of "won" even though the phrase does not appear in the embedded clause. Making this inference depends on the accurate resolution of two ambiguities, each of which must be performed over a potentially unbounded distance. The system has to:

1. find the antecedent of the relative pronoun "who," and

2. determine the antecedent's implicit position in the embedded clause — this is also called finding the *gap*.

This chapter focuses only on the first problem: locating the antecedent of the relative pronoun.[2]

Although relative pronoun disambiguation seems a simple enough task, there are many factors that make it difficult:

S1. We wondered who stole the watch.
S2. Tony saw *the boy* who won the award.
S3. *The boy* who gave me the book had red hair.
S4. Tony ate dinner with *the men* from Detroit who sold computers.
S5. I spoke to *the woman* with the black shirt and green hat over in the far corner of the room who wanted a second interview.
S6. I'd like to thank *Jim, Terry, and Shawn*, who provided the desserts.
S7. I'd like to thank *our sponsors, GE and NSF*, who provide financial support.
S8. We talked with *the woman and the man* who were/was dancing.
S9. We talked with *the woman and the man* who danced.
S10. *The woman* from Philadelphia who played soccer was my sister.
S11. The awards for *the children* who pass the test are in the drawer.

Figure 8.2: Relative Pronoun Antecedents.

Initially there is the problem of distinguishing uses of wh-words that are relative pronouns from uses that are not. Making this distinction will be important because uses of "who" that are not relative pronouns (e.g., interrogatives like S1 of Figure 8.2) have no antecedent.

[2]Locating the gap is an equally difficult problem because the gap may appear in a variety of positions in the embedded clause: the subject, direct object, indirect object, or object of a preposition. Chapter 3.1.3 describes the solution to the gap-finding problem employed by the CIRCUS parser.

The head of the antecedent of a relative pronoun does not appear in a consistent position or syntactic constituent. In both S2 and S3, for example, the antecedent is "the boy." In S2, however, "the boy" is the direct object of the preceding clause, while in S3 it appears as the subject of the preceding clause. On the other hand, the head of the antecedent is the phrase that immediately precedes "who" in both cases. S4, however, shows that this is not always the case. The antecedent head may be very distant from its coreferent wh-word[3] (e.g., S5).

The antecedent is not always a single noun or compound noun. In S6, for example, the antecedent of "who" is a conjunction of three phrases and in S7, either "our sponsors" or its appositive "GE and NSF" is a semantically valid antecedent.

Disambiguation of the relative pronoun may depend on information in the embedded clause. In S8, for example, the antecedent of "who" is either "the man" or "the woman and the man," depending on the number of the embedded clause verb.

Sometimes, the antecedent is truly ambiguous. For sentences like S9, the real antecedent depends on the surrounding context.

Locating the antecedent requires the assimilation of both syntactic and semantic knowledge. The syntactic structure of the clause preceding "who" in sentences S10 and S11, for example, is identical (a noun phrase followed by a prepositional phrase). The antecedent in each case is different, however. In S10, the antecedent is the subject, "the woman;" in S11, the antecedent is the prepositional phrase modifier, "the children."

As a direct result of these difficulties, NLP system builders have found the manual coding of rules that find relative pronoun antecedents to be very hard. In addition, the resulting heuristics are prone to errors of omission and may not generalize to new contexts. For example, the UMass/MUC-3 CIRCUS system[4] [Lehnert *et al.*, 1991] began with nineteen rules for finding the antecedents of relative pronouns. These rules included both structural and semantic knowledge and were based on approximately 50 instances of relative pronouns. As counter-examples were identified, new rules were added (approximately ten) and existing rules changed. Over time, however, we became increasingly reluctant to modify the rule set because the global effects of local rule changes were difficult to measure.[5] In addition, most rules were based on a class of sentences that the UMass/MUC-3 system had found to contain important information. As a result, the hand-coded rules tended to work well for relative pronoun disambiguation in sentences

[3]Relative pronouns like who, whom, which, that, where, etc. are often referred to as wh-words.

[4]UMass/MUC-3 is the version of the CIRCUS parser (see Chapter 3) used for the MUC-3 performance evaluation (see Chapter 4).

[5]One possibility not considered in this work is to use a mixed-paradigm approach to finding antecedents of relative pronouns. In the same way that ANAPRON [Golding, 1991, Golding and Rosenbloom, 1991] and CABARET [Rissland and Skalak, 1991] combine case-based and rule-based reasoning, we might use handcrafted rules to represent a small set of safe, clear-cut relative pronoun disambiguation heuristics and then rely on cases to represent the exceptions.

of this class (93% correct in one set of 50 texts), but did not generalize to sentences outside of this class (78% correct for the same set of 50 texts). The rules do play a large role in processing texts from the MUC-3 corpus, however — approximately 25% of the sentences contain at least one relative pronoun, and "who" occurs in approximately one out of ten sentences. Given that the system as a whole performed very well on texts from this corpus in the MUC-3 evaluation, we believe that the hand-coded heuristics were fairly reliable.

8.1.1 Current Approaches to Relative Pronoun Disambiguation

Although descriptions of natural language processing systems do not usually include the algorithms used to find relative pronoun antecedents, current high-coverage parsers seem to employ one of three approaches for relative pronoun disambiguation. Systems that use a formal syntactic grammar often directly encode information for relative pronoun disambiguation in the grammar. Alternatively, a syntactic filter is applied to the parse tree and any noun phrases for which coreference with the relative pronoun is syntactically legal are passed to a semantic component, which determines the antecedent using inference or preference rules (see, for example, Correa [1988], Hobbs [1986], Ingria and Stallard [1989], Lappin and McCord [1990]). The third approach employs hand-coded disambiguation heuristics that rely mainly on semantic knowledge but also include syntactic constraints (e.g., CIRCUS). However, there are problems with all three approaches:

1. the grammar must be designed to find relative pronoun antecedents for all possible syntactic contexts;
2. the grammar and/or inference rules require tuning for new corpora;
3. in most cases, the approaches unreasonably assume a completely correct parse of the clause preceding the relative pronoun.

In the remainder of the chapter, we describe WHirlpool, an instantiation of the Kenmore knowledge acquisition framework that learns heuristics for relative pronoun disambiguation and avoids the above problems.

8.2 Instantiating the Kenmore Framework for WHirlpool

To learn heuristics for relative pronoun disambiguation we instantiate the three components of the Kenmore architecture as shown in Figure 8.3. The Latin American Terrorism corpus was developed for the MUC-3 and MUC-4 performance evaluations and was described in Chapter 4.3.1. The CIRCUS conceptual sentence analyzer [Lehnert, 1990] was described in Chapter 3. It takes as input a single sentence and produces as output a semantic case frame representation of the meaning of the sentence. As was the case for MayTag (Chapter 5), the learning algorithm employed by WHirlpool is a case-based reasoning (CBR) algorithm.

As an instantiation of Kenmore, WHirlpool's method for relative pronoun disambiguation closely resembles MayTag's approach to acquiring lexical knowledge:

Corpus:	Latin American Terrorism
Sentence Analyzer:	CIRCUS
Inductive Learning Algorithm:	CBR (1-nearest neighbor)

Figure 8.3: Instantiating Kenmore for WHirlpool.

1. In the training phase, the sentence analyzer and a human supervisor together create a case base of relative pronoun disambiguation cases for a randomly selected set of sentences from the corpus.

2. After training, for each new occurrence of a wh-word, WHirlpool retrieves the most similar case(s) from the case base using a representation of the context of the wh-word as the problem case.

3. Finally, WHirlpool uses the antecedent information in the retrieved case(s) to find the antecedent for the wh-word in the problem case.

One difference between WHirlpool and MayTag is the approach each uses to automate the design of an adequate case representation. As described in Chapter 6, MayTag relied on a decision tree approach to discard the irrelevant features in a baseline case representation. The modified case representation was shown to perform better than a baseline representation that included all features and better than a representation in which expert knowledge was used to discard irrelevant features. We expected irrelevant features to be only part of the feature set selection problem in the relative pronoun disambiguation task. As a result, WHirlpool employs a very different approach for feature set selection that is based on evidence from psycholinguistic studies of sentence processing. It augments a baseline case representation by explicitly encoding cognitive biases into the representation. The possibility of using both the decision tree and cognitive bias approaches to improving a baseline case representation is discussed in Section 9.5.3.

Indexing:	none, flat case base
Similarity Metric:	weighted 1-nearest neighbor
Case Selection:	none (only one case retrieved)
Solution Policy:	weak adaptation of retrieved solution
Automated Improvements to	
Baseline Case Rep:	cognitive bias approach

Figure 8.4: Components of WHirlpool's Case-Based Inductive Learning Algorithm.

For reference, the components of WHirlpool's case-based inductive learning algorithm are summarized in Figure 8.4. In the sections that follow, we describe and evaluate

WHirlpool's training and application phases in detail using a baseline case representation (Sections 8.3-8.5). The final section of the chapter presents and evaluates WHirlpool's cognitive bias approach to improving the baseline case representation (Section 8.6).

8.3 WHirlpool's Training Phase

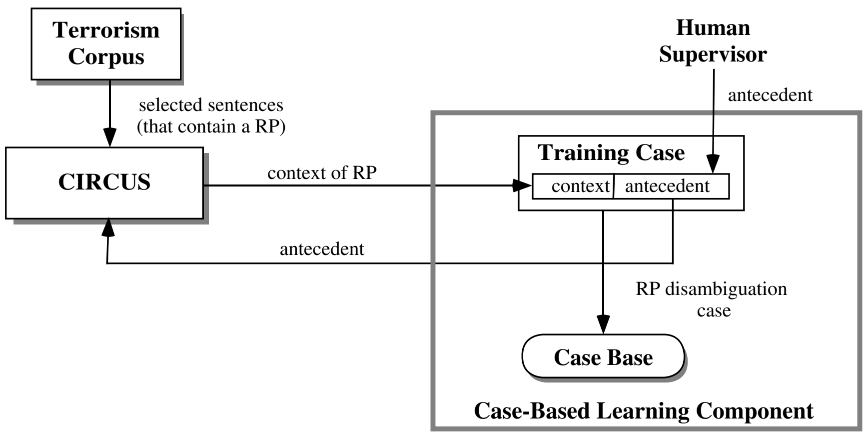

Figure 8.5: WHirlpool Training Phase.

The goal of WHirlpool's training phase (Figure 8.5) is to create a case memory of relative pronoun disambiguation decisions. Our initial efforts have concentrated on the resolution of "who" because the hand-coded heuristics for that relative pronoun were the most complicated and difficult to modify. We expect that the resolution of other wh-words will fall out without difficulty. To create the case base, WHirlpool first randomly selects a set of sentences that contain wh-words[6] from the terrorism corpus and presents them to CIRCUS for analysis. CIRCUS processes each sentence and, with a human supervisor, creates a case as each wh-word is encountered. As described in Chapter 1, cases have two parts. The *context* portion describes the context in which the wh-word occurred and is a representation of the current state of the CIRCUS parser. It is supplied automatically by CIRCUS. The second part of the case, called the *solution* part in Chapter 1, encodes the *antecedent* of the wh-word for the current example. It is supplied by a person during training using a menu-driven interface. The human supervisor simply chooses the phrase or phrase combination from the current sentence that represents the antecedent. WHirlpool encodes the user's choice into the solution part of the case and then stores the entire case in the case base. Finally, WHirlpool sends the antecedent back to CIRCUS so that CIRCUS can update its state information and continue processing the training sentence. At the end of training, WHirlpool will have created one case for each wh-word that occurred in the training sentences.

[6]Not all of the uses of the wh-words in those sentences may be relative pronouns. Some will be interrogatives, for example, but WHirlpool will be responsible for noting the distinction.

8.3.1 Case Representation

As described above, each WHirlpool case represents the disambiguation decision for a single occurrence of a relative pronoun. In theory, all knowledge acquisition systems designed within the Kenmore framework should employ the same case representation. Currently, however, the case representations for MayTag and WHirlpool differ in a few respects that are noted below. Chapter 9 re-examines the role of using a single ambiguity resolution case representation in the Kenmore framework.

Very generally, WHirlpool cases consist of a set of attribute-value pairs, and include any information about the state of the CIRCUS parser that may help determine the antecedent of the current relative pronoun. Like MayTag, WHirlpool begins with a baseline case representation and then uses an automated approach to tune it for the current acquisition task. The context portion of WHirlpool's baseline case representation contains two types of features: one or more **constituent** attribute-value pairs, and two **most-recent** features. The solution portion of the case contains a single **antecedent** attribute-value pair. The paragraphs below briefly describe each attribute type in the baseline representation using the cases in Figure 8.6 as examples.

S1: [The judge] [of the 76th court] [,] who

constituents: *(s human) (s-pp1 physical-target)*
most-recent: *(mr-phrase physical-target) (mr-syn-type comma)*
antecedent: *(antecedent ((s)))*

S2: [The police] [detained] [Juan Bautista] [and] [Rogoberto Matute] [,] who ...

constituents: *(s human) (v t) (do proper-name) (do-np1 proper-name)*
most-recent: *(mr-phrase proper-name) (mr-syn-type comma)*
antecedent: *(antecedent ((do-np1 mr-phrase)))*

S3: [Her DAS bodyguard] [,] [Dagoberto Rodriquez] [,] who...

constituents: *(s human) (s-np1 proper-name)*
most-recent: *(mr-phrase proper-name) (mr-syn-type comma)*
antecedent: *(antecedent ((mr-phrase) (s mr-phrase) (s)))*

Figure 8.6: Relative Pronoun Disambiguation Training Cases.

WHirlpool cases include one **constituent** attribute-value pair for each phrase in the current clause.[7] The attribute describes the *syntactic class* and *position* of the phrase as

[7]Note that we are using the term "constituent" rather loosely to refer to all noun phrases, prepositional phrases, and verb phrases **prior to any attachment decisions**.

it was encountered by CIRCUS. In S1 of Figure 8.6, for example, "of the 76th court" is represented with the attribute **s-pp1** because it is a prepositional phrase that immediately follows the subject. In S2, "Juan Bautista" and "Rogoberto Matute" are represented with the attributes **do** and **do-np1**, respectively, because CIRCUS recognizes them as the direct object and the noun phrase that directly follows the direct object. (The true direct object of the sentence is the conjoined noun phrase, but CIRCUS does not make these attachment decisions.) The subject and verb constituents (e.g., "the judge" in S1 and "detained" in S2) retain their traditional **s** and **v** labels, however — no positional information is included for those attributes.

The value associated with a **constituent** attribute is simply the semantic feature that best describes the phrase's head noun. Like the semantic feature taxonomy used in the joint ventures domain (Section 5.3.1), the taxonomy used in the terrorism domain is a multi-level hierarchy of general and specific semantic features. However, only values at the top level of the hierarchy (i.e., the general semantic features) were used for this application because the handcrafted heuristics for relative pronoun disambiguation accessed the semantic features at this level. The top level of the hierarchy contains the following semantic features: *human*, *proper-name*, *location*, *entity*, *physical target*, *organization (org)*, and *weapon*. The subject constituents in Figure 8.6 receive the value *human*, for example, because their head nouns are marked with the "human" semantic feature in the lexicon. Because CIRCUS does not associate semantic features with verbs, cases note only the presence of verbs using the value *t*.

In addition, every case includes two **most-recent** features that represent the phrase last recognized by CIRCUS. The **mr-syn-type** attribute-value pair specifies the syntactic class of the most recent phrase and the **mr-phrase** attribute represents its semantic class. The only exception to this is when the relative pronoun is preceded by an item of punctuation. In these cases the value of **mr-syn-type** is simply the item of punctuation. In all sentences of Figure 8.6, for example, the value of **mr-syn-type** was *comma*. The value of the **mr-phrase** feature, however, varies across the sentences. In S2, the phrase recognized just before "who" was "Rogoberto Matute," resulting in a **mr-phrase** value of *proper-name*. In S1, "of the 76th court" is the most recent phrase and a *physical-target*.

WHirlpool's context features intersect with the set of "global context features" used in the MayTag lexical acquisition system. WHirlpool's **most-recent** features correspond to the **last-constituent** features in MayTag, for example. However, WHirlpool includes an attribute-value pair for every phrase recognized in the current clause, while MayTag's case representation includes global context features only for the major constituents of the current clause. In addition, WHirlpool cases include none of the "concept activation" features, "specific semantic feature" attributes, or "local context features" used in MayTag's case representation.

As mentioned above, the solution portion of a WHirlpool case contains a single **antecedent** attribute-value pair. This feature encodes information regarding the position of the relative pronoun antecedent. For CIRCUS, the antecedent is the head of the phrase that the relative pronoun refers to — without any post-modifier phrases. Therefore, the value of this attribute is a list of the constituent attributes (and/or the mr-phrase attribute) that represent the location of the antecedent head. In S1, for example, the antecedent of "who" is "the judge." Because this phrase occurs in the subject position, the value of the antecedent attribute is *(s)*. A value of *(none)* is used if no antecedent is required for this use of "who" or if none can be determined.

Sometimes, however, the antecedent is actually a conjunction of constituents. In these cases, we represent the antecedent as a list of the constituent attributes associated with each element of the conjunction. Look, for example, at sentence S2. Because "who" refers to "Juan Bautista" and "Rogoberto Matute," the antecedent can be described as *(do mr-phrase)* or *(do do-np1)*. Although the lists represent equivalent surface forms, we choose the more general *(do mr-phrase)*.[8] S3 shows yet another variation of the antecedent attribute-value pair. In this example, an appositive creates three semantically equivalent antecedent values, all of which become part of the antecedent feature:

"Dagoberto Rodriguez"	*(mr-phrase)*
"her DAS bodyguard"	*(s)*
"her DAS bodyguard, Dagoberto Rodriguez"	*(s mr-phrase)*

8.3.2 Case Base Construction

Using the case representation described above, WHirlpool creates a case base of relative pronoun disambiguation cases from a small set of sentences in the terrorism corpus. Each training sentence is presented to CIRCUS, which processes the sentence and creates a case each time a wh-word is encountered. The CIRCUS parser automatically creates all **constituent** and **most-recent** features of the training cases as a side effect of syntactic analysis. As noted above, CIRCUS recognizes only low-level constituents like noun phrases, prepositional phrases, and verb phrases, but makes no initial attachment decisions. This makes construction of the cases easier for CIRCUS, but transfers responsibility for many difficult attachment decisions to the case-based reasoning component. During WHirlpool's application phase, that component must recognize all phrases that comprise a conjunction of antecedents and must specify at least one of the semantically valid antecedents in the case of appositives.

Sometimes, however, postponing an attachment decision until after the relative pronoun antecedent has been located makes the decision easier. Consider the prepositional phrase attachment decision for "in the restaurant" in the sentence, "I talked with the men in the restaurant." Depending on the context, "in the restaurant" modifies either "talked" or "the men."[9] If we know that "the men" is the antecedent of a relative pronoun, however

[8]This form is more general because it represents both *(do do-np1)* and *(do do-pp1)*.

[9]In yet another reading, "in the restaurant" may modify the entire clause, i.e., as if the sentence had been "In the restaurant, I talked with the men."

(e.g., "I talked with the men in the restaurant, who offered me the job"), it is probably the case that "in the restaurant" modifies "the men."

It should be noted that only specification of the antecedent attribute-value pair requires human intervention via a menu-driven interface that displays the antecedent options. The user chooses the phrase or phrases that comprise the antecedent and this information is encoded by WHirlpool in the **antecedent** feature. As each training case becomes available, it is stored in the case base of relative pronoun disambiguation decisions. In addition, the antecedent is passed back to CIRCUS so that the appropriate syntactic buffers can be instantiated with the antecedent and processing of the embedded clause can begin. After training, WHirlpool will have produced one case for every occurrence of a wh-word in the training sentences.

8.4 WHirlpool's Application Phase

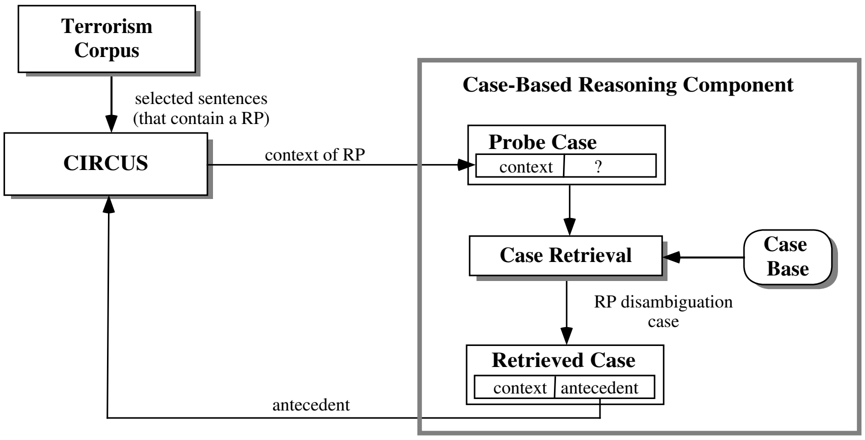

Figure 8.7: WHirlpool Application Phase.

Once the case base has been constructed, WHirlpool can use it to predict the antecedent of a wh-word in new contexts (Figure 8.7). When WHirlpool encounters a wh-word in application mode, it first creates a problem case. The problem case is just a training case that is missing the **antecedent** attribute-value pair. During training, the human supervisor specified the value of this feature, but during the application phase, the case retrieval algorithm will retrieve the training case that is most similar to the problem case and use its antecedent feature to select the antecedent for the novel case. (The specific algorithm used to find the best training case will be described in the next section.) Consider sentence S1 of Figure 8.8, for example. In response to the problem case for this sentence, WHirlpool retrieved a case that specifies the direct object (*do*) as the location of the antecedent. Therefore, WHirlpool chooses the current contents of the direct object — "the hardliners" — as the antecedent of "who" in S1.

Sometimes, however, the single retrieved case lists more than one option as the antecedent. In these cases, we rely on the following simple heuristics to choose an antecedent:

1. Choose the first option whose constituents are all present in the problem case.

2. Otherwise, choose the first option that contains at least one constituent present in the problem case and ignore those constituents in the retrieved antecedent that are missing from the problem case.

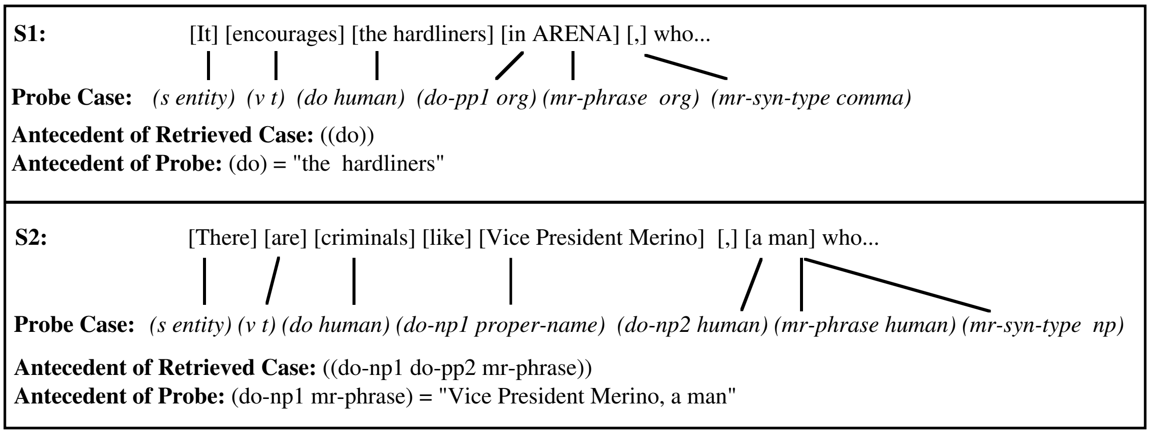

Figure 8.8: WHirlpool Case Retrieval.

From a case-based reasoning perspective, this post-processing of the retrieved antecedent could be considered a very weak and purely syntactic form of *case adaptation*. In general, WHirlpool tries to choose an antecedent that is consistent with the context of the problem case (heuristic 1) or to generalize the retrieved antecedent so that it is applicable in the current context (heuristic 2). S1 of Figure 8.8 illustrates the first case adaptation filter. In S2, however, the retrieved case specifies an antecedent from three constituents, only two of which are actually represented in the problem case. Therefore, we ignore the missing constituent *do-pp2* and look to just *do-np1* and *mr-phrase* for the antecedent.

8.4.1 Case Retrieval in WHirlpool

As of yet, we have not described the specific case retrieval algorithm used in WHirlpool. It is basically the same as MayTag's nearest-neighbor approach to case retrieval:

1. Compare the problem case to each case in the case base, counting the number of context features that match (i.e., match = 1, mismatch = 0).

2. Return the highest scoring case as well as any ties.

3. Let the retrieved cases vote on the value for the antecedent of the problem case.

As described, WHirlpool's case retrieval algorithm is a simple 1-nearest neighbor (1-nn) algorithm. Unfortunately, there is a problem with the algorithm that didn't arise for MayTag. All MayTag cases have the same number of features, but the same is not true for cases in WHirlpool. Although WHirlpool cases always have two **most-recent** features, there is no guarantee that two cases will have any additional attributes in common. The problem cases for S1 and S2 in Figure 8.8, for example, have three additional attributes in common (i.e., both have **s**, **v**, and **do** constituent attributes). A sentence like the following, however, produces a problem case that has no overlapping constituent attributes with either S1 or S2:

Sentence: In most respects, the man who...
Problem case: *(pp1 entity) (np2 human) (mr-phrase human) (mr-syn-type np)*

The question for the similarity metric is how to handle attributes that appear in the problem case but are missing from a training case and vice versa. If the case retrieval algorithm determines similarity by simply counting the number of attribute-value pairs in a training case that match those in the problem case, then two training cases that match the same number of problem case features will achieve the same similarity score regardless of the percentage of training case features correctly matched. Consider the following scenario:

Training case 1:	((s attack) (s-pp1 human) (v t) (do organization) (mr-phrase human) (mr-syn-type comma))
Training case 2:	((s attack) (s-pp1 human) (mr-phrase human) (mr-syn-type comma))
Problem case:	((s attack) (s-pp1 human) (mr-phrase human) (mr-syn-type comma))

Given the problem case, the case retrieval algorithm described above assigns the same similarity score to both training cases even though training case 2 is an exact match for the problem.

The case retrieval algorithm employed by WHirlpool is actually a modified version of the 1-nn algorithm that systematically handles this problem. First, we describe all cases in terms of a normalized set of features:

1. Derive a *normalized feature set* by keeping track of every attribute that occurs in the training cases.

2. Augment the training cases and problem case to include every feature of the normalized feature set, filling in a *nil* value if the feature does not pertain to the particular case.

This means that WHirlpool cases, like MayTag cases, now will always have the same number of features. However, most of the features in a normalized case will have *nil* values because they will not apply to the current context. To ensure that the 1-nn case retrieval algorithm focuses on features that are present rather than missing from the problem case, we modify the original 1-nn case retrieval algorithm to award full credit for matches on features present in the problem cases and to allow partial credit for matches on missing features. This is done by associating with each feature a weight that indicates the importance of the feature in determining case similarity and by using a *weighted* 1-nn case retrieval algorithm:

1. **Give a problem case, first set the weight associated with each feature in the normalized feature set.** For each feature, f, in the normalized feature set, N, set the weight associated with f, w_f:

   - w_f = 0.2 if f is missing from the (unnnormalized) problem case,[10]
   - w_f = 1 otherwise.

2. **Compare the (normalized) problem case to all cases in the case base.** Compare the problem case, P, to each training case, T, in the case base and calculate, for each pair:

$$\sum_{i=1}^{|N|} w_{N_i} * match(P_{N_i}, T_{N_i})$$

   where N_i is the ith feature in N, P_{N_i} is the value of N_i in the problem case, T_{N_i} is the value of N_i the training case, and $match(a, b)$ is a function that returns 1 if a and b are equal and 0 otherwise.

3. **Return the case with the highest score as well as all ties.**

4. **Let the retrieved cases vote on the value of the antecedent.**

8.5 Evaluation of WHirlpool Using the Baseline Case Representation

To evaluate WHirlpool and its weighted 1-nn case retrieval algorithm, we use a 10-fold cross validation testing scheme. Specifically, WHirlpool first generates cases for all 241 examples of "who" from three sets of 50 texts from the terrorism corpus. Next, the 241 examples are randomly partitioned into ten non-overlapping segments of 24 cases. (One case appears in none of the ten segments.) In each of ten runs, we reserve the cases in one segment for testing and store the remaining 217 cases in the case base. For each test case, we invoke the weighted 1-nn case retrieval algorithm to predict the antecedent. Results are averaged across the ten runs.

The results using the weighted 1-nn case retrieval algorithm and the baseline case representation are shown in Table 8.1. WHirlpool's performance is compared with that of the hand-coded heuristics of the UMass/MUC-3 system and a default strategy that simply chooses the most recent phrase as the antecedent. The hand-coded heuristics perform significantly better (at the 95% level) than the default rule and WHirlpool's performance falls somewhere between the two. Chi-square significance tests indicate that WHirlpool's prediction accuracy does not differ from either the default rule or the hand-coded heuristics.

[10]Other values for the missing features weight were tested as well.

Table 8.1: WHirlpool Results (% correct).

WHirlpool	Default Rule	Hand-Coded Heuristics
76.2	74.3	80.5

8.6 Using Cognitive Biases to Improve a Baseline Case Representation

The above results were posted using WHirlpool's baseline case representation described in Section 8.3.1 and the weighted 1-nn case retrieval algorithm described in Section 8.4.1. This section describes WHirlpool's automated approach to improving a baseline case representation. It shows that explicitly encoding cognitive biases into a case representation can improve the performance of the case-based reasoning algorithm. Like MayTag's decision tree approach to feature set selection, the cognitive bias approach to improving a baseline case representation is able to discard irrelevant features from a representation. This is accomplished by reducing the weight associated with a feature, possibly to zero. The cognitive bias approach, however, entails a number of additional manipulations to the original case representation including increasing the importance of an attribute and adding new features to the feature set. Because the cognitive bias and decision tree approaches to improving a case representation draw from very different sources of bias (i.e., psychological bias vs. information theoretic bias), it may prove advantageous to combine the approaches. This possibility is addressed in Chapter 9.

In the experiments below, we modify WHirlpool's baseline case representation in response to three cognitive biases:

1. the tendency to rely on the most recent information,
2. restricted memory limitations, and
3. the heightened accessibility of the subject of a sentence.

Using the weighted 1-nn case retrieval algorithm, we compare each of the modified case representations to the baseline and measure the effects of each bias on relative pronoun antecedent prediction. Our experiments show that the overall performance of the case-based reasoning algorithm improves as each of the cognitive biases and limitations is incorporated into the case representation. Unless otherwise noted, all results shown will be 10-fold cross validation averages. In particular, we emphasize that the same ten training and test set combinations as the baseline experiments of Section 8.5 will be used in all subsequent experiments. This procedure ensures that differences in performance are not attributable to the random partitions chosen for the test set.

8.6.1 *Incorporating the Recency Bias*

In processing language, people consistently show a bias towards the use of the most recent information (e.g., Frazier and Fodor [1978], Gibson [1990], Kimball [1973], Nicol [1988]). In particular, Cuetos and Mitchell [1988], Frazier and Fodor [1978], and others have investigated the importance of recency in finding the antecedents of relative pronouns. They found that there is a preference for choosing the most recent noun phrase in sentences of the form NP V NP OF-PP, with ambiguous relative pronoun antecedents, e.g.:

The journalist interviewed the daughter of the colonel who had had the accident.

In addition, Gibson *et al.* [1993] looked at phrases of the form: NP1 PREP NP2 OF NP3 RELATIVE-CLAUSE. E.g.,

...the lamps near the paintings of *the house* that was damaged in the flood.
...the lamps near *the painting* of the houses that was damaged in the flood.
...*the lamp* near the paintings of the houses that was damaged in the flood.

He found that the most recent noun phrase (NP3) was initially preferred as the antecedent. People had the greatest difficulty when the middle noun phrase (NP2) was the antecedent of the relative pronoun although this difference was not significantly greater than when NP1 was the antecedent. Finally, recognizing antecedents in the NP2 and NP1 positions were significantly harder than recognizing the most recent noun phrase as the antecedent.

We translate this *recency bias* into representational changes for the training and problem cases in two ways. The first is a direct modification to the attributes that comprise the case representation, and the second modifies the weights to indicate a constituent's distance from the relative pronoun.

In the first approach, we label the **constituent** attribute-value pairs by their position *relative to the relative pronoun*. This establishes a right-to-left labeling of constituents rather than the left-to-right labeling that the baseline representation incorporates. In Figure 8.9, for example, "in Congress" receives the attribute *pp1* in the right-to-left labeling because it is

Sentence: [It] [was] [the hardliners] [in Congress] who...
Baseline Representation: *(s entity) (v t) (do human) (do-pp1 entity) (mr-phrase entity) (mr-syn-type prep-phrase) (antecedent ((do)))*
Right-to-Left Labeling: *(s entity) (v t) (np2 human) (pp1 entity) (mr-phrase entity) (mr-syn-type prep-phrase) (antecedent ((np2)))*

Figure 8.9: Incorporating the Recency Bias Using a Right-to-Left Labeling.

a prepositional phrase one position to the left of "who." Similarly, "the hardliners" receives the attribute *np2* because it is a noun phrase two positions to the left of "who." The right-to-left ordering yields a different feature set and, hence, a different case representation. For example, the right-to-left labeling assigns the same antecedent value (i.e., *pp2*) to both of the following sentences:

- "it was a message from *the hardliners* in Congress, who..."
- "it was from *the hardliners* in Congress, who..."

The baseline (left-to-right) representation, on the other hand, labels the antecedents with distinct attributes — *do-pp1* and *v-pp1*, respectively.

In the second approach to incorporating the recency bias, we increment the weight associated with a **constituent** attribute as a function of its proximity to the relative pronoun (see Table 8.2). The feature associated with the constituent farthest from the relative pronoun receives a weight of one, and the weights are increased by one for each subsequent constituent. All features added to the case as a result of feature normalization (not shown in Table 8.2) receive a weight of one. In addition, we assign the maximum weight to both the **mr-phrase** and **mr-syn-type** features.

Table 8.2: Incorporating the Recency Bias by Modifying the Weight Vector.

Phrase	**Feature**	**Baseline Weight**	**Recency Weight**
It	s	1	1
was	v	1	2
the hardliners	do	1	3
in Congress	do-pp1	1	4
	mr-phrase	1	5
	mr-syn-type	1	5
who...			

The results of experiments that use each of the recency representations separately and in a combined form are shown in Table 8.3. To combine the two implementations of the recency bias, we first relabel the attributes of a case using the right-to-left labeling and then initialize the weight vector using the recency weighting procedure described above. The table shows that the recency weighting representation alone tends to degrade prediction of relative pronoun antecedents as compared to the baseline. Both the right-to-left labeling and combined representations improve performance — they perform significantly better than the default heuristic, but do not yet exceed the level of the hand-coded heuristics. The final row of results will be described below.

As shown in Table 8.3, the combined recency bias outperforms the right-to-left labeling despite the fact that the recency weighting tends to lower the accuracy of relative pronoun antecedent prediction when used alone. The right-to-left labeling appears to provide a representation of the local context of the relative pronoun that is critical for finding antecedents. The disappointing performance of the recency weighting representation, on the other hand, may be caused by (1) its lack of such a representation of local context, and (2) its bias against antecedents that are distant from the relative pronoun (e.g., "...to

Table 8.3: Results for the Recency Bias Representations (% correct).

Baseline	76.2
R-to-L Labeling	79.2
Recency Weighting	75.8
R-to-L + RecWt	80.0
Hand-Coded Heuristics	80.5
Default Heuristic	74.3
Baseline w/o Recency	69.2

help especially *those people* living in the Patagonia region of Argentina, who are being treated inhumanely...”). Nineteen of the 241 cases have antecedents that include the often distant subject of the preceding clause. Finally, the recency bias performs well despite the fact that the baseline representation already provides a built-in recency bias. The baseline represents the constituent that precedes the relative pronoun up to three times in the baseline representation — as a **constituent** feature, the **mr-phrase** feature, and the **mr-syn-type** feature.[11] The last row in Table 8.3 shows the performance of the baseline representation when this built-in bias is removed by discarding the **mr-phrase** and **mr-syn-type** features and disallowing references to them in the **antecedent** class value.

8.6.2 Incorporating the Restricted Memory Bias

Psychological studies have determined that people can remember at most seven plus or minus two items at any one time [Miller, 1956]. More recently, Daneman and Carpenter [1983, 1980] show that working memory capacity affects a subject’s ability to find the referents of pronouns over varying distances. King and Just [1991] show that differences in working memory capacity can cause differences in the reading time and comprehension of certain classes of relative clauses. Moreover, it has been hypothesized that language learning in humans is successful precisely because limits on information processing capacities allow children to ignore much of the linguistic data they receive [Newport, 1990]. Some computational language learning systems (e.g., Elman [1990]) actually build a short term memory directly into the architecture of the system.

WHirlpool’s baseline case representation does not necessarily make use of this *restrict memory bias*, however. Each case is described in terms of the normalized feature set, which contains an average of 38.8 features across all partitions of the 10-fold cross validation. Unfortunately, incorporating the restricted memory limitations into WHirlpool’s case representation is problematic. Previous restricted memory studies (e.g., short term memory

[11]This means that when the constituent immediately preceding “who” in the problem case and a training case match, that constituent accounts for a greater percentage of the similarity score than does any other constituent.

studies) do not state explicitly what the memory limit should be — it varies from five to nine depending on the cognitive task and depending on the size and type of the "chunks" that have to be remembered. In addition, the restricted memory bias alone does not state which chunks, or features, to keep and which to discard.

To apply the restricted memory bias to the baseline case representation, we let n represent the memory limit and, in each of five runs, set n to one of five, six, seven, eight, or nine. Then, for each test case, the system randomly chooses n features from the normalized feature set, sets the weights associated with those features to one, and sets the remaining weights to zero. This effectively discards all but the n selected features from the case representation.

Table 8.4: Results for the Restricted Memory Bias Representation (% correct). (*'s indicate significance with respect to the original baseline result shown in **boldface**, $* \rightarrow p = 0.05$).

Memory Limit	Baseline	R-to-L + RecWt
none	**76.2**	80.0
9	78.3	81.2*
8	74.2	81.2*
7	76.2	80.0
6	75.8	80.4
5	75.0	81.7*

Results for the restricted memory bias representation are shown in Table 8.4. The first column of results shows the effect of memory limitations on the baseline representation. In general, the restricted memory bias with random feature selection degrades the ability of the system to predict relative pronoun antecedents although none of the changes is statistically significant. This is not surprising given that the current implementation of the bias is likely to discard relevant features as well as irrelevant features. We expect that this bias will have a positive impact on performance only when it is combined with cognitive biases that provide feature relevancy information. This is, in fact, the case. The final column in Table 8.4 shows the effect of restricted memory limitations on the combined recency representation. To incorporate the restricted memory bias and the combined recency bias into the baseline case representation, we (1) apply the right-to-left labeling, (2) rank the features of the case according to the recency weighting, and (3) keep the n features with the highest weights (where n is the memory limit). Ties are broken randomly.

We expected the merged representation to perform rather well because the combined recency bias representation worked well on its own and because the restricted memory (RM) bias essentially discards features that are distant from the relative pronoun and rarely included in the antecedent. As shown in the last column of Table 8.4, four out of five RM/recency variations posted higher accuracies than the combined recency representation. In fact, three of the RM/recency representations now outperform the original baseline representation (shown in boldface) at the 95% significance level. (Until this point, the best representation had been the combined recency representation, which significantly outperformed the default heuristic, but not the baseline case representation.)

8.6.3 Incorporating the Subject Accessibility Bias

A number of studies in psycholinguistics have noted the special importance of the first item mentioned in a sentence. In particular, it has been shown that the accessibility of the first discourse object, which very often corresponds to the subject of the sentence, remains high even at the end of a sentence [Gernsbacher *et al.*, 1989]. This *subject accessibility bias* is an example of a more general *focus of attention bias*. In vision learning problems, for example, the brightest object in view may be a highly accessible object for the learning agent; in aural tasks, very loud or high-pitched sounds may be highly accessible. We incorporate the subject accessibility bias into WHirlpool's baseline representation by increasing the weight associated with the constituent attribute that represents the subject of the clause preceding the relative pronoun whenever that feature is part of the normalized feature set.

Table 8.5: Results for the Subject Accessibility Bias Representation (% correct).

Baseline	**Baseline SubjWt=2**	**Baseline SubjWt=5**	**Baseline SubjWt=7**	**Baseline SubjWt=10**
76.2	75.0	74.2	73.7	73.3

Table 8.5 shows the effects of allowing matches on the subject attribute (**s**) to contribute two, five, seven, and ten times as much as they did in the baseline representation. Weights on the **s** attribute were chosen arbitrarily. Results indicate that incorporation of the subject accessibility bias never improves performance of the learning algorithm, although dips in performance are never statistically significant. At first it may seem surprising that this bias does not result in a better representation. Like the recency bias, however, WHirlpool's baseline representation already encodes the subject accessibility bias by explicitly recognizing the subject as a major constituent of the sentence (i.e., **s**) rather than by labeling it merely as a low-level noun phrase (i.e., **np**). It may be that this built-in encoding of the bias is adequate or that, like the restricted memory bias, additional modifications to the baseline representation are required before the subject accessibility bias can have a positive effect on the learning algorithm's ability to find relative pronoun antecedents.

Table 8.6 shows the effects of merging the subject accessibility bias with both recency biases (R-to-L refers to the right-to-left labeling and RecWt refers to the recency weighting representation) and the restricted memory bias (RM). The results in the first column (Baseline) are just the results from Table 8.5 — they indicate the performance of WHirlpool's baseline case representation with various levels of the subject accessibility bias. The

Table 8.6: Additional Results for the Subject Accessibility Bias Representation (% correct). (*'s indicate significance with respect to the original baseline result shown in **boldface**, $* \rightarrow p = 0.05$, $** \rightarrow p = 0.01$; RM refers to the memory limit).

Subject Weight	Baseline	SubjAcc R-to-L RecWt	SubjAcc R-to-L RecWt RM=5	SubjAcc R-to-L RecWt RM=6	SubjAcc R-to-L RecWt RM=7	SubjAcc R-to-L RecWt RM=8	SubjAcc R-to-L RecWt RM=9
none	**76.2**	80.0	81.7*	80.4	80.0	81.2*	81.2*
2	75.0	79.6	*84.2***	*82.5**	*82.1**	81.2*	80.8
5	74.2	78.3	79.6	79.2	78.3	80.4	79.6
7	73.7	77.5	79.6	79.2	77.9	76.7	77.9
10	73.3	76.7	79.6	79.2	78.3	80.4	79.6

second column shows the effect of incorporating the subject accessibility bias into the combined recency bias representation. To create this merged representation, we first establish the right-to-left labeling of features and then add together the weight vectors recommended by the recency weighting and subject accessibility biases. As was the case with the baseline representation, incorporation of the subject accessibility bias steadily decreases performance of the learning algorithm as the weight on the subject constituent is increased. None of the changes is statistically significant.

The remaining five columns of Table 8.6 show the effects of incorporating all three cognitive biases into the baseline case representation. To create this representation, we (1) relabel the attributes using the right-to-left labeling, (2) incorporate the subject and recency weighting representations by adding the weight vectors proposed by each bias, (3) apply the restricted memory bias by keeping only the n features with the highest weights (where n is the memory limit) and choosing randomly in case of ties. Results for these experiments indicate that some combinations of the cognitive bias parameters work very well together and others do not. In general, associating a weight of two with the subject constituent improves the accuracy of the learning algorithm as compared to the corresponding representation that omits the subject accessibility bias. (Compare the first and second rows of results). In particular, three representations (shown in italics) now outperform the best previous representation (which had the r-to-l labeling, recency weighting, memory limit = 5 and achieved 81.7% correct).[12]

8.6.4 Discussion

It should be emphasized that modifications to the baseline case representation in response to each of the individual cognitive biases are performed **automatically** by WHirlpool,

[12]In addition, the best-performing representation now outperforms the hand-coded relative pronoun disambiguation rules (84.2% vs. 80.5%) at the 90% significance level.

subject to the constraints provided in Table 8.7. Upon invocation of WHirlpool, the user need only specify (1) the names of the biases to incorporate into the case representation, and (2) any parameters required for those biases (e.g., the memory limit for the restricted memory bias). WHirpool's approach to improving a case representation should also be applicable to learning tasks in other domains. The right-to-left labeling implementation of the recency bias, for example, can be applied to any learning task for which (1) there is a temporal ordering among the features, and (2) this ordering is denoted in the attribute names. To invoke this bias, the user must supply a function that maps the original attribute names to new attribute names.

Table 8.7: Cognitive Bias Modifications.

Bias	**Assumptions**	**Parameters**
Recency (r-to-l labeling)	Attribute names indicate recency	Function mapping original attribute names to new attribute names
Recency (recency weighting)	Attributes in original case are provided in inverse recency order	None
Restricted Memory	None	memory limit
Focus of Attention (subject accessibility)	None	Weight factor, attribute associated with object of focus, e.g., the subject

In addition, WHirlpool relies on the following general procedure when incorporating more than one cognitive bias into the baseline representation:

1. First, incorporate any bias that relabels attributes (e.g., r-to-l labeling).

2. Then, incorporate biases that modify feature weights by adding the weight vectors proposed by each bias (e.g., recency weighting, subject accessibility bias).

3. Finally, incorporate biases that discard features (e.g., restricted memory bias), but give preference to those features assigned the highest weights in Step 2.

One problem that has not been addressed in the discussions above is how to select automatically the combination of cognitive biases that will achieve the best performance for a particular learning task. For the relative pronoun task, we exhaustively enumerated all combinations of available cognitive biases and chose the combination that peformed best in cross-validation testing. Because this method will get quickly out of hand as additional biases are included or parameters tested, future work should investigate less costly alternatives to cognitive bias selection.

In spite of its incorporation of cognitive biases, WHirlpool tends to make two classes of errors. First, a portion of WHirlpool's errors are due to insufficient training: either the problem case required an antecedent combination never seen in any of the training cases or the problem case involved a new syntactic context for the wh-word. In one sentence, for example, "who" was preceded by a preposition, i.e., "regardless of who," and no context similar to this one had been encountered during training. When the semantic and syntactic structure of the novel clause differs significantly from any represented in the context features of the training cases, we cannot expect accurate retrieval from the case base. The remaining errors exhibited by WHirlpool's learned heuristics generally involve relative pronoun antecedents that are difficult even for people to specify. Some examples are provided below. (In each example below, the antecedent chosen by WHirlpool is indicated in *italics*; the correct antecedent is shown in **boldface** type.)

1. "... 9 rebels died at the hands of **members** of *the civilian militia*, who resisted the attacks"
2. "...*the number* of **people** who died in Bolivia..."
3. "...**the rest** of *the contra prisoners*, who are not on this list..."

8.7 Summary

This chapter presented WHirlpool, an instantiation of the Kenmore framework that locates the antecedent of the relative pronoun "who." We find that our automated approach to relative pronoun disambiguation performs as well as a set of hand-coded rules for relative pronoun disambiguation and performs significantly better than a default heuristic that chooses the most recent phrase as the antecedent. The chapter also presented a cognitive bias approach to improving a baseline case representation. The experiments of Sections 8.6.1-8.6.3 showed that performance of WHirlpool's case-based algorithm steadily improved as each of the available cognitive biases was incorporated into its baseline case representation. Although one would not expect monotonic improvement to continue forever, it is clear that explicit incorporation of cognitive biases into the case representation can improve the learning algorithm performance for the relative pronoun disambiguation task. Table 8.8 summarizes these results.

Table 8.8: Summary of Cognitive Bias Results.

Case Representation	**% Correct**
Baseline w/o Built-in Recency Bias	69.2
Default Heuristic: Choose Most Recent Phrase	74.3
Baseline	76.2
Baseline + Recency Bias	80.0
Hand-Coded Heuristics	80.5
Baseline + Recency Bias + Restricted Memory Bias (limit=5)	81.7
Baseline + Recency Bias + Restricted Memory Bias (limit=5) + Subject Accessibility Bias (subj wt=2)	84.2

CHAPTER 9

ESTABLISHING CONSTRAINTS ON THE KENMORE FRAMEWORK

This thesis presents the Kenmore framework for knowledge acquisition for natural language processing systems. As described in the introduction and elsewhere, Kenmore requires three major components: a robust sentence analyzer, a corpus of texts, and an inductive learning module. This chapter draws from the results of previous chapters and establishes constraints on each component that must be satisfied for successful instantiation of the framework on a new problem. In addition, it offers practical advice on how to attack a new parsing ambiguity problem within the Kenmore framework. In general, the following tasks must be performed:

1. Choose a corpus.

2. Choose a sentence analyzer.

3. Set up variable-depth semantic knowledge structures as well as any necessary background syntactic knowledge.

4. Choose the parsing problem.

5. Specify the case representation and the method for improving case representation.

6. Choose the inductive learning algorithm. (For case-based approaches, this includes choosing the case-indexing method, the similarity metric, and the method for combining, or deciding among, solutions proposed by the retrieved cases.)

7. Train and test the system.

The remaining sections of the chapter discuss each step separately. The final section of the chapter shows how one might use Kenmore to learn heuristics for *many* parsing decisions simultaneously rather than one at a time.

9.1 The Corpus

Kenmore espouses a corpus-based view of knowledge acquisition for natural language processing. Disambiguation heuristics are implicitly encoded in the case base as training sentences from the corpus are processed by the sentence analyzer and a human supervisor. In particular, the thesis demonstrates the performance of Kenmore only using domain-specific corpora. As discussed in Section5.5.4, it is not clear that Kenmore will be useful with corpora containing totally unrestricted texts because the length of the human-supervised training phase would increase dramatically. If the supervised training phase can be transformed into an unsupervised one, however, then Kenmore should be usable with unrestricted text. (Extending Kenmore in this direction will be discussed in Chapter 10.)

In general, Kenmore will be judged successful if the heuristics it learns perform accurate disambiguation within a specific language processing task. Therefore, a very basic criterion for the corpus is simply that it contain examples of the kinds of sentences, stories, or text fragments that are most important for the NLP system to understand. If the goal of the NLP system is to summarize stories about terrorist events, for example, then the training corpus primarily should contain texts that describe terrorist events. If, on the other hand, it will be important for the NLP system to distinguish sentences/texts that contain relevant information from those containing no relevant information, then the training corpus should contain a mixture of both relevant and irrelevant text excerpts.

The minimum size of the corpus will vary depending on four related factors: the scope of the overall natural language task, the particular problem in sentence analysis for which heuristics need to be acquired, the variety of sentence structures that the system must process, and the size of any syntactic and semantic taxonomies or knowledge structures. As the scope of the natural language understanding task expands, for example, it may be important that the NLP system obtain a deeper understanding of the input texts. This, in turn, may require that (1) the system process sentences with a greater variety of syntactic structures, and (2) the system recognize finer distinctions among semantic categories. Both (1) and (2) imply that Kenmore will require additional training sentences. In general, as any associated semantic and syntactic taxonomies grow or the NLP task becomes more complicated, Kenmore will require additional training sentences, and hence, a bigger corpus.

Even so, experiments with the MayTag and WHirlpool systems, indicate that surprisingly few training sentences are necessary to obtain adequate performance for many real-world tasks. WHirlpool used 108 training sentences (90% of the base set of 120 sentences) to acquire relative pronoun disambiguation heuristics for use with texts in the terrorism domain. MayTag used 157 training sentences to learn lexical disambiguation heuristics for use with texts in the business joint ventures domain. Kenmore's reliance on knowledge beyond word frequencies may account for its needing such a small training corpus — training cases in Kenmore encode the state of the parser as well as semantic information for individual words while many statistical approaches maintain only individual word counts and possibly part-of-speech information. As a result, Kenmore needs fewer training cases than approaches that fail to include abstract views of the current context. Our current experiments, however, indicate that the corpus should contain a minimum of 100-200 sentences.

9.2 The Sentence Analyzer

All work described here relies on the CIRCUS conceptual sentence analyzer. However, Kenmore can be instantiated with any parser as long as it conforms to the following constraints. First, the sentence analyzer must be robust enough to process large amounts of text. However, only a partial syntactic parse of each sentence is required. The sentence analyzer need only recognize individual noun phrases, prepositional phrases, and verb phrases without making any attachment decisions. Parsers that perform full syntactic analyses may, in fact, have to "undo" some attachment decisions to create cases in attribute-value pair format. Alternatively, the inductive learning component could accept cases that represent trees rather than flat structures. In addition, the parser should be able to locate the major syntactic constituents of each clause — the verb, subject, direct object, and/or indirect object. In theory, either a serial or parallel parser can be used: Kenmore would be consulted to resolve the current ambiguity directly or to prune one or more existing representations maintained by the parser.

Second, the parser must also be able to observe its own state. That is, it should be able to provide a description of all syntactic and semantic structures it builds and maintains during sentence processing. Although the sentence analyzer used here relies heavily on semantic knowledge, this is not a constraint enforced by the Kenmore framework. Nonetheless, all results obtained in earlier chapters relied on case representations that included this semantic information and further experimentation would be required to determine whether the framework is successful without this semantic knowledge. We emphasize, however, that the type of information maintained by a parser affects how well and how easily disambiguation heuristics can be learned rather than whether the parser can be used within the Kenmore framework.

9.3 The Taxonomies

To instantiate the Kenmore framework to solve a new problem in sentence analysis, system developers must also set up any taxonomies that will be required. In the lexical acquisition task, for example, MayTag tagged words in an incoming text with the appropriate syntactic and semantic class. In order to do this, we provided MayTag with part-of-speech and semantic feature hierarchies. The taxonomies used in both the MayTag and WHirlpool systems observed the guidelines established in Chapter 4 regarding variable-depth knowledge representations. To ensure that new instantiations of the Kenmore framework are successful in the larger language processing task in which they are being used, we suggest that system developers abide by these guidelines as well.

In short, any semantic hierarchies should focus on the representation of domain-specific knowledge and should be detailed enough to make the distinctions required in the domain of interest. Any syntactic hierarchies should be detailed enough to allow the parser to accomplish its goals, e.g., segmentation of incoming text into low-level constituent phrases. Further experimentation would be required to determine which of these constraints can be relaxed without any loss in performance.

In Section 7.2.2 we found that MayTag was better at assigning domain-dependent semantic features than domain-independent ones. This implies that Kenmore will be more successful when the NLP tasks require primarily domain-dependent taxonomies. On the other hand, the results of Section 7.2.2 indicate that Kenmore should perform adequately on even domain-independent semantic features if those features are fairly general ones. The key in determining the success of Kenmore on any particular task is to balance two conflicting constraints. First, the underlying taxonomies must be detailed enough to support the natural language task. Second, the underlying taxonomies should be general enough to allow adequate coverage of each value in the taxonomy across the training cases. In semantic feature tagging, for example, Kenmore will not learn to recognize words denoting *humans* or *company-names* unless it is given a reasonable number of examples of each. In general, the first constraint supersedes the second because the second constraint can always be satisfied by extending the training phase until enough cases have been created.

9.4 The Parsing Task To Be Learned

Kenmore provides a framework within which solutions to problems in sentence analysis can be learned. The only constraints on the choice of parsing problem are that

1. the sentence analyzer be able to recognize an instance of the problem, and

2. the problem be recast as a classification problem.

As an example, consider the problem of prepositional phrase attachment. To satisfy the first constraint, the parser must know when a prepositional phrase attachment decision will need to be made — just after each prepositional phrase is recognized. This is important because the parser must create a case whenever an instance of the problem is encountered.

In general, lexical ambiguities (e.g., concept activation, semantic feature tagging), require that a decision be made for individual words. Therefore, the parser should create a lexical disambiguation case as each word is recognized. Structural ambiguities (e.g., relative pronoun disambiguation, prepositional phrase attachment, understanding conjunctions and appositives, analysis of compound nouns), on the other hand, represent attachment decisions rather than tagging decisions. As a result, a case should be created when the attachment decision must be made (e.g., after the prepositional phrase has been recognized for prepositional phrase attachment and after the wh-word has been recognized for relative prounoun disambiguation).

To satisfy the second constraint, prepositional phrase attachment must be recast as a classification problem: we must specify the features that will encode the context of the ambiguity as well as what kind of "class" information should be associated with each prepositional phrase attachment decision. This amounts to determining what the "context" and "solution" parts of the cases should look like. In the Kenmore framework, the representation of context depends more on the parser than on the ambiguity task — it will be discussed in the next section. The structure of the class information depends on the type of ambiguity being resolved. For lexical ambiguities, the class information is an item(s) from the appropriate taxonomy (e.g., the concept taxonomy for concept tagging, the semantic feature hierarchy for semantic feature tagging). For structural ambiguities like prepositional phrase attachment, the class information specifies an attachment point. In relative pronoun disambiguation, for example, the attachment point is the antecedent of the relative pronoun, i.e., the phrase that the upcoming embedded clause modifies. For each phrase in a conjunction (other than the first one), the attachment point is the preceding phrase in the conjunction (e.g., in "I saw Nelson and Winnie," "Winnie" attaches to "Nelson"). For some attachment decisions, we may also want to include additional information as part of the class information. In compound noun analysis, for example, it is also important to indicate the semantic relationship between nouns in the compound as well as their attachment points. In "the Northampton sushi bar," "Northampton" and "sushi" modify "bar," but it would be more informative to note that the "bar" is "*in* Northampton" and "*serves* sushi."

Given our experience with MayTag and WHirlpool, we can generalize as to how the two constraints described above can be satisfied for new problems in sentence analysis.

It should be noted that specifying the nature of the class information to associate with some parsing problems may be difficult. The problem arises not because it is difficult to view the task as a classification problem, but because the problem involves a number of decisions, some of which need to be modified as more of the sentence is uncovered. Consider the following sentence:

I saw Mary Ford, the mayor of Northampton, and Bill Clinton, our president.

In this sentence, "Mary Ford" and "the mayor of Northampton" are in apposition to one another and the conjunction joins two entities, "Mary Ford" and "Bill Clinton." Without the appropriate background knowledge, however, we might initially interpret "the mayor of Northampton" as an element of the conjunction (rather than as an appositive) until we reach the phrase "our president." Examples like this indicate that the class information associated with an ambiguity sometimes may have to include adjustments to prior decisions that are now deemed incorrect.

9.5 The Case Representation: Representing Context

The "context" portion of Kenmore cases encodes the state of the parser at the point of an ambiguity. As discussed in earlier chapters, the success of the inductive learning algorithm, and hence, the success of Kenmore, depends critically on using an appropriate representation of context. Unfortunately, designing a good case representation is by far the most difficult step in the instantiation of the Kenmore framework. It is time-consuming and knowledge-intensive. Given the results from preceding chapters, we have established a single overriding constraint on case representation design that must be satisfied for the learning task to succeed: *the baseline case representation must be one that is natural for the sentence analyzer*. This constraint is necessary because the sentence analyzer is solely responsible for providing the context portion of the cases. This constraint also limits the number of appropriate representations to consider. A purely syntactic sentence analyzer, for example, will not be able to include semantic information the case representation. In addition, the learning algorithm constrains case representation design because it limits the structure of the cases. The inductive learning components used in MayTag and WHirlpool, for example, require that cases be presented in attribute-value pair format.

9.5.1 Using a Uniform Case Representation

One question that was not addressed in the experiments of earlier chapters is the degree to which a single case representation should be used across all parsing problems within the Kenmore framework. Does each problem in sentence analysis require its own representation of context or will a single representation of context suffice? Does the real answer lie somewhere in between? MayTag was able to learn solutions to a number of problems in lexical ambiguity using the same baseline case representation in conjunction with a decision tree approach to feature set selection that located relevant features in the baseline representation. This implies that Kenmore may be able to handle many problems in lexical ambiguity resolution using the same baseline case representation as long as that baseline representation is tuned for each lexical ambiguity task using some feature selection techniques. Because we tackled only one structural ambiguity task within the Kenmore framework, it is impossible to know whether a single case representation will also suffice for all structural ambiguities.

Still, given the results of experiments with MayTag and WHirlpool, it is not clear whether Kenmore should use the same baseline case representation for lexical and structural ambiguities alike. MayTag and WHirpool used very similar, but not identical, baseline case representations. For example, MayTag cases include information for each word in a five word window surrounding the unknown word, but WHirlpool cases include word-level information only for the word that precedes the relative pronoun. WHirlpool cases, on the other hand, include an attribute-value pair for all major constituents and low-level phrases in the current clause. MayTag cases include only information for the major constituents of the current clause. Given these differences in case representation, one might conclude that lexical ambiguity resolution and structural ambiguity resolution require access to different sources of knowledge. In particular, it would appear that lexical ambiguity resolution requires access to a finer-grained representation of local context than does structural ambiguity resolution, while structural ambiguity resolution requires a more complete representation of global context.

While these hypotheses may be true, we still advocate the use of a uniform baseline case representation for all problems to be solved within the Kenmore framework. In addition, however, we advocate the use of automated approaches that tune the baseline representation for each problem in sentence analysis by locating relevant parts of the context and by modifying the representation in response to appropriate biases. Figure 9.1 explicitly indicates the use of such techniques in the Kenmore framework. This approach separates ambiguity resolution into two distinct tasks: (1) the task of gathering together all possible disambiguation cues, and (2) the task of deciding which of the available cues are relevant to the current problem. In Kenmore, the sentence analyzer performs the first task and the inductive learning component performs the second. One advantage of this separation is that it simplifies the job of the parser — the parser supplies the same "context" regardless of the type of ambiguity. In addition, it ensures that the learning component has access to all available sources of information as it learns a solution to a particular problem in sentence analysis.

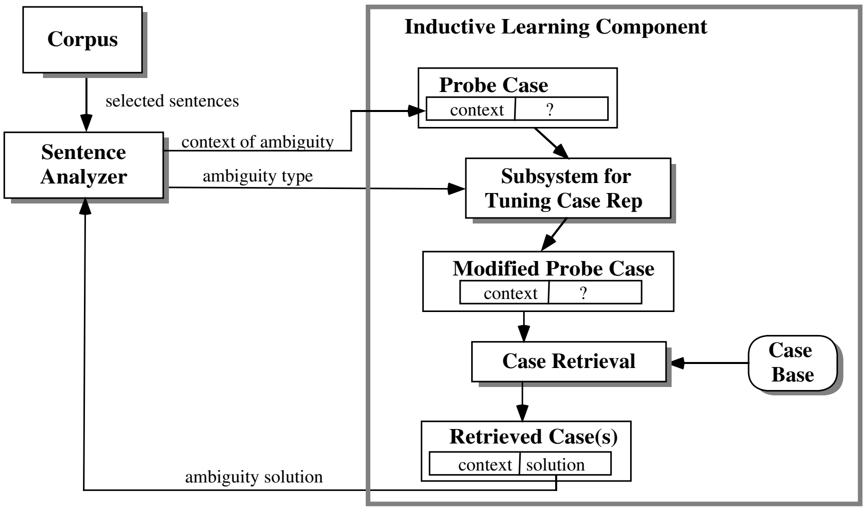

Figure 9.1: Modified Kenmore Framework.

9.5.2 Righthand Context

Another issue in the representation of context for Kenmore that remains to be addressed is the role of righthand context. Sometimes the resolution of an ambiguity problem hinges on information that *follows* the ambiguity. As an example, consider the sentence:

I saw the daughter of the colonel, who fought in Vietnam.

Before seeing the embedded clause, it is impossible to know if the antecedent of "who" is "the daughter" or "the colonel." However, after the embedded clause has been processed, it appears fairly certain that "who" refers to "the colonel." Kenmore cases, however, include little or none of that righthand context. MayTag cases, for example, include information for just the two words that follow the unknown word and WHirlpool cases encode only the preceding context. We have chosen to ignore the succeeding context because the parser has not processed that portion of the input in its left-to-right traversal of the text. To handle circumstances where the righthand context is important, however, the general procedure for ambiguity resolution in Kenmore could be modified as follows. During training, when an ambiguity is encountered that cannot be resolved based on the lefthand context (and possibly a very limited lookahead), the solution portion of the case might note the need for righthand context as well as indicating the best guess for the ambiguity resolution decision. If this case were retrieved during the application phase, the parser would know to resolve the ambiguity initially based on the "best guess" and to consider revising the decision as more of the sentence is processed. These modifications complicate both the generation of the solution part of the case during training and the processing of the solution part of the case during application.

9.5.3 Improving the Case Representation Automatically

In general, Kenmore needs to modify a baseline case representation in three ways in order to learn solutions to problems in sentence analysis. It must:

1. Discard irrelevant attributes.
2. Determine the relative importance of relevant attributes.
3. Add new attributes when the existing ones are inadequate for the learning task.

Chapters 6 and 8.6 present two domain-independent methods for improving a case representation. MayTag's decision tree approach to feature set selection performs the first task (and could be extended to perform the second by taking advantage of the specific information gain value associated with each attribute). WHirlpool's cognitive bias approach to improving a baseline case representation spans all three types of modifications. When combined with the recency bias, the restricted memory bias discards irrelevant features from the representation by setting their weights to zero. The recency weighting and subject accessibility biases note the relative importance of features by modifying the weights associated with them. The right-to-left labeling recency bias is a weak method for adding new attributes the representation. In the experiments of Chapters 6 and 8.6, we applied each of the methods for improving a case representation separately — the decision tree approach was used for the lexical ambiguity task and the cognitive bias approach was used with the structural ambiguity task. However, it is also possible to apply both approaches to an ambiguity task if the combination improves performance.[1]

---

[1]To implement the combined approach, we first apply the decision tree approach to the baseline case representation and then apply the cognitive bias approach to the resulting case representation. Alternatively,

Because creating a good instance representation is crucial for the success of the learning algorithm, and because the learning algorithm itself does not typically actively help in choosing or tuning this representation[2], additional methods for automatically improving a baseline case representation should be explored. In particular, it will be important to find methods that perform modifications of the third type — adding new features to improve performance of the learning algorithm. The sections below describe two such methods. Like the techniques used in MayTag and WHirlpool, any new approaches should be automated.

9.5.3.1 Making Finer Distinctions

One method for adding features to a case representation is to expand an existing attribute-value pair into two or more features so that each dimension of knowledge represented by the original feature is encoded separately. As an example, we might expand a **constituent** feature in WHirlpool into two features that represent syntactic and semantic knowledge separately. Instead of encoding the phrase "in New York" as *(pp1 location)*, we might represent it as *(pp1 in)* and *(pp1-gen-sem location)* or even *(constit1 pp)*, *(constit1-prep in)*, and *(constit1-gen-sem location)*. This type of modification allows the case retrieval algorithm to match on each piece of knowledge separately and is especially useful when retrieval of conflicting cases is a problem. (Two retrieved cases conflict when they have the same similarity score, but different class information.) By using a more detailed constituent representation, fewer such conflicts should occur.

9.5.3.2 Derived Features and the Issue of Transfer

A second method for adding features to an existing case representation is to use derived features — higher level features that are derived from low-level ones. Some of the features currently used in MayTag and WHirlpool cases can be considered derived features. All of the features that represent major syntactic constituents, for example, combine information from each of the words that comprise it and also incorporate a decision by the sentence analyzer regarding the structure of the clause or sentence. There are many more opportunities to incorporate derived features in Kenmore cases, however. In particular, we might include features that allow Kenmore to benefit from its own ambiguity resolution decisions. Currently, Kenmore's cases represent a flattened view of the structure of the current clause. However, as Kenmore is accessed to resolve various structural ambiguities during sentence analysis, its decisions could be added to the existing case representation dynamically as derived features. Consider the following sentence,

---

we can use the information gain measure to assign an initial set of weights to the features in the baseline case representation and then apply the cognitive biases.

[2]There are exceptions, of course. E.g., the constructive induction systems of Callan and Utgoff [1991] and Fawcett and Utgoff [1992].

> Police also spoke with the man beside the white Ford Bronco, who allegedly drove the vehicle.

When WHirlpool encounters this sentence, it creates a case that contains **constituent** and **most-recent** features and is based on the state of the parser when "who" is recognized. If we assume that Kenmore will be used to make all attachment decisions, however, then the prepositional phrase attachment decision for "beside the white Ford Bronco" already would have been made. Because this decision may, in turn, inform the relative pronoun disambiguation decision, we could let the sentence analyzer include it as part of the context in the relative pronoun disambiguation case. This type of derived feature allows a "transfer" of knowledge among disambiguation tasks. In the above example, solving a prepositional phrase attachment ambiguity may help with relative pronoun disambiguation.

9.6 The Inductive Learning Component

There are a number of constraints on Kenmore's learning component. First, the learning component must be inductive. This is a basic assumption of the Kenmore framework. Second, the structure of the case representation used by the learning algorithm should be general enough to accommodate naturally the representations employed by the sentence analyzer. Although the attribute-value pair representation appears to be adequate for use with the CIRCUS parser, inductive learning algorithms that allow structured case representations (e.g., FOIL [Quinlan, 1990]) would be worth exploring, especially for use with sentence analyzers that create and maintain such structured representations. Finally, it is best if the learning algorithm is incremental: the description of the concept being learned is best updated after each training case without reprocessing all of the preceding training cases. This allows the human-supervised training phase to become progressively easier as more cases are acquired without slowing down the training process.

9.7 Training Issues

In order for Kenmore to learn solutions to problems in sentence analysis, the training phase must meet the following constraints. First, *training sentences should be selected randomly from the corpus*. In addition, because the work discussed here assumes the task of domain-specific text processing, it may make sense to choose the majority of training sentences from relevant rather than irrelevant texts for some, but not all, learning tasks. For semantic feature and part-of-speech disambiguation, for example, it is best if the training sentences are randomly chosen from sentences that contain information relevant to the text-processing task because it will be performance on words from these sentences that affects the overall performance of the NLP system. Irrelevant sentences effectively should be ignored by the NLP system, making the assignment of correct semantic feature and part-of-speech tags for lexical items in these sentences unimportant. However, there are disambiguation tasks that can benefit from the presence of irrelevant sentences in the training corpus. One example is the activation of domain-specific concepts, which was performed by MayTag. If a concept is activated, the entire sentence is considered relevant by CIRCUS. Therefore, it will be important to include irrelevant sentences in the training set so that the learning component can learn to distinguish relevant from irrelevant concept activation contexts.

A second constraint on the training phase is that it *continue long enough to gather an adequate number of training cases*. We saw in Chapter 5.5.2.1 that increasing the number of training cases by 50% provided increases in the accuracy of general and specific semantic feature tagging by 19% and 12%, respectively. Kenmore's training phase can end when one of two conditions is met: (1) overall classification accuracy is above some predetermined value, or (2) when learning curves indicate that performance is no longer increasing or that it is increasing at some very small rate. In training MayTag for the UMass-Hughes MUC-5/TIPSTER system, we loosely adhered to the first condition and stopped training when MayTag's performance across all words in the training sentences (including the function words) surpassed 85% accuracy level. In general, however, learning curve data will provide an earlier indication of when the training phase should end. One training scenario is to choose training sentences randomly until the learning curves indicate that learning has slowed considerably. Then, we can examine the system's performance on individual classes and select additional training sentences to cover underrepresented classes.

Third, *the training phase should be sensitive to the different types of knowledge being learned* because different types of training examples may be required to learn a particular concept in sentence analysis. For example, the MayTag's semantic feature set contains two types of features — *set list* features that are used to tag a small, fixed set of words and *open list* features that will be associated with a much broader set of lexical items. Virtually any word can be tagged as a *company-name* or *company-alias*, for example, but generally only a few expressions are used in the joint ventures domain to denote the president of a company — approximately 99% of the 113 texts that refer to the president of a company use either the word "president" or "head." Therefore, *company-name* and *company-alias* would be considered open list semantic features, while *president* would be a set list feature. Choosing training sentences randomly will be adequate for learning open list features, but we may want to choose specific examples for the set list features to ensure adequate representation in the case base since many of the set list features tend to appear less frequently in the corpus.

Finally, *Kenmore's training phase should test a number of values for k* when a k-nearest neighbors inductive learning component is being used. All of our experiments tested a few values of k, but additional research is required in order to generalize from current results to determine when specific values of k are appropriate.

9.8 Putting It All Together

Throughout the thesis we have assumed that Kenmore tackles each problem in sentence analysis more or less in isolation: MayTag handled three kinds of lexical ambiguity simultaneously, but relative pronoun disambiguation heuristics were learned in a separate instantiation of Kenmore. This section outlines how Kenmore might be used to learn solutions to multiple parsing problems for a new natural language system.

1. **Determine all problems in sentence analysis that must be tackled.** These will include both lexical and structural disambiguation problems and problems that include syntactic as well as semantic decisions.

2. **Choose the corpus, sentence analyzer, and inductive learning algorithm(s).** These should be chosen subject to the constraints described above in Sections 9.1, 9.2, and 9.6. If the inductive learning component is a case-based component, the case-indexing method, similarity metric, and case selection methods should be chosen.

3. **Set up taxonomies to be associated with each problem.** In most cases, these will be associated with lexical ambiguities. However, some structural ambiguities may have associated taxonomies. Compound noun analysis, for example, may require a set of semantic relations that indicate the ways in which two nouns can be related.

4. **Determine how the class information associated with each problem will be supplied by the human supervisor.** For all lexical tagging problems tackled by MayTag, for example, the user was presented with the associated taxonomy and then asked to choose which item(s) should be used to tag the current word. For relative pronoun disambiguation, the user was presented with all low level constituents from the clause preceding the relative pronoun and was asked to choose which phrase(s) represented the antecedent of the relative pronoun. The same information could be presented to the user for prepositional phrase attachment decisions, but the user would be asked to choose the phrase that the current prepositional phrase modified.

5. **Design the baseline case representation.** This should be done subject to the constraints discussed in Section 9.5 above.

6. **Choose the training sentences and train on all sentence analysis tasks simultaneously.** The user proceeds through the training sentences, one by one, specifying supervisory information for all ambiguity decisions in the same pass. Given our results using MayTag, the lexical ambiguity decisions associated with each word can be stored together in a single case and added to a lexical ambiguity decision case base. Until additional experiments indicate otherwise, a separate case base should be maintained for each structural ambiguity decision.

7. **Use Kenmore during training to suggest a solution to each ambiguity encountered.** As cases are gathered, they can be immediately accessed by Kenmore to show the human supervisor the system's best guess for each new ambiguity. In this way, training becomes progressively easier.

8. **Monitor Kenmore's performance during training.** This amounts to running cross-validation experiments in the background to determine when training on each sub-problem in sentence analysis can terminate. As training progresses, the user need not be asked to supply supervisory information for tasks for which enough training cases have been gathered. Methods for improving the case representation also could be explored in the background during training.

CHAPTER 10

CONCLUSIONS

This thesis presents a general framework for tackling the knowledge-engineering bottleneck for natural language processing systems at the level of sentence analysis. The Kenmore knowledge acquisition framework exploits an on-line corpus using a robust parsing strategy and symbolic machine learning techniques, and requires minimal human intervention. In Kenmore's partially automated acquisition phase, the solution to a particular problem in sentence analysis is learned. The object of the training phase is to create a case base of ambiguity resolution episodes for a particular type of ambiguity. This is accomplished by selecting a set of sentences from the corpus, presenting them to a sentence analyzer for processing, and creating a case every time an instance of the ambiguity occurs. Cases are created by the sentence analyzer and a human supervisor. In Kenmore's application phase, we use the case base without human intervention to resolve ambiguities in novel sentences. Whenever the sentence analyzer encounters an ambiguity, it creates a problem case, compares the problem case to each case in the case base, retrieves the most similar training case, and sends the solution stored there back to the parser to be used as the solution in the current situation.

Below we discuss the contributions of the work and describe a number of directions for future work.

10.1 Contributions of the Research

The thesis makes two major contributions to the field of natural language processing. **First, it demonstrates that symbolic inductive machine learning and case-based techniques can be used to learn solutions to significant problems in sentence analysis.** We instantiated Kenmore in systems that learn heuristics for handling relative pronoun disambiguation, part-of-speech disambiguation, semantic feature disambiguation, and domain-specific concept disambiguation. In addition, the solutions learned by Kenmore have been incorporated into a working natural language processing system that has demonstrated success in processing real-world text. As such, we have shown the feasibility of truly trainable, portable, customized sentence analyzers.

Although symbolic machine learning algorithms are designed to find regularities in data, it is still somewhat surprising that they can be used to learn solutions to problems in sentence analysis within the context of real-world natural language processing tasks. The natural language learning tasks present a number of challenges for the inductive learning system:

- *The training cases contain noise.* Some of the noise is in the form of inconsistent and/or incorrect class information provided by the human supervisor and some of the noise is in the form of incorrect feature values provided by the sentence analyzer. Because the concepts being learned are open-textured concepts that lack distinct boundaries, some portion of the noise is unavoidable.

- *Kenmore cases contain missing features and irrelevant features.*

- *There is no guarantee that the cases will include all features necessary for the learning task to succeed.* The case representation is limited by the information maintained by the CIRCUS sentence analyzer.

- *Kenmore requires incremental learning of concepts.*

- *The machine learning system must be able to learn concepts with many classes.* The tasks tackled within the MayTag and WHirlpool systems have a minimum of ten class values.

As a second major contribution of the work, we have introduced a new approach to ambiguity resolution in sentence analysis. By viewing all ambiguities as classification problems, Kenmore is able to address uniformly a range of problems in sentence analysis each of which traditionally had required a separate computational mechanism. We believe that it is this uniform view of ambiguity resolution that allows Kenmore to learn solutions to such a wide variety of problems in sentence analysis. In particular Kenmore has handled syntactic, semantic, lexical, and structural ambiguities, and has accommodated both the acquisition and application of disambiguation heuristics within a single architecture.

Kenmore's approach to knowledge acquisition for natural language processing systems has a number of advantages over traditional methods for acquiring the background knowledge needed for sentence analysis. First, Kenmore entirely eliminates the need for generating and maintaining explicit, hand-coded disambiguation heuristics because the case base and case retrieval algorithm together implicitly define a set of disambiguation heuristics. By encoding context as part of a lexical definition, for example, MayTag allows the meaning of a word to change dynamically in response to surrounding phrases without the need for explicit lexical disambiguation heuristics. Applying the framework to solve new problems in sentence analysis is also easy because the same case-based method is used to acquire solutions to syntactic and semantic problems at both the lexical and structural levels. No hand-coded heuristics are required to drive the acquisition process. Builders of traditional NLP systems, on the other hand, design separate methods for each class of problem encountered during sentence analysis.

Compared to purely statistical approaches to knowledge acquisition for natural language systems, Kenmore requires relatively little training. In the experiments described described throughout the thesis, we obtain promising results after training on only a small number of sentences. This makes the method especially appropriate for use with small corpora where statistical approaches fail due to lack of data. In addition, Kenmore's training phase does not require the expertise of computational linguists. With the possible exception of part-of-speech tagging, supervisory information can be provided by anyone who can understand texts from the training corpus. The main constraint for the human supervisor is that the training remain consistent, i.e., the protocol for handling various classes of decisions does not change as training progresses. Another advantage of Kenmore is its incremental learning of disambiguation heuristics, which enables training to become progressively easier for the human supervisor. Kenmore can access the existing case base to suggest solutions to each ambiguity and rely on the supervisor only to override incorrect predictions.

Finally, Kenmore's case-based approach to knowledge acquisition provides a flexible control structure for combining multiple sources of knowledge, making it easy to explore the usefulness of different types of knowledge in solving problems in sentence analysis. An interesting result of our experiments within this case-based framework is that solutions to problems in sentence analysis invariably make only limited use of the available knowledge sources. Without exception, we found that the ambiguity resolution task is better performed when irrelevant pieces of knowledge in the case representation were ignored.

The thesis has also made a number of secondary contributions. We have shown that a decision tree algorithm can be used to find the relevant features in a case representation. A case retrieval algorithm that uses this decision tree approach to feature set selection was shown to perform better than a pure decision tree algorithm, a pure case-based algorithm, and an algorithm that relied on expert knowledge to discard irrelevant features. In addition, we have shown that cognitive biases can improve the performance of the learning algorithm by being automatically incorporated into the case representation.

10.2 Future Directions

The work presented in this thesis can be extended in many directions, some of which have been discussed in earlier chapters, but are summarized again here.

10.2.1 Broader Domains

Kenmore has been presented as a framework for knowledge acquisition within limited domains. An obvious direction that must be explored is how to extend Kenmore to deal with texts from broader, unrestricted domains and, ultimately, with entirely open-ended text. We believe that the use of unrestricted domains will affect Kenmore's ability to perform lexical ambiguity tasks, but will not critically affect its performance on structural ambiguity tasks. The problem is that the taxonomies that underlie lexical acquisition grow tremendously as the domains become broader. In turn, the length of the training phase must be extended in order to gather a representative set of training cases from the corpus. Exceptions to this phenomenon might occur if the natural language task associated with the unrestricted domain required only that high level distinctions be made among lexical items (e.g., if "soldier" and "president" could be tagged as "human" rather than "military" and "official"). Then, the taxonomy would not be expected to grow very much and the current Kenmore framework would be adequate. Our experience with the WHirlpool system, on the other hand, has shown that high level semantic distinctions may be adequate for handling structural ambiguities and there will therefore be less of a problem with an expanding semantic feature taxonomy. Unrestricted domains cause a different problem for structural disambiguation — as the domain becomes broader, the style of the texts becomes increasingly varied and, hence, more training examples will be required. Although exploratory investigations in this area are clearly needed, we believe that the increase in stylistic variations will not extend the training phase prohibitively because they affect the global sentence structure more than local sentence structure and most ambiguities can be resolved using a very limited local context.

One approach to using Kenmore for lexical acquisition with texts from unrestricted domains is to build a collection of knowledge bases for specialized domains, e.g., business joint ventures, terrorism. Then, to understand a text, the NLP system first determines the domain of the text and then accesses the knowledge base associated with that domain. A better approach to dealing with unrestricted domains in Kenmore was discussed in Chapter 5.5.4. There we proposed that Kenmore begin with a general, on-line knowledge base (e.g., an online dictionary or thesaurus) and use its case-based approach to decide which of the definitions (e.g., parts of speech, semantic tags) listed for a word is appropriate in a given context. Each case in the case base includes the usual context and solution portions, but the context part could be expanded to include a description of the options available for the current word in the on-line dictionary. The solution part of the case specifies the definition that is correct in the current context or, if none were correct, describes the novel definition. This approach allows system developers to start with a predefined taxonomy of syntactic and semantic types rather than designing them from scratch.

10.2.2 Unsupervised Training Methods

The current major bottleneck in the Kenmore system is the human-supervised training phase and, as described above, this bottleneck will become even worse as Kenmore is used in increasingly broad domains. As a result, future work will explore the use of unsupervised inductive learning techniques rather than supervised ones. If we assume that Kenmore has already been trained to handle problems in lexical ambiguity, (e.g., part-of-speech ambiguity, semantic feature ambiguity), then a minimally supervised approach that may work well for problems in structural ambiguity is the following. During the training phase, Kenmore should create cases that contain only their context portions. Because the parser supplies all of the needed information automatically, no human supervisor is needed. Next, the cases are clustered using any of a number of clustering techniques. Finally, a human supervisor is consulted to provide supervisory information for one case in each cluster and all cases in the cluster are assigned the same class. At this point, all training cases have both context and solution portions and Kenmore can be run in application mode using the usual case retrieval methods to find the training case most similar to the problem case.

The above approach will be less useful for lexical tagging tasks. The context portion of each lexical disambiguation case will only be able to include the individual words that precede and follow the current word. None of the higher level attributes like the part of speech or semantic features of the preceding or following words or any constituent level information will be available because that is the information that the system is trying to learn. Without this higher level information as part of the context vector, **many** more cases will be needed to obtain reasonable clusters. As a result, there will be many more clusters for the human supervisor to "seed" with supervisory information.

10.2.3 Smarter Supervised Training

We can also explore methods that reduce the amount of training required for purely supervised learning methods. Lewis and Gale [1994] present one such approach that was found to reduce the number of manually tagged training cases by 500-fold for a given level of performance. The approach is based on results from computational learning theory and involves making a better choice of training examples by training on the most difficult examples. It could be incorporated into the Kenmore framework as follows. First, generate a (fairly large) base set of cases that contain only the context portion of the case. We will refer to these as the "context-only cases." Because no supervisory information is required, the context-only cases can be generated automatically by the sentence analyzer — no human intervention is required. Then set up a small case base with training cases for which the human supervisor has provided class information. Apply the case retrieval algorithm to the set of context-only cases and locate the case whose nearest-neighbor had the lowest similarity score. This will be the context-only case for which the case retrieval algorithm is least certain of its class. Next, let the human supervisor supply class information for this case and add it to the case base. Continue this process until the cross-validation testing indicates that an adequate level of performance has been attained or improvements in performance have ceased.

Another way to improve Kenmore's training phase is to make more direct use of the errors made during training. Because Kenmore knows which training case(s) are responsible for each of its decisions, we might explore automated approaches for using this credit assignment information to modify the similarity metric or the case representation (see Section 10.2.5 below), or to add and remove cases from the case base.

10.2.4 Problems That Cross Clause and Sentence Boundaries

In addition to applying Kenmore to additional structural and lexical problems within a single clause, it will also be interesting to apply Kenmore to problems that span clauses or even sentences. Some problems include pronoun resolution, general anaphora problems, noun and verb phrase ellipsis, and understanding infinitive complements. Some problems may require very different case representations than we have seen thus far because their scope is less local in nature. In addition, these problems may require different approaches to improving the case representation.

10.2.5 Dynamic Modification of Case Representation

This thesis presented two approaches for improving a baseline case representation automatically. However, additional machine learning methods for modifying a case representation should be explored. As discussed in Chapter 9, it will be especially important to devise methods that systematically add new features to a representation by exploiting the structure and meaning of the representations maintained and created by the sentence analyzer.

10.2.6 Environment for Natural Language System Development

We also plan to continue to explore the use of symbolic inductive machine learning techniques as tools for guiding natural language system development. This will include extending Kenmore into an integrated environment in which natural language systems can be designed, built, and evaluated, and in which the role that various knowledge sources play in the conceptual analysis of text can be explored. All of the extensions and modifications described in the sections above should be incorporated into this environment for system development.

10.2.7 Higher Level Knowledge Structures

The goal of this thesis is to address the knowledge engineering bottleneck for natural language processing systems. We have presented Kenmore as a framework in which the knowledge required by an NLP system can be systematically acquired. However, the work described in this thesis just begins to explore the role that machine learning techniques can play in the larger knowledge-bootstrapping process required for increasingly deeper analysis of increasingly complex natural language understanding tasks. An important research direction to pursue will be to test the ability of the framework for the acquisition of some of the higher level knowledge structures that conceptual analysis of text requires. These include the acquisition of scripts, plans, causal knowledge, and mental models. As a side effect, we can begin to use Kenmore to acquire knowledge structures from text for use by other AI systems and to gain insight into the possibility of using raw text directly as a knowledge base.

BIBLIOGRAPHY

[Aha and Kibler, 1987] Aha, D. and Kibler, D. Learning Representative Exemplars of Concepts: An Initial Case Study. In *Proceedings of the Fourth International Conference on Machine Learning*, pages 24–30, U. C. Irvine, Irvine, CA, 1987. Morgan Kaufmann.

[Aha *et al.*, 1991] Aha, D., Kibler, D., and Albert, M. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.

[Aha, 1989] Aha, D. Instance-Based Learning Algorithms. In *Proceedings of the Sixth International Conference on Machine Learning*, pages 387–391, Cornell University, Ithaca, NY, 1989. Morgan Kaufmann.

[Almuallim and Dietterich, 1991] Almuallim, H. and Dietterich, T. G. Learning With Many Irrelevant Features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552, Anaheim, CA, 1991. AAAI Press / MIT Press.

[Ashley and Rissland, 1988] Ashley, K. D. and Rissland, E. L. A cased-based approach to modeling legal expertise. *IEEE Expert*, 3(3):70–77, 1988.

[Ashley, 1990] Ashley, K. D. *Modelling legal argument: Reasoning with cases and hypotheticals*. MIT Press, Bradford Books, Cambridge, MA, 1990.

[Ayuso *et al.*, 1987] Ayuso, D. M., Shaked, V., and Weischedel, R. M. An environment for acquiring semantic information. In *Proceedings of the 25th Annual Meeting of the ACL*, pages 32–40. Association for Computational Linguistics, 1987.

[Bareiss *et al.*, 1989] Bareiss, E. R., Porter, B. W., and Murray, K. S. Supporting start-to-finish development of knowledge bases. *Machine Learning*, 4:261–285, 1989.

[Bareiss, 1989] Bareiss, E. R. *Exemplar-based knowledge acquisition: a unified approach to concept representation, classification, and learning*. Academic Press, Boston, MA, 1989.

[Berwick, 1983] Berwick, Robert. Learning word meanings from examples. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 459–461, Karlsrsuhe, Germany, 1983.

[Birnbaum and Selfridge, 1981] Birnbaum, L. and Selfridge, M. Conceptual Analysis of Natural Language. In Schank, R. and Riesbeck, C., editors, *Inside Computer Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Bosch and Daelemans, 1993] Bosch, A. van den and Daelemans, W. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of European Chapter of ACL*, pages 45–53, Utrecht, 1993. Also available as ITK Research Report 42.

[Bozsahin and Findler, 1992] Bozsahin, H. Cem and Findler, V. Nicholas. Memory-Based Hypothesis Formation: Heuristic Learning of Commonsense Causal Relations from Text. *Cognitive Science*, 16(4):431–454, 1992.

[Branting and Porter, 1991] Branting, L. K. and Porter, B. W. Rules and Precedents as Complementary Warrants. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 3–9, Anaheim, CA, 1991. AAAI Press / MIT Press.

[Branting, 1991] Branting, L. K. *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*. PhD thesis, University of Texas, Austin, TX, 1991.

[Brent, 1990] Brent, Michael R. Semantic Classification of Verbs from their Syntactic Contexts: Automated Lexicography with Implications for Child Language Acquisition. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 428–437, Massachusetts Institute of Technology, Cambridge, MA, 1990.

[Brent, 1991] Brent, Michael R. Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 209–214, University of California, Berkeley, CA, 1991. Association for Computational Linguistics. Also in Computational Linguistics, Vol. 19, No. 2.

[Bresnan, 1982] Bresnan, J. *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, 1982.

[Brill, 1992] Brill, E. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*. Association for Computational Linguistics, 1992.

[Brill, 1993] Brill, E. *A Corpus-Based Approach to Language Learning*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1993.

[Brill, 1994] Brill, E. Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722–727. AAAI Press / MIT Press, 1994.

[Brown *et al.*, 1991] Brown, P., Della Pietra, S., Della Pietra, V., and Mercer, R. Word Sense Disambiguation Using Statistical Methods. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 264–270, University of California, Berkeley, CA, 1991. Association for Computational Linguistics.

[Brown *et al.*, 1992] Brown, P., Della Pietra, V. J., deSouze, P. V., Lai, J. C., and Mercer, R. L. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4), 1992.

[Callan and Utgoff, 1991] Callan, J. and Utgoff, P. A Transformational Approach to Constructive Induction. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 122–126, Northwestern University, Chicago, IL, 1991. Morgan Kaufmann.

[Carbonell, 1978] Carbonell, J. Politics: Automated ideological reasoning. *Cognitive Science*, 2(1):27–51, 1978.

[Cardie and Lehnert, 1991] Cardie, C. and Lehnert, W. A Cognitively Plausible Approach to Understanding Complicated Syntax. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 117–124, Anaheim, CA, 1991. AAAI Press / MIT Press.

[Cardie, 1992a] Cardie, C. Corpus-Based Acquisition of Relative Pronoun Disambiguation Heuristics. In *Proceedings of the 30th Annual Meeting of the ACL*, pages 216–223, University of Delaware, Newark, DE, 1992. Association for Computational Linguistics.

[Cardie, 1992b] Cardie, C. Learning to Disambiguate Relative Pronouns. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 38–43, San Jose, CA, 1992. AAAI Press / MIT Press.

[Cardie, 1992c] Cardie, C. Using Cognitive Biases to Guide Feature Set Selection. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 743–748, Indiana Univeristy, Bloomington, IN, 1992. Lawrence Erlbaum Associates. Also in Working Notes of the 1992 AAAI Workshop on Constraining Learning with Prior Knowledge, San Jose, CA.

[Cardie, 1993a] Cardie, C. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 798–803, Washington, DC, 1993. AAAI Press / MIT Press.

[Cardie, 1993b] Cardie, C. Using Decision Trees to Improve Case-Based Learning. In Utgoff, P., editor, *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32, University of Massachusetts, Amherst, MA, 1993. Morgan Kaufmann.

[Charniak, 1993] Charniak, E. Equations for Part-of-Speech Tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, Washington, DC, 1993. AAAI Press / MIT Press.

[Chen *et al.*, 1993] Chen, T., Soo, V., and Lin, A. Learning to Parse with Recurrent Neural Networks. In *Proceedings of European Conference on Machine Learning Workshop on Machine Learning and Text Analysis*, pages 63–68, 1993.

[Chinchor *et al.*, 1993] Chinchor, N., Hirschman, L., and Lewis, D. Evaluating Message Understanding Systems: An Analysis of the Third Message Undestanding Conference (MUC-3). *Computational Linguistics*, 19(3):409–449, 1993.

[Church and Hanks, 1990] Church, K. and Hanks, P. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16, 1990.

[Church and Mercer, 1993] Church, K. and Mercer, R. Introduction to the Special Issue on Computational Linguistics Introduction to the Special Issue on Computational Linguistics Using Large Corpora. *Computational Linguistics*, 19:1–24, 1993.

[Church *et al.*, 1991] Church, K., Gale, W., Hanks, P., and Hindle, D. Using Statistics in Lexical Analysis. In Zernik, U., editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 115–164. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[Church, 1988] Church, K. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143. Association for Computational Linguistics, 1988.

[Cognitive Systems, 1992] Cognitive Systems, Inc. *ReMind: Release Notes Version 1.1*. Boston, MA, 1992.

[Correa, 1988] Correa, N.. A Binding Rule for Government-Binding Parsing. In *Proceedings of COLING-88*, Budapest, 1988.

[Cottrell, 1986] Cottrell, G. *A Connectionist Approach to Word Sense Disambiguation*. Morgan Kaufmann, 1986.

[Cuetos and Mitchell, 1988] Cuetos, F. and Mitchell, D. C. Cross-Linguistic Differences in Parsing: Restrictions on the Use of the Late Closure Strategy in Spanish. *Cognition*, 30(1):73–105, 1988.

[Cullingford, 1986] Cullingford, R. *Natural Language Processing*. Rowman and Littlefield, Totowa, NJ, 1986.

[Daelemans *et al.*, 1994] Daelemans, W., Durieux, G., and Gillis, S. The Acquisition of Stress: A Data-Oriented Approach. *Computational Linguistics*, 20(3):421–451, 1994.

[Daelemans, to appear] Daelemans, W. Memory-Based Lexical Acquisition and Processing. In *Lecture Notes in Artificial Intelligence, Machine Translation and the Lexicon*. to appear.

[Dagan and Itai, 1991] Dagan, I. and Itai, A. A Statistical Filter for Resolving Pronoun References. In Feldman, Y. A. and Bruckstein, A., editors, *Artificial Intelligence and Computer Vision*, pages 125–135. Elsevier Science Publishers, North Holland, 1991.

[Daneman and Carpenter, 1980] Daneman, M. and Carpenter, P. A. Individual Differences in Working Memory and Reading. *Journal of Verbal Learning and Verbal Behavior*, 19:450–466, 1980.

[Daneman and Carpenter, 1983] Daneman, M. and Carpenter, P. A. Individual Differences in Integrating Information Between and Within Sentences. *Journal of Experimental Psychology: Learning, Memory,and Cognition*, 9:561–584, 1983.

[DeJong, 1979] DeJong, G. F. *Skimming Stories in Real Time: An Experiment in Integrated Understanding*. PhD thesis, Yale University, 1979. Also available as Tech Report YALEU/CSD/RR #158.

[Delannoy *et al.*, 1993] Delannoy, J. F., Feng, C., and Matwin, S.and Szpakowicz S. Knowledge Extraction from Text: Machine Learning for Text-to-rule Translation. In *Proceedings of European Conference on Machine Learning Workshop on Machine Learning and Text Analysis*, pages 7–13, 1993.

[DeMarcken, 1990] DeMarcken, C. Parsing the LOB Corpus. In *Proceedings of the 28th Annual Meeting of the ACL*. Association for Computational Linguistics, 1990.

[Dyer, 1983] Dyer, M. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. MIT Press, Cambridge, MA, 1983.

[Elman, 1990] Elman, J. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[Fawcett and Utgoff, 1992] Fawcett, T. and Utgoff, P. Automatic feature Generation for Problem Solving Systems. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 144–153, University of Aberdeen, UK, 1992. Morgan Kaufmann.

[Fisher and Riloff, 1992] Fisher, D. and Riloff, E. Applying Statistical Methods to Small Corpora: Benefitting from a Limited Domain. In *Working Notes of AAAI Fall Symposium Series*, pages 47–53, Cambride, MA, 1992. AAAI Press.

[Frazier and Fodor, 1978] Frazier, L. and Fodor, J. D. The Sausage Machine: A New Two-Stage Parsing Model. *Cognition*, 6:291–325, 1978.

[Gernsbacher *et al.*, 1989] Gernsbacher, M. A., Hargreaves, D. J., and Beeman, M. Building and Accessing Clausal Representations: The Advantage of First Mention Versus the Advantage of Clause Recency. *Journal of Memory and Language*, 28:735–755, 1989.

[Gibson *et al.*, 1993] Gibson, E., Pearlmutter, N., Canseco-Gonzalez, E., and Hickok, G. Cross-linguistic Attachment Preferences: Evidence from English and Spanish. In *Sixth Annual CUNY Sentence Processing Conference*, University of Massachusetts, Amherst, MA, 1993. Only abstract in the Sentence Processing Conference proceedings. Full manuscript to appear in journal.

[Gibson, 1990] Gibson, E. Recency Preferences and Garden-Path Effects. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Massachusetts Institute of Technology, Cambridge, MA, 1990. Lawrence Erlbaum Associates.

[Golding and Rosenbloom, 1991] Golding, A. R. and Rosenbloom, P. S. Improving Rule-Based Systems through Case-Based Reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 22–27, Anaheim, CA, 1991. AAAI Press / MIT Press.

[Golding, 1991] Golding, A. R. *Pronouncing Names by a Combination of Case-Based and Rule-Based Reasoning*. PhD thesis, Stanford University, 1991.

[Goodman, 1991] Goodman, M. A Case-Based, Inductive Architecture for Natural Language Processing. In *Working Notes of the Machine Learning of Natural Language and Ontology AAAI Spring Symposium*, Stanford University, March 1991.

[Granger, 1977] Granger, R. Foulup: A program that figures out meanings of words from context. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 172–178. Morgan Kaufmann, 1977.

[Grefenstette, 1992] Grefenstette, G. SEXTANT: Exploring unexplored contexts for semantic extraction from syntactic analysis. In *Proceedings of the 30th Annual Meeting of the ACL*, pages 324–326, University of Delaware, Newark, DE, 1992. Association for Computational Linguistics.

[Grishman *et al.*, 1986] Grishman, R., Hirschman, L., and Nhan, N. T. Discovery procedures for sublanguage selectional patterns: Initial experiments. *Computational Linguistics*, 12:205–215, 1986.

[Grosz and Sidner, 1986] Grosz, B. J. and Sidner, C.L. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[Hammond, 1989] Hammond, K. J. *Case-based planning: Viewing planning as a memory task*. Academic Press, Boston, 1989.

[Hart, 1961] Hart, H. L. A. *The Concept of Law*. Oxford University Press, Oxford, 1961.

[Hartigan, 1975] Hartigan, J. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

[Hastings *et al.*, 1991] Hastings, P., Lytinen, S., and Lindsay, R. Learning Words from Context. In *Proceedings of the Eighth International Workshop on Machine Learning*, Northwestern University, Chicago, IL, 1991.

[Hindle and Rooth, 1993] Hindle, D. and Rooth, M. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19(1):103–120, 1993.

[Hobbs, 1986] Hobbs, J. Resolving Pronoun References. In Grosz, B., Sparck Jones, K., and Webber, B., editors, *Readings in Natural Language Processing*, pages 339–352. Morgan Kaufmann, Los Altos, CA, 1986.

[Ingria and Stallard, 1989] Ingria, R. and Stallard, D. A computational mechanism for pronominal reference. In *Proceedings of the 27th Annual Meeting of the ACL*, Vancouver, 1989. Association for Computational Linguistics.

[Jacobs and Zernik, 1988] Jacobs, P. and Zernik, U. Acquiring Lexical Knowledge from Text: A Case Study. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 739–744, St. Paul, MN, 1988. Morgan Kaufmann.

[Jain, 1989] Jain, Ajay. A Connectionist Architecture for Sequential Symbolic Domains . Technical Report CMU-CS-89-187, Carnegie Mellon University, 1989.

[Jelinek, 1985] Jelinek, F. Markov Source Modeling of Text Generation. In Skwirzinski, J., editor, *Impact of Processing Techniques on Communication*. Dordrecht, 1985.

[Keil and Kelly, 1987] Keil, Frank C. and Kelly, Michael H. Developmental changes in category structure. In Harnad, S., editor, *Categorical Perception: The Groundwork of Cognition*, pages 491–510. Cambridge University Press, Cambridge, 1987.

[Kimball, 1973] Kimball, J. Seven Principles of Surface Structure Parsing in Natural Language. *Cognition*, 2:15–47, 1973.

[King and Just, 1991] King, J. and Just, M. A. Individual Differences in Syntactic Processing: The Role of Working Memory. *Journal of Memory and Language*, 30:580–602, 1991.

[Kolodner, 1983] Kolodner, J. L. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7(4):243–280, 1983.

[Kolodner, 1993] Kolodner, J. *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA, 1993.

[Laird, 1987] Laird, J.E. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[Lakoff, 1987] Lakoff, G. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. The University of Chicago Press, Chicago, IL, 1987.

[Lange and Dyer, 1989] Lange, T. and Dyer, M. High-Level Inferencing in a Connectionist Network. *Connection Science*, 4(2), 1989.

[Lappin and McCord, 1990] Lappin, S and McCord, M. A syntactic filter on pronominal anaphora for slot grammar. In *Proceedings of the 28th Annual Meeting of the ACL*, University of Pittsburgh, Pittsburgh, PA, 1990. Association for Computational Linguistics.

[Lebowitz, 1980] Lebowitz, M. *Generalization and Memory in an Integrated Understanding System*. PhD thesis, Yale University, 1980. Also available as Tech Report YALEU/CSD/RR #186.

[Lehman *et al.*, 1993] Lehman, J., Lewis, R., and Newell, A. Integrating Knowledge Sources in Language Comprehension. In Rosenbloom, P., Laird, J., and Newell, A., editors, *The SOAR Papers: Research on Integrated Intelligence*, volume 2, pages 1309–1314. The MIT Press, Cambridge, MA, 1993.

[Lehnert *et al.*, 1990] Lehnert, W., Cardie, C., and Riloff, E. Analyzing Research Papers Using Citation Sentences. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 511–518, Cambridge, MA, July 1990. Lawrence Erlbaum Associates.

[Lehnert *et al.*, 1991] Lehnert, W., Cardie, C., Fisher, D., Riloff, E., and Williams, R. University of Massachusetts: Description of the CIRCUS System as Used in MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pages 223–233, San Mateo, CA, 1991. Morgan Kaufmann.

[Lehnert *et al.*, 1992] Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., and Soderland, S. University of Massachusetts: Description of the CIRCUS System as Used in MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 282–288, San Mateo, CA, 1992. Morgan Kaufmann.

[Lehnert *et al.*, 1993a] Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., and Goldman, S. University of Massachusetts/Hughes: Description of the CIRCUS System as Used for TIPSTER Text. In *Proceedings, TIPSTER Text Program (Phase I)*, pages 241–256, San Mateo, CA, 1993. Morgan Kaufmann.

[Lehnert *et al.*, 1993b] Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C., and Goldman, S. University of Massachusetts/Hughes: Description of the CIRCUS System as Used in MUC-5. In *Proceedings of the Fourth Message Understanding Conference (MUC-5)*, pages 277–291, San Mateo, CA, 1993. Morgan Kaufmann.

[Lehnert *et al.*, 1994] Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., and Soderland, S. Evaluating an Information Extraction System. *Journal of Integrated Computer-Aided Engineering*, 1(6), 1994.

[Lehnert, 1978] Lehnert, W. *The Process of Question Answering*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[Lehnert, 1990] Lehnert, W. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In Barnden, J. and Pollack, J., editors, *Advances in Connectionist and Neural Computation Theory*, pages 135–164. Ablex Publishers, Norwood, NJ, 1990.

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of ACM SIGIR*, Dublin, Ireland, 1994.

[Lytinen and Roberts, 1989] Lytinen, S. and Roberts, S. Lexical Acquisition as a By-Product of Natural Language Processing. In *Proceedings, IJCAI-89 Workshop on Lexical Acquisition*, 1989.

[Lytinen, 1984] Lytinen, S. *The Organization of Knowledge in a Multi-lingual, Integrated Parser*. PhD thesis, Yale University, 1984. Also available as Tech Report YALEU/CSD/RR #340.

[Marcus *et al.*, 1993] Marcus, M., Marcinkiewicz, M., and Santorini, B. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Marcus, 1991] Marcus, Mitchell. The Automatic Acquisition of Linguistic Structure from Large Corpora: An Overview of Work at the University of Pennsylvania. In *Working Notes of the Machine Learning of Natural Language and Ontology AAAI Spring Symposium*, Stanford University, March 1991.

[Martin, 1992] Martin, J. Computer Understanding of Conventional Metaphoric Language. *Cognitive Science*, 16(2):233–270, 1992.

[Matwin and Szpakowicz, 1993] Matwin, S. and Szpakowicz, S. Text Analysis: How Can Machine Learning Help? In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 33–42, Vancouver, BC, 1993.

[McClelland and Kawamoto, 1986] McClelland, J. L. and Kawamoto, A. H. Mechanisms of sentence processing: Assigning roles to constituents of sentences. In McClelland, J. L., Rumelhart, D.E., and PDP Research Group, the, editors, *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 2. Psychological and biological models*. MIT Press, Cambridge, MA, 1986.

[Miikkulainen and Dyer, 1987] Miikkulainen, R. and Dyer, M. Building Distributed Representations without Microfeatures. Technical Report TR UCLA-AI-87-17, UCLA, 1987.

[Miikkulainen and Dyer, 1989] Miikkulainen, R. and Dyer, M. A Modular Neural Network Architecture for Sequential Paraphrasing of Script-Based Stories. Technical Report UCLA-AI-89-02, UCLA, 1989.

[Miller, 1956] Miller, G. A. The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. *Psychological Review*, 63(1):81–97, 1956.

[Muggleton, 1992] Muggleton, S. H. *Inductive Logic Programming*. Academic Press, 1992.

[Newell, 1990] Newell, A. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.

[Newport, 1990] Newport, E. Maturational Constraints on Language Learning. *Cognitive Science*, 14:11–28, 1990.

[Nicol, 1988] Nicol, J. *Coreference Processing During Sentence Comprehension*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[Pazzani, 1991] Pazzani, M. J. A computational theory of learning causal relationships. *Cognitive Science*, 15:401–424, 1991.

[Quinlan, 1983] Quinlan, J. R. Learning Efficient Classification Procedures and Their Application to Chess End Games. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, San Mateo, CA, 1983.

[Quinlan, 1986] Quinlan, J. R. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Quinlan, 1990] Quinlan, J. R. Learning Logical Definitions from Relations. *Machine Learning*, 5:239–266, 1990.

[Quinlan, 1992] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.

[Resnik and Hearst, 1993] Resnik, P. and Hearst, M. Structural Ambiguity and Conceptual Relations. In *Proceedings of the ACL Workshop on Very Large Corpora*, Ohio State, 1993. Association for Computational Linguistics.

[Resnik, 1993] Resnik, P. *Selection and Information:A Class-Based Approach to Lexical Relationships*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1993.

[Riesbeck and Martin, 1985] Riesbeck, C. and Martin, C. Direct Memory Access Parsing. In Riesbeck, C. and Kolodner, J., editors, *Experience, Memory and Reasoning*. Lawrence Erlbaum, Hillsdale, NJ, 1985.

[Riesbeck and Schank, 1978] Riesbeck, C. and Schank, R. comprehension by computer: Expectation-base analysis of sentences in context. In Levelt, W. J. M. and Flores d'Arcais, G.B., editors, *Studies in the Perception of Language*. John Wiley & Sons, New York, 1978.

[Riesbeck and Schank, 1989] Riesbeck, C. and Schank, R. *Inside Case-Based Reasoning*. Erlbaum, Northvale, NJ, 1989.

[Riesbeck, 1975] Riesbeck, C. Conceptual Analysis. In Schank, R., editor, *Conceptual Information Processing*. North Holland, Amsterdam, 1975.

[Riloff and Lehnert, 1992] Riloff, E. and Lehnert, W. Classifying Texts Using Relevancy Signatures. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 329–334, San Jose, CA, 1992. AAAI Press / MIT Press.

[Riloff, 1993] Riloff, E. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816, Washington, DC, 1993. AAAI Press / MIT Press.

[Rissland and Ashley, 1986] Rissland, E. L. and Ashley, K. Hypotheticals as Heuristic Device. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 289–297, Philadelphia, PA, 1986. Morgan Kaufmann.

[Rissland and Skalak, 1991] Rissland, E. L. and Skalak, D. B. CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 34:839–887, 1991.

[Rissland *et al.*, 1984] Rissland, E. L., Valcarce, E. M., and Ashley, K. D. Explaining and Arguing with Examples. In *AAAI-84, Proceedings of the National Conference on Artificial Intelligence*, pages 288–294, Austin, TX, 1984. American Association for Artificial Intelligence.

[Rissland, 1990] Rissland, E. L. Artificial Intelligence and Law: Stepping Stones to a Model of Legal Reasoning. *The Yale Law Journal*, 99(8):1957–1981, 1990.

[Rosch and Mervis, 1975] Rosch, E. and Mervis, C. Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology*, 7:573–605, 1975.

[Rosch, 1973] Rosch, E. Natural Categories. *Cognitive Psychology*, 4:328–350, 1973.

[Samuelsson and Rayner, 1991] Samuelsson, C. and Rayner, M. Proceedings of the AAAI Spring Symposium on Machine Learning of Natural Language and Ontology. In *Working Notes of the Machine Learning of Natural Language and Ontology AAAI Spring Symposium*, pages 143–145, Stanford University, 1991. DFKI:Kaiserslautern,FRG. Check for more recent ref.

[Samuelsson, 1991] Samuelsson, C. *Using Explanation-Based Learning to Speed Up Natural Language Systems*. PhD thesis, Royal Institute of Technology, Stockholm, 1991.

[Schank and Riesbeck, 1981] Schank, R. and Riesbeck, C. *Inside Computer Understanding: Five Programs Plus Miniatures*. Lawrence Erlbaum, Hillsdale, NJ, 1981.

[Schank, 1975] Schank, R. *Conceptual Information Processing*. North Holland, Amsterdam, 1975.

[Schank, 1982] Schank, R. *Dynamic Memory: A theory of learning in computers and people*. Cambridge University Press, New York, 1982.

[Selfridge, 1986] Selfridge, M. A computer model of child language learning. *Artificial Intelligence*, 29:171–216, 1986.

[Sidner, 1983] Sidner, C. L. Focusing in the Comprehension of Definite Anaphora. In Brady, M. and Berwick, R. C., editors, *Computational Models of Discourse*, pages 267–330. Massachusetts Institute of Technology, Cambridge, MA, 1983.

[Simmons and Yu, 1992] Simmons, Robert F. and Yu, Yeong-Ho. The Acquisition and Use of Context-Dependent Grammars for English. *Computational Linguistics*, 18(4):391–418, 1992.

[Skalak and Rissland, 1990] Skalak, D. and Rissland, E. Inductive Learning in a Mixed Paradigm Setting. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 840–847, Boston, MA, 1990. AAAI Press / MIT Press.

[St. John, 1992] St. John, Mark F. The Story Gestalt: A Model of Knowledge-Intensive Processes in Text Comprehension. *Cognitive Science*, 16(2):271–306, 1992.

[Stanfill and Waltz, 1986] Stanfill, C. and Waltz, D. Toward Memory-based Reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

[Stetham, 1991] Stetham, S. Proceedings of the AAAI Spring Symposium on Machine Learning of Natural Language and Ontology. In *Working Notes of the Machine Learning of Natural Language and Ontology AAAI Spring Symposium*, pages 169–173, Stanford University, 1991. DFKI:Kaiserslautern,FRG.

[Utgoff, 1989] Utgoff, P. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[Utgoff, 1994] Utgoff, P. An Improved Algorithm for Incremental Induction of Decision Trees. In Cohen, W. and Hirsh, H., editors, *Proceedings of the Eleventh International Conference on Machine Learning*, pages 318–325, Rutgers University, New Brunswick, NJ, 1994. Morgan Kaufmann.

[Waltz and Pollack, 1985] Waltz, D. and Pollack, J. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science*, 9(1):51–74, 1985.

[Weischedel *et al.*, 1993] Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., and Palmucci, J. Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359–382, 1993.

[Wermter and Lehnert, 1989] Wermter, S. and Lehnert, W. A hybrid symbolic/connectionist model for noun-phrase understanding. *Connection Science*, 1(3), 1989.

[Wermter, 1990] Wermter, S. Combining Symbolic and Connectionist Techniques for Coordination in Natural Language. In *Proceedings of the 14th German Workshop on Artificial Intelligence*, Eringerfeld, Germany, 1990.

[Wilensky, 1978] Wilensky, R. *Understanding goal-based stories*. PhD thesis, Yale University, 1978. Also available as Tech Report YALEU/CSD/RR #140.

[Wilensky, 1991] Wilensky, R. Extending the Lexicon by Exploiting Subregularities. Technical Report UCB/CSD 91/618, Computer Science Division (EECS), University of California, Berkeley, 1991.

[Winograd, 1983] Winograd, T., editor. *Language as a Cognitive Process*, volume 1. Addison-Wesley Publishing Company, Reading, MA, 1983.

[Wittgenstein, 1953] Wittgenstein, L. *Philosophical Investigations*. Macmillan, New York, 1953.

[Woods, 1970] Woods, W. Transition network grammars for natural language analysis. *CACM*, 13(10), 1970.

[Yarowsky, 1992] Yarowsky, David. Word-Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. In *Proceedings, COLING-92*, 1992.

[Yarowsky, 1994] Yarowsky, David. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proceedings of the 32th Annual Meeting of the ACL*, 1994.

[Zelle and Mooney, 1993] Zelle, J. and Mooney, R. Learning Semantic Grammars with Constructive Inductive Logic Programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 817–822, Washington, DC, 1993. AAAI Press / MIT Press.

[Zelle and Mooney, 1994] Zelle, J. and Mooney, R. Learning Semantic Grammars with Constructive Inductive Logic Programming. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 748–753, Seattle, WA, 1994. AAAI Press / MIT Press.

[Zernik, 1991] Zernik, U. Introduction. In Zernik, U., editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 1–26. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.