Semantic Lexicon Acquisition for Learning Natural Language Interfaces

Cynthia Ann Thompson

Report AI99-278 December 1998

cthomp@csli.stanford.edu
http://www-csli.stanford.edu/~cthomp/
Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

Semantic Lexicon Acquisition for Learning Natural Language Interfaces

by

Cynthia Ann Thompson, B.S.,M.A.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 1998

Semantic Lexicon Acquisition for Learning Natural Language Interfaces

Approved by
Dissertation Committee:

Raymond Mooney

Claire Cardie

Benjamin Kuipers

Risto Miikkulainen

Elaine Rich

To my parents who gave me my wings, and to Bill who gave me the wind

Acknowledgments

I did not complete this dissertation in isolation. Many people directly and indirectly contributed to the ideas contained within; many also contributed to the sanity and financial well-being of the author. If I have forgotten anyone, please do not be offended, for you are all in my heart, even if you are not in my head at the moment that I am writing this. First, I would like to thank my advisor, Ray Mooney, who taught me much about how to be a good academic researcher. His advice and input were invaluable in assisting me with the research for this dissertation. I must express my gratitude to Ben Kuipers, who acted much as an unofficial backup advisor. I enjoyed our conversations about life, the universe, and everything. Thanks also to my other committee members: Claire Cardie, Risto Miikkulainen, and Elaine Rich.

Robert Rodman was instrumental in encouraging me to go to graduate school in the first place, for which I thank him. Anita Borg and her work with `systers` helped keep me going once I got here. Many others had a positive influence on this research and on my life while I was at the University of Texas. Special mention goes to my office mates who helped me survive: Paul Baffes, Mary Elaine Califf, Harold Chaput, Dan Clancy, Tara Estlin, Bert Kay, Jeff Mahoney, Emilio Remolina, Rupert Tang, Tal Tversky, and especially John Zelle for discussions on our research, Mike Hewett for the chocolate, and Sowmya Ramachandran for being my best (female) friend.

Thanks to my translators: Agapito Sustaita, Esra Erdem, and Marty Mayberry. I am grateful to many others, including Chris Edmondson-Yurkanan, Vicki Almstrum, Ted Pedersen, and Mike Cline. Thanks to the Austin RC community, for reminding me of my complete brilliance, especially Ruth. Then there are the many friends who reminded me to also have fun: Lyn Pierce (for lunches and fun), Debbie Miller (for thoughts and games), Dave Pierce (for being a great brother-in-law), Michael Bogomolny (for haiku and lunches), and Todd and Mary West (for showing me the way).

I would like to thank my family, without whom nothing is possible. My parents were always a great support, even though they worried too much. My brothers didn't quite get why I was doing this to myself, but thought it was cool anyway. My family-in-law welcomed me with open arms and tons of confidence. Finally, no amount of thanks is enough for Bill, for being my best (male) friend, for helping me keep touch with reality, and just for being you.

Thanks go to Jeff Siskind for permission to use his software and for his help in getting it working with our data. Portions of this research were supported by the National Science Foundation under grant IRI-9310819 and through a grant from the Deimler-Benz Palo Alto Research Center. The author was also partially supported by a scholarship from Trident.

Cynthia Ann Thompson

The University of Texas at Austin
December 1998

Semantic Lexicon Acquisition for Learning Natural Language Interfaces

Technical Report AI98-269

Cynthia Ann Thompson, Ph.D.
The University of Texas at Austin, 1998

Supervisor: Raymond Mooney

A long-standing goal for the field of artificial intelligence is to enable computer understanding of human languages. A core requirement in reaching this goal is the ability to transform individual sentences into a form better suited for computer manipulation. This ability, called semantic parsing, requires several knowledge sources, such as a grammar, lexicon, and parsing mechanism.

Building natural language parsing systems by hand is a tedious, error-prone undertaking. We build on previous research in automating the construction of such systems using machine learning techniques. The result is a combined system that learns semantic lexicons and semantic parsers from one common set of training examples. The input required is a corpus of sentence/representation pairs, where the representations are in the output format desired. A new system, WOLFIE, learns semantic lexicons to be used as background knowledge by a previously developed parser acquisition system, CHILL. The combined system is tested on a real world domain of answering database queries. We also compare this combination to a combination of CHILL with a previously developed lexicon learner, demonstrating superior performance with our system. In addition, we show the ability of the system to learn to process natural languages other than English. Finally, we test the system on an alternate sentence representation, and on a set of large, artificial corpora with varying levels of ambiguity and synonymy.

One difficulty in using machine learning methods for building natural language interfaces is building the required annotated corpus. Therefore, we also address this issue by using *active learning* to reduce the number of training examples required by both WOLFIE and CHILL. Experimental results show that the number of examples needed to reach a given level of performance can be significantly reduced with this method.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A long-standing goal for the field of artificial intelligence is to enable computer understanding of human languages. Much progress has been made in reaching this goal, but much also remains to be done. Before artificial intelligence systems can meet this goal, they first need the ability to *parse* sentences, or transform them into a representation that can be more easily manipulated by computers. Several knowledge sources are required for parsing, such as a grammar, lexicon, and parsing mechanism.

Natural language processing (NLP) researchers have traditionally attempted to build these knowledge sources by hand, often resulting in brittle, inefficient systems that take many hundreds of hours to build. Overcoming this "knowledge acquisition bottleneck" by applying methods from machine learning is the main goal of this dissertation. We apply methods from *empirical* or *corpus-based* NLP to learn a semantic lexicon, and from *active learning* to reduce the annotation effort required to learn both semantic parsers and lexicons.

The "knowledge acquisition bottleneck" is particularly difficult in the case of knowledge needed for natural language processing, where the amount of information to encode is particularly large, and it is particularly difficult to make such knowledge complete and correct. To attack this problem, recent years have seen the explosion of various computational techniques applied towards the goal of automatically acquiring the knowledge and tools needed to process language (see Brill and Mooney (1997) for an overview). These techniques have drawn from various fields including statistics, machine learning, connectionism, and information-theoretic methods. Within this larger context of empirical NLP, our research focuses on machine learning and statistical methods to automate the construction of semantic lexicons and semantic parsers, from input consisting of sentences annotated with their appropriate semantic representations.

In the empirical approach to language acquisition, two tasks are performed. First, a training corpus is built. The type of corpus depends on the type of analysis desired. Typical corpora consist of sentences paired with parse trees, database queries, or semantic forms such as case-role representations. Another possible input form is syntactically tagged text. Both semantic and syntactic knowledge is often present in the sentence representations. Alternatively, unannotated corpora may be used in the case of systems using unsupervised learning. This dissertation focuses on supervised learning. Figure 1.1 shows an example of part of a corpus of sentences paired with database queries, represented in a logical form that can be executed in Prolog.

The second task required in the empirical approach to language acquisition is to design

```
What is the capital of the state with the biggest population?
answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).

What is the highest point of the state with the biggest area?
answer(P, (high_point(S,P), largest(A, (state(S), area(S,A))))).

What state is Texarkana located in?
answer(S, (state(S), eq(C,cityid(texarkana,_)), loc(C,S))).

What capital is the biggest?
answer(A, largest(A, capital(A))).

What is the area of the United States?
answer(A, (area(C,A), eq(C,countryid(usa)))).

What is the highest point in the state with the capital Des Moines?
answer(C, (high_point(B,C), state(B), capital(B,A),
       eq(A,cityid('des moines',_)))).
```

Figure 1.1: Sample Annotated Corpus

```
([capital], capital(_,_)),           ([state], state(_)),
([biggest], largest(_,_)),           ([in], loc(_,_)),
([highest,point], high_point(_,_)), ([area], area(_,_)),
([population], population(_,_)),    ([capital], capital(_)),
```

Figure 1.2: Sample Semantic Lexicon

and build the acquisition system itself. This system is then trained on the corpus of interest, and the system learns to map each input sentence into its desired representation(s). By using such acquisition systems, an application designer need only decide upon and design a suitable representation, leaving the difficult issue of constructing a parser (or other grammar formalism) to the machine learning system. The system described in this dissertation can in turn aid the parser acquisition system by learning word meanings, thus shortening the inductive leap required.

One knowledge source required by systems that learn to map natural language sentences into deeper semantic representations is the *semantic lexicon*. Here we use this term (often shortened to just *lexicon*) to refer to a list of words (or multi-word phrases) paired with their meanings. We have developed a system, WOLFIE (WOrd Learning From Interpreted Examples), to automatically learn semantic lexicons from a corpus of sentences paired with their meanings. Experimental results show that WOLFIE is able to learn correct and useful mappings. The output of the system can be used to assist a larger language acquisition system; in particular, it is currently used as part of the input to CHILL (Zelle & Mooney, 1993), a parser acquisition system. CHILL requires a word-to-meaning lexicon as background knowledge in order to learn to parse into deep semantic representations. By using WOLFIE, one of the inputs to CHILL is automatically provided, thus easing the task of parser acquisition. Part of a semantic lexicon for the corpus in Figure 1.1 is shown in Figure 1.2.

Though there are existing computational lexicons (e.g., WordNet, (Beckwith, Fellbaum, Gross, & Miller, 1991)) and on-line dictionaries (e.g., *Longman Dictionary of Contemporary English*

(*LDOCE*), (Proctor, 1978)), automated lexicon acquisition is important for several reasons. First, language is constantly changing: new words are created and additional senses are added to existing words. Second, existing lexicons are often customized to one domain, and new domains require slightly or even radically different meanings for many words. Finally, the organization of a lexicon by hand is pain-staking work, and a more logical or useful representation is likely to be found more quickly by automated methods. The automation of lexicon acquisition would enable lexicons to be updated continuously for each new domain with no need to hand-code the lexicon entries.

Before we proceed, a brief discussion of words and their meanings is warranted. As in Miller, Beckwith, Fellbaum, Gross, and Miller (1993), *word form* (or just *word* when the use is obvious) here refers to a "physical utterance or inscription," and *word meaning* "to the lexicalized concept that a form can be used to express." Further, in the context of this dissertation, a word meaning is a symbol, or set of symbols, corresponding to a word form and denoting information about the word that is in some way useful to the performance of intelligent reasoning in the domain in question. In other words, a word's meaning is what we want the Natural Language Understanding (NLU) system to infer (e.g., add to its current knowledge about the sentence being processed), when it sees the word's form. From this definition, it is clear that the meaning (or meanings) of a word is dependent on the representation used by the NLU system at hand. In some simple cases, for example, the meaning of **man** will just be `man`. In this dissertation, word forms will be set in bold font (e.g., **man**), and their meanings in tele-type font (e.g., `[person,sex:male,age:adult]`).

Many past systems designed to automate semantic lexicon acquisition (Fukumoto & Tsujii, 1995; Haruno, 1995; Johnston, Boguraev, & Pustejovsky, 1995; Webster & Marcus, 1995) focus only on acquisition of verbs or nouns, rather than all types of words. Also, these either do not experimentally evaluate their systems, or do not show the usefulness of the learned lexicons. Other approaches (Berwick & Pilato, 1987; Hastings & Lytinen, 1994) to lexicon acquisition assume access to an initial lexicon and parser. A related trend (Boguraev & Briscoe, 1989) is the extraction of lexical knowledge from machine-readable dictionaries such as *LDOCE*. However, lexicons built in this way typically need further tailoring in order to be useful in the domain of intended application.

The induction of the semantic lexicon is not straightforward for several reasons. First, there are a large number of possible meanings that could be induced for each phrase in a sentence: forcing Chill to also learn a lexicon as it is learning parsers would result in an exponential increase in the number of hypotheses to consider, as the size of sentence representations increases. Second, many words have multiple *senses*, or meanings. Therefore, knowing the meaning of a word for some sentences does not guarantee knowing its meaning in all situations. Finally, some words may appear very rarely in a corpus, so that the amount of evidence for inferring the correct meaning for them is quite sparse.

This dissertation presents a lexicon acquisition system, Wolfie, that learns a mapping of words to their meanings, that overcomes many of the limitations of previous work, and that circumvents the difficulties just listed. Although the papers mentioned above and others have also addressed the lexicon learning problem, our method differs from these by combining five features. First, the only background knowledge needed for lexicon learning is in the training examples themselves. Second, interaction with a system, Chill, that learns to parse is demonstrated. Third, a simple, greedy algorithm is used for efficiency to acquire word meanings. Fourth, the system is adaptable to different sentence representations. Finally, the mapping problem is eased by making a *compositionality* assumption that states that the meaning of a sentence is composed from the meanings of the individual words and phrases in that sentence, in addition, perhaps to some "connecting" information specific to the representation at hand. This assumption is similar to the linking rules of Jackendoff (1990).

The WOLFIE learning algorithm exploits the compositionality assumption. By assuming that a sentence meaning is composed from word meanings, we also assume that each component of the sentence representation can be mapped back to the meaning of only one word or phrase in the sentence: the one that has that component as part of its meaning representation. Thus, during lexicon acquisition, word meanings are derived from the components of the representations of sentences in which each word appears. The compositionality assumption decreases the number of meanings that need to be considered for each word. While in broad linguistic domains, compositionality may not always hold, we are primarily interested in acquiring language processing information about one domain at a time, rather than a broad-coverage parser. The goal is not to simply obtain a parse, but to obtain one that is useful for achieving some task, such as answering a question or performing a command. In these situations, compositionality often does hold. In addition, since we are able to learn the meanings of multiple-word phrases, some of the arguments against compositionality (e.g., idioms) may be overcome.

We have tested the utility of the lexicon acquisition algorithm on one real-world corpus and two artificial corpora, with positive results. The ability of the learned lexicons to aid a parser acquisition program were measured for the real corpus and for one of the artificial corpora. The real-world corpus contains natural language questions about U.S. geography paired with their logical query representations, that can then be used to extract answers to the questions from a database. We also show the applicability of the technique to three diverse natural languages other than English: Spanish, Turkish, and Japanese, using translations of the same corpus. We compared these results to the results obtained with lexicons learned by a comparable system developed by Siskind (1996), showing a significant improvement for our system in most circumstances. One of the artificial corpora was based on that of McClelland and Kawamoto (1986), and augmented with Conceptual Dependency (Schank, 1975) information. Finally, we used a set of artificially generated corpora to test the scalability of our algorithm, with encouraging results.

While building an annotated corpus is arguably less work than building an entire NLP system, it is still not a simple task. Redundancies and errors can creep into the data. A goal should be to also minimize the amount of data that is annotated, yet still reach a reasonable level of generalization performance with the system learned from training on that data. In the case of natural language, there is frequently a large amount of unannotated text available. We would like to automatically, but intelligently, choose which of the available sentences to annotate.

We do this here using a technique called *active learning*. Active learning is an emerging research area in machine learning that features systems that automatically select the most informative examples for annotation and training (Cohn, Atlas, & Ladner, 1994). The primary goal of active learning is to reduce the number of examples that the system is trained on, thereby reducing the example annotation cost, while maintaining the accuracy of the acquired information. To demonstrate the usefulness of our active learning techniques, we compared the accuracy of parsers and lexicons learned using examples chosen by active learning to those learned using randomly chosen examples, finding that active learning can save significant annotation cost over training on randomly chosen examples. This savings is demonstrated in the U.S. geography domain.

In summary, this thesis demonstrates a machine learning technique for inducing semantic lexicons; and by building on previous research an entire natural language interface can be acquired from one training corpus. Further, this thesis demonstrates the application of active learning techniques to minimize the amount of data required to annotate as training input for the integrated learning system.

The remainder of the dissertation is organized as follows. Chapter 2 gives more background information on CHILL, and introduces Siskind's lexicon acquisition system, which we will compare to WOLFIE in Chapter 4. Chapter 3 formally defines the learning problem and describes the WOLFIE algorithm in detail. In Chapter 4 we present and discuss experiments evaluating WOLFIE's performance in learning lexicons in a database query domain, and Chapter 5 contains experimental results on two artificial corpora. Next, Chapter 6 describes and evaluates our use of active learning techniques for both CHILL and WOLFIE. Chapters 7 and 8 discuss related research and future directions. Finally, Chapter 9 summarizes our research and results.

Chapter 2

Background

Our approach was developed in the context of learning lexicons to support semantic parsing. As we mentioned in Chapter 1, we use the word *semantic lexicon* to refer to a mapping from words to representations of their meanings. The particular meaning representation is determined by the domain at hand and the desired form for sentence representations. The initial motivation for learning lexicons was so they could be used to bootstrap a parser acquisition system, CHILL (Zelle, 1995). After CHILL acquires a parser, it also uses the lexicons learned to map novel sentences into representations of their meanings. Section 2.1 describes CHILL in more detail.

The most closely related previous research into automated lexicon acquisition is that of Siskind (1996). As we will be comparing our system to his in Chapter 4, we describe the main features of his research in Section 2.2.

2.1 CHILL

The output produced by WOLFIE can be used to assist a larger language acquisition system; in particular, it is currently used as part of the input to the system CHILL. CHILL uses *inductive logic programming* (Muggleton, 1992; Lavrač & Džeroski, 1994) to learn a deterministic shift-reduce parser written in Prolog. The input to CHILL is a corpus of sentences paired with semantic representations, the same input required by WOLFIE. The output is a deterministic parser that maps sentences into parses.

Figure 2.1 shows the basic components of the CHILL system. First, during Parsing Operator Generation, the training examples are analyzed to formulate an overly-general shift-reduce parser that is capable of producing parses from sentences. Next, in Example Analysis, the overly-general parser is used to parse the training examples to extract contexts in which the generated parsing operators are actually useful. The third step is Control-Rule Induction which employs a general ILP algorithm to learn rules that characterize these contexts. Finally, Program Specialization "folds" the learned control-rules back into the overly-general parser to produce the final parser.

The component of CHILL that will be most relevant to our discussion is the initial overly-general parser. This parser varies from one domain and representation language to the next, but it is a straight-forward matter to determine the type of parsing operators needed to parse sentences into a given representation language. The parser is capable of translating a given sentence into many parses representing its meaning, including the correct one(s). Therefore, to increase efficiency,

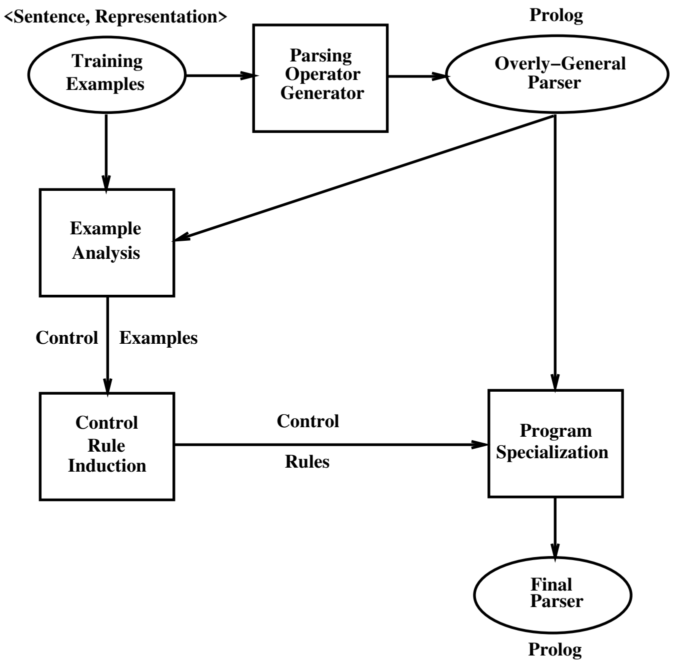


Figure 2.1: The CHILL Architecture

the search for the correct parse during example analysis is limited by a check for compatibility with the correct representation(s), given as training input. Given a correct lexicon, the overly-general parser should be able to parse all of the training examples. The parser is specialized by the learner to produce a final parser that generates only the correct parse(s) for each training sentence, without requiring any access to those parses.

The workings of the overly-general parser will be relevant to our later discussion. Suppose we want to produce a case-role analysis (Fillmore, 1968) of the sentence: "The man ate the pasta.", which we represent as

```
[ate, agt:[man, det:the], pat:[pasta, det:the]].
```

The current parse state is represented by the contents of the parse stack and the remaining portion of the input buffer (Tomita, 1986). Parsing begins with an empty stack and an input buffer containing the entire sentence. At each step of the parse, either a word is shifted from the front of the input buffer onto the stack, or the top two elements on the stack are popped and combined to form a new element that is pushed back onto the stack. A notion of locality is enforced here, since only the top two elements can be combined, not elements from any arbitrary depth in the stack. The sequence of actions and stack states for our simple sentence is shown Figure 2.2. The action notation *(x label)*, indicates that the stack items are combined via the role *label* with the item from stack position x being the head. Choosing the correct action to apply at any given point in the deterministic parse requires a great deal of knowledge. This knowledge is encoded in the rules learned by CHILL.

In this dissertation, we will primarily discuss CHILL in the context of acquiring parsers that map natural-language questions directly into Prolog queries that can be executed to produce an answer (Zelle & Mooney, 1996). Following are two sample queries for a database on U.S. Geography

Action	Stack Contents
	[]
(shift)	[the]
(shift)	[man, the]
(1 det)	[[man, det:the]]
(shift)	[ate, [man, det:the]]
(1 agt)	[[ate, agt:[man, det:the]]]
(shift)	[the, [ate, agt:[man, det:the]]]
(shift)	[pasta, the, [ate, agt:[man, det:the]]]
(1 det)	[[pasta, det:the], [ate, agt:[man, det:the]]]
(2 obj)	[[ate, obj:[pasta, det:the], agt:[man, det:the]]]

Figure 2.2: Shift-Reduce Case-Role Parsing of "The man ate the pasta."

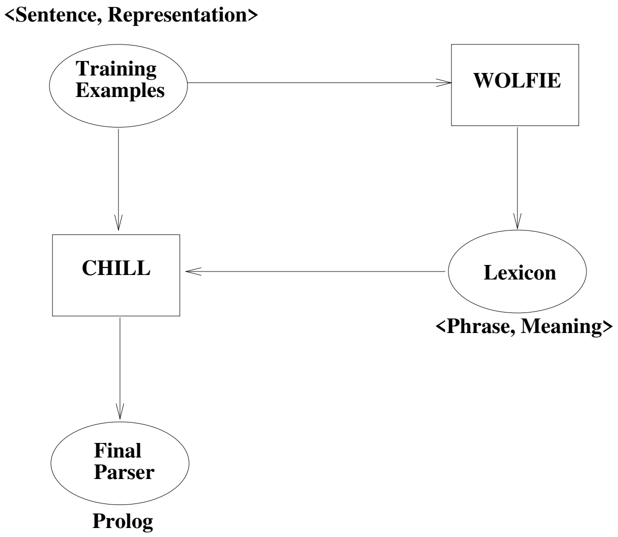


Figure 2.3: The Integrated System

paired with their corresponding Prolog query:

> What is the capital of the state with the biggest population?
> `answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).`
> What state is Texarkana located in?
> `answer(S, (state(S), eq(C,cityid(texarkana,_)), loc(C,S))).`

Given a sufficient corpus of such sentence/representation pairs, CHILL is able to learn a parser that correctly parses many novel questions into logical queries.

CHILL requires a lexicon as background knowledge in order to learn to parse into deeper semantic representations. By using WOLFIE, the lexicon is provided automatically, easing the task of parser acquisition. Figure 2.3 illustrates the inputs and outputs of the complete system. The output of WOLFIE is a lexicon of *(phrase, meaning)* pairs, where the phrases contain one or two words, and the meanings are derived from the representations of sentences in which the phrase appears. WOLFIE will be discussed in more detail in Chapter 3 and throughout the dissertation.

2.2 The Work of Jeff Siskind

Siskind (1996, 1994, 1992) describes a system that learns word to meaning mappings from corpora similar to those we employ. The main difference in the input is that in his training examples, each sentence may be paired with multiple hypothesized meanings, a problem he refers to as *referential uncertainty*, while we assume that each sentence is paired with only one meaning. A second minor difference is that his system requires the representation of a sentence to be variable free, e.g., "John walked to school." would be represented as `go(john, to(school))`. His claim is that sentences do not contain unfilled argument positions. While this may be true for declarative sentences, questions may contain unfilled argument positions. As demonstrated by the queries in the previous section, we do allow variables in the sentence representations presented to CHILL and WOLFIE.

The motivational focus for his system is different than for ours. He primarily hopes to gain insight into the cognitive processes used by children in learning the lexicon, while we focus on learning a lexicon that will be useful for parsing. Because of this focus, his algorithm encodes some cognitive principles. First, a learner can filter out impossible meanings of a sentence based on partial knowledge of the meanings of the words in the sentence. This principle, the *utilizing partial knowledge* principle, is primarily focused towards handling referential uncertainty. Second is the principle of *cross-situational* inference, in which a learner can determine the meaning of a word by finding something in common across several uses of that word.

The third principle is that of *covering* constraints, which is based on the assumption that all components of the sentence meaning must be derived from the meanings of the words in that sentence. This, together with previously learned information, helps the learner determine which meaning fragments must be contained in a word's meaning. For example, if the learner sees the sentence "**John runs.**" paired with the meaning `run(john)`, and knows that **John** means `john`, then it can infer that **runs** must mean `run(_)` since that portion of the sentence meaning must be derived from some word in the sentence, and **John** is already accounted for.

Finally is the principle of *exclusivity*, which states that each fragment of a sentence representation is due to only one word or contiguous phrase in the sentence. In the previous example, if the learner knows that **John** means `john`, this principle lets us infer that **runs** does *not* mean `john`.

An incremental, version-space approach is taken by Siskind's system. Lexicon acquisition proceeds in two stages. The first stage learns *which* symbols in the representation are to be used in the final conceptual expression that represents the meaning of a word. The second stage learns *how* these symbols are put together to form the final representation. For example, when learning the meaning of the word **raise**, the algorithm may learn the set {`CAUSE, GO, UP`} during the first stage and put them together to form the expression `CAUSE(`x`, GO(`y`, UP))` during the second stage.

During the first stage, the algorithm maintains two sets of conceptual symbols for each word. One, the *necessary conceptual-symbol* set, contains conceptual symbols that the algorithm has determined *must* be a part of a word's meaning. The other, the *possible conceptual-symbol* set, contains conceptual symbols that the algorithm has determined *can* be a part of a word's meaning. These sets form the lower and upper bounds on the actual (learned) conceptual-symbol set for that word form. The first stage of learning uses the above principles and each sentence/representation pair in turn to loosen the lower bound and constrain the upper bound. Conceptual symbols are added to the initially empty necessary set, and removed from the initially unconstrained possible set, until the two have converged. To handle ambiguity, the algorithm allows these sets to be split so that there are multiple such "version spaces" for each word, one for each word-sense.

Once a word-sense has converged in stage one, learning for that sense proceeds to stage two. During this stage, the algorithm maintains a set of conceptual expressions for each word sense. This set is initialized to all possible expressions that can be formed out of the symbols appearing in the converged set arrived at in stage one. Expressions that can not be recombined to form the meaning of a sentence in which a word appears are removed from this set until it contains a single expression, at which point learning stops for the word. Note that since the algorithm is incremental, only the current and future uses of a word are checked in this way. As the algorithm is processing input sentence/representation pairs, some words that it is learning might be in stage one, while others have reached stage two, while yet others might be completely learned.

Siskind (1996) shows the effectiveness of his approach on a series of artificial corpora. While the system handles noise, ambiguity, referential uncertainty, and very large corpora, the usefulness of lexicons learned is only compared to the "correct," artificial lexicon. No demonstration of the system's usefulness in real natural language domains has been performed until now. From time to time throughout this dissertation, we will provide further elaboration on various aspects of Siskind's work, to compare and contrast it with our own.

Chapter 3

Lexicon Learning and WOLFIE

3.1 Problem Definition

Although in the end our goal is to acquire an entire natural language interface, we currently divide the task into two parts, the lexicon acquisition component and the parser acquisition component. In this section, we discuss the problem of acquiring semantic lexicons that assist parsing and the acquisition of parsers. The training input consists of natural language sentences paired with their meaning representations. From these pairs we want to extract a lexicon consisting of phrases paired with their meaning representations. Some example training pairs and lexicons were shown in Figures 1.1 and 1.2. To present the learning problem more formally, some definitions are needed. While in the following we use the terms "string" and "substring," these extend in a straight-forward way to natural language sentences and phrases, respectively. The labeled trees are just our meaning representations.

Definition: Let Σ_V, Σ_E be finite alphabets of vertex labels and edge labels, respectively. Let V be a set of vertices, $E \subseteq V \times \Sigma_E \times V$ a set of labeled edges, and l a total function $l : V \rightarrow \Sigma_V$. l is called a "vertex labeling." The triple $G = (V, E, l)$ is a *labeled graph.*

Definition: A *labeled tree* is a connected, acyclic labeled graph.

Figure 3.1 shows the labeled tree t_1 associated with the meaning of the sentence s_1: "The girl ate the pasta with the cheese." In Prolog list form, the meaning is:

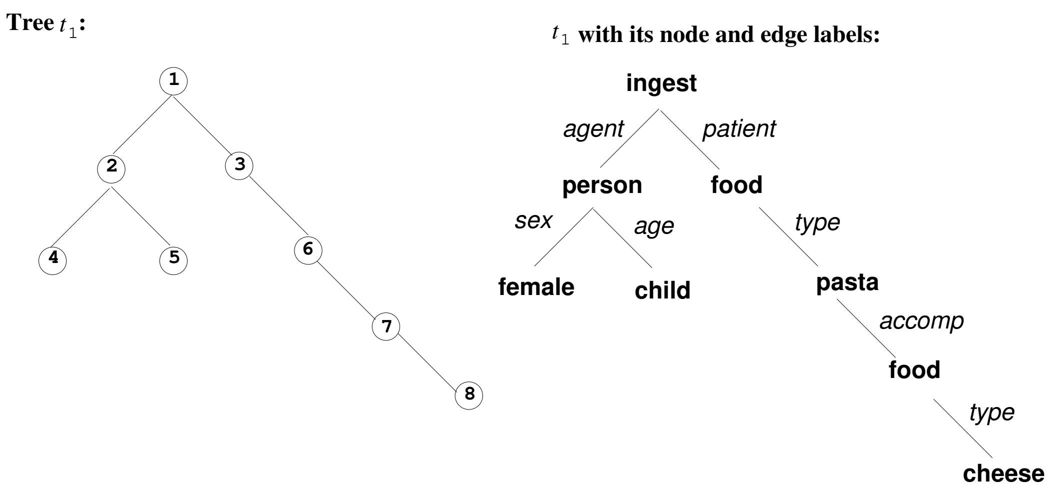
```
[ingest, agent:[person,sex:female,age:child],
      patient:[food,type:pasta, accomp:[food,type:cheese]]].
```

In this example, the function l is:
{(1, `ingest`), (2, `person`), (3, `food`), (4, `female`), (5, `child`), (6, `pasta`), (7, `food`), (8, `cheese`)}.

Definition: An *interpretation* f of a string s to a labeled, rooted tree t is a one-to-one function mapping some of the substrings of s into $nodes(t)$ such that $root(t) \in range(f)$. The substrings in the domain of f must be disjoint.

The interpretation provides information on what parts of the meaning of a sentence come from which of its phrases. This information is only about phrases that have meaning at the "lowest level," or at the terminal nodes of a semantic grammar that is then used to construct the meanings of longer phrases and sentences. In Figure 3.1, we show an interpretation of s_1 to t_1 where f maps

String: "The girl ate the pasta with the cheese."



An Interpretation from "The girl ate the pasta with the cheese." to t_1 :

f("girl") = 2
f("ate") = 1
f("pasta") = 3
f("the cheese") = 7

Figure 3.1: Labeled Graphs and Interpretations

"girl" to node 2, "ate" to node 1, "pasta" to node 3, and "the cheese" to node 7. Note that "with" is not in the domain of f. Because we disallow overlapping substrings in the domain, both "cheese" and "the cheese" could not map to nodes in t_1.

Definition: Given an interpretation f of string s to tree t, and an element p of $domain(f)$, the *meaning* of p relative to s, f, t is the connected subgraph of t whose nodes include $f(p)$ and all its descendents *except* any other nodes of $range(f)$ and their descendents. This subgraph is also a tree, and let us assign $f(p)$ as its root. For $p \notin domain(f)$, the meaning of p is the "empty tree."

Figure 3.2 shows the meanings of each phrase in the domain of the interpretation function shown in Figure 3.1. Here we have shown only the labels on the nodes and arcs for readability.

Definition: Given a set of strings $S = \{s_1, \ldots, s_i, \ldots, s_n\}$, a set of trees $T = \{t_1, \ldots, t_i, \ldots, t_n\}$, and a set of interpretation functions $F = \{f_1, \ldots, f_i, \ldots, f_n\}$, where f_i maps s_i to t_i, let the *language* $\mathcal{L} = \{p_1, \ldots, p_k\}$ of S be the union of all possible substrings[1] of each element in S. For each $p_j \in \mathcal{L}$, the *meaning set* of p_j, denoted $M_{S,T,F}(p_j)$[2], is the set of all meanings of p_j relative to s_i, t_i, f_i for $1 \leq i \leq n$. We consider two meanings to be the same if they are isomorphic trees taking labels into account.

For example, given the second sentence s_2: "The man ate the cheese." with associated meaning t_2 pictured in Figure 3.3, and f_2("the cheese") $= 3$, the meaning set for "the cheese" with respect to $S = \{s_1, s_2\}$, $T = \{t_1, t_2\}$, $F = \{f, f_2\}$, would be {`[food, type:cheese]`}, just

[1]We consider two substrings to be the same string if they contain the same characters in the same order, irrespective of their positions within the larger string in which they occur.

[2]We omit the subscript on M when the sets S, T, and F that the meanings are relative to are obvious from context.

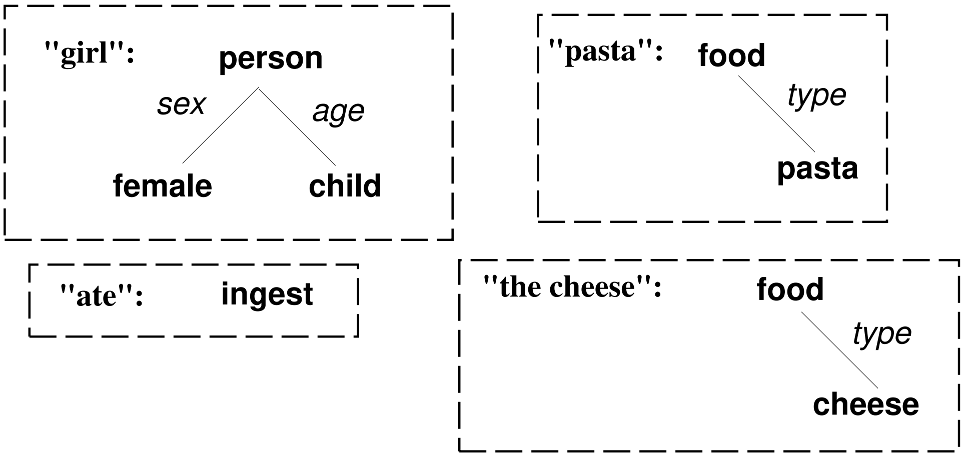


Figure 3.2: Meanings

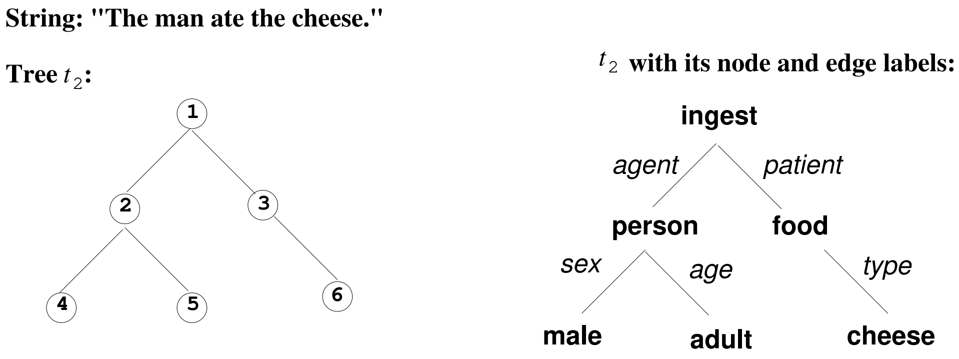


Figure 3.3: A Second Tree

one meaning though f and f_2 map "the cheese" to different nodes in the two trees, because the subgraphs denoting the meaning of "the cheese" for the two functions are isomorphic.

Definition: The *size* of F given S and T is the sum of the sizes of the meaning sets of all $p_j \in \mathcal{L}$, or $\sum_{j=1}^{|\mathcal{L}|} \mid M(p_j) \mid$.

For example, the size of $F = \{f, f_2\}$ given $S = \{s_1, s_2\}$ and $T = \{t_1, t_2\}$, and restricting ourselves to substrings of length two, is 16, as shown in Figure 3.4. In the figure, the symbol $\perp$ is used to indicate the empty tree.

Definition: Given a set of strings $S = \{s_1, \ldots, s_i, \ldots, s_n\}$, a set of trees $T = \{t_1, \ldots, t_i, \ldots, t_n\}$, and a set of interpretation functions $F = \{f_1, \ldots, f_i, \ldots, f_n\}$, the *covering lexicon* expressed by S, T, F is

$$\{(p, m) : p \in \bigcup_{1 \leq i \leq n} domain(f_i), m \in M(p)\}.$$

The covering lexicon expressed by our ongoing example is
{ ("girl", `[person, sex:female, age:child]`), ("ate", `[ingest]`),
("pasta", `[food,type:pasta]`), ("the cheese", `[food,type:cheese]`) }.
Because we only include in the covering lexicon phrases (substrings) that are in the domains of the

"the":	$\perp$	**"the girl":**	$\perp$
"girl":	**[person,sex:female,age:child]**	**"girl ate":**	$\perp$
"ate":	**[ingest]**	**"ate the":**	$\perp$
"pasta":	**[food,type:pasta]**	**"the pasta":**	$\perp$
"with":	$\perp$	**"pasta with":**	$\perp$
"cheese":	$\perp$	**"with the":**	$\perp$
"man":	$\perp$	**"the cheese":**	**[food,type:cheese]**
		"the man":	$\perp$
		"man ate":	$\perp$

Figure 3.4: The Size of F

f_i's, words with the empty tree as meaning are not included in the covering lexicon. Note also that we will in general use "phrase" to mean substrings of sentences, whether they consist of one word, or more than one.

The Lexicon Acquisition Problem:
Given: a set of strings $S = \{s_1, \ldots, s_n\}$ and a set of trees $T = \{t_1, \ldots, t_n\}$,
Find: a set of interpretation functions, $F = \{f_1, \ldots, f_n\}$, such that the size of F is minimized. If such a set is found, we say we have found a *minimal* set of interpretations (or a *minimal covering lexicon*). □

Less formally, a learner is presented with a set of sentences (S) paired with their meanings[3] (T); the goal of learning is to find the smallest lexicon possible. This lexicon is the paired listing of all phrases in *domain*(F) (where F is the set of interpretation functions found) with each of the elements in their meaning sets. The motivation for finding an interpretation set of minimal size is the usual bias towards simplicity of representation. While this definition allows for phrases of any length, we will usually want to limit the length of phrases to be considered for inclusion in the domain of the interpretation functions, for efficiency purposes.

This definition of the lexicon acquisition problem differs from that given by other authors, including Riloff and Sheperd (1997), Siskind (1996), Manning (1993), Brent (1991) and others. For example, Riloff and Sheperd (1997) define semantic lexicons as the clustering into groups of similar words. Manning (1993) and Brent (1991) describe work on learning selectional restrictions. Our definition focuses on deriving a word's meaning from representations of the sentences in which it appears.

Our definition of the problem makes some assumptions about the training input. Discussion of the implications of these assumptions and thoughts on removing some of them in the future is postponed to later in this chapter and to Chapter 8. First, by making f a function instead of a relation, the definition assumes that the meaning for each phrase in a sentence appears once in

[3] assumed to be represented as trees

the representation of that sentence, the *single-use* assumption. Second, by making f one-to-one, it assumes *exclusivity*, that each node in a sentence's representation is due to only one phrase in the sentence. Third, it assumes that a phrase's meaning is a connected subgraph of a sentence's representation, not a more distributed representation, the *connectedness* assumption. While the first assumption may not hold for some representation languages, it does not present a problem in the domains we have considered. The second and third assumptions are perhaps less problematic with respect to general language use. The single-use assumption is not made by Siskind (see Section 2.2), and it is not clear whether or not he could handle word meanings distributed over a representation. He does also assume exclusivity.

Our definition also assumes *compositionality*: that the meaning of a sentence is derived from the meanings of the phrases it contains, but is not derived from external sources such as noise. In other words, all the nodes of a sentence's representation are included within the meaning of some word or phrase in that sentence. Since we allow multi-word phrases in the lexicon (e.g., ([**kick the bucket**], `die(_)`)), this assumption seems fairly unproblematic. In addition, future versions could allow a loosening of this restriction to allow for information about the context of surrounding sentences or of the situation to be included in the meaning of a sentence, and for noisy input. Compositionality is also assumed by Siskind.

This definition also allows training input in which:

1. Sentences may be ambiguous (be paired with more than one meaning).

2. Words and phrases may have multiple meanings. That is, ambiguity (polysemy) may occur in the lexicon.

3. Several phrases may map to the same meaning. That is, synonymy may occur in the lexicon.

4. Some words in a sentence may map to the empty tree, leaving them unused in the assignment of words to meanings.[4]

5. Phrases of contiguous multiple words can indicate a meaning of part of a sentence's representation.

This definition of the lexicon acquisition problem does not fit cleanly into the traditional definition of learning for classification. For each phrase, we are given examples of concepts (the meaning of the sentence in which the phrase appears) that contain the meaning of the phrase, and we are trying to pick out the relevant "features," or nodes of the representation, corresponding to the correct meaning. However, there are additional constraints imposed by our assumptions of single-use, connectedness, and compositionality. To complicate matters further, each training example may contain information about multiple classes (phrases), and negative examples are not available.

In some ways the problem is related to clustering, which is also capable of learning multiple, potentially non-disjoint categories. However, the clusters to learn are not necessarily related to each other in a hierarchical taxonomy. Also, it is not clear how a clustering system could be made to learn the phrase to meaning mappings needed for parsing. Finally, current systems that learn multiple concepts commonly use examples for other concepts as negative examples of the concept

[4]These words may, however, serve as cues to a parser on how to assemble sentence meanings from word meanings.

currently being learned. The implicit assumption made by doing this is that concepts are disjoint, an unwarranted assumption when there is synonymy.

In the next section, we discuss several possible ways to attempt to solve the lexicon acquisition problem, and their strengths and weaknesses. The final section describes the approach we took to solve the problem, and its implementation in WOLFIE.

3.2 Potential Approaches

A first attempt to solve the Lexicon Acquisition Problem might be to examine all interpretation functions across the corpus, then choose the one(s) with minimal size. The number of possible interpretation functions for an input pair is dependent on both the size of the sentence and its representation. In a sentence with w words, there are $\Theta(w^2)$ possible phrases, not a particular challenge. However, the number of possible interpretation functions grows extremely quickly with the size of the input. For a sentence with p phrases and associated tree with n nodes, the number of possible interpretation functions is:

$$p!(n-1)! \sum_{i=1}^{min(p,n)} \frac{1}{(i-1)!(n-i)!(p-i)!}. \tag{3.1}$$

The derivation of the above formula is as follows. We must choose which phrases to use in the domain of f, and we can choose one phrase, or two, or any number up to $min(p,n)$ (if $n < p$ we can only assign n phrases since f is one-to-one), or

$$\binom{p}{i} = \frac{p!}{i!(p-i)!}$$

where i is the number of phrases chosen. But we can also permute these phrases, so that the "order" in which they are assigned to the nodes is different. There are $i!$ such permutations. We must also choose which nodes to include in the range of the interpretation function. We have to choose the root each time, so if we are choosing i nodes, we have $n-1$ choose $i-1$ nodes left after choosing the root, or

$$\binom{n-1}{i-1} = \frac{(n-1)!}{(i-1)!(n-i)!}.$$

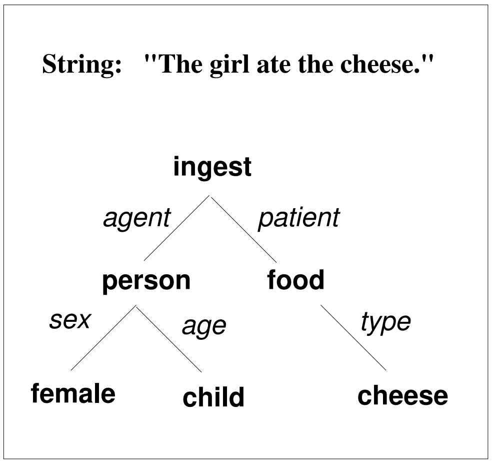
The full number of possible interpretation functions is then

$$\sum_{i=1}^{min(p,n)} \frac{(n-1)!}{(i-1)!(n-i)!} \times \frac{p!}{i!(p-i)!} \times i!,$$

which simplifies to Equation 3.1.

When $n = p$, the largest term of the summation in Equation 3.1 is $p!$, which grows faster than exponentially with p, so in general the number of interpretation functions is too large to allow enumeration. Therefore, finding a lexicon by examining all interpretations across the corpus, then choosing the one(s) with minimal size, is clearly not tractable.

Instead of finding all interpretations, one could find a set of candidate meanings for each phrase, from which the final meaning(s) for that phrase could be chosen. One way to do this is



Covers the input:

("girl", [person,sex:female,age:child]),
("ate", [ingest]),
("cheese", [food,type:cheese])

Does not cover the input:

("girl", [person,sex:female]),
("ate", [ingest]),
("cheese", [cheese])

Figure 3.5: Covering Example

to *fracture* (Siskind, 1992) the meanings of sentences in which a phrase appears, which in the tree representation formalism corresponds to finding all possible connected subgraphs. This would also lead to an exponential blowup in the number of candidate meanings for a phrase: A lower bound on the number of connected subgraphs for a full binary tree with N nodes is obtained by noting that any subset of the $L = (N+1)/2$ leaves may be deleted and still maintain connectivity of the remaining tree. Thus, counting all of the ways that leaves can be deleted gives us a lower bound of $2^{(N+1)/2}$ fractures.[5] While in practice we have not encountered very large trees, this could limit the scalability of an approach that uses fractures.

This exponential blowup would also be a barrier if we were to force CHILL to induce the lexicon on its own. One would have to let each phrase potentially map to all fractures of all representations of sentences in which it appears. Even with small representations, this would likely lead to a system with poor generalization ability.

Another difficulty with fracturing is that though it may generate candidate lexicon entries, the lexicon is no longer guaranteed to be a *covering* lexicon as defined in Section 3.1. For example, Figure 3.5 shows an example of a sentence/representation pair and two lexicons, one that does cover the pair and one that does not. Either of these lexicons might be generated by a scheme that uses fracturing alone.

If we could efficiently find some good candidates, a standard induction algorithm could then attempt to use this list as a source of training examples for each phrase. However, any attempt to use the list of candidate meanings of one phrase as negative examples for another phrase would be flawed. The learner could not know in advance which phrases are possibly synonymous, and thus which phrase lists to use as negative examples of other phrase meanings. Also, many representation

[5] Thanks to net-citizen Dan Hirshberg for help with this analysis

components would be present in the lists of more than one phrase. This is a source of conflicting evidence for a learner, even without the presence of synonymy. Since only positive examples are available, one might think of using most specific conjunctive learning, or finding the intersection of all the representations for each phrase, as proposed by Anderson (1977). However, because of polysemy, the meaning for a given phrase is potentially disjunctive, and then this intersection would be empty.

Because of the above obstacles, we do not use a standard induction algorithm to find a lexicon. Instead, we generate a set of candidate lexicon entries, from which the final learned lexicon is derived by greedily choosing the "best" lexicon item at each point, in the hopes of finding a final covering lexicon. This method "carve outs" one subgraph at a time from a sentence's representation and assigns it to the meaning of a phrase in that sentence. We are finding meanings, not interpretation functions. Therefore, simply checking whether the root is in the range of the interpretation function does not ensure that all portions of the representation (nodes in the tree) have been accounted for by phrase meanings. In other words, it does not guarantee that we find a covering lexicon. Even without this constraint, using a greedy search would also not guarantee covering of the input, and of course it also does not guarantee that a minimal lexicon is found. As we will discuss, however, there is more to our search than just a greedy choice of lexicon entries; further, our experimental results demonstrate that our greedy search performs well, and that learning is quite fast.

The problem we address is that of finding a "good" set of candidate meanings for each phrase, and then manipulating them as needed to form a final lexicon that covers the input. We try to ensure the former by using the Largest Isomorphic Connected Subgraphs (LICS) between pairs of representations of sentences in which a phrase appears as initial candidate meanings for the phrase. We will describe LICS more thoroughly in Section 3.3. We then attempt to ensure the latter by further restricting the candidate meanings as needed, by appealing to our assumptions of single-use, connectedness, and exclusivity, before adding them to the final learned lexicon.

Choosing good final lexicon items from the candidate set could be attempted in several ways. One is the "histogram" paradigm, in which a count is made of the number of times each candidate meaning appears with each phrase and the pair with the largest count is chosen first.[6] However, this does not discriminate well between common and rare words. A second way, the one we use, is to maximize the prediction of meanings given phrases. We attempt to find the "maximally common" meaning for each phrase while still allowing coverage of most of the input. The next section expands on these ideas to describe our implemented algorithm, WOLFIE.

3.3 The WOLFIE Algorithm and an Example

This section describes the WOLFIE algorithm, implemented in Prolog and outlined in Figure 3.6. In the current implementation, we only consider learning phrases of at most two words for efficiency reasons. Future work includes efficiently including potentially meaningful phrases of more than two words. First, the initial list of candidate meanings for a phrase is formed by finding the LICS

[6]Thanks to J. Siskind for initial discussions of this paradigm

For each phrase, p (of at most two words):
    1.1) Collect the training examples in which p appears
    1.2) Calculate LICS from (sampled) pairs of these examples' representations
    1.3) For each l in the LICS, add (p, l) to the set of candidate lexicon entries
Until the input representations are covered,
        or there are no remaining candidate lexicon entries do:
    2.1) Add the "best" (phrase, meaning) pair to the lexicon
    2.2) Update candidate meanings of phrases occurring in the same sentences
            as the phrase just learned
Return the lexicon of learned (phrase, meaning) pairs

Figure 3.6: WOLFIE Algorithm Overview

between sampled pairs of representations of the sentences in which the phrase appears.[7] Using the LICS between representations is a bottom-up (i.e., specific to general) method; bottom-up methods have proven their success in previous machine learning research. For this problem, it is a useful technique that helps limit the number of candidates to be considered. We later demonstrate that with enough training examples, the candidates it generates are good ones. As we will explain, we use the exclusivity assumption explicitly in the algorithm to help avoid over-specialization, a potential problem with using bottom-up methods.

After generating an initial set of candidate lexicon entries, a greedy covering search is started to choose the entries to be included in the final learned lexicon. We use a greedy algorithm to limit the time and space complexity of our algorithm. At each step of the search, the best (phrase, meaning) pair is chosen from the candidates, according to a heuristic described below, and added to the final lexicon. Each choice of a lexicon item can constrain the candidate meanings for other phrases, since each subgraph of the representation can be accounted for by only one phrase in the sentence. This is because we assume that the interpretations mapping sentences to representations are functions, not relations. The algorithm is described in more detail in the next several sections. If any of the assumptions made by the problem definition are violated, we do not guarantee that a covering lexicon will be found; however, the system can still be run and learn a potentially useful lexicon.

The WOLFIE algorithm has been implemented to handle two kinds of semantic representations. First is a case-role representation augmented with Conceptual Dependency (CD)(Schank, 1975) information; examples were shown in Figures 3.1 and 3.3. Experimental results using this representation will be discussed in Section 5.1. The second representation handled is a logical query representation; examples were shown in Figure 1.1. This representation is also tree structured as is shown in Figure 3.7.

Since we will use the logical query representation in the algorithm example, we now describe the algorithm for finding the Largest Isomorphic Connected Subgraphs (LICS) for this representation. The LICS algorithm for the CD representation will be discussed in Section 5.1. The LICS is

[7] As discussed in the example, phrases that appear in only one sentence have as their candidate meaning the representation of the sentence in which they appear.

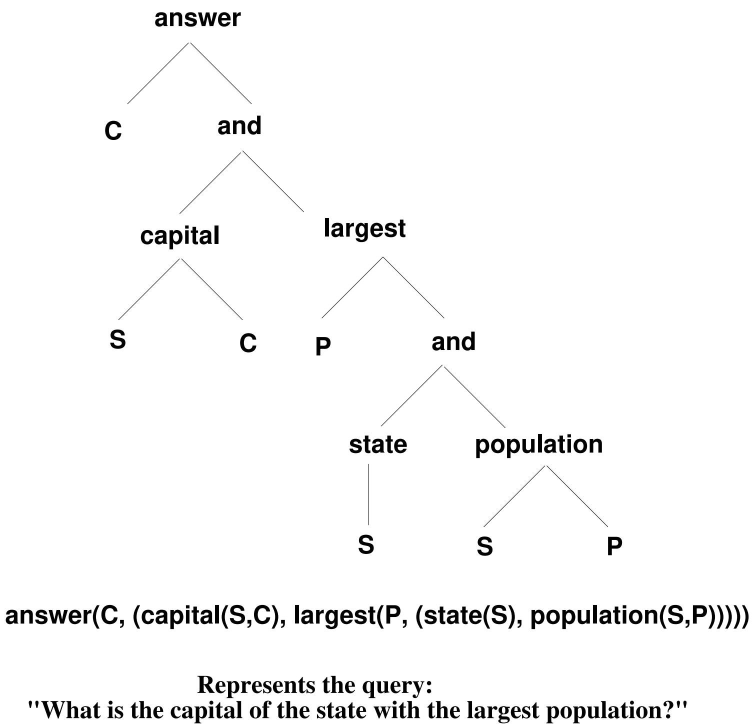


Figure 3.7: Database Queries as Trees

just what it says: the problem is that we are given two labeled graphs, and we want to find the LICS of the two graphs, where isomorphism takes the labels of the two graphs into account. The Largest Common Subgraph problem is solvable in polynomial time if both inputs are trees (Garey & Johnson, 1979), which they will be in our representations.

Finding common meanings between two trees in the logical query representation is complicated by variable renaming and conjunction. Therefore, to find the LICS between pairs of query representations, we use LICS plus a method similar to finding the Least General Generalization (LGG) of first-order clauses (Plotkin, 1970). Summarizing that work, the LGG of two clauses is the least general clause that subsumes both clauses; our version of LGGs simply operates over terms instead of clauses. To find common meanings then, we find the LICS between two trees, forming a rooted tree with root labeled r in each tree. Then for the subtrees rooted at r in the input trees, we find their LGG.

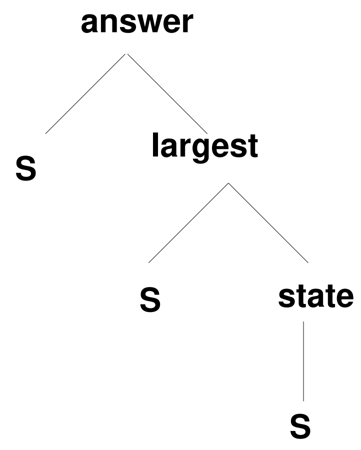
For example, given the trees in Figures 3.7 and 3.8, or in Prolog form:

```
answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).

answer(S, largest(S, state(S))).,
```

the common meaning is `largest(_,state(_))`.[8] The steps are illustrated in Figures 3.7, 3.8, and 3.9. The LICS between the two trees (again, excluding `answer`) is just `largest`, but we use the subtree rooted at `largest` in each tree as each input to the LGG. A second example, demonstrating

[8]Since CHILL initializes the parse stack with the `answer/2` predicate, it is first stripped from the input given to WOLFIE. While this means that representations are forests, not trees, the learning problem still holds, as the root (`answer`) is implicitly assigned a meaning.



answer(S, largest(S, state(S)))

Represents the query:
"What is the largest state?"

Figure 3.8: A Second Database Query

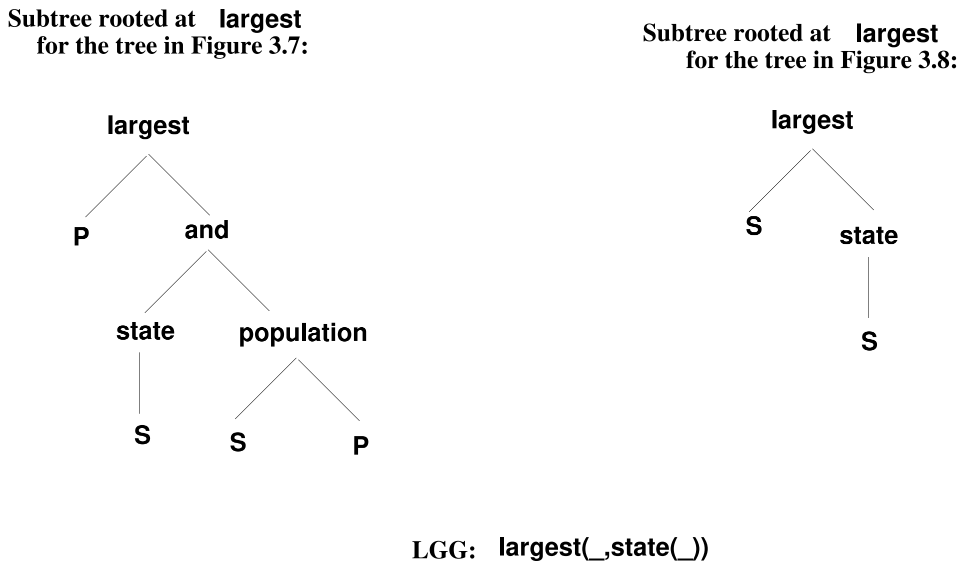


Figure 3.9: LICS Plus LGGs

a result with common variables, are the trees

```
(state(S), loc(C,S))

(high_point(B,C),loc(C,B),state(B),capital(B,A))
```

with a common meaning of `(state(N), loc(_,N))`. Other systems that deal with finding generalizations of structured objects include OCCAM (Pazzani, 1985), UNIMEM (Lebowitz, 1987), and LABYRINTH (Thompson & Langley, 1991).

3.4 Generating Candidate Lexicon Items

We now describe the algorithm in more detail with the help of an example from the U.S. Geography query domain. The first step is to derive an initial set of candidate meanings for each one and two word phrase. For example, let us suppose we have the following pairs as input:

1. What is the capital of the state with the biggest population?
   `answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).`
2. What is the highest point of the state with the biggest area?
   `answer(P, (high_point(S,P), largest(A, (state(S), area(S,A))))).`
3. What state is Texarkana located in?
   `answer(S, (state(S), eq(C,cityid(texarkana,_)), loc(C,S))).`
4. What capital is the biggest?
   `answer(A, largest(A, capital(A))).`
5. What is the area of the United States?
   `answer(A, (area(C,A), eq(C,countryid(usa)))).`
6. What is the population of a state bordering Minnesota?
   ```
   answer(P, (population(S,P), state(S), next_to(S,M),
        eq(M,stateid(minnesota)))).
   ```
7. What is the highest point in the state with the capital Des Moines?
   ```
   answer(C, (high_point(B,C), loc(C,B), state(B), capital(B,A),
        eq(A,cityid('des moines',_)))).
   ```

As a simplification, assume that sentences are stripped of phrases that we know *a priori* have empty meanings (although in general this is not required). In the example sentences, these phrases are [**what**], [**is**], [**with**], and [**the**]. We will also assume that database constants (location names) are given to the learning algorithm as background knowledge, in order to keep the explanation simpler. We generate candidate lexicon entries for a phrase by finding the LICS between pairs of representations of the sentences in which that phrase appears. The sets of initial candidate meanings for some of the phrases in this corpus are:

Phrase	LICS	Derived from Sentences
[**capital**]:	`largest(_,_)`	1,4
	`(capital(A,_),state(A))`	1,7
[**biggest**]:	`largest(_,state(_))`	1,2
	`largest(_,_)`	1,4;2,4
[**state**]:	`largest(_,state(_))`	1,2
	`state(_)`	1,3;2,3
	`(capital(A,_),state(A))`	1,7
	`(high_point(B,_),state(B))`	2,7
	`(state(S),loc(_,S))`	3,7
[**highest,point**]:	`(high_point(B,_),state(B))`	2,7
[**located**]:	`(state(S),loc(_,S))`	3
[**in**]:	`(state(S),loc(_,S))`	3,7

Note that [**state**] has five candidate meanings, each generated from different pairs of sentences in which it appears. Also, [**capital**] and [**biggest**] each have two candidate meanings. For phrases that only appear in one sentence, we use the entire representation of the sentence in which they appear as an initial candidate meaning. In this corpus, such phrases include [**located**] and [**bordering**]. As we will see, this type of pair typically has a low heuristic score, so the meaning will usually be generalized (in step 2.2 of the algorithm, described later) to just the correct portion of the representation, if any, by the time it is added to the final lexicon. Finally, if a phrase is ambiguous, the pairwise matchings to generate multiple candidate items, together with the updating of the candidate set in step 2.2, enable multiple meanings to be learned for it.

One thing to notice about these candidate meanings, is that the overly-specific meanings (e.g., `largest(_,state(_))` for **state**) are typically derived from sentences that have more than one phrase in common. The algorithm cannot distinguish *a priori* which phrase corresponds to which predicate(s), but again candidate generalization and appealing to exclusivity will usually distinguish which is which. We could force the algorithm to only generate LICS from representations of sentences that only have one phrase in common, but there are not always such sentences for every phrase in the corpus.

3.5 Finding the Best Pair

After deriving the candidate lexicon items, the greedy search begins. The heuristic used to evaluate candidate (phrase, meaning) pairs is the sum of three weighted components, where p stands for phrase and m stands for meaning:

1. $P(m|p) \times P(p|m) \times P(m) = P(m|p)^2 \times P(p)$.
2. The generality of m.
3. The amount of orthographic overlap between p and m.

The first measure, *meaning utility*, is analogous the the cluster evaluation heuristic used by Cobweb (Fisher, 1987), which measures the utility of clusters based on attribute-value pairs and categories, instead of meanings and phrases. The goal of this measure is to maximize the probability of predicting the correct meaning for a randomly sampled phrase. The equality holds by Bayes Theorem; looking at the right side of the equation, $P(m|p)^2$ is the *expected* number of times that meaning m can be correctly guessed for a given phrase, p. This expectation assumes a guessing strategy that is probability matching. In other words, it assumes that a meaning m for p is guessed with probability $P(m|p)$ and that this guess is correct with the same probability. The second term, $P(p)$, biases the term by how common the phrase is.

Looking at the left side of the equation, the first term helps bias the learner towards lexicons with low ambiguity, by preferring phrases that indicate a particular meaning with high probability. The second term helps reduce unneeded synonymy in a similar way. Finally, lexicon items with a large value for $P(m)$ are those for which we would have the most confidence in having found a correct mapping, and thus should be chosen first. The probabilities are calculated from the training input and then are updated as learning progresses so that they do not include counts for meanings and phrases already covered by previously learned items. We will see how this updating works as we continue through our example of the algorithm.

The formula is calculated as follows:

$$P(m|p)^2 \times P(p) = \frac{count(m,p)^2}{count(p)^2} \times \frac{count(p)}{\sum_{j=1}^{N} count(p_j)},$$

which we can simplify to the following:

$$\frac{count(m,p)^2}{count(p) \times \sum_{j=1}^{N} count(p_j)} \tag{3.2}$$

Here, $count(m,p)$ is the number of times that m and p appear in a sentence representation pair where both m and p are as yet uncovered by items in the learned lexicon, and $count(p_j)$ is the number of times that phrase p_j remains uncovered in the input. Finally, N is the number of unique phrases in the input.

The second measure, *generality*, prefers pairs with general meanings over those with more specific ones, and is the negation of the number of nodes in the tree-structured representation of the meaning. Learning a meaning with fewer terms helps evenly distribute the predicates in a sentence's representation among the meanings of the phrases in that sentence, and thus leads to a lexicon that is more likely to be correct. To see this, we note that some pairs of words tend to frequently co-occur, and so their joint representation (meaning) is likely to be in the list of candidate meanings for both words. By preferring a more general meaning, we easily ignore these incorrect joint meanings. In the candidate set above for example, if all else were equal, the generality portion of the heuristic would prefer `state(_)`, with generality value -1, over `largest(_,state(_))` and `(capital(A,_),state(A))`, each with generality value -2, as the meaning of **state**.

The final measure, *orthographic overlap*, is used since in some domains phrases have many characters in common with their meanings, as in [**area**] and `area(_)`. To compute this value, we measure the length, n, of the largest number of characters in common between any prefix of the characters in the words in the phrase, and any substring of the characters in the terms and predicates in the meanings.[9] We do not use standard longest common substring because in this

[9]We actually find the largest number of characters in common between prefixes of each *word* in the phrase and the meaning, but ensuring that these do not overlap, by removing the longest common substring from the meaning if needed and recomputing the number of characters in common between the remaining meaning and the other word(s) in the phrase.

domain we typically only realize a benefit if *prefixes* of the words match the meaning. We then divide n by the number of characters in the phrase and by the number of characters in the term and predicate names, and average the two. Since orthographic overlap does not occur in all domains, we demonstrate in our experimental results that this component of the heuristic has a small effect on learning.

For purposes of this example, we will use a weight of 10 for the first and third parameters, and a weight of 1 for the second. The first and third components have smaller values than the second, so they have higher weights. Results are not overly-sensitive to the heuristic weights. Automatically setting the weights using cross-validation on the training set (Kohavi & John, 1995) had little effect on overall performance. If multiple candidate pairs have the same heuristic value, we choose less "ambiguous" phrases first and prefer short phrases over longer ones. A phrase is considered more ambiguous than another if it currently has more meanings in the partially learned lexicon.

We show the calculation of the heuristic measure for some of the above twelve pairs, and the resulting value for all. In this example, we omit the summation in the denominator in Equation 3.2 since it has the same value for all candidate pairs in each iteration of the loop. However, it is used in the experimental results presented in the next chapter, since using it equalizes the weight of the meaning utility portion of the heuristic with increasing numbers of training examples.

([**capital**], `largest(_,_)`): $10(2^2/3) + 1(-1) + 10 * 0 = 19.25$,

([**capital**], `(capital(A,_),state(A))`):
$10(2^2/3) + 1(-2) + 10(7/7 + 7/12)/2 = 19.25$,

([**biggest**], `largest(_,_)`): 29,

([**biggest**], `largest(_,state(_))`): 11.3,

([**state**], `largest(_,state(_))`): 15.1,

([**state**], `state(_)`): $10(4^2/4) + 1(-1) + 10((5/5 + 5/5)/2) = 49$,

([**state**], `(capital(A,_),state(A))`): 15.1,

([**state**], `(high_point(B,_),state(B))`): 14.7,

([**state**], `(state(S),loc(_,S))`): 16.1

([**highest,point**], `(high_point(B,_), state(B))`):
$10(2^2/2) + 1(-2) + 10((9/12 + 9/15)/2) = 24.75$,

([**located**], `(state(S), loc(_,S))`):
$10(1^2/1) + 1(-2) + 10(3/7 + 3/8)/2 = 12.0$,

([**in**], `(state(S), loc(_,S))`): 18.

The best pair by our measure is ([**state**], `state(_)`), so it is added to the lexicon.

3.6 Constraining Remaining Candidates

As noted by Tishby and Gorin (1994), acquisition techniques based only on correlations between words and their meanings can be unfocused, as the number of possible correlations is very large. We avoid these problems by using correlational information (our meaning utility measure) *together* with the other parts of the heuristic. To further restrict the number of possible correlations,

For each candidate (phrase, meaning) pair (p, m)
  If p occurs in some same input pairs as the lexicon pair just learned then
    If the nodes of m overlap the nodes in the meaning of the lexicon pair just learned then
      If all occurrences of m are now covered then
        Remove (p, m) from the set of candidate pairs
      Else
        Adjust the heuristic value of (p, m) as needed to account
          for newly covered nodes of the input representations
      Generalize m to remove covered nodes, obtaining m', and
      Calculate the heuristic value of the new candidate pair (p, m')
If no candidate meanings remain for an uncovered phrase then
  Find new candidate meanings from uncovered representations and
  Calculate their heuristic values

Figure 3.10: The Candidate Generalization Phase

we use the principle of exclusivity: one of the key ideas of the algorithm is that each (phrase, meaning) choice can constrain the candidate meanings of phrases yet to be learned. This happens in step 2.2 in Figure 3.6. Such constraints exist because of the assumption that each portion of the representation is due to at most one phrase in the sentence (exclusivity). Therefore, once part of a sentence's representation is covered by the meaning of one of its phrases, no other phrase in the sentence can be paired with that meaning (for that sentence).

Here is an example of this concept, not a part of our larger ongoing example. Assume we have only the following three sentence/representation pairs:

What is the capital of the state with the biggest population?
`answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).`

What state is Texarkana located in?
`answer(S, (state(S), eq(C,cityid(texarkana,_)), loc(C,S))).`

What is the highest point of the state with the biggest area?
`answer(P, (high-point(S,P), largest(A, (state(S), area(S,A))))).`

From these examples, the meaning of [**state**], the only phrase (other than closed class words) common to all three sentences, is determined (by doing pairwise LICS) to be `state(_)`, and the pair ([**state**], `state(_)`) is added to the lexicon. Before adding it, the candidate meaning for [**biggest**] is `largest(_, state(_))` (the LICS of the representations of the two sentences containing "biggest"). However, since `state(_)` is now covered by ([**state**], `state(_)`), it can be eliminated from consideration as part of the meaning of [**biggest**], and the candidate meaning for [**biggest**] is generalized to `largest(_,_)`. We use an operation analogous to set difference when finding the remaining uncovered nodes of the representation when generalizing meanings to eliminate covered nodes from candidate pairs.

The candidate generalization step is described in Figure 3.10. Returning to our running example, generalizing candidate entries proceeds as follows. First, we note that all occurrences of [**state**] are covered by the (just learned) pair ([**state**], `state(_)`), so all of its other candidate meanings are discarded. Next, we search for other candidate pairs whose meanings and heuristic values might need to be updated. In our example, all of the phrases in the candidate pairs we have been working with appear in some same input pair as ([**state**], `state(_)`).

First, for [**capital**], nodes in `(capital(A,_), state(A))` overlap nodes in `state(_)`. Further, `state(_)` is now covered in all sentences, so `(capital(A,_), state(A))` is removed from the set of candidate meanings for [**capital**], and generalized to `capital(_,_)`, with a heuristic value of 22.3. In a similar manner, the second candidate meaning for [**biggest**], `largest(_,state(_))` becomes `largest(_,_)`. But ([**biggest**], `largest(_,_)`) is is already a candidate pair, so nothing further is done with it. Also, the candidate meaning for [**highest,point**] is generalized to `high_point(_,_)`, with a heuristic value of 27.3. The candidate meaning for [**located**] becomes `loc(_,_)`, with a heuristic value of 15.1, and the candidate meaning for [**in**] becomes `loc(_,_)`, with a value of 19.

In the second iteration of the greedy loop, the best pair (of those we show here) is ([**biggest**], `largest(_,_)`). This would cause `largest(_,_)` to be eliminated from consideration as a candidate meaning for [**capital**], since `largest(_,_)` is now covered by [**biggest**]. Therefore, new candidate meanings are derived for [**capital**], using LICS, resulting in `capital(_,_)`. Since this is already a candidate meaning, we continue the search. The best pair in the next iteration, ([**highest,point**], `high_point(_,_)`), would not effect any other candidate pairs. In the next iteration of the greedy loop, the best pair is ([**capital**],`capital(_,_)`). This would leave sentence 4 as the only uncovered occurrence of [**capital**], so the entire uncovered representation of the sentence, `capital(_)`, is used as a candidate meaning. Had there been two or more uncovered occurrences, we would have derived new LICS, if possible, from the remaining uncovered representations. The value of the pair ([**capital**], `capital(_)`) is calculated as $10 * (1^2/1) + 1(-2) + 10(7/7 + 7/7)/2 = 18$. Notice the denominator of the first term is one, because we use the *uncovered* occurrences of the phrase in the calculation.

The greedy search continues until the lexicon covers the training corpus, and the entire final learned lexicon is:

([**state**], `state(_)`),
([**area**], `area(_,_)`),
([**population**], `population(_,_)`),
([**biggest**], `largest(_,_)`),
([**highest,point**], `high_point(_,_)`),
([**capital**], `capital(_,_)`),
([**in**], `loc(_,_)`),
([**capital**], `capital(_)`),
([**states**], `(next_to(S,_), state(S))`).

The unusual meaning for [**states**] is due to the fact that **states** occurs only once in this sample corpus, in a context with **borders**, which in reality should map to `next_to(_,_)`. In a larger corpus, these two words are much more likely to be paired with their correct meanings in the learned lexicon.

We can now mention some other distinctions between our system and Siskind's. As described in Section 2.2, his algorithm has two learning stages, first learning which symbols should appear in a words meanings, and then learning how those symbols combine together to form one meaning, or subgraph. By using common subgraphs and exclusivity, we combine these two stages into one in WOLFIE. We return next to the principles embodied in Siskind's system, first defined in Section 2.2. First, *cross-situational* inference is used by both systems, though neither system uses the principle in its strongest sense such that all occurrences of a word would be required to have some common meaning, which would preclude ambiguous words. *Covering* constraints are also used by WOLFIE, though because of its greedy nature, it allows pieces of the representation of a sentence to remain uncovered. This is as it should be since all representation formalisms may not comply with this constraint. Finally, both systems use *exclusivity*.

In summary then, this research and Siskind's share many of the same assumptions; the main differences between the two are computational. First, his system operates in an incremental or on-line fashion, discarding each sentence after processing it, while ours is batch. Second, his search for word meanings is most analogous to a version space search, while ours is a greedy search. While perhaps capable of doing so, his system does not currently learn meanings for multiple-word phrases, while ours does. His system also does not make any use of the order of the words in a sentence, while we use this information to construct the phrases that we consider adding to the lexicon. Finally, and perhaps most significantly, his system does not compute statistical correlations between words and their possible meanings, while ours does.

In the end, though, our main concern is with the relative learning performance of the two systems in different circumstances. The above differences might help us determine why the algorithms perform differently, but first we should examine whether they do perform differently. We turn to this issue in the next chapter.

Implementation

There are some details of the implementation not yet discussed. One consideration is that the ultimate use of the lexicon learned will be to add it to CHILL's overly-general parser, so that the lexicon will also be included in the final, learned parser. It is possible to find a lexicon that covers the input but does not allow the parsing of that input. The training parser that is used by CHILL in most of the experiments in the next chapter enforces a notion of *locality*, assuming that the representation of each phrase occurs in a similar position in the sentence as in the representation. More precisely, only contiguous elements on the parse stack can be combined.

For example, in "The girl ate the pasta with the cheese." with meaning as shown in Figure 3.1, an alternate interpretation function is f_1("girl") $= 8$, f_1("ate") $= 1$, f_1("pasta") $= 3$, f_1("the cheese") $= 2$, with lexicon
{("girl", `[cheese]`), ("ate", `[ingest]`), ("pasta", `[food, type:pasta, accomp:food]`),
("the cheese", `[person, sex:female, age:child]`)}. This would not allow parsing: due to the parser's locality constraints, `[cheese]` could not be embedded into `[food, type:pasta, accomp:food]` as needed because of the intervening structure that would be present on the parse stack during the (attempted) parse. Figure 3.11 shows the sequence of parse states. It is slightly more complicated than the simple shift-reduce sequence shown previously in Figure 2.2, because instead of shifting words onto the stack, we shift their meaning. We use "intro" instead of "shift" as the name of the associated action to highlight this difference. Note that some intro's simply move a word off of the input string with now associated change to the parse stack.

Action	Stack Contents	Input String
	[]	[the,girl,ate,the,pasta,with,the,cheese]
(intro)	[]	[girl,ate,the,pasta,with,the,cheese]
(intro)	[[cheese]]	[ate,the,pasta,with,the,cheese]
(intro)	[[ingest], [cheese]]	[the,pasta,with,the,cheese]
(intro)	[[ingest], [cheese]]	[pasta,with,the,cheese]
(intro)	[[food, type:pasta,accomp:food], [ingest], [cheese]]	[with,the,cheese]
(2 accomp)	[[ingest,accomp:[food,type:pasta,accomp:food]], [cheese]]	[with,the,cheese]
(intro)	[[ingest,accomp:[food,type:pasta,accomp:food]], [cheese]]	[the,cheese]
(intro)	[[ingest,accomp:[food,type:pasta,accomp:food]], [cheese]]	[cheese]
(intro)	[[person,sex:female,age:child], [ingest, accomp:[food,type:pasta,accomp:food]], [cheese]]	[]
(2 agt)	[ingest,agt:[[person,sex:female,age:child], accomp:[food,type:pasta,accomp:food]], [cheese]]	[]

(No more actions resulting in consistent parse states)

Figure 3.11: Unsuccessful Parse of "The girl ate the pasta with the cheese."

In order to help bias WOLFIE's search towards finding covering lexicons that also allow parsing of the training input, we added a component to the search that checks to confirm that covered sentences are parsable by CHILL's overly-general parser. If this is not the case, we know that some phrase in the sentence has a meaning that is not useful to CHILL. Therefore, whenever a sentence is covered, we check whether it can be parsed. If not, we retract the most recently learned pair, and adjust that phrase's candidate meanings to omit that meaning. We call this the *parsability heuristic*. This heuristic is another facet of our system that differs from the work of Siskind. While it is not crucial to learning good lexicons, it exploits information that Siskind's algorithm does not currently take advantage of, and could be very useful in domains with complex sentences.

As demonstrated by the example in this chapter, WOLFIE is able to initialize the lexicon with background knowledge such as closed class words or database constants. This enables the search to focus on phrases and portions of the representation that are uncovered by this knowledge. The only search parameter other then the weights on the heuristic components is the maximum number of pairings to consider between representations when finding LICS, which we set to 3 in our experiments, except where otherwise noted. We could find all pairings, as we did in the example, but this leads to excessive duplication in the LICS found, and we have found that a small sample is all that is needed to find good candidates.

Chapter 4

Wolfie Experimental Results: Learning for Database-Query Parsing

Our hypothesis is that useful and correct semantic lexicons can be learned by Wolfie. One way to test this is by comparing the learned lexicons to hand-built lexicons for the same domain. Another is to calculate the percentage of training or test sentences for which a covering lexicon has been learned. Finally, we could use the learned lexicons to assist a larger language learning system. We will discuss results from the latter two perspectives in this chapter, and from all three in the next.

4.1 Methodology and Data

For experiments in this chapter, we used the following experimental methodology. We split the data into disjoint training and test sets, and repeated this split for 10 different sets. To test the effectiveness of the lexicons learned by Wolfie in assisting a larger language learner, we did the following. After using the training set to learn a lexicon with Wolfie, that lexicon was then used as background knowledge for Chill, which used the same training set to learn a parser. The test examples were parsed using the learned parser, the resulting queries submitted to the database, the answers compared to those generated by the correct representation, and the percentage of correct answers recorded. By linking Wolfie and Chill, we demonstrate the utility of the learned lexicons for a real-world performance task. The gold standard for evaluating parsers for database queries is whether the system produces the correct answer to a given question; by comparing the answers retrieved from the database by a derived parse to the correct answer, we compare to this gold standard.

For all significance tests we used a two-tailed, paired t-test and a significance level of $p \leq 0.05$. The weights on the portions of the heuristic measure for Wolfie were as follows, except where otherwise noted: meaning utility: 10, generality: 1.0, orthographic overlap: 10. We used a maximum of 3 paired LICS per phrase except where otherwise noted, after finding that increasing the number of pairs does not significantly effect the results.

This chapter describes our experimental results on a database query application. The first corpus we discuss contains 250 questions about U.S. geography paired with logical representations. This domain was chosen due to the availability of an existing hand-build natural language interface to a simple geography database containing about 800 facts. The original interface, *Geobase*, was supplied with Turbo Prolog 2.0 (Borland International, 1988). The questions were collected from uninformed undergraduates and mapped into logical form by an expert. Examples from the corpus were given in the previous chapter. This corpus contains 71 words excluding closed class words and database constants. To broaden the test, we had the same 250 sentences translated into Spanish, Turkish, and Japanese. The Japanese translations are in word-segmented Roman orthography. Translated questions were paired with the appropriate logical queries from the English corpus. Appendix A shows examples of the input. The learning curves that follow were generated by choosing a random set of 25 test examples and then creating lexicons and parsers using increasingly larger subsets of the remaining 225 examples.

We compared our system to that developed by Siskind (1996). As discussed in Section 2.2, Siskind's system is an on-line (incremental) learner, while ours is batch. To make a closer comparison between the two, we ran his in a "simulated" batch mode, by repeatedly presenting the corpus 500 times, analogous to running 500 epochs to train a neural network. To use his system, written in Lisp, we had to modify the input slightly. For example, the representation
`largest(P,(capital(C),population(C,P)))` was changed to
`(largest2 pop (and (capital1 cap) (population2 cap pop)))`.
The numbers on predicate names are used to distinguish them from their arguments. Tokens were used in lieu of variables for the predicate arguments, since his system does not accept representations with variables in predicate arguments.

In this application there are many terms such as state and city names whose meanings are easily extracted from the database. Therefore, all tests below were run with such names given to the learner as an initial lexicon, although this is not required for learning in general.

4.2 Comparisons using English

The first experiment was a head-to-head comparison between WOLFIE and Siskind's system on the original English corpus. We did not give either system access to background knowledge about closed class words, such as **the**, but did give both access to knowledge of database constants, such as **austin**. Since Siskind has no measure of orthographic overlap, and it could arguably give our system an unfair advantage on these data, we ran WOLFIE with a weight of zero for this component. We also did not use the parsability heuristic or phrases of more than one word for this test. By making these adjustments, and by running his system in batch mode, we attempted to obtain the fairest head-to-head comparison between the two systems.

We also compare CHILL's performance using learned lexicons to CHILL's performance using a lexicon built by hand. This lexicon was developed by John Zelle when he tested CHILL in this domain, since at that time WOLFIE was not yet available. Our hypothesis is that parser acquisition will not be significantly adversely affected when using lexicons learned by WOLFIE as opposed to the hand-built lexicon. Part of the difference between the performance when CHILL is given hand-built versus learned lexicons may be due to the presence of words in the test set that are not in the training set. In the curves below, we show the effect of removing these items from the hand-built lexicon (labeled CHILL-testlex in all Figures), to show the success of learning as compared to building a lexicon by hand from the same set of examples.

Figure 4.1 shows learning curves for CHILL when using the lexicons learned by the "ablated"

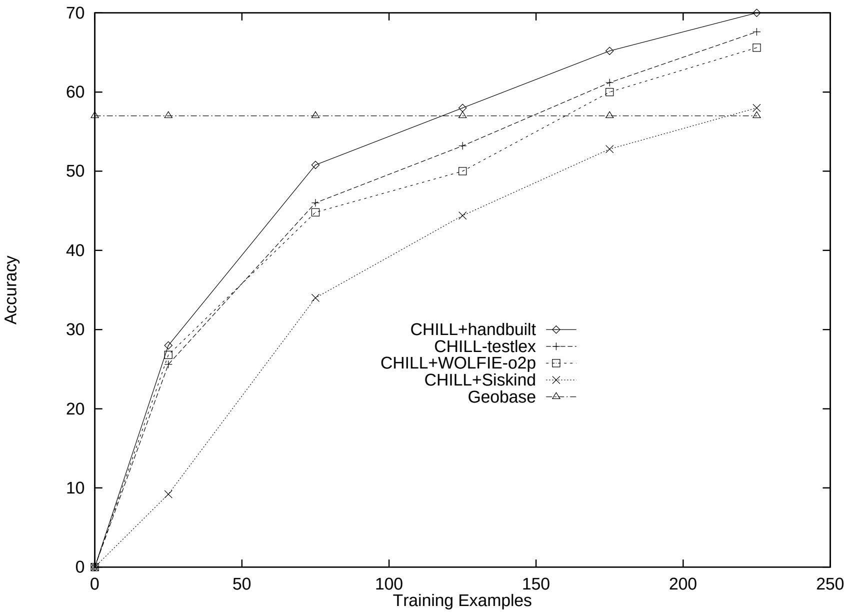


Figure 4.1: Accuracy on English Geography Corpus

version of WOLFIE (CHILL+WOLFIE-o2p, where "-" stands for the subtraction of features, and o, 2, and p, stand for orthographic overlap, two-word phrases, and the parsability heuristic, respectively) and by Siskind's system (CHILL+Siskind). The uppermost curve (CHILL+handbuilt) is CHILL's performance when given the full hand-built lexicon. Finally, the horizontal line shows the performance of the *Geobase* benchmark.

The results show that a lexicon learned by WOLFIE led to parsers that were almost as accurate as those generated using a hand-built lexicon. The best accuracy is achieved by the hand-built lexicon, followed by the hand-built lexicon with words in the test set removed, followed by WOLFIE, followed by Siskind's system. All the systems do as well or better than *Geobase* by 225 training examples. The differences between WOLFIE and Siskind's system are statistically significant at all training example sizes except 125. These results show that WOLFIE can learn lexicons that lead to successful learning of parsers, and that are better from this perspective than those learned by a competing system. Also, comparing to the CHILL-testlex curve, we see that most of the drop in accuracy from a hand-built lexicon is due to words in the test set that the system has not seen during training.

Finally, we also ran Siskind's system in the standard incremental mode. His system run in batch mode on this test averaged 58% accuracy at 225 examples, versus incremental mode which attained 49.2% accuracy, giving evidence that batch mode does improve his system.

As noted above, these tests were run with the meaning of database constants provided as background knowledge. Next, we examined the effect of also providing closed-class words as background knowledge. Figure 4.2 shows the resulting learning curves. For these tests, we also show the effect of including the orthographic overlap and parsability heuristics for WOLFIE, as well as the ability to learn phrases of both one and two words (CHILL+WOLFIE). Also, for both versions of WOLFIE we increased the maximum number of paired LICS per word to 10. The additional background knowledge and the parsability heuristic make little difference in the overall

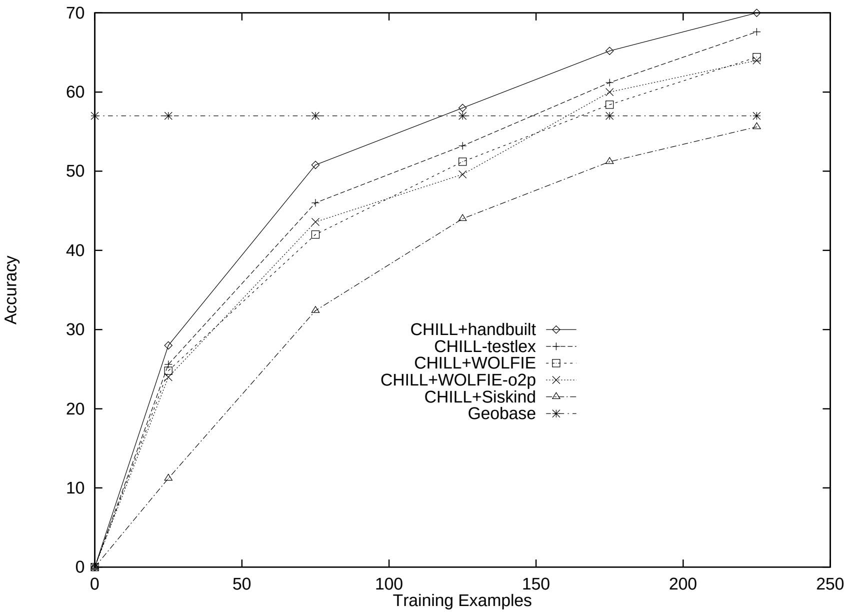


Figure 4.2: Accuracy Given Closed Class Words

performance of the learned parser. The differences between Siskind's system and WOLFIE without parsing, phrases, or orthographic overlap are statistically significant at all example set sizes. On the other hand, the differences between the parsers learned when given the hand-built lexicon but *not* lexicon items appearing only in the test set (CHILL-testlex), versus parsers learned with the ablated WOLFIE, are not statistically significant at any data point.

One of the implicit hypotheses of our approach is that coverage of the training pairs implies a good lexicon. We measured the coverage of a sentence/representation pair by counting the percent of the terms and predicates in the pair's representation that could be covered by learned meanings of the phrases in its sentence. We now compare the coverage of WOLFIE's lexicons to those of Siskind's and verify that WOLFIE's have better coverage. For the first experiment above, WOLFIE's lexicons covered 100% of the 225 training examples on average, while Siskind's covered 78.1%. For the second experiment, the coverages were 100% (for the best version of WOLFIE) and 94.5%, respectively. This may account for some of the performance difference between the two systems.

Further differences may be explained by the percentage of training examples usable by CHILL, which is the percentage parsable by its overly-general parser. For the first experiment, CHILL could use 95.7% of the 225 examples when given the lexicons learned by WOLFIE but only 79.5% of the examples when given lexicons learned by Siskind's system. When the lexicon learners are given closed class words, these percentages rise to 98.7% and 84.6%, respectively. These differences most likely contribute to the differences seen in the generalization accuracy of CHILL, since using Siskind's lexicons effectively reduces the number of control examples input to CHILL's induction phase (see Figure 2.1). Though this is on the training sentences and the previously discussed results were on test sentences, these differences are comparable to the differences on the final accuracy of the parsers learned by CHILL.

Finally, we were disappointed that adding the parsability heuristic to WOLFIE did not lead to improved performance of the learned lexicons in aiding CHILL: the differences between WOLFIE

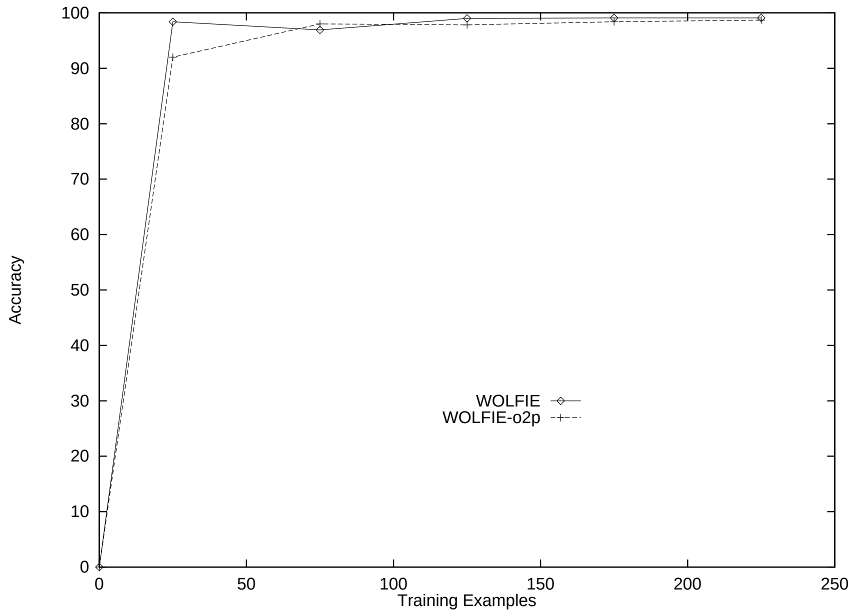


Figure 4.3: Train Set Parsing Accuracy

and WOLFIE-o2p were not statistically significant at any training set size. We investigated the performance of the lexicons in aiding the parsing of the *training* sentences, during CHILL's example analysis phase. Figure 4.3 shows the percent of these sentences that were successfully parsed during CHILL's example analysis stage. The WOLFIE curve shows the accuracy with lexicons learned with all options added to WOLFIE's heuristics, versus WOLFIE-o2p, which shows the accuracy with lexicons learned without overlap, parsability, or phrases. The accuracy of the training parser using the lexicons learned by WOLFIE with all options is statistically significantly better than the ablated WOLFIE at 125 and 225 examples. This indicates that all of the benefits of lexicon learning may not be reflected in the final parser learned by CHILL in all cases. However, while the differences were statistically significant, they were very small, and no ultimate advantage is reflected in the final learned parsers. In sum, the good results for both versions show that the heuristic measures alone, without parsability, are sufficient to learn a lexicon that is useful for parsing. With a different corpus, perhaps the parsability heuristic would be needed to help focus the search.

4.3 Comparisons using Spanish

Next, we examined the performance of WOLFIE and Siskind's system on the Spanish version of the corpus. For Spanish, as well as the other languages discussed below, some minor changes were required in CHILL's training parser. As mentioned in Section 2.1, the overly-general parser uses a compatibility check with the correct parse of a sentence to limit the search for control examples. The check had to be broadened because of the word-order differences among the various languages. In other words, the locality assumption no longer holds for these languages in general, since we use the original representation from the English sentences as the basis for our compatibility check.

For the comparison with Siskind's system, we again omitted orthographic overlap, two-word phrases, and the parsability heuristic from WOLFIE. Figure 4.4 shows the results. In these tests,

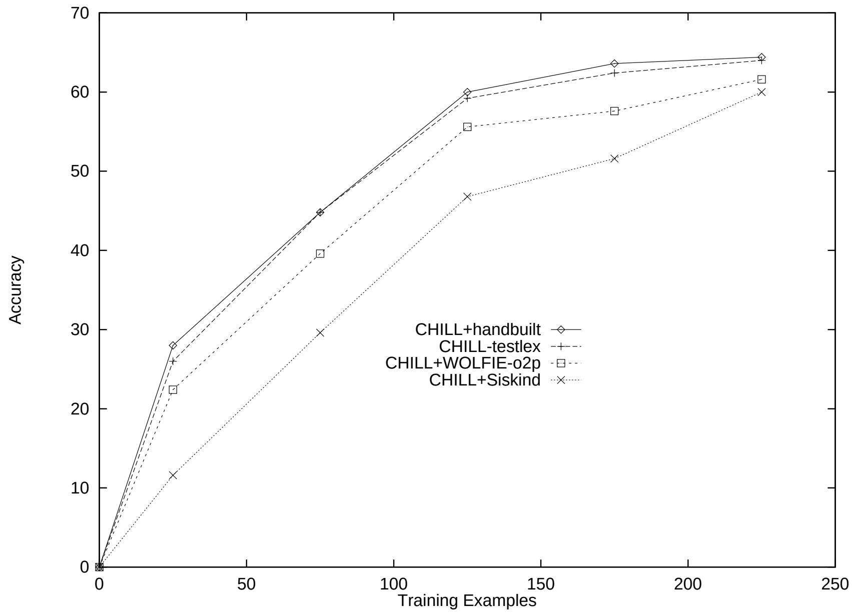


Figure 4.4: Accuracy on Spanish

we also gave closed class words to the lexicon learners as background knowledge. The performance difference between WOLFIE-o2p and CHILL-testlex are not statistically significant at any training set size, while the differences between WOLFIE-o2p and Siskind are statistically significant except at 225 examples.

4.4 Accuracy on Other Languages

We also had the geography query sentences translated into Japanese and Turkish, and ran similar tests to determine how well WOLFIE could learn lexicons for these languages, and how well CHILL could learn to parse them. We did not run Siskind's algorithm on the other two languages. Figure 4.5 shows the results. For all four of these tests, we used the parsability heuristic, but did not give the learner access to the closed class words of any of the languages. We also set the weight of the orthographic overlap heuristic to zero for all four languages, since this gives little advantage in the foreign languages. The performance differences among the four languages are quite small, demonstrating that our methods are not language dependent. This further demonstrates that the orthographic overlap portion of the heuristic is not needed to learn useful lexicons.

4.5 LICS versus Fracturing

We would like to examine how the various components of the algorithm contribute towards its success. One component not yet examined is the candidate generation method. As mentioned in Section 3.2, we could use fractures of representations of sentences in which a phrase appears to generate the candidate meanings for that phrase. We used this approach and compared it to using the standard method of using the largest isomorphic connected subgraphs of sampled pairs of representations as candidate meanings. To attempt a more fair comparison, we also sampled

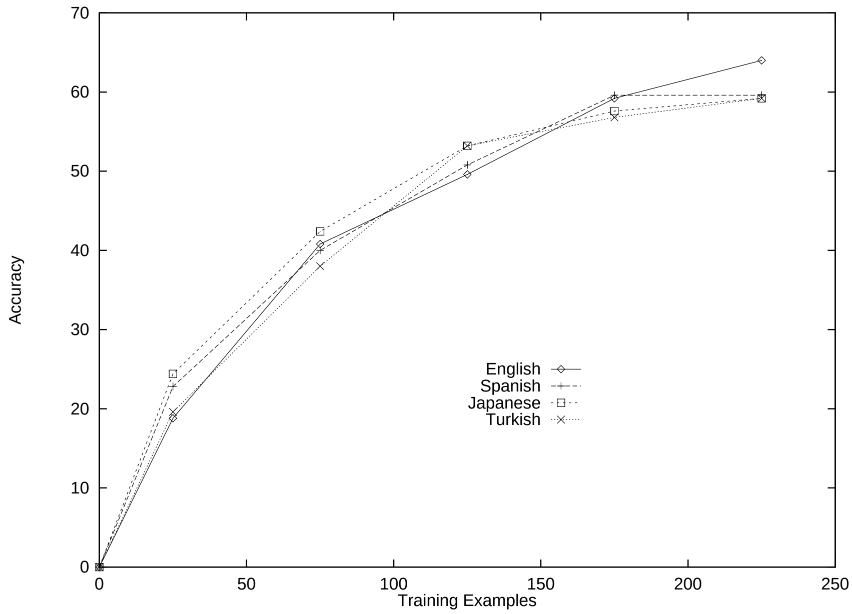


Figure 4.5: Accuracy on All Four Languages

representations from which to fracture, using the same number as the number of pairs sampled for LICS.

The accuracy of CHILL when using the resulting learned lexicons as background knowledge are shown in Figure 4.6. Using fracturing shows a small gain at first, but there is little or no advantage as more training examples are seen; none of the differences between the two systems are statistically significant. In addition, the number of initial candidate lexicon entries from which to choose is much larger for fracturing than our traditional method, as shown in Figure 4.7. Finally, the learning time when using fracturing is greater than that when using LICS, as shown in Figure 4.8, where the CPU time is shown in seconds.

In summary, these differences show the utility of LICS as a method for generating candidates: a more thorough method does not result in better performance, and also results in longer learning times. In a domain with larger representations, the differences in learning time would likely be even more dramatic.

4.6 A Larger Corpus

We next ran some experiments on a larger, more diverse corpus of sentences from the geography domain, where the additional sentences were collected from computer science undergraduates. The set of questions in the experiments discussed up until now was collected from students in the German department, with no special instructions on the complexity of queries desired. The computer science students tended to ask more complex queries than the original questions: their task was to give 5 sentences and their logical query representation for a homework assignment. They were requested to give at least one sentence whose representation included a predicate containing embedded predicates, for example `largest(S, state(S))`, and we asked for variety in their sentences. There were 221 new sentences, for a total of 471 (including the original 250 sentences). Examples

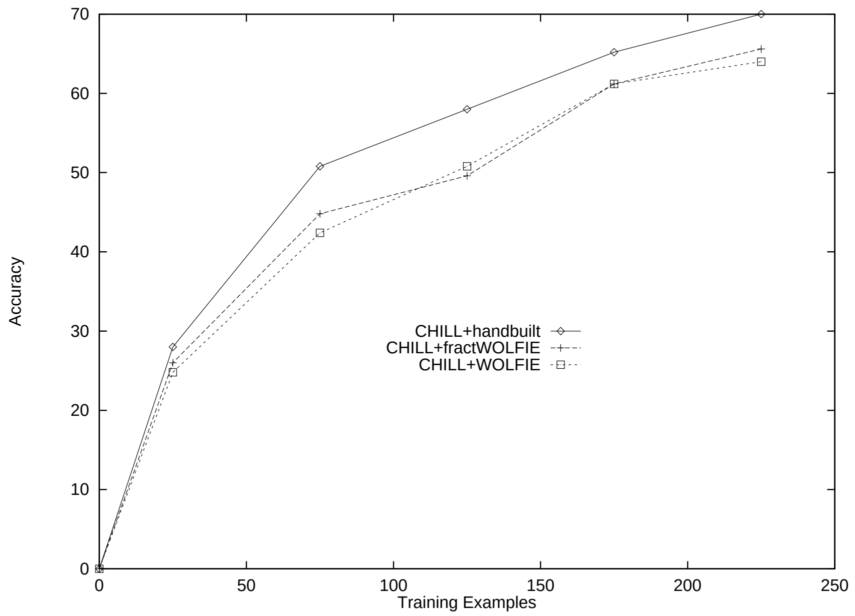


Figure 4.6: Fracturing vs. LICS: Accuracy

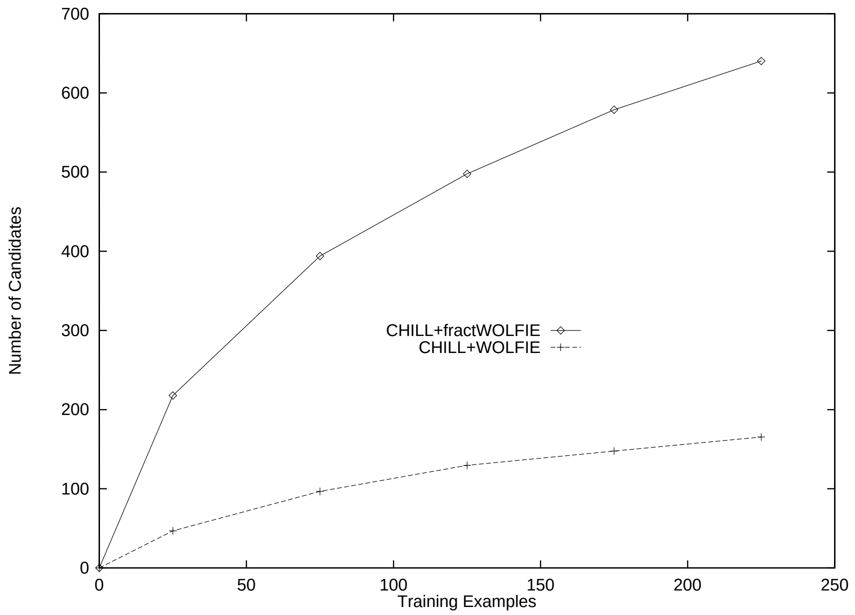


Figure 4.7: Fracturing vs. LICS: Number of Candidates

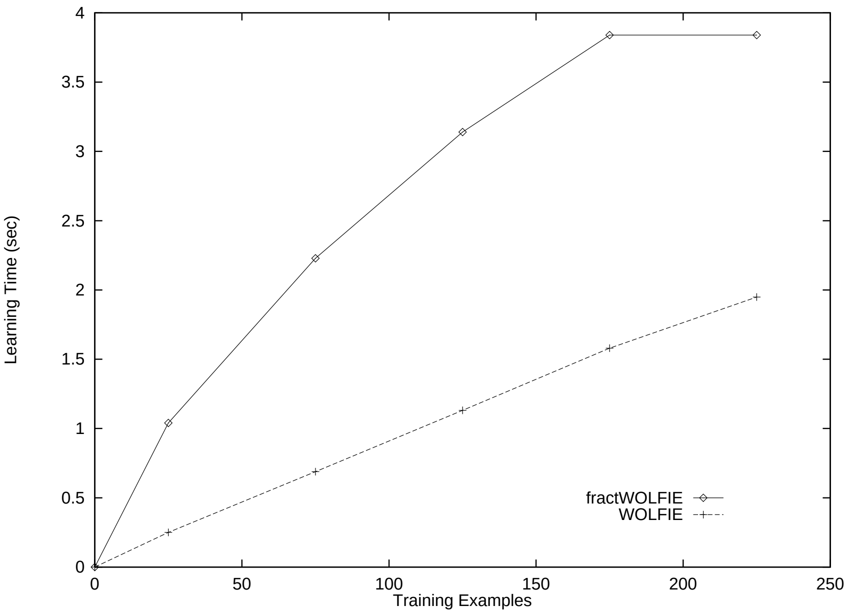


Figure 4.8: Fracturing vs. LICS: Learning Time

of the new queries are given in Appendix A.

For these experiments, we split the data into 425 training sentences and 46 test sentences, for 10 random splits, then trained WOLFIE and then CHILL as before. We did not compare our system to Siskind's in this case, assuming that the results would be comparable to those above. Our goal was simply to see whether WOLFIE was still effective for this larger corpus, since there were approximately 40 novel words in the new sentences. Therefore, we tested against the performance of CHILL with an extended hand-built lexicon. For this test, we gave the system access to background knowledge about closed class words, and gave a weight of zero to the orthographic overlap component of the heuristic. We also did not use the parsability heuristic or phrases of more than one word for this test, since these do not seem to make a significant difference in this domain.

Figure 4.9 shows the resulting learning curves. None of differences between CHILL and WOLFIE are statistically significant, probably because the difficulty of parsing overshadows errors in word learning. Also, the improvement of machine learning methods over the Geobase hand-built interface is much more dramatic for this corpus than for the smaller one.

4.7 Discussion

These results show that WOLFIE can successfully learn lexicons that are useful for real natural language tasks. The differences between WOLFIE and Siskind's system in this domain are probably due to several factors. First, as Siskind's system does not use constraints between word meanings to focus its search, it tended to learn lexicon entries that were overly specific. It was also more likely to learn multiple senses for words that were actually not ambiguous in this corpus. His system also seemed to have difficulty handling the skolemization of the input that we performed in order to make the representations variable-free. Perhaps additional tuning of the parameters of his system would improve the results slightly, but we did try several variants and reported the best results

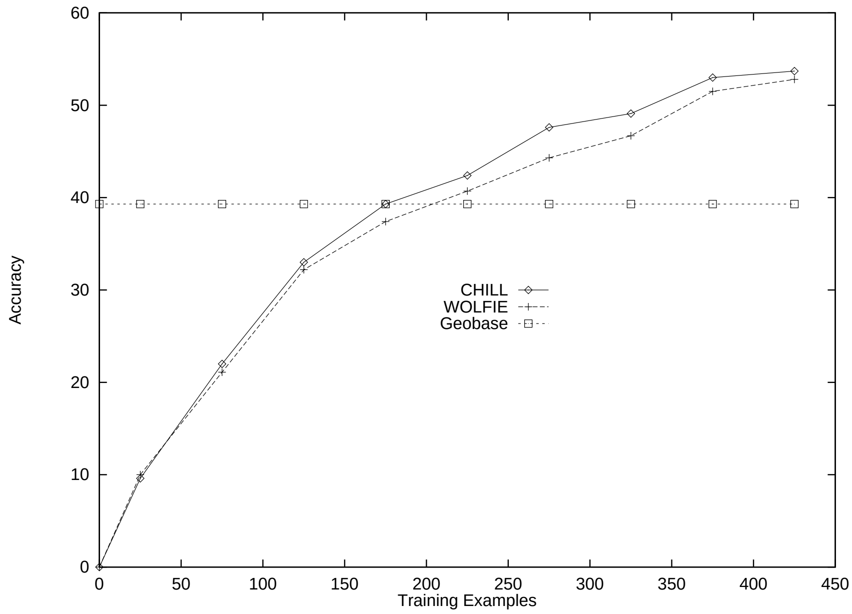


Figure 4.9: Accuracy on the Larger Geography Corpus

above. Examples of lexicons learned by each system are given in Appendix B.

Another factor to consider is learning time. While it is difficult to directly compare the two systems since one is written in Prolog and the other in Lisp, the learning time of the two systems is approximately the same if Siskind's system is run in incremental mode, just a few seconds with 225 training examples. If we run Siskind's system in batch mode, the time to learn a lexicon is far greater with his system than with ours, around 1000 seconds of CPU time for 250 iterations and 225 examples.

Finally, it is gratifying, while a bit surprising, to see that orthographic overlap is not needed to learn good lexicons. We thought that at least for small numbers of training examples, it would benefit the learner. This is not demonstrated in these experiments, perhaps because not all of the desired lexicon entries have a tight correspondence between the characters in their phrase and the characters in their meaning.

Chapter 5

Wolfie Experimental Results: Artificial Corpora

The previous chapter showed that Wolfie successfully learns lexicons for a real-world corpus. However, this demonstrates success on only a relatively small corpus and with one representation formalism. We now show that our algorithm scales up well with more lexicon items to learn, more ambiguity, and more synonymy. This is difficult to control when using real data as input. Also, there are no large corpora available that are annotated with semantic parses. We therefore present here experimental results on two artificial corpora.

The first corpus demonstrates the ability of Wolfie to learn representations of a different form than a logical query representation, that of a case-role representation (Fillmore, 1968) augmented with Conceptual Dependency (CD) (Schank, 1975) information. The sentences, however, are examples of real English sentences. In the second corpus, both the sentences and their representations are completely artificial, and the sentence representation is a variable-free representation, as suggested by the work of Jackendoff (1990) and others.

5.1 Case-Role Plus Conceptual Dependency

In a traditional case-role representation the sentence "The man ate the cheese." is represented by
`[ate, agt:[man, det:the], pat:[cheese, det:the]]`.
where `agt` denotes the agent of the action `ate` and `pat` the patient. But in the CD-like representations used here, the same sentence is represented by
`[ingest,agt:[person,sex:male,age:adult],pat:[food,type:cheese]]`.
This representation was chosen to demonstrate the learning of complicated structures as word meanings. The corpus used here is based on that of McClelland and Kawamoto (1986). This corpus consists of 1475 sentence/case-structure pairs containing 30 unique words. The sentences were artificially produced from a set of 19 templates. The case-structure portion of the original pairs was modified as illustrated above. The resulting corpus is hereafter referred to as the modified M & K corpus.

Finding LICS for two sentence representation in the modified M & K representation is the same as finding them for standard labeled trees, taking into account that both the nodes and the arcs are labeled. For example, given the representations

`[ingest, agt:[person,sex:male,age:adult], pat:[food,type:cheese]]`
and
`[propel, instr:[inst,type:ball], pat:[person,sex:male,age:child]]`,
the LICS is `[person, sex:male]`.

In the following experiments, a random set of 650 unique training sentences was chosen, starting with 50 examples, and incrementing by 100, for each of five trials. The weights on the portions of the heuristic measure for WOLFIE were 20 for meaning utility, zero for orthographic overlap, and 2 for generality. For this test, we gave the system information about closed class words, and did not use the parsability heuristic or two word phrases. Finally, we used a maximum of 3 paired LICS per word.

The first evaluation of the success of the system was made by measuring the percentage of "correct" word meanings learned, when compared to a hand-built lexicon. There are several ways to calculate this metric. The one used here is as follows. First, a *partial correctness* score is given to each learned pair. This is done by measuring how similar the learned representation for a word is to a correct one. For example, if a correct meaning for a word is `[person, sex:female, age:adult]`, the learned meaning `[person, sex:female]` is closer to being correct than `[person]`. There are several ways to measure this similarity. Our approach is the following. Each meaning learned for a word is compared to each correct meaning for that word, and the following score is computed. If the roots match, this contributes 1/3 credit towards the score. Assuming the roots match, then if all arc labels from the root match, this contributes another 1/3 towards the score. Correct children under all these matching labels contribute the final 1/3. If these children do not completely match, their partial correctness is computed recursively. For example, a learned meaning of `[person,sex:male]` compared to a correct meaning of `[person,sex:female,age:adult]` scores 1/3 for the matching root, (1/2*1/3) for one correct label out of two possible, and scores zero for no matching children under the matching label, for a total score of 0.5. The best such score when compared to all correct meanings is the partial correctness score for each $(word, meaning)$ pair learned.

Next, for each word in the hand-built lexicon, the partial *intersection accuracy* for that word is calculated. The formula for calculating this is $(P/N + P/C)/2$, where P is the partial correctness score calculated as in the previous paragraph summed over all learned meanings for the word, N is the number of meanings learned for the word, and C is the number of correct meanings for the word. This intersection accuracy takes into account two types of errors: learning an incorrect meaning for a word, and failing to learn a correct meaning for a word. It is the average of *precision* and *recall*, with respect to the partial correctness score. These are two measures commonly used in the Computational Linguistics community. Precision measures the percentage of the lexicon entries (i.e., (word, meaning) pairs) that the system learns that are correct. Recall measures the percentage of the lexicon entries in the hand-built lexicon that are correctly learned by the system.

$$precision = \frac{\#\text{ correct pairs}}{\#\text{ pairs learned}}$$

$$recall = \frac{\#\text{ correct pairs}}{\#\text{ pairs in hand-built lexicon}}$$

The intersection accuracy of a learned lexicon is the average intersection accuracy of all word forms in the corpus.

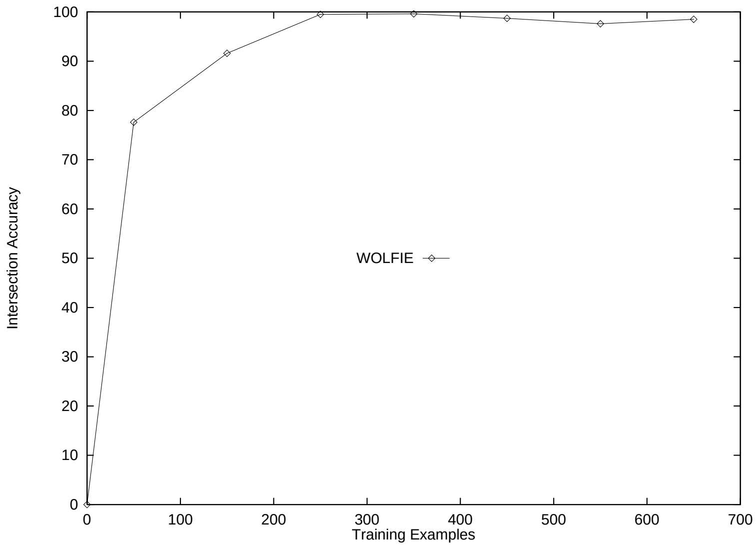


Figure 5.1: Intersection Accuracy for the Modified M & K Corpus

Figure 5.1 shows the average intersection accuracy of the lexicons learned by WOLFIE for the modified M & K corpus. In four of the trials 100% accuracy was obtained at 450 examples, offset by a trial with 93.3% accuracy.

Next, the learned lexicons were used to assist CHILL in its learning process. With the original M & K corpus, the shift operation specified a shift of each word token directly onto the parse stack from the input stack, as illustrated in Figure 2.2. In the new representation, as a word token is shifted off the input stack, its meaning is shifted onto the parse stack, as in Figure 3.11. In the past, these "introduce" operations had to be generated by hand. In this study, we use the lexicons learned by WOLFIE as the basis of the introduce operators input as background knowledge by CHILL.

As in the geography domain, the the generalization accuracy of the learned parsers was tested. After training CHILL on a subset of the corpus, novel sentences were given to the learned parsers to determine whether they could be parsed correctly. As with the lexicon, two types of errors can occur: an incorrect analysis can be generated, or a correct analysis can fail to be generated. The intersection accuracy can be measured in terms of either the partial or exact correctness of the produced analyses, where the partial correctness is computed in the same manner as the partial correctness of word meanings above. In the following, the exact correctness is used. In this case, intersection accuracy is $(C/D + C/A)/2$, where C is the number of produced analyses that completely match a correct analysis, D is the number of distinct analyses produced, and A is the number of correct analyses.

Figure 5.2 shows the resulting accuracies. The topmost curve shows the accuracies of the parsers when trained and tested on the original M & K corpus. The next curve shows the results when CHILL is trained and tested on the modified M & K corpus, and given the hand-built lexicon as background knowledge. Finally, the bottom curve shows the accuracy on the modified M & K corpus when CHILL is given the lexicons learned by WOLFIE as background knowledge. CHILL when run with the original M & K corpus obtained 97.4% accuracy at 650 examples, while CHILL with the modified M & K corpus had more difficulty. If provided with a correct lexicon, it obtained 92.5%

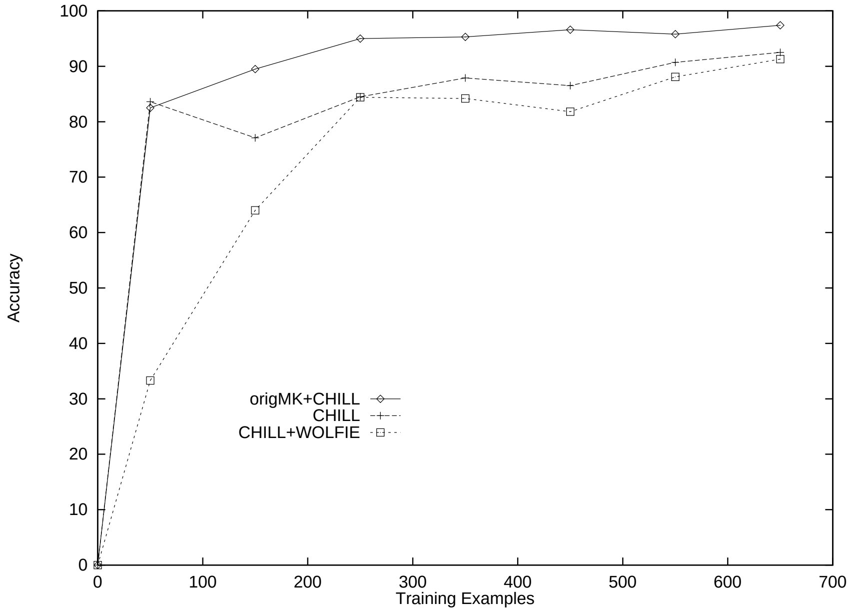


Figure 5.2: CHILL Accuracy on the M & K Corpus

accuracy, and when provided with the partially correct lexicon learned by WOLFIE, it obtained 91.3% accuracy. CHILL with the hand-built lexicon is only statistically significantly[1] better than CHILL using WOLFIE's learned lexicons when given 50 examples, but not at the other points on the learning curve.

5.2 A Second Set of Artificial Corpora

This section discusses results on a series of artificially generated corpora.[2] For each corpus, a random lexicon mapping words to simulated meanings was first constructed. This *original* lexicon was then used to generate a corpus of random utterances each paired with a meaning representation. After using this corpus as input to WOLFIE, the learned lexicon was compared to the original lexicon, and *weighted precision* and *weighted recall* were measured. Precision and recall were defined in the previous section. To get weighted measures, we then weight each pair in these formulas by the word's frequency in the entire corpus (not just the training corpus). This models how likely we are to have learned the correct meaning for an arbitrarily chosen word in the corpus.

We generated several lexicons and associated corpora, varying the ambiguity rate (number of meanings per word) and synonymy rate (number of words per meaning). Meaning representations were generated using a set of "conceptual symbols" that combined to form the meaning for each word. The number of conceptual symbols used in each lexicon will be noted when we describe each corpus below. In each lexicon, 47.5% of the senses were variable-free to simulate noun-like meanings, and 47.5% contained from 1 to 3 variables to denote open argument positions to simulate verb-like meanings. The remainder of the words (the remaining 5%) had the empty meaning to simulate function words. In addition, the functors in each meaning could have a depth of up to 2 and an arity

[1] again, we used a two-tailed, paired t-test and a significance level of $p \leq 0.05$.

[2] Thanks to Jeff Siskind for the initial corpus generation software, which we enhanced for these tests.

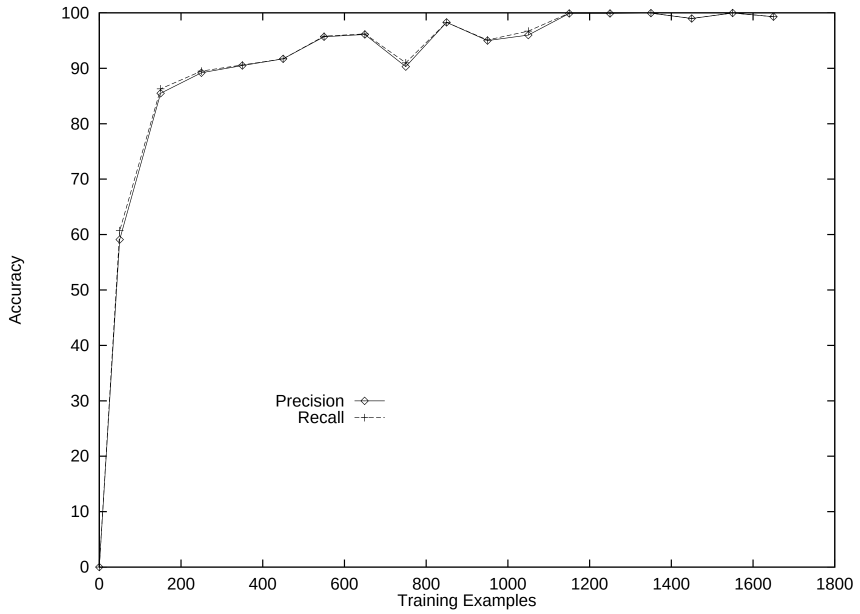


Figure 5.3: Baseline Artificial Corpus

of up to 2. An example noun-like meaning is `f23(f2(f14))`, and a verb-meaning `f10(A,f15(B))`; the conceptual symbols in this example are `f23`, `f2`, `f14`, `f10`, and `f15`. By using these multi-level meaning representations we demonstrate the learning of more complex representations than those in the geography database domain: none of the hand-built meanings for phrases in that lexicon had functors embedded in arguments. A grammar was used to generate utterances and their meanings from each original lexicon, with terminal categories selected using a distribution based on Zipf's Law (Zipf, 1949). Under Zipf's Law, the occurrence frequency of a word is inversely proportional to its ranking by occurrence.

We started with a baseline corpus generated from a lexicon of 100 words using 25 conceptual symbols and no ambiguity or synonymy; 1949 sentence/meaning pairs were generated. We split this into 5 training sets of 1700 sentences each. NOTE: Talk about the weights of the learning parameters. Figure 5.3 shows the weighted precision and recall curves for this initial test. This demonstrates good scalability to a slightly larger corpus and lexicon than that of the U.S. geography query domain.

A second corpus was generated from a second lexicon, also of 100 words using 25 conceptual symbols, but increasing the ambiguity to 1.25 meanings per word. This time, 1937 pairs were generated and the corpus split into 5 sets of 1700 training examples each. Weighted precision at 1650 examples drops to 65.4% from 99.3%, and weighted recall to 58.9% from 99.3%. The full learning curve is shown in Figure 5.4. A quick comparison to Siskind's performance on this corpus confirmed that his system achieved comparable performance, showing that with current methods, this is close to the best performance that we are able to obtain on this more difficult corpus. One possible explanation for the smaller performance difference between the two systems on this corpus versus the geography domain is that in this domain, the correct meaning for a word is not necessarily the most "general," in terms of number of nodes, of all its candidate meanings. Therefore, the generality portion of the heuristic may negatively influence the performance of

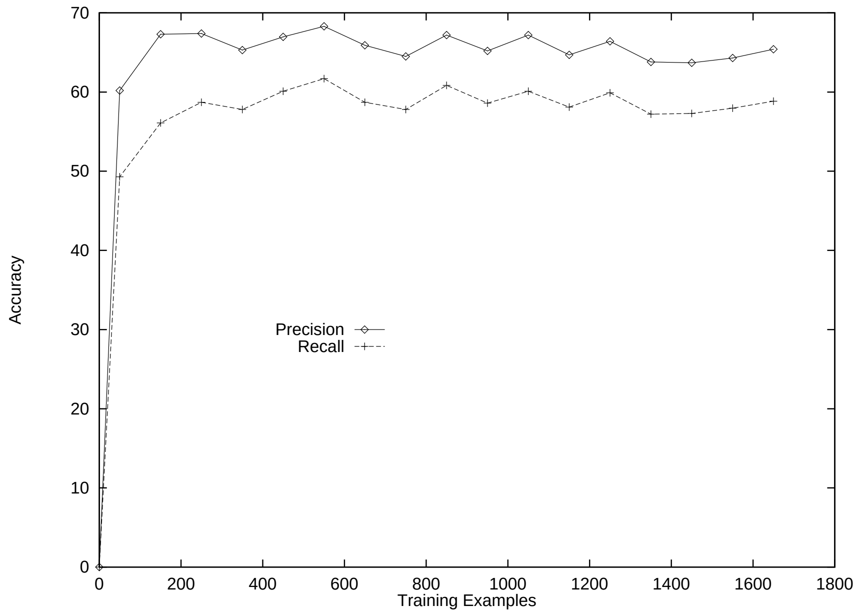


Figure 5.4: A More Ambiguous Artificial Corpus

Ambiguity Level	*Number of Examples*
1.0	150
1.25	450
2.0	1450

Table 5.1: Number of Examples to Reach 75% Precision

WOLFIE in this domain.

Finally, we show the change in performance with increasing ambiguity and increasing synonymy. Figure 5.5 shows the weighted precision and recall at 1050 examples at increasing levels of ambiguity, holding the synonymy level constant. Figure 5.6 shows the results at increasing levels of synonymy, holding ambiguity constant. Increasing the level of synonymy does not effect the results as much as increasing the level of ambiguity, which is as we expected. Having multiple competing meanings for a word decreases the amount of evidence available for each meaning, making the learning task more difficult. On the other hand, increasing the level of synonymy does not have the potential to mislead the learner.

The number of training examples required to reach a certain level of accuracy is also informative. In Table 5.1, we show the point at which a standard precision of 75% was first reached for each level of ambiguity. Note, however, that we only measured accuracy after each set of 100 training examples, so the numbers in the table are approximate.

A second, more rigorous test of scalability was performed on two corpora generated from lexicons an order of magnitude larger than those in the above tests. In these tests, we use a lexicon containing 1000 words and using 250 conceptual symbols. We give results for a corpus with no ambiguity and a corpus generated from a lexicon with ambiguity and synonymy similar to that

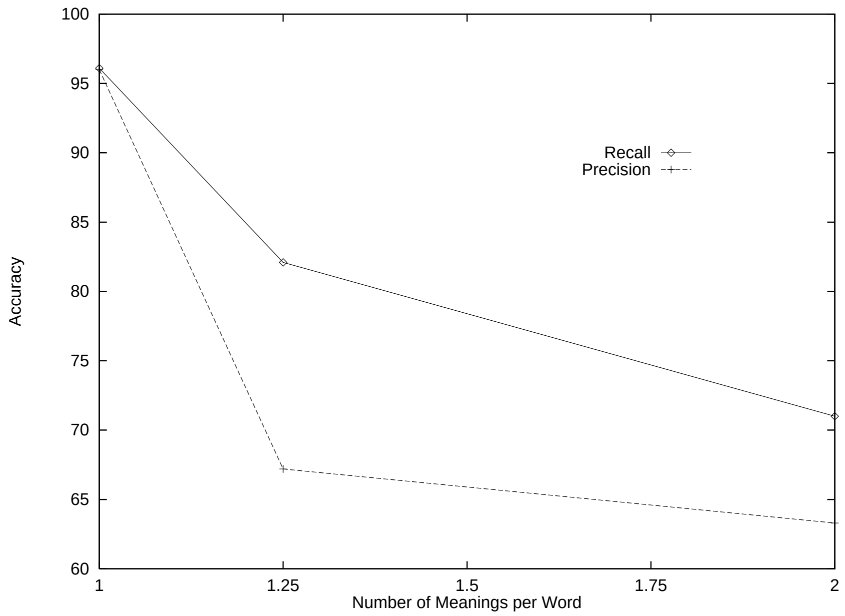


Figure 5.5: Increasing the Level of Ambiguity

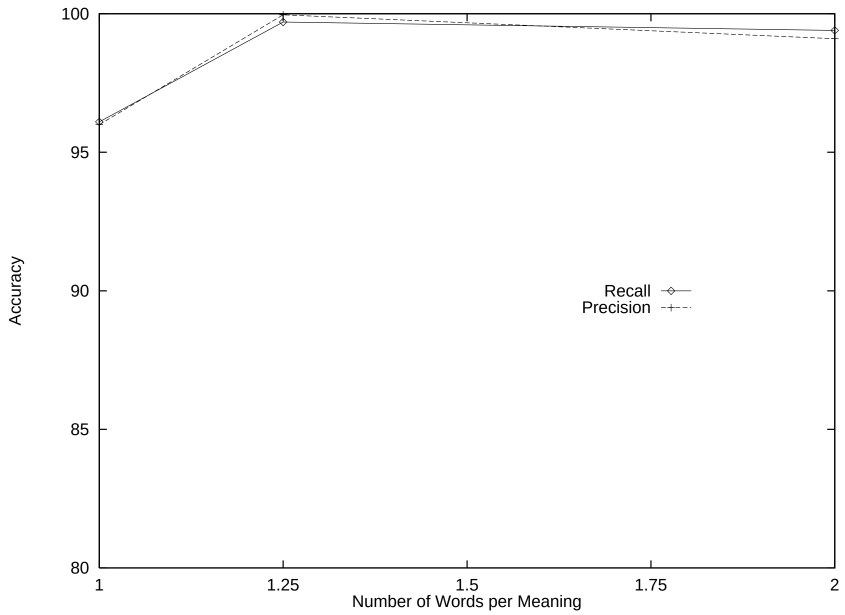


Figure 5.6: Increasing the Level of Synonymy

found in the WordNet (Beckwith et al., 1991) database: the ambiguity there is approximately 1.68 meanings per word and the synonymy 1.3 words per meaning. These corpora contained 9904 and 9948 examples, respectively, and we split the data into five sets of 9000 training examples each. For the easier large corpus, the maximum average of weighted precision and recall was 85.6%, at 8100 training examples, while for the harder corpus, the maximum average was 63.1% at 8600 training examples.

5.3 Discussion

The results on the modified M & K corpus demonstrate the ability of WOLFIE to learn lexicons for a representation formalism different than the logical query representation of the previous chapter. We showed both the intersection accuracy of learned lexicons and the effectiveness of the learned lexicons in aiding CHILL to learn parsers. Intersection accuracy seems to be a fairly good measure of how close the learned meanings for a word are to the desired meanings. An alternative is to measure the percentage of words that were completely correctly learned, which is also high for this corpus. For example, at 650 examples, an average of 98.1% of the words were completely correct, compared to an average intersection accuracy of 98.5%.

The results on the second set of artificial corpora were more mixed. Ambiguity is difficult to handle but not impossible. By increasing the size of the training set, similar accuracies can by reached when ambiguity is present as compared to when it is not. Increasing the size of the lexicons to learn also decreases the accuracy of the learned lexicons, but not drastically. In addition, since for at least one corpus our results were comparable to those of Siskind's system, we feel that these difficulties exist for any state-of-the-art system. Learning reliably in ambiguous situations may simply require more information than is available in the data alone. For example, it is hypothesized that children take advantage of prosody and context when learning word meanings.

Finally, learning times on these artificial corpora are quite low. For example, the average learning time was 11 seconds CPU time for 1050 sentences for the corpus with 2.0 meanings per word. In summary, these results show that WOLFIE scales up reasonably well to large training set sizes containing synonymy and ambiguity. They also demonstrate that WOLFIE is general enough to apply to more than one kind of semantic representation formalism. Further research is warranted to investigate improving the accuracy of the system for highly ambiguous corpora.

Chapter 6

Active Learning

As indicated in the previous chapters, we have built an integrated system for language acquisition that is flexible and useful. However, a major difficulty remains: the collection of training corpora. Though annotating sentences is still arguably less work than building an entire system by hand, the annotation task is also time-consuming and error-prone. Further, the training pairs often contain redundant information. We would like to minimize the amount of annotation required while still maintaining good generalization accuracy.

To do this, we turned to methods in *active learning.* Active learning is an emerging research area in machine learning that features systems that automatically select the most informative examples for annotation and training (Cohn et al., 1994). The primary goal of active learning is to reduce the number of examples that the system is trained on, while maintaining the accuracy of the acquired information. Active learning is particularly attractive in natural language learning tasks, since there is an abundance of unannotated text, and we would like to only annotate the most informative sentences.

In this chapter, we explore the use of active learning for natural language interfaces, and demonstrate that with active learning, fewer examples are required to achieve the same accuracy obtained by training on randomly chosen examples. By turning to active learning, we change the focus of discussion, which until now has been primarily on lexicon acquisition. We investigate active learning methods for parser acquisition, lexicon acquisition, and the combination of both.

6.1 Background

There are only a few NLP tasks to which active learning methods have been applied previously in the literature, including part of speech tagging (Dagan & Engelson, 1995), text categorization (Lewis & Catlett, 1994; Liere & Tadepalli, 1997), and information extraction (Califf, 1998). The first two of these are traditional classification tasks, while the tasks we address here, and those in Califf (1998) are more complex. The application of active learning to tasks requiring complex outputs has only been briefly investigated. Our goal with this research is to investigate whether and how active learning methods can be useful in non-classification domains.

Active learning includes two general approaches, the first using *membership queries*, and the second using *selective sampling.* When membership queries are used, the learner constructs examples that are then labeled by an expert (Angluin, 1988). Because of the difficulty of constructing such examples in the case of natural language systems, and the ease of obtaining text, we focus here on selective sampling methods. In selective sampling, we attempt to choose for annotation, from unlabeled examples, those that are most informative for learning. Past work in this area has emphasized two approaches, *certainty-based* methods (Lewis & Catlett, 1994; Cohn et al., 1994), and *committee-based* methods (Liere & Tadepalli, 1997; Dagan & Engelson, 1995; Freund, Seung, Shamir, & Tishby, 1997).

In the certainty-based paradigm, the system is trained on a small number of annotated examples to learn an initial classifier. Next, the system examines unannotated examples and attaches certainties to the predicted annotation of those examples, based on the estimates of this initial classifier. The k examples with the lowest certainties are then presented to the user for annotation, and then to the system for retraining. Several methods for attaching certainties have been used, but they typically attempt to estimate the probability that a classifier consistent with the prior training data will classify a new example correctly.

In the committee-based paradigm, a diverse committee of classifiers is created, again from a small number of examples. Next, each committee member attempts to annotate additional examples. The examples most informative to the learner are those with the most conflict in committee annotation, so they are presented to the user for annotation, and to the system for retraining. A diverse committee, but one in which each member is consistent with the prior training data, will produce the highest disagreement on examples that are most uncertain with respect to the possible classifiers that could be obtained by training on that data.

Figure 6.1 gives the high level pseudocode for both certainty-based and committee-based selective sampling. In an ideal situation, k would be set to one to make the most informed decisions in future choices, but to maintain efficiency in application to batch learning systems, it is typically set higher than one.

```
Apply the learner to n annotated bootstrap examples,
  creating one classifier or a committee of them.
Until there are no more unannotated examples or
  the expert is unwilling to annotate more examples, do:
    Use the most recently learned classifier/committee to analyze each unannotated example.
    Find the k examples with the lowest annotation
      certainty/most disagreement among committee members
    Have the expert annotate these examples
    Train the learner on the bootstrap examples and all examples annotated to this point
```

Figure 6.1: Selective Sampling Algorithm

6.2 Application to Parser Learning

We first investigated the application of active learning to the acquisition of parsers by CHILL. While CHILL's training examples are sentence's paired with their representation, the examples given to the control-rule induction algorithm (Section 2.1) are much more specific. They are created by

Learn a parser with the examples annotated so far
Attempt to parse each unannotated sentence
Unparsed sentences have the lowest annotation certainty
  To rank within these use:
$$\frac{\text{\# of operators used in attempted parse}}{\text{length of sentence}}$$
Parsed sentences have a higher annotation certainty
  To rank within these use:
$$\frac{\text{Total certainty of operators used in parse}}{\text{\# of operators used}}$$
  The certainty of an operator is the number of annotated examples
    that were used to induce its control information

Figure 6.2: Active Learning for CHILL

the overly-general parser, which analyzes the input pairs to extract contexts in which each parsing operator (e.g., shift, reduce) should and should not be employed. The control-rule induction phase then employs a general ILP algorithm to learn rules that characterize these contexts, and these rules are then incorporated into the final learned parser. The data that our active learning component chooses examples from is in the form of sentences, not these more specific examples of contexts for parsing operators. Further, the information needed for the classification task is not available by directly examining the parse obtained on a new, unannotated, sentence.

To attempt to choose the most informative sentences to annotate, we use certainty-based techniques. Because of the indirectness of the classification task, we first analyze the certainty of the operators used in the attempted parse of a sentence, then use this to actually choose which sentences to request for annotation in the next round of training. The technique used by CHILL's active learner is summarized in Figure 6.2 and discussed in the next few paragraphs.

Given an unannotated example, that is, a natural-language sentence, the parser learned by CHILL may or may not be able to derive a parse for the sentence. Sentences may be unparsable for a variety of reasons. For example, the context learned to allow the application of an operator may be overly specific, disallowing the operator's (correct) application in a novel sentence. Or, it could be that an operator instance has not been seen in any of the training examples annotated to date, so that the learned parser contains no information at all about it.

If a sentence cannot be parsed, obviously it would be a good candidate for annotation and inclusion in the next round of training. However, there are often more unparsable sentences than the desired batch size (k) for a round of active learning, so we need a method for distinguishing between them, and choose for annotation those that would be most useful to CHILL. To do this, we simply measure the maximum number of operator clauses that were successfully applied while attempting to parse the sentence. We then divide this by the number of words in the sentence, to give us an estimate of how close the parser got to obtaining a complete parse. The sentences obtaining the lowest value are chosen first for annotation.

If the number of unparsable examples is less than the total number of examples to choose,

k, then we choose for annotation the most informative sentences from the parsable ones. To do this, we assign a measure of confidence to each parse obtained, by looking at the operator clauses used during the parse. Recall that the control rules to select each operator clause are induced from positive and negative examples of the context in which the operator can be applied. Note that each operator clause is a specific instantiation of one of the more general operators (e.g., shift, reduce) used by the parser. The number of examples used to induce the control rules for an operator is our measure of confidence in the clause: operators that are induced from few training examples are more likely to be in error. We use the number of examples as a measure of confidence because, as pointed out by Holte, Acker, and Porter (1989), small disjuncts (those that cover few examples) are more error prone than large ones. We then average this confidence over all operators used in the parse of the sentence to obtain the measure used to rank the entire sentence.

To further decrease annotation effort in the database query domain, we do not request annotations for sentences that vary only in database constant names from others that have been chosen for annotation in the current round, nor for sentences that contain a subset of the words present in a sentence already chosen for annotation in the current round. This screening process will increase the diversity of the examples chosen in each round, and thereby increase the amount of information to be gained in each round of choosing examples for annotation. In the future, we could avoid this by choosing one example at a time. Making this computationally efficient would necessitate making CHILL an incremental learner, to avoid redundant work during each round of learning.

6.3 Experimental Results for Parser Acquisition

We tested the effectiveness of active learning on the U.S. geography query domain described in Chapter 4. We choose a random set of test examples and train on actively or randomly chosen subsets of the remaining examples, for each of 10 trials. We compare the accuracy obtained using active learning to that obtained without active learning. We (randomly) choose $n = 25$ bootstrap examples from the training examples, then either actively or randomly choose $k = 25$ examples at a time to add to the training examples for the next round of learning. The result of learning is evaluated after each round, using the same accuracy measure as used in the previous experiments with this corpus.

The results, in Figure 6.3, show an advantage in intelligently choosing examples, with the advantage starting out small, increasing as more evidence is gathered, then decreasing again as the available training examples are exhausted. The advantage starts out small because at first the active learner does not have very much information on which to base its decision for choosing examples. We see the decrease in advantage at maximum number of examples because with our limited amount of data, the amount of information that can be learned from it eventually reaches its limit, around 70% accuracy; active learning and random learning are eventually training on the same set of examples. However, the number of examples required to reach the maximum level of accuracy is significantly reduced when using active learning. To get within 5% of this accuracy requires around 125 examples when using active learning, but 175 examples when they are randomly selected, a savings of 22%. Also, to surpass the performance of *Geobase* requires 100 examples when using active learning, versus 125 when examples are chosen randomly, a savings of

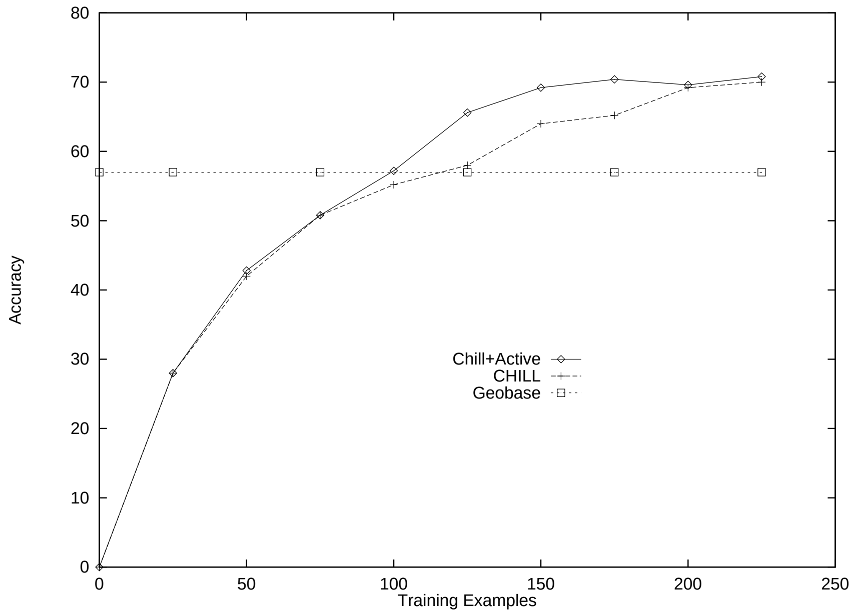


Figure 6.3: Active Learning on the Geography Corpus

11%. The differences between active and random choice are statistically significant at 125 and 175 training examples.

We also performed some experiments with the larger geography corpus discussed in Section 4.6. We trained CHILL with actively chosen versus random examples, as for the smaller corpus. For this corpus we used $n = 50$ bootstrap examples and a batch size of 25. The results are shown in Figure 6.4. Here the savings with active learning is about 150 examples to reach an accuracy close to the maximum accuracy obtained by randomly sampling, or about a 35% annotation savings. Obviously this is a more difficult corpus, but still we are able to choose examples that allow some savings in annotation cost. We also noticed that the proportion of errors made by the system when it was trained with the examples chosen by active learning was lower at the top of the learning curve than for randomly chosen examples, as shown in Figure 6.5. The error is calculated by measuring the number of sentences in the test set that were parsed, but whose parse retrieved incorrect answers from the database, and dividing this by the total number of sentences in the test set.

6.4 Application to Lexicon Acquisition

The above experiments examined parser acquisition in isolation, assuming that a correct lexicon was available. One would like to choose examples that allow good generalization when learning both semantic lexicons and parsers. We first examined certainty-based active learning techniques to choose examples for annotation when learning lexicons in isolation. There are again two levels of certainty, at the level of the individual lexicon entries, and at the sentence level. The value of the heuristic measure for a given lexicon entry is a useful metric by which to estimate how sure we are of the correctness of the entry.

To choose the sentences to be annotated in each round, we first bootstrapped an initial

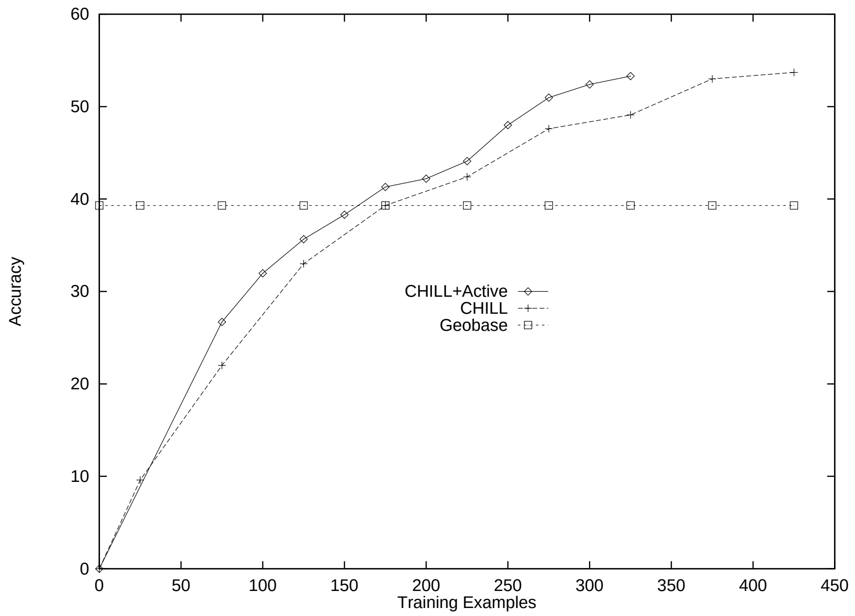


Figure 6.4: Active Learning on Larger Geography Corpus

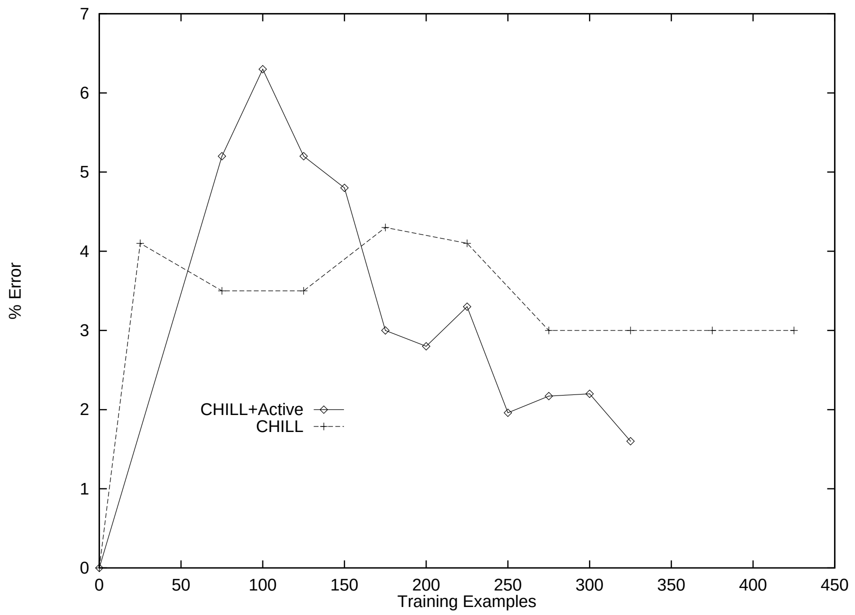


Figure 6.5: Error Rate on Larger Geography Corpus

Learn a lexicon with the examples annotated so far
1) For each phrase in an unannotated sentence:
   If it has entries in the learned lexicon,
      then its certainty is the average of the heuristic values of those entries
   Else, if it is a one-word phrase
      then its certainty is zero
   Else, do not count other multi-word phrases
2) To rank sentences use:

$$\frac{\text{Total certainty of phrases from step 1}}{\text{\# of phrases counted in step 1}}$$

Figure 6.6: Active Learning for WOLFIE

lexicon from a small corpus, keeping track of the heuristic values of the learned items. For each unannotated sentence, we took an average of the heuristic values of the lexicon entries learned for phrases in that sentence, giving a value of zero to unknown words, and eliminating from consideration any words that we assume are known in advance, such as database constants. The sentences with the lowest values were chosen for annotation, added to the bootstrap corpus, and a new lexicon learned. This scheme is summarized in Figure 6.6.

6.5 Experimental Results for Lexicon Acquisition

We applied active learning for lexicon acquisition to the smaller geography query corpus. An initial lexicon was learned using 25 bootstrap examples. The method described in the previous section was used to choose 10 additional examples per round, and lexicons and parsers were learned from these examples using WOLFIE then CHILL, the latter using the learned lexicons as background knowledge. To simplify the trials, we used WOLFIE without overlap, two-word phrases, or the parsability heuristic. The accuracy of the resulting learned parsers was compared to the accuracy of those learned using randomly chosen examples to learn lexicons and parsers, as in Chapter 4. Figure 6.7 shows the accuracy on unseen data of parsers learned using the lexicons learned by WOLFIE when examples are chosen randomly and actively.

There is an annotation savings of around 110 examples by using active learning: the maximum accuracy is reached after 115 examples, versus 225 with random examples. While the advantage of using active learning is clear once enough examples have been seen, the learning curve for active learning does show a dip below the random curve at 75, 85, and 95 examples. Since we are learning both lexicons and parsers, but only choosing examples based on WOLFIE's certainty measures, the chosen sentences may not necessarily also help CHILL in achieving better accuracy. We investigate the integration of the two active learners in the next section.

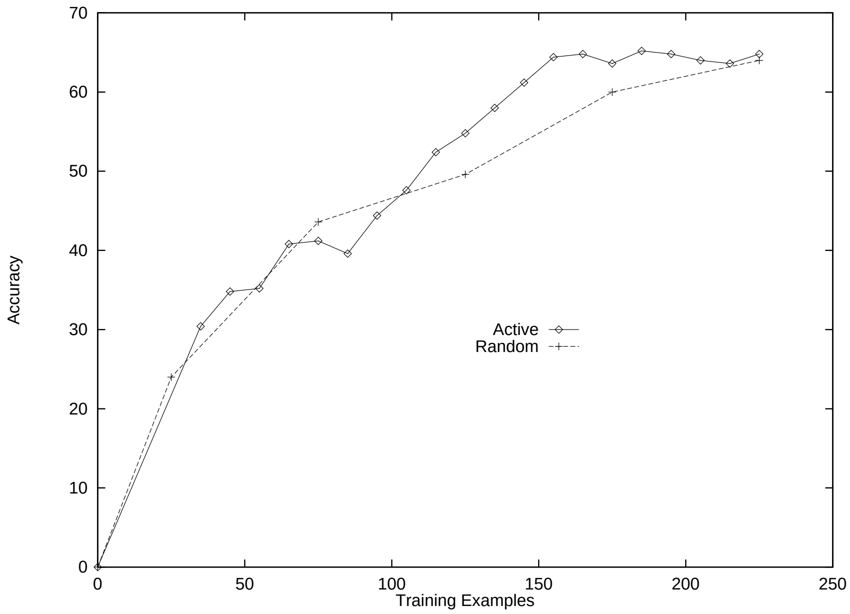


Figure 6.7: Using Lexicon Certainty for Active Learning

6.6 Active Learning for the Integrated System

When learning both lexicons and parsers, we would like our active learning component to choose examples that are useful for both. We are using the same training data for two different tasks, each of which may find a different set of sentences most useful towards improving its own learning. The results in the previous section show that concentrating on active learning for WOLFIE alone may not ensure that the examples chosen are useful from the point of view of parser acquisition.

The simplest method to combine the two systems for choosing useful examples is to let the two systems alternate in choosing the uncertain examples from the point of view of their own uncertainty. We did this in each iteration of choosing examples: each system ranked the sentences according to its certainty in their annotation, as described above. To choose a batch size of 10, for example, the most uncertain example according to CHILL was first chosen for annotation; then, the most uncertain example according to WOLFIE was compared to the previously chosen example, and chosen for annotation if it was not a variant (with respect to database constants) of that example; if the example was a variant, the next most uncertain example that was not a variant was chosen. This alternate choosing of examples continued until 10 examples were chosen, 5 from each system.

6.7 Experimental Results: Full System Active Learning

We tested this method on the smaller U.S. geography query corpus, using 25 bootstrap examples and a batch size of 10. Figure 6.8 shows the results, including a curve, labeled `ActiveChill`, in which the examples were chosen using only CHILL's certainty metric. The results here are also rather mixed. Using both systems for actively choosing examples shows a clear example early in the learning curve, but later dips below the curve for using CHILL alone to actively choose examples. However, using both systems does outperform the random choice of examples. Using both systems

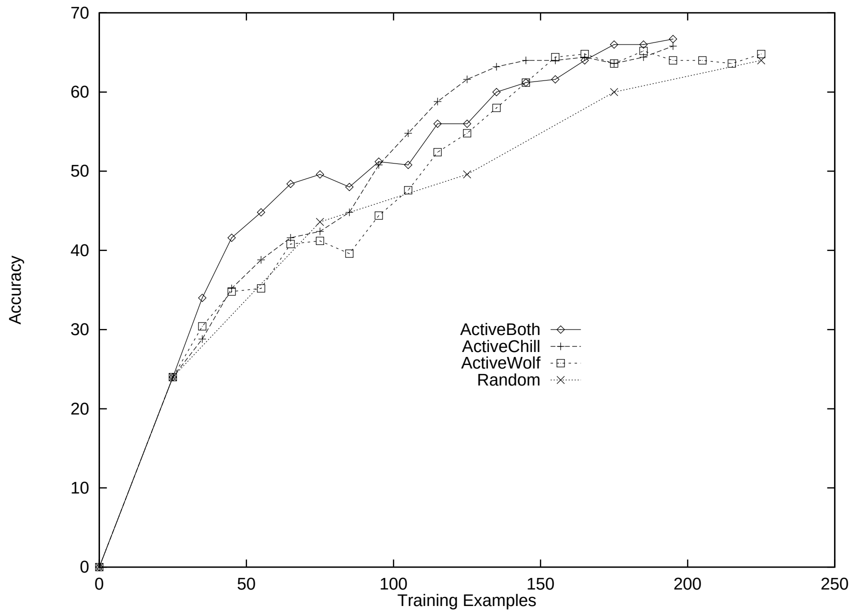


Figure 6.8: Active Learning for Lexicon and Parser Learning

is statistically significantly better than using just CHILL at 45, 55, and 75 training examples, and statistically significantly better than using just WOLFIE at 45,55, 75, and 85 examples. On the other hand, using just CHILL is statistically significantly better than using just WOLFIE at 105,115,125, and 135 examples. The experiments on random data were only run at 25,75,125,175, and 225 examples, so we can only compare the difference at these points. Given that, using both systems to actively choose examples significantly outperformed the random choice of examples at 75 examples, and using CHILL significantly outperformed random choice at 125 examples. Further, using both systems to actively choose examples requires annotating about 60 fewer examples than using random examples, for a savings of 26%.

6.8 Discussion

Our hypothesis was that combining CHILL's active learner with WOLFIE's active learner would improve on the results of either system choosing examples on their own. Our results only partially confirmed this hypothesis: for smaller numbers of training examples, the hypothesis holds, but when there are few examples left to choose from, it does not. In hindsight there are some reasons why it might have failed. First, when both lexicons and parsers are being learned, the lexicons learned affect the success of parser acquisition. When CHILL's learned parser evaluates new sentences to choose for active learning, it will have trouble both with sentences that contain unseen structure, and sentences with words that are not in the lexicon (or not learned correctly by WOLFIE). Therefore, it is also choosing sentences based on unknown words. So perhaps learning is boosted early on just by collecting more information about unknown words, which both systems effectively do in the beginning. On the other hand, WOLFIE alone is not as successful, in the middle portions of the learning curve, at choosing examples that are useful from the point of view of parser learning, but CHILL's uncertainty metric is successful, so the combined system loses its advantage, due to

WOLFIE's poor choice of examples. Future work should investigate more effective ways of combining the two learners.

6.9 Committee-based Active Learning

When we began exploring active learning for parser acquisition, we first investigated several techniques to generate a committee of parsers, so as to use committee-based active learning. These techniques included: varying the learner's induction parameters, randomizing the order of presentation of the training examples, stopping induction at different points, and even trying to force CHILL to learn more general and more specific parsers based on a version space-like approach (Cohn et al., 1994). These attempts did not generate very diverse committees, partially perhaps because of the small number of examples available for each specific parsing operator, and also probably because CHILL's learner is robust against many of the above variations.

Another attempt at a committee-based learner that was partially successful was a method inspired by the version space approach, but instead of forcing the *learner* to be biased, we used the learned parser as a starting point and biased *it*, as follows. From the learned parser we generate a more specific parser that parses only the training examples (or sentences varying from the training sentences only in constant names such as "Austin" or "Mississippi"). In other words, each operator is restricted so that it is only applicable in contexts in which it was applied during example analysis. The second, more general, committee member was also formed by modifying the original learned parser, but this time to allow the application of an operator in *any* contexts *except* those that led to inconsistent states during example analysis. In addition, it is given background knowledge of the names of the predicates of the query language, and which predicates could possibly be embedded in the others as arguments. This information is used to derive additional variants of the basic operators using predicate combinations not seen during training.

Examples were then chosen for annotation by measuring the amount of disagreement between the committee members in the operators used during the parse of the examples. In more detail, for each sentence, we first attempt to parse it with each committee member (the most general parser, the most specific parser, and the original learned parser), and note the operators used during the attempted parse, whether a final interpretation is found or not. For each pair of committee members, we next measure the number of operator applications that the two attempted parses had in common, calling this $|I_{AB}|$, where the subscript indicates which pair of parses are being compared, the general parse (G), the specific parse (S), and the original parse (O). The formula to find the disagreement is then the average of:

1. $(|I_{SG}|/|S| + |I_{SG}|/|G|)/2$
2. $(|I_{SO}|/|S| + |I_{SO}|/|O|)/2$
3. $(|I_{GO}|/|G| + |I_{GO}|/|O|)/2$,

where $|S|$, for example, is the number of operator applications in the most specific parse. The sentences with the smallest resulting value are those that the three parsers disagree on the most in terms of valid operator application, so could potentially contain the most new useful information if annotated, so these were chosen first for annotation in this scheme. Any ties are broken by choosing the examples with the smallest average number of operators used by the committee members during parsing.

This approach did not perform as well as the certainty based method described above. This technique attempts to find training examples that "narrow the distance" between a specific and a general parser. However, the learner also needs to see several examples of the use of each operator. The certainty-based method, at least during the early stages, at least ensures that something new is learned for each newly annotated example, without overwhelming the learner with too much new information at once. Further, the committee-based approach has only been implemented and briefly tested with one query representation, and it is not clear how well or easily it could be applied to a different representation formalism, because of the overhead of generalizing and specializing the learned parser. The certainty-based approach, on the other hand, is easily applicable to new representations.

Another approach could be to combine the certainty-based method in Section 6.2 with the most general parser. For each unannotated example, we could use the most general parser to generate all possible operators that it may provide information about. This could be used as a measure of the amount of new information to be gained by annotating each example. This measure could then be combined with the certainty measure in Section 6.2, to generate an even more useful active learner.

We finally tried to generate a committee by giving different committee members different sets of training examples as suggested in Matan (1995). Instead of using completely disjoint training sets however, we used *bagging* (Breiman, 1996) to divide the annotated examples into groups for each committee member. Then, to choose examples for annotation, each committee member attempts to parse the example, and the amount of disagreement among the members with respect to this attempted parse is then measured, as described earlier in this section. The examples with the most disagreement are chosen for annotation.

However, there are three disadvantages to this approach. First, this forces us to run the learner multiple times, increasing the learning time, which perhaps may overshadow the gains of active learning. The fact that each learner is given a smaller number of examples than it would otherwise, however, could partially overcome this objection. On the other hand, most researchers who have investigated committee-based active learning have done so using a committee of simpler learners than CHILL. A second disadvantage of this method is that the committee members are likely to disagree on examples for which the parsing operators are "uncertain" anyway (because the rare operators will be those appearing only in some examples, so only some committee members, so will lead to disagreement), which is estimated by certainty-based active learning with less training effort.

Finally, there is no straight-forward, principled way to combine the votes of the committee members in the case of learning a parser; even the method described earlier in this section is quite ad hoc. The difficulty comes because each sentence yields multiple training examples (for parsing operator application), and measuring the committee member disagreement over multiple categorizations is not as straightforward as for one. One option is to use vote entropy as in Engelson and Dagan (1996), but it is not clear what to measure. They use

$$-\frac{1}{logk} * \sum_{c} \frac{V(c,e)}{k} log \frac{V(c,e)}{k},$$

where $V(c, e)$ denotes the number of members assigning example e to class c, and k is the number of committee members. However, parses do not contain information about just one class. One could look at the entropy between the operators applied in a specific parse state, treating these as the classes, but different committee members would have different parse states, resulting in little overlap between classes. Alternatively, one could look at the entropy between the operators applied during the parse of each example for each committee member (instead of entropy between classification as in the above equation). But then the "probabilities" would not sum to one, because multiple operators are applied in an example. Therefore, perhaps the best way to combine votes is to use the agreement measure described earlier in this section.

Initial attempts at using this kind of committee-based active learning resulted in improvements very similar to those obtained using the certainty-based active learning discussed above, and so were abandoned for now due to higher run times.

Chapter 7

Related Work

This dissertation research draws from many subdisciplines within artificial intelligence. These include machine learning, computational linguistics, and inductive logic programming. Some of these connections have been mentioned in previous chapters, while here we more extensively review the literature in lexicon learning, active learning, and related areas. While we may have omitted some tangentially related research, we have attempted a comprehensive summary of the most closely related work.

7.1 Lexicon Learning

There are two predominant views of the semantic lexicon acquisition task to be found in the related literature. First, techniques for finding the meaning of a word can be based on the company it keeps. This has also been called *collocative meaning*, first by Leech (1974). This view is the basis of the statistical paradigm, which is exemplified by Church and Hanks (1990), Zernick (1991), Dyer (1991), and others. Second, techniques for finding the meaning of a word can be based on the semantic properties associated with its use, also called *conceptual meaning* by Leech (1974). The latter is the approach taken by this dissertation.

Focusing on systems that base word meanings on the semantics of their use, the work can be further subdivided. First are systems that learn semantic categories for words. These include distributional clustering systems such as those of Pereira, Tishby, and Lee (1993) and Lund, Burgess, and Atchley (1995). Systems such as that of Riloff and Sheperd (1997), that group semantically similar words together, also fall into this category. While these methods are perhaps useful for finding synonym sets and other word groupings, the learned knowledge is not directly applicable for aiding the transformation of sentences into their representations.

A second focus of semantic lexicon learning systems is the acquisition of relations between words or categories of words. For example, several systems (Grimshaw, 1981; Brent, 1991; Manning, 1993; Basili, Pazienza, & Velardi, 1996) acquire information about words' subcategorization or selectional restrictions. Another example of this type of system is that of Brent (1990). His system is restricted to learning only verb meanings, and assumes a simple parser is available to the learner. He demonstrates the ability to learn to distinguish between stative and non-stative verbs; while this is a useful kind of semantic information to learn, he demonstrates no ability to learn other types of semantic information. Although this information may aid a semantic parser, it does not directly contribute to transforming words into a representation of their meaning.

Finally, the lexicon learning task can be grounded in the representations presented to the learner, where understanding is demonstrated by the ability to parse, answer questions, or perform a similar linguistic analysis. This latter is the approach that this research, and that of several others (discussed below), have taken. Most of the early systems taking this approach were given access to a large amount of background knowledge and pre-existing parsing capabilities, while more recent work has moved more in the other direction, for example with the use of large corpora and statistical methods. WOLFIE's approach is between these two extremes, assuming access to parses but not parsers, and using statistics in its heuristic measure. Exploring this category of systems further, the type of knowledge and representations used can be further subdivided into learning to execute actions, learning mappings to visual input, and learning a representation to enable semantic parses of sentences.

7.1.1 Learning to Execute Actions

First, the task of lexicon learning can be grounded in learning to take actions in the world. In other words, to demonstrate understanding, an agent's performance is analyzed. Suppes, Liang, and Böttner (1991) use this approach. A robot is trained on cognitive and perceptual concepts and their associated actions, and learns to execute simple commands. Along similar lines, Tishby and Gorin (1994) have a system that learns associations between words and actions, but they use a statistical framework to learn these associations, and do not handle structured representations.

7.1.2 Learning Mappings to Visual Input

One of the earliest to explore a kind of language and lexicon acquisition was Siklossy (1972). The input to the system is sentences paired with "pictures" in a functional language (FL). The system is also given a procedure for representing relations among the objects in the input; this structure helps program discover structure in the sentence. The system learns *patterns* for transforming FL back to natural language. The algorithm is incremental, with some dependence on the order in which sentences are presented; it also seems to depend on the ordering of words to be somewhat synchronized with the ordering of their meanings in the FL.

Moran (Salveter, 1979, 1982) incrementally learns frame-like conceptual structures for verb meanings from simulated perceptual input paired with linguistic input. However, unlike ours, the system is given world knowledge in the form of *isa* and *superset/subset* links, which provides much information. Also, the linguistic input is in the form of a set of pairs in the form `(casename, value)`, instead of natural language sentences. However, the system learns somewhat complex representations in the form of case-frame-like representations describing change of state information, as opposed to the one-to-one representations learned by WOLFIE.

Feldman and his colleagues at Berkeley (Feldman, Lakoff, & Shastri, 1995) are actively pursuing neural network models to acquire semantic concepts. The latest system by Bailey (1997) is given examples of pictures paired with natural language descriptions that apply to the picture, and learns to judge whether a new sentence is true of a given picture. The focus in their work is on the acquisition of spatial terms and verbs. In a similar vein, Nenov and Dyer (1994) describe a neural network model to map between visual and verbal/motor commands.

While our research currently restricts itself to text input, this simplifies the learning task so that vision or speech recognition does not have to be learned in addition to language. In principle, a system converting paired sentences and line drawings into a semantic representation of that sentence could be used as a front end to WOLFIE.

7.1.3 Learning for Semantic Parsing

The systems most closely related to WOLFIE learn mappings from text input to semantic representations, usually in support of further processing such as parsing. In other words, there is an implicit procedural semantics associated with the acquired knowledge. For example, an early system, RINA (Zernik, 1987), acquires verb-particle combinations that can later be used by a parser. Unlike WOLFIE, the system is an attempt to model the task of a second language learner, and requires a large amount of user interaction. Because of this focus, extensive background knowledge is available to the system, including an initial hierarchical lexicon. No real world tests have been performed on the system. Fukumoto and Tsujii (1995) put forth interesting methods for learning semantic lexicon information, but demonstrate only verb learning.

We have already discussed the research by Siskind (1996) in detail. While his system was developed from the point of view of cognitive modeling, the representations learned can be used to support further processing. He has demonstrated the scalability of his system to input containing noise, ambiguity, synonymy, and referential uncertainty. As our experiments demonstrated, WOLFIE is more successful at lexicon learning in a domain directly supporting semantic parsing.

Several systems (Russell, 1993; Hastings & Lytinen, 1994; Knight, 1996) learn new words from context, assuming that a large initial lexicon and parsing system are available. For example, Hastings and Lytinen (1994) present an incremental learner that searches for word meanings in a predetermined semantic hierarchy. Haruno (1995) describes a system for learning the semantics of verbs, but requires background knowledge in the form of a thesaurus. WOLFIE focuses on learning from minimal background knowledge, though it can take advantage of an initial lexicon if one is available.

7.1.4 Learning Syntax and Semantics

Some lexicon acquisition systems learn information that crosses the boundaries between syntax and semantics. For example Foul-Up, developed by Granger (1977), starts with an initial lexicon and parser and hypothesizes the meanings of unknown words in novel sentences based on this knowledge. Another example is CHILD (Selfridge, 1986), which tries to model children's acquisition of language, integrating the acquisition of word meanings and syntax. The syntax learned is in the form of positional rules, and the semantic representations are derived from Conceptual Dependency. Limitations of CHILD include heavy reliance on careful crafting of the training input and no handling of ambiguity.

Webster and Marcus's (1995) system learns information only about verbs from sentence/representation pairs, where the representation includes both syntax (e.g., phrasal information) and semantics (e.g., agent and patient). The system learns syntactic properties, predicate/argument structure, and a mapping between the two. Pedersen and Chen (1995) describe a method for acquiring syntactic and semantic features of an unknown word. They assume access to an initial concept hierarchy, and do not present any experimental results.

In several systems, learning to parse and learning word meanings is more closely coupled than the WOLFIE and CHILL combination. For some systems, acquisition of meaning is based on extensive parsing capabilities (Zernick & Dyer, 1987; Lytinen & Roberts, 1989). Berwick (1983), Selfridge (1986), Miikkulainen (1993), and Cardie (1993) also integrate the learning of syntax, semantics, and the lexicon. For example, Berwick's (1983) system is given syntactic constraints from a parser and uses analogical matching to compare known to unknown verbs. GENESIS (Mooney, 1990) integrates some word learning with a larger system for schema acquisition. Acquisition here is not limited to a single sentence of context, but an entire story.

7.2 Other Related Work

7.2.1 Translation Lexicons

This work also has ties to the work on automatic construction of translation lexicons (Gale & Church, 1991; Catizone, Russell, & Warwick, 1993; Kumano & Hirakawa, 1994; Wu & Xia, 1995; Melamed, 1995). These systems use input in the form of aligned pairs of sentences in two different natural languages. While most of these methods also compute association scores between pairs (in their case, word/word pairs) and use a greedy algorithm to choose the best translation(s) for each word, they do not take advantage of the constraints between pairs. One exception is Melamed (1996); however, his approach does not allow for phrases in the lexicon or for synonymy within one text segment, while ours does. Finally, the mappings learned for translation lexicons are string to string mappings, instead of the string to structure mappings learned by WOLFIE.

7.2.2 Acquisition from MRDs

Many researchers (Amsler, 1981; Walker & Amsler, 1986; Byrd, Calzolari, Chodorow, Klavans, Neff, & Risk, 1987; Boguraev & Briscoe, 1989; Karov & Edelman, 1996; Rigau, Rodríguez, & Agirre, 1998) have investigated the extraction of lexical information from Machine Readable Dictionaries (MRDs). While these methods may be capable of generating general information, additional information is needed from some source to tailor the information to each domain. There is potential, therefore, for integrating the MRD extraction results with the lexicons learned by WOLFIE.

7.3 Active Learning

Cohn et al. (1994) were among the first to discuss certainty-based active learning methods in detail. They focus on a neural network approach to actively searching a version-space of concepts. Liere and Tadepalli (1997) apply active learning with committees to the problem of text categorization. They show improvements with active learning similar to those that we obtain, but use a committee of Winnow-based learners on a traditional classification task. Dagan and Engelson (1995) also apply committee-based learning to a classification task, that of part-of-speech tagging. In their work, a committee of Hidden-Markov Models is used to select examples for annotation. Lewis and Catlett (1994) use *heterogeneous* certainty-based methods, in which a simple classifier is used to select examples that are then annotated and presented to a more powerful classifier.

All of these methods are applied only to a simple classification task, while our work applies active learning to the more difficult tasks of parser and lexicon acquisition. As discussed in Chapter 6, these tasks are more difficult because of the indirect nature and complexity of classification.

Chapter 8

Future Directions

8.1 Lexicon Learning

8.1.1 Improving the Algorithm

With sufficient training data, the current version of the algorithm does a good job of learning a lexicon that covers the data and is useful to CHILL. There are several improvements that could be made to improve the results on corpora similar to the ambiguous artificial one and on other noisy corpora. First, we could move to a best-first search instead of a greedy one. With dependency-directed backtracking, this could allow the algorithm to correct itself when it finds that it is unable to cover the data with the learned lexicon. Improving the search may also allow the algorithm to better handle noisy inputs, by allowing the use of more intelligent methods than simple thresholding, which is the best we could do within the framework of a greedy search.

There are additional components that could be added to the lexicon candidate pair evaluation heuristic. One example is a component that would bias the lexicon towards enforcing *locality* constraints, when doing so would aid parsing. Locality enables the learner to prefer a situation in which if two words are "close to" each other in the sentence, then their meanings are also "close" together in the sentence's representation. For example, in the pair

What is the highest point in the state with the capital Des Moines?

```
answer(C, (high_point(B,C), loc(C,B), state(B),
       capital(B,A), eq(A,cityid('des moines',_)))).
```

if (**capital**, `capital(_,_)`) is a lexicon entry, then a locality bias would prefer pairing **state** with `state(_,_)` over pairing it with `high_point(_,_)`.

Another component that could be added to the heuristic may also improve the lexicons learned for some corpora. In the artificial corpora explored in Chapter 5 and also in some real world domains, the "correct" meaning for each word is not necessarily the most general one in terms of specifying less information (fewer nodes in trees as in our generality portion of the heuristic measure). This sense of generality does not necessarily imply generality in the sense of covering more examples. In this case, a correct meaning for a phrase, p, is more likely to be obtained by prefering LICS from representations whose sentences have few or no phrases in common other than p, than by prefering LICS that have fewer nodes. Therefore, adding a component to the heuristic that counts the number of other phrases that source sentences for LICS had in common, and that prefers meanings generated from sentences containing few phrases in common, would potentially boost the performance of WOLFIE in domains for which correct meanings do not necessarily specify less information than incorrect ones.

Another possibility is to use knowledge about syntax or morphology to bias the search. For example, if we know a word is a noun, and have background knowledge about which portions of the sentence representation correspond to objects, we could prefer mapping nouns to objects. Another form of background knowledge that could be useful is WordNet (Beckwith et al., 1991), a hierarchically organized computational lexicon. The distance in the WordNet hierarchy between words and the names used in a sentence's semantic representation could be measured, and used as an additional component of the heuristic. This might be similar to the way that Resnik (1995) uses semantic similarity in WordNet to disambiguate nouns. The method in Brunk and Pazzani (1995) to compute lexical distance in WordNet might also be adapted to this task; they use it in the context of theory revision.

As discussed in Chapter 4, fracturing performs slightly better than LICS when the training set is small; to overcome this, we could fracture in cases where we think it might improve the results, and use LICS otherwise. Two cases where this would be useful are immediately obvious. First, if a word appears in only one sentence, there are no pairs of representations to use to generate a LICS. Our current heuristic of using the entire representation as an initial candidate meaning for such words, and then using constraints between word meanings to generalize the representation does not work well in all cases. Fracturing should ensure a greater probability of successfully learning meanings for such words. Second, a rare word may have only empty trees as the result of finding LICS, indicating that the word is ambiguous. It may help to fracture the representations of sentences in which such a word appears to generate candidate meanings for it. Common words that have only an empty LICS are more likely to be function words.

The current algorithm is not guaranteed to cover the training set. One difficult case is situations in which the node labels of meaning representations for multiple words intersect (usually, those with similar meanings), and these words appear together in some of the same sentences. For example, "The boy hit the person." is represented as

```
[ptrans, agt:[person,sex:male,age:child], pat:[person]]
```

in the CD representation formalism; if the algorithm first determines that **person** maps to `person`, it would not be able to determine which occurrence of `person` in the representation is accounted for by this meaning.

A second difficult case for covering arises when a word is very common but ambiguous, and one of its uses is as a function word, depending on context. An example in the geography corpus is **in**, which can mean either `loc(_,_)` or `[]` in different situations. Finally, forcing the algorithm to consider closed class words as candidate lexicon entries also sometimes causes difficulties, or at least errors in the lexicon. A way to handle these covering difficulties might be to eliminate consideration or generation of a candidate lexicon entry unless it covers a sentence/representation pair, when combined with the candidate meanings of the other words in the sentence. Another possibility is to explicitly include the empty tree in candidate lexicon entries when a LICS is empty.

Handling more flexible and longer phrases is also desirable. While it would not add much complexity to consider longer phrases, it is often the case that the words in a meaningful "phrase" do not follow one another consecutively in a sentence. These are more like the "patterns" used in information extraction tasks. If we wanted to consider such patterns, we would have to establish some arbitrary threshold for the portion of the input in which a pattern should occur to justify its inclusion in our candidate set, to avoid an exponential blowup in the number of patterns considered. Even then, our current heuristic may ignore such patterns even when they are meaningful, due to their sparseness in the input, compared with the words in the pattern. Phrases to add to the set of candidate pairs to be considered could be generated by statistical evaluation of corpus co-occurrences. Progress along this front is discussed in Smadja (1991).

8.1.2 Generalizing the Problem Definition

While the current problem definition handles most of the cases that we considered in our experiments, some extensions to the definition may be needed in other domains, tasks, or for other representation formalisms.

First, some representations may require making f, the interpretation function, a relation instead of a function, as is allowed in Siskind's system. This would eliminate our single-use assumption, allowing a phrase's meaning to appear multiple times in the representation of that sentence. For example "The man moved." could be represented as

```
[ptrans, agt:[person,sex:male,age:adult],
        pat:[person,sex:male,age:adult]].
```

Also, the problem definition would need more drastic modification if not all components of the meaning representation were due to the words in the sentence. This may be the case in translation or dialogue systems; in the latter case, multiple sentence context may be included in a sentence's meaning. For example, ellipses often occur in dialogue, such as someone asking "What is the largest city in New York?", and next asking "What about Texas?". Another possibility is to allow a phrase and the words in it to each map to a different meaning in the sentence representation. For example, in "What state has the highest elevation?", represented as

```
answer(B, highest(A, (state(B), high_point(B,A)))),
```

allowing **highest** to map to `highest(_,_)` and **highest elevation** to map to `high_point(_,_)` may be more useful than mapping **elevation** to `high_point(_,_)`, since questions about elevations can be either about highest or lowest elevations.

Introducing more integration between parser and lexicon acquisition could improve the results for both. For example, there might be a way to learn to order the lexicon in a way that increases parser efficiency. As another possibility, we could give a partial parse to WOLFIE when CHILL has problems. This could be used to try to distinguish between errors in learning and ambiguous words. Collocations could be kept to help CHILL with disambiguation. The parsability heuristic could be improved to determine *which* word in an unparsable sentence is causing an incorrect parse, rather than assuming that the most recently learned word is causing the problem. Another use of syntactic knowledge could be to use the results of a part-of-speech tagger to detect potentially ambiguous words.

Another desirable extension is to learn more flexible and structured lexicons, so that the lexicons can be used by more complex parsing systems. One possibility here is to combine our method with previous research on clustering, thesaurus construction, or learning selectional restrictions. Perhaps WordNet could be used to help evaluate potential hierarchical organization options for a learned lexicon. It could also be used to bias the current greedy search, by prefering to associate the lowest words in the hierarchy with the most specific meanings. Another possibility is to use categories, from WordNet or automatic clustering methods, to learn a general representation beyond the information explicitly present in the input. A simple technique would be to generate lexicon items by retrieving synonyms of learned words. For example, in the geography query domain, this would result in learning that all state names should be mapped to `eq(_,stateid(<statename>))`, or in the CD domain, learning that all people are mapped to `[person, sex:<?>, age:<?>]`.

As mentioned in Chapter 7, we could combine this research with the work on extracting lexical information from Machine-Readable Dictionaries (MRDs). The learning from MRD methods could be used to build the initial (domain-independent) lexicon, then an extension of Wolfie could be used to build on that lexicon, and specialize it for use in a specific domain.

Another avenue to pursue could be to generalize the input beyond a text-based format. For example, the future may see vision systems capable of mapping from video to some symbolic representation of what is going on in the scene. Rajagopalan and Kuipers (1994) discusses recent progress: a system for mapping from diagrams and text to database relations. If this input is annotated with the audio that accompanied the scene, such inputs could be used in an extension of Wolfie and Chill. Some of the work by Regier (1991) and others at Berkeley has made related strides in this direction.

This type of input would require enabling the system to handle a type of referential uncertainty, since there would be many objects in a scene that are not referred to in a sentence. An alternative to the disjunction of possible meanings used in Siskind's work to represent referential uncertainty is to represent all of the objects in the scene and their relationships in a large graph-structured representation. This would be more easily extracted from video input than Siskind's version of referential uncertainty, and would mesh well with our methods for finding relevant meanings. The task of the learner would be to determine which subgraph(s) of the representation map to the meaning of the sentence, then proceed from there to learn word meanings.

Finally, the basic algorithm is potentially applicable to other learning problems involving multiple, ambiguous concepts. These include learning translation lexicons, learning for diagnosis, and learning rules.

8.1.3 Handling Unknown Words

There may be situations in which the system has to process a sentence that contains words that it has not previously seen. It would be advantageous if the system could hypothesize a meaning for these words, without having access to the meaning of the sentence, but based purely on context. It should also recognize new meanings for previously seen words when only given a sentence as input.

There are several ways to attack the problem of unknown words, some of which have already been explored in previous research (Granger, 1977; Russell, 1993; Hastings & Lytinen, 1994; Knight, 1996), though none have solved the problem completely. One possibility is to integrate more statistical and co-occurrence knowledge into the lexicons learned by Wolfie, and use this knowledge to infer the meanings of new words. For example, the context in which the unknown word occurs could be compared to the contexts in which known words have been used; the system could then learn similar meanings for words appearing in similar contexts. New meanings of previously seen (but ambiguous) words could be detected by noting their use in an unusual context, and again comparing that context to contexts in which known words have occurred. Adding part-of-speech tags to the training input could also help constrain the search.

We could also increase the interaction between WOLFIE and CHILL to help hypothesize meanings for unknown words. Perhaps we could use the type of parsing error encountered by CHILL to help WOLFIE deduce what kind of changes might be needed in the lexicon. CHILL could just shift when it sees an unknown word, and pass a partial parse back to WOLFIE. Also, if a sentence takes a relatively long time for CHILL to parse during its example analysis phase, this may indicate that some word in that sentence has a wrong meaning in the lexicon.

WordNet could help here as well. If we see "The fragle ate", we would tend to associate the meaning of **fragle** with concepts that are close, in the WordNet hierarchy, to other words that acted as the agent of an eating verb.

Finally, we note that the problem of unknown words is also a difficult one for people. We do not often correctly deduce the meaning of a novel word from context alone, but instead ask for a meaning or look it up in a dictionary. Another option is to ignore it and treat it as noise. Even in the absence of knowledge of the word, we can still usually deduce a partial meaning of the sentence, and we would like our system to do the same.

8.1.4 Additional Evaluation

After making some of the above improvements, more evaluation than that presented here would be possible. For example, we could compare our methods to purely statistical approaches such as Expectation Maximization (Dempster, Laird, & Rubin, 1977). Such methods do not take exclusivity into account; using a statistical method to help rank correlations and then our method to generalize and constrain meanings may improve our results. Another possible evaluation technique is to obtain an unannotated corpus and an existing semantic parser. We could then parse the corpus with the existing parser and use the results as training input to "reverse engineer" the original lexicon and parser. The performance of the learned and hand-built lexicon/parser systems could be compared on novel inputs.

While we show initial success here with a CD-like representation formalism, we would like to demonstrate the feasibility of our approach more broadly. For example, the method should be evaluated on corpora of paired sentences and SQL queries. In general, more annotated corpora are needed as training input for semantic acquisition systems such as ours. Using active learning should help minimize the annotation burden required to build such corpora.

8.2 Active Learning

We now turn to possible directions in the area of active learning for parser and lexicon acquisition. While the results obtained to date are promising, there are many areas for potential improvement. First, the example selection heuristics could be further refined. We could incorporate operator certainty metrics for unparsable sentences, instead of only using them for parsable sentences, as we do now. In the longer term, we could utilize a probabilistic parser (one that assigns probabilities to each analysis of a sentence) and use the probabilities to estimate the confidence in a parse.

Alternatively, the contexts in which a parsing operator is applicable could be loosened to allow a sentence to be parsed, and the more loosening needed, the more likely we are to choose that sentence for annotation.

Second, we would like to further investigate committee-based learners for parser acquisition. As discussed in our initial studies in Section 6.9, building a good, diverse committee for CHILL is not straightforward. Perhaps a committee of simpler learners could be used to select examples for annotation and training by CHILL; a similar method is presented by Lewis and Catlett (1994).

Using incremental versions of both WOLFIE and CHILL could allow the use of a smaller batch size during active learning. One technique to make WOLFIE incremental would involve including the previously learned lexicon items as candidate entries from which the next lexicon is chosen. They could also be used as a source for LICS from which new candidates are generated, in addition to the current candidate generation method. By using a full memory incremental system that remembers training examples from each round of learning, the heuristic value calculation could remain unchanged from the current version.

Making CHILL an incremental learner would likely prove to be more straightforward. CHILL uses a compression approach in its search for the best parser to cover the training input. The contexts for operators used during the parsing of new input could be added to the the parser learned from previous input, and this combined parser used to initialize CHILL's search. The compression algorithm could then continue its search in the standard fashion. Califf (1998) uses a similar method in an incremental version of a system to learn information extraction rules.

Finally, we would like to test these methods on other data sets and in other domains, such as the acquisition of control rules for planning (Estlin & Mooney, 1997). Active learning techniques like those used here may help reduce the annotation burden in this and other complex domains.

Chapter 9

Conclusion

The ability to understand sentences is crucial to support many higher-level tasks in language understanding systems. To understand sentences, the system must first understand the words in those sentences. Constructing these resources by hand is laborious and error-prone. The research in this dissertation makes progress in easing this burden. The main focus of this dissertation has been the acquisition of semantic lexicons to be used with a system that learns semantic parsers. A secondary focus has been the application of active learning techniques to minimize the annotation burden required to learn lexicons and parsers. These achievements are both useful steps towards our long term goal of automatically building natural language understanding systems.

We have described and evaluated a system, WOLFIE, that takes as input a corpus of sentence/representation pairs, and acquires a semantic lexicon that covers, or accounts for, that corpus. A greedy approach and a compositional assumption are used for efficiency. The system differs from previous work in its use of bottom-up techniques for generating candidate lexicon pairs, and in its exclusivity assumption to constrain these candidate pairs. A heuristic measure, based on corpus statistics and the lexicon learned so far, is also used to evaluate candidates. The learned lexicons can be used as background knowledge as input to the parser acquisition system, CHILL.

We demonstrated the usefulness of the learned lexicons in a real-world domain. Experimental results involving the U.S. geography query corpus show improvement over a previously developed lexicon acquisition system. We also demonstrated WOLFIE and CHILL's applicability to languages other than English. Experiments on two artificial corpora show that the algorithm scales up reasonably well as more ambiguity and synonymy are added to the language, and that is flexible enough to handle multiple representation formalisms. However, further experimentation on larger real-world corpora is needed.

While this research eases the construction of NLP systems, it does necessitate the construction of annotated corpora. While this is arguably an easier task than building an entire NLP system, it is still somewhat difficult. Therefore, we also investigated certainty-based active learning techniques to reduce the sentence annotation effort. Experimental results demonstrated that the number of examples required to achieve close to the maximum possible level of performance was reduced when using actively chosen versus randomly chosen examples. When using active learning just to learn parsers, a savings of 22% annotation cost was realized over using randomly chosen examples. When learning both lexicons and parsers, active learning demonstrated a savings of 26% annotation cost.

In conclusion, this research has made strides towards the automatic construction of natural language systems, with demonstrated application to a natural language database interface, and potential application to translation and dialogue systems. This research holds promise for easing the construction of natural language interfaces, making the large amount of information stored on computers world-wide more easily accessible for use by machines and people.

Appendix A

Sample Training Input

This appendix contains a sampling of the sentence/representation pairs from the U.S. Geography query domain used in the experiments in this research.

A.1 Original Geography Corpus

The full corpus for the orginal geography corpus in English is given in Zelle (1995). Here we show some of the sentences in English, Spanish, Japanese, and Turkish, along with the logical query representation for each sentence.

What is the capital of the state with the largest population?
Que es la capital de el estado con la poblacion mas grande?
Mottomo ookii jinkou ga aru shuu no shuto wa nan desu ka?
Nufusu en fazla olan eyaletin baskenti nedir?
answer(C, (capital(S,C), largest(P, (state(S), population(S,P))))).

What state is Des Moines located in?
En que estado se encuentra Des Moines?
Demoin wa dono shuu ni arimasu ka?
Des Moines hangi eyalettedir?
answer(S, (state(S), eq(C,cityid('des moines',_)), loc(C,S))).

What capital is the largest in the US?
Que capital es la mas grande en los US?
Gasshuukoku no mottomo ookii shuto wa nan desu ka?
US deki en genis baskent hangisidir?
answer(A, largest(A, capital(A))).

What is the area of Maine?
Que es la area de Maine?
Meen no chiiki wa nan desu ka?
Maine nin yuzolcumu nedir?
answer(A, (area(C,A), eq(C,stateid(maine)))).

What is the total population of the states that border Texas?
Que es el total de la poblacion de los estados que bordean a Texas?
Tekisasu ni rinsetsu suru shuu no zentai no jinkou wa ikura desu ka?

Texas a komsu eyaletlerin toplam nufusu nedir?
answer(P, (population(S,P), state(S), next_to(S,M), eq(M,stateid(texas)))).

What is the highest point in the state with the capital Des Moines?
Que es el punto mas alto de el estado con la capital Des Moines?
Demoin no shuto ga aru shuu no mottomo takai chiten wa nan desu ka?
Baskenti Des Moines olan eyaletin en yuksek noktasi nedir?
answer(C, (high_point(B,C), loc(C,B), state(B), capital(B,A),
    eq(A,cityid('des moines',_)))).

A.2 New Geography Queries

Below are sample queries from the new sentences collected from computer science students.

What is the longest river that does not run through Texas?
answer(B, longest(B, (river(B), \+ (traverse(B,A), eq(A,stateid(texas)))))).

What is the population density in the state with capital Austin?
answer(C, (density(B,C), state(B), capital(B,A), eq(A,cityid(austin,_)))).

Which states have a major city named Austin?
answer(B, (state(B), loc(A,B), major(A), city(A), eq(A,cityid(austin,_)))).

How many rivers do not traverse the state with the capital Albany?
answer(A, count(B,(river(B), \+ (traverse(B,C), state(C),
    capital(C,D), eq(D,cityid(albany,_)))),A))).

How many states border on the state whose capital is Boston?
answer(C, count(N, (state(N), next_to(N,W), state(W), capital(W,B), eq(B,cityid(boston,_))),C))).

How many states border at least one other state?
answer(C, count(S, (state(S), next_to(S,R), state(R)),C))).

Of the states washed by the Mississippi River which has the lowest point?
answer(B, lowest(L, (state(B), traverse(A,B), eq(A,riverid(mississippi)), low_point(B,L))))).

How many states have a higher point than the highest point of the state with the largest capital city in the US?
answer(M, count(S, (state(S), high_point(S,P), higher(P,Q), high_point(B,Q),
    largest(C, (state(B), capital(B,C)))),M))).

What is the length of the longest river that runs through Texas?
answer(C, longest(R, (len(R,C), river(R), traverse(R,T),
    eq(T,stateid(texas)))))).

What is the largest state that borders the state with the lowest point in the USA?
answer(S, largest(S, lowest(P, (state(S), next_to(S,L), state(L),
    low_point(L,P)))))).

Appendix B

Sample Learned Lexicons

This appendix contains examples of the lexicons learned by WOLFIE and by Siskind's system in the U.S. geography query domain. All lexicons were learned using 225 training examples.

B.1 English

WOLFIE's learned lexicon for one split:

```
[state],[state(_)].                 [states],[state(_)].
[rivers],[river(_)].                [border],[next_to(_,_)].
[population],[population(_,_)].     [through],[traverse(_,_)].
[cities],[city(_)].                 [largest],[largest(_,_)].
[major],[major(_)].                 [lowest],[low_point(_,_)].
[smallest],[smallest(_,_)].         [area],[area(_,_)].
[longest],[longest(_,_)].           [biggest],[largest(_,_)].
[shortest],[shortest(_,_)].         [long],[len(_,_)].
[have],[loc(_,_)].                  [large],[size(_,_)].
[length],[len(_,_)].                [points],[high_point(_,_)].
[50],[sum(_,_,_)].                  [highest],[largest(_,_)].
[most],[most(_,_,_)].               [high],[elevation(_,_)].
[populations],[population(_,_)].[surround],[next_to(_,_)].
[surrounding],[next_to(_,_)].       [whats],[city(_)].
[not],[\+_].                        [elevation],[highest(_,_)].
[lowest],[smallest(_,_)].           [usa],[city(_)].
[total],[sum(_,_,_)].               [point],[highest(_,_)].
[has],[loc(_,_)].                   [city],[city(_)].
[highest],[high_point(_,_)].        [in],[loc(_,_)].
[capital],[capital(_,_)].           [river],[river(_)].
[live],[population(_,_)].           [density],[density(_,_)].
[many],[count(_,_,_)].              [borders],[next_to(_,_)].
[bordering],[next_to(_,_)].         [greatest],[largest(_,_)].
[capital],[capital(_)].             [country],[highest(_,_)].
[cross],[traverse(_,_)].            [big],[area(_,_)].
[higher],[higher(_,_)].             [citizens],[population(_,_)].
[has],[largest(_,_)].               [elevation],[elevation(_,_)].
[of],[loc(_,_)].                    [most],[(population(_,A),largest(A,_))].
```

Siskind's system on the same training input learned the following lexicon. Notice that his system explicitly lists empty meanings for words.

```
[how], [count(_,_,_) ].                       [how], [].
[how], [state].                               [long], [len(_,_)].
[river], [state].                             [river], [].
[lowest], [(smallest(_,_), state(_) )].       [lowest], [low_point(_,_)].
[point], [high_point(_,_)].                   [point], [].
[point], [(highest(_,_), high_point(_,_))].   [point], [country].
[in], [].                                     [in], [state].
[in], [loc(_,_)].                             [in], [city].
[in], [].                                     [in], [( capital(_))].
[population], [population(_,_)].              [population], [].
[of], [].                                     [of], [population(_,_)].
[combined], [].                               [all], [].
[all], [( state(_))].                         [50], [].
[states], [state(_)].                         [states], [].
[spot], [state].
[biggest], [(largest(_,_), river(_), loc(_,_))].
[biggest], [(largest(_,_), city(_), loc(_,_))].
[biggest], [(largest(_,_))].
[city], [( state(_), capital(_,_))].
[city], [].                                   [city], [city(_)].
[many], [].                                   [many], [count(_,_,_)].
[citizens], [population(_,_)].                [live], [].
[high], [(elevation(_,_), high_point(_,_))].  [high], [].
[highest], [high_point(_,_)].                 [highest], [].
[highest], [capital(_,_)].                    [run], [].
[through], [( state)].                        [state], [( state(_))].
[state], [(state(_), loc(_,_))].              [state], [].
[largest], [(largest(_,_), state(_))].        [largest], [largest(_,_)].
[largest], [(largest(_,_), city(_), loc(_,_))].
[largest], [(largest(_,_), river(_), loc(_,_))].
[area], [area(_,_)].
[people], [(population(_,_), capital(_,_) )]. [people], [population(_,_)].
[capital], [].                                [capital], [capital(_,_)].
[rivers], [(count(_,_,_), river(_), loc(_,_))].
[rivers], [(river(_), loc(_,_))].
[rivers], [river(_)].                         [border], [next_to(_,_)].
[located], [(state(_), loc(_,_))].            [points], [].
[surrounding], [].                            [has], [state(_)].
[has], [(state(_), loc(_,_), city(_))].
[has], [(largest(_,_), high_point(_,_), elevation(_,_))].
[greatest], [largest(_,_)].
[density], [(largest(_,_), density(_,_))].
[density], [density(_,_)].                    [most], [].
[populated], [(largest(_,_), population(_,_) )].
[bordering], [next_to(_,_)].
[major], [(major(_), loc(_,_))].
[major], [major(_)].
[smallest], [(smallest(_,_), state(_))].
[smallest], [(smallest(_,_), city(_), loc(_,_))].
[smallest], [smallest(_,_)].
[longest], [(longest(_,_), river(_), loc(_,_))].
```

```
[longest], [(longest(_,_), river(_))].
[cities], [(city(_), loc(_,_))].             [go], [state(_)].
[usa], [].
[contains], [(state(_), loc(_,_),  high_point(_,_))].
[shortest], [(shortest(_,_), river(_), loc(_,_))].
[shortest], [(shortest(_,_), river(_) )].
[borders], [(state(_), next_to(_,_))].
[borders], [(most(_,_,_), next_to(_,_))].
[peak], [].                                 [country], [].
[have], [state].                            [have], [].
[named], [].                                [us], [].
[us], [capital(_)].                         [not], [\+_].
[whats], [].                                [flow], [].
[length], [len(_,_)].                       [total], [sum(_,_,_)].
[elevation], [(highest(_,_))].              [elevation], [state].
[higher], [(state(_), higher(_,_))].        [meters], [state].
[runs], [].                                 [surround], [(next_to(_,_))].
[united], [].
[cross], [(count(_,_,_), traverse(_,_))].
[big], [area(_,_)].                         [mountain], [high_point(_,_)].
[large], [size(_,_)].
[large], [(size(_,_), city(_))].
[populations], [].
[populous], [(largest(_,_), population(_,_))].
```

B.2 Spanish

```
[estado],[state(_)].              [estados],[state(_)].
[grande],[largest(_,_)].          [alto],[high_point(_,_)].
[bordean],[next_to(_,_)].         [rios],[river(_)].
[poblacion],[population(_,_)].    [por],[traverse(_,_)].
[ciudades],[city(_)].             [capital],[capital(_,_)].
[bajo],[low_point(_,_)].          [viven],[population(_,_)].
[area],[area(_,_)].               [largo],[len(_,_)].
[densidad],[density(_,_)].        [pequeno],[smallest(_,_)].
[cuantos],[count(_,_,_)].         [elevacion],[high_point(_,_)].
[bordea],[next_to(_,_)].          [nombre],[loc(_,_)].
[bordeando],[next_to(_,_)].       [grande],[size(_,_)].
[combinada],[sum(_,_,_)].         [capital],[capital(_)].
[estado],[loc(_,_)].              [contiene],[loc(_,_)].
[cuadrados],[area(_,_)].          [pais],[highest(_,_)].
[rodean],[next_to(_,_)].          [poblaciones],[population(_,_)].
[tamano],[area(_,_)].             [total],[sum(_,_,_)].
[se],[loc(_,_)].                  [baja],[smallest(_,_)].
[montana],[high_point(_,_)].      [alta],[largest(_,_)].
[no],[\+_].                       [todos],[loc(_,_)].
[alto],[elevation(_,_)].          [alta],[highest(_,_)].
[cuantas],[count(_,_,_)].         [us],[city(_)].
[pequena],[smallest(_,_)].        [us],[river(_)].
[largo],[longest(_,_)].           [corto],[shortest(_,_)].
[rio],[river(_)].                 [usa],[city(_)].
[puntos],[high_point(_,_)].       [altos],[next_to(_,_)].
[puntos],[higher(_,_)].           [ciudadanos],[population(_,_)].
```

```
[estados],[highest(_,_)].
[ciudad],[city(_)].
[mayores],[major(_)].
[cual],[most(_,_,_)].
[personas],[population(_,_)].
[poblado],[population(_,_)].
[tiene],[elevation(_,_)].
[cual],[loc(_,_)].
[en],[loc(_,_)].
[cruzan],[traverse(_,_)].
[punto],[highest(_,_)].
[mas],[largest(_,_)].
[tiene],[loc(_,_)].
[de],[loc(_,_)].
```

B.3 Turkish

```
[yuksek],[high_point(_,_)].
[hangi],[state(_)].
[sehir],[city(_)].
[nehir],[river(_)].
[buyuk],[largest(_,_)].
[daki],[loc(_,_)].
[alcak],[low_point(_,_)].
[nehirler],[river(_)].
[baslica],[major(_)].
[tane],[count(_,_,_)].
[sehirler],[city(_)].
[genis],[largest(_,_)].
[uzun],[longest(_,_)].
[eyalet,nedir],[state(_)].
[sehri],[city(_)].
[komsu],[next_to(_,_)].
[cok],[most(_,_,_)].
[eyaletinin],[state(_)].
[yi],[next_to(_,_)].
[50],[sum(_,_,_)].
[nufusa],[population(_,_)].
[nehri,nedir],[river(_)].
[deki],[loc(_,_)].
[akar],[traverse(_,_)].
[kilometrekaredir],[area(_,_)].
[alan],[area(_,_)].
[sahiptirler],[higher(_,_)].
[eyaletindeki],[loc(_,_)].
[uzunluktadir],[len(_,_)].
[bulunan],[state(_)].
[taki],[loc(_,_)].
[aki],[loc(_,_)].
[nehirin],[river(_)].
[genistir],[size(_,_)].
[baskentidir],[capital(_,_)].
[genisliktedir],[size(_,_)].
[eyaletin,en],[loc(_,_)].
[bakimindan],[capital(_)].
[sehre],[city(_)].
[sahiptir],[largest(_,_)].
[eyaletten],[state(_)].
[buyuktur],[area(_,_)].
[nehri,kac],[count(_,_,_)].
[ulkedeki],[const(_,countryid(usa))].
[siniri],[next_to(_,_)].
[nufusu],[population(_,_)].
[a],[state(_)].
[gecer],[traverse(_,_)].
[baskenti],[capital(_,_)].
[nokta],[const(_,countryid(usa))].
[yasamaktadir],[population(_,_)].
[uzunlugu],[len(_,_)].
[kucuk],[smallest(_,_)].
[yuzolcumu],[area(_,_)].
[vardir],[loc(_,_)].
[eyaletin],[state(_)].
[kisa],[shortest(_,_)].
[eyalettedir],[loc(_,_)].
[yogunlugu],[density(_,_)].
[yuksektir],[elevation(_,_)].
[alana],[area(_,_)].
[nehirlerin],[river(_)].
[gecmektedir],[traverse(_,_)].
[nun,kac],[river(_)].
[sahiptir],[highest(_,_)].
[sehirlerin],[city(_)].
[baskentinde],[capital(_,_)].
[baskent],[capital(_)].
[uzundur],[len(_,_)].
[baskentinin],[capital(_,_)].
[kucuktur],[smallest(_,_)].
[fazla],[largest(_,_)].
[alani],[area(_,_)].
[yusek],[high_point(_,_)].
[iceren],[highest(_,_)].
[vatandas],[population(_,_)].
[eyaletlerdeki],[highest(_,_)].
[yogunluguna],[density(_,_)].
[yuksek],[largest(_,_)].
[nufus],[population(_,_)].
[sahiptir],[loc(_,_)].
[nehre],[river(_)].
[insana],[population(_,_)].
[acisindan],[loc(_,_)].
[buyuktur],[largest(_,_)].
[eyalete],[state(_)].
```

```
[yukseltiye],[elevation(_,_)].
[eyaletlerin],[state(_)].
[nufuslari],[population(_,_)].
[gecen],[state(_)].
[nin],[loc(_,_)].
[gecmez],[\+_].
[kalabalik],[population(_,_)].
[az],[(smallest(A,_),population(_,A))].
[toplam],[sum(_,_,_)].
[icinden],[traverse(_,_)].
[eyaletlerden],[traverse(_,_)].
[ya],[state(_)].
[hangi],[traverse(_,_)].
[en],[largest(_,_)].
```

B.4 Japanese

```
[shuu],[state(_)].
[gasshoukoku],[const(_,countryid(usa))].
[rinsetsu],[next_to(_,_)].
[jinkou,wa],[population(_,_)].
[kawa,wa],[river(_)].
[yooku],[const(_,stateid('new york'))].
[nagasa],[len(_,_)].
[nannin],[population(_,_)].
[arimasu,ka],[loc(_,_)].
[ikutsu],[count(_,_,_)].
[kawa,ga],[river(_)].
[mitsudo],[density(_,_)].
[jinkou,ga],[population(_,_)].
[mijikai],[shortest(_,_)].
[ooku],[most(_,_,_)].
[ookisa],[size(_,_)].
[shuto,wa],[capital(_)].
[kawa,wo],[river(_)].
[ga,arimasu],[highest(_,_)].
[mawari],[next_to(_,_)].
[arimsua],[highest(_,_)].
[gojuu],[sum(_,_,_)].
[chiten,ga],[highest(_,_)].
[amerika],[river(_)].
[kaibatsu,ga],[elevation(_,_)].
[daitoshi],[major(_)].
[ooi],[largest(_,_)].
[hotondo],[largest(_,_)].
[wo],[traverse(_,_)].
[suru],[loc(_,_)].
[nan],[size(_,_)].
[takai],[high_point(_,_)].
[mottomo,ookii],[largest(_,_)].
[toshi,wa],[city(_)].
[nagarete],[traverse(_,_)].

[chiisai],[smallest(_,_)].
[hikui],[low_point(_,_)].
[chiiki],[area(_,_)].
[shuto],[capital(_,_)].
[chuto],[capital(_,_)].
[nagai],[longest(_,_)].
[desu],[loc(_,_)].
[toshi],[city(_)].
[ookina],[major(_)].
[nagareru],[traverse(_,_)].
[takasa],[elevation(_,_)].
[jinkou,de],[population(_,_)].
[agete],[loc(_,_)].
[okisa],[area(_,_)].
[motto],[higher(_,_)].
[heihou],[area(_,_)].
[nagai],[largest(_,_)].
[suru,shuu],[highest(_,_)].
[wa,mottomo],[largest(_,_)].
[wa],[city(_)].
[jinkou],[population(_,_)].
[hito],[population(_,_)].
[donna],[state(_)].
[shimasu],[major(_)].
[kawa],[river(_)].
[no],[loc(_,_)].
```

Bibliography

Amsler, A. (1981). A taxonomy for english verbs and nouns. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, pp. 133–138 Stanford, CA.

Anderson, J. R. (1977). Induction of augmented transition networks. *Cognitive Science*, *1*, 125–157.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Bailey, D. R. (1997). *When Push Comes to Shove: A Computational Model of the Role of Motor Control in the Acquisition of Action Verbs*. Ph.D. thesis, University of California at Berkeley, Berkeley, CA.

Basili, R., Pazienza, M., & Velardi, P. (1996). An empirical symbolic approach to natural language processing. *Artificial Intelligence*, *85*, 59–99.

Beckwith, R., Fellbaum, C., Gross, D., & Miller, G. (1991). WordNet: A lexical database organized on psycholinguistic principles. In Zernik, U. (Ed.), *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pp. 211–232. Lawrence Erlbaum, Hillsdale, NJ.

Berwick, R. (1983). Learning word meanings from examples. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 459–461.

Berwick, R. C., & Pilato, S. (1987). Learning syntax by automata induction. *Machine Learning*, *2*(1), 9–38.

Boguraev, B., & Briscoe, E. (Eds.). (1989). *Computational lexicography for natural language processing*. Longman, London.

Borland International (1988). *Turbo Prolog 2.0 Reference Guide*. Borland International, Scotts Valley, CA.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *14*, 123–140.

Brent, M. (1990). 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*, pp. 428–437.

Brent, M. (1991). Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 209–214.

Brill, E., & Mooney, R. (1997). An overview of empirical natural language processing. *AI Magazine*, *18*(4), 13–24.

Brunk, C., & Pazzani, M. (1995). A lexically based semantic bias for theory revision. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 81–89 San Francisco, CA. Morgan Kaufman.

Byrd, R., Calzolari, N., Chodorow, M., Klavans, J., Neff, M., & Risk, O. (1987). Tools and methods for computational lexicology. *Computational Linguistics*, *13*(3), 219–240.

Califf, M. E. (1998). *Relational Learning Techniques for Natural Language Information Extraction.* Ph.D. thesis, University of Texas, Austin, TX.

Cardie, C. (1993). A case-based apprach to knowledge acquisition for domain-specific sentence analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 798–803.

Catizone, R., Russell, G., & Warwick, S. (1993). Deriving translation data from bilingual texts. In *Proceedings of the First International Lexical Acquisition Workshop.*

Church, & Hanks (1990). Word association norms, mutual information and lexicography. *Computational Linguistics*, *16*(1), 22–29.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*(2), 201–221.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 150–157 San Francisco, CA. Morgan Kaufman.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Dyer, M. (1991). Lexical acquisition through symbol recirculation in distributed connectionist networks. In Zernik, U. (Ed.), *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pp. 309–337. Lawrence Erlbaum, Hillsdale, NJ.

Engelson, S., & Dagan, I. (1996). Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* Santa Cruz, CA.

Estlin, T. A., & Mooney, R. (1997). Learning to improve both efficiency and quality of planning. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 1227–1232.

Feldman, J., Lakoff, G., & Shastri, L. (1995). The neural theory of language project `http://www.icsi.berkeley.edu/ntl`. International Computer Science Institute, University of California, Berkeley, CA.

Fillmore, C. J. (1968). The case for case. In Bach, E., & Harms, R. T. (Eds.), *Universals in Linguistic Theory*. Holt, Reinhart and Winston, New York.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Fukumoto, F., & Tsujii, J. (1995). Representation and acquisition of verbal polysemy. In *Papers from the 1995 AAAI Symposium on the Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, pp. 39–44 Stanford, CA.

Gale, W., & Church, K. (1991). Identifying word correspondences in parallel texts. In *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, NY.

Granger, R. (1977). FOUL-UP: a program that figures out meanings of words from context. In *Proceedings of the Fifth International Joint Conference on Artificial intelligence*, pp. 172–178.

Grimshaw, J. (1981). Form, function, and the language acquisition device. In Baker, C., & McCarthy, J. (Eds.), *The logical problem of language acquisition*, pp. 165–182. M.I.T. Press, Cambridge, MA and London, England.

Haruno, M. (1995). A case frame learning method for Japanese polysemous verbs. In *Papers from the 1995 AAAI Symposium on the Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, pp. 45–50 Stanford, CA.

Hastings, P., & Lytinen, S. (1994). The ups and downs of lexical acquisition. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 754–759.

Holte, R. C., Acker, L., & Porter, B. (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 813–818 Detroit, MI.

Jackendoff, R. (1990). *Semantic Structures*. The MIT Press, Cambridge, MA.

Johnston, M., Boguraev, B., & Pustejovsky, J. (1995). The acquisition and interpretation of complex nominals. In *Papers from the 1995 AAAI Symposium on the Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, pp. 69–74 Stanford, CA.

Karov, Y., & Edelman, S. (1996). Learning similarity-based word sense disambiguation from sparse data. In *Proceedings of the Fourth Workshop on Very Large Corpora* Copenhagen.

Knight, K. (1996). Learning word meanings by instruction. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 447–454 Portland, Or.

Kohavi, R., & John, G. (1995). Automatic parameter selection by minimizing estimated error. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 304–312 Tahoe City, CA.

Kumano, A., & Hirakawa, H. (1994). Building an MT dictionary from parallel texts based on linguistic and statistical information. In *Proceedings of the Fifteenth International Conference on Computational Linguistics*.

Lavrač, N., & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning*, *2*(2), 103–138.

Leech, G. (1974). *Semantics*. Penguin Books Inc.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 148–156 San Francisco, CA. Morgan Kaufman.

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 591–596 Providence, RI.

Lund, K., Burgess, C., & Atchley, R. A. (1995). Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pp. 660–665.

Lytinen, S., & Roberts, S. (1989). Lexical acquisition as a by-product of natural language processing. In *Proceedings of the First International Lexical Acquisition Workshop* Detroit, MI.

Manning, C. D. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 235–242 Columbus, Ohio.

Matan, O. (1995). On-site learning. Tech. rep. LOGIC-95-4, Logic Group, Stanford University.

McClelland, J. L., & Kawamoto, A. H. (1986). Mechanisms of sentence processing: Assigning roles to constituents of sentences. In Rumelhart, D. E., & McClelland, J. L. (Eds.), *Parallel Distributed Processing, Vol. II*, pp. 318–362. MIT Press, Cambridge, MA.

Melamed, I. (1995). Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Proceedings of the Third Workshop on Very Large Corpora*.

Melamed, I. (1996). Automatic construction of clean broad-coverage translation lexicons. In *Second Conference of the Association for Machine Translation in the Americas*.

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. MIT Press, Cambridge, MA.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1993). Introduction to WordNet: An on-line lexical database. Available by ftp to clarity.princeton.edu.

Mooney, R. J. (1990). *A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding*. Pitman Publishers, London.

Muggleton, S. H. (Ed.). (1992). *Inductive Logic Programming*. Academic Press, New York, NY.

Nenov, V. I., & Dyer, M. G. (1994). Perceptually grounded language learning: Part 2–DETE: A neural/procedural model. *Connection Science*, *6*(1), 3–41.

Pazzani, M. J. (1985). Explanation and generalization based memory. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pp. 323–328 Irvine, CA.

Pedersen, T., & Chen, W. (1995). Lexical acquisition via constraint solving. In *Papers from the 1995 AAAI Symposium on the Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, pp. 118–122 Stanford, CA.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* Columbus, OH.

Plotkin, G. D. (1970). A note on inductive generalization. In Meltzer, B., & Michie, D. (Eds.), *Machine Intelligence (Vol. 5)*. Elsevier North-Holland, New York.

Proctor, P. (Ed.). (1978). *Longman Dictionary of Contemporary English*. Longman Group, Harlow, Essex, UK.

Rajagopalan, R., & Kuipers, B. (1994). The figure understander: a system for integrating text and diagram input to a knowledge base. In *Proceedings of the Seventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* Seattle, WA.

Regier, T. (1991). Learning spatial concepts using a partially-structured connectionist architecture. Tech. rep. TR-91-050, Berkeley.

Resnik, P. (1995). Disambiguating noun groupings with respect to WordNet senses. In *Proceedings of the Third Workshop on Very Large Corpora*, pp. 54–68 Cambridge, MA.

Rigau, G., Rodríguez, H., & Agirre, E. (1998). Building accurate semantic taxonomies from monolingual mrds. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pp. 1103–1109 Montreal, CA.

Riloff, E., & Sheperd, K. (1997). A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pp. 117–124 Providence, Rhode Island.

Russell, D. (1993). *Language Acquisition in a Unification-Based Grammar Processing System Using a Real World Knowledge Base*. Ph.D. thesis, University of Illinois, Urbana, IL.

Salveter, S. (1979). Inferring conceptual graphs. *Cognitive Science*, *3*(2), 151–166.

Salveter, S. (1982). Inferring building blocks for knowledge representation. In Lehnert, W., & Ringle, M. (Eds.), *Strategies for Natural Language Processing*, pp. 327–344. Lawrence Erlbaum, Hillsdale, NJ.

Schank, R. C. (1975). *Conceptual Information Processing.* North-Holland, Oxford.

Selfridge, M. (1986). A computer model of child language learning. *Artificial Intelligence*, *29*(2).

Siklossy, L. (1972). Natural language learning by computer. In Simon, H. A., & Siklossy, L. (Eds.), *Representation and meaning: Experiments with Information Processsing Systems.* Prentice Hall, Englewood Cliffs, NJ.

Siskind, J. M. (1992). *Naive Physics, Event Perception, Lexical Semantics and Language Acquisition.* Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Siskind, J. M. (1994). Lexical acquisition in the presence of noise and homonymy. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 760–766.

Siskind, J. M. (1996). A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, *61*(1), 39–91.

Smadja, F. (1991). From n-grams to collocations: An evaluation of Xtract. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics.*

Suppes, P., Liang, L., & Böttner, M. (1991). Complexity issues in robotic machine learning of natural language. In Lam, L., & Naroditsky, V. (Eds.), *Modeling Complex Phenomena, Proceedings of the 3rd Woodward Conference*, pp. 102–127. Springer-Verlag.

Thompson, K., & Langley, P. (1991). Concept formation in structured domains. In Fisher, D., Pazzani, M., & Langley, P. (Eds.), *Concept Formation: Knowledge and Experience in Unsupervised Learning*, pp. 127–161. Morgan Kaufman, San Mateo, CA.

Tishby, N., & Gorin, A. (1994). Algebraic learning of statistical associations for language acquisition. *Computer Speech and Language*, *8*, 51–78.

Tomita, M. (1986). *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, Boston.

Walker, D., & Amsler, R. (1986). The use of machine-readable dictionaries in sublanguage analysis. In Grishman, R., & Kittredge, R. (Eds.), *Analyzing Language in Restricted Domains*, pp. 69–83. Lawrence Erlbaum Associates, Hillsdale, NJ.

Webster, M., & Marcus, M. (1995). Automatic acquisition of the lexical semantics of verbs from sentence frames. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics.*

Wu, D., & Xia, X. (1995). Large-scale automatic extraction of an English-Chinese translation lexicon. *Machine Translation*, *9*(3-4), 285–313.

Zelle, J. M. (1995). *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers.* Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 96-249.

Zelle, J. M., & Mooney, R. J. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 817–822 Washington, D.C.

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* Portland, OR.

Zernick, U. (1991). Train1 vs. train2: Tagging word senses in corpus. In Zernik, U. (Ed.), *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pp. 91–112. Lawrence Erlbaum, Hillsdale, NJ.

Zernick, U., & Dyer, M. (1987). The self-extending phrasal lexicon. *Computational Linguistics*, *13*(3-4), 308–327.

Zernik, U. (1987). Language acquisition: Learning a hierarchy of phrases. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 125–132 Milan, Italy.

Zipf, G. (1949). *Human behavior and the principle of least effort.* Addison-Wesley, New York, NY.

Vita

Cynthia Ann Thompson was born in Kingston, New York, on June 10, 1966, the daughter of Nannette Ray and Ralph Edward Thompson. After completing her work at Independence High School in Charlotte, North Carolina, she entered North Carolina State University in Raleigh, North Carolina. During four separate semesters in college, she worked at NASA/GSFC in Greenbelt, Maryland, as a participant in the cooperative education program. She received the Bachelor of Science in Computer Science from North Carolina State University in May, 1989, graduating magna cum laude. During the next two years, she was employed as a consultant with the Information Consulting Group, which became a part of McKinsey & Company during her employment. In August, 1991, she entered the Masters program of the Department of Computer Sciences at the University of Texas at Austin. In the Spring of 1993, she was inducted into The Honor Society of Phi Kappa Phi. In the summer of 1993, she completed a Master's Thesis and received the Master of Arts degree in Computer Sciences. Cynthia married William Pierce in March of 1996, and she and her husband have two pets, a dog Loki and a cat Romulus. Cynthia has accepted a postdoctoral fellowship at the Center for the Study of Language and Information, at Stanford University, where she will begin working as soon as possible after her defense.

Permanent Address: 3925 La Donna Avenue
Palo Alto, CA 94306

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.