Incremental Nonmonotonic Parsing through Semantic Self-Organization

Marshall R. Mayberry, III

Report AI-TR-04-310 Apr 2004

martym@cs.utexas.edu
http://www.cs.utexas.edu/users/nn/

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

Copyright

by

Marshall Reeves Mayberry, III

2003

The Dissertation Committee for Marshall Reeves Mayberry, III
certifies that this is the approved version of the following dissertation:

Incremental Nonmonotonic Parsing through Semantic Self-Organization

Committee:

Risto Miikkulainen, Supervisor

Raymond Mooney

Benjamin Kuipers

Bruce Porter

Stephen Wechsler

Incremental Nonmonotonic Parsing through Semantic Self-Organization

by

Marshall Reeves Mayberry, III, B.S., B.S., M.S.

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

August 2003

Acknowledgments

Many people have contributed in one way or another to the completion of this dissertation and of my graduate studies. I sincerely thank you all. The encouragement, guidance and support I have always received from my adviser Risto Miikkulainen throughout all of these years has been invaluable, as have been the numerous discussions with Raymond Mooney. Also, I have benefitted immeasurably from input from the rest of my committee, Bruce Porter, Benjamin Kuipers, and Steven Wechsler, who guided me on the objectives of my research. I am deeply grateful for the friendship and emotional support I found among other colleagues and friends: Jim and Tasca Bednar, Yoonsuck and Ahran Choe, Tino Gomez and his brother Oliver. Mike and Darla Hewett, Cindy Thompson and Bill Pierce, Mike and Yumiko, thanks for many years of friendship, hikes, ice cream, and chess. I am very grateful also to Naren, Sowmya, Thomas and Yuko, Evangeline, Jun, and Ezra for putting up with my never-ending questions about your language, your music, and your culture. It has been fascinating getting to know you all.

Deep gratitude also goes to my parents, Sally and Reeves Mayberry, for always believing in me and their unconditional support in everything I have set out to do. To the rest of my family René, Trellis, and Kim; Liba, Ruby, Nora; Angie, Aziz, Yasmin, and Ahmed; Charis, Roberto, Rosarito, Robertito, and Alejandro; thank you for your love during this long enterprise. J.R., and Lulu: I immensely enjoyed those summers you all spent with us playing games and feeding squirrels around UT.

But above all, I want to thank my wife, Coquis, for her unwavering and loving support throughout my dissertation ordeal.

MARSHALL R. MAYBERRY, III

The University of Texas at Austin
August 2003

Incremental Nonmonotonic Parsing through Semantic Self-Organization

Publication No. _____

Marshall Reeves Mayberry, III, Ph.D.
The University of Texas at Austin, 2003

Supervisor: Risto Miikkulainen

Subsymbolic systems have been successfully used to model several aspects of human language processing. Subsymbolic parsers are appealing because they allow combining syntactic, semantic, and thematic constraints in sentence interpretation and nonmonotonically revising that interpretation while incrementally processing a sentence. Such parsers are also cognitively plausible: processing is robust and multiple interpretations are simultaneously activated when the input is ambiguous. Yet, it has proven very difficult to scale them up to realistic language. They have limited memory capacity, training takes a long time, and it is difficult to represent linguistic structure.

A new connectionist model, INSOMNet, scales up the subsymbolic approach by utilizing semantic self-organization. INSOMNet was trained on semantic dependency graph representations from the recently-released LinGO Redwoods HPSG Treebank of sentences from the VerbMobil project. The results show that INSOMNet accurately learns to represent these semantic dependencies and generalizes to novel structures. Further evaluation of INSOMNet on the original VerbMobil sentences transcribed with annotations for spoken language demonstrates robust parsing of noisy input, while graceful degradation in performance from adding noise to the network weights underscores INSOMNet's tolerance to damage. Finally, the cognitive plausibility of the model is shown on a standard psycholinguistic benchmark, in which INSOMNet demonstrates expectations and defaults, coactivation of multiple interpretations, nonmonotonicity, and semantic priming.

Contents

List of Figures

Chapter 1

Introduction

All grammars leak. - Edward Sapir

The first year of the new millenium has come and gone, and still HAL seems as much science fiction today as the infamous psychotic computer did in 1968 when *2001: A Space Odyssey* anticipated mankind's first steps on the Moon the following year. If we could talk to computers like modern-day Doolittles as we talk among ourselves, the ramifications would change our lives more fundamentally than did the advent of the computer itself. The formalization of language in the 1950s under Chomsky's generative framework with its emphasis on a Universal Grammar (UG) that is innate and, therefore, predetermined, led to widespread optimism that we could rationally specify the foundation of language and encode that knowledge into a computer. Once the core language faculty was in place, language learning should follow through exposure by simply tuning prespecified parameters.

Yet, roughly half a century later, that optimism has largely waned. Early work in semantic analysis of language had become bogged down in the sheer volume of domain-specific knowledge that had to be built into systems to perform even modest language understanding. These systems also proved to be tremendously brittle when extended beyond the domains on which they were constructed.

The resurgence of an empirical approach to natural language processing over the past fifteen years has come through a variety of statistical learning techniques. These techniques have capitalized on, and in turn fostered the growth of, large corpora from which knowledge can be automatically acquired rather than stipulated *a priori*.

Whereas statistical techniques built on a symbolic basis have proven very successful on tasks that lend themselves to symbolic description, such as part-of-speech tagging and syntactic parsing, another class of statistical model that has a decidedly non-symbolic foundation has proven as equally powerful when applied to domains for which a symbolic description would be virtually impossible, such as modeling human aphasia.

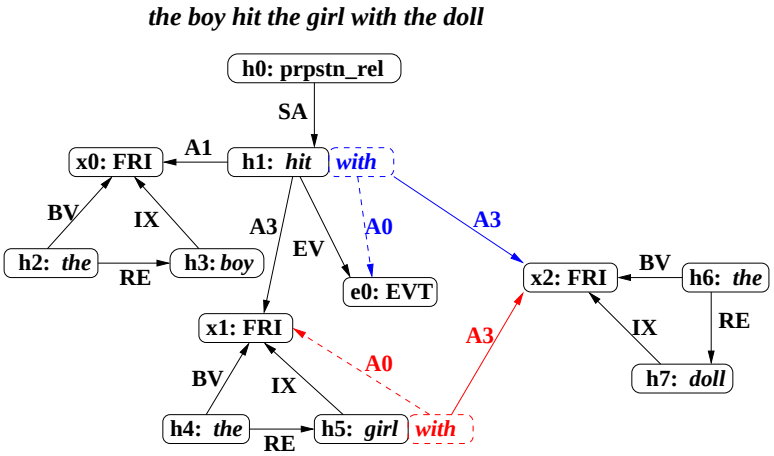


Figure 1.1: **Natural Language Understanding.** (***Color figure***) The primary task of natural language understanding is to transform an input sentence into a semantic representation that shows how the words and constituents of the sentence relate to one another. This figure shows a graphical description of a semantic representation for the sentence *the boy hit the girl with the doll*. The graph nodes represents semantic constituents and the arcs between them denote how they relate to each other. The details of this semantic representation will be explained in Chapter 2.

These *subsymbolic* (alternatively, connectionist or neural network) systems are inspired by the biological model of the brain as a system that processes information through massively interconnected simple processing units based on idealized neurons. In this framework concepts are not represented by discrete symbols, but rather are distributed across the units in such a way that all units contribute to the concept's representation. These *distributed representations* naturally give rise to a number of plausible cognitive behaviors, including robustness to noise, damage, and incomplete information, concept gradation (in which a representation may be simultaneously similar to other concepts), and automatic generalization arising from the same units being used to represent all knowledge items in the system.

1.1 Task and Goals

Despite their success in modeling cognitive effects in natural language processing tasks, subsymbolic models have met with limited success when applied to parsing language into linguistically validated structures.

Figure 1.1 illustrates the main task that will be the central objective of this dissertation. In order to be useful as large-scale cognitive models, subsymbolic systems need to be able to parse a sentence into a detailed semantic representation. Such a linguistic structure indicates how constituents in the sentence, such as words and phrases, are related. Moreover, the semantic formalism used must accommodate the inherent strengths of neural networks to allow for the modeling of

cognitively plausible behavior that has been the primary research focus in the connectionist field.

Why is subsymbolic parsing a desirable goal? The main promise for both cognitive modeling and engineering is that it accurately accounts for the holistic nature and nonmonotonicity of natural language processing. Over the course of the parse, the network maintains a holistic parse representation at the output. Words processed later in a sentence can change the developing representation so that the network can recover from incorrect earlier decisions. This way, the network can more effectively resolve lexical ambiguities, attachments, and anaphoric references during the course of parsing. Indeed, multiple interpretations are maintained in parallel until disambiguating information is encountered in the input stream (cf. Onifer and Swinney 1981; MacDonald et al. 1992; MacDonald 1993). This is evidently how humans process natural language, what good parsers should do, and what subsymbolic parsers promise to deliver.

The purpose of this dissertation is to show that deep semantic parsing of sentences from real-world dialogues is possible using neural networks: a subsymbolic system can be trained to read a sentence with complex grammatical structure into a holistic representation of the semantic features and dependencies of the sentence. This research breaks new ground in two important respects. First, the model described in this dissertation, the Incremental Nonmonotonic Self-Organization of Meaning Network (INSOMNet), is the first subsymbolic system to be applied to parsing sentences into deep semantic representations derived from a hand-annotated treebank of sentences with realistic complexity. Second, whereas almost all previous work has focused on the representation and learning of syntactic tree structures (such as those in the Penn Treebank), the semantic representations taken up in this study are actually dependency graphs. The challenge of developing a subsymbolic scheme for handling graph structures led to the method of self-organizing the semantic frames that serve as the graph nodes. This semantic self-organization in turn results in a number of interesting cognitive behaviors that will be analyzed in this work.

1.2 Guide for the reader

This dissertation is structured as follows. Chapter 2 lays the groundwork for the INSOMNet model by reviewing important issues in the representation of natural language, its cognitive aspects, how neural networks have been applied to model these aspects, and the linguistic foundation for the semantic representations used by INSOMNet. Chapter 3 then describes the INSOMNet model in detail. In Chapters 4 and 5, INSOMNet is evaluated on a hand-annotated treebank of real-world sentences using two variations of semantic dependency graph, one of which is relatively elementary, and the other much more detailed. In Chapter 6, the robustness of INSOMNet is evaluated on the original speech-annotated sentences from which the sentences in the treebank were derived, as well as tested for how well the network can tolerate noise added to its weights. Chapter 7 evaluates INSOMNet's validity with respect to cognitive modeling. Future research directions are presented in Chapter 8. Finally, Chapter 9 gives a summary of the dissertation research and results.

We have followed a number of conventions throughout the dissertation for purposes of clarity of exposition:

We rather than "I" is used even though there is a single author because "I" sounds a little too pretentious to that author.

Italic typeface is used for words, tokens, and sentences. Every sentence can be assumed to have a stop symbol, usually the period. This typeface is also used for system parameters, and linguistic terms.

Bold typeface is used for linguistic annotations, such as frames and slot names.

Sans Serif is used for architectural components and linguistic feature names.

Color Scheme signifies low activations ($0.0, 0.3$) with shades of blue, intermediate values $[0.3, 0.7]$ with green, yellow or light brown, and high activations ($0.7, 1.0$) with reds to magentas. The extrema are white (0) and black (1).

Chapter 2

Background

Since at least Panini in the 5th century BCE, the human mind has seen structure in natural language and has sought to define it. The most natural way for us to describe language is to circumscribe it, to delimit it–a direct appeal to our thought processes, which have evolved as highly efficient classifiers. Thus, just as Panini systematically described Sanskrit with some 4000 sutras in his opus *Astadhyayi* and, in the process, anticipated much of modern formal language theory, people have sought to distill every aspect of language into crystalline purity. This drive, hardly abated, still fuels the controversy between competence and performance, the presumed modularity of the language faculty, the associated nativist debate, and a host of other issues that, in effect, subscribe to the notion of Leibniz' calculus of reasoning, an essentially symbolic language of thought, wherein any dispute could be resolved with the words, "Gentlemen, let's compute."

In this chapter we review the approaches that have been taken to understanding human sentence processing, the tasks in which they have had the most success, and the characteristics of the approaches that have contributed to those successes. Because the model described in this dissertation is built on a connectionist foundation, we will also discuss the cognitive issues of semantic parsing, ambiguity, incremental processing, and nonmonotonicity with respect to their plausibility, that has been the goal of most work in connectionist modeling. With these issues in mind, we will review related work in connectionist natural language processing. Finally, we will describe the linguistic foundation that has informed the design of the INSOMNet model that is the subject of the current work, and conclude with how all of these pieces will be brought together in the coming chapters.

2.1 Approaches to Natural Language Understanding

As Steven Abney aptly describes it, the "holy grail" of linguistics is to describe language in the way that it is produced, comprehended, acquired, and the way it varies and changes (Abney 1996). In short, human language is the focus, and we researchers want to build systems that will ultimately

S	→ **DP VP**	**D**	→ ***the***
DP	→ **D NP**	**N**	→ ***boy***
NP	→ **N**	**N**	→ ***girl***
NP	→ **N PP**	**N**	→ ***hammer***
VP	→ **V**	**N**	→ ***doll***
VP	→ **V DP**	**V**	→ ***hit***
VP	→ **VP PP**	**V**	→ ***moved***
PP	→ **P DP**	**P**	→ ***with***

Figure 2.1: **Symbolic Grammar.** A simple *context-free grammar* defines how a sentence can be represented as a set of rule expansions from the nonterminal start symbol **S** to the words in the sentence, the terminals *the*, *boy*, *girl*, *hammer*, *doll*, *hit*, *moved*, and *with*. A sentence **S** can be expanded into a determiner phrase **DP** and a verb phrase **VP**. The **DP** can, in turn, be expanded into a determiner **D** and a noun phrase **NP**. The **NP** can be rewritten as a bare noun **N** or **N** optionally followed by a prepositional phrase **PP**. The **VP** may be an intransitive verb **V**, a transitive verb **V** followed by a **DP**, or expanded recursively into another **VP** and **PP**. The **PP** breaks out into a preposition **P** and a **DP**. The nonterminals, **D**, **N**, **V**, and **P**, can be rewritten with their associated words, as shown in the right column. Notice that this grammar allows the **PP** to either modify a **DP** or a **VP** to account for the prepositional phrase attachment ambiguity in the sentence *the boy hit the girl with the doll*. The ability of a CFG to represent ambiguity and recursion in human language makes it a powerful computational tool in NLP.

communicate with us. But, in order to do so seamlessly, our systems will have to essentially act human. They will have to, in some way, capture the human side of language use, not just an idealization of it. It is an exciting undertaking, for nothing will more clearly define who we are than the insights that are revealed in building an agent that can communicate with us on our own terms.

It is useful to see where we are in this process of building systems that understand language, and how the approaches researchers have taken have shaped our view of language.

2.1.1 Symbolic

The paradigm of the descriptive, structured, and compositional approach to natural language processing is the symbolic grammar. In its simplest form, a grammar is a set of rules that defines how the constituents of a sentence are related, from the most basic (e.g., words), to the most encompassing (e.g., the sentence, **S**, or its alias, the "start" symbol). A great many variations on grammar have been explored in computational linguistics. One simple but especially powerful idea, the *context-free grammar* (CFG) adds recursion to a finite set of rules to generate an infinite language, allowing constituents to be embedded within themselves. Figure 2.1 gives an example of a simple CFG for a small segment of English. The grammar is specified by a set of rules that shows how a *nonterminal* category can be expanded into one or more other nonterminal categories or replaced by the words in the grammar, which are the *terminal* symbols. This grammar begins with the traditional notion in English that a sentence is divided into a subject and a predicate. The subject is a determiner phrase **DP** and the predicate is a verb phrase **VP**. The **DP** is rewritten as a determiner **D** followed by a noun phrase **NP**, which can be expanded as a bare noun **N**, yielding any of the noun phrases *the boy*, *the*

girl, *the hammer*, or *the doll*. The **NP** may also be expanded into a **N** followed by a prepositional phrase **PP**, which, in turn, can only be expanded as a preposition **P** (to be replaced by *with*) and a **DP**. Notice that the recursive definition of **NP** means that any number of prepositional phrases may follow a noun phrase: *boy with the girl with the doll with the hammer ..., ad nauseum*. The **VP** may designate an intransitive verb **V** having no object (such as *moved*), a transitive verb **V** followed by its direct object **DP** (*hit* or *moved*), or it can be recursively expanded into a **VP** and a **PP**. In both the recursive rule expansions of **NP** and **VP**, the **PP** is said to attach to the **N** (called *low-attachment*) or **VP** (*high-attachment*) to indicate which phrase the **PP** modifies. The two rules allow the representation of the two interpretations for the prepositional phrase attachment ambiguity in the sentence *the boy hit the girl with the doll*. If the interpretation is that the girl has the doll, then the sentence is rewritten as

> **[S [DP [D the] [NP [N boy]]] [VP [V hit] [DP [D the] [NP [N girl] [PP [P with] [DP [D the] [NP [N doll]]]]]]]]**.

Otherwise, if it is the boy who has the doll and uses it to hit the girl, then the sentence is rewritten as

> **[S [DP [D the] [NP [N boy]]] [VP [VP [V hit] [DP [D the] [NP [N girl]]]] [PP [P with] [DP [D the] [NP [N doll]]]]]]**.

The CFG can be used to represent such ambiguity and many aspects of human language that display context-free behavior, such as embedded phrases and clauses. But it is difficult to capture in a CFG other aspects of language such as word order, long-distance dependencies, agreement, or lexical properties, such as the fact that the **VP** rule for an intransitive verb should only apply to the verb *moved* because *hit* requires an object in English.

A more sophisticated variant, the *unification grammar*, can represent these language aspects. Whereas in a CFG all nonterminals are atomic category labels, a unification grammar associates bundles of features called *feature structures* to nonterminals and lexical items that are used as constraints on how constituents can be composed. The features in a feature structure have values associated with them, some of which may themselves be feature structures.

The appeal of symbolic grammars is their transparency: they mean what they have been designed to represent. Linguistic notions can be carefully analyzed, codified, and compared against language exemplars. Generalizations and simplifications of a grammar can provide insight into the human language faculty. Such was the case with the structure of phrases: the order of the head word (the noun in a noun phrase, the verb in a verb phrase, the preposition in a prepositional phrase, etc) and its roles is the same for almost all phrases in any given language, providing a phrase template, so to speak, for that language. This observation led to the development of X-bar Theory (Chomsky 1970), which collapsed the traditional phrase markers of NP, VP, DP, PP, and AP (as well as some less traditional ones like IP and CP that carry inflection and complement information, respectively) into a generic XP category. It also inspired the use of rule schemas, such

$\mathbf{S}_{1.0}$	$\rightarrow$ **DP VP**	$\mathbf{D}_{1.0}$	$\rightarrow$ ***the***
$\mathbf{DP}_{1.0}$	$\rightarrow$ **D NP**	$\mathbf{N}_{0.3}$	$\rightarrow$ ***boy***
$\mathbf{NP}_{0.6}$	$\rightarrow$ **N**	$\mathbf{N}_{0.3}$	$\rightarrow$ ***girl***
$\mathbf{NP}_{0.4}$	$\rightarrow$ **N PP**	$\mathbf{N}_{0.2}$	$\rightarrow$ ***hammer***
$\mathbf{VP}_{0.3}$	$\rightarrow$ **V**	$\mathbf{N}_{0.2}$	$\rightarrow$ ***doll***
$\mathbf{VP}_{0.5}$	$\rightarrow$ **V DP**	$\mathbf{V}_{0.4}$	$\rightarrow$ ***hit***
$\mathbf{VP}_{0.2}$	$\rightarrow$ **VP PP**	$\mathbf{V}_{0.6}$	$\rightarrow$ ***moved***
$\mathbf{PP}_{1.0}$	$\rightarrow$ **P DP**	$\mathbf{P}_{1.0}$	$\rightarrow$ ***with***

Figure 2.2: **Statistical NLP.** The same grammar as in Figure 2.1, but with probabilities assigned to the rules yields a PCGF. Notice that the probabilities on all of the rules with the same category (e.g., **VP**) sum to 1.0, which means that these are the only permissible expansions of the rules in this grammar. The probabilities give the likelihood of a given expansion. For example, in this grammar, low-attachment of the prepositional phrase is more likely than high-attachment because the rule for the **NP** $\rightarrow$ **N PP** expansion has a higher probability (0.4) than the probability of 0.2 associated with the verb phrase expansion **VP** $\rightarrow$ **VP PP**. The probabilities are usually compiled from a statistical analysis of a corpus. In this way, the PCFG can be given an empirical foundation to better represent real human language usage.

as the head-complement and head-modifier rules of Head-driven Phrase Structure Grammar (Pollard and Sag 1994), widening the linguistic divide between transformational and lexicalist approaches. With each new such insight, a grammar can be refined and improved.

Yet, the very explicitness of a symbolic grammar is its greatest weakness. As clearly as the rule-like behavior of language stands out, so do the exceptions. Grammars must constantly be revised to broaden their coverage to handle extragrammatical effects such as dysfluencies, fillers and pauses, dialectical variations, and scores of other phenomena of language usage that humans take for granted. Moreover, languages change, as new usages gain currency. Language variation means that new rules must constantly be added or refined, and that often conflicting rules must coexist. Grammar development is dauntingly time-consuming and ultimately unfinishable.

2.1.2 Statistical

Although statistical analysis and description of linguistic data was very much the standard before Chomsky's inauguration of generative syntax in the late 1950's, it was not until annotated corpora became widespread that statistical NLP came into its own. Broadly speaking, statistical NLP spans a variety of disciplines, from speech processing to information retrieval, and uses a variety of techniques. They include both supervised techniques, where target data is given, and unsupervised techniques, in which the system must induce meaningful categories. But in assigning structure and meaning to sentences, the standard approach has been to supplement rules in symbolic grammars, either hand-crafted beforehand or read off from an annotated corpus, with probabilities derived from a statistical analysis of the corpus (Charniak 2000).

Figure 2.2 shows one of the most common statistical approaches that uses such probabilities. The grammar is the same context-free grammar in Figure 2.1, but with probabilities assigned to

the production rules. This association of probabilities with the rules in a context-free grammar defines a *Probabilistic Context-Free Grammar*, or PCFG. The grammar in this example illustrates the common prepositional phrase attachment ambiguity that has been the subject of much research both within the parsing community and psycholinguistics. As has often been noted in empirical studies of English, the prepositional phrase is more likely to attach to a preceding noun phrase than a verb phrase. This empirical data is represented in the PCFG by associating a higher probability with the noun phrase rule **NP** $\rightarrow$ **N PP** (0.4) than with the verb phrase expansion **VP** $\rightarrow$ **VP PP** (0.2). However, even with probabilities, such a simple grammar is unable to account for the selectional preferences of the lexical items in the grammar. For example, this grammar would treat *hammer* as a modifier rather than an instrument in the sentence *the boy hit the girl with the hammer*, whereas a person would most likely make the opposite association. As a step toward overcoming such shortcomings, statistical analyses have been incorporated into unification grammars and lexicalized PCFGs.

Rule probabilities allow a statistical parser to rank all the parses generated from a grammar for a given sentence according to an evaluation metric. The advantages over purely symbolic models are manifold: the use of statistics captures correlations in the data that would otherwise have to be reified in the grammar, most often at an immense cost in the complexity of the grammar. Moreover, the statistics are learned, which can both offset the grammar developers' (necessarily) incomplete knowledge of language, as well as make the grammar more robust to peculiarities of the corpus. The grammar underlying the system is still transparent, as in a symbolic system. However, the statistical model may not be so enlightening, since its mathematical basis can often be quite opaque. Yet, with a grammar prespecified as in a symbolic system, the statistical system is also limited by the rules of the grammar. Moreover, because first-order statistics are not sensitive to sentence structure, the trend in statistical NLP has been to incorporate more and more features on which to condition the probabilities. For example, Collins (1999) evaluates several parsers that progressively include more context information, while all context is considered in the Data-Oriented Parsing approach of Bod (2001). Although quite powerful, the statistical approach essentially recasts the symbolic requirement for complete domain knowledge in a new statistical light. A grammar must still be specified and the most informative features for the statistical model have to be identified. What is needed is an approach that can develop useful features on its own from the data. Such an approach is described next.

2.1.3 Subsymbolic

A third approach to natural language processing that has been widely used in modeling psycholinguistic phenomena in the past two decades takes advantage of the characteristics of subsymbolic, or connectionist, systems. These systems are inspired by the biological model of the brain as a network of massively interconnected simple processing units, the neurons, which allow parallel computation with soft constraints. Such *artificial neural networks* develop representations that automatically

Figure 2.3: **Subsymbolic NLP.** (***Color figure***) A defining characteristic of subsymbolic systems is that they have no explicit grammar. Subsymbolic approaches typically use distributed representations to extract and represent a grammar from the training data. Accordingly, there is no clear breakout of the traditional phrasal categories. Rather, a statistical method of the distributed representations is required to characterize the grammar the network has learned. Yet, inevitably, the components will often only loosely correspond to linguistic notions such as phrasal category, but more often be an amalgamation of several such categories. The best characterization, then, for the network's "grammar" is that it is a blend of rules with inter-rule interpolations possible, as illustrated in the figure. It is difficult to pull out of the distributed representation using a standard grammar since doing so forces the network to conform with the grammar's discrete rules.

distribute information over many units so that all units more or less contribute to the representation and damage to any given unit only makes the representation slightly less accurate. These distributed representations automatically give rise to a variety of interesting cognitive phenomena. For example, neural networks have been used to model how syntactic, semantic, and thematic constraints are seamlessly integrated to interpret linguistic data, lexical errors resulting from memory interference and overloading, aphasic and dyslexic impairments resulting from physical damage, biases, defaults and expectations that emerge from training history, as well as robust and graceful degradation with noisy and incomplete or conflicting input (Allen and Seidenberg 1999; McClelland and Kawamoto 1986; Miikkulainen 1997a, 1993; Plaut and Shallice 1993; St. John and McClelland 1990).

Figure 2.3 emphasizes the differences between the subsymbolic and symbolic/statistical paradigms. Whereas the latter two approaches generally assume an underlying grammar to generate parse trees to be either manipulated or ranked, the subsymbolic approach has no *explicit* grammar. Instead, the grammar must be induced through training, and often becomes implicit in the activation of the neural network. Accessing the grammar requires statistical methods such as principal component analysis to map the most salient features in the network into features familiar to linguists. As often as not, the features in the network are an amalgam of linguistic features and, therefore, do not necessarily match the linguistic notions that are built into the symbolic and statistical grammars.

Yet, despite their many attractive characteristics, neural networks have proven very difficult to scale up to parsing realistic language. Training takes a long time, fixed-size vectors make learning long-distance dependencies and composition of structures difficult, and the format of the training data can impose architectural constraints, such as binary parse trees that are very deep and force more information to be compressed in higher nodes, thereby making the sentence constituents harder to recover. Progress has been made by introducing a number of shortcuts such as concentrating on small artificial corpora with straightforward linguistic characteristics (Berg 1992; Ho and Chan 2001; Sharkey and Sharkey 1992), building in crucial linguistic heuristics such as Minimal Attachment and Right Association (Lane and Henderson 2001; Mayberry and Miikkulainen 1999), or foregoing parse structures altogether in order to concentrate on more tractable subproblems such as clause identification (Hammerton 2001) and grammaticality judgments (Lawrence et al. 2000;

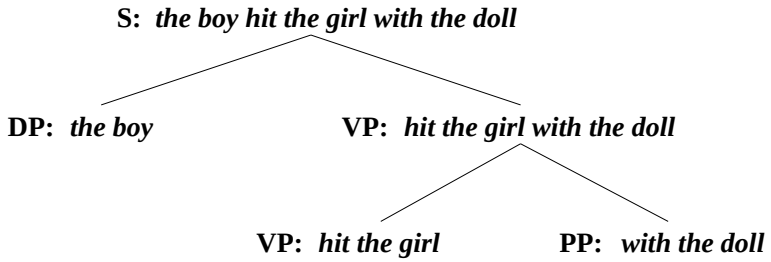


Figure 2.4: **Linguistic Structure.** Parsing in the context of traditional NLP is a process in which a sentence is converted into a linguistic structure which represents the relationships among the words in the sentence. The grammar that governs the parsing process describe these relationships. Here, for example, the sentence *the boy hit the girl with the doll* is divisible into two constituents, *the boy* and *hit the girl with the doll*, according to the rules of English grammar. A natural language processing system must have a mechanism which can retain information in memory which can be used later in a sentence to satisfy syntactic and semantic constraints.

Allen and Seidenberg 1999; Christiansen and Chater 1999).

2.2 Cognitive Issues in Natural Language Understanding

Because they are generated according to constraints, distributed representations integrate information very concisely. However, these distributed representations only capture correlations from training data. A fundamental assumption in linguistics is that mental representations are structured compositionally rather than simply an amalgamation of statistical correlations. Much of linguistics is concerned with describing the processes that build up these structured representations during the course of language comprehension. Such processes have been difficult to model in connectionist systems because neural network architectures have limited memory capacity (Stolcke 1990; Miikkulainen 1996). Accordingly, their application has been typically confined to toy grammars and demonstrations based on predicting the next word in a sentence or filling in static thematic roles, as will be described in Section 2.3.3.

This problem can be illustrated with the simple example in Figure 2.4. A sentence such as *the boy hit the girl with the doll* is transformed, or *parsed*, into a *phrase structure* representing the relationships among its constituents. The sentence is divided into a subject, the **DP** *the boy*, and a predicate, the **VP** *hit the girl with the doll* in this example. The parser must be able to detect such phrasal boundaries. The problem is particularly challenging because language is inherently recursive. Any phrase, in principle, can contain any number of words, and the words can have dependencies, such as agreement, on others that are arbitrarily far apart in the sentence. The subject could as well be *the nattily dressed boy who liked the girl who stood at the window ...*, with more and more words open to modification. Such variable and extensible structures have been a major challenge to the neural network approach. The networks do not have explicit grammar rules, but

must retain all of the information needed to build structures in memory. However, memory is limited in a fixed length representation, so previous information gradually degrades as new information is processed. Moreover, the structures themselves lose information as they become more complicated because they, too, are described by fixed-length representations.

The pervasive *ambiguity* of natural language complicates parsing considerably. Ambiguity arises at all levels of linguistic analysis, from lexical to structural to semantic to pragmatic. Furthermore, the boundaries between these levels are not distinct. For example, the ubiquitous prepositional phrase attachment ambiguity, as Groucho Marx so cleverly abuses in his famous quip

> I shot an elephant in my pajamas. How he got in my pajamas, I'll never know.

is one that cannot be reliably resolved on the basis of syntax alone. In the worst case, world knowledge is the only guide to the intended meaning.

As just described, in the symbolic and statistical NLP communities, which tend to be based on syntactic grammars, parsing is often taken as the task of converting a sentence into a unique, linguistically-preferred, syntactic tree. Because grammars tend to license a multitude of possible parse trees for a given input sentence, parsing involves the additional burden of consistently ranking the possible parses produced by a grammar so that the preferred parse tree always comes out on top. Such parse selection is usually called *parse disambiguation*, but the ambiguity stems from the grammar, and not necessarily natural language (although the more accurate the grammar, the more overlap there will be). Yet, as noted in Collins (1999), parsing can also be understood as the task of finding all syntactically well-formed parse trees for a sentence, leaving aside the burden of disambiguation and ranking. This latter notion of parsing has generally been adopted in the psycholinguistics community where the nature of ambiguity itself and how humans so effortlessly accommodate it is of theoretical interest. Not surprisingly, connectionist systems, as models of human performance, usually adopt the approach of finding all parses for a given sentence, in which each part of the parse is graded according to how likely it is to be included in the sentence interpretation. Accordingly, this approach to parsing is used in this thesis. Furthermore, although parsing has been traditionally regarded as a syntactic process, we adopt the more general notion of parsing as the task of converting a sentence into a linguistically-motivated structured representation, not necessarily exclusively syntactic. In this dissertation, the focus will be primarily on *semantic parsing*, in which the linguistic representation denotes meaning.

The cognitively plausible approach to parsing ambiguous sentences would mimic the process employed by humans during the course of sentence comprehension. The best evidence so far suggests that language understanding is fundamentally an incremental process. By pruning away unlikely interpretations as a sentence is processed, *incrementality* provides a tractable way of dealing with the combinatorial explosion of analyses licensed by a grammar for ambiguous constituents. Approaches to parsing that require components of a constituent to be assembled before the constituent itself can be constructed, such as chart and shift-reduce parsing, are particularly vulnerable to the issue of combinatorial complexity. The reason is that the components must be retained in

memory for indefinite lengths of time until enough constituents have been constructed to allow disambiguation (Church and Patil 1982; Barton et al. 1987). However, psycholinguistic evidence that multiple interpretations of a sentence are initially coactivated and the unviable interpretations later suppressed provides further support for the incremental approach (Hudson and Tanenhaus 1984). Moreover, not all ambiguities are always resolved, and indeed some ambiguities, such as modification by a prepositional phrase, may remain underspecified (i.e., the prepositional phrase cannot be said conclusively to modify any particular constituent, but may rather modify several at the same time; Hindle and Rooth 1993).

Numerous psycholinguistic studies based on analyses of eye-tracking data strongly suggest that people not only process sentences incrementally, but actively anticipate thematic roles well before they encounter the words that would fill those roles (Tanenhaus et al. 1995). Connectionist networks are particularly adept at developing such defaults and expectations based on their training (Allen and Seidenberg 1999; McClelland and Kawamoto 1986; Miikkulainen 1997a).

There is one more issue that motivates the connectionist approach to sentence processing: *nonmonotonicity*, or active revision of an interpretation in light of later context. Nonmonotonicity can be seen in many guises, from the reinterpretation of lexical meaning to the suppression of coactivated interpretations (Onifer and Swinney 1981; MacDonald et al. 1992; MacDonald 1993), and, in the extreme case, in garden-pathing (Small et al. 1988; Marslen-Wilson 1989).

To remain a cognitive model, a connectionist sentence processing system should account for semantic parsing, ambiguity, incrementality, and nonmonotonicity. We will next review the components on which most connectionist sentence processing systems have been built to model psycholinguistic studies.

2.3 Foundations of Connectionist Sentence Processing Systems

In the past two decades a number of neural network architectures have been proposed to represent linguistic structure. In this section, we will take a brief inventory of some of the most prominent of these architectures. Beginning with an overview of representation in neural networks, we will review two architectures that use supervised training, the Simple Recurrent Network (SRN; Elman 1990) and Recursive Auto-Associative Memory (RAAM; Pollack 1990), to model recurrency in neural network approaches to language processing. We will also introduce an unsupervised algorithm, the Self-Organizing Map (SOM; Kohonen 1990, 1995), a version of which is used in the current research as a means of organizing semantic representations in order to help the network learn an efficient way to use its memory resources. These basic architectures have come to serve as a foundation for most of the connectionist approaches to sentence processing that have since followed, including the approach presented in this dissertation.

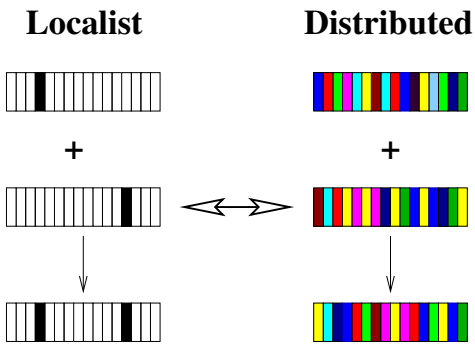


Figure 2.5: **Localist vs Distributed Representations.** (***Color figure***) In a localist representation (left), each unit stands for one word. In a distributed representation (right), all the units contribute to the representation of one word. Because localist representations are discrete (each unit may take on a set number of values, usually binary), several different words can be represented simultaneously in one representation, but doing so in a distributed representation requires that each word be represented less accurately. However, the number of words that can be stuffed into a localist representation is limited by the number of units, whereas the capacity of a distributed representation is far higher, limited only by the resolution of the unit activations. Localist and distributed representations are the endpoints of a continuum of possible representations (shown by the double arrow). Intermediate types of representation, such as coarse-coding, combine features of both by limiting how information is distributed among the units.

2.3.1 Local and Distributed Representations

An important issue in any natural language processing system is how the words and structures built from them are to be encoded. One approach is straightforward: each unit in the representation stands for a distinct word (see Figure 2.5). Such *localist* representations are useful in many contexts, but leave open the question of how to encode structure. They are also limited by the length of the representation, since the representation must at least accommodate all the words to be represented. In distributed, or subsymbolic, representations, items are represented across all units as a pattern of activation. In this manner, similar items develop similar representations. Moreover, the number of items that can be represented is limited not so much by the number of units in the representation as by the resolution of the unit activations. That is, if the units are continuous-valued, they can theoretically represent an infinite number of items. Continuous-valued units in distributed representations allow interpolation between patterns, which results in good generalization and robustness to noise.

Thus, there is a basic trade-off between localist and distributed representations: accuracy for capacity. Localist representations are exact; the precision of each unit in the representation serves as a hard, inviolable constraint which limits the number of items that can be represented. If, for example, two items require the same unit for their representation, they cannot both be represented. On the other hand, distributed representations are approximate; the activation of each unit imposes only a soft constraint on the complete representation of a given item. That is, the activation does not have to be exact. Small deviations in the unit activations tend to cancel each other out without necessarily

losing the identity of the item represented, allowing these units to contribute to the representation of other items. In short, it is the combination of *all* the unit activations of a representation that determines what it stands for, and each may contribute more or less to one representation and more or less to other representations.

This limit in representational capacity also affects generalization. Generalization results from interpolation between unit activations. The units in distributed representations can take on continuous values, allowing for interpolation. In contrast, the units in localist representations are discrete because they can only take on a set number of values (such as 0 and 1 in the case of binary units). Generalization is limited because the set of values imposes hard constraints on interpolated activations.

However, it is useful to keep in mind that localist and distributed representations are only the end points of a spectrum of possible representations. The more values that each unit in a localist representation can represent and the more units that contribute to the representation of a single item, the more distributed the representation becomes. Conversely, the more the value of any one unit in a distributed representation is dedicated to the representation of an item, the more localist it behaves.

A common intermediate representation is *feature encoding*, which still has mostly localist properties, but is more distributed in the sense that more than one unit contributes to the representation of a word. As a consequence, many more words can be represented in a feature-encoded representation than a strictly localist one. Feature encodings are useful for representing lexical items because they allow for explicit word recognition. Yet, they permit more lexical information such as syntactic category to be encoded in the representation that can be used by a neural network to learn a given task.

When a neural network is permitted to develop its own distributed representations, such as in the network's hidden layer (described next in Section 2.3.2) or through a mechanism like FGREP (Forming Global Representations with Extended BackPropagation; Miikkulainen 1993), the representations tend to develop much finer distinctions called *microfeatures*. These microfeatures are the activations over sets of units that work together to encode the types of features that we attribute to symbols. However, different sets of units will come to represent features that a grammar designer would normally assign to one specific set of units.

Along this spectrum of representation, it proves to be much more difficult to represent structure in localist representations. Simply activating many units in parallel to represent the constituents in the structure reveals nothing about how the words relate to one another, only that they are correlated. Such relations must themselves be somehow reified in the units as well. The resulting combinatoric explosion of possible structures to be represented also makes representing them explicitly infeasible. However, distributed representations can encode structure as well as terminal symbols (words). Encoding structure in a distributed representation involves compressing the representations for two or more constituents into a single representation via an *encoder* in such a way that those constituents can be reconstructed through a *decoder*. The sections on the SRN and par-

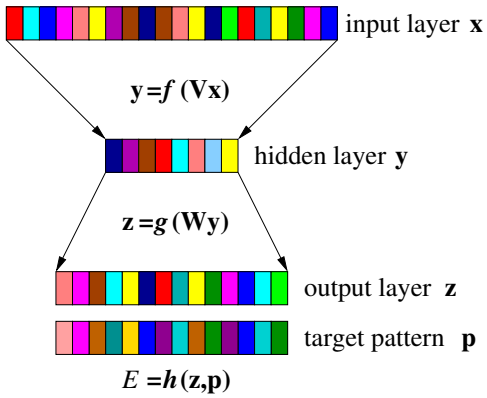


Figure 2.6: **Two-layer FeedForward/Backpropagation Network.** (***Color figure***) A pattern of activation is presented to the input layer as a vector **x** and propagated through the weight matrix **V** between the input layer and hidden layer. The resulting vector **Vx** is passed through a squashing function f so that the hidden layer activation pattern $\mathbf{y} = f(\mathbf{Vx})$ remains bounded. Similarly, the output layer activation pattern **z** is the vector $g(\mathbf{Wy})$. Typically, f and g are the same function. The output pattern **z** is compared with a target pattern **p**, and a scalar error $E = h(\mathbf{z},\mathbf{p})$ assessed according to a cost function h. The weight matrices **V** and **W** are then modified based on E.

ticularly the RAAM network (see Sections 2.3.3 and 2.3.4 below) will deal with this issue in more detail. First, however, we will describe the general process by which distributed representations are developed.

2.3.2 Backpropagation Networks

Figure 2.6 shows a basic two-layer backpropagation network, which we will use to first give an overview of the network's operation, and then to describe the derivation of the backpropagation algorithm. The network has two weight matrices, **V**, connecting the input layer to the hidden layer, and **W**, connecting the hidden layer to the output layer. The network is commonly called a two-layer network because the units in the input layer are not computed, but simply presented as an initial pattern of activation **x**, whereas activation of the hidden layer and output layer are computed in the following way. The hidden layer activation pattern **y** is calculated as a function f of the vector **Vx** in order to keep the elements of **y** within a predefined range. For this reason, f is often called a *squashing function*. The output layer pattern of activation **z** is determined in the same manner as the result of $g(\mathbf{Wy})$ for a squashing function g, which is generally the same as f. Once the output pattern has been calculated, it is compared with a *target* pattern **p** and an error $E = h(\mathbf{z},\mathbf{p})$ is assessed. The function h is called a *cost function* and provides a metric of the similarity of the output **z** to the target **p**. This error is used to adapt the weights **V** and **W** so that **z** gradually converges on **p**. The weights are changed slowly because **p** is just one of a set of patterns P to which patterns from an input space X are to be mapped, in effect approximating a function

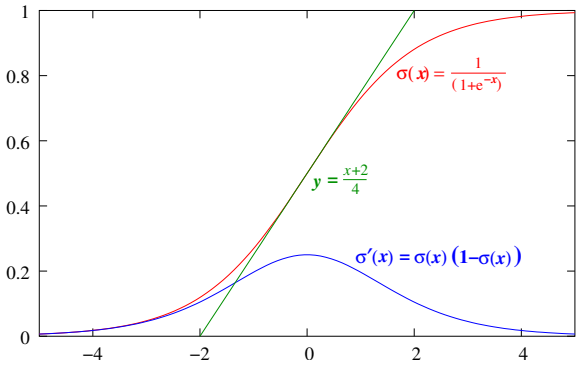


Figure 2.7: **Logistic Function.** (***Color figure***) The logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is a standard sigmoid function used in backpropagation networks to map the real line to the (0,1) interval. In its midrange, it is very nearly collinear with $y = (x+2)/4$. In addition to this linear approximation, the logistic function has another mathematical property that makes it computationally useful: its derivate $\sigma'(x)$ can be succinctly expressed in terms of $\sigma(x)$ itself as $\sigma(x)(1-\sigma(x))$.

$M : X \rightarrow P$. Given enough non-linear units in the hidden layer, it has been proven that such an approximation can be made arbitrarily close, making the basic two-layer backpropagation network a *universal function approximator* (Hornik et al. 1990). Backpropagation is an optimization algorithm used to find weights that give the lowest overall error mapping X to P. We will first show how the weights are adapted, and then describe how the backpropagation algorithm can be generalized to multi-layer networks.

Training a backpropagation network consists of two phases. The forward phase propagates the input pattern through the network to the output layer, and the backward phase redistributes the error E between the output activation and the target pattern over the weights by minimizing this error.

Forward propagation begins with the input pattern of activation **x** presented to the network and propagated through the weight matrix **V**. The resulting vector **Vx** is passed through a sigmoid to produce an activation pattern **y** in the hidden layer. The standard sigmoid that is used in backpropagation networks is the logistic function (Figure 2.7) because of its nice mathematical properties. The logistic function is semilinear, which means it is differentiable and monotonically increasing, and it bounds the real numbers to the interval (0,1), and is very nearly linear in its midrange. Furthermore, its derivative can be re-expressed in terms of the logistic function itself:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} \tag{2.2}$$

$$\sigma'(x) = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} \tag{2.3}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \tag{2.4}$$

These properties make the function computationally efficient.

We will denote the elements of **x** by x_i, those of **y** by y_j, and those of **z** by z_k. The weights in **V** and **W** are then v_{ji} and w_{kj}, respectively, using standard matrix notation. The activation of each unit in the hidden layer is calculated as

$$r_j = \sum_i v_{ji} x_i \tag{2.5}$$

$$y_j = \sigma(r_j) = \frac{1}{1 + e^{-r_j}} \tag{2.6}$$

where r_j is the net input to a hidden layer unit. Typically, a bias unit is added to the input layer and set to 1 in order to allow the sigmoid to be displaced from the origin by adjusting the bias weight. Equation 2.5 remains unchanged, but with i ranging over each unit in the input layer including the bias unit.

The resulting hidden layer **y** activation is then propagated through the weight matrix **W** to produce an activation pattern in the output layer **z**, which is calculated for each unit as

$$s_k = \sum_j w_{kj} y_j \tag{2.7}$$

$$z_k = \sigma(s_j) = \frac{1}{1 + e^{-s_j}} \tag{2.8}$$

with s_k denoting the net input to an output layer unit. As before, a bias unit may be added to the hidden layer and set to 1 with no change in Equation 2.7 except to accommodate the extra unit in the range of j.

Once the output pattern is produced, the backward propagation (or *backpropagation*) phase is started. The output pattern **z** is compared to a target pattern **p**, and an error signal E is determined according to a cost function. A standard cost function is the *sum of squared errors*:

$$E = \frac{1}{2} \sum_k (p_k - z_k)^2 \tag{2.9}$$

Because the cost and logistic functions are differentiable, this error can be reformulated as a differentiable function of the weights, producing an *error curve* amenable to numerical analysis. The goal, then, becomes a search for the set of weights that minimize this function. Accordingly, the backpropagation algorithm can be derived as *gradient descent*, in which a minimum for the function is found by greedily following the curve in the direction of its *steepest* (most negative) slope. The derivatives $\frac{\partial E}{\partial w_{kj}}$ and $\frac{\partial E}{\partial v_{ji}}$ denote how E changes as the connection weights w_{kj} and v_{ji} change, respectively. The chain rule on $\frac{\partial E}{\partial w_{kj}}$ gives

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial s_k}{\partial w_{kj}} \frac{\partial z_k}{\partial s_k} \frac{\partial E}{\partial z_k}$$

where each term in the product is determined by differentiating the equations 2.8, 2.7, and 2.9 with respect to w_{kj}, s_k, and z_k, respectively:

$$\frac{\partial s_k}{\partial w_{kj}} = \frac{\partial (\sum_j w_{kj} y_j)}{\partial w_{kj}} = y_j$$

$$\begin{aligned}\frac{\partial z_k}{\partial s_k} &= \sigma'(s_k) &= z_k(1-z_k)\\ \frac{\partial E}{\partial z_k} &= \frac{\partial(\frac{1}{2}\sum_k (p_k - z_k)^2)}{\partial z_k} &= -(p_k - z_k)\end{aligned}$$

The result

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial s_k}{\partial w_{jk}}\frac{\partial z_k}{\partial s_k}\frac{\partial E}{\partial z_k} = -y_j z_k(1-z_k)(p_k - z_k)$$

represents the slope of the error curve evaluated at w_{kj}.

The error signal δ_k defines the contribution of unit z_k to the total cost E:

$$\delta_k = \sigma'(s_k)\frac{\partial E}{\partial z_k} = z_k(1-z_k)(p_k - z_k)$$

and the weight adjustment for w_{kj} allots the error δ_k according to the strengh of y_j:

$$\Delta w_{kj} = \eta \delta_k y_j \tag{2.10}$$

where η is the learning rate, or step size, to be taken.

Once the δ_k error signals are found, the chain rule can then be used to determine $\frac{\partial E}{\partial v_{ji}}$:

$$\begin{aligned}\frac{\partial E}{\partial v_{ji}} &= \frac{\partial r_j}{\partial v_{ji}}\frac{\partial y_j}{\partial r_j}\sum_k \frac{\partial s_k}{\partial y_j}\frac{\partial z_k}{\partial s_k}\frac{\partial E}{\partial z_k}\\ &= \frac{\partial r_j}{\partial v_{ji}}\frac{\partial y_j}{\partial r_j}\sum_k \frac{\partial s_k}{\partial y_j}\delta_k\end{aligned}$$

where the three partial derivatives, $\frac{\partial r_j}{\partial w_{ji}}$, $\frac{\partial y_j}{\partial r_j}$, and $\frac{\partial s_k}{\partial y_j}$, are found by differentiating the equations 2.7, 2.5, and 2.6 with respect to w_{ji}, r_j, and y_j, respectively:

$$\begin{aligned}\frac{\partial r_j}{\partial w_{ji}} &= \frac{\partial(\sum_i w_{ji}x_i)}{\partial w_{ji}} &= x_i\\ \frac{\partial y_j}{\partial r_j} &= \sigma'(r_j) &= y_j(1-y_j)\\ \frac{\partial s_k}{\partial y_j} &= \frac{\partial(\sum_j w_{kj}y_j)}{\partial y_j} &= w_{kj}\end{aligned}$$

The result is

$$\frac{\partial E}{\partial v_{ji}} = \frac{\partial r_j}{\partial v_{ji}}\frac{\partial y_j}{\partial r_j}\sum_k \frac{\partial s_k}{\partial y_j}\delta_k \tag{2.11}$$

$$= -x_i\sigma'(r_j)\sum_k w_{kj}\delta_k \tag{2.12}$$

$$= -x_i\delta_j \tag{2.13}$$

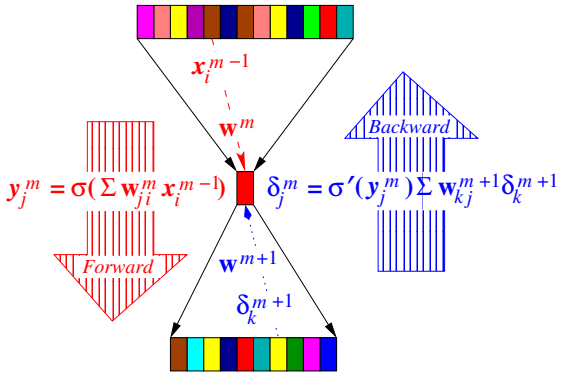


Figure 2.8: **General Forward/Backpropagation.** (***Color figure***) The activation and error for each unit in a layer are calculated independently of all other units in the layer. In the forward phase, the activation of a unit y_j^m in layer m is the weighted sum of the vector $\mathbf{x}^{m-1}$ passed through the sigmoid function σ. In the backward phase, the error signal δ_j^m is the weighted sum of the error signals δ_k^{m+1} from layer $m+1$ proportioned by $\sigma'(y_j^m)$, which allocates most of the error from y_j^m to the sigmoid's midrange, as shown in Figure 2.7.

where the contribution of unit y_j to E is defined as

$$\delta_j = \sigma'(r_j) \sum_k w_{kj}\delta_k = y_j(1-y_j) \sum_k w_{kj}\delta_k \tag{2.14}$$

and the weight change Δv_{ji} then given by

$$\Delta v_{ji} = \eta \delta_j x_i \tag{2.15}$$

in a manner analogous to Equation 2.10.

Generalizing Equation 2.13 to networks with more than two layers is straightforward because the derivative $\frac{\partial E}{\partial v_{ji}}$ only depends on the activation of x_i propagated to connection v_{ji} and the error signal δ_j backpropagated from the previous layer. The general equations for a matrix of weights w_{ji}^m between layer $m-1$ and layer m, with error signal δ_j^m proportional to the weighted sum of the error signals δ_k^{m+1} from the previous layer $m+1$ are given by:

$$\frac{\partial E}{\partial w_{ji}^m} = -x_i^{m-1}\delta_j^m \tag{2.16}$$

$$\delta_j^m = \sigma'(y_j^m) \textstyle\sum_k w_{kj}^{m+1}\delta_k^{m+1} \tag{2.17}$$

$$\Delta w_{ji}^m = \eta \delta_j^m x_i^{m-1} \tag{2.18}$$

where i indexes units x in layer $m-1$, j indexes units y in layer m, and k indexes units z in layer $m+1$, as shown in Figure 2.8.

Backpropagation has a direct interpretation as a blame assessment algorithm. The error signal at the output layer is distributed backward over the weights of the network in the opposite

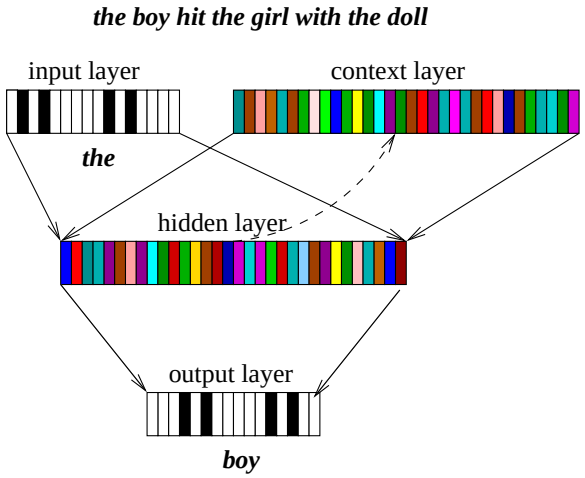


Figure 2.9: **Next word prediction with SRN.** (***Color figure***) The input sentence from Figure 2.4, *the boy hit the girl with the doll*, is shown being presented to the network one word at a time. The coarse-coded representation for the current input word *the* is shown at top left. At each step, the previous activation of the hidden layer is copied (indicated by the dotted arrow) to the context layer assembly to serve as context for the current input. This activation, together with the current input word is propagated to the hidden layer of the SRN network through a matrix of weights (indicated by solid arrows) that connect all the units in one layer to another. The hidden layer is further propagated to the output layer through another set of weights. In most applications of this architecture, the network generates the representation of the next word in the sentence (here, *boy*) as its output. All connections (weights) are trained through backpropagation (Rumelhart et al. 1986).

way that the activation is propagated forward from the input to the output layer. Those weights that contribute most to the error for a given input are adjusted proportionately.

2.3.3 Simple Recurrent Networks

Since its introduction in 1990, the Simple Recurrent Network (SRN; Elman 1990) has become a mainstay in connectionist natural language processing tasks such as lexical disambiguation, prepositional phrase attachment, active-passive transformation, anaphora resolution, and translation (Allen 1987; Chalmers 1990; Munro et al. 1991; Touretzky 1991).

The SRN is a backpropagation network with an extra assembly of units (the previous hidden layer or context layer) that serves as temporal context for the input (Figure 2.9). A sequence of inputs is presented to the network one word at a time. The network propagates the activations from the current input and the context layer forward through the network to the output layer. As described in Section 2.3.2, an error signal is then determined and backpropagated to change the weights between the layers in order to better approximate the output. The hidden layer is then copied to the context layer assembly to serve as context for the input word at the next time step.

Figure 2.9 shows an example SRN at the beginning of processing the sentence *the boy hit*

the girl with the doll. The representation for the first word *the* was presented to the input layer of the SRN. The context layer at this point was blank because there was no previous context for the first word of the sentence. These layers (input and context) were propagated together to the hidden layer, which formed a representation to serve as memory for the next word in the sentence, *boy*. The hidden layer representation was propagated to the output layer, for which the next word (*boy*) served as a target. The hidden layer was copied to the context layer assembly, the representation for the next word was loaded into the input layer, and the process was repeated in this manner until an end-of-sentence marker was read.

As each word in the sentence is read in, the memory represented by the hidden layer degrades. The reason is that the hidden layer is fixed in length, and is forced to represent previous items less accurately in order to accommodate new ones. This *memory problem* has limited the use of the SRN to learning fairly simple targets such as localist or feature-encoded word representations.

The aim of the network is to match the target as closely as possible. In this manner, the network is typically trained, as in the task above, to "predict" the next word in the sentence from all the words which have occurred previously. Many researchers have demonstrated the behavior of the architecture on a next-word prediction task in order to evaluate what types of linguistic phenomena were learnable (Elman 1991; Elman et al. 1996; Weckerly and Elman 1992; Christiansen and Devlin 1997). They have shown, for example, that the SRN is able to identify clausal boundaries in sentences with relative clauses. The network was able to maintain long-distance dependency information across relative clauses in the hidden layer such as agreement between a subject and verb. However, the memory capacity of the SRN was limited, and it was unable to encode dependencies across intervening context-independent phrases.

Although prediction has been a useful method of showing how a neural network can acquire grammatical knowledge from the training data, it is not a task in itself. A more cognitively and linguistically founded approach would attempt to model the building of structures that represent meaning. That is what people do when they understand language, and that is what artificial systems should also do to be useful. A step in this direction has been made with SRNs that map a sentence into a set of case roles, as in Figure 2.10. Based on the theory of thematic case roles (Fillmore 1968), case-role analysis assumes that the syntactic structure of the sentence is specified beforehand, and the goal is to assign the proper roles to the words in the sentence. For example, given a simple sentence with subject, verb, object, and a with-clause, the network's task is to assign those words to the thematic roles **agent**, **act**, **patient**, and **modifier** or **instrument** depending on the selectional preferences of the words in the training corpus (as determined by frequency; Miikkulainen and Dyer 1991; McClelland and Kawamoto 1986). As before, the sentence is read in one word at a time, and the network is trained to map the sentence into the words that fill the case roles for that sentence. By giving these same words as static targets for each word of the sentence, the network will develop expectations and defaults, as shown in Figure 2.10. The network has only read the word *girl*, but is already expecting *hammer* as an **instrument**. But when the word *doll* is later read in, the activation

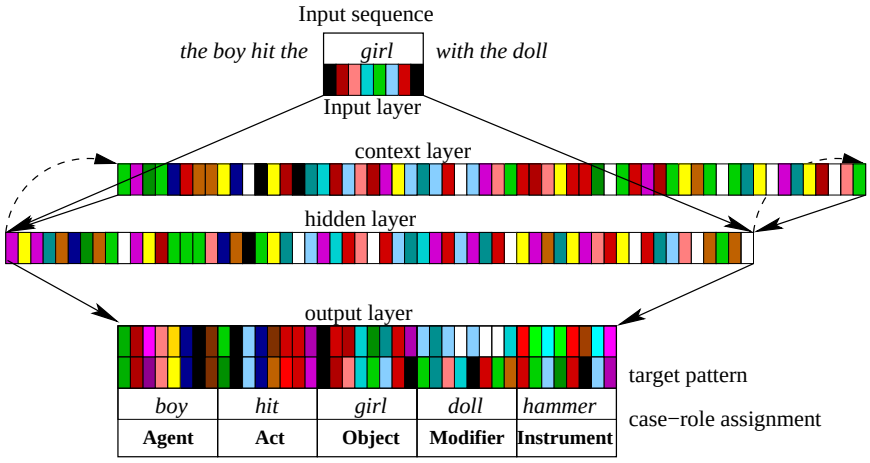


Figure 2.10: **Case-role analysis with SRN.** (***Color figure***) The same input sentence *the boy hit the girl with the doll* as in Figure 2.9 is presented to the case-role analysis network. The distributed representation for the current input word, *girl*, is shown at top center. As before, propagation of the input proceeds through a set of weights to the hidden layer, with the previous hidden layer saved as the current context layer. The hidden layer activation is further propagated to the five output layer assemblies that have been chosen to stand for selected semantic roles. The network demonstrates expectations by anticipating the word *hammer* as an instrument for *girl*, based on its training experience. However, the modifier *doll* is also weakly activated, and will become strongly activated when that word is later read in as input, with the activation of the instrumental output becoming consequently weak.

for *hammer* falls off, and the activation for *doll* will become strongly activated. If, instead of *doll*, which only occurs in the modifier sense, the word were *ball*, then both the **instrument** and as **modifier** would be coactivated. The **modifier** sense, however, would be more strongly activated because that is the sense that is most frequent in the training set. This shows that the network also demonstrates nonmonotonicity in the coactivation of multiple senses in the face of ambiguity and its subsequent resolution. This behavior is a very desirable property for a cognitively motivated system to have, and has served as a constraint on the implementation of INSOMNet.

While this approach works well for sentences with fixed structure, it becomes untenable for the full complexity of language. Stopgap solutions, such as fixing the number of any given role, only work on toy grammars and small corpora, and are both linguistically and cognitively distasteful. The building of structures that represent meaning should not impose hard constraints on the depth of the structure. To be cognitively valid, the ability to build up more complex structures should be subject to soft constraints that result in human-like errors in processing. In Section 2.4, architectures that incorporate structure encoding are briefly described. They make use of another standard connectionist NLP architecture, called RAAM, which is described next.

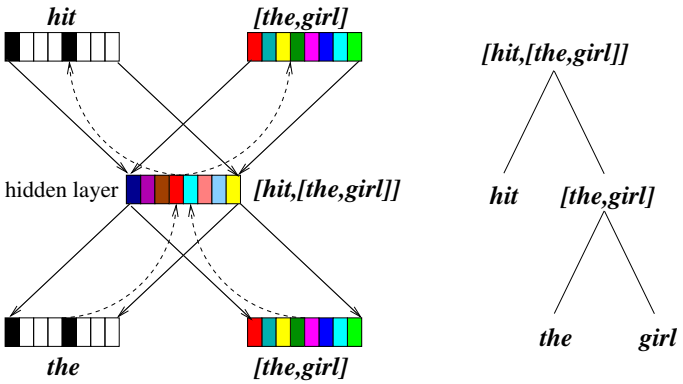


Figure 2.11: **Recursive Auto-Associative Memory.** (***Color figure***) In order to encode a tree structure such as that on the right in a fixed-length distributed representation, the tree's constituents need to be compressed. In the binary RAAM shown here, two eight-unit input representations, *hit* and *[the,girl]* are combined into a single compressed eight-unit representation, *[hit,[the,girl]]*. This compressed representation develops in the hidden layer through auto-associating the input as output. The subtree *[the,girl]* is itself a compressed representation of its constituents, *the* and *girl*, saved from the hidden layer using the same network. In this manner, progressively deeper structures can be encoded from simpler constituents. All the weights (indicated by arrows) are trained through backpropagation. The dotted lines indicate that the compressed hidden layer representation can be recursively used as input and target when building the tree structure, and the output can be further decoded from the hidden layer.

2.3.4 Recursive Auto-Associative Memory

A way of combining two or more substructures into one is essential to representing structures in a neural network. The Recursive Auto-Associative Memory (RAAM; Pollack 1990) provides such a mechanism. RAAM is a three-layer backpropagation network in which two input representations, or assemblies, are compressed into a single representation that can be decoded to recover the original inputs. The network is trained to reproduce, or *auto-associate*, its input at its output; at the same time a compressed representation of its input/output develops in the hidden layer. Because the compressed representation has the same length as that of its constituents, it can be used for building up yet deeper structures in a recursive fashion (Figure 2.11).

For example, in Figure 2.11, the tree *[hit,[the,girl]]* is encoded from *hit* and *[the,girl]*. The subtree *[the,girl]* has already been encoded from *the* and *girl*. The leaf representations for *hit*, *the*, and *girl* must be specified beforehand. Usually they are given a simple localist or feature encoding.

Like the SRN, RAAM suffers from a memory problem. As deeper structures are encoded, information from earlier constituents such as the leaves themselves is gradually degraded to the point where it is eventually unrecoverable. The reason here is that the constituents are compressed in the hidden layer. The hidden layer is fixed-length and, consequently, the more items it is forced to represent, the less accurately it is able to represent them.

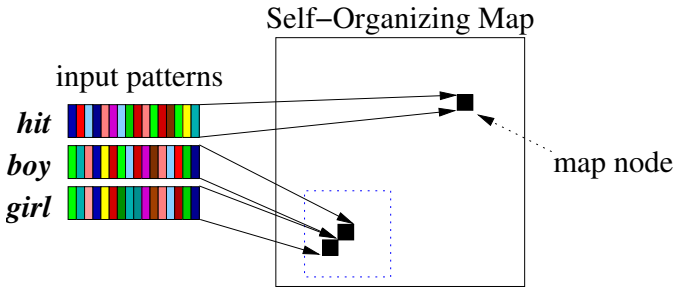


Figure 2.12: **Self-Organizing Map.** (***Color figure***) The figure shows the common two-dimensional map together with three example input patterns *hit*, *boy*, and *girl*, the latter two of which are similar. There is a single set of weights between the input layer, to which the input patterns are presented, and the map nodes on the map. Trained on a set of such patterns, the Self-Organizing Map develops a two-dimensional clustering of its input space while preserving much of its topology. Thus, inputs that have similar patterns in the input space will often end up near each other on the map, such as *boy* and *girl* in the figure, while dissimilar patterns, such as *hit*, will tend to be mapped to units farther away. The reason that input patterns get clustered in this manner on the map is that the self-organization algorithm develops a local response by activating the map node whose weights are closest to an input pattern. These weights and the weights of surrounding map nodes are then adapted towards the input's activation pattern. Over time, nearby map nodes become similar and thus respond to similar input patterns.

2.3.5 Self-Organizing Map

The Self-Organizing Map (SOM; Kohonen 1990, 1995), is a neural network that develops a two-dimensional clustering of its input space of patterns while preserving the topology of that space by mapping similar input patterns to nearby units (map nodes) on the map. In Figure 2.12 each input pattern is shown being mapped onto a particular map node called the maximally-responding unit, or *winner*, as measured by a metric such as Euclidean distance. Thus, for an input pattern $\mathbf{x}$, the response n_{ij} of each map node (i, j) (where i and j are indices on a two-dimensional map) is calculated as

$$n_{ij} = 1.0 - \frac{||x - m_{ij}||}{d_{\max}} \tag{2.19}$$

where $\mathbf{m}_{ij}$ is the weight vector for unit (i, j) and $d_{\max}$ is the maximum distance of two vectors in the input space (e.g. $\sqrt{2}$ in the 2-D unit square). The winner is the map node which satisfies

$$i, j = \text{argmax}\ n_{ij} \tag{2.20}$$

The training algorithm proceeds by randomly initializing the weights between each node in the map and the input layer and setting the *neighborhood* (the dashed square in Figure 2.12) to at least half the diameter of the map. The set of input patterns are presented randomly to the map, with each such presentation of the full set taken as an *epoch*. As each pattern is presented, the map is searched for the map node whose weights most closely match the input pattern using Equation 2.20. When identified, the winner's weights and the weights of all units within its neighborhood

are adjusted toward the input pattern according to

$$\Delta\mu_{i,j} = \alpha(t)[x - \mu_{ij}](i,j) \in N_c(t) \tag{2.21}$$

After some number of epochs, the neighborhood is decreased, and the learning rate lowered. In this way, the node weights slowly converge to represent the input space.

The weights of the winner and all the nodes in its *neighborhood* are updated according to the standard SOM adaptation rule (Kohonen 1990, 1995) to better approximate the current input. The size of the neighborhood is set at the beginning of the training and reduced as the map becomes more organized. The *learning rate*, which is the amount of weight update on each trial, is also set beforehand and reduced according to a predefined schedule that often is linearly or exponentially decaying.

The Self-Organizing Map has been used in a number of NLP tasks, such as data mining (Kohonen et al. 2000; Honkela et al. 1998), speech recognition (Kangas et al. 1992), and parsing (Mayberry and Miikkulainen 1999) because of its ability to induce salient features of its input space. This ability is utilized in INSOMNet on a version of the SOM in order to develop semantic representations from a corpus.

2.4 Previous Connectionist Parsers

Most parsers in connectionist NLP have used the SRN and RAAM as foundations for sentence processing. Other parsers have been symbolic-subsymbolic hybrid systems such as PARSEC (Jain 1991) and NNEP (Henderson 1994). These latter systems have used symbolic components to get around the problems of neural networks such as the memory problem associated with structure encoding, but suffer many of the drawbacks associated with symbolic systems, including catastrophic failure and the need for designing the system. Nevertheless, because of its prominence, we will begin with a brief description of PARSEC, and round up the review with a look at two recent models that have been applied to large-scale language tasks: the Simple Synchrony Network (SSN; Lane and Henderson 2001), a model based on NNEP, and the Connectionist Sentence Comprehension and Production model (CSCP; Rohde 2002). The neural network approaches, of course, all share the advantages and disadvantages of subsymbolic systems, the chief among these being the memory problem and the associated difficulty in building up sizable structures.

2.4.1 PARSEC

PARSEC (Jain 1991) is an incremental connectionist parser of spoken language that learns to assign input words into a three level structure for phrases, clauses, and the full sentence through a series of feedforward "mapping" modules. The parser was trained on the *Conference Registration Task*, which consisted of 204 sentences taken from a 400-word lexicon. It proved to be robust to

speech effects such as restarts and repairs, as well as ungrammaticality and fillers, when tested on the Air Travel Information Service (ATIS) task. However, there is very little structural information given in the output representations, other than those implied by the phrases and clauses themselves. How these constituents relate to each other is left unspecified. Regardless, PARSEC clearly demonstrated the some of the most attractive properties of neural networks such as robustness to noisy and ungrammatical input.

2.4.2 Reilly's RAAM-based Parser

Reilly demonstrated one of the earliest and henceforth most common approaches to subsymbolic parsing of syntactic structure (Reilly 1992). A RAAM network was first trained to encode a set of 16 syntactic parse trees into distributed representations. An SRN was then trained to read the representation of the sentence one word at a time, outputing the representation of the complete parse tree at every step. For example, as each item in the input sequence `d n p d n v d a n` was read in successively, the SRN attempts to produce the parse result `[[[d,n],[p,d,n]],[v,[d,[a,n]]]]`. The SRN was then tested on four novel sentences, and the final parse results were decoded to determine how well the system learned the constituent structures.

Reilly's architecture was able to learn 11 of the 16 training sentences. In the four test sentences, it was able to produce some substructures correctly, but failed to generate the correct parse for the complete sentence. Although Reilly did not directly state that the SRN is unable to retain enough information in memory to produce the correct parse at the output, this can be inferred from the results he gave in the paper. The SRN failed to learn those trees with the deepest structure in both the training and test sets. Reilly did, however, fault the limited capacity of both the RAAM and SRN networks for the poor generalization to novel sentences.

2.4.3 Sharkey and Sharkey's Modular Parser

Sharkey and Sharkey took Reilly's approach a step further by incorporating a feedforward network to map the output from an SRN to RAAM parse tree representations (Sharkey and Sharkey 1992). Three different connectionist architectures, an SRN, a feedforward network, and a RAAM, were all trained separately and then combined into a four-layer architecture that could parse structurally ambiguous sentences in terms of prepositional phrase attachment, embedded clauses and relative clauses.

The RAAM decoder network was trained to develop the structures that will serve as output of the integrated network. The SRN encoder network is trained on the standard prediction task of the next work in the input. The feedforward network was trained to map the hidden layer representation at the end of the process to the corresponding RAAM representation so that the structure could be decoded into its proper constituents. The network was able to disambiguate all of the 1280 sentences it was trained on and 75.6% of the 320 novel sentences in the test set. Cluster analysis of the hidden

unit activations revealed that the SRN was capturing significant structural information.

Sharkey and Sharkey's model was one of the first purely modular approaches to connectionist sentence processing, but generalization was still very limited. Sharkey and Sharkey did not provide an analysis of the actual errors that the network made, concentrating their discussion rather on the structures that the network was able to capture, but the poor generalization results implicate the limited memory capacity of the SRN parser and RAAM decoder.

2.4.4 Berg's Xeric Parser

In the Xeric parser (Berg 1992), RAAM and SRN were integrated into a five-layer architecture. The encoder part of a RAAM network was inserted between the input and context layers and the hidden layer of the SRN, and the decoder part was inserted between the hidden layer and the output layer. The input is a feature encoding of the current word, which is propagated through the encoder to the hidden layer, and further propagated through the decoder to the output. The output corresponds to a template structure based on a simplified form of X-Bar theory (Sells 1985), composed of a specifier, head word, and up to two complements. Features include the syntactic category of the word, its number and person, and several parameters that depend on the category (such as tense for verbs). Additionally, a 9-unit ID tag identifies words having identical syntactic features.

The network was trained on a corpus of 1000 fairly simple, unambiguous sentences using a variant of backpropagation through time (Rumelhart et al. 1986). Training was not perfect, with percentage correct ranging between $91 - 99\%$ depending on the depth of the X-Bar structures used to represent the sentences. On the testing corpora (also consisting of 1000 sentences), the percentage correct ranged between $89 - 99\%$. Berg's analysis of the error showed that 53% of the errors the network made were due to the ID tags in the output units. However, the ID units constituted only a tenth of the total number of output units. Thus, a disproportionate number of the errors the Xeric parser made was trying to identify constituents. This is the hallmark of the memory problem that has prevented the application of recurrent neural networks to interesting linguistic phenomena.

Nonetheless, Xeric was one of the first integrated architectures that could handle and represent complex recursive phrase structure constructs. An integrated architecture is desirable because it permits the encoding and building of structure to happen simultaneously.

2.4.5 Ho and Chan's Confluent Preorder Parser

The confluent preorder parser (Ho and Chan 1997) also learned to process sentences one word at a time, but the output representations were generated by a preorder traversal of the sentences' parse trees by a variation of RAAM for sequences called SRAAM (Pollack 1990). The sequence generated by a preorder traversal of the parse tree of a sentence yields more elements than are in the sentence itself. The extra elements are the internal nodes of the parse tree. The sentence is just the terminals. To deal with the extra elements, Ho and Chan use a dual-ported SRAAM (Chrisman

1991) to develop representations for the preorder traversal sequences. In the dual-ported SRAAM, a process called confluent inference forces the hidden layer to develop representations that subserve two different tasks. In Ho and Chan's model, these tasks are developing a representation of the surface sentence and, at the same time, the preorder traversal of the sentence's parse tree.

The network was trained on 82 sentences, and tested on 30 sentences from a syntactic grammar adapted from a grammar introduced by Pollack in his paper on RAAM (Pollack 1990) and was able to generalize to 93.75% of the test set. This is a reasonable amount of generalization, but the corpus of sentences was relatively small and uncomplicated, and had no semantic constraints that could make the task more difficult.

A drawback of this approach, as acknowledged in the paper, is that the network does not develop representations that correspond to the internal structure of a sentence. For example, internal syntactic phrase structures within a sentence, such as `p d a n`, would still need to be trained separately in order for the parser to develop a representation (e.g. `[p,[d,[a,n]]]`) for those structures due to the preorder traversal strategy used. The parser only learns to parse complete sentences. The reason this is undesirable is that linguistics research suggests that such phrases are indeed processed as meaningful constituents in themselves and then integrated into the meaning of the sentence that is being developed during parsing.

2.4.6 Miikkulainen's SPEC

The Subsymbolic Parser for Embedded Clauses (SPEC; Miikkulainen 1996) was a system based on the SRN (the Parser) specifically designed to handle relative clauses. In order to accomplish this, two other components, the Stack and the Segmenter, were included in the system. The parser read in words one at a time and formed a case-role representation at its output. The Stack was a RAAM network, and had the task of saving and restoring the context of the Parser as the relative clause segments were encountered during parsing. The Segmenter, a simple feedforward network, used the Parser's hidden layer and the next input word to learn clause boundaries. It used the boundary information to control the execution of the Parser so that it would use the right context (from the Stack) depending on the level of the relative clause being parsed. Each module was trained separately with basic clause constructs so that, when the system was integrated, it could generalize very well to novel sentence structures. Trained on only 100 randomly selected sentences from a grammar adapted from Elman's that featured semantic restrictions (Elman 1991), the network successfully parsed the entire corpus of 98,100 sentences.

When the representation on the stack was artificially lesioned by adding noise, SPEC exhibited very plausible cognitive performance. Shallow center embeddings were easier to process, as were sentences with strong semantic constraints in the role bindings. When the parser made errors, it usually switched the roles of two words in the sentence, which is what people also do in similar situations. A symbolic representation of the stack would make modeling such behavior very difficult.

SPEC demonstrated remarkable generalization ability, but much of this was due to its specialized design for relative clauses. Moreover, the output for the parser was simply case-role representations which were only as deep as the level of embedding of relative clauses. However, the investigation into the cognitive performance of the network did reveal the memory limitations of the Stack and Parser.

2.4.7 SSN

The Simple Synchrony Network (SSN; Lane and Henderson 2001) is a SRN-based system that uses Temporal Synchrony Variable Binding (TSVB; Shastri and Ajjanagadde 1993) to represent structures and generalize across sentence constituents. The introduction of TSVB was motivated by the *binding problem*, which arises when multiple entities are each represented with multiple features. TSVB indicates which features are bound to which entities through the use of synchrony of activation pulses. The SRN component of the system handles the non-pulsing units in the system which represent information about the sentence as a whole, while the pulsing units contain information about constituents. This approach was taken to deal with the $O(n^2)$ potential parent-child dependencies between constituents. The use of incremental processing allows the SSN to consider only how to incorporate the incoming word into the syntactic structure so far processed.

The network operates on the part-of-speech (POS) tags for the words in the sentence. As the SSN incrementally processes the sentence, it outputs the POS tag for the parent, as well as the intermediate parent-child dependencies holding between that parent and previous sentence constituents. In this way, all parent-child dependencies between parent and children will accumulate in the output, so that the full syntactic structure of the sentence is represented in the output at the end of the sentence.

Applied to a subset of the SUZANNE corpus, the most successful version of the architecture achieved between 70% and 80% average precision/recall, which is comparable to the performance of parsers using Probabilistic Context-Free Grammars, described in Section 2.1.2.

2.4.8 CSCP

The Connectionist Sentence Comprehension and Production model (CSCP; Rohde 2002) is also a system based on the SRN that has been designed to both comprehend a sentence incrementally and reproduce the sentence from its compressed encoding in the 500-unit hidden message layer. The system is composed of two modules, the message encoder/decoder system and the comprehension, prediction, and production system, which are trained separately. The encoder/decoder system first encodes a set of "Propositions" of the form **(action, role, role-filler)**. For example, the triple (chased, agent, dog) tells us that in the sentence *the dog chased the car*, the dog is doing the chasing. A set of such triples has been shown to be equivalent to a semantic network (Hinton 1981). Message decoding is accomplished through a query network (St. John and

McClelland 1988, 1990), which combines the message layer with a proposition query layer, in which one of the three parts is missing (hence, all three parts have to be probed; moreover, this process is repeated for each new proposition presented to the network, resulting in $3n(n+1)/2$ queries and quadratic training time in the number of propositions). Error is calculated at the proposition response layer and backpropagated to the proposition query layer.

The other module of the CSCP contains the comprehension and production systems. Three-syllable encodings of words (some words have been forced into three syllables through abbreviation) are presented to the comprehension system at the word input layer. Word input proceeds through a hidden layer to the comprehension gestalt layer and then to the message layer, both of which are recurrent. The production system goes in the reverse order from the message layer to a recurrent production gestalt layer through a hidden layer and finally out to a prediction layer. The prediction layer, as the name implies, predicts the next word in the sentence.

The CSCP is fundamentally a model of human sentence processing designed to model empirical experiments on a wide range of psycholinguistic phenomena including nested relative clauses and main verb/reduced relative, sentential complement, subordinate clause, and prepositional phrase attachment ambiguities. It is also designed to model production errors and structural priming effects.

The model was trained on about four million sentences generated according to the "Penglish" grammar, a hand-crafted PCFG with limits in the depth of recursion. The probabilities for the rules in Penglish were statistically determined from the Penn Treebank.

The CSCP model represents the state-of-the-art in connectionist modeling of psycholinguistic data. It is not a parsing system, but can only fill in queries presented to it, as the "Comprehension" in its title attests. Moreover, it is trained on sentences and propositions generated from a well-designed and consistent grammar of English that has been tailored to the phenomena under study. As such, it lacks a number of common word types that show up in standard corpora such as modals, predicate adjectives, comparatives or superlatives, gerunds, infinitives, and proper nouns. Questions and commands have been left out, as have subjunctive mood, perfect progressive, and compound future tenses. Nevertheless, it is truly a system worth aspiring to.

2.4.9 Discussion

Two of the approaches taken up here were based on predicting the next word in the sentence (Reilly and Sharkey and Sharkey). Prediction is easier for the SRN to handle because it does not have to retain as much information in memory as it would if it were being trained to output structured representations. In Berg's Xeric parser, the network did learn structured linguistic representations, but the phrases were fairly simple overall. It was trained like a RAAM network and, so, suffered the same memory problems that arise from compression, as discussed in Section 2.3.4.

Another difference between the models described reveals two common strategies in connectionist modeling. All of the systems are *modular*, with the sole exception of Berg's Xeric model, which is an *integrated* system. The modular approach allows the modules of the system to be trained

independently of one another, which can greatly simplify the task undertaken. Yet, an integrated system makes no assumptions about how the system should be broken up into modules and, furthermore, often results in better training because there are no interfaces through which there is often the cost of lost accuracy in error information. Moreover, from a cognitive standpoint, an integrated system more closely resembles the observed functioning of the language faculty. The integrated system develops its own internalized clustering of the training space, and so, in a sense, generates its own modules with soft boundaries. This approach has been taken with INSOMNet in order to keep the possibility open of using the model as a true language acquisition system.

The large-scale parsers are much more powerful systems. The SSN is a *syntactic* parser that performs well on the task it was evaluated on, but there is a strict limit on the number of phases that can keep the pulsing units distinct. The CSCP is an impressive demonstration of a connectionist system that is able to make the subtle distinctions that are of central interest in psycholinguistics. Yet it can only respond to questions asked of it, and did not model extragrammatical phenomena that pervade language use, although speech errors in production were indeed modeled. However, being a purely connectionist system, there is little doubt that this is simply a research avenue yet to be explored.

On the question of parsing, there still remains the issue of how the structures will be built up from the input sentence; that is, what is the best way to parse the sentence. Previous approaches have either attempted to map the input sentence to its parse tree directly (Reilly, Sharkey and Sharkey, and Berg), or map the sentence incrementally to a preorder traversal of the sentence's parse tree (Ho and Chan). Neither approach reveals how the same constituent (such as a phrase or clause) can appear in multiple places in the sentence. For example, the representation for the phrase *the boy* in the sentence *the girl who saw the boy liked the dog* is embedded in the representation for *the girl who saw the boy*. A more realistic linguistic approach would identify and treat this phrase separately and then integrate it into the sentence representation during parsing. Miikkulainen comes closest to this approach, but the parser was designed to specifically handle relative clauses, relying on a controller to symbolically "push" and "pop" the appropriate context representations from the neural network Stack in order to generate case-role representations at the output. The INSOMNet model performs no such explicit stack manipulations, but rather relies on the self-organization of semantic features in order to determine the proper storage of intermediate parse results. The linguistic basis for the semantic encoding used in the model is described next.

2.5 Linguistic Foundation

Until very recently, almost all work on applying machine learning techniques has focused on part-of-speech tagging and syntax because of the availability of large-scale syntactic corpora on which meaningful analysis could be accomplished in these areas. In this respect, the Penn Treebank (Marcus et al. 1993) has become the *de facto* standard for such analyses. Yet, as Mooney (1999) points

out, this focus on tractable problems has led to a "scaling up by dumbing down" trend in NLP research. The real AI goal of producing semantic interpretations for sentences has remained elusive. There are two primary, but related, reasons for this relative lack of progress. The first is the scarcity of large-scale corpora with rich semantic annotation. The second is the difficulty of semantic annotation itself, which has contributed to its scarcity. Recently, a medium-sized corpus of deep semantic annotations called the Redwoods Treebank has become available. Before describing Redwoods, we will provide some background on the grammar formalism, *Head-driven Phrase Structure Grammar*, on which most of the annotations are based, as well as an overview of the project, *Linguistic Grammars Online*, in which the Treebank has assumed a central role.

2.5.1 Head-driven Phrase Structure Grammar

Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag 1994) is a constraint-based lexicalist (unification) grammar formalism that differs from the more traditional derivational approaches to linguistics by emphasizing the role of the lexicon in understanding linguistic phenomena. Lexical information such as phonology, morphology, syntax, and semantics are regarded as word constraints that interact to produce meaning for the sentence constituents which the words comprise. These lexical components are bundled together in a typed feature structure called a *sign*, the attributes of which are determined from the sign's place in the HPSG type hierarchy. The type hierarchy defines all possible signs in the grammar and is designed to capture linguistic properties of a language. Because of its parsimony and predictive power, HPSG has become a central theory in linguistics research, with an ever-growing body of literature and resources.

Much of this research is being conducted with tools and corpora developed under the Linguistic Grammars Online (LinGO) project at the Center for the Study of Language and Information (CSLI) at Stanford University.

2.5.2 Linguistic Grammars Online

The LinGO includes a variety of tools for grammar and corpus development and analysis. The Linguistic Knowledge Builder (LKB) system is the primary grammar and lexicon development environment for use with constraint-based linguistic formalisms. It includes the English Resource Grammar (ERG; Flickinger 2000), a large-scale, broad-coverage grammar of English based on the HPSG framework. Corpus development is facilitated by the [Incr TSDB()] Tool (Oepen 2001), which is a database profiling system that handles the storage, retrieval, and analysis of parse results from the LKB system.

2.5.3 Redwoods Treebank

The LinGO Redwoods Treebank (Oepen et al. 2002) is a recent project that provides much more detailed syntactic and semantic analyses than standard corpora used in computational linguistics,

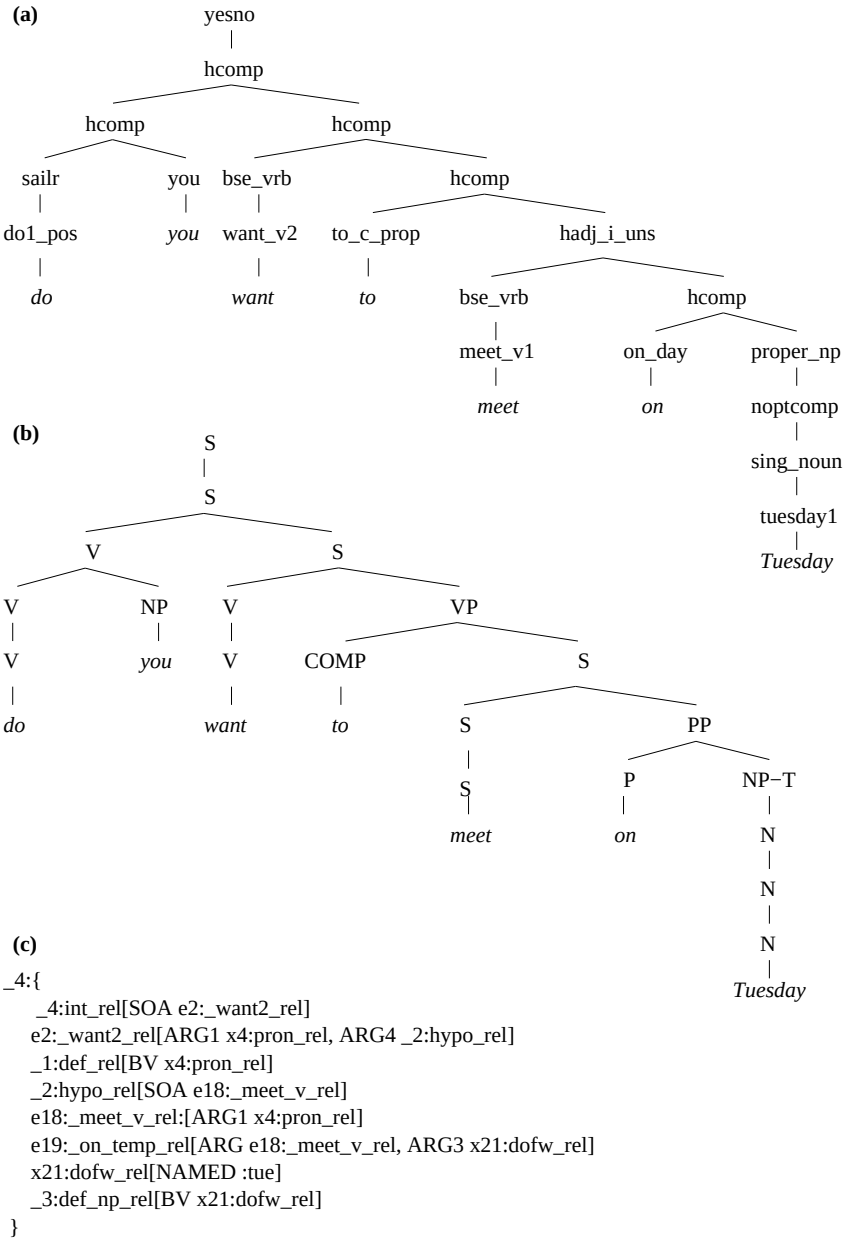


Figure 2.13: **Sentence Representations Extractable from the Redwoods Treebank.** As illustrated, three types of representations can be extracted from the Redwoods Treebank for the sentence *do you want to meet on Tuesday?*. The topmost (a) is a derivation tree using HPSG rule and lexical tags from the ERG, the center figure (b) is a derived phrase structure tree labeled with traditional POS tags like those in the Penn Treebank, and the bottommost (c) is an elementary dependency graph extracted from the full MRS representation of the sentence. *Source: (Toutanova et al. 2002).* Used with kind permission of the author.

such as the Penn Treebank (Marcus et al. 1993). The sentences in the Redwoods Treebank are taken from the recently completed VerbMobil project (Wahlster 2000), which features transcribed face-to-face dialogues in an appointment and travel arrangement domain. The sentences have been annotated (mostly by one linguist up until recently) with detailed HPSG analyses licensed by the ERG. Figure 2.13 shows the three types of sentence analyses that can be extracted from the Redwoods Treebank using the [Incr TSDB()] Tool. The topmost (a) representation, the HPSG derivation tree, is the primary format, from which the complete HPSG feature structure for the sentence can be retrieved. The center tree (b) is a traditional phrase structure format for compatibility with Penn Treebank-style research. The bottom format (c) is a semantic structure that encodes basic predicate-argument relationships in a sentence. It is derived from the full Minimal Recursion Semantics representation of the sentence (described in the next section), which can also be extracted from the Redwoods Treebank with the [Incr TSDB()] Tool.

The finer detail of the sentence annotations provided by Redwoods does raise new concerns with respect to machine learning approaches. The depth of the linguistic information should facilitate analysis of such traditionally difficult NLP issues as prepositional phrase attachment by providing more lexical information. Yet, that finer detail also exacerbates the issue of data sparsity which has been a constant concern in NLP.

2.5.4 Minimal Recursion Semantics

Syntax and semantics are integrated in the ERG, but the LKB has a module which extracts a powerful, theory-neutral semantic representation called Minimal Recursion Semantics (MRS; Copestake et al. 2001) from HPSG feature structures. MRS derives its representational power from the use of explicit pointers, called *handles*, to avoid the recursive substructures of conventional semantic representations, such as the Predicate Calculus. The resulting *flat semantics* structure of MRS has a significant advantage over recursive representations, both from computational and psycholinguistic standpoints closely related to the motivation for the incremental processing of sentences given in Section 2.2. In particular, the reference to substructures through handles rather than recursively embedding them directly in the semantic representation permits ambiguities to remain unresolved until disambiguating information is processed. This is important computationally because only one *underspecified* (i.e., more general) representation needs to be maintained in memory at any given time, and that representation can be refined as a sentence is processed. On the other hand, recursive representations lead to a combinatorial explosion as ambiguities multiply, all of which must be kept in memory until they can be pruned away by later context. The cognitive appeal of using underspecified representations is similar. As a sentence is processed incrementally, an early general semantic interpretation of the words may be possible due to defaults or expectation, and nonmonotonically revised as later words are processed, in keeping with the cognitive motivations outlined in Section 2.2.

A case in point is illustrated in Figure 2.14 comparing how the sentence *every man loves*

every man loves some woman

- Predicate Calculus
  1. every(x, man(x), some(y, woman(y), love(x, y)))
  2. some(y, woman(y), every(x, man(x), love(x, y)))
- Minimal Recursion Semantics
  - h1: every(x, h4, h6) (Predicate Calculus sense 1: h6 = h2)
  - h2: some(y, h5, h7) (Predicate Calculus sense 2: h7 = h1)
  - h3: love(x, y)
  - h4: man(x)
  - h5: woman(y)

Figure 2.14: **Minimal Recursion Semantics.** This figure demonstrates the advantages of MRS over Predicate Calculus with respect to scopal ambiguity on the sentence *every man loves some woman*. There is the default interpetation easily perceived by most that, for each man, there is a corresponding woman that he loves. This is the first sense given under the Predicate Calculus heading. Yet, another interpretation is possible, given appropriate context. Metaphorically, if the woman is Aphrodite, it could be claimed that every man loves her, as the symbol of love. This is the second interpretation. The difference lies in the scopes of the quantifiers *every* and *some*. A parser using Predicate Calculus as a semantic representation has to keep both interpretations in memory until sufficient context is given to allow resolution of the scopal ambiguity. MRS avoids specifying any interpretation by using *handle*s to label the predicates. The quantifiers are ternary predicates, requiring a *bound variable* (the x and y that go with *man* and *woman*, respectively) to indicate what is being quantified. They also have a *restriction* that refers to the predicates **man(**x**)** and **woman(**y**)** themselves. Lastly, they have a *scope* that specifies the portion of the universe that the quantifiers govern. Through the use of handle identification, either Predicate Calculus interpretation can be invoked: instantiating **h6** with **h2** yields the default first interpretation, whereas instantiating **h7** with **h1** gives the second. Leaving the handles **h6** and **h2** uninstantiated allows the ambiguity to be *underspecified*, i.e., unresolved.

some woman is represented in Predicate Calculus and in MRS. This sentence is a classic example of scopal ambiguity, and it is also interesting from a cognitive standpoint. The default interpretation, based on real-world knowledge, strongly favors the first reading under the heading "Predicate Calculus" in Figure 2.14. This formula states that there is a different woman for each man that he loves. The reason for this interpretation is that the predicate *some* falls within the scope of *every*. It often takes a second for people to see the second interpretation, given in the second formula, in which *some* has wide scope, governing *every*. Here, the meaning is that there is one *particular* woman that every man loves, a notion that usually only crops up in poetry. In Predicate Calculus, both interpretations have to be maintained until the ambiguity can be resolved.

MRS avoids committing to any interpretation by introducing a level of indirection through the use of handles to label predicates, such as the **h4** handle for **man(**x**)** in Figure 2.14. Some predicate arguments use handles to point to their fillers. An example is the quantifier **every(**x**,**

Abbreviation	Semantic Arguments
SA	*state-of-affairs* role
A0	*arg0* role
A1	*arg1* role
A3	*arg3* role
BV	*bound variable* role
RE	*restriction* role
IX	*instance* role
EV	*event* role
EVT	*event* feature structure
FRI	*full referential index* feature structure

Figure 2.15: **ERG Semantic Annotations.** The ERG links argument structure to a set of semantic roles. This table describes the semantic annotations, together with our abbreviations, used in the examples in this section. A complete list is given in Appendix A.

h4,h6), which is a ternary predicate with implicit arguments *bound variable*, *restriction*, and *scope*. The restriction is filled by the predicate labeled **h4** (i.e., *every* is restricted to the set of men), and the bound variable x ranges over that set. The handle **h6** in the scope slot is an example of underspecification because there is no predicate labeled **h6** among the predicates in the set; i.e., the scope is left unspecified. In order to specify the scope, the handle **h6** must be identified with another handle. Thus, in Figure 2.14, the default interpretation of the sentence is recovered when **h6** is instantiated with the handle **h2** that labels the predicate **some(**y**, h5, h7)**. Similarly, the poetic interpretation is invoked with the instantiation of **h7** in the scope slot of the *some* quantifier with the handle **h1** of *every*. Leaving the argument handles uninstantiated keeps the ambiguity unresolved, while instantiating both would allow for coactivation of the two interpretations.

The use of underspecification in MRS is not limited to scopal ambiguity. Figure 2.16 shows a graphical illustration of the MRS representation for the sentence *the boy hit the girl with the doll* using abbreviations we use for semantic annotations from the ERG. Table 2.15 gives a breakout for each abbreviation used in the examples in this section. Each node in the graph has a handle and a predicate (given by the corresponding word in the sentence if there is one). The arguments of the predicate are represented by the arcs coming from the predicate's node; the labels on the arcs are abbreviations for the argument names. Taken together, the arcs give the predicate's subcategorization, and the fillers for each argument are the handles of the nodes pointed to.

The sentence is declarative, as indicated by the semantic relation **prpstn_rel** in the top node labeled by the handle **h0**. Sentence types subcategorize for a *state-of-affairs*, which is indicated in Figure 2.16 by the arc labeled **SA**. The filler for this role is the handle **h1**, which is the label of the node for the main predication of the sentence, the verb *hit*. Verbs have an *event* role, and transitive verbs, such as *hit*, have an *arg1* role and an *arg3* role, which correspond to the thematic roles of **agent** and **patient**, respectively. These three roles are represented in the figure by the arcs

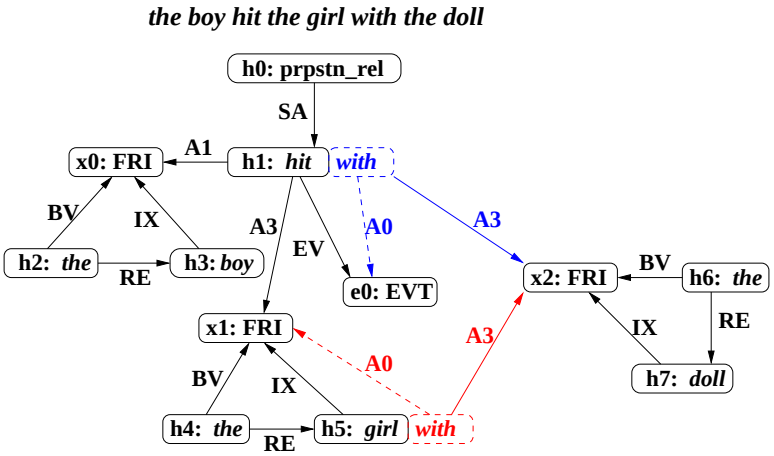


Figure 2.16: **Representation of Prepositional Phrase Attachment in MRS.** (***Color figure***) This graph represents the sentence *the boy hit the girl with the doll*. Nodes in the graph are labeled with a handle and the word in the sentence (or semantic relation if there is no corresponding word). A dashed node connected to another node represents attachment. In this case, both nodes have the same handle, which is how MRS represents modification. The arcs that come out of a node indicate subcategorization and are labeled with the arguments that the word or semantic relation takes. The handles that fill those roles are the labels of the nodes the arcs point to. The top node in the graph is labeled by the handle **h0** and semantic relation **prpstn_rel**, indicating the sentence is declarative. A **prpstn_rel** has a single argument, the **SA** arc, that points to the main predication of the sentence in the node labeled by the handle **h1** and word *hit*. The event index of *hit* is the handle **e0** of the node with semantic type **EVT**. Each noun phrase in the sentence is represented by a set of nodes for the determiner, the noun, and its index. For example, the semantics of the noun phrase *the boy* is given by the three nodes labeled with handles **h2**, **h3**, and **x0**. The determiner, *the*, has the handle **h2**, and its roles, **BV** and **RE**, are filled by **x0** and **h3**, respectively. The noun *boy* with handle **h3** has a single role, **IX**, filled by the handle of the index **x0**. The index has semantic type **FRI**. The agent of *hit*, indicated by the arc labeled **A1**, is also filled by the **x0** handle. Similarly, the patient, **A3**, is filled by the handle **x1**, which is the index for *girl*. Lastly, *doll* is the object of the preposition *with*, and so the handle of its index **x2** fills the preposition's **A3** role. The two interpretations of the sentence are illustrated by two *with* nodes. In the verb-attachment case, the *with* node has the same handle **h1** as *hit*, and its **A0** role is the index handle **e0**. In the noun-attachment case, the *with* node has the same handle **h5** as *girl*, and its **A0** role is the index handle **x1**.

labeled **EV**, **A1**, and **A3**. The filler for the verb's **EV** argument is the index **e0** for an *event structure* with semantic type **EVT** that specifies features for the verb such as tense, aspect, and mood. These features are not shown in Figure 2.16 to save space. The semantics of the noun phrases in the sentence are represented by three sets of nodes. Each set represents the determiner, the noun, and an index that indicates the features of the noun, such as gender, number, and person. The determiner is a quantifier which subcategorizes for the *bound variable* (**BV**) and *restriction* (**RE**) arguments (the *scope* role is empty in the current release of the Redwoods Treebank). The **BV** role is filled with the noun's index, and the **RE** role, with the noun's handle. The noun has a single *instance* (**IX**) role, filled with its index. The noun phrase, *the boy*, will make the representation clearer. In Figure 2.16,

the boy hit the girl with the doll

- h0: prpstn_rel(h1)
- h1: hit(x_0, x_1, e_0) (Instrumental interpretation: h8=h1 and h9=e_0)
- h2: the(x_0, h3, _)
- h3: boy(x_0)
- h4: the(x_1, h5, _)
- h5: girl(x_1) (Modifier interpretation: h8=h5 and h9=x_1)
- h6: the(x_2, h7, _)
- h7: doll(x_2)
- h8: with(h9,x_2)

Figure 2.17: **Structural Ambiguity in MRS.** As in the scopal ambiguity described in Figure 2.14, prepositional phrase attachment ambiguity can be represented in MRS through underspecification. In this case, both the preposition's handle **h8** and its first argument **h9** are instantiated to resolve the ambiguity. Identifying (or *sharing*) **h8** with the verb's handle **h1** and **h9** with the verb's event index e_0 gives the instrumental interpretation, whereas instantiating **h8** with the handle **h5** of **girl(**x_1**)** and **h9** with its index x_1 gives the modifier interpretation. The ambiguity remains unresolved if **h8** and **h9** are left uninstantiated.

the node for the noun *boy* is labeled with the handle **h2**, which fills the **RE** role for the governing determiner *the*. The index of *boy* is the handle **x0**, which labels the node with semantic type **FRI**, indicating that *boy* is a *full referential index*. The index handle **x0** binds the determiner and noun through their **BV** and **IX** roles, respectively, and fills the **A1** role of *hit* to indicate that *boy* is the agent of the verb. Similarly, the index handle **x1** fills the verb's **A3** role to denote the patient. The handle **x2** is the index for the noun *doll* and fills the **A3** role of the preposition *with*. The preposition can either modify the verb for the instrumental sense of the sentence, or the noun for the modifier sense. In MRS, modification is represented by *conjunction* of predicates; for example *big red doll* is denoted by $\bigwedge$[**big(**x**)**, **red(**x**)**,**doll(**x**)**]. The n-ary connective $\bigwedge$ is replaced by a handle, which is distributed across the operands so that each predicate has the same handle (an operation we call *handle-sharing*). In the case of verb-attachment, the verb *hit* and the preposition *with* both share the handle **h1**, and the preposition's **A0** role is filled with the verb's event structure handle **e0**. For noun-attachment, *with* has the same handle **h5** as *girl*, and its **A0** role points to the index **x1** of *girl*. Figure 2.17 shows how these predicates are similar to the MRS structure described earlier in Figure 2.14.

For neural networks, flat semantics with its associated advantages is a very useful form of representation. It helps to circumvent the severe memory limitations of the standard approach to

representing nested structure, such as parse trees, through RAAM as described in Section 2.3.4, while still allowing for the inherent cognitive behavior of neural networks.

2.6 Conclusions

As successful as connectionist systems have been to modeling human language performance, they have been exceedingly difficult to scale up to the competence-based tasks that have been the standard fare in the symbolic and statistical community. Chief among these is the parsing of a sentence into a structure which corresponds to the linguist's intuition about how the constituents of the sentence relate to each other. Creative solutions have been employed to get around the memory limitations associated with structure encoding, but to date, none have matched the performance of statistical systems. Yet, the effort to apply neural networks to the tasks commonly pursued in the symbolic/statistical communities, such as parse disambiguation, often entails sacrificing the very properties that make connectionist models attractive to researchers interested in cognition, such as their inherent robustness and generalization capabilities, as well as their ability to exhibit rule-like behavior without having a system of explicit rules to rely on. This passage between the Scylla of computational utility and the Charybdis of cognitive plausibility has dictated a number of design decisions in the course of developing INSOMNet. The model, its components, and their motivation will be described next.

Chapter 3

The INSOMNet Model

INSOMNet (Incremental Nonmonotonic Self-Organization of Meaning Network) is a three-layer neural network parser. The first layer of INSOMNet combines a standard Simple Recurrent Network (SRN; Elman 1990) with a map of input words (SARDNet; Mayberry and Miikkulainen 1999) to process a sentence incrementally and generate patterns in a second layer that encode the semantic interpretation of the sentence. The second layer is a two-dimensional map of patterns that are decoded in a third layer to yield the interpretation of the sentence. In this chapter, we will describe the architecture of INSOMNet and its components in detail. We will begin with motivation for the organization of INSOMNet's second layer, as the model's main technical contribution, and go on to show how the network is activated and trained.

3.1 Motivation

As discussed in Chapter 2, previous approaches to connectionist modeling of natural language processing have focused either on syntactic parsing of linguistically-annotated corpora, or on psycholinguistic phenomena using specialized semantic grammars developed to highlight particular cognitive issues.

INSOMNet was designed to bridge this gap. In order to do so, the model must be able to satisfy two contrasting goals: it must both model language competence by distilling a sentence into a discrete linguistic conceptual structure, but also model language performance by retaining the cognitive properties of connectionist models, such as gradedness. To be a convincing parsing model, INSOMNet must scale up beyond toy grammars to the large corpora that have traditionally been the province of statistical and symbolic approaches. But these approaches typically assume a grammar, the rules of which are either written by hand or read off from the parse-tree labels in a syntactic corpus (Charniak 2000). The *a priori* specification of a grammar places hard constraints on what can be modeled. For example, prepositional phrase attachment ambiguity cannot be accounted for on the basis of statistical analysis of syntactic corpora alone. Lexical semantics and, in the most

general case, discourse, must also be considered. Connectionist models have been traditionally applied to language issues for which the specification of a grammar would be impossible, such as robust language processing or modeling human aphasia.

In this chapter, we first describe the semantic representation into which INSOMNet parses sentences. We then describe the components of INSOMNet, how they are activated as the network processes a sentence into a semantic representation, and how each is trained. We conclude with a discussion of the modeling issues that have influenced INSOMNet's design.

3.1.1 Semantic Representation

The linguistic conceptual structure we use for semantic representation is Minimal Recursion Semantics (MRS; Copestake et al. 2001). As described in Section 2.5.4, MRS uses flat semantics to encode linguistic dependencies and to avoid the combinatorial scoping problems that arise from ambiguities during sentence processing. Recall that flat semantics makes use of predicate labels called *handles* (i.e., *pointers* or *addresses*) that serve as fillers for argument slots in the set of MRS frames.

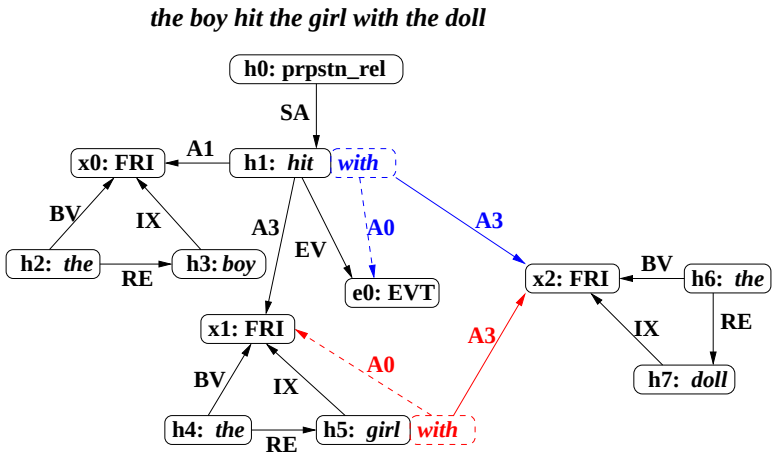
The MRS graph from Figure 3.1 can be rendered as a set of frames to serve as targets for INSOMNet. Each frame has the form

| **Handle Word Semantic-Relation Subcategorization-Type** <**Argument-List**> |.

For example, the node labeled **h1: hit** in the middle of the MRS graph in Figure 3.1 is described by the frame

| **h1 hit _arg13_rel A0A1A3DMEV _ x0 x1 _ e0 _** |

Although already described at length in Chapter 2.5.4, we will summarize here the elements in the frame representation given in Figure 3.1. The first element, **h1**, is the **Handle** (node label) of the frame; other frames can include this element in their argument slots to represent a dependency to this frame. For instance, **h1** fills the *state-of-affairs* (**SA**) slot in the topmost node, **h0 prpstn_rel**, as indicated by the arc labeled **SA**. The second element, *hit*, gives the **Word** for this frame (where applicable; many frames, such as the aforementioned **h0 prpstn_rel** frame that denote phrasal and clausal constituents, have no corresponding word in the input sentence). The third element is the **Semantic-Relation** for the frame. For purposes of exposition, we use the **Semantic-Relation** as the node value unless the frame has a corresponding **Word**. The third, **A0A1A3DMEV**, represents the **Subcategorization-Type** and is shorthand for the argument roles that the semantic relation takes; that is, it is the set of arcs from a node given in a canonical order (the arcs **A0** and **DM** are not shown in Figure 3.4 because their slots are empty). In this case, the subcategorization type indicates that *hit* is a transitive verb with three arguments: the agent is the frame the **A1** arc points to; the patient, the frame the **A3** arc points to; and the event, the frame the **EV** arc points to. The arc labels



h0	—	**prpstn_rel**	**SA**	**h1**	—	—	—	—	—	
h1	*hit*	**arg13_rel**	**A0A1A3DMEV**	—	**x0**	**x1**	—	**e0**	—	
h1	*with*	**miscprep_ind_rel**	**A0A3DMEV**	**e0**	**x2**	—	—	—	—	
e0	—	**EVT**	**DVASMOTN**	bool	−asp	ind	past	—	—	
h2	—	**def_explicit_rel**	**BVDMRESC**	**x0**	—	**h3**	—	—	—	
h3	*boy*	**diadic_noarg_nom_rel**	**A3IX**	—	**x0**	—	—	—	—	
x0	—	**FRI**	**DVGNPNPT**	−	masc	3sg	prn	—	—	
h4	—	**def_explicit_rel**	**BVDMRESC**	**x1**	—	**h5**	—	—	—	
h5	*girl*	**diadic_noarg_nom_rel**	**A3IX**	—	**x1**	—	—	—	—	
x1	—	**FRI**	**DVGNPNPT**	−	fem	3sg	prn	—	—	
h5	*with*	**miscprep_ind_rel**	**A0A3DMEV**	**x1**	**x2**	—	—	—	—	
h6	—	**def_explicit_rel**	**BVDMRESC**	**x2**	—	**h7**	—	—	—	
h7	*doll*	**reg_nom_rel**	**IX**	**x2**	—	—	—	—	—	
x2	—	**FRI**	**DVGNPNPT**	−	neu	3sg	prn	—	—	

Figure 3.1: **MRS Frameset Format for INSOMNet.** (***Color figure***) This figure shows the set of frames that correspond to the MRS dependency graph in Figure 2.16 (shown above for convenience). The first four fields, **Handle**, **Word**, **Semantic-Relation**, and **Subcategorization-Type**, are a part of every frame and describe the nodes in the graph. How the following six fields are interpreted depends on the value of the **Subcategorization-Type**. This field encodes the names of the arguments taken by the **Semantic-Relation** in the frame and has at most six fields. These argument roles correspond to the labeled arcs in the graph. For example, the node labeled **h1: hit** is a transitive verb denoted by the semantic relation **arg13_rel**, and has a **Subcategorization-Type** of **A0A1A3DMEV**. Three of the roles, **A1** (*arg1*), **A3** (*arg3*), and **EV** (*event*), have fillers (**x0**, **x1**, and **e0**, respectively), which are represented by arcs from the **h1: hit** node. The other two arguments, **A0** (*arg0*) and **DM** (*dimension*), are left unfilled, and are represented by "_" (the **Null** symbol). Because the sentence *the boy hit the girl with the doll* can be ambiguous, there are two frames containing the **Word** *with*: one which shares its handle (**h1**) with the *hit* frame, and the other which shares its handle (**h5**) with the *girl* frame. These frames correspond to the *with* nodes in the graph. Note that handle sharing is indicated by literally affixing the *with* nodes to their attachment nodes. During training, only one of these frames is presented, so that INSOMNet is only exposed to unambiguous interpretations. It must learn to represent and distinguish both interpretations.

themselves are abbreviations for MRS argument role names (e.g., **A1** is *arg1*, **EV** is *event*, and **BV** is *bound variable*). The rest of the frame (_ **x0 x1** _ **e0**) lists the **Argument**s (fillers) corresponding to these roles. Some slots are empty (**"__"** or **Null**) and, consequently, have no corresponding arcs in Figure 3.1. The handles refer to the other nodes in the MRS graph to which the arcs point.

It is important to point out two properties of MRS handles that have influenced the design of INSOMNet. First, a given handle does not necessarily uniquely identify a frame (node) in the MRS graph. Rather, it can be used to refer to several frames. As explained toward the end of Section 2.5.4, a handle can be used to denote predicate conjunction to represent linguistic relations that are optional (in both where and whether they may occur), or that may occur more than once and therefore may require more than one frame to represent, such as adjuncts (as in the example above), modifiers (such as adjectives and relative clauses), or verb-particle constructions (e.g., "*work* something *out*").

The second property illustrates an important difference between symbolic and subsymbolic representations. In the symbolic MRS specification, handles are arbitrary designators (e.g., the label **h1** has no meaning in itself). However, in a neural network, handles have to be encoded as patterns of activation. In the approach taken in this dissertation, the handles are designed to be dynamically associated with patterns that represent core semantic features, such as predicate argument structure. How these handle patterns are developed is an important aspect of the model and will be described in detail in Section 3.4.2.

3.2 Network Architecture and Activation

Before we begin to describe the architecture, its activation and training, it would be useful to motivate its design in terms of the semantic representation in Figure 3.1. Figure 3.2 shows how a graph- or frame-based representation such as that in Figure 3.1 might be represented in a grid, where the cells in the grids hold the components of individual MRS frames. The leftmost grid only shows the graph nodes and values without the labeled arcs. These arcs are added in the rightmost grid to indicate how the cells are denoted according to their fillers.

Figure 3.3 reveals how the use of the subcategorization information implicitly represents the labeled arcs. As can be seen, this representation completely describes the graph in Figure 3.1. Rather than with symbolic components, these frames are encoded through distributed representations, and a *decoder* network is required to pull out their individual components. How INSOMNet does this is described next.

INSOMNet is a supervised model of comprehension and parsing. The input to the entire model is a sentence in the form of a sequence of words presented to the Input Layer of the Sequence Processor. The targets for the model are a sequence of frame representations such as those in Figure 3.1 that are used to generate error signals for the Semantic Encoder/Decoder and Sequence Processor modules, as well as to self-organize the Frame Selector module (to be

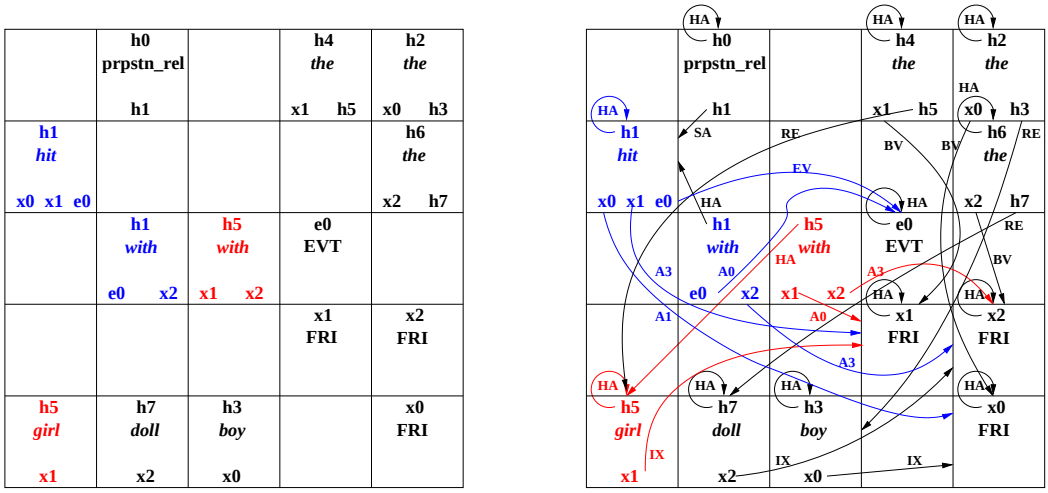


Figure 3.2: **Stages toward Representing Semantic Graphs in Neural Networks.** (***Color figure***) The grid on the left represents the basic symbolic information in the nodes and values from Figure 3.1, and the grid on the right adds the labeled arcs for roles and their fillers.

	h0 prpstn_rel SA h1		h4 *the* BVDMRESC x1 h5	h2 *the* BVDMRESC x0 h3
h1 *hit* A0A1A3DMEV x0 x1 e0				h6 *the* BVDMRESC x2 h7
	h1 *with* A0A3DMEV e0 x2	h5 *with* A0A3DMEV x1 x2	e0 EVT DVASMOTN	
			x1 FRI DVGNPTPN	x2 FRI DVGNPTPN
h5 *girl* A3IX x1	h7 *doll* IX x2	h3 *boy* A3IX x0		x0 FRI DVGNPTPN

Figure 3.3: **Representing a Semantic Graph in Neural Networks.** (***Color figure***) Subcategorization information can be used to indicate roles and fillers instead of explicit arcs.

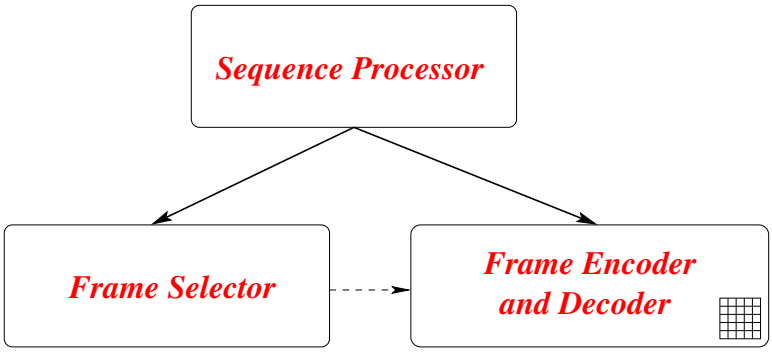


Figure 3.4: **Overview of the INSOMNet Architecture.** (***Color figure***) The INSOMNet model consists of three operational modules based on how they function together. The Sequence Processor reads the input sentence in one word at a time and activates both the Frame Selector and the Semantic Frame Encoder and Decoder. The Semantic Frame Encoder and Decoder encodes the MRS dependency graph for the semantic interpretation of the sentence as it is incrementally processed. Frame Selector is trained to select frames in a graded manner corresponding to an *a posteriori* probability that those frames belong to the current semantic interpretation. The solid arrows indicate weights and the dashed arrow represents a conceptual correspondence between the modules.

described in Section 3.4.2.

The INSOMNet sentence parsing architecture (Figure 3.4) consists of six components, which we will consider in three operational modules:

Sequence Processor

- A SRN to read in the input sequence.
- A SARDNet Map that retains an exponentially decaying activation of the input sequence.

Semantic Frame Encoder and Decoder

- A Frame Map that encodes the frames of the MRS dependency graph.
- A Frame Node Decoder Network that generates the components of the frame representations.

Frame Selector

- A Frame Node Modulator Map that learns to control the activation levels of the encoded frame patterns in the Frame Map.
- A self-organized Frame Node Indicator Map used to provide addresses for those Frame Nodes in the Frame Map holding encoded frames, as well as to serve as a target to train the Frame Node Modulator Map.

We describe each module in turn in the following sections, but first give an overview of the activation of the network.

The Sequence Processor takes as input a sequence of the words from a sentence. Each word is sequentially presented to the Input Layer of the Sequence Processor. The SardNet Map is an enhancement to the basic Simple Recurrent Network that forms the core processing component of the Sequence Processor; its purpose is to help the network retain long-distance dependencies by keeping token identities of input words explicit whenever possible.

The Sequence Processor generates patterns of activation in both the Frame Selector and the Semantic Encoder and Decoder modules according to how those modules have been trained.

The input signals from the Sequence Processor result in a pattern of activation across the Frame Map component. These patterns are divided up over a grid so that each cell in the grid represents the encoding of one particular frame that may belong to the semantic interpretation of the input sentence. In this way, the Frame Map functions as a square grid of second hidden layers, each cell (or Frame Node) holding a compressed MRS frame representation. Which patterns are actually to be considered as targets are indicated by the Frame Selector module.

All Frame Nodes holding compressed MRS representations are decoded by a set of shared weights to recover the components for the complete, uncompressed MRS frame representation. These components are developed by approximating the target components of the frames that make up the actual interpretation of the sentence.

The Sequence Processor also activates the Frame Selector module to activate just those frames that belong to the current semantic interpretation. Because the Frame Node Modulator Map is trained to approximate the binary targets in the Frame Node Indicator Map, it yields an *a posteriori* distribution of graded frame activations that represent INSOMNet's confidence that a particular frame belongs to the interpretation.

3.3 INSOMNet Activation

Having presented an overview of how INSOMNet is activated, we will describe the activation of the modules in detail.

3.3.1 Sequence Processor

The Simple Recurrent Network (SRN; Elman 1990) is the standard neural network architecture for sequence processing, and it forms the basis for the INSOMNet architecture as well. As described in Section 2.3.3, the SRN reads a sequence of distributed word representations as input and activates the Frame Map to develop a semantic interpretation of the sentence at the output. At each time step, a copy of the hidden layer is saved and used as input during the next step, together with the next word. In this way, each new word is interpreted in the context of the entire sequence read so far, and the final sentence interpretation is gradually formed at the output.

The SARDNet Map (Mayberry and Miikkulainen 1999) is included to solve the long-term memory problem of the SRN. SARDNet is a self-organized map for a sequence of word repre-

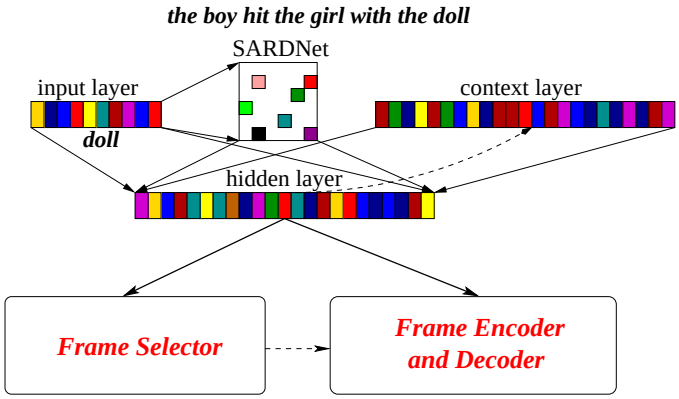


Figure 3.5: **Sequence Processor.** (***Color figure***) The sequence processor of the INSOMNet model handles the incremental processing of the sentence input. It consists of a Simple Recurrent Network and the SARDNet Map. The SRN component reads in the sentence one word at a time. In the figure, the representation for the current input word *doll* is shown at the top left. A unit corresponding to the current input word is activated on the SARDNet Map (at top center) at a value of 1.0 and the rest of the map is decayed by a factor of 0.9. As each word is read into the input buffer, it is both mapped onto the SARDNet map and propagated to the hidden layer. A copy of the hidden layer is then saved (as the context layer) to be used during the next time step.

sentations (James and Miikkulainen 1995). As each word from the input sequence is read in, its corresponding node in the SARDNet map is activated at a value of 1.0, and the rest of the map decayed by a factor of 0.9. The winning node is then removed from competition. If an input word occurs more than once or two inputs map to the same node, the next closest available node is activated. Together with the current input and context layer, the SARDNet Map is used as input to the hidden layer. The SARDNet helps to identify each input token, information that would otherwise be lost in a long sequence of SRN iterations.

3.3.2 Semantic Frame Encoder and Decoder

The Frame Map is the primary innovation of INSOMNet, which allows the dynamic instantiation of the semantic interpretation of an input sequence. In the current model, the Frame Map consists of a 12×12 map of Frame Nodes. Each node in the map itself is a 100-dimensional vector. (However, these model parameters are arbitrary, and larger and smaller maps also work.) Thus, the Frame Map is implemented as a second hidden layer (which can be regarded as a grid of second hidden layers). As a result of processing the input sequence, a number of these nodes will be activated; that is, a particular pattern of activation appears over the units of the Frame Nodes. These patterns of activation encode the MRS frames that constitute the semantic interpretation of the current sentence. Which MRS frame a given Frame Node will represent is not stipulated as part of the network design; rather, INSOMNet must learn to associate nodes with frames having

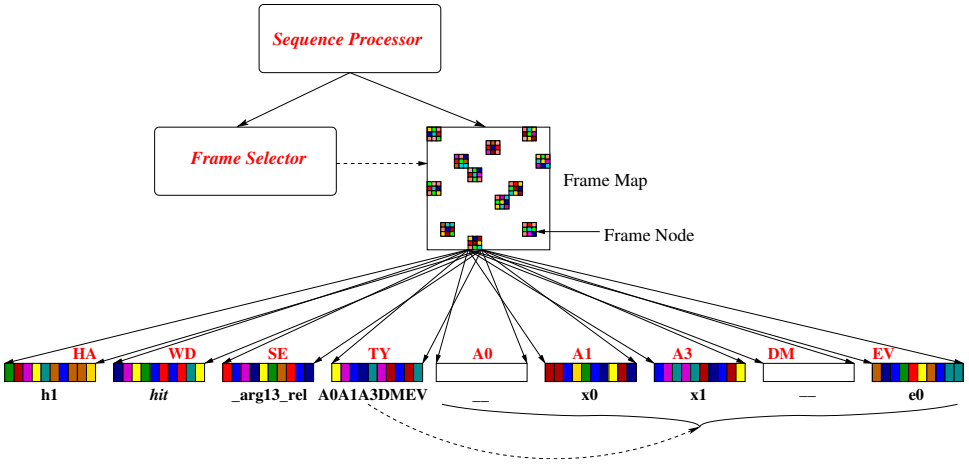


Figure 3.6: **Semantic Frame Encoder and Decoder.** (***Color figure***) The output frame shown in the figure is actually a decoding of the pattern (the multi-shaded squares) in the Frame Map. The other patterns in the Frame Map correspond to the other nodes in the full MRS dependency graph shown in Figure 2.16; their decodings are not shown. The hidden layer from the Sequence Processor is propagated to the Frame Map, which is a 12×12 map of Frame Nodes, each consisting of 100 units (shown here as 3×3 patterns). The units in each Frame Node are connected through a set of shared weights that comprise the Frame Node Decoder network to an output layer representing a frame. In this way, the Frame Map is actually functions as a second hidden layer. Thus, for example, the Frame Node in the top right of the Frame Map decodes into the frame | **h1 hit _arg13_rel A0A1A3DMEV _ x0 x1 _ e0** |. Note that the argument slots and their fillers are bound together by virtue of the handle representation (such as **h1** between *hit* and the **SA** slot of **prpstn_rel**).

similar semantic structure.

The patterns in the Frame Nodes are decoded into their corresponding MRS frames through the weights in the Frame Node Decoder. The same set of weights in the Decoder are used for each node in the map. This weight-sharing enforces generalization among common elements across the many frames in any given MRS dependency graph.

Because the Frame Map is based on self-organization of compressed encodings of MRS frames (as will be described in Section 3.4.2) that represent MRS handles to serve as *frame addresses*, similar frames cluster together on the map. For example, patterns encoding determiners will tend to occupy one section of the map, the various types of verbs another, nouns yet another, and so on. However, although each node becomes tuned to particular types of frames, no particular Frame Node is dedicated to any given frame. Rather, through different activation patterns over their units, the nodes are flexible enough to represent different frames, depending on what is needed to represent the input sequence. For example in Figure 3.6, the node at the bottom center of the Frame Map decodes into the | **h1 hit _arg13_rel A0A1A3DMEV _ x0 x1 _ e0** | frame for this particular sentence. In another sentence, it could represent a different verb with a slightly different subcategorization type (e.g. **A0A1A4DMEV**) and its associated arguments. This feature of

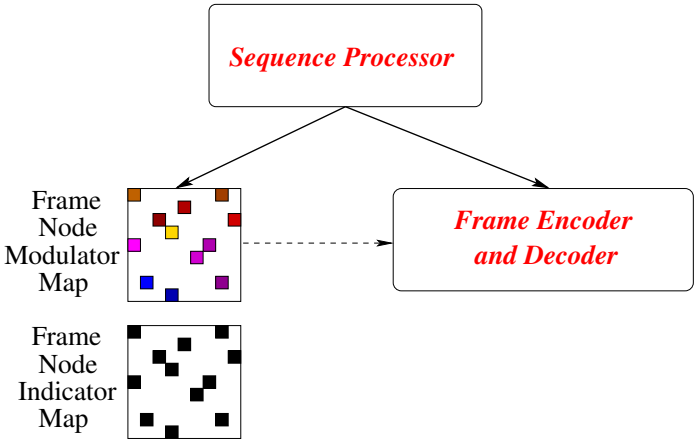


Figure 3.7: **Frame Selector.** (***Color figure***) The Frame Selector network has an output layer of units that are in one-to-one correspondence with the Frame Nodes in the Frame Map (indicated by the dashed arrow; cf. Figure 3.6 to note the similarity in their respective patterns). Each unit's activation strength represents the network's confidence that the corresponding frame node is supposed to be active in the Frame Map, in effect serving as a mechanism for frame selection. Due to the gradedness of the unit activations, this selection is not binary, allowing for the tentative inclusion of frames according to the current input sequence. In this way, the Frame Node Modulator Map allows for the modeling of such psycholinguistic effects as expectations and defaults, multiple frame (sense) coactivation in the face of ambiguity, semantic priming, and nonmonotonic revision of an interpretation as a sentence is incrementally processed. The Frame Node Indicator Map, which is also in one-to-one correspondence with the nodes in the Frame Map, is used during training to semantically self-organize the Frame Map, and also serves as a target for the Frame Node Modulator Map. How this is done will be taken up in Section 3.4 on training the Frame Node Modulator Map component. During the activation of INSOMNet in the course of processing a sentence, the weights of the Frame Node Indicator Map, now self-organized, determine the frames in the Frame Map to which the arguments in a decoded Frame Node pattern correspond. This development of representations for handles is explained in more detail in Section 3.4.2.

the architecture makes the Frame Map able to represent semantic dependency graphs dynamically, enhancing generalization.

3.3.3 Frame Selector

The role of the Frame Node Modulator Map (Figure 3.7) is to model graded frame selection by representing the activation levels of the Frame Nodes in the Frame Map, with which they are in one-to-one correspondence. Graded frame selection is used to allow for cognitive modeling of such psycholinguistic effects as expectations and defaults, semantic priming, multiple sense coactivation in the face of ambiguity, along with nonmonotonic revision of an interpretation (such as sense preference upon encountering disambiguating information) in the course of sentence processing. Thus, as each word is read in by the Sequence Processor, INSOMNet indicates the current semantic interpretation by activating units on the Frame Node Modulator Map to select the Frame

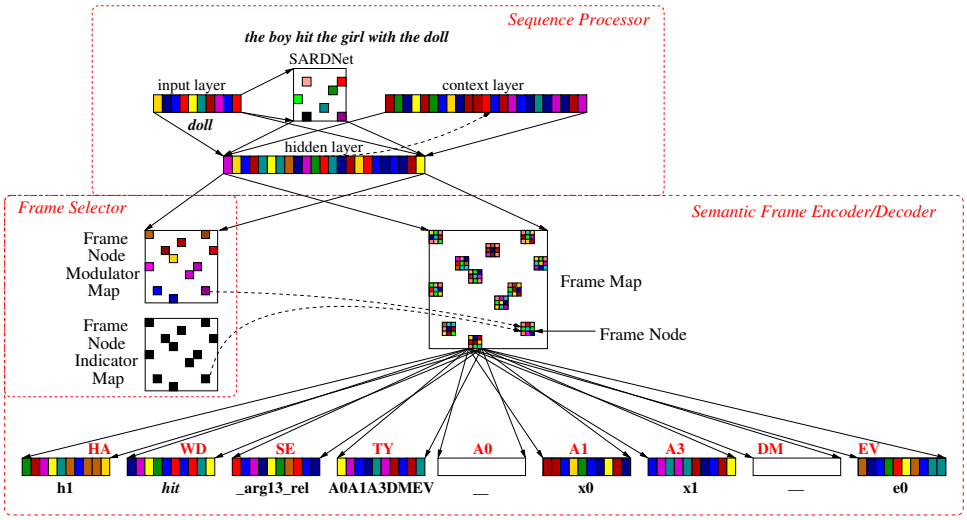


Figure 3.8: **The INSOMNet Architecture.** (***Color figure***) INSOMNet processes sentences incrementally to build up semantic interpretations that can be actively revised during processing.

Nodes encoding the MRS frames in that interpretation. The Frame Node Indicator Map is used to train the Node Modulator, a process which is explained in the next section.

The Frame Selector module was developed for the practical reason that it saved training time. The weights connecting the hidden layer of the Sequence Processor to each node in the Frame Map could have been trained to control the level of activation of the Frame Node patterns themselves, but doing so introduces the inefficiency of having to train the majority of nodes to be null vectors (there are 144×100 weights between the hidden layer of the Sequence Processor and Frame Map), since only a fraction of the 144 nodes would be activated for any sentence. The result is a strong bias toward the null vector, which had been found to hinder training for those nodes that should be activated.

3.4 Training INSOMNet

Now that we have examined INSOMNet module by module, it would be useful to consider the network itself. Figure 3.8 puts all of the components just described together into a conceptual whole.

INSOMNet can be trained in a single phase, but using two distinct phases results in slightly better performance. In the first phase, only the components using self-organization, the SARDNet Map and the Frame Node Indicator Map, are trained. Once these maps have settled, the second phase begins, in which the rest of the network is trained.

We first present an overview of the training process, then describe training each module in detail.

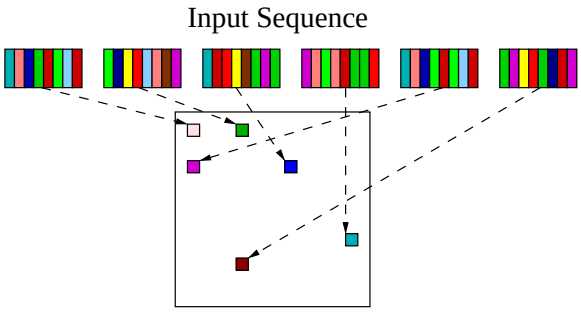


Figure 3.9: **SARDNet.** (***Color figure***) Each word is input to the SARDNet Map sequentially, which builds a representation for the sequence word by word. SARDNet is a map of a sequence of word representations, and is trained through the Self-Organizing Map (SOM) algorithm (Kohonen 1984, 1990), with the important modification that winning units remain activated and not competitive, but are decayed by 0.9. In this manner, the map develops a unique representation for the input sequence. As a localist encoding of the input sequence, SARDNet is subject to the disadvantages of such representations. Yet, because the map is coupled with the distributed layers of the SRN, this problem rarely arises. The reason is that the sequence that develops on the map is itself a pattern, such that similar sentences entail similar SARDNet representations, that not only help the SRN retain long-distance dependency information in order to develop the correct output, but also tends to offset the brittleness of the localist map representation by helping the SRN latch gradient information that is propagated back to the beginning of the sequence.

The Sequence Processor is trained through backpropagation-through-time, receiving error signals from both the Frame Selector and Semantic Encoder and Decoder modules. The SardNet map is trained by a variation on the SOM map for sequences.

Target frames are presented to the decoding portion of the Semantic Frame Encoder and Decoder module, and the output from each Frame Node that holds the compressed frame representation is compared with the target to generate an error signal. All the error signals determined in this manner are backpropagated up through the Semantic Encoder and Decoder to the Sequence Processor, with appropriate weight changes made.

The error signal from the Frame Selector is generated internally as the Frame Node Modulator Map is compared with the self-organized Frame Node Indicator Map. The Frame Node Indicator Map itself is self-organized on the basis of compressed representations of the frames that then are used as representations for the MRS frame handles.

3.4.1 Sequence Processor

The Simple Recurrent Network is trained with backpropagation-through-time (BPTT; Williams and Zipser 1989; Lawrence et al. 2000) to improve the network's ability to process longer sequences. With BPTT, the SRN is effectively trained as if it were a multi-layer feedforward network, with the constraint that the weights between each layer are shared. SARDNet (see Figure 3.9) enhances this ability by allowing the SRN to latch information in order to propagate gradients back to the

beginning of the sentence.

3.4.2 Semantic Frame Encoder/Decoder

The approach taken to training the Frame Map was developed to solve one of the shortcomings in previous models of incremental sentence interpretation, as described in Section 2.3.3. There, it was shown how a SRN can be applied to the task of case-role analysis, but that the prespecification of the number and types of roles was a major obstacle to scaling this approach to sentences with realistic complexity.

What if, instead of specifying the role each assembly in the output would represent, the network were allowed to discover which role should map to which output assembly (Frame Node) by some means? It would still be necessary to stipulate the maximum number of nodes and, thus, roles, that any given sentence could have, but the assemblies could be much more flexible in which role they stood for in any given sentence.

With this goal in mind, a way of self-organizing the Frame Map was sought so that those nodes best suited to encode a given role in a sentence would reliably come to do so. One elegant way to do this would be to use the patterns in the nodes themselves in the course of training the network to self-organize the Frame Node Indicator Map, but this would require already knowing which node would be used to represent which frame by the end of the sentence. This information is not available at the beginning of the sentence, nor could it be determined without propagating to all the nodes in the Frame Map, and then performing a search and sorting on some metric to determine which pattern gives the desired frame decoding, and the node that the frame should have been mapped to, resulting in substantially increased training time.

The next best approach is to develop a complementary mechanism that also encodes frames, but which can do so apart from activating any link from the hidden layer to a node in the Frame Map. To accomplish this task, we self-organize the Frame Node Indicator Map, whose units are in one-to-one correspondence with the Frame Nodes in the Frame Map; we will come back to this point presently, but it helps to think of the Frame Map itself as being self-organized since the patterns that develop on it do indeed show topological organization during training by virtue of the fact that the Shared Decoder weights are used to decode them into their frame constituents.

To this end, we turn to RAAM to develop compressed frame representations through auto-associating their constituents. This approach has a second advantage as well: the compressed frame encoding can be used as the handle for the frame represented by the encoding. This handle, in turn, can be used as the filler of a slot that points to the frame. In effect, the handle is a *content-addressable* pointer to the Frame Node that encodes the complete frame. An example will help make this clearer. In Figure 3.10, the frame constituents of **girl diadic_n_rel A3IX _ x1** are auto-associated through a RAAM network, and the resulting hidden layer, which is a compressed representation of the frame, is assigned to the handle **h5**, the label for this frame. Similarly, the frame constituents (which now include the **h5** handle representation) of **the _def_rel BVDMRESC x1 _**

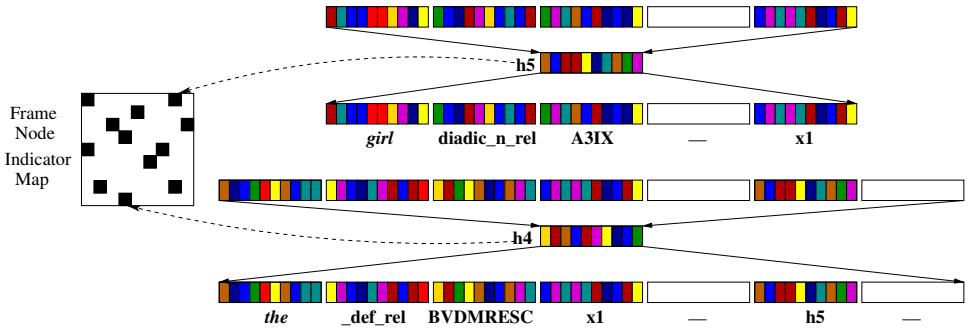


Figure 3.10: **Handles as Basis of Self-organization.** (***Color figure***) The figure shows the encoding of two handles, **h5** and **h4** (top and bottom figures on the right). Once the representation for **h5** has been developed, it can then be used as a filler for the *restriction* (the slot corresponding to **RE** in the subcategorization type **BVDMRESC** that signals a determiner in the bottom right figure). This approach implements binding. The **h5** and **h4** representations are also used to self-organize the Frame Node Indicator Map (map on the left). The activation of the winning units in the Frame Node Indicator Map then serve to address the corresponding Frame Node in the Frame Map that will hold the frame pattern that can then be decoded with the Decoder Nework into its proper frame constituents. The dashed arrows point to the units in the Frame Node Indicator Map that the two patterns map.

h5 _ are auto-associated through the RAAM network, with the resulting hidden layer assigned to the handle **h6**. Note that the first frame must be compressed before the second so that the second can use its handle **h5** in its *restriction* (**RE**) slot.

We can develop a compressed representation for each frame in the MRS dependency graph, because we know that the graph is acyclic. Thus, starting from the leaves of the graph (which will be the frames specifying nominal or verbal features (such as **x1**, a *full referential index* with features of -, fem, 3sg, and prn), we encode these features to get a compressed representation for the handle **x1**, which can be used in the frames that have these leaf frames (which we term *feature frames*) as fillers (for example, the | **h5 girl diadic_n_rel A3IX _ x1** | frame, where **x1** is the filler for the *instance* **IX** argument. Continuing in this manner, we eventually reach the top frame | **h0 _ prpstn_rel SA h1** |, which can be compressed using the handle representation for **h1**.

Once we have all of the handles for the frames in the MRS dependency graph, we can present them in reverse order (so as to preserve sentence order as much as possible) to the Frame Node Indicator Map. The Frame Node Indicator Map is also trained as a SARDNet map (see Figure 3.9), but with a decay rate of 1.0, since the nodes are supposed to indicate simply whether a frame is present or not. At this point, we know which frame is supposed to be in which Frame Map node, so during training, the Frame Node serves as a second hidden layer and the frames as the output layer. The appropriate frames are presented as targets for the patterns in the Frame Node layer, and the resulting error signals are backpropagated through the Frame Node Decoder weights to the Frame Node layer and on up to the first hidden layer of the SRN.

It is important to note that self-organizing maps (like SARDNet) are conventionally trained

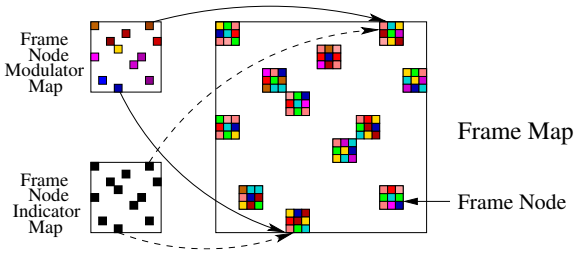


Figure 3.11: Frame Map **Self-organization.** (***Color figure***) Associated with the Frame Map is a Frame Node Indicator Map, whose units are in one-to-one correspondence with the nodes in the Frame Map. If a unit is active on the Frame Node Indicator Map, then the pattern in the corresponding Frame Node is a target frame and should be decoded into a MRS frame. Whether it actually can be decoded is indicated by the the activation strength of the corresponding unit in the Frame Node Modulator Map. The Frame Node Indicator Map is self-organized on the basis of the compressed frame encodings that become the representations for the frame handles. The Frame Node Modulator Map uses the Frame Node Indicator Map as its target to learn which frames should be active for any given sentence to control their activation strength.

with static targets (or those drawn from an input space with a fixed distribution). However, the use of RAAM forces us to train the Frame Node Indicator Map with a set of moving targets, but targets that are nevertheless slowly converging to stable patterns, which is typical of RAAM training. But this factor does complicate training INSOMNet because as long as the Frame Node Indicator Map is training, the patterns can oscillate among nodes. Yet, it is precisely this property that gives the Frame Map topological organization. The primary complication occurs in training the Frame Node Modulator Map, described in the next section.

3.4.3 Frame Selector

The Frame Selector takes the Sequence Processor hidden layer activation and propagates it through to the Frame Node Modulator Map (see Figure 3.11). The output is compared with the self-organized Frame Node Indicator Map, and the resulting error signal is backpropagated to the hidden layer. In this way, the Modulator output comes to represent the network's confidence that the activated frames are part of the semantic interpretation of the input.

However, training the Frame Selector network is not started until the Frame Node Indicator Map is essentially finished self-organizing (say, the neighborhood is at 0, or the learning rate is below some threshold). Otherwise, the network tries to learn units that might be oscillating since the RAAM network has likely not converged to a set of stable patterns, resulting in units that should otherwise be 0.0 being activated.

3.5 Conclusion

The INSOMNet parser is an incremental model that uses the novel technique of self-organizing the nodes in the Frame Map layer to let the network discover an optimal placement of frame encodings. The nodes hold the activation patterns that can be decoded into the frame constituents of the MRS dependency graph that represents the semantic interpretation of the sentence being processed. A Modulator Network indicates the network's confidence that the nodes corresponding to activated units represent frames that are part of the current interpretation. This approach allows for the modeling of psycholinguistic effects having to do with expectations, ambiguity, and semantic priming.

Chapter 4

Basic Performance Evaluation

An important goal of this research is to provide a basic performance comparison between INSOMNet and a non-trivial (as opposed to n-gram or nonlexicalized PCFG) statistical model on the same corpus. A number of statistical models have been recently evaluated on the Redwoods Treebank (Toutanova and Manning 2002; Toutanova et al. 2002) on the task of parse selection. Unfortunately for our purposes at present, most of these models rely heavily on the syntactic analyses of the HPSG parse trees in the corpus, whereas we have concentrated primarily on semantics. Nevertheless, in an as-yet unpublished study, a conditional log-linear model has been informally evaluated on a particular type of semantic dependency graph that captures the core features of the full MRS structure. This semantic model extends the model trained on *semantic dependency trees* (Toutanova and Manning 2002; Toutanova et al. 2002) to a subset of the MRS structure called *elementary semantic dependency graphs*. We will begin the comparison study with a description of these elementary semantic dependency graphs, the dataset they were used in, and follow up with a description of the experiments that were run to evaluate INSOMNet's performance on the same dataset and task. We will compare and discuss the results and relate them to the differences in the two models.

# parses	1	2	5	10	20	50	100	250	500
# sentences	1478	2040	762	502	413	94	12	4	2
percentage	27.85	38.44	14.36	9.46	7.78	1.77	0.23	0.08	0.04

Figure 4.1: **Parse Statistics.** Almost 75% of the 5307 sentences in the elementary dataset have two or more parses licensed by the ERG, but only 112 ($\approx$ 2%) have more than 50.

4.1 Elementary CSLI Dataset

The dataset used for comparison in this chapter is a corpus of 5307 sentences from the June 10, 2001 version of the Redwoods Treebank. These sentences were selected on the basis that the annotator chose exactly one analysis licensed by the ERG as the preferred parse.

One factor to be considered is that the corpus has a number of repeated sentences, including one sentence, *let us see*, that shows up 134 times. Nevertheless, there remain 4388 unique sentences in the corpus. Following Toutanova et al. (2002), I give a breakout of the number of sentences having one or more analyses to give a sense of the parse disambiguation task that will be the subject of the experiments in Section 4.2.1.

As can be seen from Table 4.1, roughly 28% of the analyses yield a single parse, while almost 40% have two. Only two parses had 500 and 507 parses, respectively, to be ranked. The comparatively small number of parses is a testament to the accuracy of the ERG.

The average length of sentences in the Redwoods Treebank is seven words, and the average lexical (the number of sense distinctions) and structural ambiguity (the number of syntactic analyses licensed by the English Resource Grammar) is 4.1 and 8.3, respectively (Toutanova et al. 2002).

The sentences in the dataset are annotated with elementary semantic dependency graphs for the 5307 sentences. Elementary semantic dependency graphs (see Figure 4.2 for an example) are a subset of the full MRS structure designed to capture the basic predicate argument structure of a sentence while omitting some detail. Since we designed INSOMNet to represent the full MRS structure and the experiments in subsequent chapters use the full encoding, a comparison between elementary semantic dependency graphs and the full MRS dependency graphs will be taken up in Chapter 5, in which the performance of INSOMNet on the more complex semantic representation is evaluated.

4.2 Models and Experimental Setup

As the architecture and training of INSOMNet has already been described at length in Chapter 3, we will first review the statistical model against which INSOMNet is compared, and then describe the dataset format required to represent elementary semantic dependency graphs for INSOMNet. We will also describe the models' respective experimental setups, and conclude with a comparison between the two models.

4.2.1 Conditional Log-Linear Model

We begin with a look at the conditional log-linear model over elementary semantic dependency graphs that INSOMNet is compared against. We first describe the model and the features it was trained on, and then present results.

The *conditional log-linear model* is a statistical approach that uses the *maximum-entropy* framework to estimate the probability of an analysis t_i from among a set of analyses $T = \{t_1, \ldots, t_k\}$ (typically, these are parse trees–hence, the variable t_i–but need not be) for a sentence s over a set of m specified features $F = \{f_1, \ldots, f_m\}$ selected by the modeler. The value of each feature f_j is the number of times that feature occurs in the sentence. Corresponding to each feature f_j is a weight λ_j that can take on any real value. The weights are adapted to maximize the entropy[1] of the model (i.e., make the distribution as uniform as possible to minimize model bias) while keeping the model's estimated probabilities as close to the empirical probabilities of the training set. A typical approach to learning the weights is with *generalized iterative scaling*, which is guaranteed to converge (Manning and Schütze 1999), but requires binary features. An optimization technique called *conjugate gradient* does not have this constraint and was used instead in Toutanova and Manning (2002).

The task is to classify a sentence s over the set of analyses T. The conditional probability for a particular analysis t_i is given by the equation[2]

$$P(t_i|s) = \frac{X}{Z} \tag{4.1}$$

where $P(t_i|s)$ is the probability of analysis t_i given the sentence s. The numerator

$$X = e^{\sum_{j=1}^{m} f_j(t_i)\lambda_j} \tag{4.2}$$

represents a weighted sum of the features, and the denominator

$$Z = \sum_{i'=1}^{k} e^{\sum_{j=1}^{m} f_j(t_{i'})\lambda_j} \tag{4.3}$$

is the *partition function* (Charniak 2000), which serves as a normalizing constant to yield a proper probability distribution by taking into account all of the parse possibilities. The log-linear model[3] is termed *discriminative* because it attempts to estimate the posterior probability $P(t_i|s)$ directly from the feature set rather than using Bayes' rule to calculate that probability through a joint probability $P(t_i, s)$ over the sentences s and analyses $t_1, \ldots, t_k$, as in the approach used by *generative* classifiers (Ng and Jordan 2002). Both types of models are compared in Toutanova and Manning (2002), and the conditional log-linear model is found to be superior on all variations of the models examined. The success of the conditional log-linear model in this domain is not surprising given the

[1] That is, we select the model q from the set of models over the same feature set, Q_F, by calculating each model's entropy $H(q) = -\sum_{t_i} q(t_i)\log q(t_i)$.

[2] It is worth noting that this equation is really a form of the *soft-max* function, which is a generalization of the logistic function. In effect, it gives a smoothed out version of the winner-take-all competitive model.

[3] The model is called *log-linear* because it can be rewritten as a linear combination of the logs of the weights and features: $\log P(t_i|s) = \sum_{j+1}^{m} f_j(t_i)\lambda_j - \log Z$. Using logarithms instead of the equation as given has important computational implications. Particularly, it avoids round-off errors when the weights become too large or small.

many advantages such models have over generative models. It is easily adaptable to new features sets selected by the modeler, and, more importantly, makes no independence assumptions that usually do not hold in the natural language domain. Thus, the modeler is free to choose features, even if they overlap. Yet, there is a tradeoff in specificity of the features chosen and their frequency in the corpus. If they are too specific, they will occur rarely, leading to data sparsity and loss of performance due to overfitting the training data. The approach used in Toutanova and Manning (2002) to combat this tendency is to impose a Gaussian prior over the distribution for smoothing. Smoothing entails assuming that the weights (as given by e^{λ_j}) have a normal prior probability distribution and finding their maximum *a posteriori* distribution (Chen and Rosenfeld 1999).

Before describing the feature set, we will first show how the elementary semantic dependency graph in Figure 2.13 (c) was preprocessed for the statistical model. The top index **_4** was eliminated, as well as all other handles such as **e2** and **x21**. This left just the relation, its arguments, and their fillers. Thus, the elementary semantic dependency graph becomes

int_rel[SOA _want2_rel]
_want2_rel[ARG1 pron_rel, ARG4 hypo_rel]
def_rel[BV pron_rel]
hypo_rel[SOA _meet_v_rel]
_on_temp_rel[ARG1 _meet_v_rel, ARG3 dofw_rel]
dofw_rel[NAMED :tue]
def_np_rel[BV dofw_rel]

For each of the semantic relations having n arguments, Toutanova defined $2n + 1$ features according to the template:

X_rel arg$_1$ **arg**$_2$ **... arg**$_n$
X_rel arg$_1$ **filler**$_1$
X_rel arg$_2$ **filler**$_2$
$\vdots$
X_rel arg$_n$ **filler**$_n$
arg$_1$ **filler**$_1$
arg$_2$ **filler**$_2$
$\vdots$
arg$_n$ **filler**$_n$

These features represent the argument labels for each relation, the role that relation's argument tends to go with, as well the relations the role itself tends to take. Even though these features clearly overlap, the use of maximum entropy allows the modeler to select the most appropriate features without having to worry about independence assumptions. As an example of how these

features are applied to the predicate **_want2_rel[ARG1 pron_rel, ARG4 hypo_rel]**, we get the set of features:

_want2_rel ARG1 ARG4
_want2_rel ARG1 pron_rel
_want2_rel ARG4 hypo_rel
ARG1 pron_rel
ARG4 hypo_rel

The total number of features thus specified from the corpus is 15522. Of these features, fully 30% of them only occur once (thus the need for smoothing), while the one feature **def_rel BV** occurs 74914 times. These features are encoded as high-dimensional sparce vectors.

Recall that we are comparing INSOMNet's performance against a conditional log-linear model on the same dataset over features that were selected from the elementary semantic dependency graph annotations, without recourse to the syntactic information in the HPSG signs. This particular study was performed at our request, and the resulting average accuracy of the model was 67.44%[4] using tenfold cross-validation under an exact tree match parse selection criterion. The average training accuracy was 75.00%. Accuracy was measured over the number of sentences for which the parser chose the correct analysis. If m analyses for a sentence are ranked equally, as often happened, then $\frac{1}{m}$ credit was given.

For the sake of completeness, we will also report the results for a similar study published in Toutanova et al. (2002) over *semantic dependency trees* annotated with HPSG schema names[5] to show how much the extra syntactic information provided by the schema annotations and the tree format helps in the parse disambiguation task. The study compares a generative model using a PCFG with a conditional log-linear model on the same features. Using tenfold cross-validation, the PCFG model achieved an accuracy of 69.05%, whereas the log-linear model resulted in almost an 8% improvement, with an accuracy of 74.30%. Yet, this performance is still quite far behind the main contribution of the study, which is a log-linear model that combined the semantic dependency tree model with a tagger and one that used up to four ancestors (labels from parent nodes). The combined PCFG model has an accuracy of 79.84% and the corresponding log-linear model yielded 82.65%.

The authors present a preliminary error analysis of 68 sentences from one of the test sets the combined parser failed on. We will reserve their analysis for Section 4.4, where we can compare with the types of errors that INSOMNet makes.

do you want to meet on tuesday .

h0	—	**int_rel**	**SA**	**e0**	—	—	—	—	—	
e0	—	**_want2_rel**	**A0A1A3A4DMEV**	—	**x0**	—	**h1**	—	—	
h1	—	**hypo_rel**	**SA**	**e1**	—	—	—	—	—	
e1	—	**_on_temp_rel**	**A0A3DMEV**	**e2**	**x1**	—	—	—	—	
e2	—	**_meet_v_rel**	**A0A1DMEV**	—	**x0**	—	—	—	—	
h2	—	**def_rel**	**BVDMRESC**	**x0**	—	—	—	—	—	
x0	—	**pron_rel**	**IX**	—	—	—	—	—	—	
h3	—	**def_np_rel**	**BVDMRESC**	**x1**	—	—	—	—	—	
x1	—	**dofw_rel**	**A3IXND**	—	—	***tuesday***	—	—	—	

Figure 4.2: **Elementary Semantic Dependency Graph Frameset Format.** This figure shows the set of frames that correspond to the elementary semantic dependency graph in Figure 2.13 (c) for the sentence *do you want to meet on tuesday*. As described in Section 3.1.1, the first four fields (**Handle**, **Word**, **Semantic-Relation**, and **Subcategorization-Type**) are a part of every frame, and the remaining six fields are determined by the **Subcategorization-Type**. With the exception of the argument slots, *Named* (**ND**) and *Const_Value* (**CV**), that are filled by word representations (such as *tuesday* in the figure), all argument roles are filled by pointer (**Argument**) fields or left empty (**Null** or "_").

4.2.2 INSOMNet

The elementary dataset used to train and test INSOMNet was converted from the elementary dependency graph format in Figure 2.13 (c) into the frameset format shown in Figure 4.2 (see Section 3.1.1 for a detailed description of this format, particularly with respect to how arguments are decoded based on the **Subcategorization-Type** field). Because INSOMNet is sensitive to the order of the frames (recall that the order must permit the proper encoding of the handles as explained in Section 3.4.2), the elementary semantic dependency graph frames were reordered so that the frame corresponding to any handle that occurred as an argument in some frame occurs after the frame in question. In this way, the frameset for a given sentence could be encoded in reverse order, back up to the top node. Note that the result is not a tree, since there are frames whose handles do not occur as arguments in any other frame. Also, the **Handle**s were renumbered, although the prefixes **e** and **x** were kept where they occurred (handles with an underscore, such as the **_2** label for the **hypo_rel** in Figure 2.13, were given the prefix **h**). This convention was useful in determining the leaves of the MRS graph while running INSOMNet. The only words from the input sentence that occur as targets argument slots are **ND** (*Named* for days of the week, months of the year, and people and place names) and **CV** (*Const_Value* for minutes of the hour such as *o'clock* and *thirty*). We have included these under the **Word** component heading.

Morphemes were represented as separate tokens in the input sequence, and irregular forms

[4]K. Toutanova, personal communication, 2003.

[5]These are the labels such as **hcomp** and **bse_verb** in the top tree in Figure 2.13 (a).

were replaced by their stems. For example, in the sentence *you say -d you were get -ing in tuesday night*, the input word *said* is rendered as *say -d*, and the extra morphemes *-d* and *-ing* are processed in separate steps. There were six such morphemes: *-s*, *-ing*, *-er*, *-est*, *-ly*, and *-d* (used for both simple past and perfect participles). Common inflected words, such as *am*, *are*, *were*, etc., were left unchanged. Such preprocessing is not strictly necessary, but it allows focusing the study on semantic processing without confounding it with morphology.

All non-handle frame constituents, **Subcategorization-Type**, **Semantic-Relation**, and **Word**, were given random representations, so that the burden of learning the task fell solely on INSOMNet. The pointer components, **Handle** and **Argument**, developed their own representations as described in Section 3.4.2.

4.2.3 Model Comparison

INSOMNet and the conditional log-linear model are very different models with very different goals. Nevertheless, an effort was made to compare their performance on the same dataset on similar tasks. In the course of generating the results, a number of important differences between the two models stood out. These differences are listed here:

1. The statistical model uses an *explicit* grammar; INSOMNet must learn to induce that grammar through training.

2. The grammar used in the statistical model often licenses more than one analysis for a sentence, and the task of the statistical model is to select the preferred parse based on calculating the probabilities that a given parse is best according to its feature set. INSOMNet has no explicit grammar, but rather learns to activate the relevant frames for a particular input sequence, and suppress those that do not contribute to the interpretation. For this reason, analysis of the activated frames is required to determine the network's preferred parse.

3. Features on which to condition probabilities must be selected for the statistical model; INSOMNet must induce such features from the training data.

4. The statistical model does batch processing. Sentences are treated as atomic. INSOMNet does incremental nonmonotonic processing.

4.3 INSOMNet Evaluation

We evaluated INSOMNet on the elementary dataset using tenfold cross-validation. In this section, we consider two basic desiderata that exemplify the endpoints of the language understanding/engineering continuum:

Comprehension: how is the correct semantic information *implicitly* represented in INSOMNet? Most connectionist research on cognitive modeling has focused on evaluating comprehension using statistical analysis of a SRN's hidden layer on prediction tasks or using query networks based on the Gestalt model of St. John and McClelland (1988, 1990) to show that subsymbolic models can model human performance such as grammaticality judgments and semantic biases.

Parsing: how is the correct semantic information *explicitly* represented in INSOMNet? The standard approach to connectionist parsing has relied on building structured syntactic representations of the linguistic relationships among sentence constituents using RAAM or shallow semantic representations with fixed case-role frames.

With respect to these desiderata, we evaluated INSOMNet's performance according to four criteria, beginning with the most lenient and graduating to the most severe. The loosely modular network architecture of INSOMNet makes such a graded evaluation possible and informative. Because the Semantic Frame Encoder/Decoder component is primarily responsible for comprehension and the Frame Map Modulator component is responsible for the frame selection that underlies parsing, these components operate together to yield a semantic interpretation of a sentence as it is processed incrementally. In this and the next two chapters, we concentrate on the semantic representation at the end of sentence processing, which is typically the focus of language engineering tasks such as parsing; Chapter 7 demonstrates that INSOMNet maintains cognitive plausibility by evaluating the network's performance during sentence processing.

The four evaluation criteria are designed to measure INSOMNet's performance in terms of the Semantic Frame Encoder/Decoder and Frame Map Modulator components. Parsing accuracy is evaluated with respect to the correct activation of units in the Modulator Map that correspond to the Frame Map Nodes encoding the target set of MRS frames in the semantic representation of an input sentence. Given this target set, comprehension was evaluated on the basis of the accuracy of the decoded Frame Map Node patterns. A crucial aspect of the inter-operation of these network components necessarily rests on the accuracy of the pointer fields in the decoded MRS frames. The details of how these pointer fields are interpreted in the context of each criterion will be taken up in the subsections describing each evaluation result, but an overview of the criteria is provided below for perspective:

Basic Comprehension and Parsing: comprehension and parsing are evaluated separately. Comprehension is measured on a per-field basis in terms of how well the MRS frame components are represented in the Semantic Frame Encoder/Decoder network. Pointer accuracy in the comprehension evaluation is only measured with respect to frames in the target set.

Parsing is evaluated on a per-frame basis in terms of the whether the correct frames are selected by the Frame Map Modulator network.

Exact Pointer Match: evaluation of comprehension and parsing are combined to provide a measure on a per-field basis of how well the network components work together. Pointer accuracy is measured in terms of the entire Frame Map, regardless of whether the frames pointed to are in the target set or not.

Exact Frame Match: Comprehension and parsing are both evaluated on a per-frame basis. Any error in frame decoding constitutes an error for the entire frame. Similarly, parsing is evaluated in terms of how many target frames are decoded correctly, and non-target frames are discounted from the score.

Exact Parse Match: Comprehension and parsing are both evaluated on a per-sentence basis. Any error in any frame decoding constitutes an error for the entire frameset. Accordingly, a sentence and its interpretation is given credit if and only if it is decoded completely correctly, all target frames are above threshold, and all non-target frames are below threshold.

Up to this point, we have tacitly treated frame selection as a binary process. Yet, as we have discussed in Section 3.3.3 and further elaborated on in Section 4.2, INSOMNet actually uses graded frame selection to model cognitive effects such as nonmonotonicity. In order to take this gradience into account in the following sections, each of the performance criteria are given with respect to a *frame node activation threshold* between 0.0 and 1.0. This threshold indicates the minimum activation strength a unit on the Frame Map Modulator must have for the corresponding Frame Node to be considered selected and, therefore, part of the network's semantic interpretation. Consequently, for the comprehension evaluations under each of the four criteria, we report the accuracy of decoded frames that reach threshold, while the parsing performance is provided by precision/recall curves defined in terms of frame node activation strength. *Accuracy* is simply the proportion of all decoded fields (t) that are correct (c):

$$A = \frac{c}{t}$$

Precision and recall are another two common measures used in computational linguistics to gauge how well a system is able to distinguish target items from non-target items. Thus, *true positives (tp)* are the target items the system selects; *false positives (fp)* are the non-target items selected; and *false negatives (fn)* are the target items the system fails to select. The *true negatives (tn)* are not often taken into account because they typically constitute the largest proportion of all items and, consequently, do not contribute much information to evaluation. In this framework, *precision* (P) is the proportion of items selected by a system that are also in the target set:

$$P = \frac{tp}{tp + fp}$$

and *recall* (R) is the proportion of target items actually selected:

$$R = \frac{tp}{tp + fn}$$

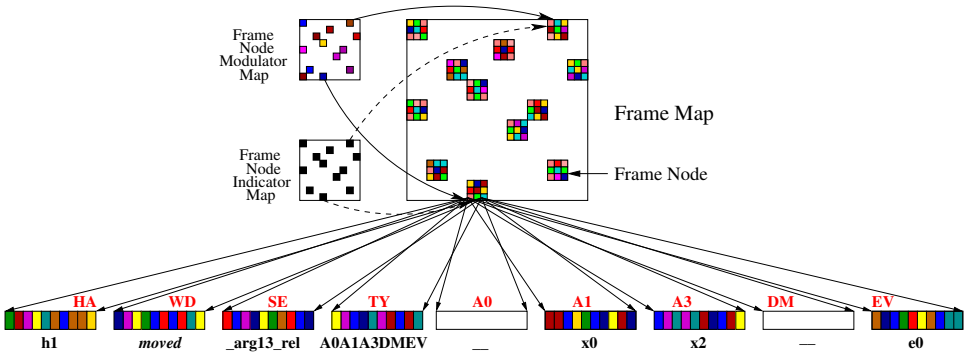


Figure 4.3: **Evaluation of Comprehension and Parsing.** (***Color figure***) Evaluation of INSOMNet's comprehension is based on how well the Frame Nodes are decoded into their proper components. For example, in this figure, all but the word *moved* and one argument handle **x2** are decoded correctly (cf. Figure 3.6), so comprehension for this frame would be $\frac{8}{10}$, or 80%. Parsing accuracy is evaluated with respect to how the units in the Frame Node Modulator Map correspond to those in the Frame Node Indicator Map above a given level of activation. Three target units (in blue, which are units 0, 64, and 75) are below a threshold of 0.5, and one unit in the lower left corner (unit 72) is a non-target unit.

These scores are typically combined into a score for overall performance called the *F-measure* (F), defined as:

$$F = \frac{2PR}{P+R}$$

assuming that precision and recall are weighted equally.

Taking frame node activation threshold into account means not only that a field must be decoded correctly, but also that frame which contains the field must be activated above threshold. The frame node activation threshold has the largest effect on the pointer components. The reason is that, in addition to the two criteria just mentioned, these pointers must also satisfy one more criterion to be counted: the frame pointed to must also be activated above threshold.

As an example, we will apply these criteria to the map modules in Figure 4.3, assuming that unit 0 at the top left corner and unit 26 near the top right of the Frame Map each has five fields, units 7 and 13 have six, unit 75 has nine fields, and all other compressed frames encode eight fields. The total number of fields across all target frames, then, is 96. Of these, we will also assume that only two, *hit* and *x1*, in the illustrated frame, are decoded incorrectly as *moved* and *x2*, respectively.

Basic Comprehension and Parsing: Comprehension and parsing are evaluated separately. Because there are two fields that are decoded incorrectly, comprehension yields a score of $\frac{94}{96} = 96.9\%$.

Parsing with a threshold of 0.5 yields a precision of $\frac{9}{9+3} = 75\%$, because nine of the target units are above threshold, while three target units are below. Recall is $\frac{9}{9+1} = 90\%$ since only one non-target unit is above threshold.

Exact Pointer Match: For the sake of argument, we will say that three decoded arguments cor-

respond to those units that are below threshold in Figure 4.3; one of these is the handle **h1**, which fills in the **SA** role of the top **prprstn_rel** frame. Because the nodes these arguments point to are below threshold, they have to be discounted under the exact pointer match criterion. Moreover, all of the fields in the frames indicated by those three units that are below threshold must also be discounted; assume there are 20 such fields (including the nine that have been expanded in Figure 4.3). Thus, comprehension accuracy would be $\frac{73}{96} = 76.04\%$ because

- Units 0, 64, and 75 are below threshold, so all of their fields are discounted, for a total of 20 (including the original two decoding errors); and
- The three arguments in other frames which point to them are also discounted (assuming they are not already included in the above count);

Recall is then $\frac{73}{73+23} = 76.04\%$, and precision is $\frac{73}{73+7} = 91\%$, when the seven fields for the non-target unit that is above activation threshold are taken into account.

Exact Frame Match: In addition to the fields that have to be discounted because they are below threshold, those frames having an argument pointing to a frame below threshold are also discounted (rather than just the one argument, as in the Exact Pointer Match above). Applying this criterion to the the three frames that have pointers to units 0, 64, and 75, and assuming those three frames account for 24 fields, the accuracy and recall is $\frac{52}{96} = 54.17\%$ and precision is $\frac{52}{52+7} = 88.13\%$.

Exact Parse Match: If any field is decoded incorrectly, the complete frameset is discounted. Thus, in this case, the original two errors described above result in accuracy, precision, and recall of 0.0.

For the experiments conducted below, INSOMNet was trained with an initial learning rate of 0.01 and the Frame Node Indicator Map given an initial learning rate of 0.4. The neighborhood was initialized to half the diameter of the Frame Map (6). The learning rate of the Frame Node Indicator Map was decayed by 0.9 and the neighborhood decremented according to the schedule

$$\text{epoch}_{i+1} = 1.5 * \text{epoch}_i + 1$$

where i indexes each parameter update. Once the Frame Node Indicator Map had stopped self-organizing when its learning rate fell below 0.001, the learning rate for INSOMNet was decayed by 0.5 according to the same schedule.

4.3.1 Basic Comprehension and Parsing Performance

In this first evaluation, we measure comprehension and parsing separately in order to establish upper bounds on INSOMNet's performance with respect to each task. Thus, we evaluate comprehension

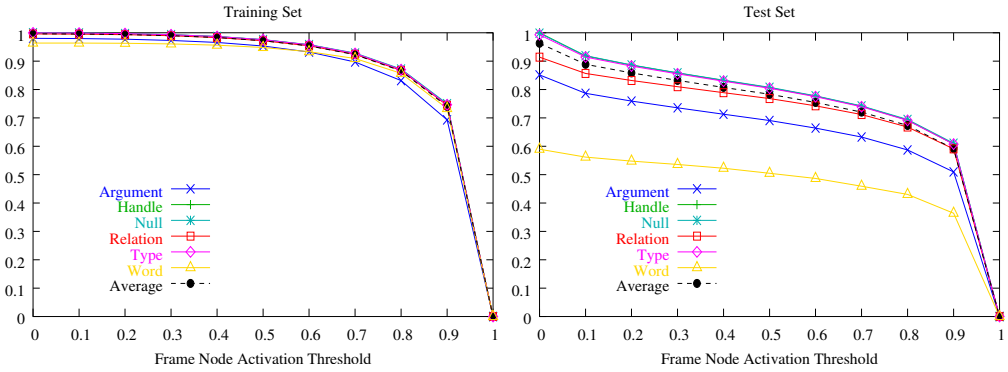


Figure 4.4: **Basic Comprehension Performance.** (***Color figure***) This figure gives a breakout of the performance of INSOMNet's comprehension on both the training and test sets. The curves represent the accuracy of the six MRS components (**Argument**, **Handle**, **Null**, **Semantic-Relation**, **Subcategorization-Type**, and **Word**) that were used to evaluate its performance across experiments, together with their **Average** (dashed line with filled circles). This breakout represents the best the network can perform on the dataset at each threshold level. Although INSOMNet learns the training set very well, it does lose some generalization on the test set, primarily due to the Frame Map Modulator network not having learned to activate the correct frames above threshold. The **Word** component shows the worst performance because it is the component most affected by data sparsity.

on a per-field basis apart from parsing (frame selection) by measuring INSOMNet's ability to accurately decode the patterns in the Frame Map on the assumption that only the target frames have been selected (activated above threshold), as would be the case if parsing were perfect. Similarly, we evaluate parsing on a per-frame basis apart from comprehension by measuring how well INSOMNet is able to select only the nodes in the target frame set without regard to decoding.

Figure 4.4 shows separate plots for the six MRS components, **Handle**, **Word**, **Semantic-Relation**, **Subcategorization-Type**, **Argument**, and **Null**, that were categorized for the type of filler they took and their role in the MRS dependency graph as described in Section 3.1.1. These components were identified during the course of developing INSOMNet in order to determine where the network was strongest and weakest. The dashed line with filled circles gives the average performance of INSOMNet on both the training and test sets measured as the proportion of fields produced correctly by the Frame Node Decoder network. In this and subsequent evaluations, performance on both training and test sets are shown to give a sense of INSOMNet's generalization ability.

The component breakout for the test set in Figure 4.4 demonstrates that INSOMNet does generalize well, but has the most difficulty with the **Word** component. Analysis of the test data suggests two reasons for this discrepancy. One reason is data sparsity: only 5.4% of frames have an argument that is filled by words from the input sentence, such as *january* or *hanover* (the *Named* and *Const_Value* slots are the only arguments in the elementary dependency graph format that do not have a pointer or **NULL** value). The other reason is related to data sparsity and is due to a technical tradeoff in the implementation. We empirically discovered that not including the **Word** field

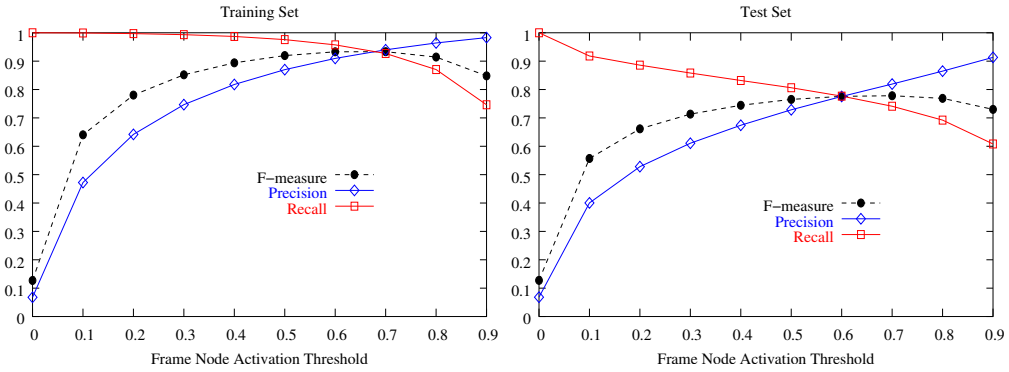


Figure 4.5: **Basic Parsing Performance.** (***Color figure***) To evaluate INSOMNet's basic parsing potential, we simply look at the the target frames above threshold, the non-target frames above threshold, and the target frames below threshold. The precision/recall curves for the training and test datasets yield a maximum F-measure of 0.77 at $x = 0.7$.

when encoding the **Handle** for a frame helped improve the accuracy of the pointer fields, whereas encoding the **Handle** with the **Word** included tended to throw these pointers off to a neighboring Frame Node, leaving the network unable to track them. Thus, not only would the **Word** be lost, but anything pointing to its frame would be lost as well.

The figure also shows the gradual decline in performance over all components as the activation threshold by which frames are considered selected increases. Recall that, to be counted, a field must both be decoded properly and the frame that contains it must be above threshold. The $x = 0$ baseline, therefore, represents the best INSOMNet can do on the test set (with all nodes activated).

Figure 4.5 shows the potential parsing performance of INSOMNet. By looking just at the activations of the units on the Modulator Map, we take those that correspond to target frames that are above threshold as true positives, those targets that failed to reach threshold as false negatives, and those corresponding to non-target frames above threshold as false positives, a maximum F-measure of 0.77 occurs at $x = 0.7$. This is possible because a very small percentage of the 144 Frame Nodes are indeed false positives. Parsing would be much more accurate if there were no distractors, but that is hard to achieve in this network due to the self-organization of the Frame Map that causes several nodes to be used for similar frames during training, and the graded frame selection retained for cognitive modeling.

These separate evaluations of comprehension and performance suggest there should be a nice way to combine their accuracies. This task is left for the following evaluation.

4.3.2 Exact Pointer Match Criterion

In combining comprehension and parsing performance, the pointer components, **Handle** and **Argument**, are evaluated in the context of the entire Frame Map. The result is that these components are more likely to be confounded by addresses of non-target Frame Nodes. The consequent overall

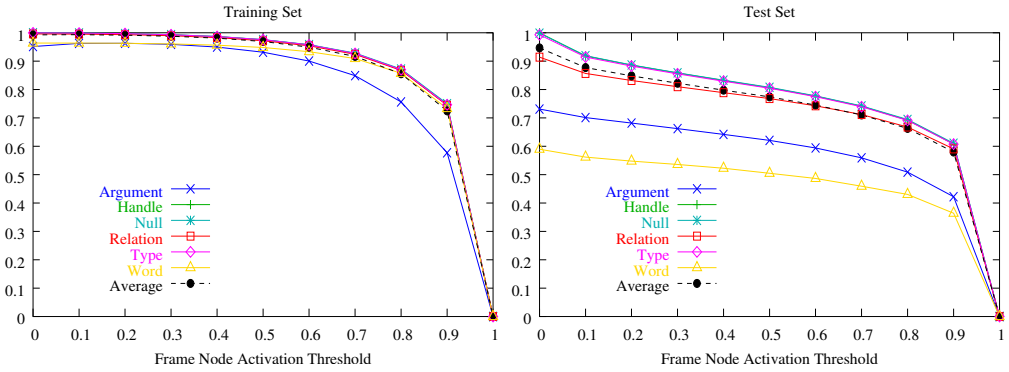


Figure 4.6: **Comprehension and Parsing under the Exact Pointer Match Criterion.** (***Color figure***) Under the exact pointer match criterion, there is somewhat more loss in generalization because now the pointer fields can be confounded by false positives, as is evident in the **Argument** component on the test set data. Note the slight upward bend in the **Argument** components on the training set. This effect is due to these pointers getting slightly more accurate as fewer false positives reach theshold, resulting in less confounding.

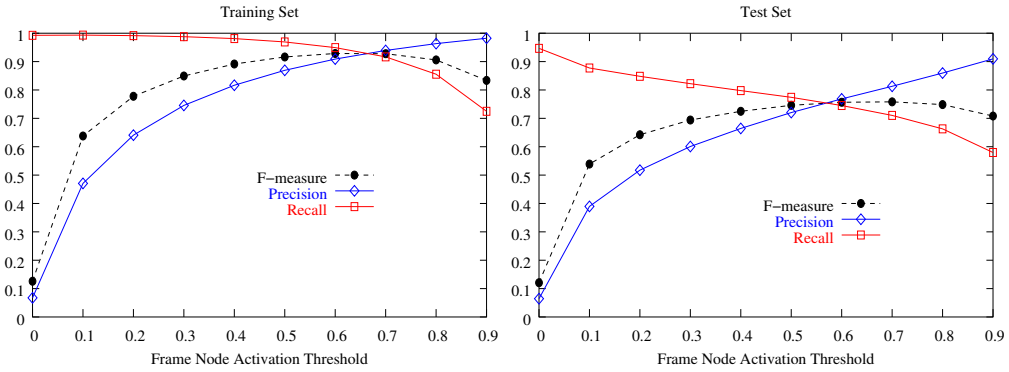


Figure 4.7: **Precision and Recall under the Exact Pointer Match Criterion.** (***Color figure***) Even under the exact pointer match criterion, there is little loss in overall parsing accuracy. Note that comprehension performance (as given in Figure 4.6) is factored into the precision and recall curves. The highest F-Measure 0.75 occurs at $x = 0.7$.

performance of the network is nevertheless only slightly worse on both the training and test sets. Moreover, there is a slight upward bend in the training curve on the **Arguments** because fewer non-target frames reach threshold, resulting in less possibility for confounding. The effect also holds in the test set, but is offset by the loss of target frames, which brings down overall performance. The only significant difference between this evaluation under the exact pointer match criterion and that in the previous evaluation in Figure 4.4 is evident in the **Argument** curve, which clearly shows the effects of self-organization on the encoded handles. Keeping the handles distinct enough to distinguish between Frame Nodes, yet similar enough to permit meaningful self-organization, is a delicate process that requires more research.

Under the exact pointer match criterion, precision and recall is measured with respect to all fields in the Frame Map. That is, we are not looking at just whether the correct frames are selected or not, as in the basic parsing evaluation in the previous section, but also factoring in comprehension. However, a penalty needs to be assessed for non-target Frame Nodes that are activated above threshold. For simplicity, we assign them the ratio of the total count of all fields to the total number of frames, which comes out to almost exactly seven fields[6]. In this evaluation, the true positives are all fields produced correctly (whose containing frames are above threshold), the false negatives, those that are either decoded incorrectly or whose frames are below threshold, and false positives those presumed seven fields in all non-target frames that are above threshold. The resulting curves are virtually identical to the curves in Figure 4.5, again because of the relative lack of false frames above threshold. The main impact again is the loss of accuracy due to false negatives and the relatively few arguments that are thrown off by the false positives. In calculating the precision and recall, we factor in the performance at each threshold, yielding a maximum F-Measure of 0.75 at the $x = 0.7$ activation level. This value is only minimally below the 0.77 given in Figure 4.5.

4.3.3 Exact Frame Match Criterion

In order to evaluate the network's performance even more strictly, the *exact frame match* criterion is usedto meausre both comprehension and parsing on a per-frame basis by only counting a frame if it is above threshold and all of its fields are decoded correctly. Accordingly, as with the basic parsing criterion, we again limit our consideration to target Frame Nodes, since we are only counting a frame as a true positive on the basis of its threshold and completely accurate decoding. If any field is incorrect or the frame itself is below threshold, it is regarded as a false negative. The false positives are the other non-target nodes that are above threshold, which we conservatively assume to be decoded correctly for lack of a metric to evaluate their fields.

The impact of the exact frame match criterion is quite visible in the test set in Figure 4.8. While the training set is virtually indistinguishable from that in Figure 4.6, the test set reveals the primary impact of the **Argument** component's inaccuracy. The dataset format results in quite a few frames having very similar representations. These representations end up occupying adjacent nodes on the Frame Map and often confound the **Argument** pointer. An error in the **Argument** field results in the entire frame being discounted, bringing down overall performance.

INSOMNet's performance under the exact frame match criterion reveals the extent of the inaccuracy of the **Argument** component. At its maximum at $x = 0.7$, the F-Measure only reaches 0.64. If the inaccuracy were limited to just the Modulator Map not activating targets above threshold, the performance should be similar to that in the previous two evaluations. However, almost every frame has one or more **Argument** fillers, so inaccuracies in these fields will have a wider impact, which is evident in Figures 4.8 and 4.9.

[6]The actual ratio on both training and test sets is 6.99.

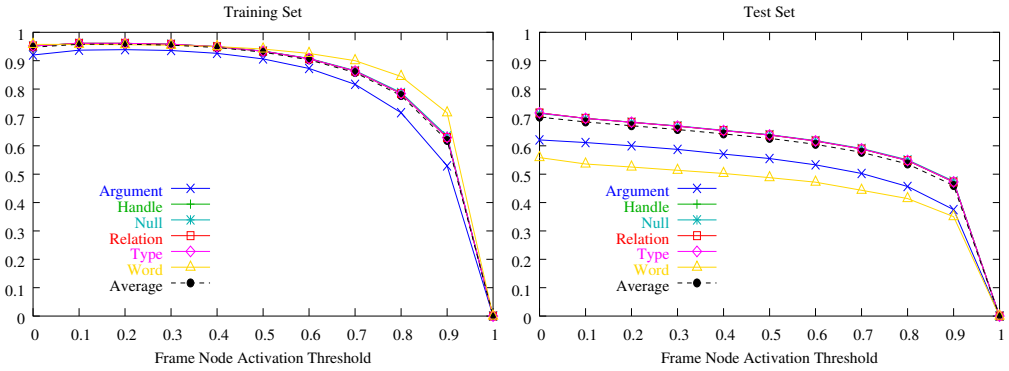


Figure 4.8: **Comprehension and Parsing under the Exact Frame Match Criterion.** (***Color figure***) Under the exact frame match criterion, only those frames above threshold whose every field is correct are counted. The training set performance is virtually identical to that for the exact pointer match criterion, but there is a marked decrease in generalization in the test set because of the loss of relevant frames that did not reach threshold due to a mismatch in at least one field. Most of the mismatches were due to the inaccuracy of the **Argument** component, which is thrown off by false positives, as well as by true positives within the target frameset.

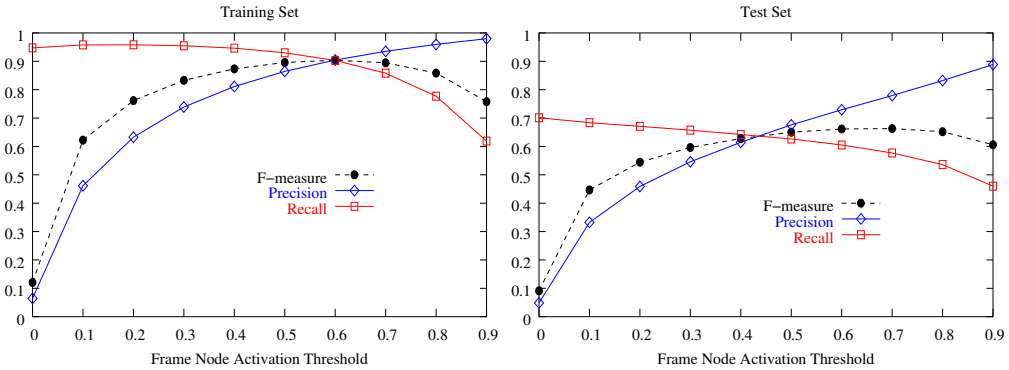


Figure 4.9: **Precision and Recall using the Exact Frame Match Criterion.** (***Color figure***) While the precision/recall curves are still good in the training set, the difficulty of the exact match criterion is apparent in the test set. With no partial credit possible for frames above threshold, these frames become false negatives, bringing down overall performance. The false positives, while few, are enough to throw off some argument fields, also adding to the increase in false negatives by contributing to a false match. The F-measure is maximum (0.64) at $x = 0.7$. As before, the performance at each activation level in Figure 4.8 has been factored into the precision/recall calculations.

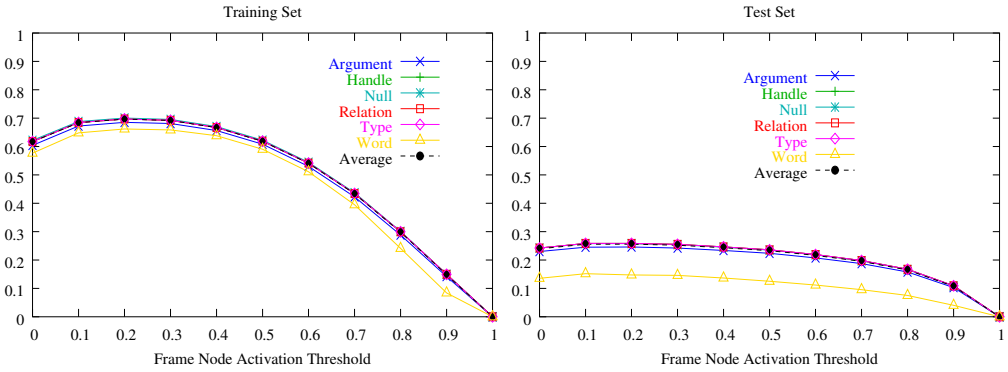


Figure 4.10: **Comprehension and Parsing under the Exact Parse Match Criterion.** (***Color figure***) Under the Exact Parse Match Criterion, all frames are discounted if any frame is below threshold or decoded incorrectly. The impact of the severe test is very evident in this figure. The upward bend that comes with fewer false positives reaching activation threshold is also evident on both training and test data.

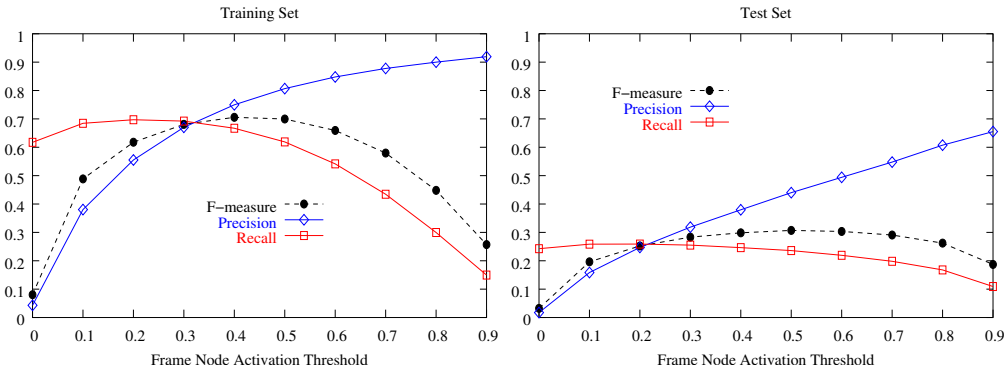


Figure 4.11: **Precision and Recall using the Exact Parse Match Criterion.** (***Color figure***) The Exact Parse Match Criterion results in the elimination of many frames due to one or more errors that those frame may have, and the interaction of the parsing criterion. The F-measure reaches a maximum of 0.31 at $x = 0.5$. As before, the performance at each activation level in Figure 4.10 has been factored into the precision/recall calculations.

4.3.4 Exact Parse Match Criterion

Under the Exact Parse Match Criterion, any error in decoding a field results in the complete frameset being discounted. The severity of this criterion is evident in Figure 4.10 on both the average over the training sets and test sets. A strong factor contributing to this performance is the method for discounting argument fields. If the frame denoted by an argument is below threshold, the result is that the entire frame containing the argument is discounted, and, accordingly, the entire frameset is lost.

Similarly, Figure 4.11 shows the impact of the Exact Parse Match Criterion on the precision and recall curves for both training and test sets using tenfold cross-validation. The overriding reason

for the poor performance measures under the Exact Parse Match Criterion is the graded nature of the Frame Selector. Most sentences yield parses that have at least one non-target frame above threshold or one target frame below threshold.

4.4 Error Analysis

4.4.1 Conditional Log-Linear Model

At the time of the June 2001 release of the Redwoods Treebank, the project was still in an early phase and there was a single Stanford undergraduate who had been trained as treebank annotator on the use of the [Incr TSDB()] Tool to make parse selections (see Section 2.5.3). Consequently, there were errors in the treebank annotation, which Toutanova et al. (2002) estimate constituted about 30% of the overall misparses in their analysis. These annotation errors limited accuracy by both counting correct parses as wrong and causing poorer training as the model tried to learn the misannotations. Another 10% were attributed to limitations in the English Resource Grammar. They also estimate that 10% of the parses counted as incorrect had more than one plausible analysis, requiring more context to yield a reliable disambiguated parse. The remaining half of the mismatches were deemed errors that could be corrected with further improvements in the statistical model. As of this writing, many of the errors in annotation and the grammar have been corrected in the current release (October 2002) of the Redwoods Treebank.

Among the actual errors in the test set analyzed, the majority were due to prepositional phrase attachment and lexical category mismatches. The model does show a strong preference for low attachment, which corresponds with numerous psycholinguistics studies, but will occasionally make the opposite choice.

4.4.2 INSOMNet

We have already alluded to the primary source of error in the test sets in the previous sections using the four evaluation criteria. More than half the errors are due to inaccuracy in the **Argument** component, but this is not surprising since this field occurs multiple times in the majority of frames. The combination of training a RAAM network to encode handles and using those representations to denote the Frame Node that is to be decoded to yield the complete frame (whose arguments can also use the handles as pointers) introduces a fair amount of complexity into training the network accurately. Many erroneous arguments point to Frame Nodes that decode into almost exactly the same frame, but for one field[7]. To get an idea of how close the misses are, consider that roughly

[7]Thus, we could trade off one of our six components for the argument, but we have chosen to assign blame to the **Argument** component to simplify the analysis. Finding a way to use the **Argument**'s preference, rather than stipulating *a priori* its target via encoding with RAAM, is research still in progress.

35% of the **Argument**s pointed to neighboring nodes[8]. Because of the clustering property of the SOM underlying the Frame Map, most of these nodes would have been viable targets had they been trained. Yet, training those nodes in addition would probably increase the number of false positives. It is clear in any case that improving the accuracy of the **Argument** component will improve the network's performance across the board.

4.5 Conclusion

In this chapter, we have evaluated INSOMNet on the same dataset as used to motivate the application of a conditional log-linear model on the parse disambiguation task. INSOMNet does not have access to a grammar, but must rather approximate the grammar implicit in the semantic annotations of sentences. Using four criteria designed to show how the components of INSOMNet work together to produce a semantic interpretation of a sentence, we have demonstrated that the subsymbolic approach can be scaled up beyond toy grammars to parse sentences with realistic complexity.

[8] It should be kept in mind that the semantic clusters are not spread evenly over the map, so similar nodes could be more than one node away from each other.

Chapter 5

Performance Evaluation on Full MRS

The comparison study of the last chapter provided a useful baseline with which to gauge INSOMNet's performance against the best statistical model that has been evaluated on the Redwoods Treebank. Yet, that research is still in an early phase, and the conditional log-linear model has only been evaluated on the elementary dependency graphs described in Chapter 4.1. INSOMNet was originally designed to deal with the full MRS dependency graphs, which include a host of features that are not part of the elementary semantic dependency graphs. The task taken up in this chapter is to evaluate how well INSOMNet can scale up to the more detailed representation of the complete MRS structure.

5.1 CSLI Detailed Dataset

The dataset (hereafter, the *detailed dataset*) used in this chapter is a corpus of 4817 sentences from the Redwoods Treebank (Version June 20, 2001) for which at least one analysis was selected as correct by the treebank annotator, although 168 had more than one preferred analysis, and one had as many as seven analyses. These ambiguous sentences were kept in order to determine if INSOMNet could learn to maintain the various interpretations. Duplicate sentences with the same interpretation, such as the ubiquitous *let us see*, were removed. The dataset contains the full MRS representation, as provided through the [Incr TSDB()] Tool for each sentence. The detailed dataset and the elementary dataset share 4367 sentences in common, but the MRS annotations are quite dissimilar, as they reflect different MRS formats. The comparison study dataset is from a slightly earlier version (June 10, 2001) of the Redwoods initiative, and only provides the elementary dependency graph representation for each sentence (see Chapter 4 for a full description).

We will first describe the extent of the differences between the full MRS and elementary dependency graph descriptions before giving the results of the experiments that were run with the more detailed data.

Figure 5.1 illustrates the differences between (a) the elementary semantic dependency graph

we can go until say one or two .

(a) Elementary semantic dependency graph

h0	—	**prpstn_rel**	**SA**	**e0**	—	—	—	—	—	
e0	—	**_can_rel**	**A0A4DMEV**	—	**e2**	—	—	—	—	
e1	—	**_until_rel**	**A0A3DMEV**	**e2**	**e3**	—	—	—	—	
e2	—	**_go_rel**	**A0A1DMEV**	—	**x0**	—	—	—	—	
e3	—	**_or_rel**	**CALHLIRHRI**	**e2**	—	**x1**	—	**x1**	—	
h1	—	**def_rel**	**BVDMRESC**	**x0**	—	—	—	—	—	
x0	—	**pron_rel**	**IX**	—	—	—	—	—	—	
h2	—	**def_rel**	**BVDMRESC**	**x1**	—	—	—	—	—	
x1	—	**numbered_hour_rel**	**APDMHRIXMI**	—	—	—	**x1**	—	—	

(b) Full semantic dependency graph

h0	—	**prpstn_rel**	**SA**	**h1**	—	—	—	—	—	
h1	*can*	**arg4_event_rel**	**A0A4DMEV**	—	**h2**	—	**e0**	—	—	
e0	—	**EV**	**DVASMOTN**	bool	−asp	ind	pres	—	—	
h2	*go*	**arg1_rel**	**A0A1DMEV**	—	**x0**	—	**e1**	—	—	
h2	*or*	**conj_rel**	**CALHLIRHRI**	**e2**	—	**x1**	—	**x2**	—	
h2	*say*	**degree_rel**	**DGDM**	**d0**	—	—	—	—	—	
h3	—	**def_rel**	**BVDMRESC**	**x0**	—	**h6**	—	—	—	
h6	*we*	**pron_rel**	**IX**	**x0**	—	—	—	—	—	
x0	—	**FRI**	**DVGNPNPT**	−	gen	1pl	std1pl	—	—	
h2	*until*	**independent_rel**	**A0A3DMEV**	**e1**	**e2**	—	—	—	—	
e1	—	**EV**	**DVASMOTN**	bool	−asp	mood	tns	—	—	
e2	—	**CRI**	**DV**	bool	—	—	—	—	—	
h4	—	**def_rel**	**BVDMRESC**	**x1**	—	**h2**	—	—	—	
h2	*one*	**numbered_hour_rel**	**APDMHRIXMI**	—	**d0**	*one*	**x1**	—	—	
x1	—	**FRI**	**DVGNPNPT**	bool	neu	3sg	prn	—	—	
h5	—	**def_rel**	**BVDMRESC**	**x2**	—	**h2**	—	—	—	
h2	*two*	**numbered_hour_rel**	**APDMHRIXMI**	—	**d0**	*two*	**x2**	—	—	
x2	—	**FRI**	**DVGNPNPT**	bool	neu	3sg	prn	—	—	
d0	—	**DI**	**DV**	bool	—	—	—	—	—	

Figure 5.1: **Dataset Comparison.** This figure shows an example of the differences between (a) the elementary semantic dependency graph and (b) the full semantic dependency graph representation for the sentence *we can go until say one or two*. The full graph contains more than twice as many frames, including modifiers (such as the frame for the word *say*), extra feature information like gender, person and number for nominals (nouns and pronouns: these frames have handles that begin with **x**; mood, tense, and aspect for verbal elements (verbs and adjectives: these frames have handles that begin with **e**), as well as basic feature information for other categories (such as adverbs: in the full semantic dependency graph, the degree modifier *say* takes the feature frame whose handle begins with **d**, which tells us its divisibility feature is generic). The full graph also has more argument roles filled in (e.g., the additional *restriction* (**RE**) in the **def_rel** frames) and uses handle-sharing (e.g., the frames containing the words *go*, *or*, *say*, *until*, *one*, and *two* all share the handle **h2**, signifying that they are treated as adjuncts). As this comparison makes clear, the full MRS dependency representation is considerably more complex than the elementary MRS dependency graphs used in Chapter 4.

sense count	1	2	3	4	5	6	7	8
word count	845	124	51	19	5	1	1	2

Figure 5.2: **Sense statistics.** This table shows the degree of semantic ambiguity of words in the scale-up dataset. Slightly more than 80% of the words were unambiguous, whereas only four had more than five senses.

used in the comparison study of Chapter 4 and (b) the full semantic dependency graph used in the current scale-up study for the sentence *we can go until say one or two*. There are several noteworthy additions that are apparent from just a cursory look at the figure. First, the **Word** field is used in the detailed dataset, whereas it was unpopulated in the comparison version. This field is the input word in the sentence that corresponds to each frame, wherever we could make this match. We will explain the motivation for doing so in the next paragraph. Second, the full MRS semantic dependency graph includes several new frames that contain the features for those semantic relations that have features, such as the **e0** frame in Figure 5.1 that carries the divisibility, aspect, mood and tense features of the modal *can* in the **h1** frame. Recall that these feature frames serve as the leaves when constructing the frame handles, as described in Section 3.4.2. Finally, handles may be shared to indicate modification or attachment, as with the **h2** handle in Figure 5.1 that occurs in the frames for the input words *go*, *or*, *say*, *until*, *one*, and *two*, as described in Section 3.1.1. There was no handle-sharing in the elementary dataset.

The original version of the detailed dataset had 1177 semantic relations, slightly more than the 1106 relations in the comparison dataset. Yet, in most cases, the semantic relation could be distinguished by the word embedded in its definition (e.g. *or* in **_or_rel**). Table 5.2 shows the number of lexical distinctions in the detailed data along with the number of words having that many senses. For example, 845 (80.2% of the total) words were associated with only one semantic relation in the corpus, as in the *_or_rel* example above. Only two words had eight senses: *get* and *like*. Additionally, there were 68 distinct semantic relations for which there was no corresponding word in the sentence, such as *prpstn_rel* and *unspec_loc_rel*.

This detachment between the word as it appears in the input sentence and the semantic relation means that the network must learn to associate the correct semantic relations with the words in the input, as was the case in Chapter 4. Although INSOMNet was able to learn this association reasonably well, we found that the network's performance improved slightly when the word was also provided as a target in the output explicitly during training, even though doing so introduced 961 new targets[1] (including our six morphological suffixes). The reason is that, in the majority

[1] Actually, we introduced 1054 tokens as targets for the **Word** slot, but 93 of them were compounds such as **how_about** that corresponded to two or more words in the input sentence, still requiring the network to make an arbitrary association since these compounds were also initialized with random representations.

of cases, the word uniquely identified the semantic relation, so there was no extra burden on the network to learn another arbitrary mapping. Nevertheless, there still remained many cases where a single input word had several senses that were distinguished by the semantic relations in the ERG. Rather than retain the redundancy of using the words and original semantic relations together to maintain the proper lexical distinctions, we elected to use the ERG to separate out the word and abstract the semantic relation up one level of its type hierarchy. As an example, the word *say* in Figure 5.1 occurs with four distinct semantic relations in the detailed dataset, but can be identified with four common relations:

_say_approx_rel, as in the sentence *we can go until say one or two*, becomes *say* and **degree_rel**;

_say_rel, as in *a moment ago you say -d five o'clock* becomes *say* and **arg123_rel**;

_say_h_rel, as in *but did you say you were free this friday* becomes *say* and **arg134_rel**;

_say_loc_rel as in *at two o'clock on the twenty eighth we say -d* becomes *say* and **arg12_rel**.

This process yielded 104 abstract relations, 71 of which were already in the original dataset (such as **numbered_hour_rel**). The remaining 33 new relations made distinctions between verb types (they often–though not always–reflected the subcategorization type; e.g., **_go_rel** is an **arg1_rel**, reflecting the **A0A1DMEV** subcategorization type for an intransitive verb), some adjectives (such as those that can take extraposed "it"), adverbs, prepositions, and, particularly, nouns (many of which were divided among temporal distinctions, such as **day_part** and **non_day_diadic_modable_rel**). This pairing of word and abstract relation preserved the uniqueness of the original semantic relation, while at the same time allowing some generalization[2]. Another motivation for the addition of the word field was to allow for the option of using phonological or orthographic encodings in both the input and output of the system (such as in research toward dealing with the out-of-vocabulary problem that will be taken up in future work).

Further comparing the annotation schemes of the two datasets, note that there are additional arguments on some frames such as including the *restriction* of determiners (see the **def_rel** frames in Figure 5.1, which both specify the handle **h2** as their restriction), as well as the inclusion of some modifiers like *say* that are omitted in the elementary dependency graphs. Lastly, because the comparison dataset was designed to follow the statistical model's frameset format as closely as possible, multiple frames sharing feature information were collapsed into one whenever possible. In the detailed dataset, those distinctions are preserved (cf. the **numbered_hour_rel** frames in Figure 5.1: in the elementary semantic dependency graph format, there is just one frame with this relation, because the words *one* and *two* are not distinguished, whereas they are in the full graph, requiring two frames for their proper representation).

[2]We say "some" here because we essentially traded the sparsity problem with the semantic relations for a sparsity problem with the words.

Dataset	Frame Component							Total
	Handle	Word	Relation	Type	Argument	Feature	Null	
Basic	51827	2829	51827	51827	48644	0	155099	362053
Scale-up	77184	36521	77184	77184	89791	80504	118932	557300

Figure 5.3: **Dataset Complexity.** This table gives a breakout of the number of fields for the seven components that are evaluated on both the elementary and the detailed datasets. In all of the components except for the **Null** component, there are at least 50% more fields in the detailed dataset than in the elementary. The slightly fewer **Null** components in the detailed dataset are a result of more fields being populated, which are barely offset by the greater number of frames altogether. Because each frame has only one **Handle**, the first field also gives the total number of frames in each dataset.

Table 5.3 provides a more quantifiable comparison of the complexity of the two datasets. Overall, the detailed dataset has more than 50% more fields. The first four components appear in every frame, but the **Word** component differs most markedly because it is populated in 47.3% of the frames in the detailed dataset, whereas only 5.4% of the elementary dataset has this field filled in. Because the **Handle**, **Semantic-Relation**, and **Subcategorization-Type** are always populated, the counts in these fields equals the total number of frames in each dataset. Most importantly for the scale-up study, there are nearly twice as many **Argument** fields in the detailed dataset than in the elementary.

In short, the full MRS dependency graph is used to give as full a semantic description of the sentence as possible from the Redwoods Treebank.

5.2 Training and Experiments

We ran INSOMNet on the detailed dataset of full MRS annotations using tenfold cross-validation. INSOMNet was trained with an initial learning rate of 0.01 and the Frame Node Indicator Map given an initial learning rate of 0.4. The neighborhood was initialized to half the Frame Map's diameter. The learning rate of the Frame Node Indicator Map was decayed by 0.9 and the neighborhood decremented according to the schedule

$$\text{epoch}_{i+1} = 1.5 * \text{epoch}_i + 1$$

where i indexes each parameter update. Once the Frame Node Indicator Map had stopped self-organizing when its learning rate fell below 0.001, the learning rate for INSOMNet was decayed by 0.5 according to the same schedule.

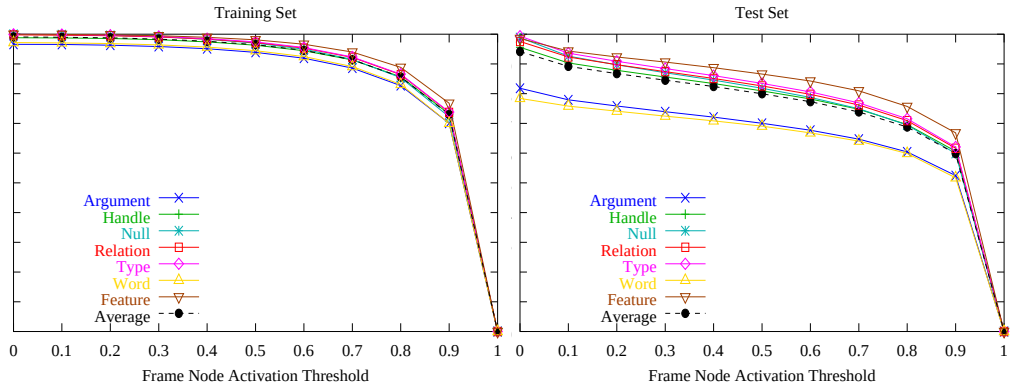


Figure 5.4: **Basic Comprehension Performance.** (***Color figure***) The curves are the seven MRS components (**Handle**, **Word**, **Semantic-Relation**, **Subcategorization-Type**, **Argument**, **Feature**, and **Null**) that were used to evaluate its performance across experiments, together with their **Average** (the dashed line with filled circles). This breakout represents the best that the network can perform on the dataset at each threshold. Generalization is good, but performance on both the **Argument** and **Word** components noticeably lag behind the other categories. The dropoff in the **Argument** performance is due to confounding within the target frameset, and the slightly poorer **Word** performance is due to data sparsity. On the other hand, the **Feature** performance is relatively high because there are far fewer features that the network has to learn.

5.3 Results

In this section, we report the same four sets of evaluations of the network's performance described in Section 4.3 for the comparison study.

5.3.1 Basic Comprehension and Parsing Performance

Recall that the first evaluation is designed to separately measure INSOMNet's comprehension ability and potential to produce the correct parse for a sentence. For the comprehension evaluation, we measure performance on a per-field basis, and ignore Frame Nodes that are not part of the target set to see how well INSOMNet can perform taking the traditional statistical/symbolic approach that the Frame Node Modulator Map only activates the target frames above threshold, and suppresses all others. The network can still be thrown off by Frame Nodes within its target set, and it sometimes is, in which case, its comprehension performance suffers. Similarly, we measure parsing performance on a per-frame basis by restricting our attention to the performance of the Frame Node Modulator Map itself in its ability to identify the target nodes, and by concentrating instead on the potential for INSOMNet to be confounded by Frame Nodes that are not part of its target set, divorced from the actual performance of the network on decoding those nodes that actually do belong.

Figure 5.4 shows separate plots for the seven MRS components, **Handle**, **Word**, **Semantic-Relation**, **Subcategorization-Type**, **Argument**, **Feature**, and **Null**, that were used in order to de-

termine INSOMNet's strengths and weaknesses. The new **Feature** category covers the arguments in the feature frames that are a part of the Full MRS format, as described in Section 5.1. Because there are only 56 static fields covered under the **Feature** category, the network tends to learn these very well. Indeed, INSOMNet seldom makes a mistake in this category; the only reason for the decline apparent in Figure 5.4 is due to the network not activating the relevant nodes on the Frame Node Modulator Map above threshold.

The component breakouts for the both the training and test sets in Figure 5.4 illustrate some of the problems the greater detail and format of the full semantic dependency graph representation causes the network. Although INSOMNet performs better on the **Word** component than it did in the comparison study, the problem with data sparsity is still apparent. Whereas, in the comparison study, the **Word**s only constituted 5.4% of the appropriate targets for the decoder, the **Word**s account for 47.3% of the their targets in the detailed dataset. Yet, many of these words only rarely, if ever, occur in the training sets in each split, so INSOMNet is unable to learn them. This difficulty is compounded by another issue related to the full MRS format. Recall that the detailed dataset adds feature frames to the frameset apart from the frames that point to them. The result is a potential disassociation from the frames that govern those feature frames[3]. As an example, the frames having the semantic relations, **_def_rel** and **arg4_event_rel** (for the word *can*) in the full semantic dependency graph of Figure 5.1 point to feature frames having the handles **x0**, **x1**, **x2**, and **e0**, respectively. INSOMNet accurately reproduces the features, but sometimes loses track of which frames point to them when they are compressed into other handles (see Section 3.4.2). Yet, fortunately this problem is not propagated further up in encoding other handles because the frames higher up in the MRS graph often include several arguments, and consequently cancel out errors by developing a unique enough representation to serve as a reliable address in the Frame Map. As is typical of RAAM, the subnetwork used to build up the structured handle representations, the similarity of the leaf feature nodes results in immediately dominating frames (those that point directly to the feature frames) confusing which frame goes with which feature frame[4].

Figure 5.5 shows the basic parsing performance of INSOMNet on a per-frame basis without regard to how well each frame is decoded. Taking as our true positives the target Frame Nodes that are above threshold as indicated by the Frame Node Modulator Map, false negatives, those that fail to reach threshold, and false positives, the non-target frames above threshold, we get an F-measure of 0.78 at the intersection of the precision/recall curves at $x = 0.7$.

5.3.2 Exact Pointer Match Criterion

With the pointer components, **Handle** and **Argument** now evaluated over the entire Frame Map, there is a greater likelihood that these fields will be confounded by the activated Frame Nodes

[3]This dissociation was not possible in the comparison study because there were no features for the words implied in the semantic relations to be disassociated from.

[4]This confusion would be equivalent, for example, to swapping subject and object in some verbs in cases.

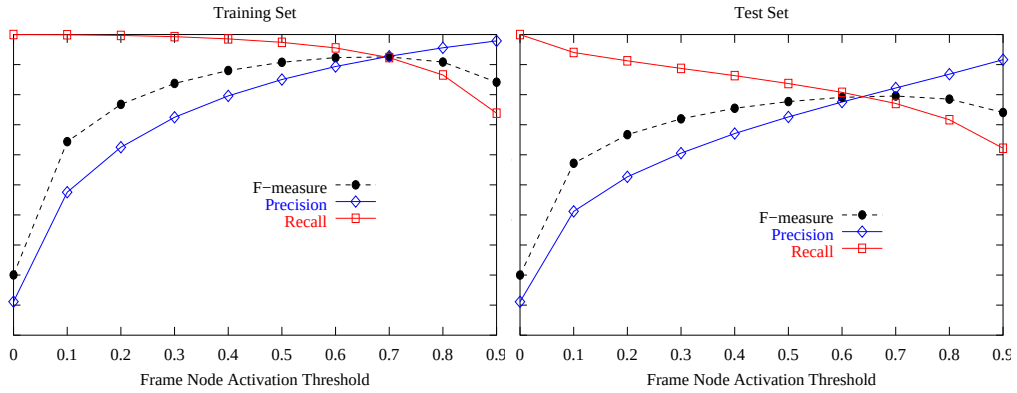


Figure 5.5: **Basic Parsing Performance.** (***Color figure***) Basic parsing performance is measured on a per-frame basis. The nodes on the Frame Node Modulator Map corresponding to the target frameset are identified, and their activations evaluated according to whether they reach threshold. The true positives are those target frames above threshold, false negatives are the target frames below threshold, and the false positives are non-target frames above threshold. The result is a maximum F-measure of 0.78 at $x = 0.7$.

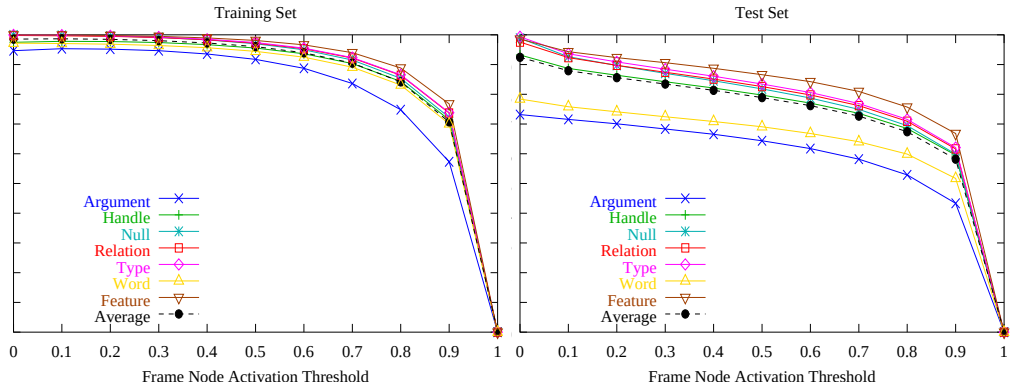


Figure 5.6: **Comprehension and Parsing using (*Color figure*) the Exact Pointer Match Criterion.** The exact pointer match criterion results in a slight decrease in **Argument** accuracy because these pointers can be confounded by other **Frame Node**s that are above threshold as indicated by the Frame Node Modulator Map.

not in the target set. Nevertheless it is a much more accurate indication of how the network would perform on a novel sentence for which we do not know its target frameset. In this evaluation, both comprehension and parsing are combined and evaluated on a per-field basis, where the true positives are all target frame fields correctly decoded and whose containing frames are above threshold, the false negatives, those target fields that are either incorrect or whose containing frames are below threshold, and false positives those fields in non-target frames that are above threshold.

The consequent overall performance of the network is only slightly worse on both the training and test sets than in the separated evaluations due to the relatively few false positives. But those

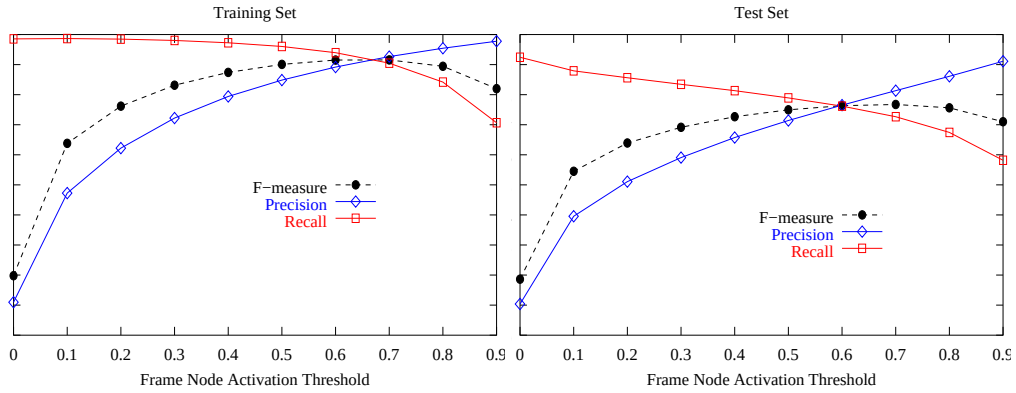


Figure 5.7: **Precision and Recall using the Exact Pointer Match Criterion.** (***Color figure***) There is little impact of the exact pointer match criterion except for a decrease in recall due to comprehension performance being factored in. The maximum F-measure is 0.76 at $x = 0.7$.

false positives that do occur have a noticeable impact on the **Argument** field. The reason is that the false positives that do reach threshold are very similar to the true positives and, so, easily confuse the network. As in the previous evaluation, the performance on the test set relative to the training set indicates that the network is generalizing relatively well. Interestingly, there is a slight upward bend in the training curve on the **Argument**s because fewer false positives reach threshold, resulting in less confounding.

Taking into account all fields in the Frame Map results in a only a slight decrease in performance, as shown in Figure 5.7. The F-Measure at $x = 0.7$ is 0.76. As in the comparison section, we take the average ratio of fields to frames[5] as the amount to discount the network's performance when it includes a false Frame Node. The resulting curves are virtually identical to the curves in Figure 5.5, again because of the relative lack of false positives above threshold. The main impact again is the loss of accuracy due to false negatives and the relatively few arguments that are thrown off by false positives and mismatches within the target frameset.

5.3.3 Exact Frame Match Criterion

INSOMNet performs relatively well when given partial credit for the fields it produces correctly. Nevertheless, it is instructive to evaluate its performance under the restriction that it produce every field correctly to get a better sense of how the network can be improved. We do so in this section by using the exact frame match criterion that only counts those frames that are above threshold and whose every field is correctly decoded.

When INSOMNet's performance is measured on a per-frame basis, there is a noticeable decline over almost all components, as is evident in Figure 5.8. Under this criterion, a target frame

[5]This ratio turns out to be 7.22 on both training and test sets.

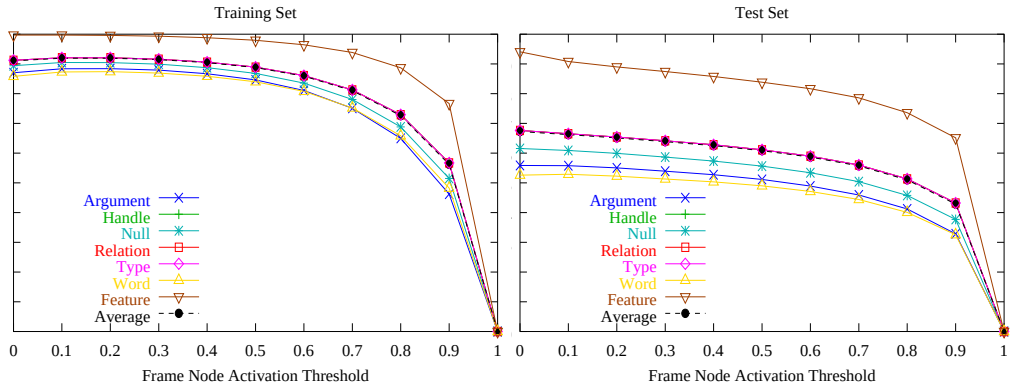


Figure 5.8: **Comprehension and Parsing using the Exact Frame Match Criterion.** (***Color figure***) Under the exact frame match criterion, we only count those frames whose every field is correct and above threshold. Here, the ability of the network to capture the **Feature** fields is evident, for these frames tend to be the most highly activated and most accurately represented. Yet, the combined impact of the missed **Word** features and the inaccuracy in the **Argument** fields due to false positive confounders and mismatches within the target set bring both the training and test set performances down considerably on all of the other features. However, there is a noticeable improvement in **Argument** accuracy in the training set due to less confounding as the activation threshold is increased. This effect does carry over to the test set, but is offset by the loss in overall performance.

is taken as a true positive on the basis of its threshold and completely accurate decoding. If any field is decoded incorrectly or the target frame itself is below threshold, it is taken as a false negative. The non-target frames above threshold are false positives, whose fields we assume to be decoded correctly. The loss in performance is the combined effect of the relatively poor performance of the **Word** and **Argument** components. Losing either of these components results in the entire frame being discounted. Recall that the best that the network could do at each threshold was shown in Figure 5.4, so the **Word** curve in that figure sets an upper bound on its performance. Similarly, the **Argument** confounding problem discussed in the previous section and evident in Figure 5.6 further impacts performance. Together, these two components bring down the performance of all components except the **Feature** field. The reason the **Feature** field is relatively immune to this effect is that only three of our added morphemes (*-s*, *-d*, and *-ing*) and two modals (*will* and *would*) occur as **Word**s in the decoded frame, and so there are no arguments to be thrown off. Consequently, the feature frames reliably show up in the target nodes, and INSOMNet tends to learn them very well.

Under the exact frame match criterion, the network has poor recall (because comprehension performance is factored in), while precision is only slightly worse than under the exact pointer match criterion. Figure 5.9 shows that the best F-measure of 0.64 is at a threshold of 0.7. Improving the accuracy of the **Argument** component would greatly improve these figures, since a frame can include up to six arguments, any one of which can be thrown off.

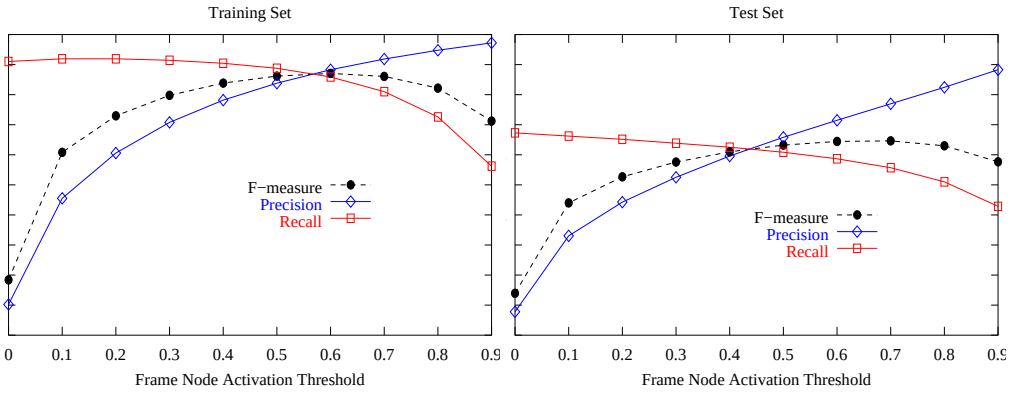


Figure 5.9: **Precision and Recall using the Exact Frame Match Criterion.** (***Color figure***) As in Figure 5.8, the combined impact of the network's poor performance on the **Word** and **Argument** fields is evident. Note that these curves factor in the comprehension performance, which results in the markedly lower graphs than in Figure 5.6. The maximum F-measure is 0.64 at $x = 0.7$.

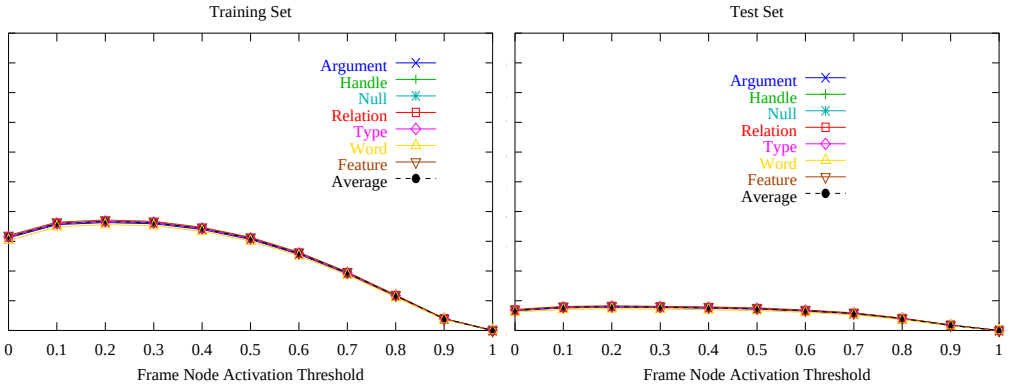


Figure 5.10: **Comprehension and Parsing under the Exact Parse Match Criterion.** (***Color figure***) Any error in decoding a frame results in the loss of the entire frameset under the Exact Parse Match Criterion. Because of the greater number of pointer fields in the scale-up dataset, there is a marked loss in performance. Even so, there is still a noticeable upward bend in the performance curves for both training and test set data.

5.3.4 Exact Parse Match Criterion

The additional complexity of the detailed dataset is apparent when the Exact Parse Match Criterion is applied. Because any error in decoding a field results in the complete frameset being discounted, the inaccuracy of the **Argument** component, as described in the previous sections, results in a significant loss on both the training and test set averages.

When the inaccuracy in the **Argument** component is coupled with the graded nature of the Frame Node Modulator Map, even one error in a frameset causes it to be discounted.

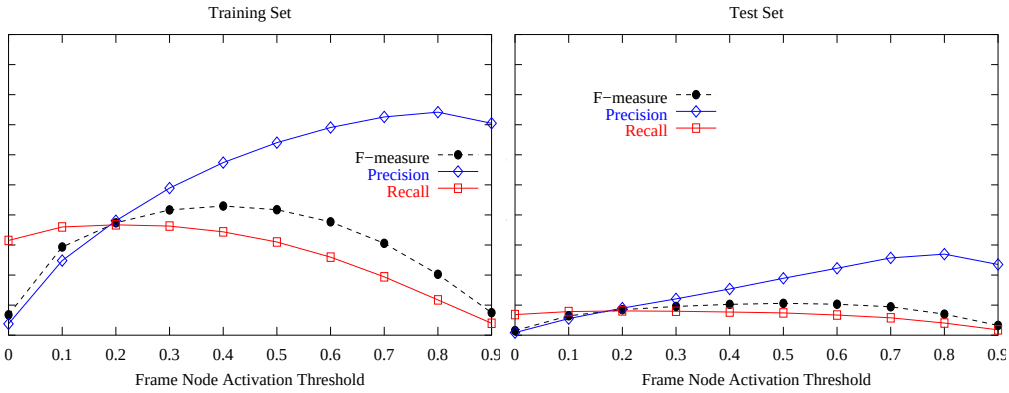


Figure 5.11: **Precision and Recall using the Exact Parse Match Criterion.** (***Color figure***) The Exact Parse Match Criterion yields a maximum F-measure of 0.11 at $x = 0.5$, factoring in the accuracy from Figure 5.10.

5.4 Discussion

The detailed dataset is a significantly more difficult dataset to learn. It is therefore surprising that INSOMNet's performance on all four evaluations is not much worse than those in Section 4.3. Further work is needed on improving the accuracy of the **Argument** and **Word** components, and reducing the number of distractor Frame Nodes that can throw off the **Argument** pointers, and, to a much less extent, the **Handle** component. Improving the accuracy of the Frame Node Modulator Map will go along way toward this goal, but runs counter to the desired behavior of INSOMNet to model such cognitive effects as semantic priming, and coactivation of multiple interpretations. Unlike the dataset in the comparison study of Chapter 4.1, the dataset used in this chapter does feature a number of sentences which had multiple interpretations. The ambiguous sentences were included for the explicit purpose of showing that the network could handle such sentences, and maintain those interpretations. This effect will be explored further in Chapter 7 on a standard benchmark on prepositional phrase attachment ambiguity to show how the network can maintain multiple interpretations and later revise those interpretations when disambiguating context is encountered during incremental processing. Because the sentences used in the Redwoods Treebank were taken from a non-psycholinguistic domain, it is difficult to use them to illustrate these phenomena. While there were ambiguous sentences included in the dataset used here, none are resolved, so the network maintains the multiple interpretations until the end of the sentence, increasing the number of purportedly "false" Frame Nodes that detract from the network's performance.

Chapter 6

Robustness Study

One of the most appealing properties of connectionist systems for the researcher interested in modeling human language use is the inherent robustness of these systems to noise and degradation. Because information is distributed across the network, it can tolerate missing units or weights, and even compensate for them by filling in partial patterns. In this chapter, we evaluate the robustness of the INSOMNet model in two ways. First, we take the original transcribed sentences from the VerbMobil project (Wahlster 2000) from which the the sentences in the Redwoods Treebank were derived and construct a dataset designed to more closely follow the actual spoken words using a condensed annotation scheme appropriate for input to INSOMNet. For each sentence, we use the MRS frameset from the corresponding sentence in the detailed dataset, and run one of the trained full-scale models from Chapter 5 on the new dataset to see how well INSOMNet is able to tolerate the added transcriptions of spoken language. The second evaluation demonstrates how robust INSOMNet is to noise added to the network weights.

6.1 Transcription of Original VerbMobil Data

The sentences that make up the dataset used in Chapter 5 were originally derived from the VerbMobil project (Wahlster 2000) and cleaned of the transcriptions to yield well-formed English sentences for the Redwoods Treebank. Those sentences that were not given at least one grammatical analysis by the ERG were not included in the Treebank. The original sentences in the Redwoods Treebank are distributed across four VerbMobil datasets (CDs 6, 13, 31, and 32). Each dataset records a number of dialogues between two people whose spoken *utterances* are divided into *turns* for each person (Alexandersson et al. 1997). The turns are hand-segmented and transcribed to give as faithful a textual representation of the audio-taped dialogue as possible.

All but CD 13 use the VerbMobil-II transcription convention developed for the second phase of the project. The VerbMobil-II is a much more detailed annotation scheme than the VerbMobil-I convention used for CD 13. The VerbMobil-II transcriptions fall into eight categories:

VerbMobil Transcription	Definition	Transcription for INSOMNet
<Smack>	human noise	#
<Swallow>	human noise	#
<Throat>	human noise	#
<Cough>	human noise	#
<Laugh>	human noise	#
<Noise>	human noise	#
<\#Squeak>	mechanical noise	#
<\#Rustle>	mechanical noise	#
<\#Knock>	mechanical noise	#
<hes>	hesitation	#
<uh>	hesitation	uh
<\"{a}h>	hesitation	uh
<uhm>	hesitation	uhm
<\"{a}hm>	hesitation	uhm
<hm>	hesitation	hm
<uh-huh>	affirmative interjection	uh huh
<mhm>	affirmative interjection	mhm
<uh-uh>	negative interjection	uh uh
<mm>	negative interjection	mm
+/X/+	repetition/correction	X
-/X/-	false start	X
<*X>	foreign word	X
~X	proper names	X
*X	neologism	X
\#X	number annotation	X
\$X	spelling out	X
[<:...>X:>]	noise interference	X
<!X>	pronunciation comment	X
<;...>	commentary	
_	articulatory interruption	
;	global comment	
<B>	breathing	
<P>	empty pause	
<*T>	technical interruption	
<*T>t	turn break	
<T_>	beggining of turn or word missing	
<_T>	end of word missing	
\%	difficult to hear	
<\%>	incomprehensible	

Figure 6.1: **VerbMobil Transcriptions.** The VerbMobil-II transcription scheme is designed to provide a faithful textual representation of all elements of a recorded dialogue. Note that X stands for elements that are preserved in the INSOMNet transciption. An empty column means that the transciption was deleted.

lexical units: words, interjections, spelling.

syntactic-semantic structure: repetitions, corrections, repairs, and punctuation that marks intonation and phrase boundaries.

nonverbal articulatory production: breathing, hesitations, and human sounds such as coughs and laughing.

noises: mechanical sounds such as clicks and microphone sounds.

pauses: breaks in the dialogue.

acoustic interference: overlapping of dialogue.

comments: pronunciation annotation for slang, dialect, contractions, and mispronunciation.

special comments: technical dialogue comments.

In order to generate the dataset of sentences that correspond to the cleaned-up sentences, we analyzed the four VerbMobil datasets to find the original sentences and translated the VerbMobil-II transcriptions into a condensed version that was consistent with the VerbMobil-I annotation of CD 13. Table 6.1 shows the VerbMobil transcription symbols, their meaning, and the simplified conventions we used for INSOMNet.

An example will help clarify the annotation process. Corresponding to the Redwoods sentence, *I am free the first through the fourth*, we found the original sentence on dataset CD 31:

```
<Laugh> <B> <uh> -/the fir=/- I am <!2 I'm> free
<uh> the #first through the #fourth .
```

The transcription labels `<Laugh>` and `<B>` are extrasentential verbal and nonverbal sounds, which we annotate as "#". we keep the false start (`-/the fir=/`), hesitation (`<uh>`), and dates (`#first` and `#fourth`), but remove the annotation symbols. We also use the pronunciation comment `<!2 I'm>` for *I am*, but render it as *i m* in the same manner we handled contractions for the elementary and detailed datasets. The resulting sentence is *# uh the fir I m free uh the first through the fourth*.

Similarly, on dataset CD 6 we found the original sentence for the Redwoods sentence *because I think I have time on each of the Mondays*:

```
<"ahm> 'cause I think I have time <;comma> on each
of the Mondays <;period> <;seos>
```

We transcribe this sentence as *uhm 'cause I think I have time on each of the Mondays* after removing the punctuation labels `<;comma>`, `<;period>`, and `<;seos>` and translating `<"ahm>` as *uhm*. We leave *'cause* as it is, even though it does not occur in any sentence INSOMNet was trained on.

This process yielded a total of 5068 sentences for the 4817 sentences in the detailed dataset. The 251 extra sentences are due to the different annotations that the counterparts for a cleaned-up sentence received on the original VerbMobil CDs. For example, each of the original annotated sentences *hm let us see*, *uh let us see*, and *# let us see # #* mapped to the Redwoods sentence *let us see* when cleaned of their annotations. Figure 6.2 shows the sentence counts together with their

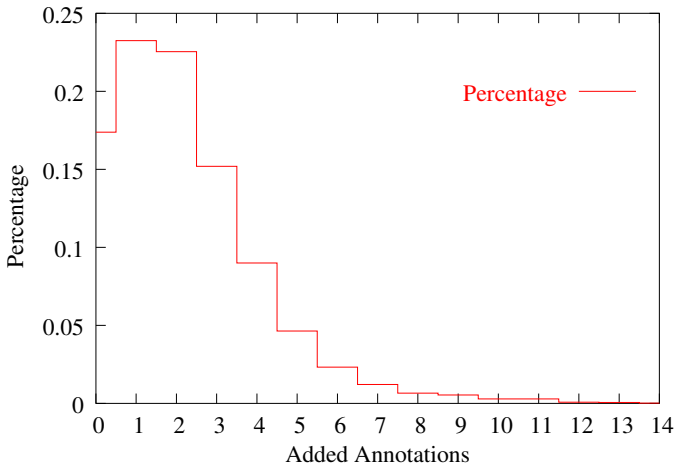


Figure 6.2: **Extra Transcription Symbols.** (***Color figure***) The x-axis denotes the difference between the length of sentences in the transcribed dataset and their cleaned-up counterparts in the Redwoods Treebank. The y-axis gives the percentage for each difference in length. Thus, $x = 0$ means a that 904 sentences, or approximately 18% of the total number of sentences in the transcribed dataset, were left unchanged. One sentence had as many as 14 extra transcription tokens added. Most (90%) had four or fewer additional tokens.

percentage in the transcribed dataset according to the number of extra annotation symbols in each sentence. Of the 5068 sentences, 4543 had four or fewer extra symbols, and 904 of these had no extra annotations. Only eight sentences had twelve or more added annotations.

6.2 Evaluation

We conducted two evaluations of the robustness of the INSOMNet model using one of the trained systems from Chapter 5.3. In the first evaluation, we tested the trained model on the transcribed dataset to measure how well INSOMNet tolerated the extra annotations. In the second evaluation, we tested the same model on the detailed datasets on which it was originally trained and tested, but injected varying amounts of Gaussian noise into the network weights.

6.2.1 Transcribed Dataset

We evaluated INSOMNet on the transcribed dataset using the exact pointer match criterion introduced in Section 4.3. Under this criterion, a handle or argument is only counted as correct if it points to the correct Frame Node that holds the compressed frame pattern it represents. Figure 6.3 gives a breakout of the model's accuracy for each frame component across Frame Node activation level thresholds, together with the precision and recall curves as a measure of the model's parsing ability. With average accuracy factored in, the highest F-measure is 0.77 at a threshold of $x = 0.7$, which is comparable to INSOMNet's performance on the unannotated sentences shown in Figure 5.7, al-

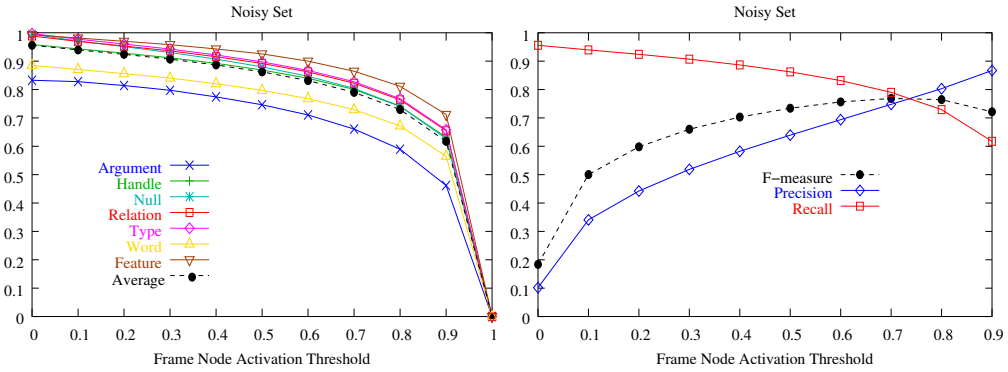


Figure 6.3: **Performance on the Transcribed Dataset.** (***Color figure***) The graph on the left gives a breakout of INSOMNet's accuracy on each frame component decoded for Frame Nodes activated above threshold. The graph on the right shows the resulting precision/recall curves, with accuracy factored in. The F-measure is highest at $x = 0.7$ with a value of 0.77.

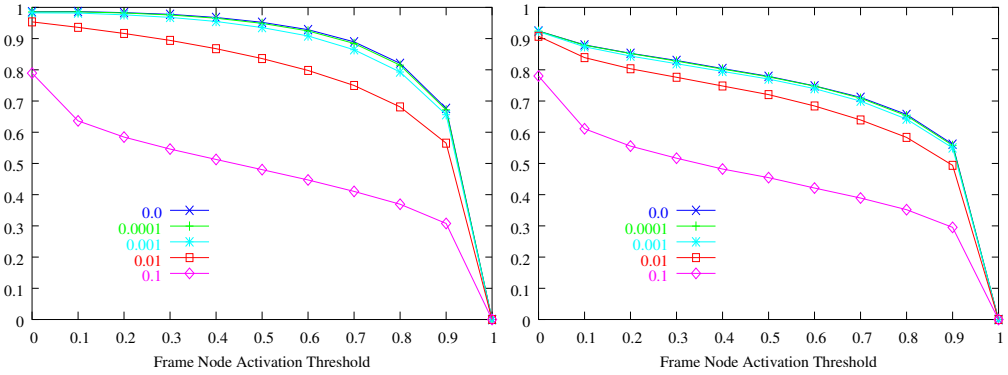


Figure 6.4: **Average Accuracy on Detailed Dataset with Added Noise.** (***Color figure***) The average accuracy across frame activation thresholds is plotted for five levels of Gaussian noise with a standard deviation ranging from 0.0 to 0.1. The average accuracy with no noise is indistinguishable from the noise level of 0.0001 on both the training and test sets. Accuracy degrades smoothly for each additional level of noise.

though it includes all sentences. These results show that INSOMNet tolerates the extra annotation symbols very well.

6.2.2 Added Gaussian Noise

For the second evaluation, we ran the trained model on the original detailed training and test datasets, but with varying amounts of noise injected into the weights. The noise added was distributed normally with a mean of 0.0 and standard deviation ranging from 0.0001 to 0.1. Figure 6.4 shows the average accuracy for each dataset. A noise level of 0.0001 is indistinguishable from no noise, while 0.001 noise results in the slightest degradation in accuracy on both the training and test datasets.

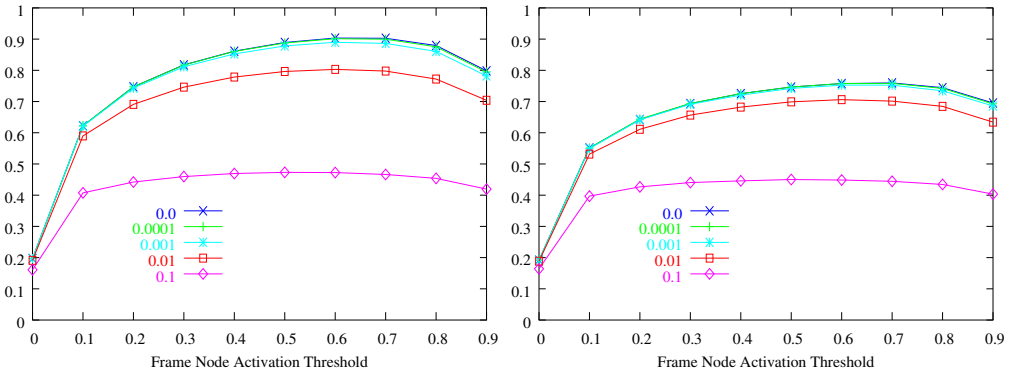


Figure 6.5: **F-Measure of Detailed Dataset with Added Noise.** (***Color figure***) These curves give the F-measure across frame activation thresholds for the five noise levels. Again, the results show graceful degradation as more noise is added.

There is a more marked fall-off with a noise level of 0.01, but INSOMNet still tolerates this noise well. However, a noise level of 0.1 results in a substantial decrease in average accuracy on both datasets, but virtually no loss in generalization, as the nearly identical 0.1 curves for the training and test sets make evident.

Figure 6.5 shows the F-measure curves for each of the noise levels. The degradation in performance mirrors the accuracy curves of Figure 6.4.

6.3 Discussion

The results show that INSOMNet is remarkably tolerant to a variety of types of noise. Even though the system was only trained on sentences that had been cleaned of the annotations in the original VerbMobil recorded dialogues, INSOMNet could still process sentences annotated with transcriptions for numerous audio effects, including

- ungrammatical input: *here is some clues*
- accents: *uhm that ist no good*
- contractions: *# # uh how 'bout some time #*
- dysfluency: *I am free # from ah , that , # from nine , until , # oh # six*

Furthermore, INSOMNet demonstrates graceful degradation with increasing levels of Gaussian noise added to the model's weights, in keeping with its connectionist foundation.

Chapter 7

Psycholinguistic Study

Up to this point, we have concentrated on INSOMNet's performance on the task of parsing and demonstrated that the network retains the robustness expected from a connectionist system. In this chapter we will show that INSOMNet also exhibits the cognitively plausible behaviors of coactivation of multiple senses, expectations and defaults, semantic priming, and nonmonotonicity, as laid out in Section 2.2 during incremental sentence processing.

7.1 Psycholinguistic Background

How do people come to a given interpretation for a sentence? Is that interpretation unique and truly independent of other possible interpretations of the same sentence? What factors contribute to the process of interpretation? Are these factors themselves independent of one another or do they interact? At which points in sentence processing do they manifest themselves? Are some factors so strong that, once applied, they cannot be overridden? These are some of the questions that have motivated a great deal of research into human linguistic performance. In this section, we will describe some of the conclusions that psycholinguistics research has drawn, as well as those questions that are still open to controversy. We will also describe the two standard types of *experience-based* models of sentence processing. Such models emphasize the role of prior language experience to determine how people process novel sentences. Finally, we will conclude with the psycholinguistic assumptions that went into the design of INSOMNet and the consequent cognitive behavior the network should exhibit.

There is overwhelming evidence that people process language *incrementally*. In sharp contrast to the approach taken by statistical broad-coverage parsers (such as the conditional log-linear model INSOMNet was compared with in Chapter 4), people do not wait until they have heard all the constituents of a sentence before arriving at an interpretation. On the contrary, there is a growing body of studies that show that people actually *anticipate* parts of a sentence in advance (Tanenhaus et al. 1995). Thus, the sequential scanning process and incremental forming of partial represen-

tations is a very plausible cognitive model for language understanding. Indeed, even theoretical linguistics has moved away from the idea that a sentence can be processed syntactically and then mapped onto (or "transformed" into) a structure that later integrates semantic information. The view now is that sentences are processed *on-line* in real time with syntax, semantics, discourse, and pragmatic information all dynamically integrated into partial linguistic descriptions that reflect the understanding of the sentence during processing (Sag 1995).

Incremental processing, however, raises the question of how people deal with the pervasive ambiguities of language during the course of understanding a sentence. If a listener does not wait until enough information in the sentence has been heard to resolve an ambiguity, then that person must either favor a particular sense (i.e., "jump to conclusions") based on what he has heard so far and how it relates to his language experience, or he must somehow maintain multiple senses without becoming bogged down in trying to consciously keep track of those interpretations. These two possibilities are not necessarily mutually exclusive. People may come to a preferred interpretation in some cases, yet entertain multiple senses in other cases, perhaps even within the same sentence.

Given the extent of ambiguity in language and the ease with which people seem to handle such ambiguity, an abundant amount of research has been conducted in psycholinguistics in order to isolate the mechanisms people use when faced with ambiguity. Such research not only informs us about how ambiguity is handled, but also yields insights into human sentence processing in general.

A number of specific types of ambiguity have been the object of particular interest. Among these are *lexical* (word-based), *syntactic* (structural), and *semantic* (meaning-based). An example of lexical ambiguity is provided by the word *ball*, which can be either the round object used in sports or a formal dance. A common syntactic ambiguity is the prepositional phrase attachment ambiguity, as can be seen in the sentence *the boy hit the girl with the ball* where it is not clear whether the ball is an instrument or modifier, although in either case, its sense as a round object is strongly preferred over its sense as a dance. We have also already encountered a typical semantic ambiguity in the sentence *every man loves some woman*, in which the way the scopes of the quantifiers *every* and *some* are embedded results in two very different interpretations. These distinctions between types of ambiguity are not always clear-cut and often overlap. Lexical and semantic ambiguity are often used interchangeably, and all three may be more or less involved in word-order phenomena. One such well-studied case is *scrambling*, where arguments of a verb may permute depending on factors like *topic* (what the discourse is about) or emphasis.

Several cognitive models have been proposed to account for how ambiguities are resolved during reading. The three most prominent in recent years have been the context-dependent, the single-access, and the multiple-access model. Although these models have been proposed primarily as accounts of how lexical ambiguity is handled, they have also been applied to the other types of ambiguity we have mentioned.

The context-dependent model (Glucksberg et al. 1986; Schvaneveldt et al. 1976) is based on the assumption that only one meaning of a word is activated at any given time, namely, the one

most appropriate to the context in which the word occurs. The primary reason is that the context primes the meaning which is most applicable, while suppressing others.

The single access (or ordered-access) model (Forster and Bednall 1976; Hogaboam and Perfetti 1975; Simpson and Burgess 1985) posits that only one active interpretation of an ambiguous sentence is maintained at any one time. If in the course of processing the sentence information is encountered that does not accord well with the active interpretation, then that interpretation is abandoned and a representation that accounts for the established information as well as for the current ambiguity is sought, most probably through backtracking to the point of ambiguity. The activation level of an interpretation is determined by the relative frequencies of the meanings of the word or words that are the source of the ambiguity. The search process for the appropriate meaning takes place serially, terminating when a fit is made, or retaining the most dominant meaning when no contextually relevant match can be found. In the strongest statement of the model (Hogaboam and Perfetti 1975), only the most dominant meaning of an ambiguous word is retrieved first, regardless of whether the context supports a subordinate meaning.

The multiple access model (Onifer and Swinney 1981; Seidenberg et al. 1982; Tanenhaus et al. 1979) suggests that several interpretations may be actively maintained when ambiguous information is encountered. At a later time, when additional input allows resolving the ambiguity, only the appropriate interpretation is retained. However, not all of the interpretations may be maintained with equal activation levels. Rather, the strength of a particular activation would be proportional to the likelihood of that interpretation being the correct one. Unlike the single access model, in which a single meaning is sought and selected, the multiple access model claims that all meanings are activated simultaneously regardless of context, but the context later influences selection of the most appropriate one.

The choice of cognitive model influences the design of computational models. In particular, the context-dependent and single-access models, both of which assert that only one interpretation is actively maintained at any given time, are often the psycholinguistic assumptions on which many probabilistic systems like statistical models that can perform active backtracking are based. Connectionist systems generally assume the multiple-access model, although many probabilistic systems also actively maintain multiple interpretations during sentence processing as a set of candidates.

As a connectionist system, INSOMNet is founded on the multiple-access model. Multiple sense co-activation is an underlying cognitive assumption of the model that is used to account for how ambiguity and its resolution are treated, semantic priming and expectations, as well as the nonmonotonic revision of the incremental interpretation of a sentence to avoid costly backtracking.

7.2 Experiments

We trained INSOMNet on the dataset used in Miikkulainen (1997b), which was based on the influential study of McClelland and Kawamoto (1986) on semantic feature generalization. In the

Category	Nouns
thing	**human animal object**
human	*man woman boy girl*
animal	*bat chicken dog sheep wolf lion*
predator	*wolf lion*
prey	*chicken sheep*
food	*chicken cheese pasta carrot*
utensil	*fork spoon*
fragileObject	*plate window vase*
hitter	*bat ball hatchet hammer vase paperweight rock*
breaker	*bat ball hatchet hammer paperweight rock*
possession	*bat ball hatchet hammer vase dog doll*
object	*bat ball hatchet hammer paperweight rock vase plate window fork spoon pasta cheese chicken carrot desk doll curtain*

Figure 7.1: **Noun Categories.** Each category can be replaced by the nouns to its right. Notice that the categories overlap and comprise a basic ontology with the **thing** category as the root.

original study, a connectionist system was trained to map syntactic constituents to thematic roles from a corpus of 152 sentences over 34 nouns and verbs for which semantic features (e.g., animacy) were prespecified. In Miikkulainen's task, the network learns to develop its own word representations through the FGREP (Forming Global Representations with Extended Backpropagation; Miikkulainen 1993) method. The FGREP method extends the standard backpropagation algorithm by propagating the signal error from the hidden layer to the input layer.

7.2.1 Training Data

Because the FGREP network had to induce its own word representations, a portion of the original McClelland and Kawamoto dataset was expanded to 1475 sentences over 30 words according to the 19 sentence templates in Figure 7.2. The words were distributed among 12 noun categories, four verbs, and the preposition *with*. The categories in Figure 7.1 define a basic ontology with **thing** as the root. The 1475 sentences themselves were generated from the 19 sentence templates by replacing each category with a noun belonging to that category.

Of the 1475 sentences, half (728) have an instrumental reading, generated by templates 4, 8, and 14. Only 112 have a modifier sense, given by templates 3 and 13. Templates 9, 11, 15, and 19 generate 221 sentences with a non-agent subject, and templates 7, 10, 12, 16, and 18 generate 144 sentences with the same syntactic structure as the sentences with a non-agent subject, but with an agent subject. For example, the sentence *the bat broke the window* has a non-agent subject when the word *bat* was to be interpreted as an inanimate object, but an agent subject when *bat* is interpreted as animate. Only 85 sentences in the dataset were truly ambiguous, thus accounting for 170 different framesets among them.

Template	Sentence Frame	Case Roles
1	The **human** ate.	agent
2	The **human** ate the **food**.	agent/patient
3	The **human** ate the **food** with the **food**.	agent/patient/modifier
4	The **human** ate the **food** with the **utensil**.	agent/patient/instrument
5	The **animal** ate.	agent
6	The **predator** ate the **prey**.	agent/patient
7	The **human** broke the **fragileObject**.	agent/patient
8	The **human** broke the **fragileObject** with the **breaker**.	agent/patient/instrument
9	The **breaker** broke the **fragileObject**.	instrument/patient
10	The **animal** broke the **fragileObject**.	agent/patient
11	The **fragileObject** broke.	patient
12	The **human** hit the **thing**.	agent/patient
13	The **human** hit the **human** with the **possession**.	agent/patient/modifier
14	The **human** hit the **human** with the **hitter**.	agent/patient/instrument
15	The **hitter** hit the **thing**.	instrument/patient
16	The **human** moved.	agent/patient
17	The **human** moved the **object**.	agent/patient
18	The **animal** moved.	agent/patient
19	The **object** moved.	patient

Figure 7.2: **Sentence Templates.** Each template is used to generate sentences by filling in categories (shown in bold type) with nouns from that category (see Table 7.1). Because a noun may belong to more than one category, ambiguous sentences are also generated. For example, the word *bat* may be both an **animal** and an **object**. Replacing these categories in templates 18 and 19 yield the sentence *the bat moved* with two case-role interpretations.

We constructed a MRS frameset for each sentence in the dataset. We will use the sentence *the boy hit the girl with the ball* to illustrate the encoding scheme we used. This sentence has the same syntactic structure as the sentence *the boy hit the girl with the doll* that we have used as a running example throughout the dissertation, but is more ambiguous in that people do not have as strong a preference for the modifier reading for *ball* as they do for *doll*. This preference is reflected in the dataset by giving both the instrumental and modifer interpretations for *the boy hit the girl with the ball*, but only the modifier reading for *the boy hit the girl with the doll*. Figure 7.3 shows the MRS frameset for *the boy hit the girl with the ball*, which is the same as that in Figure 3.1, but with *doll* replaced by *ball*. Recall that MRS uses handle-sharing to represent attachment. For sentences that have an instrumental interpretation for the prepositional phrase, the frame for the preposition *with* has the same handle **h1** as the verb it attaches to, and the **A0** role of *with* will have the same representation as the **EV** role of *hit*. The modifier reading is indicated by a *with* frame sharing the same handle **h5** as the noun *girl* it modifies, while its **A0** role will match the *girl*'s **IX** role. There are two important points to keep in mind. First, INSOMNet is only exposed to one interpretation or the other during training; the frameset in Figure 7.3 shows both interpretations for the purpose of

the boy hit the girl with the ball .

h0	—	**prpstn_rel**	**SA**	**h1**	—	—	—	—	—	
h1	***hit***	**arg13_rel**	**A0A1A3DMEV**	—	**x0**	**x1**	—	**e0**	—	
h1	***with***	**miscprep_ind_rel**	**A0A3DMEV**	**e0**	**x2**	—	—	—	—	
e0	—	**EVT**	**DVASMOTN**	bool	−asp	ind	past	—	—	
h2	—	**def_explicit_rel**	**BVDMRESC**	**x0**	—	**h3**	—	—	—	
h3	***boy***	**ani_nom_rel**	**A3IX**	—	**x0**	—	—	—	—	
x0	—	**FRI**	**DVGNPNPT**	−	masc	3sg	prn	—	—	
h4	—	**def_explicit_rel**	**BVDMRESC**	**x1**	—	**h5**	—	—	—	
h5	***girl***	**ani_nom_rel**	**A3IX**	—	**x1**	—	—	—	—	
h5	***with***	**miscprep_ind_rel**	**A0A3DMEV**	**x1**	**x2**	—	—	—	—	
x1	—	**FRI**	**DVGNPNPT**	−	fem	3sg	prn	—	—	
h6	—	**def_explicit_rel**	**BVDMRESC**	**x2**	—	**h7**	—	—	—	
h7	***ball***	**reg_nom_rel**	**IX**	**x2**	—	—	—	—	—	
x2	—	**FRI**	**DVGNPNPT**	−	neu	3sg	prn	—	—	

Figure 7.3: **Prepositional Phrase Attachment Ambiguity.** (***Color figure***) The sentence *the boy hit the girl with the ball* is an example of a prepositional phrase attachment ambiguity. Before the word *ball* is read in at the end of the sentence, INSOMNet will activate *with* frames for both the instrumental and modifier senses. The frameset in the figure shows the decoded frames once the entire sentence has been processed. For the instrumental reading, the *with* frame has the same handle **h1** as the verb *hit* to which it attaches, and its **A0** role has the same argument representation as the verb's **EV** slot (**e0**). The modifier sense is given by the *with* frame that shares its handle **h5** with that of the noun *girl* it modifies. In this case, the **A0** role of the *with* frame shares the noun's **IX** representation.

exposition. Second, the two senses will generally be encoded in separate Frame Nodes, which is how INSOMNet exhibits sense co-activation. How strongly these two senses are activated depends on where in the sentence the network is and the correlation of the processed words with each sense. For example, the sequence *the boy hit* occurs 182 times in the training set with the instrumental sense, but only 27 times with the modifier sense. But the *the boy hit the girl* occurs with both senses 7 times. Therefore, INSOMNet should have a strong preference for the instrumental reading until it reads in the word *girl*, at which point both senses should be approximately the same. Once the end-of-sentence marker is read, the activation of the Frame Node with the strongest sense should approach the value 1.0, and the activation of the Frame Node with the opposite sense should fall off to 0.0. In the case of the sentence *the boy hit the girl with the ball*, these two activations strengths should be roughly equal, whereas for *the boy hit the girl with the doll*, only the modifier interpretation should be strongly activated. That this is indeed the case will be demonstrated shortly in Section 7.3.2.

There are two other sources of ambiguity in the dataset that INSOMNet has to learn to distinguish, and both happen to depend on whether the word *bat* is used in an animate sense. One source of ambiguity arises from sentence templates 9 and 10, where *bat* can be both an **animal** and

the bat broke the vase .

h0	—	**prpstn_rel**	**SA**	**h1**	—	—	—	—	—	
h1	***broke***	**arg13_rel**	**A0A1A3DMEV**	—	**x0**	**x1**	—	**e0**	—	
h1	***broke***	**arg3_event_rel**	**A0A3DMEV**	—	**x1**	—	**e0**	—	—	
h1	***with***	**miscprep_ind_rel**	**A0A3DMEV**	**e0**	**x0**	—	—	—	—	
e0	—	**EVT**	**DVASMOTN**	bool	−asp	ind	past	—	—	
h2	—	**def_explicit_rel**	**BVDMRESC**	**x0**	—	**h3**	—	—	—	
h3	***bat***	**ani_nom_rel**	**A3IX**	—	**x0**	—	—	—	—	
h3	***bat***	**reg_nom_rel**	**IX**	**x0**	—	—	—	—	—	
x0	—	**FRI**	**DVGNPNPT**	−	neu	3sg	prn	—	—	
h4	—	**def_explicit_rel**	**BVDMRESC**	**x1**	—	**h5**	—	—	—	
h5	***vase***	**reg_nom_rel**	**IX**	**x1**	—	—	—	—	—	
x1	—	**FRI**	**DVGNPNPT**	−	neu	3sg	prn	—	—	

Figure 7.4: **Implied Instrument.** (***Color figure***) The sentence *the bat broke the vase* is lexically ambiguous: either the word *bat* is the animate agent of *broke* or it is an inanimate instrument with which the *vase* is broken. In the dataset, animate objects have the semantic relation **ani_nom_rel** and inanimate objects, **reg_nom_rel**. The extra *with* frame that shares the verb's handle **h1** indicates the instrument is *bat* because the **A3** role of *with* and the **IX** role for *bat* are bound with **x0**.

a **breaker**. If it is an **animal**, it is the agent of the sentence; otherwise, it is an implied instrument (i.e., without an explicit *with* in the sentence). Figure 7.4 illustrates how we have represented the two interpretations using the MRS formalism for the sentence *the bat broke the vase*. Unlike the case with the prepositional phrase attachment ambiguity described earlier, where the distinction is only manifested in the *with* frame, the animacy of *bat* is reflected in the frames for *broke*, *with*, as well as *bat* itself. We use a new semantic relation, **ani_nom_rel** (which is not in the ERG) as the means of distinguishing animates from inanimates, to which we assign the usual **reg_nom_rel** semantic relation. The subcategorization for the verb *broke* shows the transitive interpretation for an animate *bat*, but the instrumental sense for the *bat* used as an inanimate object by an unstated agent. We also added a frame for the implied word *with* to keep the representation consistent with the other sentences in the dataset representing instruments, such as the one described in Figure 7.3. Accordingly, the *with* frame shares its handle **h1** with the verb frame and its **A0** role is bound with the verb's **EV** role to the representation for **e0**.

The other source of ambiguity results from the same lexical distinction with the animacy of *bat* in sentence templates 18 and 19. Instantiating the **animal** category with *bat* requires an animate reading, whereas the **object** category is inanimate. The MRS encoding for the two resulting interpretations is much like the transitive/instrumental distinction in *the bat broke the vase* just described. Figure 7.5 shows the framesets we used to encode the sentence *the bat moved*. We assign **ani_nom_rel** as the semantic relation for the animate sense of *bat*, and give the verb *moved* a transitive subcatgorization. In this case, the frameset has three extra frames: the object of the

the bat moved .

h0	**—**	**prpstn_rel**	**SA**	**h1**	**—**	**—**	**—**	**—**	**—**
h1	***moved***	**arg13_rel**	**A0A1A3DMEV**	**—**	**x0**	**x1**	**—**	**e0**	**—**
h1	***moved***	**arg3_event_rel**	**A0A3DMEV**	**—**	**x0**	**—**	**e0**	**—**	**—**
e0	**—**	**EVT**	**DVASMOTN**	bool	−asp	ind	past	**—**	**—**
h2	**—**	**def_explicit_rel**	**BVDMRESC**	**x0**	**—**	**h3**	**—**	**—**	**—**
h3	***bat***	**ani_nom_rel**	**A3IX**	**—**	**x0**	**—**	**—**	**—**	**—**
h3	***bat***	**reg_nom_rel**	**IX**	**x0**	**—**	**—**	**—**	**—**	**—**
x0	**—**	**FRI**	**DVGNPNPT**	−	neu	3sg	prn	**—**	**—**
h4	**—**	**def_explicit_rel**	**BVDMRESC**	**x1**	**—**	**h5**	**—**	**—**	**—**
h5	***self***	**reg_nom_rel**	**IX**	**x1**	**—**	**—**	**—**	**—**	**—**
x1	**—**	**FRI**	**DVGNPNPT**	−	gen	3sg	refl	**—**	**—**

Figure 7.5: **Implied Patient.** (***Color figure***) The lexical ambiguity in the sentence *the bat moved* is similar to that of Figure 7.4, but there is an important difference: the patient in the ergative sense of *moved* is bound with the handle **x0** that fills the **IX** role for *bat* itself. The resulting interpretation is that an unnamed agent moved the bat. This interpretation is only invoked when the sense of *bat* is inanimate. The transitive subcategorization for *moved* is given in the frameset with the patient role identified with the *self* frame.

verb, *self*, its governing determiner (with handle **h4**), and the frame with handle **x1** that encodes the features for *self*. When the interpretation of *bat* is of an inanimate object, there is again an unstated agent that is understood to have moved the *bat*.

7.2.2 System Parameters

The dataset was divided into a training set with 1438 sentences and 37 test sentences. INSOMNet was trained with an initial learning rate of 0.01 and the Frame Node Indicator Map given an initial learning rate of 0.4. The neighborhood was initialized to half the diameter of the Frame Map (3). The learning rate of the Frame Node Indicator Map was decayed by 0.9 and the neighborhood decremented according to the schedule

$$\text{epoch}_{i+1} = 1.5 * \text{epoch}_i + 1$$

where i indexes each parameter update. Once the Frame Node Indicator Map had stopped self-organizing when its learning rate fell below 0.001, the learning rate for INSOMNet was decayed by 0.5 according to the same schedule.

7.3 Results

INSOMNet learned the case-role mapping task well, with an average performance on the test set of 95.5% using the exact pointer match criterion.

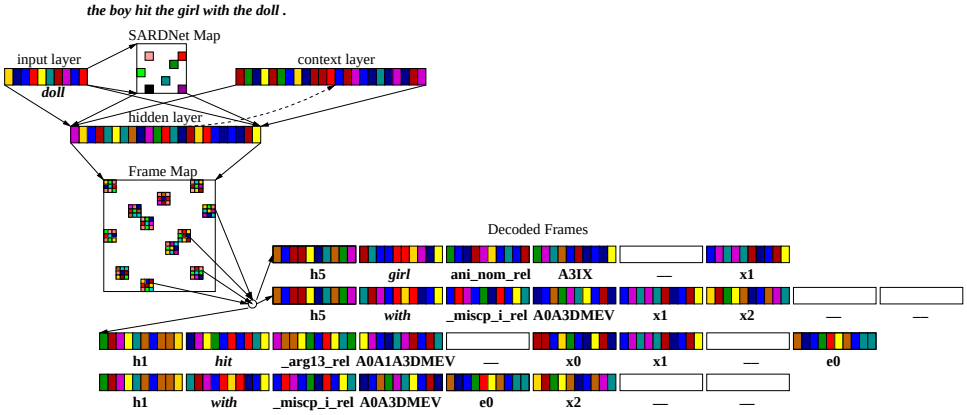


Figure 7.6: **Representing Ambiguity in INSOMNet.** (***Color figure***) Ambiguity is represented in INSOMNet through the coactivation of distinct senses. In the prepositional phrase attachment ambiguity in the sentence *the boy hit the girl with the doll*, the two interpretations of *doll* as instrument or modifier are represented in the *with* frames. If the interpretation involves *doll* as an instrument, *with* will share its handle **h1** with the verb *hit* to represent verb-attachment and its **A0** role will have the same filler as the verb's **EV** role **e0**. If instead the interpretation involves *doll* as a modifier, *with* will share its handle **h5** with the noun *girl* to represent noun-attachment and its **A0** role will have the same filler as the noun's **IX** role **x1**. If the sentence is ambiguous, both interpretations will be activated, as shown in the figure where two separate Frame Nodes are decoded into the two distinct *with* frames. Allowing such multiple representations to be explicitly activated is one of the main advantages of the Frame Map component of INSOMNet. The Decoder's shared weights are symbolized in the diagram by a small circle which receives connections from the Frame Map and sends connections to the individual frames (only four of which are shown).

In this section, we will demonstrate INSOMNet's performance four sentences, *the boy hit the girl with the doll/rock/ball/bat* with respect to ambiguity as coactivation of multiple senses, ambiguity resolution, semantic priming, nonmonotonicity, and expectation.

7.3.1 Coactivation of multiple senses

INSOMNet is trained to represent ambiguity by coactivating the Frame Node patterns on the Frame Map that encode the sense distinctions. Figure 7.6 illustrates this idea with the prepositional phrase attachment ambiguity in the sentence *the boy hit the girl with the doll*. If *doll* is to be interpreted as an instrument, then there will be a distinct Frame Node that can be decoded into a MRS frame in which *with* has the same handle representation as the verb *hit* frame (**h1**) and the same filler representation in its **A0** role as the verb's **EV** role in the *hit* frame (**e0**). If *doll* is to be interpreted as a modifier of *girl*, then there will be another distinct Frame Node that can be decoded into a MRS frame in which *with* has the same handle representation as the *girl* frame (**h5**) and the same representation in its **A0** role as the **IX** role in the *girl* frame (**x1**). The two distinct *with* frames are co-activated by the Modulator Map depending on the sequence of words INSOMNet has read in.

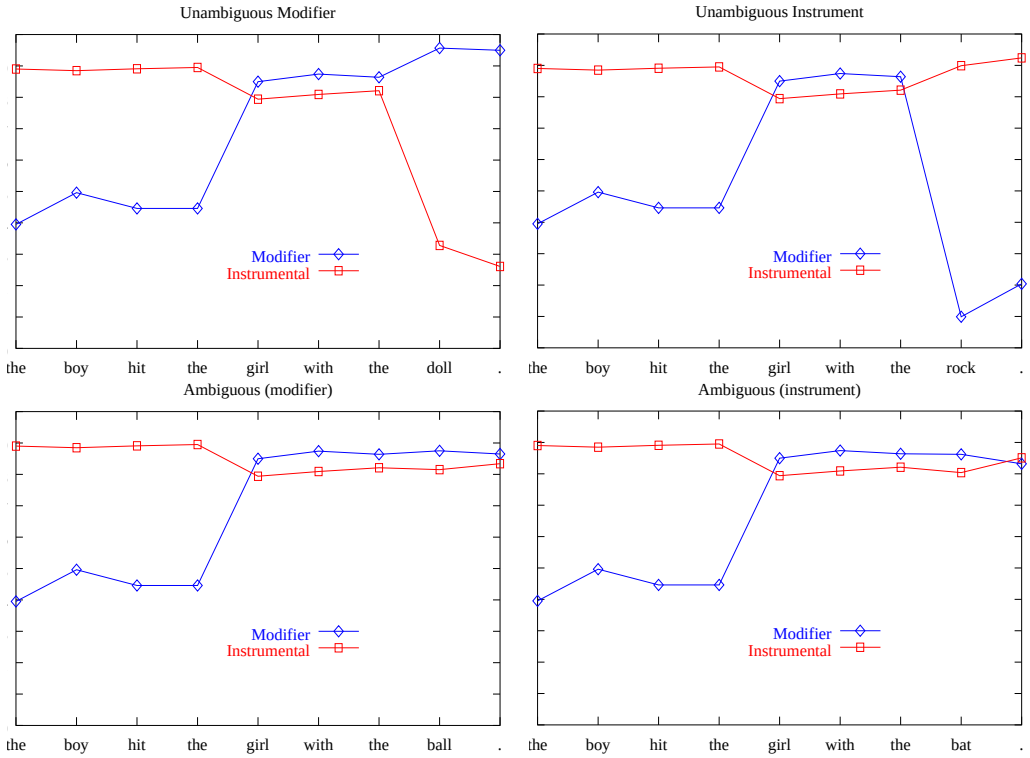


Figure 7.7: **Activation and Resolution of Alternative Senses.** (***Color figure***) The Frame Node Modulator Map controls the activation strength of the encoded frame patterns in the Frame Map as a sentence is processed. Because the prepositional phrase attachment ambiguity is represented by the co-activation of two distinct *with* frames that encode their attachment points, INSOMNet's interpretation of the ambiguity can be seen by how the activation levels of the two *with* frames fluctuate as each word is read in. This figure shows INSOMNet's interpretations of the sentences *the boy hit the girl with the doll/rock/ball/bat*.

7.3.2 Ambiguity Resolution

The network should be able to not only represent ambiguity, but disambiguate given appropriate context. To demonstrate ambiguity and its resolution, we tested a trained network on four sentences which differ only in the last word in the sentence, which may be ambiguous or not. All four sentences have the same synactic structure as *the boy hit the girl with the doll*, but, whereas this sentence is unambiguous in this corpus, having only the modifier reading, the other sentences have been selected that prefer an instrumental reading (*the boy hit the girl with the rock*, and two are ambiguous in that both interpretations are possible, although one might be ever so slightly preferred over the other. Thus, the sentence *the boy hit the girl with the ball* is truly ambiguous as both modifier and instrumental senses are acceptable, but there is a very slight preference for the modifier reading because *with the ball* has the ratio $\frac{M}{I} = \frac{114}{16} = 7.125$ while *the girl with the ball* is equally split between the two interpretations. Similarly, the sentence *the boy hit the girl with the bat* is also

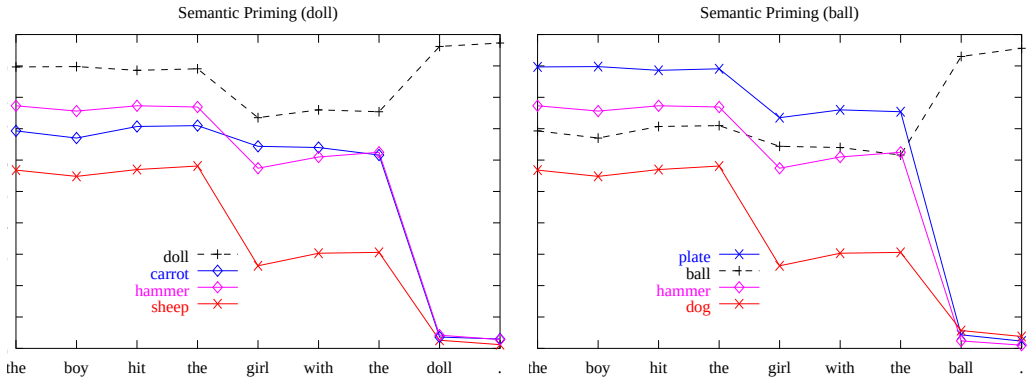


Figure 7.8: **Defaults, Expectations, and Semantic Priming.** (***Color figure***) When either the sentence *the boy hit the girl with the doll* or *the boy hit the girl with the ball* is processed, several other potential instruments and modifiers are initially activated. The dashed line shows the trajectory of the correct word. As expected, the other words are suppressed by the end of the sentence.

truly ambiguous as both modifier and instrumental senses are acceptable, but there is a very slight preference for the instrumental reading because *with the bat* has the ratio $\frac{I}{M} = \frac{116}{16} = 7.5$ while *the girl with the bat* is equally split between the two interpretations. Again, *the girl with the bat* is also equally split between the two interpretations. Figure 7.7 shows the two sense activations of the *with* Frame Node that decodes into the modifier representation and the *with* Frame Node that decodes into the instrument representation. In accordance with the frequencies of the two interpretations for the sentence segments *the boy hit* and *the boy hit the girl* as described in Section 7.2.1, INSOMNet demonstrates a preference for the instrumental reading until the word *girl* is encountered. At this point, both interpretations are roughly equal. On reading in the word *doll* and period, INSOMNet shows a strong preference for the modifier reading. With the word *rock*, the instrumental readings is activated more strongly. With the two ambiguous sentences, the activation strengths of both interpretations are roughly equal, but there is a slight preference for the modifier sense for *ball* and a slight preference for the instrumental reading for *bat*.

7.3.3 Defaults, Expectations, and Semantic priming

Figure 7.8 shows expectations and defaults that arise during sentence processing. As the sentences *the boy hit the girl with the doll* and *the boy hit the girl with the ball* are read in, several potential instruments and modifiers are activated initially. In both graphs, the dashed line shows that the activation of the frame node encoding the correct word are strengthened once the final word *doll/ball* are read in. The other activated words fluctuate as the sentences are processed, but fall off to 0 at the end of the sentence, indicating that they are not part of the final interpretation. The instrument *hammer* is present in both sentences, and follows almost an identical trajectory for both *doll* and *ball*.

Expectation based on training experience is also evident in Figure 7.7. INSOMNet has learned that instruments and, to a less extent, modifiers are likely to occur in a sentence. It activates Frame Nodes that encode words that fill these roles and modulates their activation strength while processing the sentence.

7.3.4 Nonmonotonicity

Both Figures 7.7 and 7.8 also illustrate interpretation revision. In Figure 7.7, the interpretation of the prepositional phrase attachment ambiguity is constantly revised as INSOMNet processes each new word. Because distributed representations allow for holistic computation, this revision is not restricted to a monotonic increase or decrease in activation for the competing interpretations. Rather, the activations of each interpretation may be nonmonotonically altered as warranted by context. Thus, the instrumental reading starts off highly activated, but falls off at the end of the sentence, and the modifier interpretation follows the opposite pattern.

Figure 7.8 demonstrates how words that fit the sentence thematically can become activated and then suppressed as the sentence is processed.

7.4 Discussion

The results in this chapter show that scaling INSOMNet up to parse realistic sentences from corpora has not compromised those human-like behaviors that have made neural networks appealing models to cognitive scientists. INSOMNet provides an account of both how multiple interpretations of ambiguities can be simultaneously maintained and how a given interpretation can be chosen and others suppressed when disambiguating context is encountered. Similarly, the model shows expectations and defaults, as well as semantic priming, based on its training history. Finally, the network is capable of nonmonotonic revision of an interpretation during sentence processing, allowing suppressed interpretations to be reactivated when warranted by context. This nonmonotonicity is in contrast to the types of pruning that grammar-based parsers typically use to discard unlicensed structures. Nonmonotonicity may rather be considered the subsymbolic analogue to symbolic backtracking, but can be accomplished holistically in a single step.

Chapter 8

Future Work

We designed INSOMNet with two objectives in mind. The first goal was to demonstrate that a subsymbolic sentence processor could be scaled up to parse realistic natural language. The second goal was to retain the cognitively plausible behavior that has come to be a hallmark of connectionist models.

8.1 Semantic Parsing

A central contribution of INSOMNet is its ability to parse a sentence of arbitrary complexity into an explicit semantic representation. Previous approaches to connectionist parsing have focused on constructing syntactic trees or assigning thematic case-roles to sentences with very limited syntactic structure. However, to provide an adequate semantic description of realistic natural language, a graph structure is preferable. The earliest connectionist models encoded graphs with localist representations where each node represented a concept and weights between nodes indicated relations, but such networks suffered all of the drawbacks of localist representations (see Section 2.3.1). Later models encoded graphs using variations of the RAAM network, such as labeled trees (Sperduti 1995) and folding networks trained with backpropagation through structure (Goller and Küchler 1996). Yet, RAAM-based architectures have proven notoriously difficult to scale up beyond simple grammars (Levy and Pollack 2001).

8.1.1 Representation

As described in Section 3.4.2, one of the key innovations of INSOMNet is the use of self-organization to induce patterns in the Frame Node Indicator Map that can be used as pointers by the Frame Node representations in the Frame Map. These pointers were motivated by the handles of Minimum Recursion Semantics (see Section 2.5.4) to provide for flat semantics and underspecification as a tractable means of representing ambiguity. Unlike the symbolic pointers of MRS, however, the

handle patterns used by INSOMNet are developed by self-organizing compressed representations of the MRS frames so that INSOMNet can generalize to novel frame patterns. Because the handle patterns develop semantic structure, they serve the added function of representing binding of frame arguments and their fillers, as well as the MRS method of representing predicate conjunction (such as modifiers) through a common handle.

Nevertheless, using self-organization to induce handle patterns does introduce a fair amount of complexity into the semantic representation, the INSOMNet architecture, and its training. The performance analyses in Chapters 4.3 and 5.3 could be attributed primarily to the inability of INSOMNet to distinguish handles from other handles with very similar patterns, which very often indicated neighboring nodes on the Frame Map. This confounding of handles is problematic because the compressed frame representations in neighboring nodes will generally be of semantically distinct roles, even though their handles are similar. One particularly frequent example was the handle pattern developed for MRS frames encoding the implicit determiner relation **def_rel** that governs nominals such as days of the week, proper names, and pronouns. The more frequent an input, the more map real estate self-organization will develop to represent that item. For this reason, relatively large sections of the Frame Node Indicator Map developed **def_rel** patterns to accommodate the many implicit determiners that a sentence might have. The resulting handle patterns tended to be very similar and, consequently, a source of confusion for INSOMNet.

There are a couple of ways that this problem might be countered. One simple approach would be to add distinct features to the frame encodings so that their compressed representations are forced to be more dissimilar. A better approach would be to modify the self-organizing algorithm by placing a bound on how similar map patterns can be. For example, the desired effect could be achieved by implementing an epsilon ball around each map pattern within which no updates are done.

INSOMNet was developed to demonstrate that a connectionist sentence processor could develop a semantic representation of an input sentence solely on the basis of semantic structure, such as subcategorization information, by encoding semantics in the MRS handles and using these handles as argument fillers. All non-pointer MRS components, such as words, semantic relations, and linguistic features, were given random representations. Certainly, a more complete account of sentence processing would entail more descriptive representations for these components, such as phonological and morphological features for words and semantic features for the ERG relations that represented the HPSG type hierarchy from which they are derived.

8.1.2 Architecture

Among connectionists of a more purist disposition, the idea of stipulating any network characteristic that could instead be learned is strongly frowned upon. Yet, INSOMNet has a number of such architectural stipulations required to fulfill its goal to "scale up without dumbing down;" for this reason, we regard INSOMNet as an honest effort to bridge the language understanding/engineering

divide. The explicit semantic representation INSOMNet generates for an input sentence could easily be adapted as a natural language interface to a database. There is even the possibility that these representations could be holistically transformed into equivalent representations in other languages while preserving inter-language ambiguities such as word order as part of a machine translation system (Chalmers 1990; Knight and Langkilde 2000). Moreover, INSOMNet's connectionist foundation allows the model to remain robust to human language errors, while its plausibility as a cognitive model enhances its ability to derive a meaningful interpretation despite such errors.

INSOMNet has three major design components tailored to the task of parsing a sentence into a set of frames with slots and fillers needed for the MRS representation of the sentence's interpretation. The first is the Frame Map, which is divided up into a prespecified number of Frame Nodes that puts an upper bound on the number of frames that the model can encode. The second architectural constraint is the design of the Decoder Network: it assumes each frame has a handle, word, semantic relation, and subcategorization type, and that there are a set number of roles that any predicate could take. The third component is the Frame Node Modulator Map, which is used to self-organize the Frame Map and control the activation levels for Frame Nodes to indicate graded frame selection for both parsing and cognitive modeling.

Simply adding more nodes to the Frame Map with the object of accommodating any conceivable number of frames that might occur in real language is not a very satisfying solution from a cognitive standpoint. A more cognitively defensible approach would be to reuse nodes. In the current INSOMNet model, the nodes encode compressed representations of MRS frames, but there is no reason in principle that they could not be trained to compress larger structures corresponding to several MRS frames. Groupings such as determiner/nominal/features are very common. A mechanism would be required whereby such subgraphs could be built up and compressed in one node, freeing up previously used nodes, which could then be reused to build up similar structures as a sentence is processed. How to train the network to allocate these nodes during sentence processing would be an interesting and challenging research undertaking.

Whereas the nodes in the Frame Map could be used to encode any frame representation, the Decoder Network is the INSOMNet component designed specifically for the MRS representation for sentences in the Redwoods Treebank. Certainly a similar network could be designed to decode other semantic formalism, but a more cognitively motivated approach that would also have deep linguistic implications would be for the network to induce its own role representations. In particular, we are thinking of the argument/adjunct issue. In MRS, arguments have specific role labels such as *arg1* and *scope*, but adjuncts are represented as predicate conjuncts via handle-sharing for the simple reason that there is no bound specified on the number of adjuncts in HPSG. Because the distinction between arguments and adjuncts is not truly binary in natural language, that distinction should not be imposed on the architecture. Rather, the network should be allowed to develop its own internal representation for thematic roles and adjuncts in such a way that a blend is possible. One way that this issue might be approached within the current MRS framework would be to allow

INSOMNet to develop its own subcategorization representations for predicates based on the semantics of their arguments *and* adjuncts. Such subcategorization representations would come to encode regularities in their argument structures together with the semantic characteristics of adjuncts that tend to occur with them. The gradience in the resulting representations would likely reflect the *optionality* of arguments and adjuncts based on their frequency in the corpus. Predicates having required arguments, such as strict intransives like *exist* or ditransitives like *give*, would develop associated subcategorization representations in distinct clusters. Verbs with optional arguments, such as *move*, should become associated with subcategorization representations depending on how the word is used in the sentence. Thus, for example, in the case of the sentence *the bat moved* in Figure 7.5, the network should be able to develop subcategorization representations corresponding to the transitive and ergative senses from the semantic information of its arguments, such as animacy. The semantics of adjuncts, too, would have to become encoded in the subcategorization representation (in which case it would no longer strictly represent subcategorization as such). Every verb could then be potentially associated with a variety of representations depending on the number and semantics of their adjuncts. Yet, the productivity of adjuncts versus the idiosyncratic nature of argument structure should allow for a great deal of generalization in these representations. Furthermore, the distinction between arguments and adjuncts would only be discernible from a statistical analysis of the representations associated with a given verb. For example, the required locative phrase argument for a verb like *put* would have to be inferred from the fact that locative semantics are always a part of any subcategorization representation associated with it.

The induction of subcategorization information is by no means a trivial task, and the above outline only touches on some of the issues. A complete account would certainly require a much more detailed account of lexical and phrasal semantics than that currently provided in the Redwoods Treebank. Yet, from a theoretical viewpoint, such a line of research could provide a great deal of insight into how diachronic variations in a language factor into the variety of ways a verb may express its arguments, such as the *like*-complement in English (Bender and Flickinger 199; Przepiórkowski 1999).

The third design component, the Frame Selector, does not in itself impose an architectural constraint on the model, but does have a necessary one-to-one mapping with the nodes in the Frame Map in order to self-organize it and represent graded frame selection. The graded frame selection could have been incorporated into the Frame Map itself, but would have had two drawbacks, one implementational and the other cognitive. Implementationally, all of the nodes which were not selected to contribute to the MRS representation of a sentence (on average, 90%) would have had to be trained to be null vectors, requiring substantially more training than the Frame Node Indicator Map alone. Moreover, from a cognitive viewpoint, it is not clear what a graded frame representation would actually *represent* other than a distorted compression of a MRS frame. What is graded in INSOMNet is frame *selection*, not representation. Once a frame is accessed (on the basis that is above some threshold), the frame pattern as encoded in the selected Frame Node is decoded

without regard to its activation level.

Self-organization, too, might be incorporated into the Frame Map, but this design choice would mean solving the chicken/egg issue of knowing which Frame Node should encode which MRS frame before the node pattern has been activated by propagating the hidden layer pattern to the not-yet-selected node. Moreover, the pattern in a Frame Node will also have to encode the complete frame representation, including the handle, the representation for which depends on the node selected to encode the frame. The solution of first encoding the handles as compressed representations of the rest of the MRS frames which they labeled, and then using them to self-organize the Frame Node Indicator Map, circumvented this problem, and at the same time, provided a mechanism to indicate binding by using these handles as argument fillers. As well as this works, finding a more elegant approach would certainly be a worthwhile research direction.

8.2 Cognitive Plausibility

The results of Chapters 6 and 7 demonstrate that INSOMNet retains the cognitively valid behaviors of connectionist models that have long appealed to researchers in psycholinguistics. The distributed representations that INSOMNet develops to represent the semantic interpretation of a sentence help the model remain robust to noise, both in the input sentence and added to its weights. These representations also allow INSOMNet to maintain multiple interpretations of a sentence in parallel in the face of ambiguity, as well as perform holistic transformations to nonmonotonically revise an interpretation once disambiguating context is encountered. Together with semantic parsing, these characteristics suggest that a natural extension of INSOMNet would as a basis for a spoken language system. The ability of connectionist systems to seamlessly integrate multiple source of information means that the system could be trained to incorporate prosodic and discourse information in the course of sentence processing to generate more precise semantic interpretations than is possible from the words in the sentence alone.

While the focus of this dissertation has been on scaling up a subsymbolic model of sentence processing to realistic language without losing cognitive plausibility, further research is needed to validate INSOMNet on particular psycholinguistic phenomena. In particular, the way that INSOMNet represents ambiguity should be grounded in empirical data, such as from experiments using reading times or eye-tracking. Similarly, INSOMNet exhibits defaults and expectations, and semantic priming, but these behaviors also need empirical support in order to have confidence that INSOMNet will process language the way humans do.

Chapter 9

Conclusion

In this dissertation, we presented a subsymbolic sentence processing model, INSOMNet, that is able to parse real-world corpora of sentences into semantic representations. A crucial innovation was to use self-organization to capture regularities in semantic encodings of MRS frames so that INSOMNet could learn to reliably decode compressed representations of those frames back into their components. This design choice was motivated by the need to represent MRS handles, which serve as predicate labels and, consequently, can be treated as pointers that can fill the argument slots of frames. By representing the handles as compressed frame patterns, INSOMNet was able to learn to associate arguments with the semantic features of their fillers.

We demonstrated INSOMNet on two variants of MRS representations of sentences in the Redwoods Treebank. The first variant was a subset of the full MRS sentence representation called elementary semantic dependency graphs. This demonstration was provided as the closest possible comparison between INSOMNet and a conditional log-linear model evaluated on the same dataset. An important caveat, however, is that the statistical model is designed to rank parses licensed by the ERG, whereas INSOMNet must learn the complete semantic representations themselves, as well as indicate the parse preference by selecting just those frames belonging to an interpretation and suppressing all others. The second variant was the full MRS encoding for the Redwoods Treebank sentences to show that INSOMNet was able to handle their semantic complexity. In both cases, the results showed that INSOMNet learned the semantic encodings, and generalized well to novel sentences.

As a connectionist model, INSOMNet should retain those qualities that have made them indispensable tools in psycholinguistics research, such as robustness to input errors and network damage, coactivation of multiple interpretations, expectations and defaults, semantic priming, non-monotonic interpretation revision, and incrementality.

To ensure that INSOMNet remained a robust system, we evaluated the model on the original sentences from the VerbMobil project that had been annotated with dialogue transcriptions. These sentences featured a variety of speech errors, including pauses, repairs, dysfluencies, and hesitations.

Nevertheless, INSOMNet was able to parse these sentences into their semantic interpretations nearly as well as it could their cleaned-up counterparts in the Redwoods Treebank.

As a further evaluation of its robustness, we added varying amounts of Gaussian noise to the network links to compare it on datasets from the scale-up study. As expected, the network showed graceful degradation with increasing noise.

Finally, we trained and evaluated INSOMNet on the case-role assignment task using a variant of the McClelland and Kawamoto corpus to demonstrate the model's cognitive plausibility. In this study, we focused on the course of sentence processing rather than INSOMNet's parsing ability. The results showed that INSOMNet developed an interpretation of a sentence incrementally, which it could nonmonotonically revise with context. The network was also able to maintain coactivation of ambiguous interpretations, and choose one interpretation or another on the basis of disambiguating context. In the course of developing a semantic interpretation, INSOMNet demonstrated expectations and defaults, as well as semantic priming in accordance with its training experience.

These properties and its ability to scale up to realistic language make INSOMNet a promising foundation for understanding human sentence processing in the future.

Appendix A

Semantic Representation

A.1 MRS Example Frameset

 # okay i guess i need to meet with you for about two hour -s during the week .

h0	_	**discourse_rel**	**CALHLIRHRI**	**e0**	**h1**	_	**h2**	**e0**	_
h1	**okay**	**excl_rel**	**A0EX**	_	**okay**	_	_	_	_
h2	_	**prpstn_rel**	**SA**	**h3**	_	_	_	_	_
h3	**guess**	**arg14_rel**	**A0A1A4DMEV**	_	**x0**	**h4**	_	**e0**	_
e0	_	**CE**	**DVASMOTN**	**BOOL**	**-ASP**	**IND**	**PRES**	_	_
h4	_	**prpstn_rel**	**SA**	**h7**	_	_	_	_	_
h5	_	**def_rel**	**BVDMRESC**	**x0**	_	**h6**	_	_	_
h6	**i**	**pron_rel**	**IX**	**x0**	_	_	_	_	_
h7	**need**	**arg14_rel**	**A0A1A4DMEV**	_	**x0**	**h8**	_	_	_
h8	**to**	**hypo_rel**	**SA**	**h12**	_	_	_	_	_
h12	**meet**	**arg1_rel**	**A0A1DMEV**	_	**x0**	_	**e1**	_	_
h12	**with**	**independent_rel**	**A0A3DMEV**	**e1**	**x1**	_	_	_	_
h12	**for**	**independent_rel**	**A0A3DMEV**	**e1**	**x2**	_	_	_	_
h12	**during**	**abstr_rel**	**A0A3DMEV**	**e1**	**x3**	_	_	_	_
x0	_	**FRI**	**DVGNPNPT**	**-**	**GEN**	**1SG**	**STD1SG**	_	_
e1	_	**EVT**	**DVASMOTN**	**BOOL**	**-ASP**	**MOOD**	**TNS**	_	_
h13	_	**def_rel**	**BVDMRESC**	**x1**	_	**h14**	_	_	_
h14	**you**	**pron_rel**	**IX**	**x1**	_	_	_	_	_
x1	_	**FRI**	**DVGNPNPT**	**-**	**GEN**	**2PER**	**STD2**	_	_
h15	_	**udef_rel**	**BVDMRESC**	**x2**	_	**h18**	_	_	_
h18	**hour**	**_hour_rel**	**A3DMIX**	_	_	**x2**	_	_	_
h18	**about**	**degree_rel**	**DGDM**	**d0**	_	_	_	_	_
h18	**two**	**const_rel**	**A0CVDM**	**x2**	**two**	**d0**	_	_	_
d0	_	**DI**	**DV**	**BOOL**	_	_	_	_	_
x2	**-s**	**FRI**	**DVGNPNPT**	**+**	**GEN**	**3PL**	**PRN**	_	_
h19	**the**	**def_explicit_rel**	**BVDMRESC**	**x3**	_	**h20**	_	_	_
h20	**week**	**non_day_dm_rel**	**A3IX**	_	**x3**	_	_	_	_
x3	_	**FRI**	**DVGNPNPT**	**-**	**NEU**	**3SG**	**PRN**	_	_

A.2 Semantic Annotations

A.2.1 Semantic Roles

Abbreviation	Semantic Role	Purpose
A0	ARG	predicative/scopal argument
A1	ARG1	external argument
A2	ARG2	raising argument
A3	ARG3	patient argument
A4	ARG4	propositional argument
AP	AM-PM	am/pm
BV	BV	bound variable
CA	C-ARG	discource argument
CV	CONST_VALUE	constant value
CX	C-INST	comparative index
DG	DARG	degree argument
DM	DIM	dimension
EV	EVENT	basic predicative index
EX	EXCL	exclamation
F1	FACTOR1	first addend
F2	FACTOR2	second addend
G1	ARG-1	first comparative argument
G2	ARG-2	second comparative argument
HA	HANDEL	predicate label
HI	HOUR-IND	hour index
HR	HOUR	hour
HX	HINST	head of phrasal relation
IX	INST	basic nominal relation index
LH	L-HANDEL	discourse left branch handle
LI	L-INDEX	discourse left branch index
MD	MINUTE-IND	minute index
MI	MIN	minute
MN	MAIN	main clause
ND	NAMED	named entity
NX	NHINST	non-head of phrasal relation
PR	PROP	property
PS	PRPSTN	proposition
RE	RESTR	restriction
RH	R-HANDEL	discourse right branch handle
RI	R-INDEX	discourse right branch index
RL	ROLE	role for ellipsis construction
SA	SOA	state of affairs
SB	SUBORD	subordinate clause
SC	SCOPE	scope
SN	SEASON	season
T1	TERM1	first multiplicand
T2	TERM2	second multiplicand
TL	TITLE	title
YI	YEAR-IND	year index

A.2.2 Subcategorization Frames

Abbreviation	Role breakout
A0	ARG0
A0A1A2A3DMEV	ARG0 ARG1 ARG2 ARG3 DIM EVENT
A0A1A2DMEV	ARG0 ARG1 ARG2 DIM EVENT
A0A1A3A4DMEV	ARG0 ARG1 ARG3 ARG4 DIM EVENT
A0A1A3DMEV	ARG0 ARG1 ARG3 DIM EVENT
A0A1A4DMEV	ARG0 ARG1 ARG4 DIM EVENT
A0A1DMEV	ARG0 ARG1 DIM EVENT
A0A2A4DMEV	ARG0 ARG2 ARG4 DIM EVENT
A0A3DGDM	ARG0 ARG3 DEGREE DIM
A0A3DMEV	ARG0 ARG3 DIM EVENT
A0A3DMEVIX	ARG0 ARG3 DIM EVENT INST
A0A4DMEV	ARG0 ARG4 DIM EVENT
A0CVDM	ARG0 CONST_VALUE DIM
A0DM	ARG0 DIM
A0DMEV	ARG0 ARG3 DIM EVENT
A0DMEVRL	ARG0 DIM EVENT ROLE
A0DMF1F2	ARG0 DIM FACTOR1 FACTOR2
A0DMT1T2	ARG0 DIM TERM1 TERM2
A0EX	ARG0 EXCL
A3DMIX	ARG3 DIM INST
A3IX	ARG3 INST
A3IXND	ARG3 INST NAMED
A3IXNDYI	ARG3 INST NAMED-IND
A3IXSN	ARG3 INST SEASON
A4IX	ARG4 INST
APDMHRIXMI	AM-PM DIM HOUR INST MIN
BVDMRESC	BV DIM RESTR SCOPE
CALHLIRHRI	C-ARG L-HANDEL L-INDEX R-HANDEL R-INDEX
DGDM	DARG DIM
DMHDIXMI	DIM HOUR-IND INST MINUTE-IND
DMIX	DIM INST
DV	DIVISIBLE
DVASMOTN	DIVISIBLE ASPECT MOOD TENSE
DVGNPNPT	DIVISIBLE GEN PN PRONTYPE
EVPRPS	EVENT PROP PRPSTN
G1G2CXDGDM	ARG-1 ARG-2 C-INST DARG DIM
HXNX	HINST NHINST
IX	INST
IXND	INST NAMED
IXTL	INST TITLE
MNSB	MAIN SUBORD
SA	SOA

A.2.3 Semantic Feature Types

Abbreviation	Semantic Feature Type
FRI	full referential index
RI	referential index
DI	degree index
CI	conjunctive index
CRI	conjunctive referential index
CFRI	conjunctive full referential index
E|I	event or referential index
EVT	event feature structure
CE	conjunctive event feature structure

A.2.4 Semantic Features

Abbreviation	Semantic Feature	Meaning
+	+	positive boolean
GEN	GENDER	gender
!-1SG	STRICT_NON13SG	strictly non-singular
PRN	PRONTYPE	generic pronoun type
BOOL	BOOL	boolean
3PL	3PL	third person plural
!-PRF	STRICT_NONPRF	strictly non-perfective
IND	INDICATIVE	indicative mood
PRES	PRESENT	present tense
-	-	negative boolean
1SG	1SG	first person singular
STD1SG	STD_1SG	stardard first singular
2PER	2PER	second person number
STD2	STD_2	standard second person
-ASP	NO_ASPECT	non-aspectual
FUT	FUTURE	future tense
3SG	3SG	third person singular
!2PER	STRICT_2PER	strictly second person
NEU	NEUT	neuter gender
!BOOL	STRICT_BOOL	strictly boolean
+&-	+_AND_-	both positive and negative boolean
1PL	1PL	first person plural
STD1PL	STD_1PL	standard first plural
MOOD	MOOD	generic mood
TNS	TENSE	generic tense
PRG	PROGR	progressive tense
PERNUM	PERNUM	person and number
0PN	ZERO-PRON	null pronoun
IND|MODSBJ	IND_OR_MOD_SUBJ	indicative or modal subjunctive mood
STD3	STD_3	standard third person
MODSBJ	MOD_SUBJ	modal subjunctive
ANDR	ANDRO	grammatical masculine gender
ASP	ASPECT	generic aspect
IND+MODSBJ	IND+MODSUBJ	indicative and modal-subjunctive mood
-PRF	NONPRF	non-perfective aspect
-PRG+-PRF	NONPRG+NONPRF	non-progressive and non-perfective
PRES+FUT	PRES+FUT	present and future tense
PAST	PAST	simple past tense
PRF	PERF	perfective aspect
-1SG	NON1SG	not first singular
RECP	RECIP	reciprocal pronoun
STDPN	STD_PRON	standard pronoun
-TNS	NO_TENSE	non-tensed
G918	GLBTYPE918	global type
-ASP+PRG	NOASP+PROGR	non-aspectual and progressive
3PLSG	3PL_AND_3SG	third person plural and singular
!-3SG	STRICT_NON3SG	strictly non-third singular
PRG+PRF	PROGR+PERF	progressive and perfective
PAST+FUT	PAST+FUTURE	past and future
FEM	FEM	feminine gender
!IND|MODSBJ	STRICT_IND_OR_MOD_SUBJ	strictly indicative or modal subjunctive mood
SBJ	SUBJ	subjunctive mood
MASC	MASC	masculine gender
2SG	2SG	second person singular
REFL	REFL	reflexive pronoun
PRES+PAST	PRES+PAST	present and past tense

A.3 Raw Data

```
lkb(): 1 tree for item # 1.
6 (86); derivation:
(4903 adjh_s 0 0 16 (9 okay_root 0 0 1 ("okay" 0 1))
(4902 subjh 0 1 16 (12 i 0 1 2 ("I" 1 2)) (4898 hcomp 0 2 16
(32 non_third_sg_fin_verb_infl_rule 0 2 3 (32 guess_v2 0 2 3
```

```
("guess" 2 3))) (4893 subjh 0 3 16 (36 i 0 3 4 ("I" 3 4))
(4884 hcomp 0 4 16 (54 non_third_sg_fin_verb_infl_rule 0 4 5
(54 need_v2 0 4 5 ("need" 4 5))) (4875 hcomp 0 5 16
(73 to_c_prop 0 5 6 ("to" 5 6)) (4856 hadj_i_uns 0 6 16
(3790 hadj_i_uns 0 6 13 (438 hadj_i_uns 0 6 9
(138 bse_verb_infl_rule 0 6 7 (138 meet_v1 0 6 7 ("meet" 6 7)))
(285 hcomp 0 7 9 (240 with_p 0 7 8 ("with" 7 8)) (284 you 0 8 9
("you" 8 9)))) (3701 hcomp 0 9 13 (504 for 0 9 10 ("for" 9 10))
(3595 bare_np 0 10 13 (3594 adjn_i 0 10 13 (1560 hspec 0 10 12
(578 about_deg 0 10 11 ("about" 10 11)) (581 two 0 11 12
("two" 11 12))) (3592 noptcomp 0 12 13
(3589 plur_noun_infl_rule 0 12 13 (3589 hour_n1 0 12 13
("hours" 12 13)))))))) (4306 hcomp 0 13 16 (4195 during 0 13 14
("during" 13 14)) (4305 hspec 0 14 16 (4299 the 0 14 15
("the" 14 15)) (4304 noptcomp 0 15 16
(4301 sing_noun_infl_rule 0 15 16 (4301 week1 0 15 16
("week" 15 16))))))))))))
6 (86); MRS string:

[
  INDEX: e1 [ CONJ_EVENT
              DIVISIBLE:  BOOL
              E.TENSE:  PRESENT*
              E.ASPECT:  NO_ASPECT*
              E.MOOD:  INDICATIVE* ]
  LISZT: <
          [ excl_rel
            HANDEL: h2
            ARG: v3
            EXCL: "OKAY" ]
          [ discourse_rel
            HANDEL: h4
            C-ARG: e1
            L-HANDEL: h2
            L-INDEX: v6 [ NON_EXPL
                          DIVISIBLE:  BOOL ]
            R-HANDEL: h5
            R-INDEX: e1 ]
          [ pron_rel
            HANDEL: h7
            INST: x8 [ FULL_REF-IND
                          DIVISIBLE:  -
                          PRONTYPE:  STD_1SG
                          PNG.PN:  1SG
                          PNG.GEN:  GENDER ] ]
          [ def_rel
            HANDEL: h9
```

```
  BV: x8
  RESTR: h10
  SCOPE: h11
  DIM: v12 [ NON_EXPL-IND
             DIVISIBLE:  BOOL ] ]
[ _guess_h_rel
  HANDEL: h13
  EVENT: e1
  ARG: v15
  ARG1: x8
  ARG4: h14
  DIM: v16 [ NON_EXPL-IND
             DIVISIBLE:  BOOL ] ]
[ pron_rel
  HANDEL: h17
  INST: x18 [ FULL_REF-IND
             DIVISIBLE:  -*
             PRONTYPE:  STD_1SG
             PNG.PN:  1SG
             PNG.GEN:  GENDER ] ]
[ def_rel
  HANDEL: h19
  BV: x18
  RESTR: h20
  SCOPE: h21
  DIM: v22 [ NON_EXPL-IND
             DIVISIBLE:  BOOL ] ]
[ _need2_rel
  HANDEL: h23
  EVENT: e24 [ EVENT
             DIVISIBLE:  BOOL
             E.TENSE:  PRESENT*
             E.ASPECT:  NO_ASPECT*
             E.MOOD:  INDICATIVE ]
  ARG: v26
  ARG1: x18
  ARG4: h25
  DIM: v27 [ NON_EXPL-IND
             DIVISIBLE:  BOOL ] ]
[ hypo_rel
  HANDEL: h25
  SOA: h28 ]
[ _meet_v_rel
  HANDEL: h29
  EVENT: e32 [ EVENT
             DIVISIBLE:  BOOL
             E.TENSE:  TENSE
```

```
              E.ASPECT:  NO_ASPECT*
              E.MOOD:  MOOD ]
  ARG: v30
  ARG1: x18
  DIM: v31 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ _with_rel
  HANDEL: h29
  EVENT: e33 [ EVENT
              DIVISIBLE:  BOOL
              E.TENSE:  NO_TENSE
              E.ASPECT:  ASPECT
              E.MOOD:  MOOD ]
  ARG: e32
  ARG3: x35 [ FULL_REF-IND
              DIVISIBLE:  -*
              PRONTYPE:  STD_2
              PNG.PN:  2PER
              PNG.GEN:  GENDER ]
  DIM: v34 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ pron_rel
  HANDEL: h36
  INST: x35 ]
[ def_rel
  HANDEL: h37
  BV: x35
  RESTR: h38
  SCOPE: h39
  DIM: v40 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ _for_rel
  HANDEL: h29
  EVENT: e41 [ EVENT
              DIVISIBLE:  BOOL
              E.TENSE:  NO_TENSE
              E.ASPECT:  ASPECT
              E.MOOD:  MOOD ]
  ARG: e32
  ARG3: x43 [ FULL_REF-IND
              DIVISIBLE:  +
              PNG.PN:  3PL
              PNG.GEN:  GENDER
              PRONTYPE:  PRONTYPE ]
  DIM: v42 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ _about_approx_rel
```

```
  HANDEL: h44
  DARG: d45 [ DEG-IND
              DIVISIBLE:  BOOL ]
  DIM: v46 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ card_rel
  HANDEL: h44
  ARG: x43
  CONST_VALUE: "2"
  DIM: d45 ]
[ _hour_rel
  HANDEL: h44
  INST: x43
  ARG3: v48 [ NON_EXPL
              DIVISIBLE:  BOOL ]
  DIM: v47 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ udef_rel
  HANDEL: h49
  BV: x43
  RESTR: h50
  SCOPE: h52
  DIM: v51 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ _during_rel
  HANDEL: h29
  EVENT: e53 [ EVENT
              DIVISIBLE:  BOOL
              E.TENSE:  NO_TENSE
              E.ASPECT:  ASPECT
              E.MOOD:  MOOD ]
  ARG: e32
  ARG3: x55 [ FULL_REF-IND
              DIVISIBLE:  -*
              PRONTYPE:  PRONTYPE
              PNG.PN:  3SG*
              PNG.GEN:  NEUT* ]
  DIM: v54 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ _def_rel
  HANDEL: h56
  BV: x55
  RESTR: h58
  SCOPE: h57
  DIM: v59 [ NON_EXPL-IND
              DIVISIBLE:  BOOL ] ]
[ _week_rel
```

```
          HANDEL: h60
          INST: x55
          ARG3: v61 [ NON_EXPL
                      DIVISIBLE:  BOOL ] ]
        [ prpstn_rel
          HANDEL: h14
          SOA: h62 ]
        [ prpstn_rel
          HANDEL: h5
          SOA: h63 ] >
HCONS: <  h10 QEQ h7
          h20 QEQ h17
          h28 QEQ h29
          h38 QEQ h36
          h50 QEQ h44
          h58 QEQ h60
          h62 QEQ h23
          h63 QEQ h13 > ]
```

`HCONS` outscope list is used to find immediately dominating frames.

Bibliography

Abney, S. (1996). Statistical methods and linguistics. In Klavans, J., and Resnik, P., editors, *The Balancing Act*. Cambridge, MA: MIT Press.

Alexandersson, J., Reithinger, N., and Maier, E. (1997). Insights into the dialogue processing of VERBMOBIL. Technical Report 191, Saarbrücken, Germany.

Allen, J., and Seidenberg, M. S. (1999). The emergence of grammaticality in connectionist networks. In MacWhinney, B. J., editor, *Emergence of Language*, 115–151. Hillsdale, NJ: Erlbaum.

Allen, R. B. (1987). Several studies on natural language and back-propagation. In *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA), vol. II, 335–341. Piscataway, NJ: IEEE.

Barton, G. E., Berwick, R. C., and Ristad, E. S. (1987). *Computational Complexity and Natural Language*. Cambridge, MA: MIT Press.

Bender, E., and Flickinger, D. (199). Diachronic evidence for extended argument structure. In et al, G. B., editor, *Constraints and Resources in Natural Language Syntax and Semantics*, 3–20. Stanford: CSLI Press.

Berg, G. (1992). A connectionist parser with recursive sentence structure and lexical disambiguation. In Swartout, W., editor, *Proceedings of the Tenth National Conference on Artificial Intelligence*, 32–37. Cambridge, MA: MIT Press.

Bod, R. (2001). What is the minimal set of fragments that achieves maximal parse accuracy?. In *Proceedings of the 39th Annual Meeting of the ACL*, 66–73.

Chalmers, D. J. (1990). Syntactic transformations on distributed representations. *Connection Science*, 2:53–62.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the ACL*, 132–139. Seattle, WA.

Chen, S., and Rosenfeld, R. (1999). A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Chomsky, N. (1970). Remarks on nominalization. In Jacobs, R. A., and Rosenbaum, P. S., editors, *Readings in English Transformational Grammar*, 184–221. Waltham, Mass.: Ginn-Blaisdell.

Chrisman, L. (1991). Learning recursive distributed representations for holistic computation. *Connection Science*, 3:345–366.

Christiansen, M. H., and Chater, N. (1999). Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23(2):157–205.

Christiansen, M. H., and Devlin, J. T. (1997). Recursive inconsistencies are hard to learn: A connectionist perspective on universal word order correlations. In Shafto, M. G., and Langley, P., editors, *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, 113–118. Hillsdale, NJ: Erlbaum.

Church, K., and Patil, R. (1982). Coping with syntactic ambiguity or how to put the block in the box on the table. *American Journal of Computational Linguistics*, 8(3–4):139–149.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Copestake, A., Lascarides, A., and Flickinger, D. (2001). An algebra for semantic construction in constraint-based grammars. In *Proceedings of ACL-2001*. Toulouse, France.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225.

Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Smith, A., Parisi, D., and Plunkett, K. (1996). Learning the past tense. In *Rethinking Innateness: A Connectionist Perspective on Development*, 130–147. Cambridge, MA: MIT Press.

Fillmore, C. J. (1968). The case for case. In Bach, E., and Harms, R. T., editors, *Universals in Linguistic Theory*, 0–88. New York: Holt, Rinehart and Winston.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1) (Special Issue on Efficient Processing with HPSG):15–28.

Forster, K. I., and Bednall, E. S. (1976). Terminating and exhaustive search in lexical access. *Memory and Cognition*, 4:53–61.

Glucksberg, S., Kreutz, R. J., and Rho, S. (1986). Context can constrain lexical access: Implications for interactive models of language comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 12:323–335.

Goller, C., and Küchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. *IEEE Transactions on Neural Networks*, 1:347–352.

Hammerton, J. (2001). Clause identification with long short-term memory. In Daelemans, W., and Zajac, R., editors, *Proceedings of CoNLL-2001*, 61–63.

Henderson, J. (1994). *Description Based Parsing in a Connectionist Network*. PhD thesis, University of Pennsylvania, Philadelphia, PA. Technical Report MS-CIS-94-46.

Hindle, D., and Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120.

Hinton, G. E. (1981). Implementing semantic networks in parallel hardware. In Hinton, G. E., and Anderson, J. A., editors, *Parallel Models of Associative Memory*, 161–187. Hillsdale, NJ: Erlbaum.

Ho, E. K. S., and Chan, L. W. (2001). Analyzing holistic parsers: Implications for robust parsing and systematicity. *Neural Computation*, 13(5):1137–1170.

Ho, K. S. E., and Chan, L. W. (1997). Confluent preorder parsing of deterministic grammars. *Connection Science*, 9:269–293.

Hogaboam, T. W., and Perfetti, C. A. (1975). Lexical ambiguity and sentence comprehension. *Journal of Verbal Learning and Verbal Behavior*, 14:265–274.

Honkela, T., Kaski, S., Kohonen, T., and Lagus, K. (1998). Self-organizing maps of very large document collections: Justification of the WEBSOM method. In Balderjahn, I., Mathar, R., and Schader, M., editors, *Classification, Data Analysis, and Data Highways*, 245–252. Berlin: Springer.

Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560.

Hudson, S. B., and Tanenhaus, M. K. (1984). Ambiguity resolution in the absence of contextual bias. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, 188–192. Hillsdale, NJ: Erlbaum.

Jain, A. N. (1991). *PARSEC: A Connectionist Learning Architecture for Parsing Spoken Language*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-91-208.

James, D. L., and Miikkulainen, R. (1995). SARDNET: A self-organizing feature map for sequences. In (Tesauro et al. 1995), 577–584.

Kangas, J., Torkkola, K., and Kokkonen, M. (1992). Using SOMs as feature extractors for speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*. Piscataway, NJ: IEEE.

Knight, K., and Langkilde, I. (2000). Preserving ambiguities in generation via automata intersection. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 697–702. Cambridge, MA: MIT Press.

Kohonen, T. (1984). *Self-Organization and Associative Memory*. Berlin; New York: Springer. First edition.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480.

Kohonen, T. (1995). *Self-Organizing Maps*. Berlin; New York: Springer.

Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Paatero, V., and Saarela, A. (2000). Self-organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11:574–585.

Lane, P. C. R., and Henderson, J. B. (2001). Incremental syntactic parsing of natural language corpora with simple synchrony networks. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):219–231.

Lawrence, S., Giles, C. L., and Fong, S. (2000). Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):126–140.

Levy, S., and Pollack, J. (2001). Infinite RAAM: A principled connectionist substrate for cognitive modeling. In *Proceedings of the Fourth International Conference on Cognitive Modeling*.

MacDonald, M. C. (1993). The interaction of lexical and syntactic ambiguity. *Journal of Memory and Language*, 32:692–715.

MacDonald, M. C., Just, M. A., and Carpenter, P. A. (1992). Working memory constraints on the processing of syntactic ambiguity. *Cognitive Psychology*, 24:56–98.

Manning, C., and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

Marslen-Wilson, W. D., editor (1989). *Lexical Representation and Process*. Cambridge, MA: MIT Press.

Mayberry, M. R., III, and Miikkulainen, R. (1999). Using a sequential SOM to parse long-term dependencies. In *Proceedings of the 21st Annual Conference of the Cognitive Science Society*, 367–372. Hillsdale, NJ: Erlbaum.

McClelland, J. L., and Kawamoto, A. H. (1986). Mechanisms of sentence processing: Assigning roles to constituents. In McClelland, J. L., and Rumelhart, D. E., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*, 272–325. Cambridge, MA: MIT Press.

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

Miikkulainen, R. (1996). Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, 20:47–73.

Miikkulainen, R. (1997a). Dyslexic and category-specific impairments in a self-organizing feature map model of the lexicon. *Brain and Language*, 59:334–366.

Miikkulainen, R. (1997b). Natural language processing with subsymbolic neural networks. In Browne, A., editor, *Neural Network Perspectives on Cognition and Adaptive Robotics*, 120–139. Bristol, UK; Philadelphia, PA: Institute of Physics Publishing.

Miikkulainen, R., and Dyer, M. G. (1991). Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science*, 15:343–399.

Mooney, R. (1999). Learning for semantic interpretation: Scaling up without dumbing down. In Cussens, J., editor, *Proceedings of the 1st Workshop on Learning Language in Logic*, 7–15. Bled, Slovenia.

Munro, P., Cosic, C., and Tabasko, M. (1991). A network for encoding, decoding and translating locative prepositions. *Connection Science*, 3:225–240.

Ng, A. Y., and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press. Make NIPS14.

Oepen, S. (2001). [incr tsdb()] — Competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany. In preparation.

Oepen, S., Flickinger, D., Toutanova, K., and Manning, C. (2002). LinGO redwoods: A rich and dynamic treebank for HPSG. In *Beyond PARSEVAL. Workshop of the Third LREC Conference*.

Onifer, W., and Swinney, D. A. (1981). Accessing lexical ambiguities during sentence comprehension: Effects of frequency of meaning and contextual bias. *Memory and Cognition*, 9:225–226.

Plaut, D. C., and Shallice, T. (1993). Perseverative and semantic influences on visual object naming errors in optic aphasia: A connectionist account. *Journal of Cognitive Neuroscience*, 5(1):89–117.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Pollard, C., and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Chicago, IL: The University of Chicago Press.

Przepiórkowski, A. (1999). *Case Assignment and the Complement-Adjunct Dichotomy: A Non-Configurational Constraint-Based Approach*. PhD thesis, Universität Tübingen, Germany.

Reilly, R. (1992). A connectionist technique for on-line parsing. Neuroprose.

Rohde, D. T. (2002). *A Connectionist Model of Sentence Comprehension and Production*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-02-105.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, 318–362. Cambridge, MA: MIT Press.

Sag, I. A. (1995). Taking performance seriously. Manuscript.

Schvaneveldt, R. W., Meyer, D. E., and Becker, C. A. (1976). Lexical ambiguity, semantic context, and visual word recognition. *Child Development*, 48:612–616.

Seidenberg, M. S., Tanenhaus, M. K., Leiman, J. M., and Bienkowski, M. (1982). Automatic access of the meanings of ambiguous words in context: Some limitations of knowledge-based processing. *Cognitive Psychology*, 14:489–537.

Sells, P. (1985). *Lectures on Contemporary Syntactic Theories*. Stanford, CA: Center for the Study of Language and Information.

Sharkey, N. E., and Sharkey, A. J. C. (1992). A modular design for connectionist parsing. In Drossaers, M. F. J., and Nijholt, A., editors, *Twente Workshop on Language Technology 3: Connectionism and Natural Language Processing*, 87–96. Enschede, the Netherlands: Department of Computer Science, University of Twente.

Shastri, L., and Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:417–494.

Simpson, G. B., and Burgess, C. (1985). Activation and selection processes in the recognition of ambiguous words. *Journal of Experimental Psychology: Human Perception and Performance*, 11:28–39.

Small, S. L., Cottrell, G. W., and Tanenhaus, M. K., editors (1988). *Lexical Ambiguity Resolution: Perspectives from Psycholinguistics, Neuropsychology and Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann.

Sperduti, A. (1995). Encoding of Labeled Graphs by Labeling RAAM. In (Tesauro et al. 1995), 1125–1132.

St. John, M. F., and McClelland, J. L. (1988). Learning and applying contextual constraints in sentence comprehension. In *Proceedings of the 10th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

St. John, M. F., and McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, 46:217–258.

Stolcke, A. (1990). Learning feature-based semantics with simple recurrent networks. Technical Report TR-90-015, International Computer Science Institute, Berkeley, CA.

Tanenhaus, M. K., Leiman, J. M., and Seidenberg, M. S. (1979). Evidence for multiple stages in the processing of ambiguous words in syntactic contexts. *Journal of Verbal Learning and Verbal Behavior*, 18:427–440.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., and Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Tesauro, G., Touretzky, D. S., and Leen, T. K., editors (1995). *Advances in Neural Information Processing Systems 7*. Cambridge, MA: MIT Press.

Touretzky, D. S. (1991). Connectionism and compositional semantics. In Barnden, J. A., and Pollack, J. B., editors, *High-Level Connectionist Models*, vol. 1 of *Advances in Connectionist and Neural Computation Theory,* Barnden, J. A., series editor, 17–31. Norwood, NJ: Ablex.

Toutanova, K., and Manning, C. (2002). Feature selection for a rich hpsg grammar using decision trees. In *Proceedings of CoNLL-2002*, 77–83. Taipei, Taiwan.

Toutanova, K., Manning, C. D., Shieber, S. M., Flickinger, D., and Oepen, S. (2002). Parse disambiguation for a rich hpsg grammar. In *First Workshop on Treebanks and Linguistic Theories (TLT2002)*, 253–263.

Wahlster, W., editor (2000). *Verbmobil. Foundations of Speech-to-Speech Translation*. Berlin; New York: Springer-Verlag.

Weckerly, J., and Elman, J. L. (1992). A PDP approach to processing center-embedded sentences. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, 414–419. Hillsdale, NJ: Erlbaum.

Williams, R. J., and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.

Vita

Marshall Reeves Mayberry III was born in Philadelphia, Pennsylvania on September 28, 1963, the son of Sarah Jane Heister and Marshall Reeves Mayberry Jr. After graduating from White Hall High School, Pine Bluff, Arkansas, in 1982, he entered the U.S. Air Force where he served for six years. From 1988 until 1990 he worked as a computer programmer for Martin Marietta in San Antonio, Texas. In 1990, he entered the University of Texas at Austin and received the degrees of Bachelor of Science in Computer Science and in Mathematics in 1993. That same year he entered the Graduate School of the University of Texas. He received the degree of Master of Science in Computer Science in 1995.

Permanent Address: 8100 N. MoPac Expy #248
Austin, Texas 78759 USA
martym@cs.utexas.edu
http://www.cs.utexas.edu/users/martym/

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, and extended by Bert Kay and James A. Bednar.