PARSEC:
A Connectionist Learning Architecture for Parsing Spoken Language

Ajay N. Jain

December 1991
CMU–CS–91–208

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213–3890

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

This research was sponsored in part by the Department of the Navy, Office of the Chief of Naval Research under contract N00014–91–J–1131, in part by ATR Interpreting Telephony Research Laboratories, in part by Siemens Corporation, in part by the National Science Foundation under contract EET–8716324, and in part by NEC Corporation.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government, ATR Interpreting Telephony Research Laboratories, Siemens Corporation, or NEC Corporation.

Keywords: Parsing, neural networks, natural language processing, learning, prosody, speech translation, grammar, speech recognition

Carnegie Mellon

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

PARSEC: A Connectionist Learning Architecture for Parsing Speech

AJAY JAIN

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

A. Waibel — MAJOR PROFESSOR — 12/12/1991 — DATE

David S. Touretzky — MAJOR PROFESSOR — 12/12/91 — DATE

R. F. Rashid — DEAN — 1/30/92 — DATE

APPROVED:

Paul Christmann — PROVOST — 7 February 1992 — DATE

Abstract

A great deal of research has been done developing parsers for natural language, but adequate solutions for some of the particular problems involved in spoken language are still in their infancy. Among the unsolved problems are: difficulty in constructing task-specific grammars, lack of tolerance to noisy input, and inability to effectively utilize complimentary non-symbolic information.

This thesis describes PARSEC—a system for generating connectionist parsing networks from example parses. PARSEC networks exhibit three strengths:

- They automatically learn to parse, and they generalize well compared to hand-coded grammars.
- They tolerate several types of noise without any explicit noise-modeling.
- They can learn to use multi-modal input, e.g. a combination of intonation, syntax and semantics.

The PARSEC network architecture relies on a variation of supervised back-propagation learning. The architecture differs from other connectionist approaches in that it is highly structured, both at the macroscopic level of modules, and at the microscopic level of connections. Structure is exploited to enhance system performance.

Conference registration dialogs formed the primary development testbed for PARSEC. A separate simultaneous effort in speech recognition and translation for conference registration provided a useful data source for performance comparisons.

Presented in this thesis are the PARSEC architecture, its training algorithms, and detailed performance analyses along several dimensions that concretely demonstrate PARSEC's advantages.

Acknowledgements

When younger graduate students ask me about finding a thesis topic, I tell them that they shouldn't worry too much—a thesis topic might just find them. As I neared the end of qualifiers and other mechanical hurdles, I recall three meetings that affected my thesis direction. One was a chance meeting with Dave Touretzky at one of the department's Friday afternoon parties. Dave brought up an interest he had in connectionist language processing. Another meeting with Masaru Tomita (Tommy) exposed interest in connectionist language models from a different quarter. Still another meeting with Alex Waibel and Dave Touretzky put me in possession of a large stack of papers about connectionist language models that started me off on the path that led to this thesis.

I would like to thank Alex, Dave, and Tommy, my local thesis committee, for directing me towards connectionist language processing and for providing guidance and support during the course of the work. Alex and Dave also deserve special recognition as my advisors; they were always available for discussion, and they channeled my impatience and energy in positive directions. I also thank Steve Hanson for his work as the external committee member. He lent a fresh perspective to the work.

Special thanks go to Scott Fahlman, whose work on constructive learning and empirical results with back-propagation contributed to PARSEC's success, and to Jay McClelland, whose pioneering work in connectionist NLP influenced my early work.

I am pleased to acknowledge the financial support provided by the School of Computer Science. I am grateful to the innumerable people who have made the department such a nice place to work. Sharon Burks deserves special mention for routinely going to great lengths to ensure that everybody's life moves along smoothly.

Of course, without family and friends, this would have been neither possible, nor any fun. Constant encouragement and high expectations from my parents played a signifi-

cant role in shaping my (so far) successful path. I thank them for always doing their best for me. I am also grateful for the support of my wife's parents, who helped in every way they could. My mother-in-law went so far as to commission a group of Franciscan monks to pray for a successful conclusion to my thesis (thanks guys!).

Many friends have shared the experience of graduate school during these years. On our first day, Nico Habermann told us the attrition statistics for the program. At lunch that afternoon, three of us talked about which one wouldn't make it. It remains to be seen if we all made it and beat the odds, but it will be interesting to keep in touch and find out for sure. Kevin, Vince, Yolanda, Clay, Anwar, John, and everybody else: thanks for the good times.

Most importantly though, I thank my wife Theresa. She put up with life in Pittsburgh, my more neurotic moments, and distance from her family and closest friends, but she always managed to make me laugh and keep me happy. More than anyone else, she made this worthwhile and possible. This thesis is dedicated to her.

Table of Contents

List of Figures

List of Tables

1 Introduction

The problem of spoken language processing spans a large number of sub-problems, each of which is substantial in scope. These range from speech recognition at the lowest level to language understanding at the highest. One of the critical pieces of a spoken language system is its parser. The parser bridges the gap between word sequences and meaningful structures. It must communicate with the highest and lowest level processing components.

While a great deal of research has been done developing parsers for natural language, existing solutions for some of the particular problems involved in spoken language are limited. This dissertation describes PARSEC[1], a connectionist parsing system geared toward the difficulties encountered in spoken language processing. The main testbed for the parser is a conversational speech translation task.

I am primarily concerned with demonstrating that connectionist computational models are capable of solving problems in parsing that other more traditional methods are unable to do. The focus is not on modeling human learning performance or understanding how humans process language. In this thesis, connectionist models are viewed as a tool to solve various problems in parsing.

1.1 Why Connectionist?

Traditional methods employed in parsing natural language have focused on developing powerful formalisms to represent syntactic and semantic structure along with rules for transforming language into these formalisms. Such systems generally rely on rules or

1. The name PARSEC was derived from pieces of *parsing*, *speech*, and *connectionist*.

hand-coded programs to perform their tasks, and system designers must accurately anticipate all of the language constructs that the systems will encounter. Spoken language adds complexity to the language understanding problem. It is more strictly sequential than written language. One *cannot* look ahead in on-line real-time tasks such as simultaneous interpretation. Often, output is desired *before* sentences are complete. Also, spoken language has a loose structure that is not easily captured in formal grammar systems. This is compounded by errors in word recognition. Phenomena such as ungrammaticality, stuttering, interjections, and hesitation are possible. Speech signals also contain a rich variety of currently unexploited non-symbolic information (e.g. pitch or energy patterns). Independent of these factors, systems that can be efficiently produced for specific linguistic domains are desirable. Parsing methodologies designed to cope with these requirements are needed.

Connectionist networks have three main computational strengths that are useful in such domains:

1. They learn and can generalize from examples. This offers a potential solution to the difficult problem of constructing grammars for spoken language.
2. Connectionist networks tend to be tolerant of noisy input as is present in real speech.
3. By virtue of the learning algorithms they employ, connectionist networks can exploit statistical regularities across different modalities (e.g. syntactic information and prosodic information).

The subject of this thesis is a connectionist parsing system that demonstrates these benefits.

1.2 Conference Registration

Conference registration dialogs have been proposed as a testbed for developing real-time bi-directional speech-to-speech translation systems. A conversational conference registration task forms the primary development testbed for this work. Here is an example conversation:

CALLER: Hello, is this the office for the conference?

OFFICE: Yes, that's right.

CALLER: I would like to register for the conference.

OFFICE: Do you already have a registration form?

CALLER: No, not yet.

OFFICE: I see. Then, I'll send you a registration form. Could you give me your name and address?

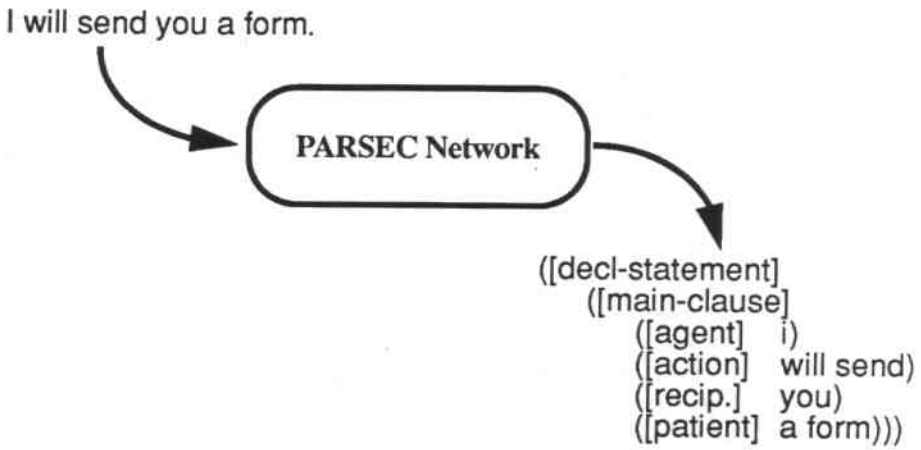
CALLER: The address is 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15236. The name is David Johnson.

OFFICE: I see. I'll send you a registration form immediately. If there are any questions, please ask me at any time.

CALLER: Thank you. Goodbye.

OFFICE: Goodbye.

FIGURE 1.1 Example input and output of PARSEC.



The conference registration dialog task (referred to later as the CR task) consists of 12 conversations using a vocabulary of slightly more than 400 words. There are over 200 unique sentences in the corpus. Appendix B contains the full text of the conversations. In addition to the text of the corpus, recordings of multiple speakers reading the conversations have been made as part of a speech recognition effort.

1.3 The PARSEC System

The PARSEC system consists of a number of tools for creating connectionist parsing networks that learn to parse from exposure to training sentences and their desired parses. A trained PARSEC network takes single sentences (presented word by word) as input, and produces a case-based syntactic/semantic interpretation as output. Figure 1.1 shows an example of PARSEC's output for "I will send you a form."

PARSEC networks are structured, modular, and hierarchically organized. They use a variant of back-propagation learning (Rumelhart, Hinton, and Williams 1986), and have been augmented with special non-connectionist mechanisms for processing symbolic structures. Building a PARSEC network requires four steps:

1. Creating a training parse file.
2. Creating a lexicon.
3. Training the network's modules.
4. Assembling the modules into a full PARSEC network.

Of these, only the first two steps require substantial human effort, but this effort is less than that required to write a grammar by hand. Writing a grammar by hand entails defining a lexicon and deciding on parses for an example set. In addition, the grammar rules must be written and debugged. The last two steps for creating a PARSEC network

(training of the modules and assembly into a full PARSEC network) are automated and require no human supervision. This is possible through the use of a constructive learning algorithm that progressively increases the power of the networks as required by the training tasks.

1.4 Performance

I characterize PARSEC's performance along three dimensions:

1. Learning ability: generalization performance on novel sentences and ease of training.
2. Noise tolerance: performance on noisy sentences (sentences including speech recognition errors and ungrammaticality).
3. Ability to utilize non-symbolic information: performance on sentence mood disambiguation where intonation is the determining factor.

1.4.1 Learning and Generalization

Clearly, any system that learns must be evaluated with respect to generalization, otherwise table-lookup strategies can masquerade as true learning. I report on several experiments designed to measure the generalization capability of PARSEC's learning algorithms.

Straightforward training of PARSEC networks results in very poor generalization performance. However, there are several ways in which knowledge about parsing and about language structure can be used to improve PARSEC's generalization performance. Using these techniques, PARSEC is able to perform well compared with traditional hand-coded grammars in tests of coverage.

Another critical issue in characterizing learning performance is how difficult it is to make the system learn well. If a learning system requires a highly specialized expert to train it, it isn't much of an improvement over a hand-coded system. This is an especially important area where connectionist learning algorithms are concerned, since they are often difficult to control and tend to be slow learners. PARSEC's learning procedures are highly automated, and an expert is not required to fine-tune parameters in order to produce a satisfactory result.

1.4.2 Noise Tolerance

Parsing is an interesting domain in which to consider the effects of noise. One doesn't see noise of the same sort as is found in signal processing where it is possible to observe subtle "smearing" or other non-catastrophic events. In the case of parsing, since the input is symbolic, the effects of noise tend to be harsh: words are inserted, deleted, substituted, or are incorrect from a grammatical standpoint. Although there has been some recent success in making grammar-based parsers more noise tolerant, many are somewhat brittle in the face of noise, and they often fail on slightly deficient input.

I evaluate PARSEC on three types of noise:

1. Noise from speech recognition errors.
2. Synthetic ungrammaticality.
3. Noise occurring in spontaneous speech (e.g. restarts and ungrammaticality).

PARSEC exhibits better noise tolerance than some grammar-based systems, but it is not an exhaustive solution to the noise problem.

1.4.3 Multi-modal Input

Synergistic combination of multiple input modalities is a desirable goal of speech processing systems. An obvious area in which this ability is required is the case of utterances where sentence mood is affected by intonation:

- "Okay."—a confirmation.
- "Okay?"—a request for confirmation.

I show that PARSEC has the ability to integrate pitch contour information with syntactic and semantic information to disambiguate mood in short utterances. This is a first step towards the goal of integrating prosodic and symbolic cues.

1.5 Outline of the Thesis

Chapter 2: Related Work

The field of connectionist language processing is fairly young, and despite a recent explosion in interest, I present a reasonably complete overview. Of course, the entire field of NLP is related, but I summarize only those systems and results that have particularly important implications for speech processing.

Chapter 3: Connectionist Parsing

Chapter 3 begins with a discussion of some of the general issues involved in an application of connectionist models to the parsing problem. The issues mainly relate to the use of symbols in a sequential task as opposed to non-symbolic static recognition tasks. Representation is discussed as it impacts learnability and performance. The methods for representing sentence structure in this thesis are also introduced. Instead of relying on recursive objects, I develop a system of non-recursive sentence representation that is more amenable to direct implementation in a connectionist network.

A number of computational issues are also discussed. Simple actions such as assignment, which aren't issues in modern programming languages, are more complex in connectionist networks. I introduce the idea of adding structure in terms of additional hardware to a network in order to facilitate better symbol processing.

The last part of the chapter describes an early connectionist parsing architecture that was the basis for this thesis work. I discuss some of the weaknesses of this early architecture, and this provides a foundation from which to define the refined PARSEC architecture.

Chapter 4: PARSEC Architecture
This chapter introduces the conference registration task, and I describe PARSEC architecture using the CR task as an example domain. Most of the chapter is a detailed description of the six modules of the baseline architecture. I also describe enhancements to the baseline architecture that were made during the course of the work. Lastly, I show an actual PARSEC network parsing some sentences and discuss its dynamic behavior.

Chapter 5: PARSEC Training
I present the Programmed Constructive Learning algorithm that was developed to enhance PARSEC's generalization ability. PCL is embedded in a three phase training procedure that controls all learning parameters automatically. This enables non-experts to train the modules of PARSEC. I summarize the four steps involved in producing a PARSEC network. I also discuss an experiment in which a non-expert successfully trained a PARSEC network for a novel task. A successful application of PARSEC to a new language (German) concludes the chapter.

Chapter 6: Generalization Experiments
This chapter is devoted to describing PARSEC's generalization performance. First, I describe the evaluation procedure, then analyze the poor performance of the first PARSEC parser for the CR task. I show three additional CR parsers, each with progressively better performance. By incorporating the techniques described in Chapters 4 and 5, the final version of PARSEC is shown to produce a network with nearly 70% generalization performance.

To provide points of comparison, the performance of three hand-coded grammars is analyzed on the same CR generalization task. Two of the grammars were constructed as part of a contest with a large cash prize for best generalization. The generalization performance of the hand-coded grammars ranges from 5% to 38%—a favorable result for the PARSEC system, whose knowledge of English is far more restricted than that of the human grammar-writers.

Chapter 7: Speech Translation and Noise Tolerance
This chapter reports PARSEC's performance on noisy input. I present an application of PARSEC to the JANUS speech-to-speech translation system. Using data that include speech recognition errors, a comparison is made between PARSEC and a hand-coded grammar implemented using an LR parser. The PARSEC network outperforms the LR parser on particularly noisy speech recognition output, but it shows an inability to assign preferences to competing sentence hypotheses. I also cover experiments using synthetic ungrammatical sentences and a limited experiment involving transcribed spontaneous utterances.

Chapter 8: Prosodic Information
Here I report on an experiment in which a PARSEC network is augmented to utilize pitch information from speech signals in order to disambiguate the mood of short utterances in which syntax alone is insufficient.

Chapter 9: Conclusion
I summarize the results and contributions of the thesis. I discuss the shortcomings of PARSEC, and present some ideas about future work with particular attention to extending PARSEC to larger parsing tasks.

Appendix A: Network Formalism
This appendix contains the equations that define the behavior of connectionist units in the networks used throughout this thesis.

Appendix B: Conference Registration Dialogs
Contains the full text of the CR task's 12 conversations.

Appendix C: Conference Registration Testing Corpus
Contains sentences used for the generalization tests of Chapter 5.

Appendix D: ATIS Sentences
Contains the sentences of the ATIS task used for the automatic learning test and some noise experiments.

Appendix E: Pitch Experiment Data
Contains some examples of the data used in the pitch augmentation to PARSEC.

2 Related Work

A large body of work is related to this thesis, ranging from other connectionist models to stochastic approaches to language processing. This chapter reviews the major connectionist work in the area and places the pieces of research in a broad context. In spite of a recent explosion of interest in connectionist NLP, the coverage is fairly complete. Of course, many non-connectionist systems have been proposed that address some of the issues in speech processing that PARSEC targets. I will briefly review some of these systems as well. The chapter concludes with some remarks about PARSEC's relationship to the reviewed work.

2.1 Early Work In Connectionist Natural Language Processing

One of the major thrusts was implementing formal grammar systems in parallel networks. Other early work in connectionist NLP was quite diverse, with emphasis on semantic issues as well as learning.

2.1.1 Connectionist Networks as Implementation Devices

These systems placed strong emphasis on exploiting parallelism and cooperative computation, with little emphasis on learning. They implemented well-known formal grammar systems using parallel connectionist hardware.

Selman (1985; Selman and Hirst 1985) developed a parser based on a Boltzmann-machine-like connectionist formalism in which grammar rules were directly incorporated into a connectionist network. The bottom layer of the network contained units that corresponded to the terminals of a context free grammar. A network representing the parse trees of all non-terminal strings not exceeding the length of the input was connected to the bottom layer—similar substructures were not redundantly represented.

Two types of units were present: *main* units representing the terminal and non-terminal symbols of the grammar and *binder* units representing how the main units could be related. A string was parsed by clamping the appropriate input units and running the stochastic network according to an annealing schedule. Selman's model showed that connectionist systems could implement rule-based processing and offered advantages due to their parallel nature.

Fanty (1985; 1986) developed a technique for producing a *deterministic* connectionist parsing network given any context-free grammar without epsilon productions. A network corresponded to a particular grammar parsed grammatical strings of a fixed (arbitrary) length quickly and deterministically. Fanty's networks contained three types of interconnected units: terminal units, non-terminal units, and match units. The former two types corresponded to the symbols of the grammar in each possible position. Match units essentially implemented the productions in the grammar. A string was presented to the network by stimulating the appropriate terminal units in the proper positions. After a fixed number of update cycles (the upper bound was linear in the length of the input), the active units of the network represented the parse of the input string. Fanty's system exploited the leverage of parallelism without requiring a lengthy relaxation process. He also explored some schemes for disambiguating multiple parses and limited learning of new grammar productions.

Charniak and Santos (1987) proposed a parsing system (CONPARSE) that strains the notion of what is "connectionist." They constructed a parsing network from a context-free grammar that required neither the length limitation nor the lengthy relaxation process of the systems described above. A parser was represented as a table of fixed width and height with processing units occupying the slots. The parser shifted in parts of speech from the lower right corner of the table. Units representing non-terminals and bindings between them were updated for a fixed number of cycles for each input token. The left and top portions of parses of long sentences were shifted out of the table and were concatenated onto the final parse. The process was deterministic and quite fast. However, due to the fixed table size, the parser could not properly process all sentences with center-embedded constructions.

Each of the systems described in this section involved implementing a formal syntactic grammar system in a connectionist formalism. These systems did not address semantic language issues nor did they acquire their grammars. However they demonstrated that parallel networks using simple processing units could provide leverage in syntactic parsing tasks.

2.1.2 Other Early Connectionist Models of Language

Spreading Activation Models

Cottrell's work (1989; Cottrell and Small 1983) focused on word sense disambiguation. He used a spreading activation/lateral inhibition model to combine semantic and syntactic information. There were four components to his system: the lexical level (each word had an input unit), the word sense level (each word unit from the previous level connected to a number of word senses), the case level (fine-grained semantic case roles), and the syntax level (intended to interact with the case level, but not well-developed). After stimulating lexical units, the other units in the network were allowed to react and

compete for activation. Information spread both laterally within each level as well as vertically to allow for interaction between different levels. The interactive activation process eventually led to an interpretation of the appropriate word senses as well as case-role bindings. This work scratched the surface of possibilities in using connectionist models for semantic language tasks.

Waltz and Pollack (1985) followed Cottrell's work. They agreed with Cottrell that all aspects of language interpretation should interact at each stage of processing. Using a similar spreading activation/lateral inhibition paradigm, they constructed a model in which context, syntax, and semantics were combined to produce semantic interpretations. For example, the sentence "John shot some bucks" has different meanings in the context of gambling versus hunting. Their system used activation between related nodes (e.g. *verb* category for "shot" was linked to verb senses "fire" and "waste" and *hunting* context was linked to "fire"). Inhibitory links were present between the sense nodes for each word (e.g. "fire" and "waste" inhibit each other). Thus, in the context of *hunting*, the sentence was interpreted to mean that John fired a gun at some male deer and struck them. They proposed that a microfeature representational scheme could be used for the activation/inhibition processes.

Models that Learned

McClelland and Kawamoto (1986) used semantic feature representations as input and output for a case role assignment network that processed single sentences. They trained a two layer network to produce case role assignments for simple single clause sentences consisting of a subject, verb, object, and an optional prepositional phrase. The network was trained to perform some lexical ambiguity resolution as well. Despite the simplicity of the task, the experiment was an important advance for connectionist models of language. The networks learned to combine both semantic and syntactic information to produce interpretations of sentences with some interaction among the constituents.

The introduction of back-propagation (Rumelhart, Hinton, and Williams 1986) offered a method to train deterministic multi-layer networks to perform very complex mappings. Hanson and Kegl (1987) developed an auto-associative connectionist model called PARSNIP. This contrasted with the supervised model of McClelland and Kawamoto. The input (and output) to PARSNIP were syntactically tagged sentences from the Brown Corpus. The architecture was a simple three layer back-propagation network that required a 7:1 compression to encode the input pattern across the hidden units. The trained network generalized quite well and exhibited a number of interesting features including: completion of sentence fragments, preference for syntactically correct sentences, and recognition of novel sentences. PARSNIP was also able to reproduce novel sentences with one level deep center-embedded patterns but had difficulty with deeper embeddings that trouble human language users. Using a simple architecture with unsupervised training, PARSNIP captured several aspects of English language structure.

2.2 Recent Work in Connectionist NLP

This section begins with some extensions to the work above—alteratives to the parallel network implementation of grammars and further development of the microfeature representational scheme. The remainder of the section includes recurrent network architectures and hybrid schemes.

2.2.1 Extensions of Early Work

Nijholt (1990) further developed Fanty's style of connectionist parsing. Nijholt defined *meta-parsing* as the process of building a connectionist parsing network for a particular grammar. He showed how to construct a connectionist Earley parser using this idea. He placed more emphasis on network construction than on parsing, but the resultant networks had the same properties as those of Fanty—speed and determinism. Nijholt's approach was still subject to a limitation on input length. As with the earlier work, learning was not the focus in Nijhholt's work.

Howell's (1988) work on VITAL was similar to Fanty's work and that of Selman. However, Howell's parser *dynamically* generated a network as needed during the parsing process. Nodes in VITAL were "templates" containing a single grammar rule plus pointers to related rules and likelihoods of rule application. An input sentence was processed by instantiating highly active terminal nodes corresponding to the words of the sentence. Additional nodes were added as indicated by the information in the templates. Activation values varied during a relaxation process where nodes competed for activation as it spread through a network, both bottom-up and top-down. Early in processing, networks grew rapidly. Then, nodes were pruned as they lost activation to more plausible alternatives. Ultimately, VITAL would settle on a single parse. Nakagawa and Mori (1988) produced a similar model based on sigma-pi units.

Gallant (1990) extended the work of Waltz and Pollack with respect to microfeature representations. He supported using feature vectors to represent semantic context instead of relying on a complex spreading activation process. In Gallant's model, words in a lexicon had some number of integer valued feature numbers representing the extent to which the words were associated with a particular feature (e.g. human). These features were entered by hand or from a lexicon. He defined a computationally simple dynamic context vector for use in word sense disambiguation. Disambiguation was possible using comparisons of the inner products of word sense feature vectors with the current dynamic context vector. By virtue of being computationally tractable, Gallant's scheme could be of pragmatic importance in a number of language problems.

Miikkulainen and Dyer (1989) showed how to learn input/output representations of the sort introduced by McClelland and Kawamoto to represent words in a case-role assignment task. They introduced the FGREP architecture. In FGREP, initially all words had random real-valued feature values. The representations were modified by extending back-propagation into the lexicon. The input and output of their networks changed until the case-role assignment task was learned. The I/O representations reflected the usage of words in the task to be learned. Through hierarchical cluster analysis, they showed that the words formed sensible clusters (e.g. humans formed a cluster and breakable objects formed a cluster). FGREP offered a method whereby microfeature representations could be automatically produced. However, FGREP required a large amount of training data to form useful representations.

2.2.2 Recurrent Networks

Elman (1988, 1990) introduced the Simple Recurrent Network—an innovation that produced a number of interesting results in connectionist NLP. An SRN is simply a standard back-propagation network with an augmented input layer that includes the

previous activation pattern of the hidden layer. An SRN processes *sequences* of input tokens, and is able to learn to encode temporal context in its hidden units' representations. Elman's initial experiments were simple prediction tasks, but he showed that the SRN could learn to encode complex concepts such as lexical classes.

Elman (1989) evaluated the SRN with respect to what types of language phenomena were learnable. He showed that the SRN could model aspects of sentence processing that required complex structural representation. For example, an SRN was trained on a word prediction task for sentences that included relative clauses. The network learned to encode long-distance dependencies such as subject/verb agreement across relative clauses. The SRN was a simple, but general, sequential learning architecture. However, the work of Cleeremans, Servan-Schreiber, and McClelland (1989) pointed out some potential deficiencies in the memory capacity of the SRN. In particular, in prediction tasks using simple grammars, the SRN seemed incapable of encoding dependencies across context independent intervening sequences.

Miikkulainen (1990a) explored the use of the SRN to model script paraphrasing. By combining the FGREP architecture with the SRN, and adding a module that processed case-role descriptions of sentences, he was able to teach (in parallel) a set of four separate subnetworks to paraphrase simple stories. The system was able to make inferences about missing events in stories.

Miikkulainen (1990b) also applied the SRN to the problem of parsing embedded clauses. The system, CLAUSES, was able to learn to process sentences containing embeddings that were presented as fragments (i.e. clausal boundaries were tagged). CLAUSES constructed case-role representations of multiple acts. While achieving interesting results, Miikkulainen suggested that more highly structured learning paradigms might be necessary to produce satisfactory performance. He pointed out that with distributed networks, interpolation between familiar examples is natural, but recognition of an input pattern as a combination of familiar pieces is somewhat beyond them.

Allen (1991) experimented with the SRN using connectionist agents to answer questions about kinship relations. Networks were trained to answer sequentially presented questions (e.g. "whois daughterof mary"), and they showed some ability to generalize—albeit with extremely low ratios of test set size to training set size. The internal representations of the SRN during processing reflected the semantics of the questions that were asked. Allen also used Jordan's (1986) sequential recurrent networks for language tasks in CLUES (Allen 1990). Here, he trained connectionist agents to answer questions about a microworld of simple objects.

St. John and McClelland (1990) applied Jordan's sequential recurrent network to other language tasks. They trained a two-stage architecture to form "sentence gestalts" of single clause sentences and probed the gestalts for information such as case-role assignment. Later, St. John (1990) extended the architecture to process stories. Without using SRN's, Dolan (1989) developed the CRAM system, a learning system based on tensor manipulation networks. It sought to unify the symbolic and subsymbolic processing paradigms. CRAM learned to read simple one-paragraph stories and could produce a summary of the story or a plan for a character in the story.

Wang and Waibel (1991) applied the SRN to the problem of script tracking in the context of dialog understanding. Following the work of Miikkulainen and Dyer, they trained a modular network to keep track of a script of events for conference registration. Using limited initial information, networks formed useful internal representations to learn the task.

Each of these approaches involved application of a recurrent learning architecture to some aspect of language processing. While showing some potential, most of the approaches raise questions about the *amount* of training data that they require.

2.2.3 Recursive Auto-Associative Memory

Pollack's introduction of the RAAM architecture may prove to be as important as Elman's SRN (Pollack 1990). RAAM offers a way to use distributed representations of fixed width to encode recursive structures such as trees. One constructs a RAAM by simultaneously training encoder and decoder networks that push representations through a fixed-width bottleneck. One can then use hidden layer activations as new terminal symbols and thereby encode nested structures such as trees within fixed-width vector representations. This has great potential for natural language applications.

Berg (1991) applied a combination of the RAAM idea and the SRN to parsing in XERIC. XERIC used phrase templates (composed of a specifier, head, and up to two complements) to encode syntax. The input to XERIC was a feature-based word representation along with the encoding of the left context of the word (the previous activation of the hidden layer). The hidden layer would sequentially develop a compact representation of the entire sentence. A decoder module was able to iteratively unroll the full structure of the parsed sentence as a series of phrase templates. XERIC was unable to handle a number of constructions (e.g. adjectives in NPs), and it had difficulty remembering and reproducing the features of individual words. However, it was one of the first connectionist language processors that could handle recursive language structures of varying length within a fixed architecture.

2.2.4 Hybrid Systems

Kwasny and Faisal (1990) developed CDP, a hybrid extension to Marcus's PARSIFAL system (Marcus 1980). They trained a back-propagation network to indicate actions to be performed on the usual symbolic data structures of a PARSIFAL parser (an input buffer and a stack). CDP was trained in two ways—deductively and inductively. In the deductive training scheme, rule-templates were created from a grammar, and the network learned to associate actions with the templates. In the inductive scheme, traces of grammar-based parses were used, and the network learned to capture the behavioral characteristics of the rules from the grammar used for the trace. The deductive scheme generally had higher performance. They showed some noise tolerance in their network-based parser over that of the purely grammar-based parser, whose actions their network was trained to reproduce.

Santos (1989) extended his earlier work with Charniak (Charniak and Santos 1987). He augmented the CONPARSE architecture with PALS (Parsing and Learning System). PALS adjusted weights associated with rules in CONPARSE. The rules were derived from a grammar, and PALS learned how to apply the rules. Sentences of arbitrary length

could be parsed, but some embedded constructions were not parsable due to fixed table size. As with CDP, the idea was to improve upon a grammar using learning.

2.3 Other Non-Traditional Systems

The two systems covered in this section are not strictly connectionist, but they are not traditional approaches to language processing either.

Phi-DmDialog was a parallel marker-passing system that was applied to the conference registration dialog task (Kitano *et al.* 1989). It lay somewhere between the symbolic approach and the connectionist approach. The system was organized into a hierarchy of levels, beginning with morphophonetics at the bottom and proceeding up to plan hierarchies. Information flowed using several types of markers that spread, collided, and combined to produce side-effects. The side-effects processed the input, produced the output, and integrated predictive information with observed input. The system did not employ learning, although it showed a new approach to information combination among interacting data sources.

Gorin *et al.* (1990) developed a system that automatically acquired a language model for a particular task from semantic-level information. The system had no predefined vocabulary or syntax, but it learned to perform a small number of actions based on interaction with users. The system was demonstrated on a department store inward-call management task where the actions were telephone connections to any of several departments. Words and phrases were represented in the architecture as nodes. The lowest layer of nodes corresponded to words, the next level to word pairs (phrases), and the last layer to the semantic actions that the system could take. Connection weights between nodes denoted a measure of mutual information. For example, the connection between the phrasal node "buy dress" and the action "connect to women's clothing" acquired a strong weight. Using mutual information for connection weights allowed for single pass training. Although currently unable to handle complex language constructs, the approach shows promise as an alternative to both the pure PDP paradigm and traditional NLP systems.

2.4 Symbolic Systems

NLP is a vast field, and there are many current research efforts that target phenomena relevant to this thesis. However, I will only review a few key approaches here.

2.4.1 MINDS

The MINDS system (Multi-modal INteractive Dialog System) was developed at CMU (Young *et al.* 1989). The goal of the research was to incorporate high-level knowledge sources into speech recognition systems to make them robust and easily usable. MINDS operated on a resource management task where a user was querying a database verbally. The system sought to directly reduce the search space for the speech recognition system by making predictions about utterances based on high-level expectations. By modeling the current state of an interactive dialog, the system would generate a temporary reduced lexicon for use by the speech recognition system. In MINDS, predictive models of the

domain, goal trees, and a user's state-of-mind were constructed by hand. MINDS achieved substantial perplexity reductions, and the system enhanced speech recognition results.

2.4.2 Statistical Models

Statistical augmentations to symbolic systems offer the potential for more robust disambiguation, better language modeling, and faster parsing. They also avoid some of the computational cost of connectionist learning algorithms. However, they seek to *refine* the behavior of hand-coded systems through training, not to *eliminate* hand-coding.

Fujisaki *et al.* (1991) were able to augment a context-free grammar with probabilities. They devised efficient versions of existing parsing algorithms to perform the probability estimation and to implement the parser once the probabilities were known. They demonstrated fairly robust parse disambiguation on sentences from several sources using only probabilistic syntactic rules.

Seneff (1989) developed a different technique in the TINA system. A context-free grammar was converted into a shared network structure. Probabilities were associated with arcs in the network. The approach described above attached probabilities to the actual rules of the grammar. By instead attaching probabilities to arcs, different probabilities could be associated with identical units that were present in different contexts. For example, the probability of an adjective following a determiner was sensitive to whether or not the determiner was sentence-initial. The probabilities helped with perplexity reduction in speech tasks as well as with disambiguation. TINA was quite successful in its application to the SUMMIT and VOYAGER speech applications (Zue *et al.* 1990a,1990b).

Church *et al.* (1991) have proposed that simple collocational constraints should play a more important role in natural language parsers, and that semantic modeling may not always be necessary. By computing mutual information statistics between words in syntactically tagged corpora, they show that it is possible to learn, for example, what things are likely to be objects of "drink." Interestingly, some of the same types of effects can be captured in some of the connectionist models discussed so far, albeit on a smaller scale (e.g. Waltz and Pollack 1985).

Magerman and Marcus (1991) developed a stochastic parser called Pearl. They incorporated a number of probabilistic components (e.g. a part-of-speech recognition module and an unknown word module) into a coherent chart-parsing framework. Pearl estimated context-sensitive probabilities from training data and was more sophisticated than Seneff's TINA system in its use of probabilities. For example, Pearl took into account the probabilities of lexical interpretations (e.g. "love" as a verb versus noun) in addition to other context in calculating the likelihood of a particular interpretation of a sentence. Magerman and Marcus reported favorable preliminary results using data from the VOYAGER system.

2.4.3 Prosody

Steedman (1991) introduced Combinatory Grammars as a possible method for making use of intonational information along with syntax in grammar-based parsers. He fol-

lowed the notation of Pierrehumbert (1975) for annotating sentences with intonational information. In many cases, intonational contour is of help in disambiguating multiple parses. However, as yet, it is not possible to automatically extract the proper intonational annotations from real speech.

Huber (1989) investigated the use of prosodic information extracted from real speech in parsing. He showed how prosody could be exploited to benefit an island-driven parsing strategy. He developed a speech segmentation algorithm to produce *intonational units*. From these, he identified areas of prominence that were used as reliable islands from which further processing spread. The system required extensive programming and design of appropriate segmentation algorithms and information usage strategies, but it was one of the first significant efforts in using prosodic information in parsing.

2.5 Where does PARSEC fit in?

PARSEC is only tangentially related to the early work in connectionist parsing that recast grammar-based formalisms into parallel networks as a method of implementation. PARSEC shares the parallelism but places a strong emphasis on learning.

The work of McClelland and Kawamoto (1986) was the basis of my very early experiments in connectionist parsing. I adopted a similar case-role parsing task and extended it to sequentially parse real sentences (Jain 1989). The current work represents the third generation of parsing models that I have developed. Each generation brought additional structure and more complicated tasks. Though PARSEC does not attack the semantic aspect of language processing with the same vigor as some of the other early connectionist work, it shares the notion of combining syntax and semantics in a unifying framework.

Kwasny and Faisal's CDP system is the most closely related to this work in spirit. We both use symbolic manipulation in combination with subsymbolic computation, but there is a key difference. CDP requires the existence of a hand-coded grammar from which to induce (or deduce) a connectionist grammar. PARSEC induces its grammar from example parses without any existing rule-based grammar to model.

From a philosophical perspective, the work that has sprung from Elman's Simple Recurrent Network is the most distant from PARSEC. PARSEC emphasizes structure and explicit symbol manipulation/representation to boost performance, but work with the SRN has emphasized distributed representations using architectures with minimal predefined structure. These paradigms are not irreconcilable, and each will likely begin to borrow from the other. Miikkulainen's work using the SRN for processing sentences with embedded clauses was a step in that direction. The next chapter will discuss the issue of structure in more detail.

Touretzky (1991) defined a hierarchy of connectionist models based on the types of mapping they learn:

- Categorizers: these are essentially N class forced-choice recognition machines.
- Associative memories: these models form internal class-based representations of patterns.

- Pattern transformers: these have a very large number of output patterns (exponential with length of input); most possible output patterns can't be seen in training.
- Dynamic inferencers: these models exhibit novel intermediate states in response to novel inputs; they can compose new combinations of information; the structure of the models reflects the systematicity of the compositionality requirement.

They key difference between pattern transformers and dynamic inferencers is that the former require an unreasonably large proportion of possible patterns as training examples because they are not able to break problems down into subparts and reassemble the solutions. A dynamic inferencer, by virtue of having dealt with the problem of compositionality, requires far fewer training examples. PARSEC may not be a full-fledged dynamic inferencer, but it does not fall cleanly into the other categories either. The conclusion addresses this question in more detail.

3 Connectionist Parsing

This chapter is an introduction to connectionist parsing as implemented in the PARSEC system. I will introduce some general issues facing connectionist parsing systems and present the major design choices regarding representation and type of connectionist computation that were made for this work. A connectionist parser developed for a non-speech problem will be presented to illustrate the basics of the approach to parsing taken in this thesis.

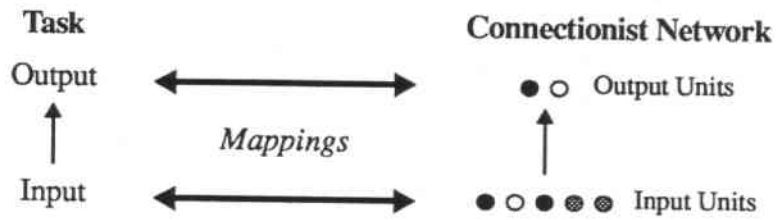
A word about the central focus of this work is in order. As mentioned in the introduction, I am primarily concerned with demonstrating that connectionist computational models are capable of solving problems in parsing that other more traditional methods are unable to do. The focus is not on modeling human learning performance or understanding how humans process language. Thus, questions about biological plausibility or comparisons to human performance are not a major concern. In this thesis, connectionist models are viewed as a tool to solve various problems in parsing. That they are to an extent biologically inspired is interesting, but not crucial. In the discussion that follows, performance concerns will dictate the design choices.

3.1 Connectionist Parsing

Parsing is an inherently symbolic and sequential task. Both aspects make it a particularly difficult task to approach using connectionist computational models, which are more easily applied to static recognition tasks. In particular, several issues arise:

- How should symbols be represented?
- How should symbols be manipulated?
- How should temporal context be captured?
- Is recurrence necessary or beneficial? What kind?

FIGURE 3.1 Task to network mappings in a connectionist network.



There are some additional considerations stemming from the problems of training connectionist networks. This work involves relatively small corpora of text (on the order of hundreds of sentences), but the networks required are quite large (tens of thousands of connections) relative to our current computational resources. Achieving good generalization from small training sets and minimizing training time for large networks are key concerns of this work.

3.1.1 Representation

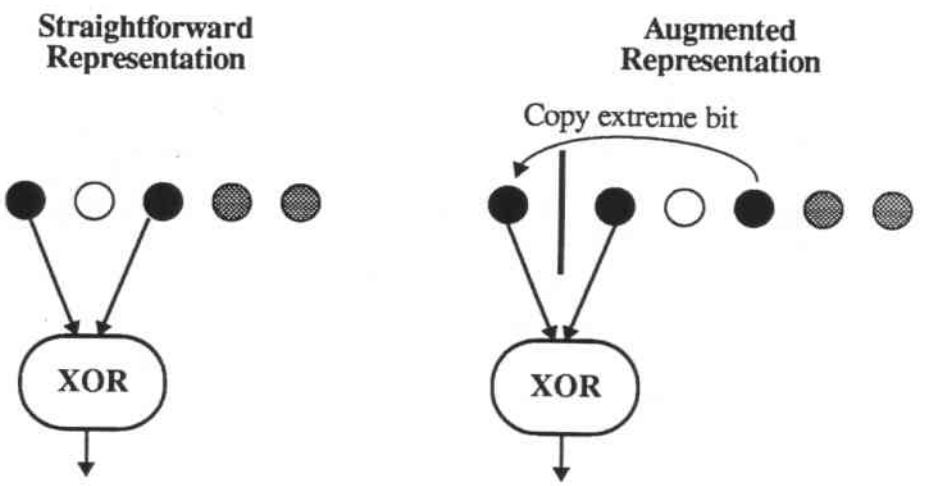
In any connectionist network that learns some task through a supervised procedure, one must decide how to represent the input and output of the network. This involves setting up bi-directional mappings from the task's input and output to patterns of activation across input and output units in a network (see Figure 3.1). The choice of mapping can have a significant impact on many aspects of network performance (e.g. learning speed and generalization performance).

In tasks where similar input patterns are to be mapped to similar output patterns, back-propagation networks will tend to generalize well. Some tasks that do not have this property can be modified by simply remapping the input patterns. An example will illustrate this point. Imagine a simple augmentation to the XOR problem. Instead of two binary inputs, there are five inputs, each with three possible values: 0, 1, and 0.5. The first few inputs take on extreme values, and the remaining ones have a value of 0.5. The task is to produce the XOR of the first value and the rightmost value that is not 0.5. The intervening values are to be ignored.

Figure 3.2 shows two representations of the problem. In the straightforward representation, the network must learn to identify the extreme bits, ignore the intervening bits, and produce the XOR of the first and last bits. One can augment the representation by adding another unit whose value is that of the last non-0.5 unit. With this new representation, the network need only learn to XOR the two important bits. The second input representation provides an easier task to learn. This is not a statement about whether or not it is *right* (in some sense) to use the augmented representation. This is only an illustration that representational changes can have an impact on the learnability of tasks. The effects of representational modifications on generalization will be shown in Chapter 6.

In a connectionist parsing network, choices about the input word representation and the output parse representation are critical to ensure adequate performance.

FIGURE 3.2 Alternate representations of augmented XOR problem.



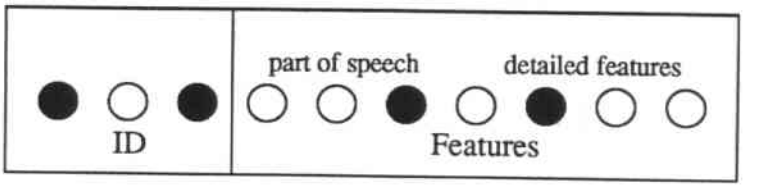
Word Representation

How should words be represented in a connectionist parsing network? The first question to answer is whether the representation should be learned or should be encoded from some external lexicon. The answer depends on the parsing task, and in particular how much data is available. For computationally tractable tasks with a sufficient amount of training data, it is possible to learn word representations through their usage. Various approaches have been explored (see Chapter 2 for more details), ranging from simple recurrent networks to modification of input word representation by extending back-propagation (Elman 1988; Miikkulainen and Dyer 1989).

If training data is not abundant, one faces a serious problem of undergeneralization. For example, in any reasonable representation, the words "a" and "an" should have the same or very similar representations (ignoring the letter-name sense of "a"). Suppose though, that "a" and "an" do not appear in the same contexts in a training corpus because of its size. It is likely that a network will arrive at different representations for the two words. This will adversely affect the performance of the network when it is required to perform on novel input. By making use of existing lexical knowledge, one can avoid the expense of acquiring such information through training and ensure that the word representations are uniform and general.

The tasks that I am interested in are designed for use in spoken language processing. Consequently, the training corpora are limited in size and vocabulary complexity. This makes learning good word representations difficult, but it also reduces the burden of hand constructing lexicons of word representations. For this work, I have used hand-crafted word representations.

FIGURE 3.3 Word representation.



In cases where it is necessary to design and construct a lexicon for the words in a task, several new issues arise:

- What features should be used?
- Tight encoding or loose encoding?
- How should multiple word senses be handled?

Figure 3.3 shows the structure of the word representation. The meanings of words are encoded as prespecified feature patterns (similar to McClelland and Kawamoto 1986). These can, in principle, contain any type of information, but for now, assume that the information is primarily syntactic and is encoded as binary patterns. The word patterns are divided into two parts: the *identification* part and the *feature* part. The ID part is an arbitrary tag, and the feature part encodes the meaning of the word. The network is only able to "see" the feature parts of the words. This prevents parsing networks from learning overly specific rules about particular words instead of general rules about major parts of speech.[1]

The encoding is fairly "loose"—a single feature bit does not change the meanings of a number of other feature bits.[2] The feature bits have both gross features (e.g. noun, verb, adjective...) and detailed features (e.g. plural, first person, proper...). Multiple word senses (e.g. "refund" as a verb versus a noun) are handled by superimposing the different feature patterns (logical OR). This would not result in meaningful feature patterns were it not for the loose encodings. PARSEC networks do not explicitly indicate which word sense is being used in their output; it is implicit in the parse representation and can be inferred easily. Of course, in more substantial domains where lexical ambiguity is a more serious problem than in speech domains, more effort would be required to handle this problem. (See Chapter 9 for more discussion of lexical ambiguity.)

1. It is possible to allow the use of ID information, and it might be useful in cases where shades of meaning are not captured by the features of words. This has not been tried in this work.

2. There is some anecdotal evidence that tight encoding of word meaning in language tasks hurts generalization (personal communication, Ye-Yi Wang). This is related to the point about the augmented XOR problem. One wants to avoid forcing a network to work very hard to figure out the meaning of its input representation. The cost of a loose encoding is a larger network.

Given a particular representation for individual words, there are different ways to represent sequences of words. For example, a word sequence can be represented spatially across several sets of word representation units. Alternatively, a word sequence can be presented to a network by using a dynamically changing activation pattern across one group of units that encodes the meanings of single words in a sequence. I have used both types of input representation in different parsers, and this will be discussed later.

Parse representation

Certain structures are more easily modeled within connectionist networks than others. For example, nested recursive structures of varied depth (the sort that are favored by linguists in representing syntax) are not easy to represent in connectionist networks.[3] "Flat" structures are preferred.

I have chosen a case-based representation (e.g. Fillmore 1968; Bruce 1975) and have augmented it somewhat. Case representations are appealing for two reasons. They produce similar representations for syntactic variants like "Mary hit John" and "John was hit by Mary." They offer a compact way of representing several of the central semantic relationships in sentences. In domains such as human-machine interfaces or in speech-to-speech translation, it is more important to capture information about *meaning* than about syntax alone. Also, case representations are easily augmented and specialized for different domains; one can add a new case-role if there is a need for one.

The work I am reporting here involves real English sentences with complex noun phrases and verb constructions.[4] Real sentences require a representation for phrases in the parse representation, but the traditional linguistic notion of a phrase involves undesirable recursion. The representation used by PARSEC relies on a non-traditional linguistic unit—called a *phrase block*. In the following sentences, the phrase blocks are delimited by brackets:

- [The man] [gave] [his dog] [a cookie].
- [The big blue whale] [was swimming] [to the boat].

Phrase blocks are simple English phrases consisting of contiguous sequences of words (e.g. the piece of a noun phrase from the determiner to the head noun is a phrase block). They are very similar to what Abney has termed "chunks" (Abney 1991a, 1991b). Figure 3.4 shows the difference between traditional linguistic phrase structure and phrase block structure. From the connectionist perspective, the non-recursive nature of phrase blocks is their key feature. Related phrase blocks can be attached using external labels instead of using nesting. Abney has compiled psycho-linguistic and other data to support the notion of non-recursive chunks being a sensible unit of language, but I adopted phrase blocks primarily because the representation obviates a number of difficult representational issues in connectionist parsing.

3. This is not to say that it is impossible. See Chapter 2 for a description of Pollack's RAAM architecture

4. Much connectionist NLP work has avoided some of the complexity of real language by focusing on content words within single clauses (e.g. "Tarzan love Jane"). In such systems, phrase level representations are not needed.

FIGURE 3.4 Two views of phrase structure.

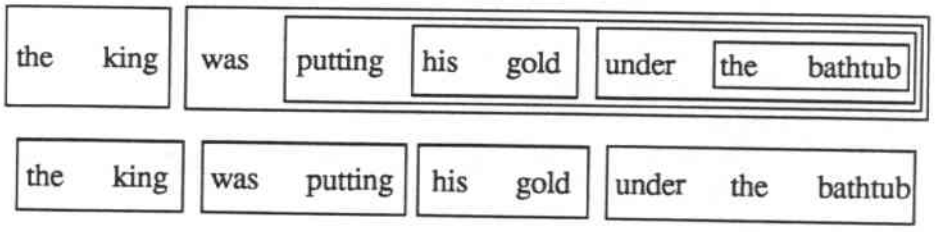


FIGURE 3.5 Alternate parse representations.

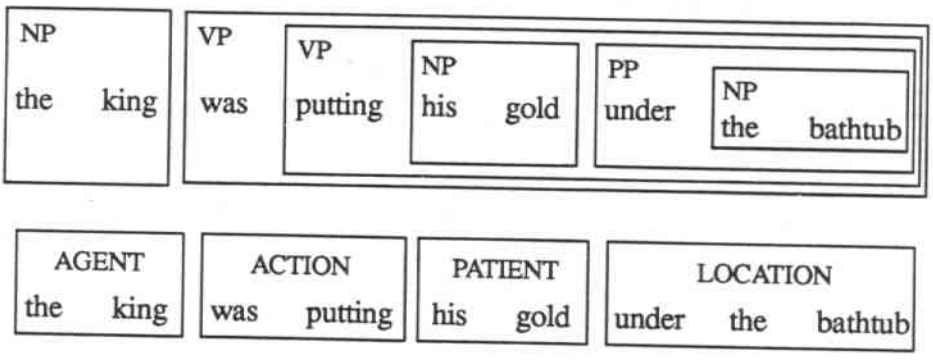


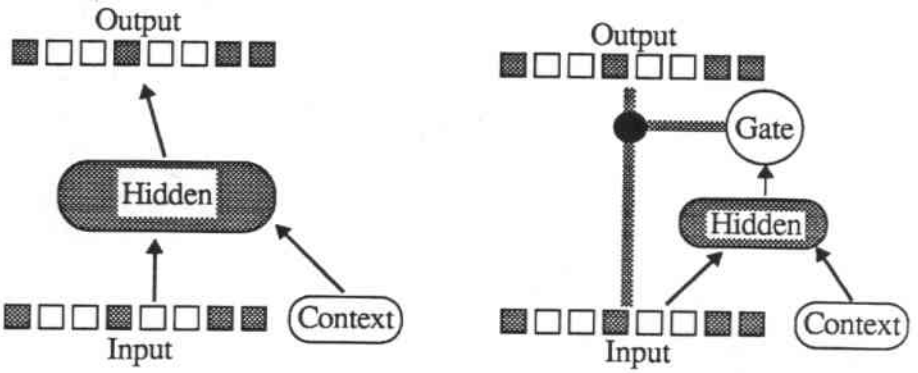
Figure 3.5 shows alternate parses of the sentences from Figure 3.4. The top parse is a more or less standard parse tree (under-specified to save space). The bottom representation is an example of the type of output that PARSEC produces. It is a simple non-recursive, semantic representation that is amenable to connectionist implementation.

3.1.2 Computational Issues

A connectionist network is a very different computing engine than LISP. In modern programming languages, questions about how to capture temporal context are easily answered. In connectionist networks, there are several ways to capture temporal context, and each has important performance consequences. Temporal context may be represented spatially as in TDNNs for speech recognition (Waibel *et al.* 1989). It is also possible for networks to learn to capture temporal context using recurrent structures (Elman 1990; Cleeremans, Servan-Schreiber, and McClelland 1989).

Consider a simple task: assignment of symbol values to slots. In a programming language, there is an assignment operator, but in connectionist networks, even after one decides how to represent symbols and slots, the notion of "assignment" is still foreign to

FIGURE 3.6 Conditional symbol assignment: two strategies.



a simple collection of units with connections. The assignment operator must be learned by the network unless some mechanism inside the network is able to perform assignment.

Network Formalism

I have developed a network formalism specifically for problems involving sequentially processed symbols (Appendix A describes the formalism, but for additional details, see Jain 1989). The major features of the formalism are:

- Well-behaved symbol buffers are constructed using groups of units.
- Units have temporal state; they integrate their inputs over time, and decay toward zero.
- Units produce the usual sigmoidal output value *and* a velocity output value. Units are responsive to both the static activation values of other units and their dynamic changes.
- The formalism supports recurrent networks.[5]

Learning is done through gradient descent using a mean-squared error measure as with standard back-propagation learning (Rumelhart, Hinton, and Williams 1986).

Symbol Manipulation

Direct support of symbol manipulation is perhaps the most controversial aspect of the formalism and deserves an illustrative example of its benefits. Figure 3.6 shows two solutions to a conditional symbol assignment problem. The pattern of activation over the input units represents a symbol that is supposed to be atomic. It is to be mapped onto the output units when a particular condition occurs over the combination of the input and context units.

5. Recurrence is explored in several parsing networks, but it does not appear in the final PARSEC architecture.

In the first case, the network must learn the condition placed on assignment of the input pattern to the output units *and* a complex auto-encoding problem to achieve the proper output pattern. In the second case, between the input and output units there are fixed-weight connections. Ignoring the gating unit's effects, the fixed-weight connections cause the output units to take on the same pattern as the input units. However, when the gating unit has low activation, the current pattern across the output units is frozen, and no activation flows from the input to output units. When the gating unit has high activation, the output units decay towards low resting values, *and* activation flows from the input to output units. The network learns the task through supervised training of the gating unit's behavior. Thus, by using the gating unit, the network need only learn the condition placed on assignment, and the symbol assignment results from the effect of the gating unit.

Training the first version of the module is more difficult, requiring more time and more data. It also has an additional failure mode. It may learn the appropriate conditions for the assignment but may undergeneralize the auto-encoding and fail to make the proper symbol assignment.[6] By supporting manipulation of symbol patterns directly, it is possible to simultaneously decrease training time and improve generalization.

3.2 Structure: Learned or Engineered

There are two approaches that lead to task-dependent structure in connectionist networks. The first is learning-based, and the second is explicit inclusion of structure by a network architect. In the learning approach, structure emerges through training general purpose architectures such as those used by Elman (1990). While such efforts have met with some success, the architectures have not been shown to be powerful enough to handle the complexity of the tasks in this work. They also tend to require a great deal of training data to show any generalization.

Certainly, it is desirable to see connectionist learning algorithms that cause complex structures to emerge spontaneously from unstructured networks. However, it seems sensible in pursuing such learning algorithms to first find out what types of structure are required or desired from a performance perspective. There is also a more pragmatic issue in building real systems. When useful information is available about a domain, it is often more efficient to simply provide the information to a network rather than expect a network to learn it. For example, in image recognition, translation invariance is easy to engineer (use weight-sharing), but it can be very difficult to learn using computationally tractable training sets.

3.3 Early Parsing Architecture

In this section, I will outline the structure of the first substantial connectionist parser that was developed during this work (Jain 1991). This is intended to make the foregoing dis-

6. This is similar to the effect described in Chapter 2 that Berg experienced with XERIC (Berg 1991). It tended to "forget" detailed features.

FIGURE 3.7 Parser's output representation of an example sentence.

```
[Clause 1: [Phrase Block 1: The dog   (RECIPIENT)]
           [Phrase Block 2: was given (ACTION)]
           [Phrase Block 3: a bone    (PATIENT)]]
[Clause 2: [Phrase Block 1: who       (AGENT)]
           [Phrase Block 2: ate       (ACTION)]
           [Phrase Block 3: the snake (PATIENT)]
 (RELATIVE: "who" refers to Clause 1, Phrase Block 1)]
```

cussion of representation and computation more concrete. The next chapter will describe PARSEC's architecture—a generalization of the architecture outlined here.

The three main goals of this early architecture were to show:

1. That connectionist networks could learn to incrementally parse non-trivial sentences.
2. How modularity and structure could be exploited in building complex networks with relatively little training data.
3. Generalization ability and noise tolerance suggestive of application to more substantial problems.

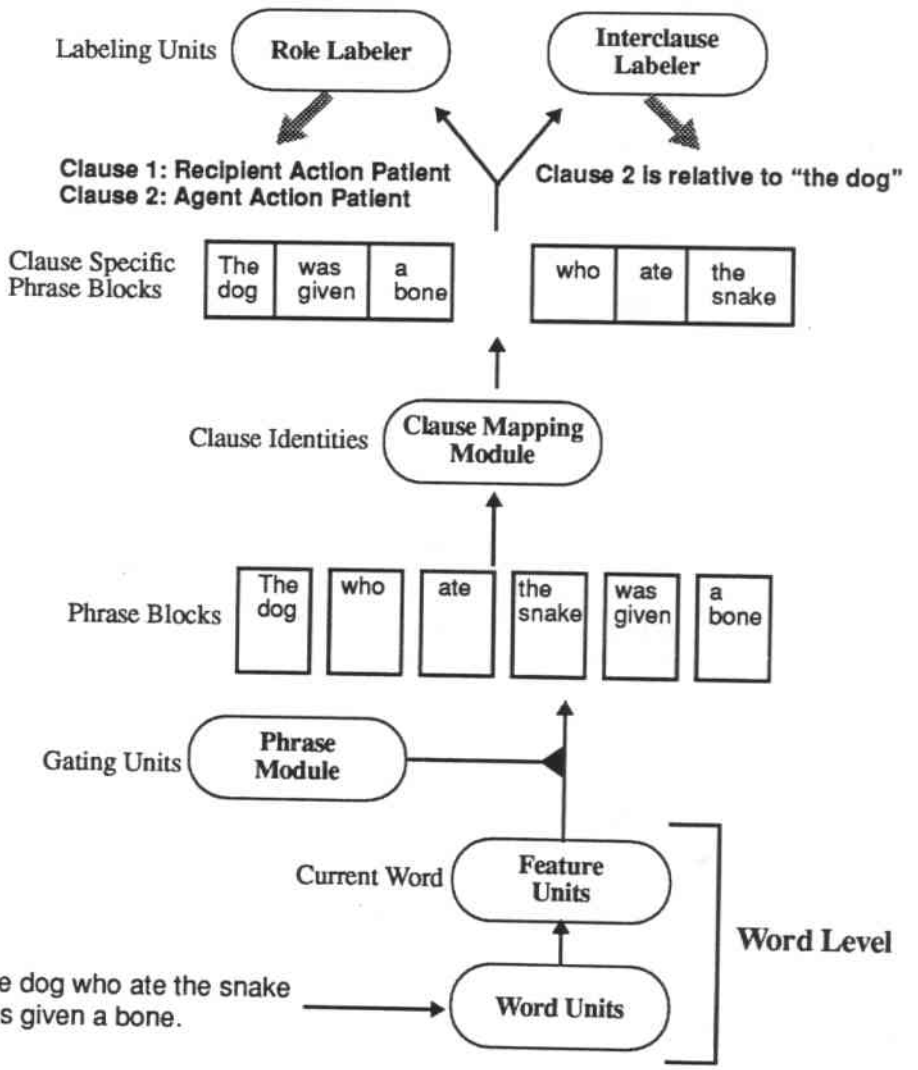
The training corpus consisted of a set of sentences with up to three clauses, including sentences with center-embedding and passive constructions. It contained over 200 sentences.[7] These sentences were grammatically interesting, but they did not reflect the statistical structure of common speech. Here are some example sentences:

- Fido dug up a bone near the tree in the garden.
- I know the man who John says Mary gave the book.
- The dog who ate the snake was given a bone.

Given the input, one word at a time, the network's task was to incrementally build a representation of the sentence that included the following information: phrase block structure, clause structure, case-role assignment, and interclause relationships. Figure 3.7 shows a representation of the desired parse of "The dog who ate the snake was given a bone." The sentence is represented as two clauses made up of phrase blocks to which role labels are assigned. The embedded relative clause is also labeled.

7. These were taken from the example set of a parser based on a left associative grammar developed by Hausser (1989)

FIGURE 3.8 Early parsing architecture.



3.3.1 Network Architecture and Data Flow

Figure 3.8 shows the network architecture. Information flows through the network as follows, beginning with the lowest level and continuing up the hierarchy. Each of the modules of this architecture uses recurrent connections between the hidden units and the output units (similar to Jordan 1986). The detailed internal structure of each module is not shown in the figure.

The *Word level* contains all of the information about word meanings that the network has. A word is presented by stimulating its associated word unit for a short time. This produces a pattern of activation across the feature units that represent the meaning of the word. As mentioned in Section 3.1.1, learning adequate word representations from

small training sets is problematic. Therefore, the connections from the word units to the feature units, which encode semantic and syntactic information about words, are compiled into the network from hand-specified feature representations.

The *Phrase module* contains gating units that learn the proper conditional assignment behavior to capture word feature patterns in the phrase blocks (see Figure 3.6 for a detailed diagram of the conditional symbol assignment task). Phrase blocks are matrices of units that contain room for up to four words. Each of the word slots in the phrase blocks (a row of units) has an associated gating unit within the Phrase module. The gating units are trained to respond to the current input word in the context of the partial sentence. The proper sequence of gating unit activations produces the following phrase block structure for the example sentence: "[The dog] [who] [ate] [the snake] [was given] [a bone]."

The *Clause Mapping module* assigns phrase blocks to clauses. For example, "[The dog] [who] [ate] [the snake] [was given] [a bone]," is mapped into "[The dog] [was given] [a bone]" and "[who] [ate] [the snake]." Each phrase block has three clause mapping units associated with it. This allows for up to three clauses in an input sentence. The unit with the highest activation "wins" the phrase block for its clause. The pattern of winners given by "1 2 2 2 1 1" produces the mapping shown in the figure. During the course of processing, the phrase blocks are remapped into clause-specific phrase blocks by subroutine according to the activation of the Mapping units. In the example, the embedded clause "[who] [ate] [the snake]" is marked as belonging to clause 2, with the remainder assigned to clause 1.

The *Role Labeling module* produces labels for the roles and relationships of the phrase blocks in each clause of the sentence (e.g. Agent, Action, Patient, etc.). For each clause, there is a matrix of units that represent role labels and attachments (X dimension corresponds to phrase block number, Y dimension corresponds to label). For the example sentence, the Role Labeling module assigns Recipient/Action/Patient and Agent/Action/Patient to the phrase blocks of the respective clauses.

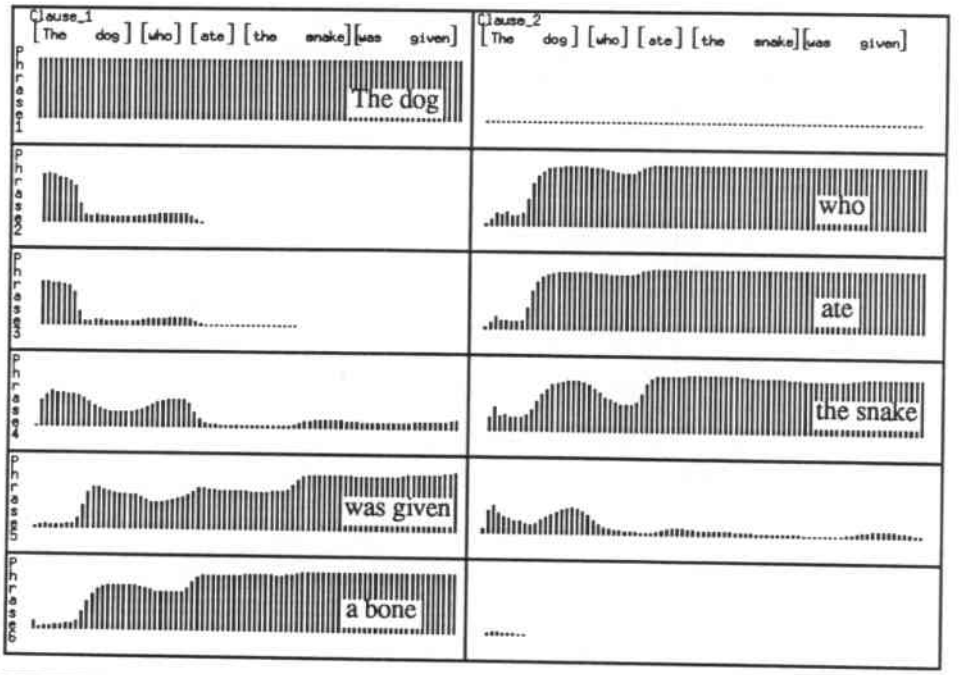
The *Interclause Labeling module* represents the interrelationships among the clauses making up a sentence. This again is represented using a matrix of units. In the example, the Interclause module indicates that clause 2 ("[who] [ate] [the snake]") is relative to phrase block 1 of clause 1 ("[the dog]").

3.3.2 Dynamic Behavior

The dynamic behavior of a trained network will be illustrated on the example sentence from Figures 3.7 and 3.8: "The dog who ate the snake was given a bone." This sentence was not in the training set of the network.

Initially, all of the units in the network are at their resting values. The units of the phrase blocks all have low activation. The word unit corresponding to "the" is stimulated, causing its word feature representation to become active across the Feature units. The gating unit associated with slot 1 of phrase block 1 becomes active, which in turn causes the feature representation of "the" to be assigned to the slot. The representation for "the" (and each subsequent word) is stimulated for 10 network update cycles to allow for adequate processing time within the recurrent modules. The gate closes as "dog" is presented. As the gate is closing, the gate for slot 2 of phrase block 1 recognizes "dog" as

FIGURE 3.9 Clause Mapping dynamic behavior.

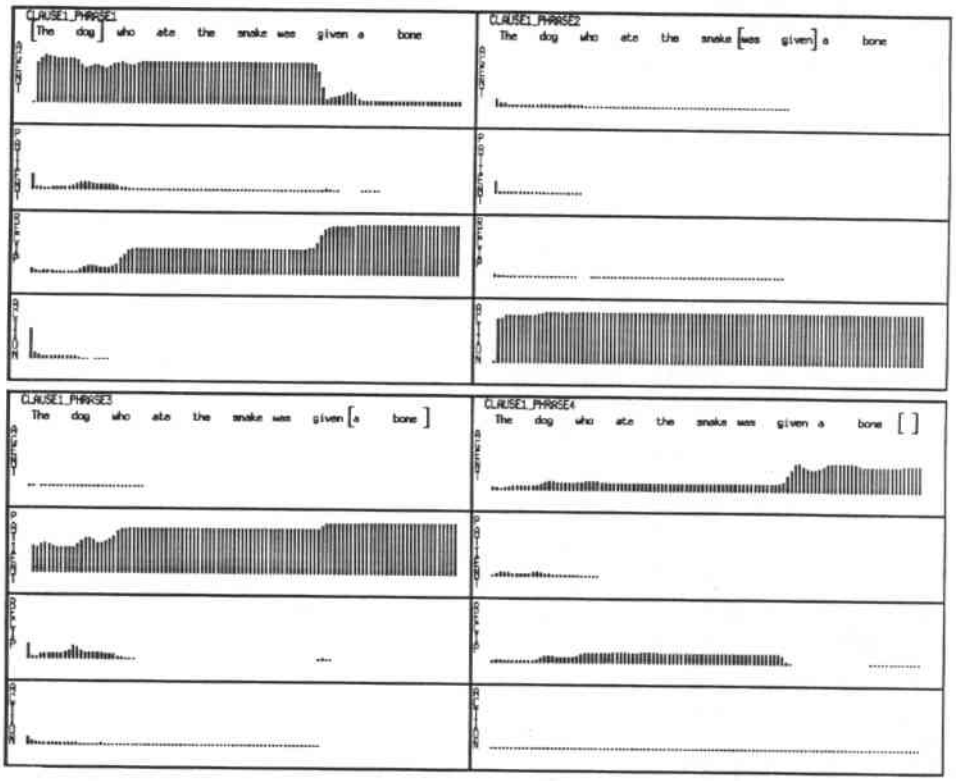


part of the current phrase block, and it opens. When "who" is presented, the gate for slot 1 of phrase block 2 recognizes it as beginning a new phrase block. The remaining words of the sentence are processed similarly, resulting in the final representation shown in Figure 3.8. While this is occurring, the higher levels of the network are processing the evolving representation across the phrase blocks.

The behavior of some of the output units of the Clause Mapping module is shown in Figure 3.9. The figure shows the time-varying activations of a 6x2 subset of the 10x3 Clause Mapping matrix. In the figure, the columns correspond to clause number and the rows to phrase blocks. Early in the presentation of the first word, the activation levels of the second column of units for phrases 2–4 rise sharply. The Clause Mapping module is hypothesizing that the first four phrase blocks will belong to the first clause—reflecting the dominance of single clause sentences in the training set. After "the" is processed, this hypothesis is revised. The network then believes that there is an embedded clause of three (possibly four) phrase blocks following the first phrase block. As a hypothesis is revised, the clause-specific phrase blocks immediately reflect the new interpretation of the sentence fragment.

The predictive behavior emerged spontaneously from the training procedure (a majority of sentences in the training set beginning with a determiner had embedded clauses after the first phrase block). The next two words ("dog who") confirm the network's expectation. The word "ate" allows the network to firmly decide on an embedded clause of

FIGURE 3.10 Dynamic behavior of Role Labeling units.



three phrase blocks within the main clause. This is the correct clausal structure of the sentence and is confirmed by the remainder of the input. The Interclause level (not shown in Figure 3.9) indicates that the embedded clause is relative to the first phrase block of the main clause. This happens at the same time that the clause module predicts the embedded clause.

The Role Labeling module processes the individual clauses as they are mapped through the Clause Mapping module. The output units for clause 1 initially hypothesize an Agent/Action/Patient role structure with some competition from a Recipient/Action/Patient role structure (the role labeling units' activation traces for clause 1 are shown in Figure 3.10). This prediction occurs because active constructs outnumbered passive ones during training. The final decision about role structure is postponed until just after the embedded clause is presented. The input tokens "was given" immediately cause the Recipient/Action/Patient role structure to dominate. The network also indicates that a fourth phrase block (e.g. "by Mary") is expected to be the Agent (not shown). For clause 2 ("[who] [ate] [the snake]"), an Agent/Action/Patient role structure is again predicted; this time the prediction is borne out (not shown).

3.3.3 Discussion of Performance

I have outlined a modular, hierarchical connectionist network architecture that learns to parse using back-propagation. It is able to parse complex sentences, including passive constructions and center embedded clauses.

The key features of the dynamic behavior are:

- The network successfully combines syntactic, semantic, and word order information.
- The network is predictive.
- The network responds quickly to right context information.
- Uncertainty is manifested by competing units. Units that correspond to different interpretations of a single constituent show opposing activity, which sometimes oscillates.

Analysis of generalization performance and noise tolerance for this network were not detailed, but some suggestive results were obtained. Novel sentences whose feature representation corresponded to training sentences (excluding ID bits) were processed correctly, as expected due to the word representation. In sentences with novel syntax (overall feature representations *not* from the training set), performance was not as good. Substitution of single words in training sentences resulting in meaningful novel sentences was tolerated almost without exception. However, substitution of entire phrase blocks caused some errors on structural assignment.

The trained network correctly processed sentences in which verbs were made ungrammatical (e.g. "We am happy."). More substantial corruptions often produced reasonable behavior. For example, the sentence, "[Peter] [was gave] [a bone] [to Fido]," received an Agent/Action/Patient/Recipient role structure. This corresponded to an interpretation of "was gave" as "gave" or "has given." Single clause sentences in which determiners were randomly deleted (to simulate speech recognition errors) were processed correctly 85% of the time. Multiple clause sentences produced more errors.

This architecture is limited in some respects. The network was extremely difficult to train. There was no general method that produced robust convergence for each of the modules. Decisions about number of hidden units and learning rates were made in an *ad hoc* fashion. The Phrase module was particularly difficult to train. The gating units had very complex dynamic behavior, requiring recurrent connections to learn the task. Without general methods for reliable convergence, application to additional tasks would be difficult.

The architecture suffers from an additional computational inadequacy. Using gating units for assigning words to phrase blocks requires that it is always possible to decide which phrase block an input word belongs to at the time it is presented. A gating unit must assign an input word to some phrase block during the time course of its presentation, and no right context information can be used. For the particular task that the network was trained for, this was not a problem.

In general, it is not always possible to decide questions of phrase block membership without right context. For example, constructions of "[verb + particle]" (e.g. "[fill in] [the form]") versus "[verb] + [prepositional phrase]" (e.g. "[summarized] [in the announcement]") are often more easily disambiguated using the additional information to the right of the particle/preposition. A more general mechanism for symbol assignment is needed.

3.4 Summary

In this chapter, I introduced some of the problems facing connectionist parsing and set out the major design choices:

- Representation: words are represented as prespecified binary feature patterns; parses use a case-based representation built on non-recursive sub-units.
- Computation: symbol manipulation primitives are built into the connectionist network formalism, rather than learned.

The preliminary parsing architecture presented in this chapter, while being able to model complex language, lacks generality in some respects.

4 PARSEC Architecture

This chapter covers the PARSEC connectionist parsing architecture. The PARSEC system addresses the shortcomings of the connectionist architecture described in the previous chapter.

4.1 Conference Registration Task

Conference registration dialogs form a nice domain for developing speech-to-speech translation systems. At CMU, there were several parallel efforts on various aspects of the problem using conference registration. Here is a sample example conversation, repeated from Section 1.2:

CALLER: Hello, is this the office for the conference?

OFFICE: Yes, that's right.

CALLER: I would like to register for the conference.

OFFICE: Do you already have a registration form?

CALLER: No, not yet.

OFFICE: I see. Then, I'll send you a registration form. Could you give me your name and address?

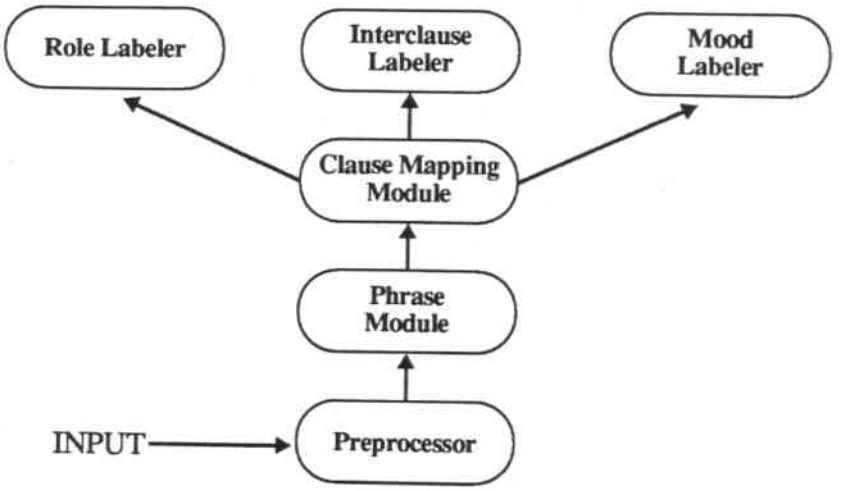
CALLER: The address is 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15236. The name is David Johnson.

OFFICE: I see. I'll send you a registration form immediately. If there are any questions, please ask me at any time.

CALLER: Thank you. Goodbye.

OFFICE: Goodbye.

FIGURE 4.1 High-level PARSEC architecture.



The conference registration dialog task corpus consisted of 12 conversations using a vocabulary of slightly more than 400 words. There were over 200 unique sentences in the corpus (Appendix B contains the full text of all 12 conversations). In addition to the text of the corpus, recordings of multiple speakers reading the conversations in an office environment were made as part of a speech recognition effort.

As with the task described in the previous chapter, the CR task included multiple-clause sentences, with both active and passive constructions. In addition, it contained questions, imperatives, conditionals, and conjunctions. It also contained domain-specific language such as telephone greetings and American addresses. Overall, the CR task used substantially more complex and varied language than was used to train the previous parser, but it did not have substantially more sentences available for training a parser.

4.2 Baseline PARSEC Parsing Architecture

Using the previous parsing architecture as a starting point, with the CR task as a testing platform, the PARSEC architecture was developed. In this section I will describe the *baseline* PARSEC architecture. In the next section, key enhancements to this architecture will be presented. The quantitative effects of the enhancements on generalization performance are discussed in Chapter 6.

Figure 4.1 is a high-level diagram of the architecture. Each of the modules performs a subpart of the total parsing task. The design of these modules was guided by a central principle: *a connectionist network should not be required to learn mundane behaviors, especially if such behaviors are difficult to learn.* The network should only be required to learn those pieces of the problem that can't be solved by other methods. If a module is supposed to transform a sequence of words into a set of separate phrase blocks, the

FIGURE 4.2 Input representation.

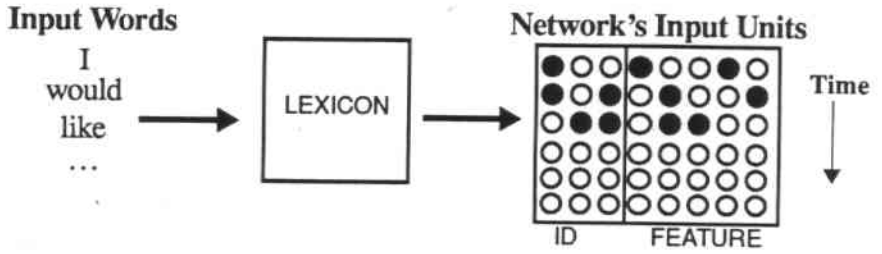
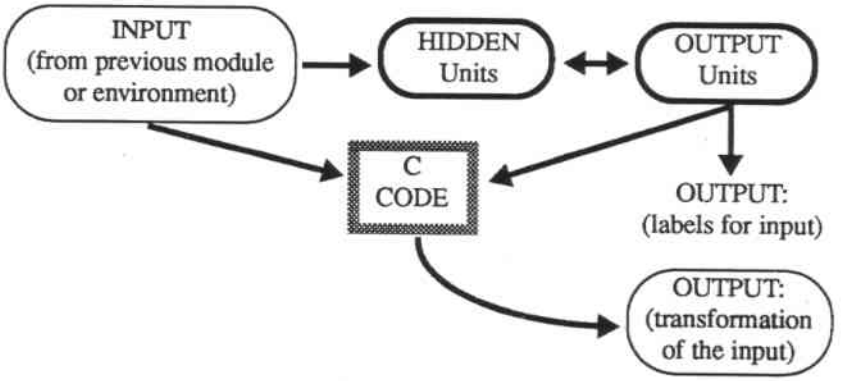


FIGURE 4.3 Generic PARSEC module structure.

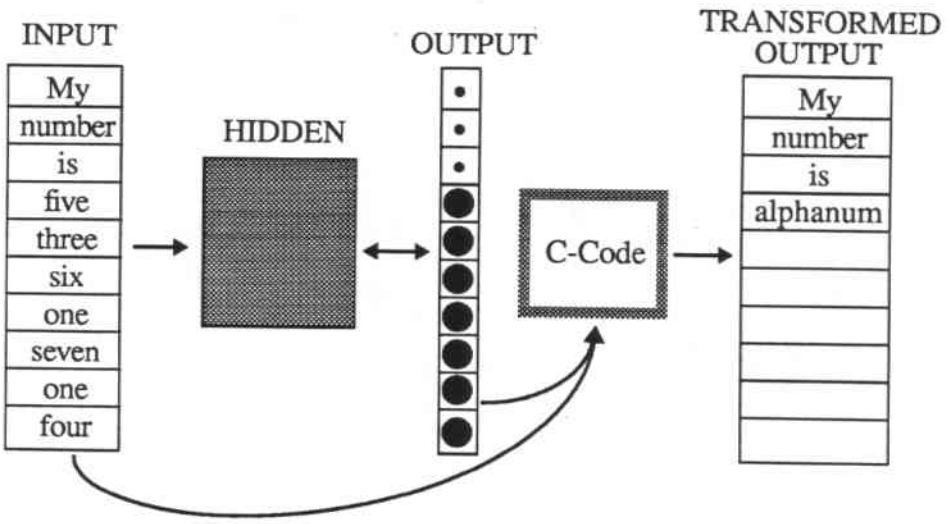


interesting part is deciding where the phrase block boundaries are, not in performing the actual transformation from the input units to the output units.

The input to PARSEC is a sequence of words corresponding to an English sentence. The mapping from word-sequence to network input is shown in Figure 4.2. As with the previous parsing architecture, the meanings of words are encoded as unlearned feature patterns (in later figures, mock unit activations will be replaced by the actual words that are being represented across the units). This architecture differs in the way the *sequence* of words is represented. Each word is looked up in a lexicon, and the activation pattern that encodes the word's meaning is applied to the proper set of input units in sequence. The input units are arranged in a 2D grid, with each row corresponding to a single slot. This input representation eliminates the need for the network to make immediate decisions about the current input word. Since the words are stored in a stable matrix of units, the network can postpone decisions when necessary by looking ahead in the input buffer.

Each of the modules has the basic structure shown in Figure 4.3. The input units reflect some intermediate representation of the developing parse. The output units can be

FIGURE 4.4 Preprocessor module.



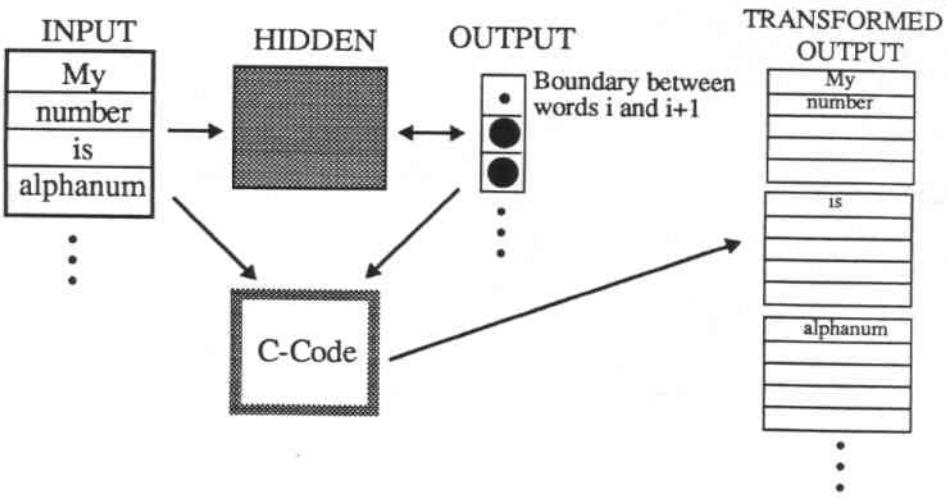
trained to either assign a label to a particular set of input units, or cause a data transformation to occur. There are recurrent connections from the output units to the hidden units. The data transformation is carried out by non-connectionist software. This is a generalization of the gating type of behavior that was described in the previous architecture. Instead of only allowing gating behavior, complex data transformations are possible through the use of clerical subroutines. PARSEC uses two types of transformation:

- Replacement of marked structures.
- Breaking structures at marked boundaries.

The bottom three modules of the PARSEC architecture (see Figure 4.1) perform data transformations, and the top three label the transformed result. I will describe the modules bottom-up.

The first module (called the Prep module) is an optional task-dependent preprocessing module (see Figure 4.4). It can be trained to perform simple filtering operations on the incoming text to simplify the operation of the other modules. Input word representations are placed in slots sequentially (see Figure 4.2). The output units make their decisions dynamically, and the result of their decisions is a (sometimes) modified representation that appears across the input units of the next module. In the CR task, the Prep Module is used to replace alphanumeric strings (like phone numbers) with a single special word "alphanum." This eliminates a potential difficulty in representing arbitrarily long alphanumeric strings. Note that this is not always a trivial task since words like "a" and "one" are lexically ambiguous.

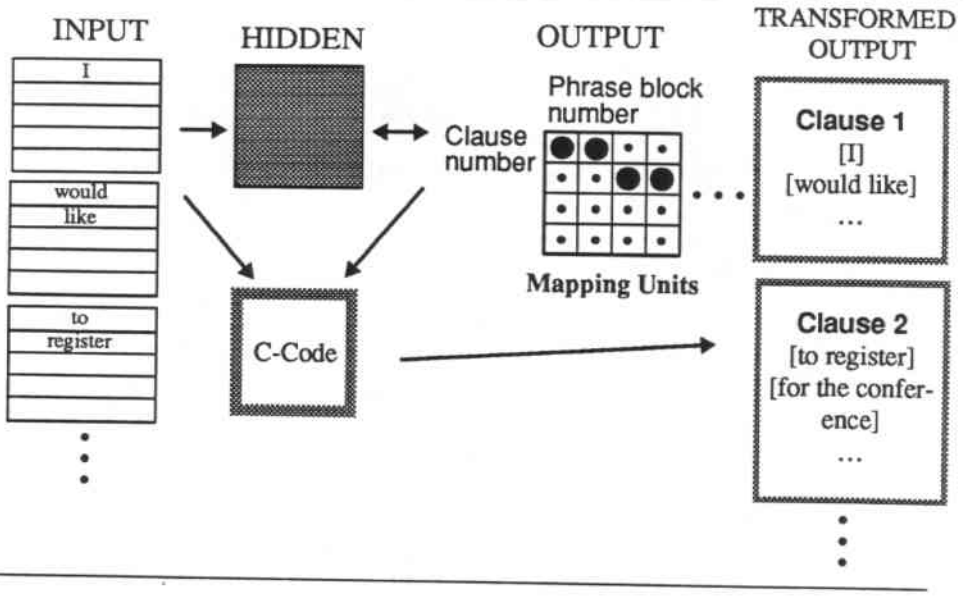
FIGURE 4.5 Phrase module.



The Phrase module (Figure 4.5) is responsible for transforming a sequence of words into phrase blocks (phrase blocks are described in Section 3.3). The input to the Phrase module is the evolving input sentence as transformed by the Prep module. The input is essentially the same as that shown in Figure 4.2, except that the words are coming from the output of the Prep module instead of from the environment. At a given time, the input matrix of the Phrase module reflects the Prep module's interpretation of the partial sentence. The output units of the Phrase module mark phrase block boundaries with high output, and other inter-word boundaries with low output. The effects of the output units appear immediately across phrase block representational units. An output value of 0.5 is the transition between non-boundary and boundary. The transformations are carried out by subroutine immediately as transitions are made. The key difference from the previous architecture is that the Phrase module need not learn complex dynamic behavior. Only the *recognition* of phrase boundaries is important. For the CR task, phrase blocks could contain up to five words.

The Clause Mapping module (abbreviated "Clause module") transforms the sequence of phrase blocks into separate clauses, similar to the Phrase module transforming the word sequence into separate phrase blocks (see Figure 4.6). The input is the phrase block representation of the sentence as indicated by the Phrase module. In the baseline architecture, the output units of the Clause module have the same structure and function as for the parser described in Chapter 3. Each phrase block has a set of associated mapping units that assign the phrase block to a particular clause. If a phrase block belongs to clause 1, the first unit in that phrase block's column of mapping units will have high activation. The decisions made by the mapping units are implemented by a subroutine in the simulation program. The transformed output is a new phrase block arrangement where each clause's phrase blocks are grouped together. This new representation is used by the labeling modules. In the CR task, up to four clauses were present, and each had room for up to seven phrase blocks.

FIGURE 4.6 Clause mapping module.



The three labeling modules use the output of the Clause Mapping module (the remapped phrase blocks) as input. The Role Labeling module (abbreviated "Roles module") must label the phrase blocks of each clause with the proper case-role or assign an attachment label that relates a phrase block to another phrase block. There are no side-effects with this module. The labeling is represented as a matrix of units (one column for each phrase block of a clause and one row for each label). Note that particular case-roles are fixed only for a particular parser, not for the general architecture. The choice of labels (both number and type) depends on the task. For the CR task, there were twelve labels (see Section 4.4).

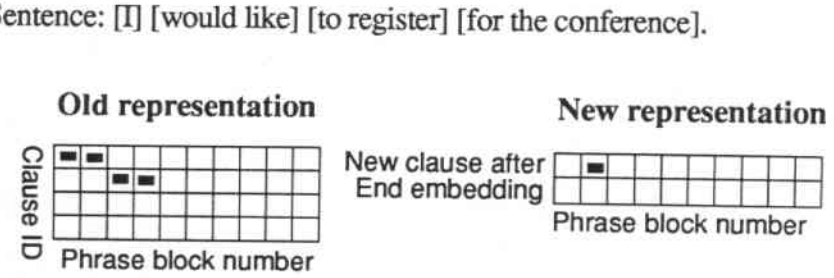
The Interclause Labeling module is very similar to the Roles module. It performs a labeling and attachment task at the level of clauses. The output of the Interclause module is represented as a matrix of units as with the Roles module (one column for each clause and one row for each label). Again, note that these labels are specified for a particular parser in some specific domain.

The Mood Labeling module (abbreviated Mood module) labels the input sentence's mood (e.g. declarative, interrogative, imperative...). There is a single output unit for each possible mood.

4.3 Architectural Enhancements

During the course of development, the PARSEC architecture underwent two major representational changes from the baseline architecture just described. Architectural constraints on connectivity within the modules were also added. The key principle

FIGURE 4.7 Alternate clause mapping representations.



embodied in these enhancements is: *wherever possible, incorporate domain knowledge to make the learning task easier.* The impact of these changes on performance will be discussed in Chapter 6.

4.3.1 Clause Mapping Representation

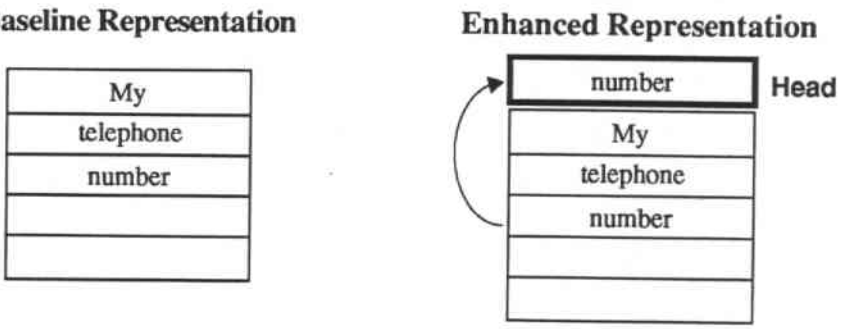
The representation of the Clause Mapping units was changed from the baseline system. Instead of being required to produce the clause *identity* of each phrase (as in Figure 4.6), the new output representation was required to indicate the *inter-phrase points* that begin new clauses and close embedded clauses. This allows for purely local decisions. For each inter-phrase junction, there are two units in the new representation: one indicating new clause boundaries, and the other closure of embedded clauses.

Figure 4.7 shows the two representations for "[I] [would like] [to register] [for the conference]." In the baseline representation, the output unit responsible for assigning "[for the conference]" to its clause needs to learn about what is happening in other previous phrase blocks in order to identify the *number* of the clause. In the new representation, each output unit only has to learn about local areas. For example, the unit that marks the clause boundary after "[would like]" only needs to look at the phrases "[would like]" and "[to register]" to make the proper decision. It does not matter where the previous phrase blocks are assigned.

An interesting effect of the new representation is that center embedding is now more limited than before. The old representation allowed arbitrary clause membership as long as one did not exceed the total number of clauses. Thus, "[The dog] [that] [the rat] [that] [the cat] [chased] [bit] [yelped]," is representable as the column winners "1 2 2 3 3 3 2 1" across the clause mapping units. However, the new representation is incapable of consistently representing that sentence along with "[The dog] [that] [was] [near the rat] [that] [the cat] [chased] [yelped]" with column winners under the old representation of "1 2 2 2 3 3 3 1." In the first case, the closure of "the cat chased" returns to "the rat" (clause 2), but in the second case, "the cat chased" returns to "the dog" (clause 1).

In PARSEC, I have adopted the convention that closure of an embedded clause always returns to the main clause. This allows for processing of sentences like "[The dog] [that] [was] [near the rat] [that] [the cat] [chased] [yelped]." But sentences like "[The dog]

FIGURE 4.8 Alternate phrase representations.



[that] [the rat] [that] [the cat] [chased] [bit] [yelped]," are not representable. This is not unreasonable behavior considering which of the two sentences is more likely to be uttered and understood by a human.

4.3.2 Phrase Block Representation

Using the baseline phrase block representation, the task that PARSEC must learn is needlessly complicated. The main "content word" of a phrase moves around depending on the phrase. For example, consider "the form" and "the registration form." In the first case, "form" appears in position 2 of the phrase, but in the second case, "form" appears in position 3 of the phrase.

Figure 4.8 illustrates the baseline and enhanced phrase block representations. On the left is the baseline version—simply a sequence of words constituting a phrase block. The head of the phrase moves around depending on the word sequence (e.g. "My telephone number" vs. "My number"). Any unit interested in making decisions based on the head of the phrase must learn to do so for each of the possible positions of the head. On the right is a modified representation. The last word in the phrase block is mapped into a canonical position (implemented by subroutine). The modified representation provides the network with a reliable salient information source.

4.3.3 Architectural Constraints

In addition to the representational changes, two restrictions on network connectivity were added to some modules of the architecture:

1. Localized input connectivity: to prevent distant context from affecting decisions that should be made locally.
2. Weight-sharing: to force position insensitivity.

To illustrate the points, consider the unit in the Phrase module responsible for learning how to mark the boundary between words 3 and 4 of the input. There is no reason to expect that information about word 9 should be useful in making this decision. However, if the information is available via connections (as in Figure 4.5), spurious correla-

tions might cause the unit for boundary 3–4 to base decisions in part on word 9. Also, the boundary detector for 3–4 will only learn about the constructions it sees *at that position*. Since each of the phrase boundary detectors is essentially performing the same task, it is possible to force position invariance by sharing weights in analogous positions.

Chapter 6 discusses the issues of localized input connectivity and weight-sharing in more detail in the context of generalization performance. The combination of representational enhancements and architectural constraints improves the coverage of PARSEC on novel sentences from less than 20% to nearly 70% on the CR task.

4.3.4 Recurrent Connections

In the baseline architecture, there are recurrent connections from the output units to the hidden units of each module. In the final PARSEC architecture, this recurrence is eliminated. The recurrent connections are not *required* to learn the parsing task (in contrast to the previous architecture), and due to an implementation issue, they hinder efficient network training. Section 6.6 discusses this issue further.

4.4 Constructing a Parser

There are four steps to constructing a PARSEC network:

1. Create a training file containing the target parses of the training sentences.
2. Create a lexicon containing the word features for the full vocabulary.
3. Train the individual modules of the parser.
4. Assemble the modules together to form the full parser.

Chapter 5 will discuss the details of parser construction. Much of the work is automated and takes little human time.

4.4.1 Example Parses

PARSEC requires example parses to learn, and they must be produced by hand. The syntax of the example parses is simple:

```
([statement]
    ([clause]
        ([action]       send)
        ([recipient]    me)
        ([patient]      a registration form)
        ([time]         in the morning)))
```

The opening parenthesis begins a parse for a single sentence. There is a label that indicates the mood of the sentence (e.g. [statement] or [question]). Next is a list of clauses, each with a label indicating the function of the clause. In this case, the [clause] label indicates that it is a main clause (see the list of labels for the CR task on page 45). Within each clause is a list of labeled phrase blocks. The label on a phrase block indicates the semantic (and sometimes syntactic) function of the words in the phrase block. The parser builder may choose any labels that he wishes to use for the particular task.

The example parse captures the required transformation and labeling information in PARSEC's modules.

Here is an example of a two clause sentence:

```
([statement]
    ([clause]
        ([action]       show)
        ([recipient]    me)
        ([patient]      all the nonstop flights))
    ([relative]
        ([action]       leaving)
        ([time]         in the afternoon)))
```

The second clause is a reduced relative clause and is understood to modify the phrase block immediately preceding it in the input. If it isn't the case that this is true for a particular task, one can either change the interpretation of the "relative" label to some other consistent meaning or use multiple labels to indicate relative clause attachment to different parts of the main clause.

The parse below shows an example of prepositional modification (marked "mod-1") as well as an unusual clause label ("greet"). For the CR task, syntactic subjects in existential constructions were labeled with "agent," but another label could have been used at the cost of some additional learning in the Roles module.

```
([question]
    ([greet]
        ([misc]         hello))
    ([clause]
        ([action]       is)
        ([agent]        this)
        ([patient]      the office)
        ([mod-1]        for the conference)))
```

The next parse shows an example of a subordinate clause (marked "sub-1"):

```
([statement]
    ([clause]
        ([agent]        i)
        ([action]       would like))
    ([sub-1]
        ([action]       to register)))
```

The last example parse shows a typical yes/no question structure with subject/aux inversion (the inverted auxiliary is marked "iaux"). The labels can indicate syntactic structures as well as semantic relationships:

```
([question]
    ([clause]
        ([iaux]         do)
        ([agent]        you)
        ([action]       already have)
        ([patient]      a registration form)))
```

In the CR task, for the Roles module, these were the 12 labels used:

- AGENT: the agent in the sentence that is performing the action, "[I] [gave] [the form] [to you]."
- ACTION: the action being performed, "[I] [gave] [the form] [to you]."

- PATIENT: the object being acted upon, "[I] [gave] **[the form]** [to you]."
- RECIPIENT: the recipient of the action, "[I] [gave] [the form] **[to you]**."
- LOCATION: location or destination, "[Meet] [me] **[at the conference site]**."
- TIME: the time of the action, "[I] [will send] [it] **[by March twentieth]**."
- STATE: state of being, "[I] [am] **[fine]**."
- HOW-ACT: the "how" in a how question: "**[How]** [can] [I] [apply]?"
- ADVERB: modification of verb, "[Thank] [you] **[very much]**."
- IAUX: auxiliary in subject/aux. inversion—a purely syntactic marker. For example, "**[Can]** [I] [help] [you]?"
- MOD-1: modification of the previous phrase block (relative to current phrase block): "[This] [is] [the office] **[for the conference]**."
- MISC: miscellaneous role. This is a catch-all for certain very low-frequency and/or unusual cases, e.g. **[Yes]**, **[Hello]**, **[If]**. However, most of the phrase blocks that are labeled with MISC belong to clauses that have special labels that indicate their function (e.g. a clause containing "hello" receives a GREET label).

For the Interclause module, these were the 8 labels:

- CLAUSE: marks independent or main clauses, "**{[I] [would] [like]}** {[to register] [for the conference]}.".
- SUB-1: marks clauses that are subordinate to the previous clause, "{[I] [would] [like]} **{[to register] [for the conference]}**."
- COND: marks clauses that are conditions of the main clause, "**{[If] [there] [are] [any questions]}** {[call] [me] [at any time]}."
- RELATIVE: means that the current clause is relative to the head of the phrase block immediately preceding it, "{[Send] [me] [the form]} **{[I] [should] [fill out]}**."
- GREET: marks clauses that function as greetings, "**{[Hello]}** {[this] [is] [the conference office]}."
- YES: marks clauses that function as affirmative answers, "**{[Yes]}** {[you] [can]}."
- NO: marks clauses that function as negative answers, "**{[No]}** {[the fee] [isn+t] [refundable]}."
- OK: marks clauses that function as confirmatory answers, "**{[Okay]}** {[I] [understand]}."

In the Mood module, declaratives and interrogatives were distinguished:

- STATEMENT: marks declarative sentences (also marks imperatives).
- QUESTION: marks interrogatives.

Note that PARSEC is not dependent on a particular labeling scheme. It has no built-in knowledge about the labels. It simply learns to assign them based on example parses.

4.4.2 A Completed PARSEC Network

The parsing network for the CR task with the best generalization performance had 15,390 non-learning units (used for the input matrix, Prep module output result, phrase

TABLE 4.1 Number of units per module in a CR parsing network.

Module	Input Rep.	Label/Transform	Hidden
Prep	1197	21	63
Phrase	1197	21	84
Clause	3420	40	10
Roles	9576	336	308
Inter	*	32	36
Mood	*	2	4

blocks, and Clause Mapping output result), 432 labeling and transformational units, 505 hidden units, and 63,654 modifiable connections. Note that many of these connections were *shared*, and the true number of free connections was much less than this total (about 6000).

- Words were represented as 57 bit feature patterns (9 ID bits + 48 feature bits). The input array allowed for up to 21 words per sentence.
- Phrase blocks contained up to five words. The phrase block representation units allowed up to ten phrase blocks per sentence.
- Clauses contained up to seven phrase blocks. The clause representation arrays allowed up to four clauses per sentence.

The unit counts are broken down in Table 4.1. An asterisk indicates that the module shared those units with the previous module.

4.4.3 Example Runs

This section shows example runs of a trained PARSEC network. Each of the figures for the first example sentence is a screen dump of the PARSEC program's display during testing of a sentence. Omitted from the displays are all hidden units, the Preprocessing module, and the intermediate representational units for all but the first clause. This was necessary because of space considerations.

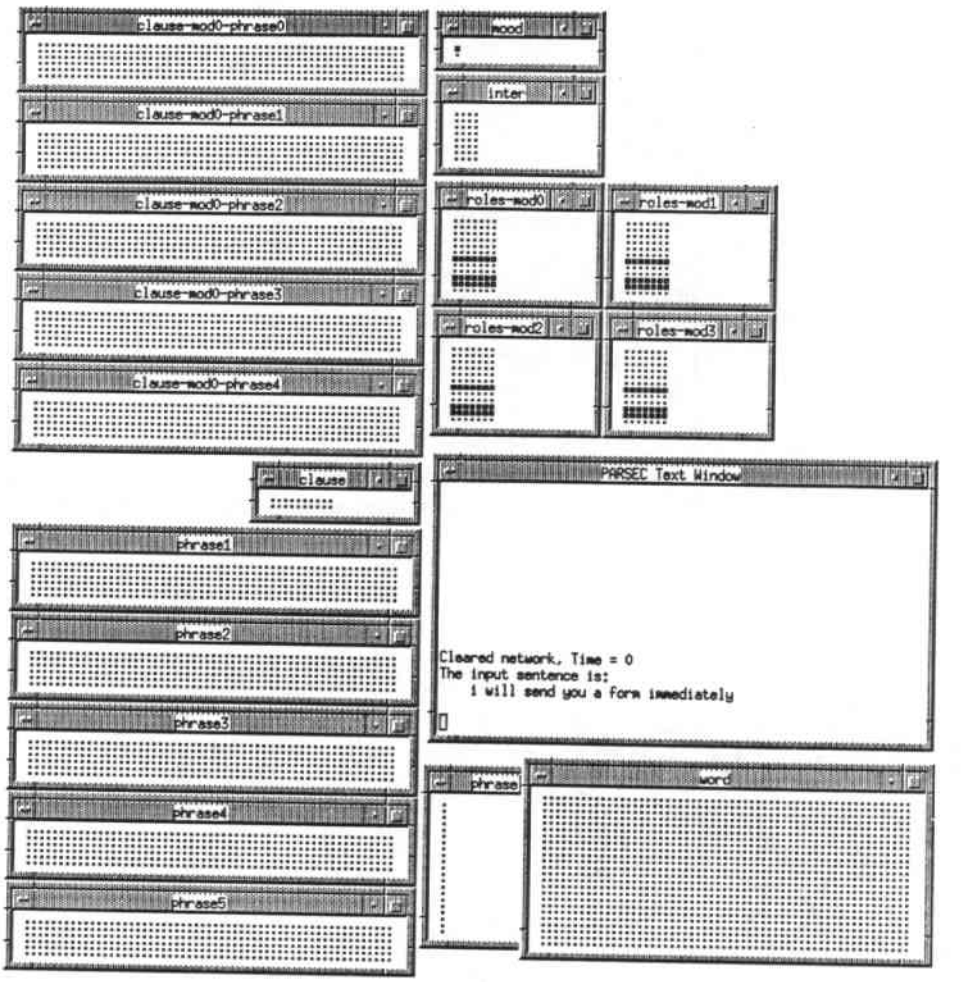
The next several figures show PARSEC parsing the sentence, "I will send you a form immediately." Each word is presented in sequence, and the network is allowed six update cycles before the next word is presented. Each of the small black rectangles represents the activation of a single unit. Block size indicates activation.

Note that the figures show a network using the enhanced architecture, and thus the representational changes discussed earlier appear in the figures. The "PARSEC Text Window" indicates the partial parse of the sentence. It shows words occupying the head slots of phrase blocks in parentheses.

The following is an explanation of each of the blocks of units in the figures:

- WORD: the set of units that forms the input to the Phrase module. It is the result of the Prep module (not shown). Each row can hold the feature representation of a single word.
- PHRASE: the output units of the Phrase module. These are a single column of units that indicate junctions between phrase blocks.
- PHRASEn: the n^{th} phrase block representational units. Note that the first row is the special head slot mentioned earlier. This is the input to the Clause module.
- CLAUSE: the output units of the Clause module. They mark beginnings of new clauses (top row) and closure of embedded ones (bottom row).
- CLAUSE-MODn-PHRASEm: the m^{th} phrase block of the n^{th} clause in the sentence. Only the first 5 phrase blocks of clause 0 are shown in the figures. These are part of the input for the labeling modules.
- ROLES-MODn: the output units of the Roles module for the n^{th} clause (columns correspond to clause-specific phrase blocks and rows correspond to labels).
- INTER: the Interclause labeling units (columns correspond to clauses and rows correspond to labels).
- MOOD: the Mood labeling units (top unit indicates STATEMENT, bottom indicates QUESTION).

FIGURE 4.9 Example run.



In Figure 4.9, the PARSEC network has just been cleared. Nearly all of the units have low activation. Some have learned high bias terms and have high output in the absence of input stimulation. For example, in the Mood module, the unit that indicates a declarative sentence has high initial activation.

FIGURE 4.10 Example run.

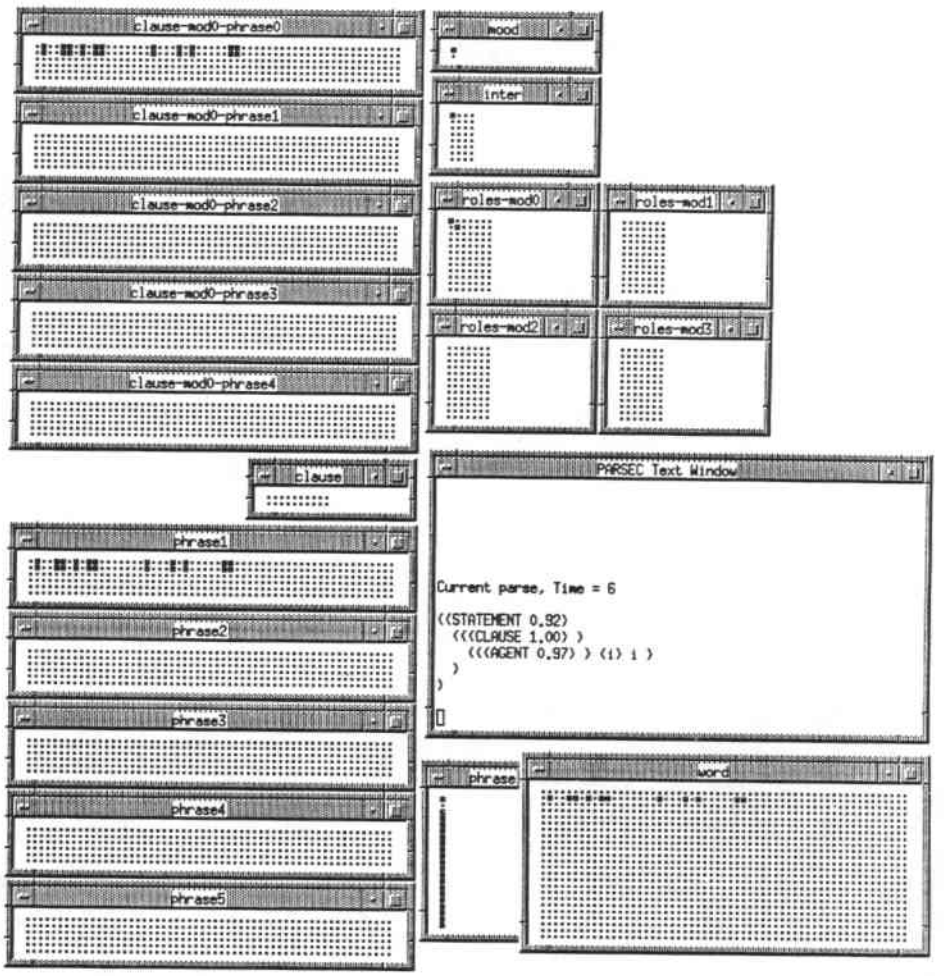
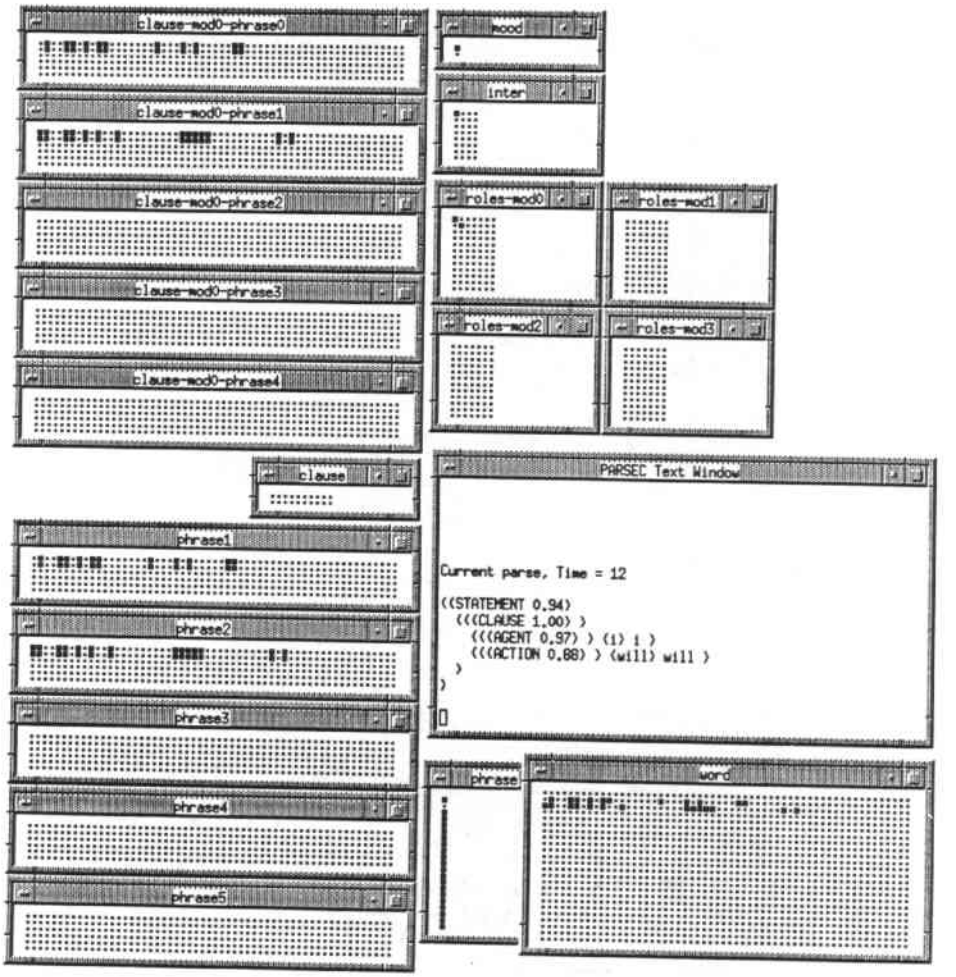


Figure 4.10 shows the state of the parse after "I" has been processed. After the features for the word "I" have been stimulated (and retained) across the proper slot of the Word units, the network has decided the following:

- "I" ends a phrase block, and the next word probably is part of a two word phrase.
- "I" is an AGENT, and the next phrase block will be an ACTION.
- So far, there is only a single clause (no activity in the Clause Mapping units), and it is a main clause.
- The mood of the sentence is declarative (a statement).

Representation of words and phrase blocks across the different intermediate units is instantaneous. As soon as any transformational unit crosses threshold, its effect is trans-

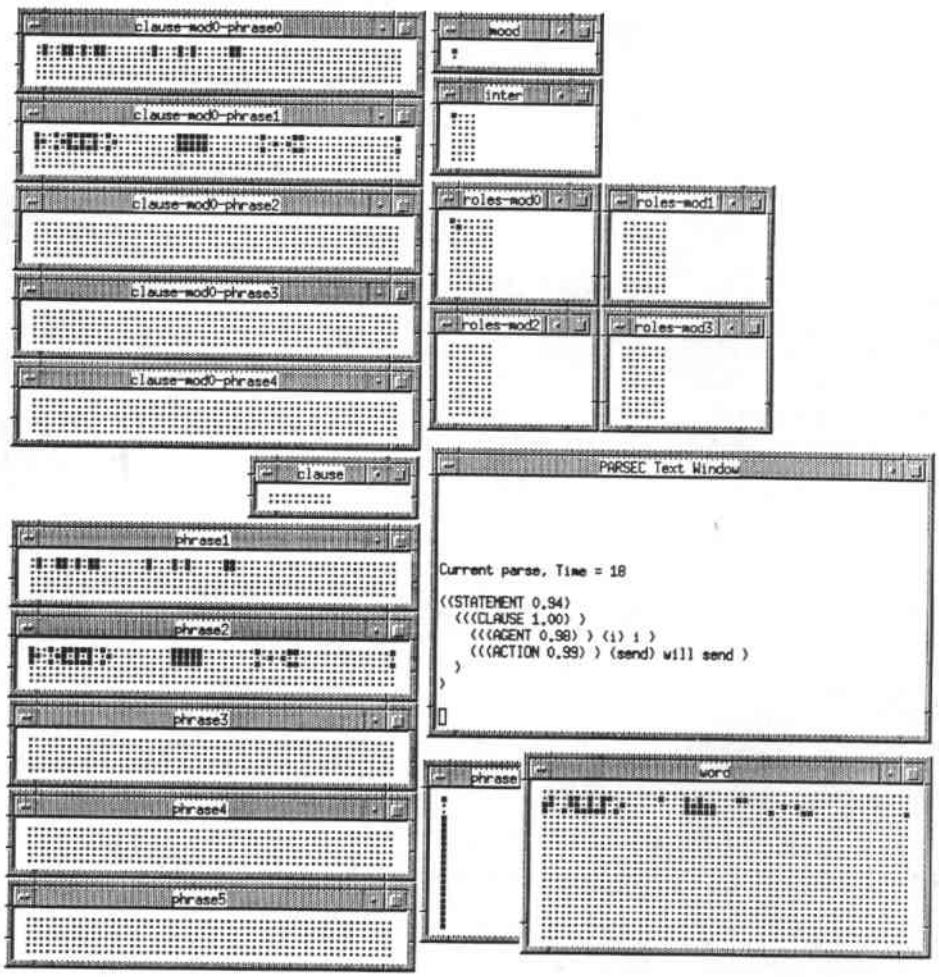
FIGURE 4.11 Example run.



mitted throughout the network. Note that "I" appears across *both* the first and second rows of the phrase block representation because it's the first element of the phrase block and it's also the head. The top row is the remapped head slot. In the printed representation, head words are indicated in parentheses.

Figure 4.11 shows the state of the parse after "I will" has been processed. The word "will" is incorporated into the parse in a new phrase block, and it receives an ACTION label, as predicted from the previous network state.

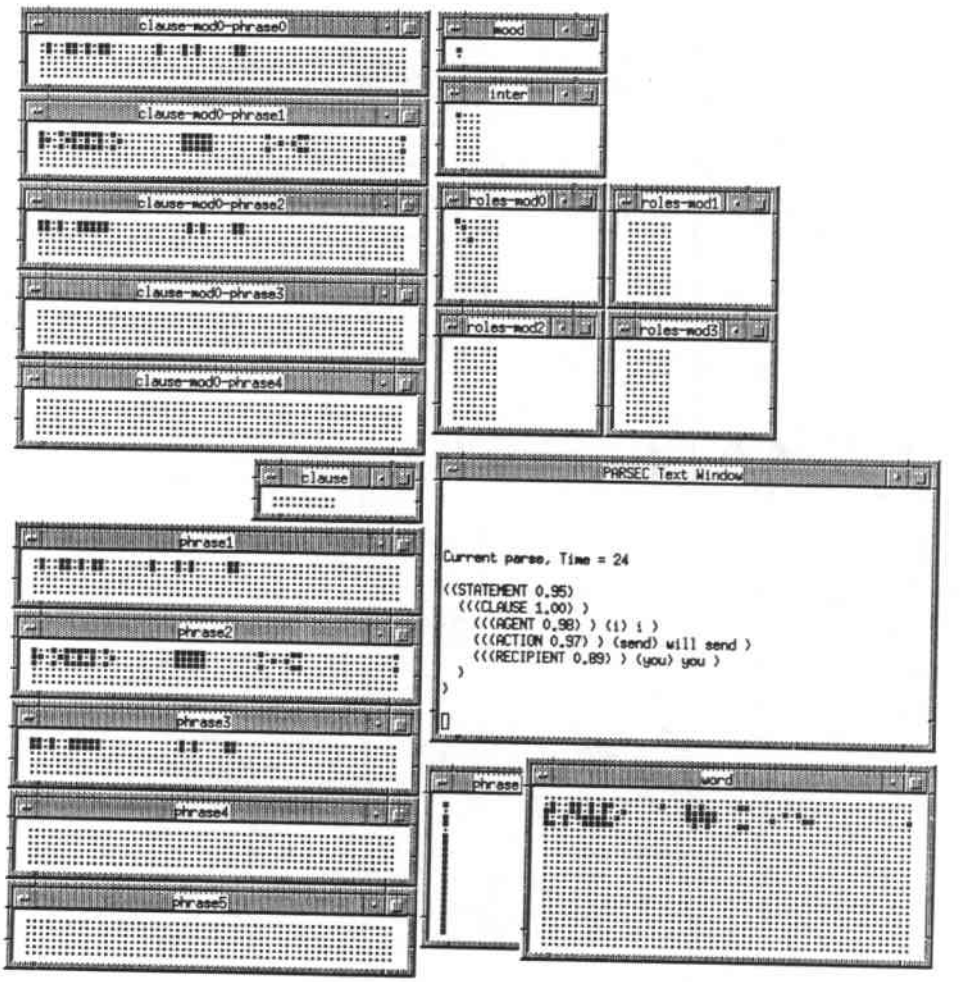
FIGURE 4.12 Example run.



In Figure 4.12, after receiving "send," the network predicts another constituent to the "[will send]" phrase block—perhaps a particle like "in." This is reflected by the two consecutive low-activation units in the Phrase module. This expectation will turn out to be wrong.

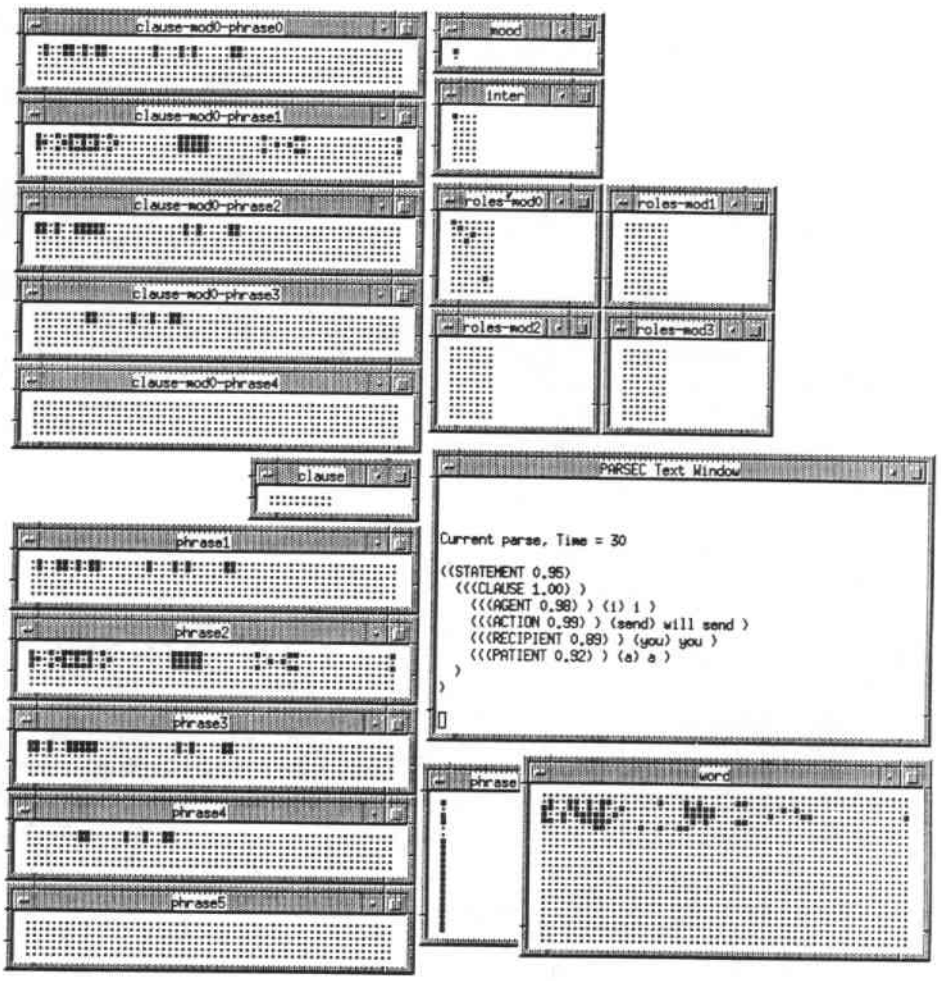
In the second phrase block, the head slot now reflects the features of "send" instead of "will" as in the previous figure. The simulation program responds immediately to new information, and performs remappings as needed.

FIGURE 4.13 Example run.



As seen in Figure 4.13, the word "you" begins a new phrase block, and it is assigned the RECIPIENT label. The network now seems to expect a two word phrase block to follow (it's right, but in the next figure, the expectation changes to a three word phrase block). Note that the network continues to show a STATEMENT mood and indicates that the sole clause is a main clause.

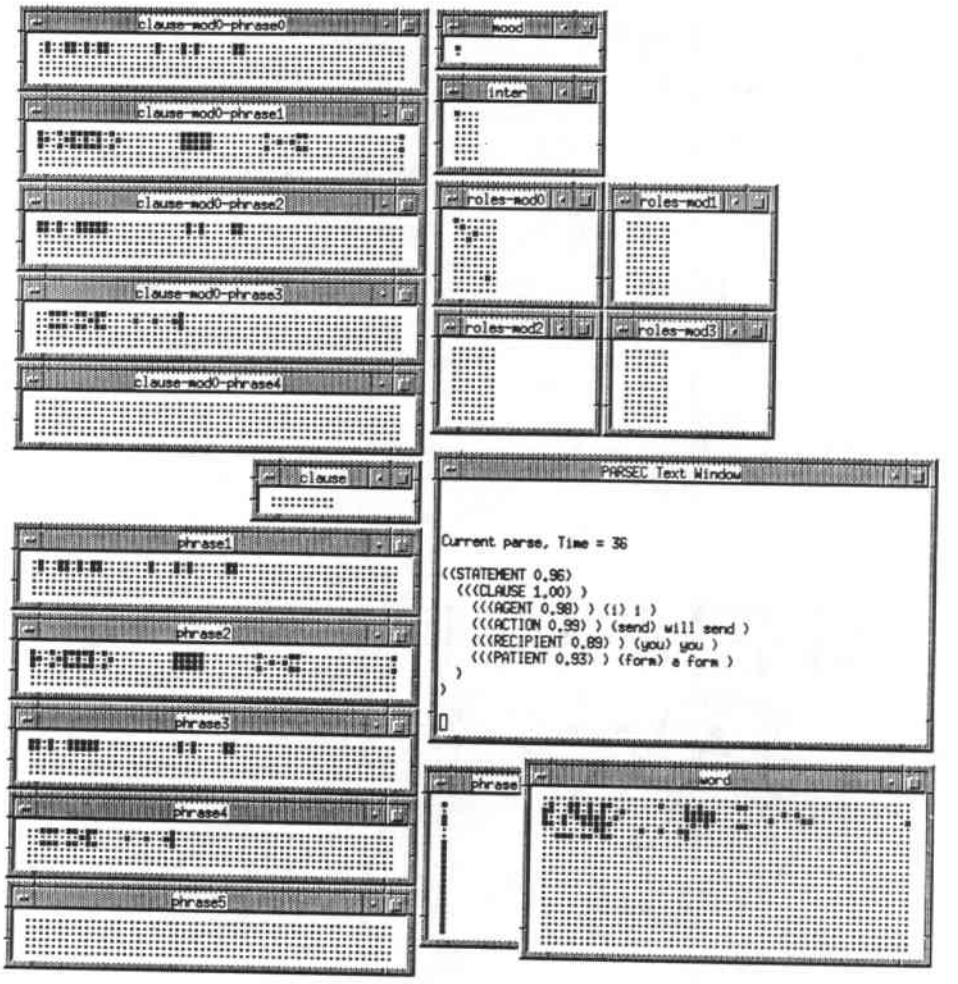
FIGURE 4.14 Example run.



When the network processes "a" (Figure 4.14), it believes that the new phrase block it is constructing should contain three words (e.g. "[a registration form]"). Such constructions were quite common in the training corpus. However, in this case, the prediction proves false. The network correctly labels the partially built phrase block as the PATIENT. The network makes a spurious prediction for phrase block 6 (STATE label) that persists until the end of the parse, but it does not affect the interpretation since the phrase block is empty.

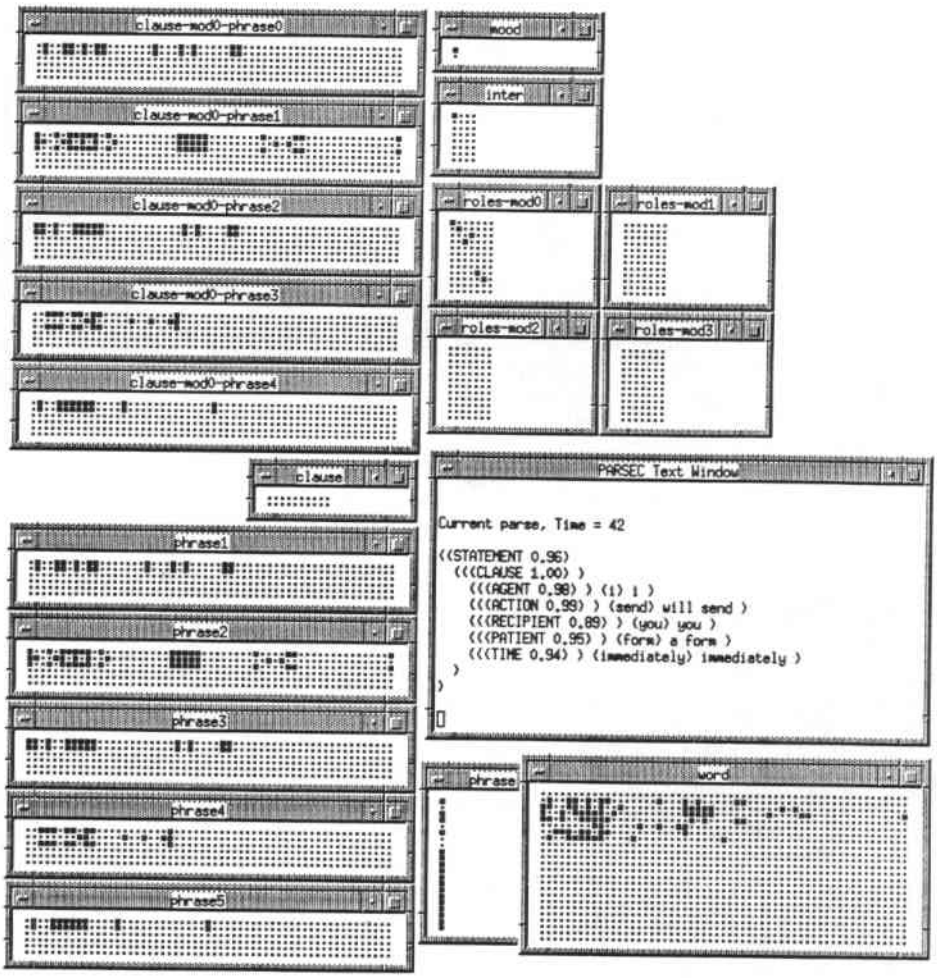
Initially, the head slot for phrase block 4 contains the representation for "a," but as with "will send," this will change with the next word.

FIGURE 4.15 Example run.



Not much happens with "form" (Figure 4.15), but the network still has not decided that "form" terminates a phrase block. This is because, as far as the network is able to tell from its sparse features, "form" might be part of a noun compound with the next word. If the network knew more about individual words, it would be able to close the phrase block at this point.

FIGURE 4.16 Example run.



In the final figure of the example run (Figure 4.16), "immediately" is incorporated into the parse. The figure shows the correct final parse of the sentence.

The previous example was fairly simple. PARSEC networks exhibit more complex behavior on multi-clause sentences. In what follows, I will discuss the behavior of PARSEC on the sentence "The titles of papers to be presented at the conference are printed in the second version of the announcement." Note that there is an embedded relative clause within a passively constructed main clause. There are also some prepositional phrases. In the interest of saving space, screen displays will not be shown. The partial parses of the sentences that the PARSEC network produces contain most of the interesting information.

Very early in the parse, while processing "the," the network produces the following partial parse:

```
((statement 0.81)
    ((clause 0.98)
        ((agent 0.30)        (the)          the)))
```

Note that there is only partial activation of the AGENT label. The activation patterns are still settling.

In the following update cycle, the label changes to PATIENT:

```
((statement 0.80)
    ((clause 0.99)
        ((patient 0.38)      (the)          the)))
```

Within a few more time steps, the network settles on the assignment of AGENT for this fragment. The assignment is incorrect, but the network will revise this initial guess.

```
((statement 0.86)
    ((clause 0.99)
        ((agent 0.71)        (the)          the)))
```

As the next word's representation becomes active, it is incorporated into the first phrase block, initially with no change of label:

```
((statement 0.88)
    ((clause 1.00)
        ((agent 0.79)        (titles)       the titles)))
```

However, the network soon assigns *both* AGENT and PATIENT labels to the phrase block:

```
((statement 0.89)
    ((clause 0.99)
        ((agent 0.63)
         (patient 0.60)      (titles)       the titles)))
```

This is followed by a loss of support for the AGENT label, probably since "titles" is not animate:

```
((statement 0.89)
    ((clause 0.99)
        ((patient 0.64)      (titles)       the titles)))
```

The word "of" is immediately labeled as part of an attached prepositional phrase:

```
((statement 0.88)
    ((clause 0.99)
        ((patient 0.81)      (titles)       the titles)
        ((mod-1 0.89)        (of)           of)))
```

The current representation solidifies with slightly higher activation prior to the next word. The word "papers" does not affect the structure much:

```
((statement 0.86)
    ((clause 0.99)
        ((patient 0.84)      (titles)       the titles)
        ((mod-1 0.99)        (papers)       of papers)))
```

"To" begins the embedded relative clause. The new clause is immediately recognized, but both the label of the new clause and the label of the new phrase block are incorrect partial activations:

```
((statement 0.88)
    ((clause 0.99)
        ((patient 0.83)      (titles)        the titles)
        ((mod-1 0.98)        (papers)        of papers))
    ([clause 0.26])
        ((patient 0.48)      (to)            to)))
```

This changes to a more reasonable interpretation rapidly. Now, the new clause is labeled as being subordinate (probably because most subordinate clauses begin with "to"). The new phrase block is labeled as possibly an ACTION or a RECIPIENT—both reasonable possibilities for "to" if local context dominates.

```
((statement 0.71)
    ((clause 0.99)
        ((patient 0.83)      (titles)        the titles)
        ((mod-1 0.98)        (papers)        of papers))
    ((sub-1 0.97)
        ((action 0.52)
         (recipient 0.63)    (to)            to)))
```

The network settles on the ACTION interpretation of "to" below.

```
((statement 0.58)
    ((clause 0.99)
        ((patient 0.83)      (titles)        the titles)
        ((mod-1 0.98)        (papers)        of papers))
    ((sub-1 0.99)
        ((action 0.69)       (to)            to)))
```

The word "be" results in a sequence of shifts:

```
((question 0.51)
    ((clause 0.99)
        ((patient 0.83)      (titles)        the titles)
        ((mod-1 0.98)        (papers)        of papers))
    ((sub-1 0.99)
        ((action 0.56)       (be)            to be)))

((statement 0.63)
    ((clause 0.99)
        ((patient 0.83)      (titles)        the titles)
        ((mod-1 0.98)        (papers)        of papers))
    ((sub-1 0.95)
        ((recipient 0.48)    (be)            to be)))

((statement 0.71)
    ((clause 0.99)
        ((patient 0.83)      (titles)        the titles)
        ((mod-1 0.98)        (papers)        of papers))
    ((rel (of papers) 0.43))
        ((action 0.86)       (be)            to be)))
```

The Mood module responds with a spurious but small fluctuation in the overall sentence mood. The Interclause module produces the correct label for the embedded clause. The Roles module produces the correct ACTION label with high confidence after temporarily labeling "to be" as a RECIPIENT. The network arrives at a stable interpretation of the sentence fragment in which the second clause is correctly labeled as being relative to

"of papers." The network incorporates "presented" with no difficulty, and it solidifies its current predicted structure.

```
((statement 0.92)
    ((clause 0.99)
        ((patient 0.83)          (titles)          the titles)
        ((mod-1 0.98)            (papers)          of papers))
    ((rel (of papers) 0.85)
        ((action 0.85)           (presented)       to be presented)))
```

After temporarily assigning "at" to a new phrase block labeled RECIPIENT, the network arrives at the proper result.

```
((statement 0.94)
    ((clause 0.99)
        ((patient 0.83)          (titles)          the titles)
        ((mod-1 0.98)            (papers)          of papers))
    ((rel (of papers) 0.82)
        ((action 0.97)           (presented)       to be presented)
        ((location 0.56)         (at)              at)))
```

The words "the conference" are processed with little revision of network predictions:

```
((statement 0.92)
    ((clause 0.99)
        ((patient 0.83)          (titles)          the titles)
        ((mod-1 0.98)            (papers)          of papers))
    ((rel (of papers) 0.81)
        ((action 0.96)           (presented)       to be presented)
        ((location 0.70)         (conference)      at the conference)))
```

The word "are" is initially placed in the wrong phrase block along with "at the conference." Then it is properly assigned to its own phrase block and to the proper clause. Eventually, the correct role label becomes most active as well:

```
((statement 0.92)
    ((clause 0.99)
        ((patient 0.83)          (titles)          the titles)
        ((mod-1 0.98)            (papers)          of papers))
    ((rel (of papers) 0.80)
        ((action 0.95)           (presented)       to be presented)
        ((location 0.84)         (are)             at the conference are)))

((statement 0.91)
    ((clause 0.99)
        ((patient 0.67)          (titles)          the titles)
        ((mod-1 0.94)            (papers)          of papers)
        ((action 0.96)           (are)             are))
    ((rel (of papers) 0.80)
        ((action 0.96)           (presented)       to be presented)
        ((location 0.82)         (conference)      at the conference)))
```

The word "printed" solidifies the current partial parse:

```
((statement 0.91)
    ((clause 0.99)
        ((patient 0.92)          (titles)          the titles)
        ((mod-1 0.99)            (papers)          of papers)
        ((action 0.99)           (printed)         are printed))
    ((rel (of papers) 0.80)
        ((action 0.95)           (presented)       to be presented)
        ((location 0.84)         (conference)      at the conference)))
```

"In" is processed and produces a MOD-1 label initially, which loses out to the LOCATION label very quickly. As "the" is processed, the correct interpretation gains support:

```
((statement 0.91)
    ((clause 0.99)
        ((patient 0.93)          (titles)          the titles)
        ((mod-1 0.99)            (papers)          of papers)
        ((action 0.99)           (printed)         are printed)
        ((location 0.54)
         (mod-1 0.52)            (in)              in))
    ((rel (of papers) 0.80)
        ((action 0.95)           (presented)       to be presented)
        ((location 0.84)         (conference)      at the conference)))

((statement 0.91)
    ((clause 0.99)
        ((patient 0.93)          (titles)          the titles)
        ((mod-1 0.99)            (papers)          of papers)
        ((action 0.99)           (printed)         are printed)
        ((location 0.90)         (in)              in the))
    ((rel (of papers) 0.80)
        ((action 0.95)           (presented)       to be presented)
        ((location 0.84)         (conference)      at the conference)))
```

The current phrase block is completed with "second version."

```
((statement 0.79)
    ((clause 0.99)
        ((patient 0.93)          (titles)          the titles)
        ((mod-1 0.98)            (papers)          of papers)
        ((action 0.99)           (printed)         are printed)
        ((location 0.86)         (version)         in the second version))
    ((rel (of papers) 0.80)
        ((action 0.95)           (presented)       to be presented)
        ((location 0.84)         (conference)      at the conference)))
```

The last phrase block is also processed with no major shifts in interpretation. The final parse of the sentence is shown below.

```
((statement 0.79)
    ((clause 0.99)
        ((patient 0.93)          (titles)          the titles)
        ((mod-1 0.98)            (papers)          of papers)
        ((action 1.00)           (printed)         are printed)
        ((location 0.90)         (version)         in the second version)
        ((mod-1 1.00)            (announcement)    of the announcement))
    ((rel (of papers) 0.80)
        ((action 0.95)           (presented)       to be presented)
        ((location 0.84)         (conference)      at the conference)))
```

The dynamic behavior of the PARSEC network was qualitatively similar to that of the early parsing architecture as discussed in the previous chapter. It made predictions about sentence structure based on partial sentences, and it revised them as more information became available.

4.5 Summary

In this chapter, I described the baseline PARSEC architecture along with some key enhancements whose performance impact will be discussed in Chapter 6. The two guiding design principles were:

1. To avoid forcing networks to learn mundane transformational operations.
2. To incorporate domain knowledge into the architecture rather than expecting the architecture to learn it from limited training data.

The architecture is modular. Each module can perform one of two actions: transformation or labeling. The modules are set up to perform three successive transformations and then to label the result using another three modules.

The architecture is quite general in principle, since any combination of transformation and labeling steps are possible. In this work though, only the size of the various modules and the labels that are applied vary.

The dynamic behavior of trained networks is quite complex. It arises from training using an incremental parsing paradigm with the requirement that the parsing networks always attempt to produce the final parse of partial sentences.

5 PARSEC Training

The previous chapter described the PARSEC architecture, but without robust learning algorithms, the architecture would prove to be of little value. In this chapter, I will describe the constructive learning technique that PARSEC uses—*Programmed Constructive Learning* (PCL). The techniques used in PARSEC's automatic training algorithm will also be presented.

There were four critical areas in training PARSEC networks:

1. Generalization: learning algorithms to produce networks that generalize well.
2. Learning speed: minimization of network size, algorithms to make "learning to completion" possible.
3. Robustness: methods to ensure that networks would reliably converge.
4. Automation: a single generalized training algorithm for use by non-experts who cannot be expected to fine-tune parameters or architectures.

The PCL technique addresses the generalization issue, but does not solve other problems. Back-propagation learning is a gradient-descent method that is often slow in practice. There are many factors that influence learning speed in such networks, including:

- Network size: the bigger the network (especially in terms of number of modifiable connections), the longer it takes to train it.
- Parameter tuning: if a lot of human parameter tuning is required, the learning process can slow down (sometimes the human makes non-optimal decisions, and the learning process cannot be constantly monitored).
- Local minima: while generally not considered to be a problem in high-dimensional input spaces, this can be a problem when one attempts to train to completion.

In early experiments, completely *ad hoc* solutions were used for each of these problems, and while I was able to produce acceptable networks, the amount of labor and uncertainty involved were unacceptable.

The next section describes the PCL algorithm. Following that is a discussion of learning effects that impact back-propagation. Then PARSEC's overall learning algorithm is described. The chapter concludes with an experiment in which a non-expert successfully trained a PARSEC network for a novel parsing task and an application of PARSEC to another language: German.

5.1 Programmed Constructive Learning

To address the generalization performance issue, I developed a new form of constructive learning (similar to Fahlman and Lebiere 1990), which is called *Programmed Constructive Learning* (PCL). It differs from other constructive techniques in that a *specific sequence* of new hidden unit types is used. Hidden unit types are specified by a network designer based on domain knowledge, and they typically have progressively widening receptive fields.

The first unit added (from type 0) relies on very local information and learns as much as it can. More units of the same type are added until no significant improvement is realized. Then, units from the next type are added in a similar manner. This process continues until the training corpus is exhausted or until no further improvement is desired. This process is automatic except for the description of the hidden unit types, which must be done for a particular domain just once.

Figure 5.1 shows some hidden unit types for PARSEC's Phrase module. Type 0 units have input connections from the word units preceding the boundary that their output units are supposed to detect. Type 1 units receive input from both sides of the word junction. Type 2 units have an even wider input field. Type 0 units are likely to learn things like "prepositions usually don't end phrases." The important point is that they can learn this using narrow, local information. This technique reduces free parameters in the system as well as enforcing a stronger locality constraint than in networks trained without PCL.

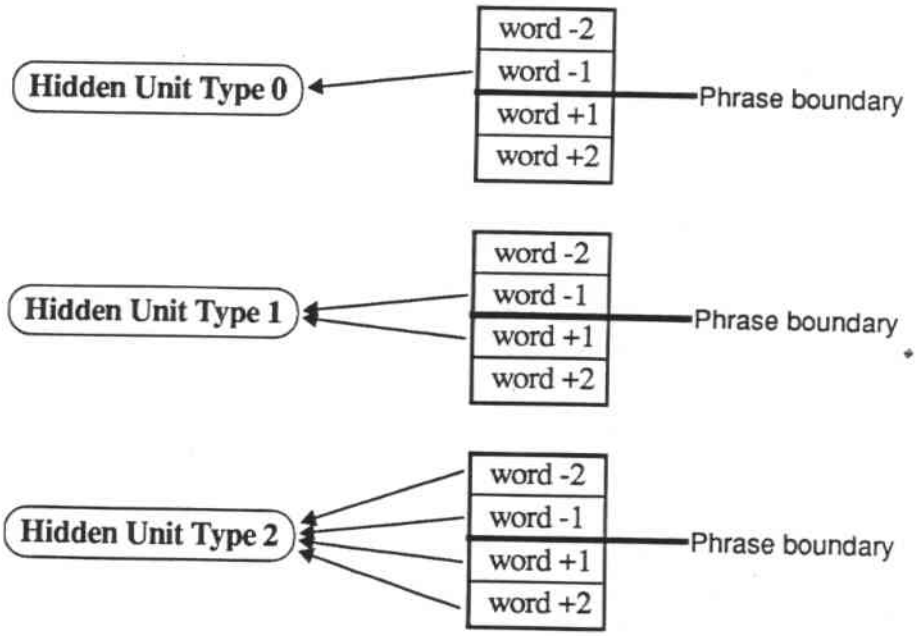
Using PCL, generalization performance of the Phrase module increased from 79% to 95%. The impact of PCL on generalization is described in detail in Chapter 6.

5.2 Learning Effects

5.2.1 The Herd Effect

Fahlman (1988, 1990) has described a phenomenon called the "herd effect." One sees this when there are a number of hidden units in a network that all start out with small, random weights. In a fully connected network, the hidden units tend to change their weights in the same ways. This results in a herd of hidden units that are all doing the same thing—attempting to minimize the principal source of error.

FIGURE 5.1 PCL hidden unit types for the Phrase module.



I have found three ways to overcome this problem:

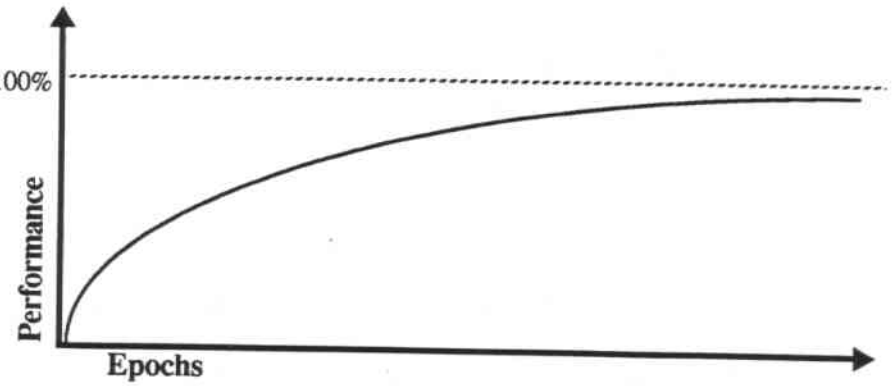
1. Don't use full connectivity between the output and hidden units. Give output units "private" hidden units.
2. Don't use full connectivity between the input units and the hidden units. Randomly connect some proportion of the input units to each hidden unit.
3. Use constructive learning techniques where hidden units are added one at a time. Each hidden unit then sees very different error corrections.

Of these methods, the third one is best. It eliminates the herd effect, and it also solves the network resource problem. The network size depends on the task at hand, and no *ad hoc* decisions need to be made. Usually the networks grow slightly larger than is required, but not excessively large. An added feature is that much of the learning takes place with relatively small networks and is thus faster per epoch than for larger networks.

5.2.2 The Wasted Hidden Unit Effect

This effect manifests itself when a network's output units are far from an obvious local minimum at the beginning of training. Consider a binary task where, most of the time, the network's output units are to have the low value, but their initial bias is 0 (and so their resting output is an intermediate value). There is a local minimum in weight-space where the network simple produces low values for all outputs. Often, a back-propaga-

FIGURE 5.2 Typical learning curve in a back-propagation network.



tion network will move to this minimum early in training by placing negative weights on the hidden to output connections.

Even though the bias term is modifiable, it is quicker for the output units to modify their multiple input connections to achieve the local minimum than it is to modify the bias terms optimally. The simple solution for this is to start the output units off at the obvious minimum, where their initial resting output values are exactly that of the low binary value. While this may seem trivial, it does save time and resources.

5.2.3 The Completion Effect and Forgetting

In most instances of back-propagation learning, one does not attempt to train the networks to completion. However, the case of learning to parse a corpus of well-formed sentences is different than the more typical application of back-propagation, where it is likely that some of the training exemplars are noisy (e.g. speech recognition or hand-written digit recognition). It is not unreasonable to expect that a parsing system that learns should, in principle, be able to learn all of the example parses presented to it.

It is true that generalization performance suffers when training to completion (see Section 6.6). However, it is useful to be able to do so in order to make better comparisons to hand-written grammars, where one can add rules until an example set is covered. In practice, it is unwise to train parsing networks to completion, since one is more likely to see novel sentences than training sentences.

In training the networks, the observed learning curve was the familiar asymptote (see Figure 5.2). The vast majority of training examples could be learned in short order. However, the last few were usually very stubborn, and this created an interesting gaming situation. In the obvious strategy of going through the training examples one at a time, during the final epochs most of the time is wasted on tokens that are learned satisfactorily. However, one cannot force the network to completely concentrate on the unlearned tokens, because the network would then forget previously learned examples.

I tried numerous methods to overcome this problem:

- Skipping learned tokens: each of the tokens that was processed successfully was skipped a number of times in future epochs depending on the number of times it had been correctly processed. This focused attention on tokens that were close to decision boundaries.
- Focusing attention: each of the unlearned tokens received more presentations per epoch than learned tokens.
- Learning rate manipulation: by adjusting the learning rate, the idea was to minimize the damage done to learned tokens while allowing unlearned tokens to be gently pushed to the proper side of the decision boundaries.
- Weight freezing: weights in trained portions of a partially built network were frozen to prevent forgetting.
- Combination methods: attempts at combining these methods in various ways were also attempted.

Unfortunately, most of the methods had problems:

- Skipping tokens: while being effective at focusing attention on unlearned tokens, the problem of forgetting was magnified unacceptably.
- Focusing attention: it was difficult to control the parameters that adjusted the number of presentations simultaneously with adjustment of the learning rate.
- Learning rate adjustment: see above.
- Weight freezing: this was also somewhat effective, however, it produced larger networks than needed.

The next section describes the PARSEC learning algorithm in the context of the effects enumerated here. The algorithm is a three phase approach that uses learning rate manipulation, attention focusing, and token skipping in different ways in each of the phases.

5.3 PARSEC Learning Algorithm

I use several terms in what follows:

- *token*: a single training example for a single module.
- *epoch*: a single pass through an entire set of tokens for a single module.
- *success*: after a forward propagation step, if the output units of the module are within some epsilon of their target values, it is called a success.
- *failure*: not a success.

The learning algorithm has three phases:

Phase 1: In this initial phase of learning, most of the tokens are learned.

Phase 2: Begins when all of the tokens in the training file can be learned (at least temporarily) during a single epoch.

Phase 3: Begins when there are very few tokens that are processed incorrectly.

There are four key parameters that control the learning:

1. *learning_rate*: A multiplicative factor that controls how steep the gradient-descent is.
2. *modify_on_success*: If true, modify weights for this token after both success and failure.
3. *max_pres_number*: The maximum number of presentations per training token.
4. *failure_proportion*: The fraction of training tokens that are processed incorrectly by the current network.

5.3.1 Main Learning Procedure

This procedure is responsible for implementing the PCL algorithm. Hidden units are added one at a time. After adding a unit, the network is trained to quiescence, and an evaluation of performance is made. If there is improvement, another hidden unit of the same type is added, otherwise, the next hidden unit type is selected. Here is the algorithm:

```
Build the initial network (no hidden units).
Loop {
    Try to add a hidden unit of the current type (initially 0).
    If out of types, terminate.
    Read parameters for current hidden unit type.
    Call learn_one_unit.
    Check performance on the training examples.
    If no failures, return.
    Check if the current hidden unit improved performance significantly.
    If not, increment hidden unit type.
}
```

The parameters that the procedure reads are specified by the hidden unit type designer. They include the minimum and maximum number of units to add to a network and the initial learning rate. These are available to allow for additional control of the network architecture, but in practice, the algorithm does not typically run into the bounds on hidden units. Also, for all but the first hidden unit added, the learning rate comes from adaptive adjustment during the previous hidden unit's learning period.

Note that PARSEC keeps the last hidden unit of each type that is added, even though those hidden units do not appreciably affect performance (as measured after they reach a plateau). However, there is no weight-freezing in this algorithm, and these units continue to adapt after units of new types are added. Although no formal comparisons were done between this algorithm and one in which the units were eliminated, the additional freedom in the extra units seemed to help learning. From a generalization performance perspective, an extra unit of an early more local type is preferable to an additional unit of a later type.

5.3.2 Primary Subroutine: learn_one_unit

This subroutine attempts to achieve the best possible performance with the current network. It is called immediately after each new hidden unit is added. No weight freezing of early hidden units is used.

```
Check performance.
Set the max_pres_number = 2 / failure_proportion.
For each epoch {
    For each token {
        For each presentation (up to max_pres_number) {
            Process token.
            If unsuccessful, or modify_on_success, modify weights.
        }
    }
    If all tokens became successful during the epoch, begin Phase 2.
    If Phase 2, and all tokens became successful, decrease learning rate.
    If Phase 2, and not all tokens became successful, increase rate.
    Check the performance of the net (return if 0 failures).
    If new best-net, store weights.
    If failure_proportion is small, begin Phase 3,
        and set modify_on_success to False.
    If 3 epochs elapse without improving average error or # failures, break.
}
Restore weights of previous best-net. Return.
```

Once a phase change occurs, the algorithm will not revert to the previous phase. There is no interaction between phase changes and changes of hidden units types.

5.3.3 Detailed Explanations

The learning algorithm just presented is robust for the different parsing tasks with which I have used it, but the focus of this thesis has been on solving problems in parsing, and not in solving problems with connectionist learning. The rules that are incorporated into PARSEC's learning algorithm represent heuristics that my experience has supported, and they seem to be well motivated. Additional experiments would be required to verify the precise advantages of the various techniques.

Multiple Token Presentation

This allows the network to concentrate on problem examples without completely skipping learned examples (as in one of the earlier variations of the algorithm). Examples that have been recently "forgotten" are relearned with very few presentations. Difficult tokens require many presentations. Variation of the maximum number of presentations keeps the number of weights changes nearly constant per epoch.

Learning Rate Adjustment

This allows the network to concentrate the most effort on difficult tokens as compared with easy tokens. After all tokens can be temporarily learned on-line, generally, most of

the tokens have been learned well (i.e. they remain successful even at the end of an epoch and not just temporarily). By reducing the learning rate, progressively more weight adjustments are made on non-learned tokens than on learned tokens. On a token that is processed successfully, on initial presentation, at most a single weight adjustment is made (during Phase 1 and 2). On an unsuccessful token, many weight adjustments are made. When the learning rate decreases, it requires more weight adjustments to achieve success for difficult tokens, and they receive proportionately more weight modifications.

Modification on Success

One might believe that it would be a good idea to *never* modify weights following a successfully processed token. However, an oscillation problem emerges when this is done. Large classes of correctly processed tokens suddenly become incorrectly processed. Then, substantial effort pushes them back, but tends to destroy the learning that occurred before.

However, in Phase 3, there are very few remaining unlearned tokens, and weights are not modified following successful processing of a token. Since there are very few unlearned tokens, and since the learned tokens have been repeatedly pushed away from decision boundaries, the oscillation problem becomes less significant compared with the benefit of concentrating all of the effort on the last remaining unlearned tokens.

Often, the last few tokens form a learning sub-problem where higher-order predicates are involved and where local minima may be a problem. For example, in the Phrase module, for the CR task, there are very few examples of "[verb + particle]" constructions, but there are numerous "[verb (or noun)] + [prepositional phrase]" constructions. The network quickly learns that prepositions like "in" begin new phrase blocks, but there is an infrequent exception when "in" is used as a particle with a verb like "send." Phase 3 allows the network to concentrate on the very difficult last tokens while making the most minimal possible changes to the decision boundaries.

Weight Freezing

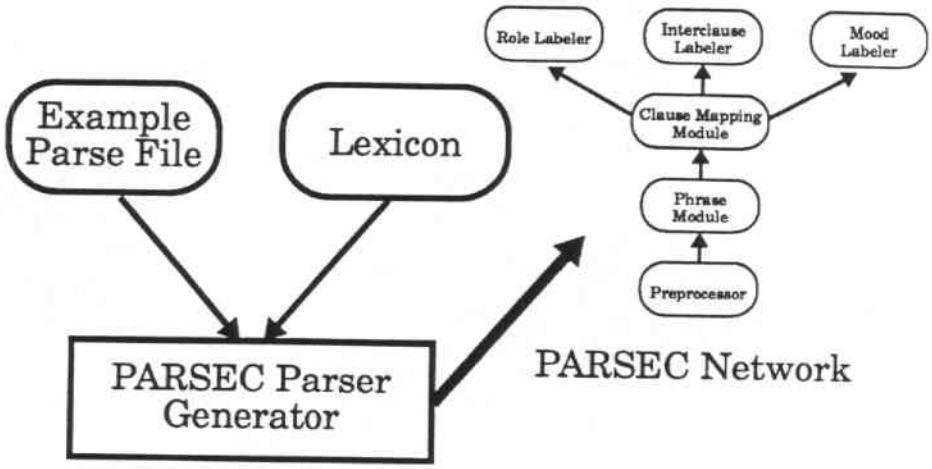
As mentioned briefly above, there is no weight freezing in the algorithm. It seemed to help a little with the problem of forgetting, but it produced larger networks than needed. In particular, additional hidden units were required at the later stages, where more complex hidden unit types are used. This tended to hurt generalization performance. Also, with only a single hidden unit participating in learning, the system was more sensitive to local minima.

5.4 Training a PARSEC Network

The actual process of training a PARSEC network deserves some discussion. Much of the work is automated by the PARSEC Parser Generator (see Figure 5.3). The four steps for producing a PARSEC network for a particular task are:

1. Create an example parse file.
2. Create a lexicon.
3. Train the 6 network modules.
4. Assemble the full network.

FIGURE 5.3 Producing a PARSEC network.



The majority of human effort is spent in the first two steps, although neither step takes too much time (approximately 1 day's work total for a task similar in size to the CR task). The training step takes the most actual time, but little supervision is required. A DecStation 3100 has been adequate for the tasks in this thesis.

5.4.1 Example Parses

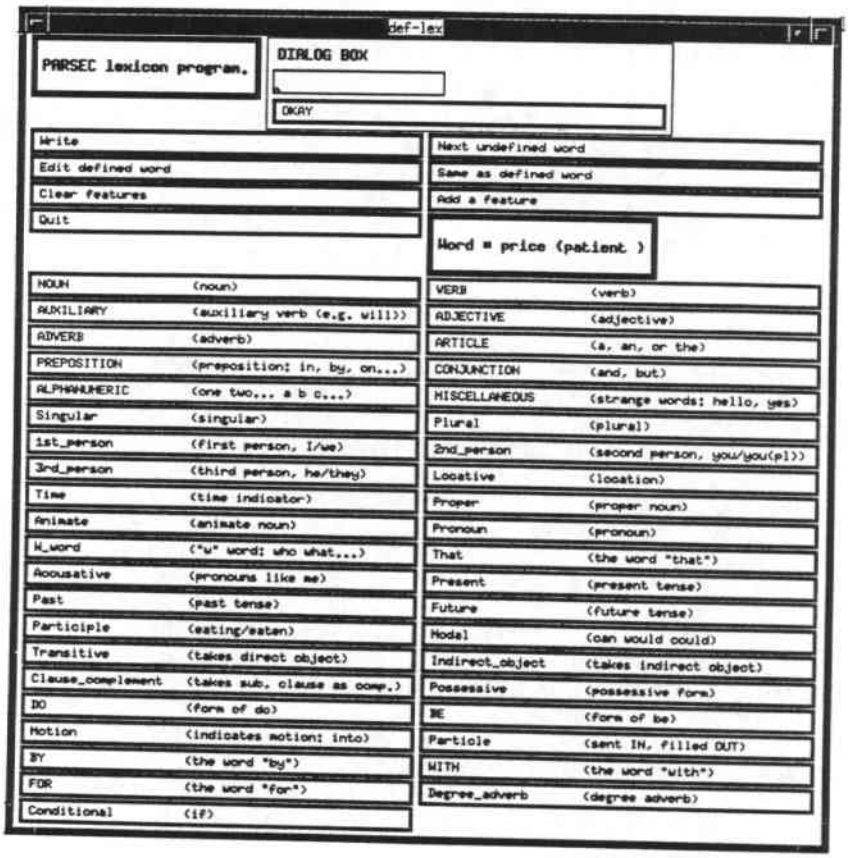
PARSEC needs to see some parses in order to learn its task. The more parses available, the better the parser. Section 4.4.1 shows examples of training parses. The main issues in producing a parse file are *consistency* and *simplicity*.

One must avoid parsing similar sentences in different ways if they have acceptable interpretations that are similar. If a particular construct is parsed differently in two places, the PARSEC network will not be able to learn both examples if there is no distinguishing information within PARSEC's contextual boundaries. Excessive numbers of labels should be avoided. Labels that appear only a few times in the examples can cause undergeneralization. The PARSEC network will not be able to form good decision "rules" for the labels.

5.4.2 Building the lexicon

PARSEC networks require that the words they process be defined as binary feature vectors. For the tasks discussed thus far, these features have been primarily syntactic, with some semantic features (e.g. animate vs. inanimate nouns). However, any features can be used. There is a utility program (referred to as "def-lex" below) that aids in the lexicon generation process.

FIGURE 5.4 Lexicon definition utility program window.



Def-lex looks at a list of the words in the training file along with some information about how they are used in the task (e.g. the role labels of the phrase blocks in which word participates). Def-lex is an X11-based program that allows a user to simply click on buttons to define the words (see Figure 5.4). The job of defining the features is made easier in three ways:

1. Other words with similar features are displayed as features are selected.
2. To define the current word, one can copy features from an already defined word.
3. Def-lex prints out informational files to help find inconsistencies in a lexicon.

Def-lex is initialized with a core set of 41 features that should be useful for any English task (see the figure for those features). It also has a core set of words (both very common function words and some examples of different types of less common words). The user can also create features. For example, if the user believes that it is critical for PARSEC

to know about a class of objects that can fly (e.g. if a role label depends on this property of a noun), a feature can be created for that.

As with the example parses, the two critical properties of the lexicon are consistency and simplicity. Three things should be avoided:

1. Multiple words with different syntactic/semantic function that have identical features.
2. Too many features for the number of example parses.
3. Words with different features that have the same function.

Of these, the first is the most important. PARSEC may not be able to learn certain constructs very well if it doesn't have reliable features for the words in the construct.

5.4.3 Training the modules

There is a utility to process the parse file and produce the actual training files for each of the six module. This program also produces architecture description files for the network modules and the PCL hidden unit types.

The most time consuming task is training the actual modules of the PARSEC network. Fortunately, this is almost completely automated. The user must start up six separate runs, one for each module, but they tend themselves (as described in the previous section). All of the parameters for controlling the learning process are automatically set by the PARSEC training program.

5.4.4 Building the full network

Recall the addition of weight-sharing to the architecture from Chapter 4. During training, the modules are not trained as full networks, but, for efficiency, they are trained as single subnetworks. The modules that result from the previous three steps are not in their final form; they must go through a replication step.

Here are some notes about the structure of the six modules during training for the final PARSEC architecture (these details are transparent to the user):

1. *Preprocessor:* trained as a full module with weight-sharing as shown in Figure 5.2.
2. *Phrase:* trained as a single subnetwork that requires replication following training. This is a very small network. Only one phrase boundary unit is present, and only the minimal amount of word context is present. The network is trained by "shifting" the sentences of the corpus through it. This saves many units during training.
3. *Clause:* same as the Phrase module.
4. *Roles:* trained as a single subnetwork. This module requires replication to account for multiple phrase blocks in a clause, and to account for multiple clauses.
5. *Interclause:* trained as a full module, no weight-sharing.
6. *Mood:* same as the Interclause module.

The replication utility takes care of assembling and replicating all of the structures for the full parsing network. It builds a network just big enough to accommodate the most complex training sentences.

With the full network assembled, the user is free to test the PARSEC parser on any sentences that conform to the vocabulary and network size limitations. The lexicon may be augmented if necessary. It is also possible to generate a network that is larger in some modules than required for the training corpus. However, one cannot increase the total number of clauses without causing some difficulty at the Mood level, and one must retrain the Interclause module.

5.5 Ease of Training

To demonstrate that generating a PARSEC network for a new task is not an unreasonably difficult task, requiring an expert, I selected a volunteer to train a parsing network for a novel domain—a subset of the sentences from DARPA's's Airline Travel Information System task.

I provided the user with brief written instructions, along with the PARSEC software. Each of the directories containing PARSEC software included additional examples and slightly more detailed usage notes than are presented here.

A set of 125 well-formed sentences was selected—73 sentences for training, and 52 for testing generalization. The set was split by random selection. Here are some example sentences:

- Show me all the non-stop flights from Dallas to Denver.
- What does V U slash one mean?
- What do the transport abbreviations mean?

Appendix D lists all of the sentences used.

5.5.1 Training the Network

The volunteer was able to train the network to completion on the training set with little difficulty. Most of the problems arose from inexperience with computers and English (the volunteer was an early-year German graduate student in Computational Linguistics). Here is a breakdown of his effort:

- Initial parse file: 3 hours.
- Initial lexicon: 2 hours.
- Evaluation and debugging of parse file and lexicon: 3 hours.

Training of the modules went smoothly, with occasional problems arising from inconsistent labeling of sentences or features of words. After fixing inconsistencies, training runs were completed.

5.5.2 Performance

The completed network was evaluated on the test sentences that were reserved out of the initial sentences. Note that the test set contained novel lexical entries. For each of the new words, PARSEC's lexicon was augmented, but the new words had not been used during training. The network achieved 67% correct on the test set (75% including near-misses). This was very similar to the performance level achieved by an expert network builder for the CR task (see Chapter 6).

5.5.3 German Language Task

Another volunteer used PARSEC to produce a network for parsing a subset of the CR task in German. PARSEC's architecture did not require any changes. The training procedure followed the same steps used for training PARSEC on a new English task. The network successfully learned the training set (the first three German CR conversations), but no performance evaluations were carried out. Application of PARSEC to non-English tasks will be an interesting area of future research.

5.6 Summary

In this chapter, I presented the algorithms for training PARSEC networks. In addition to describing the Programmed Constructive Learning algorithm, general difficulties facing back-propagation learning were discussed. Learning speed, network size, robust convergence, and algorithm automation were key factors in the development of the three phase training regime. The four step process of building a PARSEC network for a novel domain is automated wherever possible and does not require an expert. PARSEC's robust training algorithms should facilitate further application to new domains and languages.

6 Generalization Performance

Achieving good generalization in large connectionist networks is often a difficult task. The difficulty is compounded by small, statistically unbalanced training corpora in domains where it is desirable to train to completion.[1] This is exactly the case for PARSEC as applied to the CR task.

In this chapter, I will characterize PARSEC's generalization performance in detail. First, I discuss the evaluation procedure for measuring generalization and analyze the performance of the baseline PARSEC parser. Then, the effects of the architectural enhancements and training techniques discussed in the previous two chapters are analyzed on four parsing networks produced by PARSEC. The chapter concludes with a comparison of the generalization performance of PARSEC's best parsing network to an LR parser using three independently hand-constructed grammars.

6.1 Measuring Generalization Performance

In order to assess the generalization performance of PARSEC on the CR task, it was necessary to obtain a substantial corpus of sentences that were disjoint from the twelve conversations that define the CR training corpus. Two generalization sets were collected. The sentences for both sets were generated by people who were not familiar with PARSEC.

1. It is common for grammar writers to be given some corpus of sentences that must be covered by a grammar. It is their goal to write a grammar that covers the sentences in the corpus in the most general way possible. In connectionist modeling, training a network to produce perfect performance on a training corpus hurts generalization performance, and this constraint was relaxed to produce optimal generalization performance (see Section 6.6).

For the first set, volunteers were asked to read the twelve conversations of the CR task, then write down similar sentences that used the same vocabulary. The volunteers were asked to produce novel sentences using words found only in the CR task. However, they were not required to ensure that their sentences were novel or that they did not include words outside of the CR training vocabulary. This set was used during the development of the PARSEC system and, in particular, its generalization enhancements. In the sections that evaluate the different versions of PARSEC, the coverage tests were performed on this set.

The second set was generated in a less restricted manner and was not collected until after *all* parsers to be evaluated had been constructed. A large group of people were asked to each write down an imaginary conference registration dialog. They were not restricted in any way. This resulted in over 20 dialogs that included sentences of greater variety than those collected previously: e.g. sentence fragments, ungrammatical sentences, and some foreign language sentences. This second test set was used to perform a final comparison of the best PARSEC parsing network and the best hand-written grammar.

The raw sentences were edited for vocabulary. Those sentences that could be coerced into the restricted CR vocabulary by minimal changes to 2 or fewer words were so modified, and the remainder were dropped. Then, repeated sentences were eliminated along with sentences that occurred in the CR training corpus.

The first set had 127 sentences initially, and after correcting vocabulary and eliminating duplicates, there were 117 sentences. The second set had 253 unique word strings with 2 or fewer out-of-vocabulary words. After eliminating non-sentences, ungrammatical sentences, and those that could not be corrected for vocabulary without significantly distorting syntax, there were 180 sentences.

Both sets are listed in Appendix C along with the parse score for each sentence of PARSEC's best network, and the best hand-coded grammar for the CR task.

6.2 Generalization Techniques

In the previous chapters I described the basic structure of PARSEC's architecture and training algorithms, along with techniques used to improve generalization. These fall into three categories:

- Representational techniques—methods used to represent symbols and symbol structures.
- Architectural constraints—restrictions on input receptive fields, weight sharing.
- Training techniques—incremental training, incremental addition of hidden units.

Two techniques for forcing generalization were used in all versions of PARSEC. First, separation of word representations into *identification* portions and *feature* portions was used throughout. Second, PARSEC's four Role Labeling modules (one for each possible clause) shared weights.

FIGURE 6.1 Basic structure of the six modules (early PARSEC versions).

6.3 Conference Registration Parser: PARSEC V.1

For the sake of brevity, in what follows, "CRn" should be taken to mean "the PARSEC Version n parsing network for the Conference Registration task."

PARSEC Version 1 was the baseline architecture, corresponding to the description in Section 4.2. Beyond the two techniques described above, no additional attempts were made to enhance generalization. Each network module had essentially the same structure, as shown in Figure 6.1. The output units received input from the hidden units. Hidden units received input from all input units (exclusive of ID units) and all output units. This recurrent structure is similar to that described by Jordan (1986).

CR1's generalization performance was characterized by performance on the 117 sentence testing corpus. Correct parses are those that are perfect in all particulars. Some small percentage of parses are close enough to warrant a category of near-misses. The remainder are labeled incorrect. Below are two examples of near-misses.

Here the network failed to label the "please" (no units corresponding to role labels for that phrase block exceeded an output of 0.5), but it doesn't make a critical difference in the interpretation:

```
([statement]
    ([clause]
        ([action]          give)
        ([recipient]       me)
        ([patient]         your name)
        ([]                please)
```

In the next parse, the network attached "after September thirtieth" to "five hundred dollars" instead of labeling it TIME:

```
([statement]
    ([clause]
        ([agent]           the fee)
        ([action]          is)
        ([patient]         five hundred dollars)
        ([mod-1]           after september thirtieth)))
```

Note that the near-miss category does not make a qualitative difference in the performance comparisons between PARSEC and hand-coded grammars.

TABLE 6.1 CR1 performance.

	Number	Percentage	
Correct	19	16%	23%
Near Miss	8	7%	
Incorrect	90	77%	

TABLE 6.2 CR1 failures broken down by module.

	Errors	Responsibility	Performance
PREP	11	11%	91%
PHRASE	63	65%	41%
CLAUSE	8	8%	81%
ROLES	15	15%	57%
INTERCLAUSE	1	1%	95%
MOOD	0	0%	100%

CR1 generalized poorly (Table 6.1). The overall success rate was 16%. Table 6.2 shows how the errors were distributed among the modules. Since the architecture is hierarchical, failures were counted for the *first* module that failed. In this table, near misses were counted as failures. For each module, two rates are reported. The first is the percentage of errors for which the module is primarily responsible. The second is the module's performance on sentences in which it received correct input from the previous module. For CR1, the biggest source of failure was from the Phrase module. However, the performance of both the Phrase module and the Roles module was substantially worse than that for the other modules—41% and 57% respectively.

Here are two examples of failures in the Phrase module (asterisk indicates missed phrase boundaries):

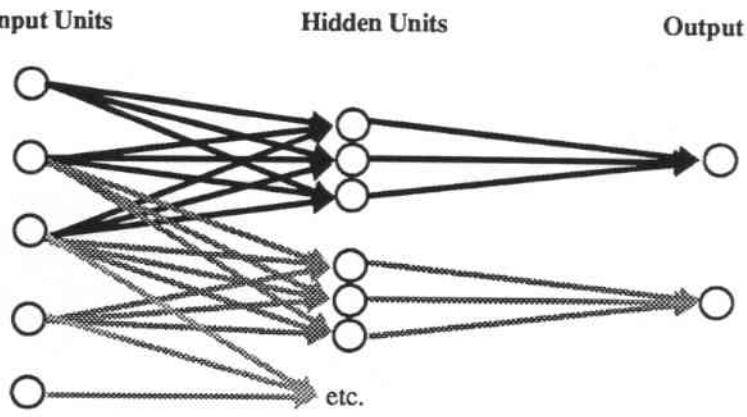
- [Could] [you] [tell] [me * the deadline] [for the conference].
- [Is] [there] [a discount] [for members * of the Information Processing Society].

In these examples, the Phrase module failed to recognize a boundary. This was most likely caused by position sensitivity acquired during training. The units that failed were tuned only to recognize those phrasal boundaries that had occurred in *those* specific positions in the training set. The unit responsible for marking a phrasal boundary between words 4 and 5 was not able to recognize that "tell me the deadline" requires a boundary between "me" and "the."

There were two basic reasons for the poor result:

1. Too many free parameters in the system given the amount of training data.
2. Output units were given access to unreliable non-local information.

FIGURE 6.2 Diagram of localized input connections with weight-sharing.



Regarding the first point, back-propagation learning is opportunistic. If it is possible to "memorize" a training pattern using excess freedom in its connections, a back-propagation network will do so. In the baseline architecture, each of the output units of the Phrase module was responsible only for learning a *particular* position, but they had access to a large number of hidden units with relatively little training data. Regarding the second point, the unit responsible for indicating a phrase boundary between word 3 and word 4 was spuriously influenced (through the hidden units) by words at very distant positions. As mentioned in Chapter 4, architectural constraints in the form of localized receptive fields and weight-sharing were added to PARSEC.

6.4 Conference Registration Parser: PARSEC V.2

There was a great deal of improvement to be made from PARSEC Version 1. Most notably, failures at the lowest three levels accounted for 84% of the errors for CR1. PARSEC Version 2 differed from Version 1 in its lower three modules.

To address the performance difficulties experienced in CR1, PARSEC's Phrase and Prep modules were modified using local receptive fields and weight-sharing. Each output unit was given its own set of hidden units that had limited, position-specific receptive fields. Localized input receptive fields prevent networks from making decisions based on distant input information that can be unreliable.

In addition, weights were shared among analogous units that differed in position. Figure 6.2 diagrams this structure. The black weights have the same values as the analogous sets of gray weights. Each output unit is "looking at" a different piece of the input in the same way as the others. This is similar to weight-sharing in Time-Delay Neural Networks (Waibel *et al.* 1989). In this type of network, each of the decision making modules is required to use *local* information. Furthermore, everything learned from one position is shared among all positions. There is a net reduction in free parameters.

TABLE 6.3 CR2 performance.

	Number	Percentage	
Correct	32	27%	36%
Near Miss	10	9%	
Incorrect	75	64%	

TABLE 6.4 CR2 failures broken down by module.

	Errors	Responsibility	Performance
PREP	3	4%	97%
PHRASE	24	28%	79%
CLAUSE	2	2%	98%
ROLES	52	61%	41%
INTERCLAUSE	2	2%	94%
MOOD	2	2%	94%

PARSEC's new Clause module was also constructed using these two techniques. In order to make this possible, the representation of the output units was changed. Recall the enhanced Clause module architecture from Chapter 4. Instead of being required to produce the clause *identity* of each phrase, it was required to indicate the *inter-phrase points* that begin new clauses and close embedded clauses. This allowed for local decisions, and hence weight-sharing. See Section 4.3.1 for additional details.

The performance of CR2 (Table 6.3) was nearly twice as good as CR1's performance (27% vs. 16%). Replacing CR1's bottom three modules had the effect of reducing the number of errors from those modules from 82 to 29—a reduction of 65%. The most dramatic improvement was in the phrase module. In CR1, it was 65% responsible for failures as compared to 28% in CR2. Its performance increased from 41% to 79%. The dramatic improvement of the early modules shifted the failure responsibility mainly to the Roles module. In CR1, the Roles module was responsible for 15% of the failures, but in CR2, it was responsible for 61%. It is interesting to note that different modules tended to fail on the same sets of sentences. Those sentences that caused problems at the bottom three levels caused problems at the Roles level in CR2.

The example sentences discussed in the previous section were correctly processed by the Phrase module of CR2:

- [Could] [you] [tell] [me] [the deadline] [for the conference].
- [Is] [there] [a discount] [for members] [of the Information Processing Society].

The first sentence then was correctly processed by the remaining modules. However, correct processing of the second sentence by the Phrase module exposed an error in the Roles module (marked by an asterisk):

- [ACTION is] [AGENT there] [PATIENT a discount] [MOD-1 for members] [*RECIPIENT of the Information Processing Society].

The phrase "of the Information Processing Society" should be labeled with MOD-1 to indicate attachment to "for members." The units responsible for performing case-role labeling had a very difficult job to learn. They had to combine information from many locations to assign any of several possible labels given relatively few training exemplars.

The task was complicated by the representation of phrase blocks. The main "content word" of the phrase moved around depending on the phrase. For example, consider "the form" and "the registration form." In the first case, "form" appears in position two of the phrase block, but in the second case, "form" appears in position three. Without ample training data, the Roles module became sensitive to head word *position* without being able to generally learn head word *content*.

6.5 Conference Registration Parser: PARSEC V.3

The only change from PARSEC Version 2 to 3 was in the representation for the phrase blocks in the Roles module. In the baseline phrase block representation, words were represented as a sequence, with each word in a phrase block occupying a single row of units. In the augmented representation, the head words of phrase blocks were mapped into a canonical place. The augmented phrase block representation was introduced in Section 4.3.2.

Generalization performance in the Roles module of CR3 increased from 41% to 67% with this simple modification.[2] The performance of CR3 (Table 6.5) was 44% correct overall. Table 6.6 shows the performance of CR3 broken down by module. In CR3, all of the modules performed better than 90% except the Phrase module (79%) and the Roles module (67%).

The example sentence that caused failure in CR2 (see above) was processed correctly by CR3, but CR3's performance still left a good deal of room for improvement. In the final version of PARSEC, I applied all of the generalization techniques used in CR1–3 plus a few more—most importantly the PCL technique introduced in Chapter 5.

6.6 Conference Registration Parser: PARSEC V.4 (final)

In PARSEC Version 3, the three principal techniques for enhancing generalization were:

2. More elaborate representational changes involving learned mappings from simple phrase blocks to highly regular phrase structures (e.g. explicit slots for determiners, nouns, main verbs, auxiliaries, etc.) were not attempted due to computational considerations, although they might prove useful.

TABLE 6.5 CR3 performance.

	Number	Percentage	
Correct	51	44%	55%
Near Miss	13	11%	
Incorrect	53	45%	

TABLE 6.6 CR3 failures broken down by module.

	Errors	Responsibility	Performance
PREP	3	5%	97%
PHRASE	24	36%	79%
CLAUSE	2	3%	98%
ROLES	30	45%	67%
INTERCLAUSE	6	9%	90%
MOOD	1	2%	98%

- Reduction of free parameters through weight sharing in the lower three modules.
- Localized input connections to enhance information reliability in the lower three modules.
- Representational changes to reduce task complexity for the Roles module.

For the final version of PARSEC, I used each applicable generalization technique for each module of the parser. In particular, weight-sharing and localized input fields were used in the lower four modules (including the Roles module), and the augmented phrase block representation was used in the Clause Mapping module as well as the Roles module. Most importantly, the PCL technique was used in PARSEC Version 4.

Another key difference in the final PARSEC architecture is that it uses *no recurrence* between output and hidden units, whereas all previous PARSEC versions used recurrent connections from the modules' output units to their hidden units. The reasons for dropping recurrence were not related to generalization performance, but rather are related to an implementation issue.

For efficiency, it is best to train single subnetworks instead of full networks that require on-line weight sharing. There is a lower memory requirement, and many fewer operations are wasted on redundant units during processing. For example, in a full Phrase module network, there might be 20 slots for words, but for most sentences, only the first few might be used. However, all of the units must be updated anyway. Training on a single subnetwork eliminates the possibility for interesting recurrence from a set of output units to hidden units. The single output unit in the subnetwork does not provide new information to the hidden units.

TABLE 6.7 CR4/CR4.com performance.

	Number	Percentage	
Correct	78 (68)	67% (58%)	78% (67%)
Near Miss	13 (10)	11% (9%)	
Incorrect	26 (39)	22% (33%)	

TABLE 6.8 CR4/CR4.com failures broken down by module.

	Errors	Responsibility	Performance
PREP	0 (0)	0% (0%)	100% (100%)
PHRASE	6 (10)	15% (20%)	95% (91%)
CLAUSE	11 (16)	28% (32%)	90% (85%)
ROLES	17 (18)	44% (38%)	83% (80%)
INTERCLAUSE	3 (4)	8% (8%)	96% (95%)
MOOD	2 (1)	5% (2%)	97% (99%)

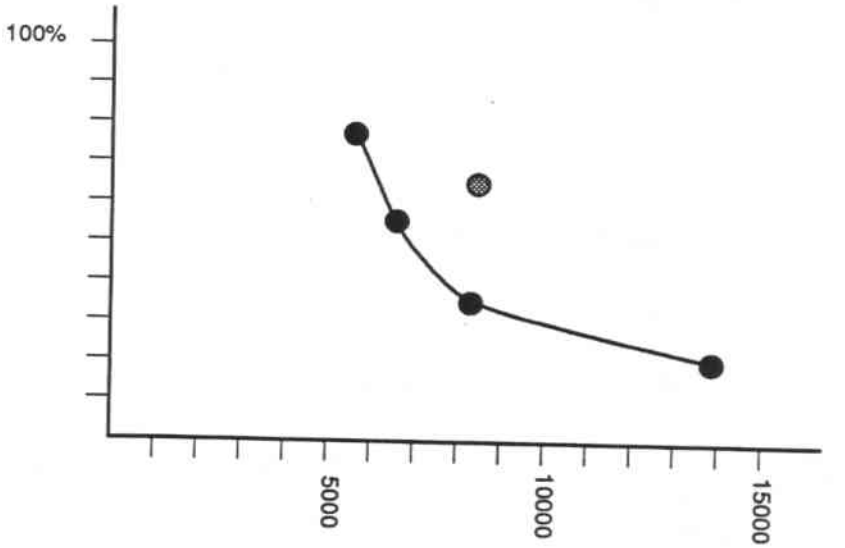
In PARSEC, as opposed to the parsing architecture described in Chapter 3, recurrence is not necessary to learn the task. In the previous architecture (in the Phrase module), input word features were available only for a brief duration, and hidden units had to have access to the output values of both the gating units and the intermediate representational units that captured phrase blocks. In PARSEC, input to each module is buffered and stable. It contains sufficient information for output units to learn their tasks.

In CR4, the modules were not trained to completion. They were trained until they reached a plateau of performance above a training performance threshold of 90%. This made a small difference in each module's performance (e.g. 91% vs. 95% generalization in the Phrase module), but the cumulative effect was important. Since modules feed into one-another, performance rates have multiplicative effects. Another CR parser (CR4.com) was trained to completion.

Table 6.7 shows the performance of CR4 and CR4.com. Numbers in parentheses are for CR4.com. A breakdown by module is shown in Table 6.8. The overall performance of CR4 was 67% (78% including near misses). This was a dramatic improvement over the performance of CR1 (it had just 16% correct in the generalization test). The performance of CR4.com dropped to 58%. Additional failures in the Phrase and Clause Mapping module cause most of the additional errors in CR4.com.

The weakest module was the Roles module (44% responsible for failures). It is easy to understand why this was the case. The complexity of the task relative to the amount of training data was the most unfavorable among all of the modules. For each phrase block, there was a choice of several role labels that had to be made. Compared with the Clause module (two choices per phrase block) and the Phrase module (one choice per word

FIGURE 6.2 Plot of generalization performance vs. number of connections.



boundary and less than three words per phrase block on average), one can see the relative difficulty in assigning twelve role labels, some of which occur infrequently. Also, role labels are more context dependent than the boundary decisions.

6.7 Discussion of CR Parsing Networks

Figure 6.3 shows a plot of network performance versus number of modifiable connections (counting shared connections only once). The four dark points are from CR1–CR4. As the number of free connections decreases, performance increases. However, the gray point falls sharply outside the curve. It corresponds to CR4.com (the PARSEC Version 4 parser trained to completion). It has more connections than both CR2 and CR3, but its performance comes close to that of CR4. The key is that CR4 and CR4.com were trained using the PCL technique. In PCL, hidden units with broad input connectivity are only added after much of the training set has been learned. The late hidden units of CR4.com accounted for a large number of *potentially* harmful connections, but those connections never became excessively sensitive to non-general features.

The errors that CR4 made fell into two categories: undergeneralizations and unavoidable errors. Undergeneralizations result from inadequate exposure to such sentences in the training corpus. Unavoidable errors result from a total lack of exposure to novel constructs. An example of the former is "Please fill out the form which I will send you." In this case, CR4 failed to properly label the relative clause (it labeled it as an independent clause). There was only one relative clause introduced by "which" in the training corpus, and such undergeneralizations are expected. A network has no good way of knowing exactly what is supposed to be novel about a new construct. Any feature that differentiates a new construct from other ones is picked up by back-propagation. Unless

there are a few training examples, a network might just guess wrong. Additional training examples would correct deficiencies of this kind.

An example of a sentence that uses a novel construct is, "How do I go about this?" The phrase "go about" uses "about" in a different way than any of the training examples. One cannot expect PARSEC to get truly novel constructs right.[3] Of course, many generalization failures fall into a gray area, but the distinction between the two types of failures is useful. In failures of the first type, additional examples of sentences similar to those already in the training corpus would probably solve the problem. In the second case, the novel sentence falls more completely outside the realm of expertise of the parser. The failures for CR4 were fairly evenly mixed between the two cases, with 51% arising from undergeneralizations, and 49% arising from novel constructs.

6.8 Comparison to Grammar-Based Parsers

In the previous sections, I described a number of generalization techniques and analyzed their impact on the performance of parsers generated by PARSEC. The section will place the generalization performance numbers in context by comparing them to the performance of a parser using three hand-coded grammars for the CR task. The comparison was made to a Generalized LR parser implemented using Tomita's algorithm (Tomita 1985, 1991).

The first grammar was written for use in the JANUS speech-to-speech translation system (see Chapter 7 for more details about JANUS). The other two grammars were written as part of a contest. A large cash prize ($700) was awarded for best coverage of the 117 sentence CR test set (the second place finisher received $300). All grammar writers were experienced (one a Computational Linguistics graduate student, the other two were active research staffers in the Center for Machine Translation at CMU).

6.8.1 Grammar 1

A grammar was written in a Lexical Functional Grammar formalism that allowed for combining syntax and semantics to produce a frame-based output parse (see Chapter 7 for more information about the output representation). Its output was somewhat different than that produced by PARSEC. The LR parser with this grammar (LR1) produced information about tense and detailed analyses of noun phrases. Apart from that, it generated essentially the same information as PARSEC. It was able to parse only 5% of the 117 test sentences correctly. To be fair, the grammar was created under time-pressure (it still required many hours of work to complete). Also, it was not constructed specifically with coverage considerations in mind. It still provides an interesting data point though. It illustrates that grammar-based formalisms are highly dependent on their grammar-writers and the conditions under which they are required to work.

3. Often, partial parses contain some useful information, but it can be unreliable. The is a parse failure heuristic that is fairly good at detecting when a parse should be good (see Sections 6.8.3 and 7.1.3).

6.8.2 Grammars 2 and 3

The following two grammars were the result of a motivated effort to produce grammars with good coverage. Each grammar took approximately 8 weeks of effort to produce, although the grammar writers were not working on them full-time. Note that the grammar writers were not forced to use any particular grammar or parsing formalism; they were only required to use an existing formal-grammar based parsing system. They made their choices based on ease of implementation in the CMU environment coupled with a desire to win the contest.

Grammar 2 consisted of context free pseudo-unification grammar rules. It took approximately 60 total hours to produce. Rule writing and debugging took the majority of the time. The LR parser using this grammar (LR2) parsed 25% of the testing sentences correctly (26% including near misses). Of the incorrect parses, 80% were NIL outputs, and 20% were non-NIL but incorrect in a major way (e.g. sentence mood or major missing or incorrectly labeled constituent). Fully 60% of the test sentences were rejected as unparsable.

Grammar 3 was also a context free grammar and took approximately the same amount of time to produce as Grammar 2. However, it used a slightly different grammar formalism, with some special rules for handling particles. LR3 parsed 38% of the test sentences correctly (39% including near-misses). Of the incorrect parses, 89% were NIL results, and 11% were non-NIL but deficient. Here again, a large portion of the test set was rejected as unparsable (54%).

6.8.3 Discussion

The performance comparison is quite clearly in PARSEC's favor. The large difference between PARSEC's best performance and the best performance of the hand-coded grammars deserves some discussion. (Appendix C shows the test sentences along with the scores CR4 and LR3.)

LR3 agreed to some degree with CR4 on which sentences were "difficult" (i.e. those that resulted in poor parses). Of the sentences for which CR4 produced poor or near-miss parses, 72% of them produced poor or near-miss parses in LR3. Just 28% of CR4's problem sentences were processed correctly by LR3. Conversely, CR4 parsed 62% of LR3's problem sentences correctly—accounting for the large performance disparity. Those sentences that caused problems for PARSEC tended to cause problems with the hand-coded grammar, but the reverse did not hold to nearly the same degree.

One key feature of all the hand-coded grammars is that they reject so many sentences as unparsable. This can occur in two situations:

1. The grammar does not recognize its final state as being a legal sentence termination.
2. The grammar fails to model some construct in a sentence.

In the both cases PARSEC has systematic advantages. First, PARSEC is trained to *incrementally* parse sentences as they develop. That is, PARSEC is trained to try to parse initial sentence fragments properly. Second, due to the various techniques for enhancing generalization that cause PARSEC to rely on local information, the decision-

making units of PARSEC are less likely to become sensitive to minor variations in semi-distant constituents. Localized receptive fields and weight-sharing both act to prevent PARSEC from becoming overly sensitive to irrelevant information.

Another interesting comparison is in the classification of incorrect parses. A critical issue is how to detect such parses. Of the three grammar-based parsers, the percentage of easily detectable (NIL) parses ranged from 80% (LR2) to 100% (LR1). Using a simple heuristic for detecting obviously wrong parses that is based on the output values of the labeling units of a PARSEC network (see Section 7.1.3), it is possible to reject 81% of the incorrect PARSEC parses as such. This heuristic also rejects 50% of the near misses as being incorrect parses. While PARSEC's ability to reject poor parses is not up to that of all the hand-coded grammars, it does fall within the range. However, there must be some trade-off between robust generalization and "tightness." This will be discussed in more detail in Chapter 7.

6.9 Final Generalization Comparison

Since the performance comparison in the forgoing section so heavily favored PARSEC, it seemed possible that the testing set was in some sense "tainted" since it had been available during PARSEC's development. As mentioned in Section 6.2, a second test set was collected *after* all of the comparisons had been completed using the initial test set. This set was both larger (180 sentences) and more diverse.

CR4 achieved 66% correct on this set (73% including near-misses). LR3 achieved 41% on this set (no near misses). These figures are not significantly different than the earlier figures. Additional analyses also parallel the foregoing discussion.

6.10 Summary

In this chapter, I have analyzed the generalization performance of PARSEC and compared it to more standard parsing methods. In order to improve PARSEC's baseline generalization three techniques were used:

1. Representational techniques: augmented phrase block representation.
2. Architectural constraints: localized receptive fields and weight-sharing.
3. Training techniques: programmed constructive learning.

These had an impact on:

- The number of free parameters to train.
- The relative importance of local versus non-local information.
- The types of information available to the network to make decisions.
- The task complexity for each of the modules.

Table 6.9 is a summary of generalization performance for each of the parsers described in this chapter, ordered by type (PARSEC or human generated) and performance on the

TABLE 6.9 Comparison of all parsers on the 117 sentence set.

	Correct	*Correct+Near*	*Sharing*	*PCL*	*APR*	*Completion*
CR4	**67%**	78%	Lower 4 mods	Yes	Yes	No
CR4.com	58%	67%	Lower 4 mods	Yes	Yes	Yes
CR3	44%	55%	Lower 3 mods	No	Yes	Yes
CR2	27%	36%	Lower 3 mods	No	No	Yes
CR1	16%	23%	None.	No	No	Yes
LR3	**38%**	39%	—	—	—	(Yes)
LR2	25%	26%	—	—	—	(Yes)
LR1	5%	5%	—	—	—	(Yes)

117 sentence testing set. PCL stands for Programmed Constructive Learning, and APR stands for "augmented phrase representation."

The performance of PARSEC's best parser for the CR task was nearly 70% (discounting near-misses). This compares very favorably with about 40% for the best of the grammar-based parsers. PARSEC learned its "grammar" from little more than 200 English sentences whereas the human grammar writers had the full knowledge of English in addition to the sentences they were required to parse correctly.

A key lesson from this chapter is that each of the enhancements in terms of representational changes and architectural constraints were facilitated by explicit structure in the architecture. In a monolithic network, it is difficult to see how one could make a serious impact on generalization performance. In the structured, modular approach presented here, it is possible to *engineer* a connectionist network and produce dramatic performance increases.

7 Speech Translation and Noise Tolerance

This chapter covers experiments that tested PARSEC's tolerance of various types of noise, including an application to the JANUS speech-to-speech translation system (Waibel *et al.* 1991; Jain *et al.* 1991). Parsing is an interesting area in which to explore noise tolerance for two reasons. One, the grammar-based approaches to parsing tend to be brittle in the face of noise. Simple problems in a sentence such as subject/verb number disagreement can trip up many grammar-based parsers. Two, the input is symbolic and atomic—not real-valued and fuzzy. The noise that is encountered often involves non-subtle changes to input patterns.

PARSEC was tested on three sources of noise:

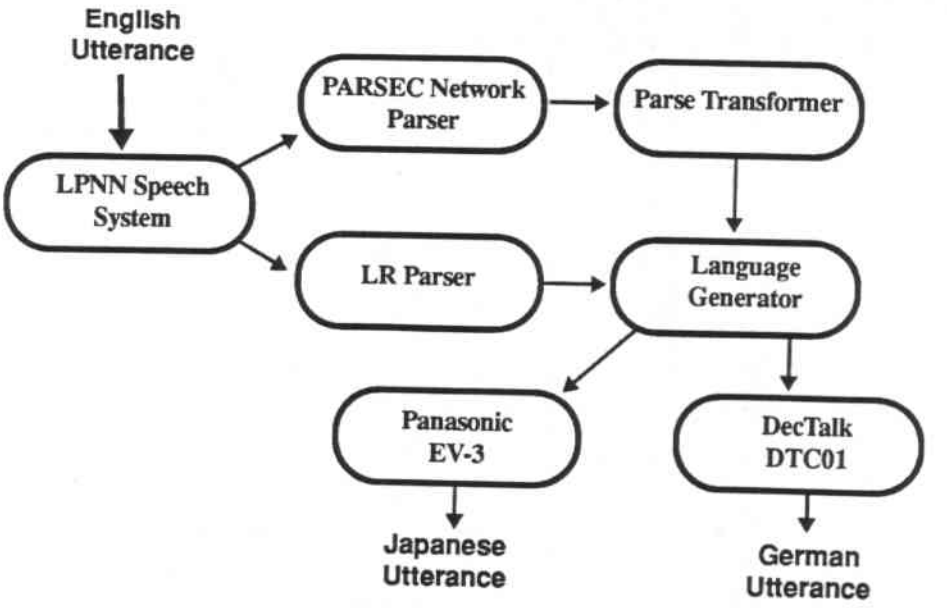
1. Speech recognition errors in JANUS.
2. Ungrammatical sentences (synthetically generated) from the CR domain.
3. Transcriptions of verbal user interaction from the ATIS corpus.

Traditional grammars tend to reject irregular input, partly by design and partly because their nature predisposes them to regular structures. PARSEC offers some degree of noise tolerance without any explicit modeling. Some researchers have developed more noise-tolerant grammar-based systems (e.g. Saito and Tomita 1988, Ward 1990), but the explicit noise-modeling that they built into their systems can be *trained* into PARSEC networks.

7.1 PARSEC in Speech-to-Speech Translation

The main goal of PARSEC's application within JANUS was to compare the performance of PARSEC with a grammar-based parser on noisy output from a speech recognizer. A secondary goal was to show a real application of a PARSEC parsing network.

FIGURE 7.1 High-level structure of the JANUS system.



JANUS is a speech-to-speech translation system developed at CMU that operates on the conference registration task. It is built in a modular fashion and allows for evaluation of components that use alternative computational strategies.

JANUS translates continuously spoken English speech utterances into Japanese and German speech utterances. Figure 7.1 shows the overall structure of the JANUS system. It can utilize two processing pathways—one with a PARSEC network as a parsing front-end, and one with an LR parser.

7.1.1 Speech Recognition and Synthesis

Speech recognition in the JANUS system is provided by a connectionist, continuous, large vocabulary, Linked Predictive Neural Network (LPNN) system (Tebelskis *et al.* 1991). This system, as used in JANUS, is speaker-dependent, has a vocabulary of 400 English words, and uses a statistical bigram grammar of perplexity 5. The LPNN module can produce either a single hypothesized textual sentence or the N best hypotheses (at a substantial cost in processing time). This system, when using the bigram grammar, produces the correct sentence as one of the top three choices in 90% of the cases, with additional gains within the top nine choices (for this work, N = 9). However, the system achieves only about 70% correct on the single best hypothesis.

In single-hypothesis mode (F-best), the parsing component must attempt to process the best LPNN hypothesis. With multiple hypotheses (N-best mode), the parser passes the first *parsable* hypothesis to the language generator (or returns failure if there are no parsable hypotheses).

Speech synthesis is provided by two commercially available devices, a Digital DECtalk DTC01 system for German output, and the Panasonic Text-to-Speech System EV-3 for Japanese output. Each of these systems takes a textual or phonetic representation of a sentence as input, and produces the sounds of the spoken utterance through an audio speaker. The following section describes the alternative translation modules.

7.1.2 Knowledge Based Machine Translation

JANUS's translation module is based on the Universal Parser Architecture (UPA) (Tomita and Carbonell 1987). It is a knowledge-based machine translation system that is capable of performing efficient multi-lingual translation. The system consists of a parsing component and a generation component. The parsing component is arbitrary (as long as it produces output in the appropriate form).

In one version of JANUS, Tomita's efficient generalized LR parsing algorithm is used as the basis for the parser (Tomita 1985). After pre-compilation of a grammar, fast table-lookup operations are all that is necessary to parse utterances. The performance of this module approaches real-time. Language generation also approaches real-time. It is performed by a system that compiles a generation grammar into LISP functions (Tomita and Nyberg 1988).

The standard UPA system requires a hand-written grammar for each language to be used for parsing and generation. The system uses a Lexical Functional Grammar formalism, and both syntactic and semantic rules are encoded in the grammar. Multi-lingual parsing is achieved by writing grammars for each of several languages. The universal parser takes text input in the source language and produces an "interlingual representation"—a language-independent frame-based representation of the meaning of the input sentence. The universal generator takes this as input, and uses the generation grammar to make the transformation into the appropriate text in the target language. Figure 7.2 shows an example of the input, interlingual representation, and the output of the JANUS system.

7.1.3 Using PARSEC in JANUS

There are two problems in trying to apply PARSEC to the JANUS system:

- PARSEC's output is not suitable for direct processing by the language generation module.
- PARSEC networks have no internal failure indicator; they will process any sentence and produce an output.

Transformation of PARSEC's output into the interlingual representation required by the generation module is accomplished by a separate program. It operates top-down using simple match rules to instantiate case-frames and their slots. The slots of the case-frames are then filled using more match rules. The algorithm is opportunistic in that it attempts to create a reasonable interlingual output representation from any input. Occasionally, the interlingual representation will cause the language generation module to produce a NIL output. This is reported as a parsing failure.

FIGURE 7.2 Example of input, interlingua, and output of JANUS.

Input

Hello is this the office for the conference.

Interlingual Representation

```
((CFNAME *is-this-phone)
 (MOOD *interrogative)
 (OBJECT ((NUMBER sg) (DET the)
          (CFNAME *conf-offce)))
 (SADJUNCT1 ((CFNAME *hello))))
```

Output

Japanese: MOSHI MOSHI KAIGI JIMUKYOKU DESUKA
German: HALLO IST DIES DAS KONFERENZBUERO

The second problem is more easily solved. A set of simple heuristics are able to reject many of PARSEC's bad parses. The four main rules are:

1. All phrase blocks must be legally labeled. A Role labeling unit must have activation of greater than 0.5 to be considered legal.
2. No phrase block should have multiple labels.
3. Each clause must have a phrase block with either an ACTION or MISC label.
4. Roles requiring an ACTION (e.g. PATIENT as opposed to MISC) must be present with an ACTION.

I also experimented with real-valued parse metrics derived from output unit activations, and this is discussed later.

7.1.4 Performance Comparison

The two versions of the JANUS system (each with a different parser) were tested on a single reading of all 204 sentences of the CR task by the speaker that the system was trained on. This was a restricted test due to limitations in the LPNN system. Sentences from outside the twelve conversations were not tested.

JANUS-LR

Table 7.1 shows the performance of JANUS using the UPA parsing/translation component (JANUS-LR) on the full database of twelve conversations in F-best and N-best modes.

TABLE 7.1 Performance of JANUS using an LR parser.

Correct recognition and translation	N: 173 F: 140	N: 176 F: 143	N: 86% F: 70%
LPNN error but OK translation	N: 3 F: 3		
Incorrect recognition and translation	N: 14 F: 10	N: 28 F: 61	N: 14% F: 30%
Incorrect recognition no parsable utterance	N: 14 F: 51		

There are four possible outcomes for processing an utterance:

1. Correct recognition and translation: the LPNN produced exactly the right word sequence as a hypothesis, and it was selected and correctly processed by the parser and generator.
2. Incorrect recognition, but OK translation: the parser chose an incorrect recognition hypothesis, but JANUS produced the proper translation anyway.
3. Incorrect recognition and translation: on choosing an incorrect recognition hypothesis, JANUS produced the wrong translation.
4. Incorrect recognition and no parsable utterance: no parsable hypotheses.

The performance was 86% correct translation in N-best mode. This number included a small number of type two outcomes. The 13% of cases where JANUS-LR failed were almost evenly split between outcomes of types three and four.

The First-best performance was substantially worse than the N-best performance (dropping from 86% to 70%). In this mode, the parsing/translation components were forced to use incorrect recognition results quite often, and the performance degradation was expected. Although, in a real system, one would probably never force the system to use only the first hypothesis, it was interesting to examine how this affected the different parsers. In F-best mode, the errors from the LPNN more closely modeled a more realistic testing condition in which either the testing utterances or the speaker would be different from the training conditions.

JANUS-NN (JANUS using PARSEC)

Table 7.2 shows the performance of JANUS using PARSEC (JANUS-NN) in the two operation modes. The network that was used was CR4.com (see Chapter 6 for details about this particular network). The N-best performance was worse than that for JANUS-LR. JANUS-NN often returned parses for incorrect hypotheses. These were frequently rejected as poor candidates by JANUS-LR. Thus JANUS-LR, with its tighter language

TABLE 7.2 Performance of JANUS using PARSEC.

Correct recognition and translation	N: 149 F: 140	N: 164 F: 157	N: 80% F: 77%
LPNN error but OK translation	N: 15 F: 15		
Incorrect recognition and translation	N: 31 F: 35	N: 40 F: 47	N: 20% F: 23%
Incorrect recognition no parsable utterance	N: 9 F: 12		

model, had a performance edge. It was able to robustly reject initial hypotheses and find the correct utterance. The PARSEC network was unable to reject incorrect recognition results as robustly as the LR parser.

However, the performance of JANUS-NN in First-best mode did not degrade nearly as much as for JANUS-LR. In fact, JANUS-NN outperformed JANUS-LR in First-best mode (77% versus 70%). The performance difference arose from the number of incorrect recognition hypotheses that JANUS-NN parsed and translated correctly.

7.1.5 Discussion

In First-best mode, both systems correctly processed the correct recognition results. This accounted for 140 tokens. JANUS-NN outperformed JANUS-LR by more often producing correct translations when the LPNN hypothesis was incorrect. The major difference between the two systems was that JANUS-NN was more likely to successfully parse an utterance that did not correspond to the actual spoken utterance. Here are two examples of incorrect recognition that were translated correctly by JANUS-NN but not JANUS-LR:

- LPNN: *Will* be expecting you.
  ACTUAL: We'll be expecting you.
- LPNN: We have a special *forms* for the summary.
  ACTUAL: We have a special form for the summary.

In both cases, the LR parser failed to return a parse. The flexibility of the PARSEC network is reflected in several ways in Tables 7.1 and 7.2. JANUS-LR reported many more failures in First-best mode than in N-best mode. When forced to use an imperfect utterance, JANUS-LR was more likely to fail to parse it than was JANUS-NN. JANUS-NN reported parsing failure much less often in First-best mode—PARSEC was able to "make do" with imperfect utterances.

The other side to this behavior occurred in N-best mode. The flexibility of the PARSEC network in JANUS-NN caused a performance loss because it sometimes did not look far enough down the hypothesis list to find the correct utterance. It simply stopped and returned a parse of an incorrect hypothesis. It's possible that more stringent parse-failure heuristics would allow JANUS-NN to more closely parallel the performance of JANUS-LR in N-best mode. However, significantly more stringent parse heuristics would begin to look somewhat task-specific. The set of heuristics that were used were well motivated.

Of course, errors in speech recognition were the root cause of incorrect translations in this test. It was known *a priori* that correct hypotheses would be properly parsed and translated, since each parser was either trained or built specifically for the 12 conversations. Currently, a number of improvements to the speech recognition component are being evaluated. These range from enhancements at the acoustic level to better language modeling (Tebelskis *et al.* 1991).

Non-Binary Parse Metrics

I attempted to develop non-binary parse evaluation schemes that made use of the connectionist parser's real-valued outputs. One scoring metric was the average of the activation values for labeling units whose outputs exceeded 0.5. Higher scores indicated higher confidence in a parse. The idea was to use a real-valued parse-metric and evaluate all N hypotheses, then choose the best one instead of selecting the first acceptable hypothesis. Unfortunately, none of the real-valued parse metrics improved JANUS-NN's performance.

There are several reasons for this. The LPNN system did not produce acoustic scoring information for its hypotheses, only rankings. This made it very difficult to know when to throw out a higher ranked hypothesis in favor of a lower one. With acoustic scoring, in a case where one LPNN hypothesis had very high acoustic score compared to the others, one could discount small differences in the parse scoring metric.

Also, since the training corpus was fairly small yet highly diverse, many of the sentences were "outliers" in some characteristic. Therefore, a large number of correct hypotheses necessarily received quite low parse scores, and lower ranked hypotheses often yielded better parse scores.

Lastly, the language model used to constrain the LPNN's search was an unsmoothed bigram grammar that was trained using only the 204 sentences of the CR task. It tended to produce either correct hypotheses or substantially different hypotheses that happened to be close to other sentences in the corpus. So, often, incorrect hypotheses would receive high parse scores. A language model that was less biased to the CR training set might have produced more divergent incorrect hypotheses, leading to lower parse scores.

Performance and Grammar Tightness

Recall the generalization performance comparisons of Chapter 5. The grammar used in the LR parser for JANUS was "Grammar 1." The PARSEC network used for JANUS (CR4.com) generalized more than ten times better than the LR parser. Given this, it's not surprising that JANUS-LR was able to outperform JANUS-NN in N-best mode. JANUS-LR essentially had a table of the 204 sentences, and it could reject almost any-

thing that deviated from the table. Note, however, that it is possible to write grammars that incorporate techniques that allow them to handle more varied input including noise (Saito and Tomita 1988). In this case though, the PARSEC network was at a substantial disadvantage because of the difference in coverage.

7.2 Synthetic Ungrammaticality

This section covers an experiment in which PARSEC and its competing grammars were evaluated on synthetic ungrammatical sentences from the CR task. Table 7.3 shows 50 ungrammatical sentences. Each is a corrupted version of a sentence from the twelve conversations of the CR task. I chose to modify sentences from the actual training corpus of the PARSEC network in order to decouple the noise issue and the generalization issue.

The corruptions ranged from making very minor changes such as replacing "a" for "an" (e.g. sentence 1) to modifying verb phrases as some foreign speakers tend to (e.g. by using "to + infinitive verb" instead of "properly conjugated verb" as in sentence 19). There were no spontaneous language phenomena such as restarts or interjections. Each of the sentences in the table has a reasonable interpretation that is very close to that of the original uncorrupted sentence. Some of the sentences may seem silly or artificial, but some people actually use similar constructions (especially in certain dialects of American English and the English of non-native speakers).

CR4.com (the PARSEC network used in JANUS) produced reasonable parses for 33 of the 50 sentences, arguably reasonable parses for 2, and poor parses for 15. Using the simple parse failure heuristic from Chapter 7, 11 of the 14 poor parses were recognized as such, 2 of the acceptable parses were labeled as poor, and the near misses were both rejected. This is fairly robust recognition of parse acceptability (about 90%).

Recall the three hand-coded grammars from Chapter 5. Their performance was:

- Grammar 1: 1 correct parses, 49 NIL outputs.
- Grammar 2: 19 correct parses, 19 NIL outputs, 12 incorrect parses.
- Grammar 3: 17 correct parses, 32 NIL outputs, 1 incorrect parse.

Grammar 1 was able to parse a single sentence whose only change was a replacement of "a" for "an." Grammar 2 came closest to PARSEC's performance, but still fell far short (38% versus 66% correct). Grammar 3, the best hand-coded grammar from a coverage standpoint, correctly parsed nearly as many sentences as did Grammar 2. When it failed, it nearly always failed by producing a NIL output instead of a poor parse. As with the grammatical coverage experiments from Chapter 5, the hand-coded grammars behaved in very different ways. This appears to be a problem with the process of hand-writing grammars. One cannot predict with certainty how such grammars will behave when stressed, since they depend so heavily on the grammar-writers.

Several seemingly simple grammatical irregularities, which PARSEC parsed well, caused problems LR2:

- "Yes, that are right." (subject/verb number disagreement)

TABLE 7.3 Table of corrupted CR sentences.

1. hello is this a office for the conference
2. yes that are right
3. i would have like to register for the conference
4. do you already have a registration forms
5. then i+ll sent you a registration form
6. could you to give me your name and address
7. the address are five thousand forbes avenue pittsburgh pennsylvania one five two three six
8. i+ll send you special form immediately
9. if there is any questions please ask me at any time
10. thanks you
11. this are the office for the conference
12. what should i did
13. first you must register with registration form
14. do you already got a registration form
15. not yet please send i a form
16. name is judy simpson
17. is a attendance fee required
18. yes two hundred dollars per person is required as registration fee
19. may i to help you
20. i like attend the conference
21. how can i to apply
22. please fill a registration form
23. do you have a one
24. okay then i send you a registration form
25. name is harry bovic
26. would you spell last name please
27. me got it
28. hello this is conference office
29. could you give me some information about application fee
30. how much will it cost if i apply in the conference right now
31. but if you are apply next month it will cost you three hundred twenty five dollars
32. the proceedings and the reception is included in the application fee
33. i am member of the information processing society
34. are there a discount for members
35. how can i to pay
36. payment be made by bank transfer
37. please remit to our bank account which be mentioned in the announcement
38. deadline are the end of the year
39. you is welcome
40. i would like to contribute a paper to conference
41. me tell you the topic of the conference
42. conference covers a wide area of research related to interpreting telephony
43. we be expecting linguists and psychologists as participants
44. what is official language of the conference
45. i don+t to understand japanese at all
46. yes there be simultaneous interpretation service into english
47. that are helpful for me
48. i would like know details of conference
49. do you have conference announcement
50. would you mind tell me your name and address

- "Do you already have a registration forms?" (plural with determiner)
- "If there is any questions, please ask me at any time." (object/verb disagreement)

More difficult grammatical irregularities caused problems for both systems. These sentence were incorrectly processed by CR4.com and LR2:

- "I like attend the conference." (instead of "would like to attend")
- "Would you mind tell me your name and address?" (instead of "telling")

There was not much agreement between the various parsers about which sentences were difficult, in contrast to the coverage tests. There were cases in which PARSEC produced poor parses, but LR2 produced good parses. For example:

- "I'll send you special form immediately." (missing determiner)
- "Not yet, please send I a form." (instead of "me")

Note that each of the hand-coded grammars was constructed without irregular input in mind. However, the PARSEC network was only trained on grammatical sentences (the same as those that the grammar writers were required to cover). In applications where one must attempt to process any input with few rejections (e.g. in human-machine interfaces), PARSEC's behavior is desirable.

PARSEC networks show some tolerance for ungrammaticality without explicit noise modeling. But if one views grammars as models of language, grammars (or grammarless parsing networks) that accept ungrammatical input could be considered undesirable. Building prescriptive models of language was not the focus of this work.

7.3 Spontaneous Speech Effects: The ATIS Task

DARPA's ATIS task uses a machine interface paradigm where users verbally query an airline reservation system for information and book flights. Part of the ATIS effort involves detailed transcriptions of the interaction including spontaneous language phenomena. Whereas in all of the previous experiments with PARSEC, the networks were trained on grammatical input, this experiment included noisy input in the training and testing sets.

Another key difference between this and previous PARSEC parsing networks was that this network was trained to produce primarily semantic domain specific parses instead of the more general parse structure used previously. This was done to more closely approximate the SQL queries required for the full ATIS task of database information retrieval. Here is an example of a training parse for this experiment:

```
([statement]
    ([list-flights]
        ([fly]          show me)
        ([mode]         nonstop)
        ([travel]       flights)
        ([arr-time]     arriving at three pm)
        ([to-loc]       to denver)))
```

The case-role labels reflect the semantics of the constituents that they label. The structure is amenable to conversion into an SQL query of a database of airline information.

This is much different than the parse would have been under the previous parse representation (e.g. as used by the novice PARSEC trainer in Chapter 5 for ATIS):

```
([statement]
    ([clause]
        ([action]     show)
        ([recipient]  me)
        ([patient]    nonstop flights)
        ([mod-1]      to denver))
    ([rel (nonstop flights)]
        ([action]     arriving)
        ([time]       at three pm)))
```

Despite the substantial differences between the parse representations, the PARSEC system needed no revision. The parse file and lexicon for the new task simply reflected the characteristics of the new parse representation. Note that the notion of phrase block is somewhat different—corresponding to semantic chucks instead of more syntactically oriented chunks. The parse file had many domain-specific role labels, and the lexicon had additional lexical features such as "can-fly."

Currently there are several thousand utterances available in the ATIS corpus. However, only 100 were used for training the PARSEC network here. The coverage of the trained network on novel sentences was not particularly impressive (63% adequate[1] parses of a test set that included some novel lexical entries), but the network learned to process the noisy training sentences without difficulty. In a limited evaluation on novel noisy sentences, the network seemed more sensitive to novel constructs than noise *per se*. Substantially more sentences would be needed for training and testing to fully understand the effects of noise in this domain. However, in principle it is possible to model noise effects directly within the PARSEC framework.

7.4 Summary

In this chapter, I explored PARSEC's behavior on three kinds of noisy input. In the first test, on data including speech recognition errors in the JANUS system, PARSEC was able to outperform an LR parser on particularly noisy data. But it showed an inability to assign preferences to sets of competing hypotheses, and its performance was worse than that of an LR parser when each system was given access to multiple candidate hypotheses. However, the LR parser had the advantage of an extremely tight grammar that aided in robust hypothesis selection.

In the second test, on synthetic ungrammatical sentences from the CR task, PARSEC was able to outperform three hand-written grammars. In this test, the network was trained on grammatical input only, but it showed some noise tolerance without any explicit modeling. The third test suggested that further improvements on parsing noisy sentences might be possible by explicitly training on noisy examples.

1. This number included parses that were not perfect in all particulars, but were good enough so that a proper response could be generated within the ATIS paradigm.

8 Prosodic Information

The utility of prosodic information in speech processing systems is clearly established. For example, Waibel (1988) showed that speech recognition systems could benefit from several types of supra-segmental information: e.g. duration, pitch, intensity, and stress. However, building systems that make effective use of prosody has proved difficult. Steedman (1991) developed a grammar formalism that made use of intonational annotations to aid in parsing. While being an important step, there is still no known method to automatically extract the intonational information that his system used. In order to begin to make effective use of prosody, systems must be able to use what is available in the speech waveform. Huber (1988) made some progress using prosodic information derived from speech data, but his system was hand-designed.

This chapter describes a limited experiment demonstrating that PARSEC is able to effectively learn to utilize intonation derived automatically from actual speech waveforms.

8.1 Task

In conversations (especially over the phone), it is common for people to intone sentences that are grammatically declarative to indicate that they are seeking confirmation of the declarative statement. For example, consider this conversation fragment:

Secretary: Hello.

Caller: Hello, my name is Harry Bovic.

Secretary: How can I help you?

Caller: I've registered already, but I haven't received a packet yet.

Secretary: I see... Your name is Bovic?

In the last sentence, in the absence of intonational information (and without dialog-level context), a parser will assign a declarative mood—this is incorrect. The secretary is attempting to confirm the name of the caller not trying to tell him his name.

I trained PARSEC to utilize pitch contours extracted from actual speech waveforms so that trained networks could disambiguate short declarative sentences intoned as statements from those intoned as questions. The network was trained such that in the absence of any pitch information, it would assume the declarative mood.

8.2 Data

I collected multiple utterances of several short sentences (up to three words) from two speakers for the majority of the data. The recordings were made under benign conditions with a 16KHz sample rate and 16 bit samples. Each of the utterances was spoken as a statement and as an intoned question ten times by two speakers (one male and one female). These were the utterances:

1. "Okay"
2. "Right"
3. "That's right"
4. "You have one"
5. "That's okay"
6. "Conference office"

This amounted to 240 utterances. Some other utterances were recorded for additional testing, and these will be described later.

8.3 Signal Analysis

Pitch detection is not a solved problem in speech analysis. It is a very complex perceptual phenomenon. A pitch detector must select those peaks in an input waveform corresponding to the primary harmonic signal in voiced regions of the waveform. I was fortunate to be able to use a neural network based speaker-independent pitch tracker developed at CMU and at the Oregon Graduate Center (Barnard *et al.* 1991; Zhou 1991). The pitch tracker was trained using high-quality speech (as was recorded for this work), but it was trained on a different database (TIMIT data) than was used here.

The input to the pitch tracker is an unprocessed digital waveform, and the output is a list of waveform sample numbers that correspond to valid pitch pulses. Computation of the pitch frequencies from this is a simple matter. One nice property of the pitch tracker is that it indicates no pitch pulses for unvoiced regions of an utterance. This saves an additional and possibly troublesome step in the pitch calculation. Recognition of voiced regions is non-trivial.

Figure 8.1 shows an example pitch output for a male speaker on "Okay?" There are some problems with noisy pitch values, but overall, the pitch contour is quite nice. The majority of incorrect pitch values come from pitch doubling and from unvoiced regions

FIGURE 8.1 Example of unsmoothed pitch contour.

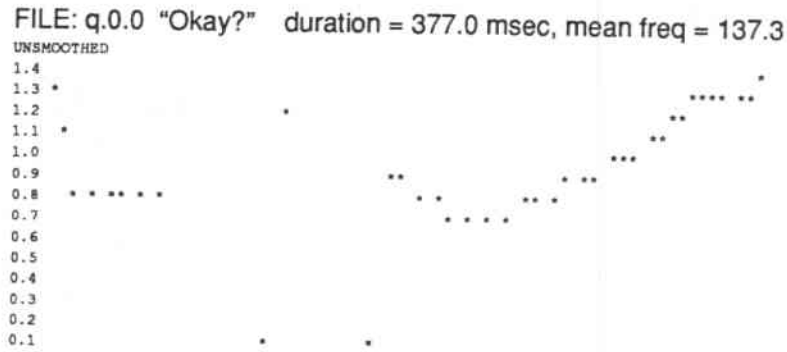
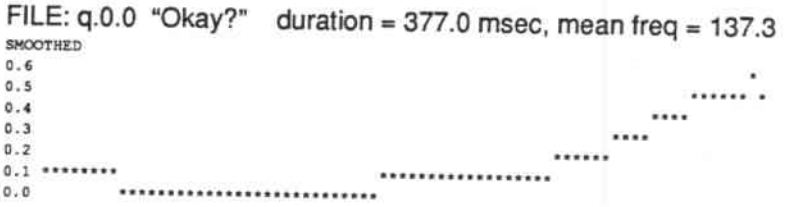


FIGURE 8.2 Example of smoothed pitch contour ("Okay?).



where the pitch tracker failed to suppress pitch peak detection. In Figure 8.1, the area where there are few pitch points plotted corresponds to the unvoiced "k" in "okay." Pitch doubling occurs when peaks in the waveform corresponding to higher order harmonics are misrecognized as being part of the primary harmonic signal.

I devised a very simple method to process and smooth the pitch contour to make PARSEC's job a little easier:

- Pitch pulses whose neighbors give very different pitch values are deleted.
- The entire pitch contour is normalized according to the average pitch, and it is remapped onto a fixed-width one-dimensional vector (length 75).
- The vector is smoothed using a window average. Portions of the vector with no pitch values receive window average values from the neighboring right and left areas with pitch values.

Figure 8.2 shows a smoothed contour. Note that the smoothed contours are translated such that their minimum pitch value is 0. The Y axis of the figure corresponds to the

FIGURE 8.3 Example of smoothed pitch contour ("Okay.").

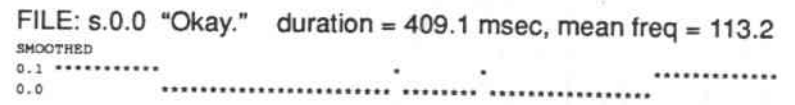
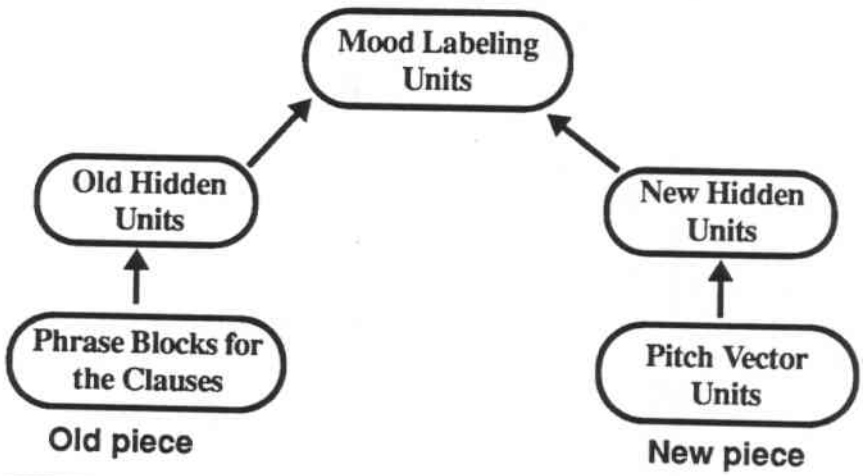


FIGURE 8.4 Augmented Mood Module.



value that PARSEC receives as input. Figure 8.3 shows the smoothed contour for "Okay." The variation is actually smaller than the somewhat coarse graph indicates. Appendix E shows additional raw and smoothed pitch contours for two speakers (one male, one female). Generally, the cleaned pitch contours are visually recognizable as coming from statements vs. questions.

8.4 Extensions to PARSEC

PARSEC required minimal augmentation. Using the fully trained Mood module (PARSEC Version 4 network) as a starting point, I added an additional set of input units to represent the time-evolving pitch contour of an utterance (see Figure 8.4). These units (the Pitch units) corresponded to the fixed-width pitch contour vector (75 units). I also defined a new PCL hidden unit type for the Mood level that received input connections from the Pitch units.

The Pitch units' values were simply the smoothed, normalized vector values from the pitch tracker. The pitch vector varied over the course of processing an input utterance to simulate an on-line pitch input. At each time step in the processing, the Pitch units were

updated with a new pitch vector corresponding to the portion of the waveform being processed.

For each of the following experiments, an augmented Mood module was trained with a data set that included training data with no pitch information (from the initial pitch-free training) plus different combinations of the pitch-labeled utterances listed in the Data section.

Note that it was certainly possible to train a new Mood module from scratch, but the demonstration that additional information from entirely new modalities could be added to a pre-existing module is useful. The new information source was utilized to the extent needed to learn novel training patterns without forgetting old ones.

8.5 Experiments

Three experiments were performed which measured PARSEC's tolerance to different variations in test conditions.

8.5.1 Novel Gender/Speaker

The training set consisted of 80% of the utterances for the male speaker, and testing was done on the remaining 20% of the male speaker's utterances and all of the female speaker's utterances. PARSEC added only one additional hidden unit to the existing Mood module to learn all of the training tokens. The trained augmented Mood module processed 100% of the test tokens correctly.

This result was somewhat surprising given that generally, in speech recognition applications, performance on new speakers (especially across genders) is substantially lower than on the speaker that was used for training.

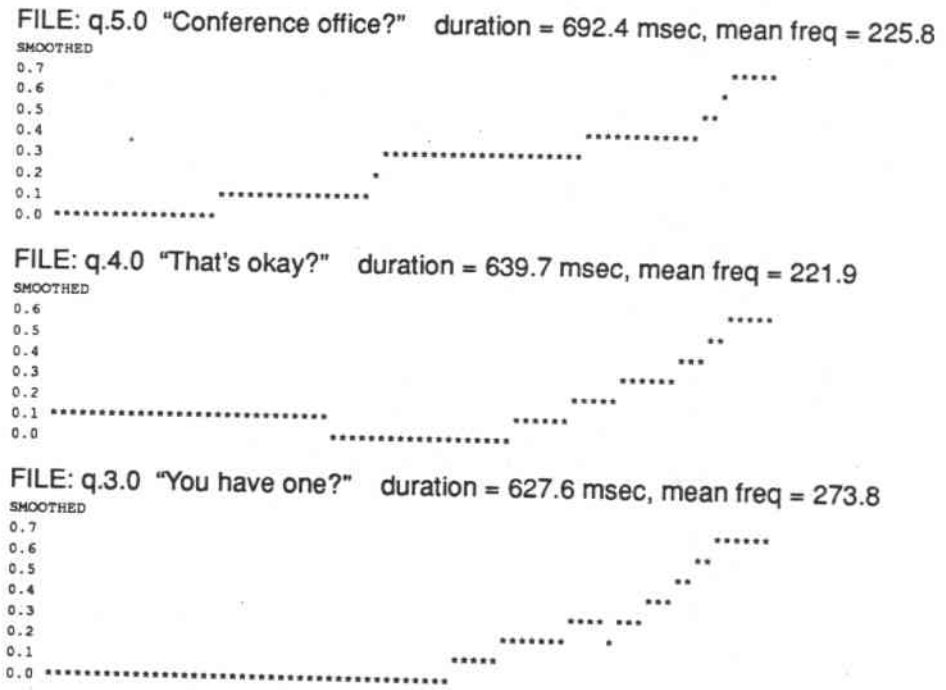
The excellent result arises from a few reasons. First, the pitch contours are normalized for average pitch and for length. This eliminates a large source of variation between speakers. Second, the differences between pitch contours for normal declarative statements and declarative statements *intoned* to be questions are substantial and fairly consistent. Third, since the network was able to see example contours for each of the utterances, it didn't really have a very difficult task to learn.

8.5.2 Novel Utterances

For this experiment, the training set consisted of 80% of four utterances for both speakers. The testing set was the remainder—including all of the examples of the two utterances not present in the training set. Note that in this experiment, PARSEC was able to see examples of *both* speaker's utterances. Again, PARSEC added only one additional hidden unit to the Mood module.

PARSEC processed all of the testing tokens correctly in this case as well. This was somewhat surprising. The pitch contours for one of the testing utterances ("Conference

FIGURE 8.5 Different pitch contours for questions.



office?") appears to be quite different than the others (see Figure 8.5). It is markedly more step-like, but this variation did not affect the augmented Mood module. The contour was still much different than that of statements, but the network captured the relevant differences without becoming sensitive to the particular question contours observed during training.

8.5.3 Combination

To test the system under an even more difficult condition, some additional utterances were collected from two male speakers (intoned as statements and questions, with three repetitions):

- "Your name is Bovic"
- "You have a form already"
- "Two hundred dollars"
- "Pittsburgh"
- "Is that correct"
- "May I help you"

These sentences are quite different from the earlier sentences in both length and syntactic content. The final two sentences are constrained grammatically to be questions

FIGURE 8.6 Weakly inflected utterance of male speaker.

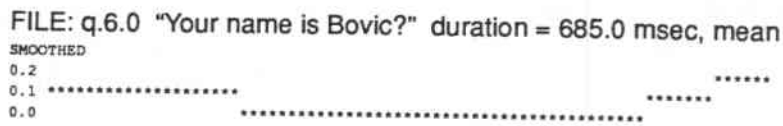
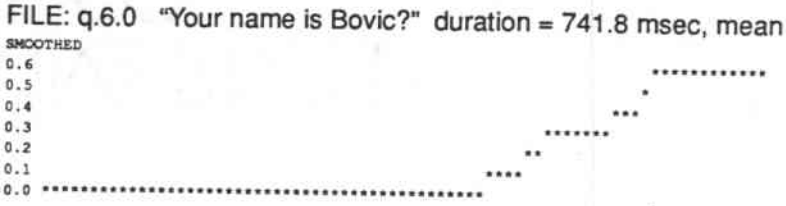


FIGURE 8.7 Normally inflected utterance.



regardless of intonation, but the speakers were instructed to attempt a very flat intonation in one case, and a natural intonation in another case.

A network was trained to process all utterances of the female speaker—all of these were different than those listed above. On the first male speaker's novel utterances, the network had 100% performance. The second male speaker produced weakly inflected utterances, and the network was able to properly process 72% of the utterances.[1] All of the failures were on utterances that were supposed to be recognized as questions. Of these, a fair proportion cause human listeners to produce incorrect or inconsistent answers. Figure 8.6 shows an example of one of the more challenging utterances. There pitch rise is extremely weak as compared with the same utterance of the first male speaker (Figure 8.7).

To test the hypothesis that it was the weakness of the second speaker's pitch rises that caused the failure, the criterion for recognition was modified slightly. Under the new criterion, an utterance was labeled as a statement only if the network's output was greater than 0.7 for the Mood module's statement unit. Otherwise, the utterance was labeled as a question. Using this criterion, the performance of the network on speaker 2 was 97%, and it remained at 100% for speaker 1. But for the weakness of the second speaker's pitch rises, it seems that the network would have achieved excellent performance. Alternatively, if the network had seen some examples of weak pitch rises in intoned questions during training (the female speaker produced sharp rises consistently) it probably would have achieved high performance.

1. The second male speaker was somewhat ill at the time of the recording. This affected his vocal ability to a degree.

The last two of these utterances were those that were grammatically constrained to be questions. The network was not trained on any such utterances using any pitch information, but these sentences were all recognized correctly as questions. There was a slight difference in the Mood module's question unit output value in the two different utterance conditions. In cases where an utterance was intoned "flatly," the output was less extreme than in cases where an utterance was intoned with a natural pitch rise.

The network had to distinguish between grammatical statements whose mood could be affected by intonation and grammatical questions whose mood could not be affected by intonation. The network learned to combine the syntactic and intonational information and generalized to a novel case when tested using new speakers of a different gender.

8.6 Application to Speech-to-Speech Translation

Without any additional modification to the existing JANUS system, JANUS using the augmented PARSEC network was able to produce the proper translations of statements that had been intoned to be questions (see Figures 9.8 and 9.9 for an example).The figures show: the network activity, PARSEC parses, and JANUS-produced Japanese translations for "This is the conference office" intoned as a statement and then as a question.

The layout of the figures is the same as that in Chapter 4, except that there is a group of units marked "vector." These units are PARSEC's input representation for the pitch contour of the utterance. The activation pattern in the Mood module of Figure 8.8 shows a high activation for the top unit (indicating a statement mood). In Figure 8.9, the activation pattern in the Mood module is inverted, with high activation in the bottom unit (indicating a question).

The following (correct) translations resulted:

- "This is the conference office." was translated to "Kaigi jimukyoku desu."
- "This is the conference office?" was translated to "Kaigi jimukyoku *desuka*?"

8.7 Summary

While being a somewhat modest demonstration in a limited domain, the result is impressive. Using a straightforward augmentation to an existing trained PARSEC network, it was possible to teach the network to utilize an entirely new data source from a non-symbolic modality. This new data source was immediately useful in the JANUS system.

PARSEC networks are not affected by their input modality. If the information is consistent, PARSEC can make use of it without the complex schemes required of many symbolic systems. In principle, it should be possible to utilize other types of information from the speech signal (such as energy patterns) to aid in different aspects of parsing.

FIGURE 8.8 PARSEC parse/translation of "This is the conference office."

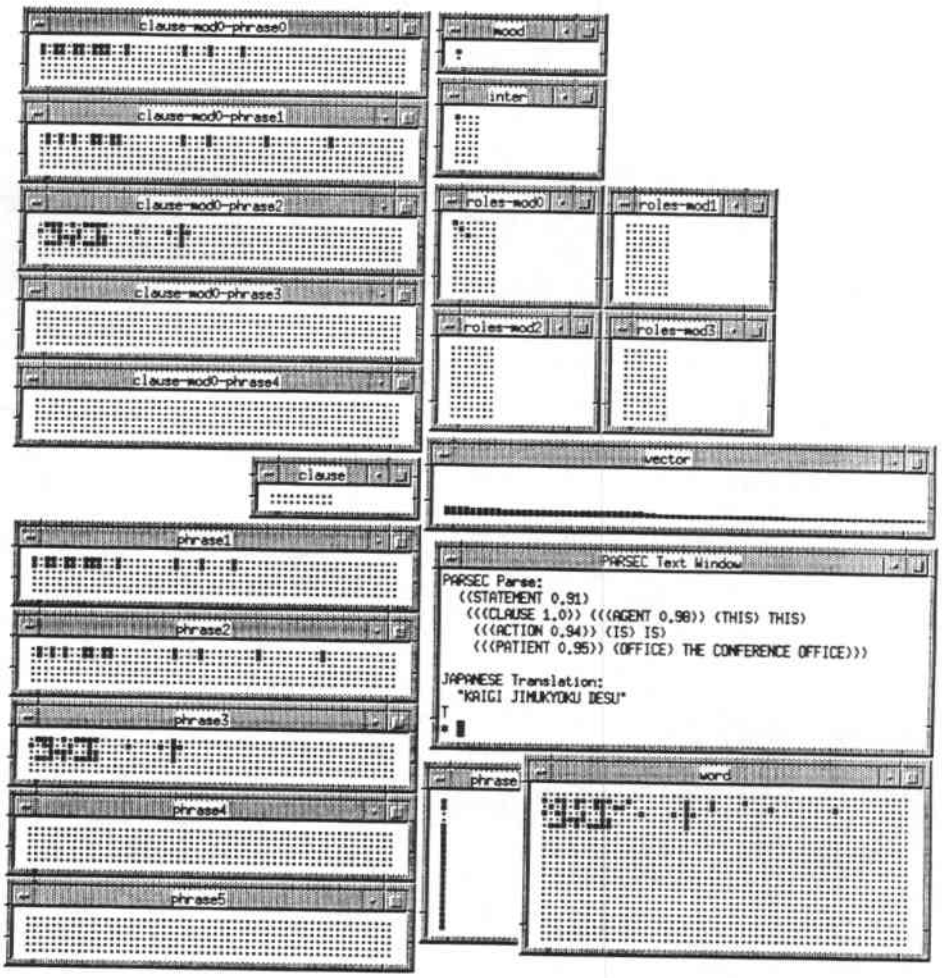
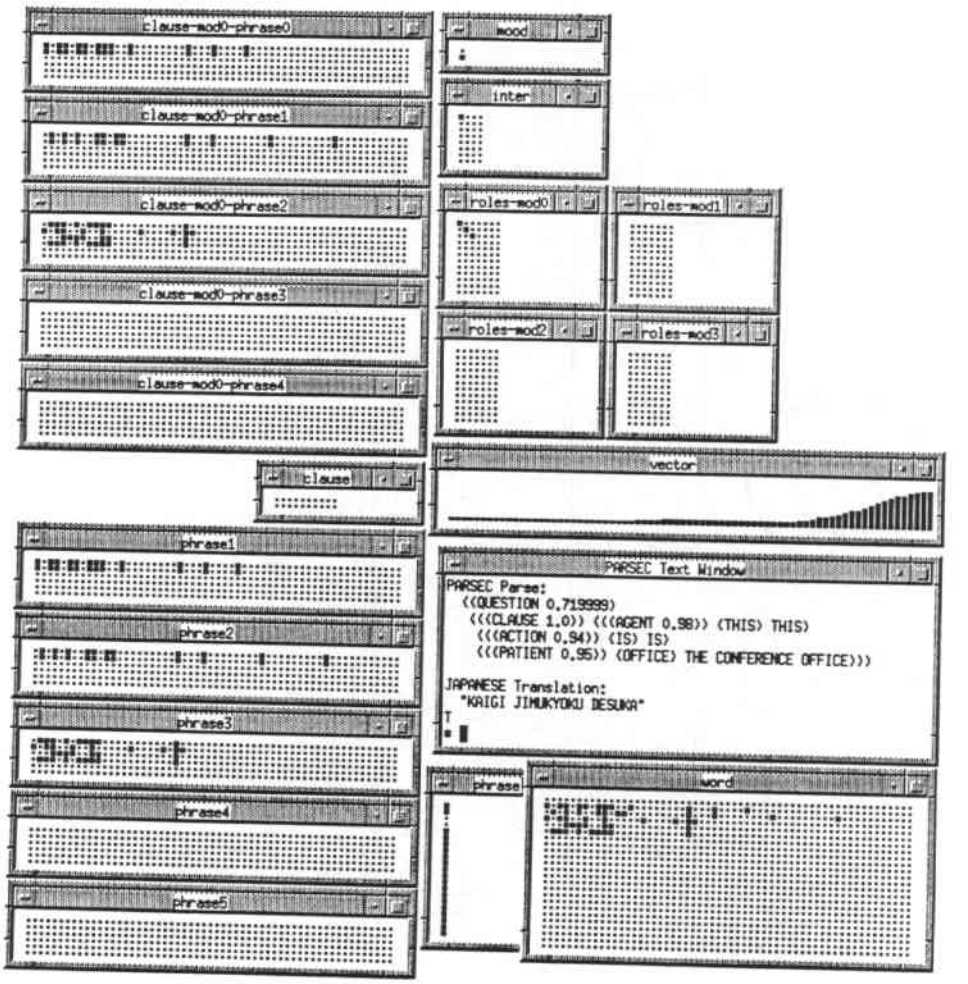


FIGURE 8.9 PARSEC parse/translation of "This is the conference office?"



9 Conclusion

The goal of this work was to develop a connectionist parsing system that demonstrated three central advantages over more traditional parsing systems:

- Ability to learn general grammars from exposure to example parses.
- Tolerance to noise of the kinds found in speech tasks.
- Ability to incorporate multi-modal input.

The PARSEC connectionist parsing system meets these requirements. In this chapter, first I summarize the primary results of the thesis. I then reintroduce the question brought up in Chapter 2 regarding where PARSEC fits within the hierarchy of computational models. Next is a discussion of the contributions of the work to the fields of natural language processing, connectionism, and speech processing. After that, some of the shortcomings of the PARSEC system are enumerated. The chapter concludes with some ideas about future research directions.

9.1 Results

The full PARSEC system includes a connectionist parsing architecture, training algorithms, and utility programs to make parser generation relatively efficient. The PARSEC system was extensively evaluated and compared with grammar-based parsers. This section briefly reviews the key features of PARSEC along with a summary of the performance results.

9.1.1 Architecture

The architecture is highly structured. The architecture is composed of two stages, a transformational stage and a labeling stage, each of which are made up of three mod-

ules. The first stage is a series of three transformational modules where PARSEC builds the basic structure of an input sentence. The second stage is a set of three parallel modules for labeling the constituents of a sentence. The modules are trained using a variant of back-propagation learning.

The two central principles in designing the final PARSEC architecture were:

1. Don't force networks to learn mundane transformations that can be done through the use of clerical programs.
2. Where possible, inject domain knowledge into the architecture.

The PARSEC architecture is a hybrid of symbolic and sub-symbolic representation and computation. Words are represented as preprogrammed binary feature patterns, but the network modules use sub-symbolic hidden representations for making decisions. PARSEC relies on connectionist learning for those tasks where the means of achieving desired behavior are not obvious.

9.1.2 Learning

The two key aspects of learning are:

1. Generalization: Does the system learn something useful, or has it just memorized its training examples?
2. Automation: Does it require excessive effort to train for a particular task?

Generalization

If trained in the most straightforward manner, a PARSEC network exhibits poor generalization. By incorporating knowledge about language, parsing, and connectionist learning into the architecture and learning algorithms, PARSEC is able to generalize very well as compared with traditional hand-written grammar-based parsers. On the CR task, PARSEC's best network achieved 67% generalization, and the best hand-coded grammar achieved 38% (discounting near-misses).

I used four techniques to enhance PARSEC's generalization performance:

1. *Weight-sharing in analogous subnetworks*: This reduces free parameters and creates position insensitivity in those modules where it is applicable.
2. *Localized receptive fields*: This prevents units from using distant, unreliable information to make decisions.
3. *Programmed constructive learning*: This forces PARSEC to learn more general "rules" initially and less general ones for exceptional cases.
4. *Representational conventions*: Where possible, internal network representations are structured so that PARSEC can locate and exploit reliable information sources (e.g. the augmented phrase block representation with a special head slot).

These techniques are not specific the CR task or even to parsing *per se*, and some of the techniques should be applicable to very diverse domains.

Automation

PARSEC's performance results would be of little practical value without a reliable automated way for generating trained networks. The training algorithms are highly automated and robust. A non-expert was able to train a PARSEC network for a novel task with little difficulty, and the resultant parser performed well. In another experiment, PARSEC was able to learn a small German parsing task—a surprising result considering that PARSEC was designed only with English in mind.

9.1.3 Noise Tolerance

PARSEC networks offer a degree of noise tolerance "for free." That is, one can train a network on a set of well-formed example sentences and expect the trained network to show a degree of noise tolerance. Networks do not become sensitive to fine-grained grammatical regularities such as subject/verb agreement. This is in contrast to the brittleness of grammar-rule-based parsing systems. Networks also show tolerance to some speech recognition errors.

In the JANUS speech-to-speech translation system, using the speech recognition system's best hypothesis (often noisy), a PARSEC network was able to increase performance from 70% to 77% over a hand-coded grammar. PARSEC was not able to robustly select from multiple ranked hypotheses, and its performance was worse than that for the hand-coded grammar in multiple-hypothesis mode (80% vs. 86%). In other experiments on synthetically generated ungrammatical sentences, PARSEC showed a wider tolerance to loose grammar than did traditional systems.

The ATIS task was used as a data source for testing tolerance to real spontaneous speech effects and ungrammatical input from actual transcriptions. In this experiment, a PARSEC network was trained on a mixed set of sentences that included some noise. The trained network showed some noise tolerance, but more extensive evaluations using larger data sets are needed.

9.1.4 Multi-modal Input

A goal in speech processing has been to be able to develop systems that effectively combine textual information with acoustic information. Connectionist learning algorithms simply respond to statistical regularities in the activation patterns of their input units. PARSEC is able to use pitch information from a speech signal in order to disambiguate mood in short utterances where syntax alone is insufficient. This work opens the door for more ambitious attempts at synergistically combining multiple information sources in speech processing.

9.2 Is PARSEC a Dynamic Inferencer?

I now return to the discussion of whether or not PARSEC fits Touretzky's notion of a dynamic inferencer (Touretzky 1991). To review, a dynamic inferencer is a model in which:

- The number of input patterns is exponential in the length of the input, and in which the model must have a similar number of internal states.

- Training set size is at most polynomial in vocabulary size.
- The model is able to use novel intermediate states in response to novel combinations of input patterns.

PARSEC seems to fall short of being able to deal with involved compositionality of the sort one would like to see in a true dynamic inferencer. For example, Touretzky shows an example of a PP attachment problem where a PP attachment influences the attachment of the parent PP:

- the car in the junkyard with the dog
- the car in the junkyard with the dog on the hood

In the first case, the preferred attachment is "the dog" to "the junkyard." In the second case, the preferred attachment changes. Attaching "the hood" to "the dog" changes its preferred attachment to "the car." PARSEC could certainly learn to perform the proper attachment for a particular set of cases, but it probably wouldn't develop the ability to generalize the behavior to truly novel situations.

Interestingly, each of PARSEC's modules is essentially a categorizer (the simplest of Touretzky's models), but through a clever representation of the input words that maintains the type/token distinction, and the hierarchical modular structure, PARSEC fulfills some of the requirements of a dynamic inferencer.

9.3 Contributions of the Thesis

This work makes contributions in natural language processing, connectionism, and speech processing.

9.3.1 Natural Language Processing

This work recasts the parsing problem as a series of pattern recognition subproblems, and it offers a non-rule-based method to solve the problems. This is in sharp contrast to traditional grammar-based parsing systems. In traditional systems, decisions about phrase structure are made by application of rigid rules. In PARSEC, such decisions are made without rules within a back-propagation hidden layer. Instead of advocating the elimination of rules or using back-propagation networks exclusively in parsing, I would interpret PARSEC's success as an opportunity for more researchers to develop parsers that rely on softer decisions.

PARSEC networks reliably induce general grammars from exposure to example parses. With additional work to extend PARSEC to more substantial domains, it may prove to be more robust than writing grammars by hand. Humans don't have the capacity to write rules of the complexity that PARSEC can learn. PARSEC can learn to combine very diverse features (syntactic, semantic, statistical, and acoustic) in subtle ways that might not occur to a human. In large corpora, there might be sufficient regularity to induce robust complex behavior of a very different nature than has been explored so far.

One aspect of performance that has not been emphasized in the thesis is parsing *speed*. PARSEC networks produce parses in *linear* time with respect to input length. This com-

pares very well with theoretical bounds on most grammar-based parsers. In the current implementation, on serial hardware, typical parses take several seconds, as compared with near real-time speed for the LR parser that was used for performance comparisons. However, owing to the parallel nature of connectionist networks, it should be possible to implement real-time versions of PARSEC networks on many types of parallel hardware.

9.3.2 Connectionism

PARSEC is a successful application of connectionist techniques to a real problem. In such a young field as connectionism that has produced high expectations and few real demonstrations, this alone is an important contribution. However, there are some additional areas in which PARSEC makes a contribution.

Structure and Connectionist Engineering

Probably the single most salient feature of the PARSEC architecture is its structure, both at a macroscopic level in terms of modules and at a finer level in terms of constraints placed on connectivity within modules. This feature differentiates PARSEC from many other connectionist architectures for language processing. The contrast is not merely cosmetic. Imagine an unstructured back-propagation network for parsing. The input and output could be essentially the same as PARSEC's, but the network would be required to learn the complex transformational and labeling steps with no guidance. Although I have not built such a network, I feel confident that it would perform very poorly on novel sentences.

By using structure in PARSEC, I was able to *engineer* the architecture. In response to positional sensitivity in the Phrase module, I added weight-sharing and local receptive fields. In response to poor generalization in the Roles module, I augmented the phrase block representation with head slots. These types of engineering improvements are difficult to imagine being learned in an amorphous parsing network. The lesson is that structure should be viewed as a tool to enhance performance.

Constructive Learning

Constructive learning techniques were very effective in PARSEC. They helped eliminate a variety of thorny learning problems and provided a way to create an automatic task-independent PARSEC parser generator. Using fully connected networks with predetermined architectures for serious performance oriented tasks may not be a wise approach to take. Of course, some hand-tuning of automatically generated architectures might afford performance improvements, but at least the constructive techniques provide a good starting point.

The Programmed Constructive Learning technique developed during this work brought a surprisingly substantial improvement in generalization performance. The principle is simple—to prevent a network from utilizing excess freedom in its hidden units' input connections, use hidden units in sequence with progressively wider input connectivity. The PCL technique should have useful applications in other domains. Any task in which a network designer has a principled idea (or even just a good hunch) of which input units have more reliable information than others is a candidate for some version of this technique.

9.3.3 Speech Processing

The two primary contributions of this thesis regarding speech processing are a method for building task-specific parsers that exhibit noise tolerance and a method for utilizing acoustic information at the language level. The former problem has traditionally been left to linguists, and while highly trained linguists can produce very good grammars, the process is time-consuming and is not without uncertainty as far as the quality of finished product.

The latter problem of incorporating acoustic information has not been adequately addressed by earlier work, nor do I claim that this work solves the problem. However, the augmented PARSEC system that uses pitch information takes an important step. It was possible for PARSEC to learn to use real pitch contours with an extremely straightforward augmentation. No complex schemes for knowledge combination were required.

9.4 Shortcomings

PARSEC does not attempt to solve many of the open problems in natural language processing. Notably, its method for handling lexical ambiguity would be insufficient for more substantial domains. Also, no attempt was made to attack anaphora resolution or other deep problems in NLP.

The restrictions on vocabulary are not too stringent because of the feature representation, but restrictions on the size of corpora that PARSEC can handle at this time are limiting. More powerful computers or better learning algorithms are probably necessary before PARSEC can attack substantially larger tasks. To make very strong claims about PARSEC relative to established NLP systems, corpora of thousands of sentences must be processed.

An undesirable aspect of PARSEC's architecture is the length constraints placed on the input and intermediate representational structures (e.g. in the CR task, phrase blocks could contain up to five words). However, many of the constraints can be eliminated since most of the modules are constructed by replication. Instead of replicating the hardware, it is possible to either dynamically generate needed hardware, or to use single copies of the small subnetworks and appeal to external hardware to manage buffering and memory. Even so, there are other constraints not so easily relaxed such as the level of center-embedding possible. This must be predetermined.[1]

9.5 Future Work

There are many possible directions to pursue. Each of the difficulties mentioned in the previous section opens an avenue of research. Of those problems, I feel that approaching larger corpora using PARSEC is the most important. It would require better solutions to the other problems, and it would allow for evaluation of PARSEC with more convincing

1. This aspect of PARSEC's behavior may not be particularly undesirable since humans seem to have a fixed center-embedding depth, beyond which they tend to fail to parse properly.

comparisons to other parsing formalisms. In particular, the problem of lexical ambiguity is interesting.

I envision a possible extension to PARSEC in which words are modeled using multiple distinct word senses as in some of the early connectionist work in NLP. However, using the same engineering approach to robust back-propagation as in the rest of PARSEC, it might be possible to *learn* to choose proper word senses. The word sense units could be provided with a number of pieces of information, ranging from direct connections from other word units to important pieces of the current parse. Lexical priming effects, syntactic constraints, and semantic interactions among words might emerge in a computationally useful way. The issue of scale is a serious one though. The obvious localist approach may prove to be computationally intractable for some time.

Another interesting possibility is tighter coupling between PARSEC and speech recognition. The possibility for synergistic interaction is especially hopeful for connectionist speech systems. The idea of exploring the use of other acoustic information types than pitch is especially appealing. Virtually any acoustic feature might prove useful to solving some aspects of the parsing problem, and PARSEC may be able to make use many types of acoustic information.

In contrast to making stronger links with lower-level processing, it is also possible to pursue the other direction—that of dialog or conversation level interaction. For example, in domains such as Conference Registration, speaker identity (caller or secretary) contains useful information. In principle, PARSEC should be able to make use of dialog level information to guide the parse in much the same manner as it made use of acoustic information.

A Network Formalism

This appendix contains the equations that define the behavior of units in the networks used throughout this thesis (for additional details, see Jain 1989). The key features of the network formalism are:

- Well-behaved symbol buffers are constructed using groups of units.
- Units have temporal state; they integrate their inputs over time, and decay toward zero.
- Units produce the usual sigmoidal output value *and* a velocity output value. Units are responsive to both the static activation values of other units and their dynamic changes.
- The formalism supports recurrent networks.

Learning is done through gradient descent using a mean-squared error measure as with standard back-propagation learning.

A network is a collection of arbitrarily connected units. Time (denoted by t) is discrete, and units are updated synchronously. For each unit i, the following are defined:

$o_i^t \equiv$ output

$v_i^t \equiv$ velocity

$a_i^t \equiv$ activity

$b_i^t \equiv$ bias

$d_i \equiv$ decay

$\Delta_i^t \equiv$ damping factor

$s_i^t \equiv$ stimulation

For each connection from unit j to unit i, there are two weights:

$w_{ij}^o \equiv$ output weight to i from j

$w_{ij}^v \equiv$ velocity weight to i from j

At time $t = 0$, the velocity and activity of each unit is zero. The bias and decay of a unit are not time dependent. The remaining quantities are updated as follows:

$$o_i^t = \sigma(a_i^t + b_i) \tag{1}$$

$$v_i^t = o_i^t - o_i^{t-1} \tag{2}$$

$$a_i^t = a_i^{t-1} d_i + \Delta_i^t s_i^t \tag{3}$$

$$\Delta_i^t = 1 - \left(\frac{a_i^{t-1} d_i}{M}\right)^2 \tag{4}$$

$$s_i^t = \sum_j (o_j^{t-1} w_{ij}^o + v_j^{t-1} w_{ij}^v) \tag{5}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

Equations 1 and 2 define the values of a unit i at time t that are externally available via output connections. Equation 3 shows how a unit behaves as it is updated. The activity is the sum of two terms: the residual activation and the damped stimulation. The residual activation is the activity remaining at time t from the activity at time t-1. If the decay value is near one, most of the activity remaining is retained; if the decay is near zero, little of the activity is retained. The stimulation comes from a unit's input connections. The damping factor prevents activation values from getting very large. Equation 1 adds in the bias term for the unit—essentially the resting activation value.

The constant M is set large enough to allow activity to be pushed into the flat region of the sigmoid function (Equation 6), but not too far. This type of unit degenerates into a standard back-propagation unit if M is infinite, all velocity weights are zero, and the decay is zero. In this work, M was 20, the decay for learning units was 0.3, and the decay for input and buffering units was 1.

The standard derivation for back-propagation learning, with some minor modifications for the temporal factors, produces the weight update equations.

B Conference Registration Dialogs

This appendix lists the 12 conversations that form the corpus for the CMU/ATR Conference Registration Task.

Conversation 1

CALLER: Hello, is this the office for the conference?

OFFICE: Yes, that's right.

CALLER: I would like to register for the conference.

OFFICE: Do you already have a registration form?

CALLER: No, not yet.

OFFICE: I see. Then, I'll send you a registration form. Could you give me your name and address?

CALLER: The address is 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15236. The name is David Johnson.

OFFICE: I see. I'll send you a registration form immediately. If there are any questions, please ask me at any time.

CALLER: Thank you. Goodbye.

OFFICE: Goodbye.

Conversation 2

CALLER: Hello.

OFFICE: This is the office for the Conference.

CALLER: I would like to take part in the conference. What should I do?

OFFICE: First, you must register with a registration form. Do you already have a registration form?

CALLER: Not yet, please send me a form.

OFFICE: Then, could you give me your name and address?

CALLER: The address is 12 Grant Street, Pittsburgh, Pennsylvania, 15133. The name is Judy Simpson.

OFFICE: I see.

CALLER: Is an attendance fee required?

OFFICE: Yes, $200 per person is required as a registration fee.

CALLER: I see. Thank you very much.

OFFICE: Goodbye.

Conversation 3

CALLER: Hello, is this the conference office?

OFFICE: Yes, that's right. May I help you?

CALLER: I would like to attend the conference. How can I apply?

OFFICE: Please fill out a registration form. Do you have one?

CALLER: No, not yet.

OFFICE: OK, then I'll send you a registration form. Would you please give me your name and address?

CALLER: My address is 114 Beechwood Avenue, Squirrel Hill, Pennsylvania, 15213. My name is Harry Bovic.

OFFICE: Would you spell your last name please?

CALLER: It's B-O-V-I-C.

OFFICE: I've got it. I'll send you the form immediately.

CALLER: Thank you very much. Goodbye.

Conversation 4

OFFICE: Hello, this is the conference office.

CALLER: Could you give me some information about the application fee for the conference? How much will it cost if I apply for the conference right now?

OFFICE: Well, let's see. It costs $250 per person. But if you apply next month, it will cost you $325. The proceedings and the reception are included in the application fee.

CALLER: I am a member of the Information Processing Society. Is there a discount for members?

OFFICE: No, there is no discount this time.

CALLER: I understand. How can I pay?

OFFICE: Payment should be made by bank transfer. Please remit to our bank account which is mentioned in the announcement. The deadline is the end of the year.

CALLER: OK, thank you very much.

OFFICE: You're welcome. Please feel free to ask if there's anything you don't understand. Goodbye.

Conversation 5

OFFICE: Hello, conference office.

CALLER: I would like to contribute a paper to the conference. Would you please tell me the topic of the conference?

OFFICE: This conference covers a wide area of research related to Interpreting Telephony. We are also expecting linguists and psychologists as participants.

CALLER: Fine. By the way, what is the official language of the conference?

OFFICE: English and Japanese.

CALLER: I don't understand Japanese at all. Is there simultaneous interpretation into English when the presentation is made in Japanese?

OFFICE: Yes, we have simultaneous interpretation service into English.

CALLER: That would be helpful for me. Thank you very much. Goodbye.

Conversation 6

OFFICE: Conference office.

CALLER: I would like to know the details of the conference.

OFFICE: Do you have a conference announcement?

CALLER: No, I don't.

OFFICE: OK, the conference will take place from August 22nd to the 25th at the New York World Trade Center. The fee for participation is $500. If you intend to present a paper, please submit a summary by March 20th. I'll send the conference announcement to you today. Would you mind telling me your name and address?

CALLER: My name is Eric Thompson. My address is 1412 Smithfield Street, Pittsburgh, Pennsylvania, 15237.

OFFICE: Would you spell your last name for me?

CALLER: Sure, it's T-H-O-M-P-S-O-N.

OFFICE: OK. Could I have your phone number too?

CALLER: Yes. 372 8418.

OFFICE: 372 8148, is that correct?

CALLER: No, it's 8418.

OFFICE: 372 8418, right?

CALLER: Yes, it is. Thank you very much, goodbye.

Conversation 7

OFFICE: Hello, conference office.

CALLER: I wonder if you could help me. I sent in the registration form for the conference. But I can't attend the conference, so I would like to cancel.

OFFICE: Could you please give me your name?

CALLER: Yes, this is Dan Cooper from Bell Labs.

OFFICE: Mr. Cooper, you have already paid $400 for your registration fee, haven't you?

CALLER: Yes, I have. Is it possible for you to refund the registration fee?

OFFICE: I am sorry we can't. As noted in the announcement, cancellation after September 27th precludes a refund. We'll send you the programs and proceedings later.

CALLER: Will somebody else be able to attend instead of me, then?

OFFICE: Yes, that's all right. Please let me know in advance who is going to attend instead of you.

CALLER: Good, I'll let you know when it's decided. Goodbye.

Conversation 8

OFFICE: Hello, conference office.

CALLER: I've heard that you have a city tour during the conference. Can we still take part in it?

OFFICE: Yes, you can. We will visit Heinz Hall, Mount Washington, and the Mellon Museum on the afternoon of August 5th. Would you like to join us?

CALLER: How much does it cost?

OFFICE: $35, that includes dinner.

CALLER: Are the speakers also participating?

OFFICE: Some of them are supposed to.

CALLER: Then I would also like to go.

OFFICE: OK. Please give me your name and the number of people in your party.

CALLER: My name is Christopher Ohara. My wife will be coming too.

OFFICE: Would you spell your first name for me, Mr. Ohara?

CALLER: Sure, Christopher, C-H-R-I-S-T-O-P-H-E-R.

OFFICE: We'll meet in front of the reception desk. Please pay the tour fee there when you arrive.

CALLER: OK, thank you very much.

OFFICE: We'll be expecting you.

Conversation 9

OFFICE: Hello, conference office.

CALLER: I have a question about topics in the conference.

OFFICE: Yes, what is it?

CALLER: There is a topic called Machine Translation in the announcement. Specifically, what is it about?

OFFICE: I'm sorry. I'm really unable to answer any technical questions. The titles of papers to be presented at the conference are printed in the second version of the announcement. Would you please take a look at it?

CALLER: Yes, I will. Please mail me the announcement as soon as possible. My address is 34 Dayton Drive, Edison, New Jersey, 37814. My name is John Mathis.

OFFICE: 34 Dayton Drive, Edison, New Jersey, 37814, John Mathis, correct?

CALLER: Yes.

OFFICE: Would you spell your last name for me please?

CALLER: Sure, it's M-A-T-H-I-S.

OFFICE: I'll send one as soon as possible. Is there anything else I can help you with?

CALLER: No, that's all thanks. Goodbye.

Conversation 10

OFFICE: Conference office.

CALLER: Can I ask you a few questions? I would like to contribute a paper to the conference. How can I apply?

OFFICE: First, you should send us a 200 word summary by March 20th. The summary will be reviewed here and we will send you a reply by May 20th. If your paper is accepted, we'll also enclose special forms for your paper. Please send them back to us by June 30th.

CALLER: Fine, what kind of form do I have to write the summary on?

OFFICE: We have a special form for the summary. Please fill it in. Then, we'll send you the application form. May I have your name and address please?

CALLER: All right, my name is George VanParis from AI Labs. My address is 34 Park Avenue, New York, New York, 23415.

OFFICE: Mr. George VanParis from AI Labs, right? Your address is 34 Park Avenue, New York, New York, 23415. Is that correct?

CALLER: Yes, it is. Please send me an application form.

OFFICE: Sure, I'll send it to you immediately. Goodbye.

Conversation 11

CALLER: Is this the conference office?

OFFICE: Yes, this is the conference office. May I help you?

CALLER: Please tell me how to get to the conference site. I'm at Station Square now.

OFFICE: Please take the subway to the U.S. Steel building, downtown. From there there is a bus to the conference center. Of course, you'll also be able to take a taxi from the downtown area.

CALLER: How much is it from Station Square to the conference center by taxi?

OFFICE: From Station Square it will cost you about $12.

CALLER: And how much does it cost from downtown?

OFFICE: From downtown, it will cost you approximately $5.

CALLER: OK, thank you very much.

OFFICE: Not at all. You're welcome.

Conversation 12

CALLER: Hello.

OFFICE: Hello, this is the conference office.

CALLER: I would like to ask you about hotel accommodations for the conference. Do you have a service that can help me find a place to stay?

OFFICE: Yes, we do. The hotels we can help you with are the Hilton Hotel and Crystal Hotel. A single room will cost you from $80 to $110 per night. A twin room ranges from $95 to $150 per night.

CALLER: Fine, which hotel is closer to the conference center?

OFFICE: I'm sorry, what did you say?

CALLER: I said "Which hotel is closer to the conference center?"

OFFICE: Oh, the Hilton Hotel is closer to the conference center.

CALLER: Then I would like to make a reservation for the Hilton Hotel. Can I leave the hotel reservation to you?

OFFICE: Sure. We'll be able to reserve rooms for you at either the Hilton Hotel or the Crystal Hotel.

CALLER: That's fine. Well, could you reserve an $80 single room at the Hilton Hotel?

OFFICE: OK. An $80 single room at the Hilton Hotel. Right?

CALLER: Yes. That's right.

OFFICE: When will you check in?

CALLER: The evening of August 4th. Checking out the morning of the 8th.

OFFICE: OK, please wait a moment. I am going to check to see if there is a vacancy. Yes, there is. Please give me your name and address.

CALLER: My name is Joe Bradshaw. The address is 54 8th Avenue, Pittsburgh, Pennsylvania, 15238.

OFFICE: Would you spell your last name please?

CALLER: It's B-R-A-D-S-H-A-W.

OFFICE: And your phone number please?

CALLER: My phone number is 331 2521.

OFFICE: OK. I've reserved a single room at the Hilton Hotel from August 4th to the 8th.

CALLER: Thanks very much. Goodbye.

C Conference Registration Testing Sets

This appendix contains the two generalization test sets. The sentences are distinct from the 12 conversations. These sentences were generated by people not associated with the development of the connectionist parser. Performance for the best PARSEC network (CR4) and the best hand-written LR grammar (Grammar 3) are shown in the two columns at right. For CR4 (first column), the markings are: GOOD, CLOSE, and BAD. For LR3 (second column), the markings are: GOOD, CLOSE, BAD, and NIL (to distinguish NIL parses from incorrect non-NIL parses).

Test Sentences	CR4	LR3
(please send me the summary immediately)	GOOD	GOOD
(you should send it immediately)	CLOSE	GOOD
(i will arrive on the twenty second)	BAD	GOOD
(is there a discount for members of the information processing society)	GOOD	GOOD
(by the way what is the official language)	CLOSE	GOOD
(how much will it cost)	GOOD	GOOD
(if i apply now will you send me a form immediately)	BAD	NIL
(i like to register)	GOOD	CLOSE
(please register me for the conference)	BAD	NIL
(who are the linguists)	GOOD	GOOD
(you must submit a twenty word summary by august)	GOOD	NIL
(when should i send the summary)	GOOD	GOOD
(could you tell me the deadline for registration)	GOOD	NIL
(can i do that)	GOOD	NIL
(please give me your name)	GOOD	GOOD

	CR4	LR3
(i am not sure)	BAD	NIL
(please fill out the form which i will send you)	BAD	NIL
(i don+t like registration forms)	GOOD	NIL
(give me your name please)	CLOSE	GOOD
(how much is the registration fee)	GOOD	GOOD
(what is your address)	GOOD	GOOD
(please fill out the form i send you)	BAD	BAD
(please fill the form that i will send you)	CLOSE	NIL
(can you mail me an announcement)	GOOD	GOOD
(can i join the city tour)	GOOD	GOOD
(what papers will be presented on august twenty second)	BAD	NIL
(can you make a reservation for me)	GOOD	GOOD
(i don+t understand)	GOOD	BAD
(how much does the tour cost)	GOOD	NIL
(go to the registration desk to meet your party)	GOOD	NIL
(i would like a registration form for the conference)	GOOD	NIL
(i would like a hotel room)	GOOD	NIL
(how can i get the proceedings)	GOOD	NIL
(i+m expecting to attend the ai conference but i don+t have a registration form)	BAD	NIL
(could you please send me one)	GOOD	GOOD
(i would like hotel information)	GOOD	NIL
(do you have the name of hotels)	GOOD	GOOD
(what is a possible place to stay)	CLOSE	CLOSE
(are some rooms free)	GOOD	NIL
(when is the first registration deadline)	GOOD	NIL
(have i made it)	BAD	GOOD
(how much do i have to pay if i register later)	BAD	NIL
(are there any accommodations at this conference)	CLOSE	NIL
(how much is it)	GOOD	GOOD
(can i register at the desk)	GOOD	NIL
(how can i get the proceedings if i don+t register for it)	GOOD	NIL
(how much do the proceedings cost)	GOOD	GOOD
(can you send me a conference announcement)	GOOD	GOOD
(when is harry bovic participating)	CLOSE	NIL
(do speakers get free registration)	CLOSE	NIL
(hello my name is harry bovic)	GOOD	GOOD
(i would like to register for the ai conference)	GOOD	NIL
(i will send you an application)	GOOD	NIL
(what is your name please)	GOOD	NIL
(my name is harry bovic b o v i c)	GOOD	NIL
(i also noted your address and phone number)	GOOD	NIL
(my address is five four four two grant street pittsburgh pennsylvania one five two three two)	GOOD	NIL
(the phone number is five five five five five five five)	GOOD	GOOD

	CR4	LR3
(do you have any technical questions about the conference)	GOOD	BAD
(yes i do)	GOOD	GOOD
(are you able to help me make a reservation at a hotel)	CLOSE	NIL
(yes i am)	GOOD	GOOD
(there are two hotels the crystal and the hilton)	BAD	NIL
(i would like a single room in the hotel closer to the conference)	GOOD	NIL
(fine that would be the hilton)	BAD	NIL
(that is okay)	GOOD	BAD
(can i give you any information about the conference)	GOOD	BAD
(i would like a discount on the proceedings)	GOOD	NIL
(that will be included in the papers with your application)	BAD	NIL
(what are the possible forms of payment)	BAD	BAD
(the correct form of payment for the conference is bank transfer)	BAD	BAD
(i am also participating in the city tour)	GOOD	GOOD
(you can register for that tour on the first afternoon of the conference)	GOOD	NIL
(tour registration will be accepted during the first afternoon of the conference)	GOOD	NIL
(is there subway service at the hotel)	GOOD	NIL
(will any speakers be participating in the tour of the city)	BAD	NIL
(what is the conference fee)	GOOD	NIL
(i have already paid my fee for the conference)	GOOD	GOOD
(i can+t attend)	GOOD	GOOD
(could my wife attend the conference in my place)	GOOD	GOOD
(my wife is mentioned in the announcement)	BAD	GOOD
(may i get a summary of the topics)	GOOD	NIL
(i would like to submit a paper to the conference)	GOOD	GOOD
(how do i go about this)	BAD	GOOD
(i will send you the form to fill out)	CLOSE	GOOD
(you should have included a three hundred word summary)	GOOD	NIL
(which is that)	GOOD	GOOD
(we must get your summary by may twentieth)	BAD	NIL
(no it is closer)	GOOD	GOOD
(can you make the hotel reservation)	GOOD	GOOD
(i am a member of the ai society)	GOOD	NIL
(will i get any discount)	GOOD	NIL
(yes there is a discount for members)	GOOD	GOOD
(is this the office for conference registration)	GOOD	NIL
(i will take your name and address and send you the correct forms)	CLOSE	NIL

	CR4	LR3
(what is your phone number please)	GOOD	NIL
(yes how may i help you)	BAD	GOOD
(would you like to register for the conference)	CLOSE	NIL
(your summary must arrive soon)	GOOD	NIL
(that hotel is closer to the conference site)	GOOD	NIL
(the hotel mentioned a discount for conference members)	BAD	NIL
(titles of papers will be included in the information forms)	GOOD	NIL
(topics to be presented will be included in the forms)	BAD	GOOD
(how do you spell that)	BAD	GOOD
(what is that)	GOOD	GOOD
(i said may twentieth)	GOOD	NIL
(the name of the hotel is the hilton)	GOOD	NIL
(pay the fee by bank transfer)	BAD	NIL
(the fee is three hundred fifty dollars)	GOOD	GOOD
(yes that includes the reception)	GOOD	GOOD
(no the fee does not pay the city tour)	GOOD	NIL
(that will be made later)	GOOD	GOOD
(the fee is five hundred dollars after september thirtieth)	CLOSE	BAD
(i will send you the application today)	GOOD	NIL
(feel free to ask any special questions)	BAD	NIL
(i will be able to answer any questions)	GOOD	NIL
(send your application on september thirtieth)	BAD	NIL

Testing Sentences (Final Set)	CR4	LR3
(i do not understand you)	GOOD	NIL
(then i would like to have some information about the conference)	CLOSE	BAD
(how can i help you)	GOOD	GOOD
(what is the fee for the registration)	GOOD	NIL
(the registration fee is three hundred fifty dollars per person)	GOOD	NIL
(can you send me a registration form)	GOOD	GOOD
(my name is heinz thompson)	GOOD	NIL
(could you spell your last name please)	GOOD	GOOD
(i will send you a form immediately)	GOOD	GOOD
(thank you goodbye)	GOOD	GOOD
(office for the conference can i help you)	GOOD	NIL
(i+ve sent a paper but haven+t heard from you)	BAD	NIL
(could you please wait)	GOOD	GOOD
(when have you sent your paper)	BAD	NIL
(oh i see)	BAD	GOOD
(that+s free)	GOOD	NIL
(do you have any questions)	GOOD	BAD
(what will you do now with the paper i sent you)	BAD	NIL
(i will send your paper right back as soon as i see it)	BAD	NIL
(am i telling to the office for the conference registration)	BAD	NIL
(yes you are)	GOOD	GOOD
(how can i help you)	GOOD	GOOD
(what do i have to do)	BAD	NIL
(you should fill out a conference registration form)	GOOD	NIL
(can+t we do that right now on the phone)	BAD	NIL
(i am sorry we can+t)	GOOD	GOOD
(how can i get a conference application form)	GOOD	NIL
(i am sure you can send me one)	GOOD	NIL
(yes of course)	GOOD	NIL
(if you give me your address)	GOOD	NIL
(i+ll send you a form and some information about the conference)	GOOD	BAD
(is there anything else i can do for you)	BAD	NIL
(so i would like you to spell very specifically)	BAD	NIL
(i would like to register for the ai conference)	GOOD	NIL
(could you say this please)	BAD	GOOD
(could you please say it)	GOOD	GOOD
(is there a special discount for the participants of the conference)	GOOD	GOOD
(yes it+s ninety dollars)	GOOD	GOOD
(you have to register soon)	CLOSE	NIL

	CR4	LR3
(there is a deadline)	GOOD	GOOD
(when is it)	GOOD	GOOD
(how can i send you the registration fee)	GOOD	GOOD
(fine could you also reserve a room for me)	GOOD	GOOD
(do you have the number)	GOOD	GOOD
(yes i do)	GOOD	GOOD
(i am included now)	GOOD	NIL
(yes you are)	GOOD	GOOD
(hello can you give me some information about the ai conference)	CLOSE	NIL
(is it still possible to get a hotel room)	CLOSE	NIL
(please wait)	GOOD	GOOD
(can you reserve a hotel room for me)	GOOD	NIL
(i can make a reservation for you if you give me your number)	BAD	NIL
(i would like to register for the conference but i don+t know how much the fee is)	BAD	NIL
(it+s two hundred dollars per person if you register immediately)	BAD	NIL
(is there any participation discount)	GOOD	NIL
(yes for linguists it+s two hundred dollars per person)	GOOD	NIL
(do you have a registration form)	GOOD	GOOD
(if you could give me your name and address then i+ll send you one immediately)	GOOD	NIL
(i+ll spell it for you)	GOOD	GOOD
(it+s t i l o s l o b o d a)	CLOSE	GOOD
(sorry could you please spell the name too)	CLOSE	NIL
(do we have two registration forms)	GOOD	NIL
(can you give me an address please)	CLOSE	GOOD
(then i+ll send you the two forms immediately)	GOOD	GOOD
(could you send us some information about rooms too)	GOOD	GOOD
(yes i+ll enclose some hotel information)	GOOD	NIL
(are you a member of the ai society)	GOOD	NIL
(can you spell your last name for me)	GOOD	GOOD
(yes it+s t e b e l s k i s)	GOOD	GOOD
(oh i+m sorry)	GOOD	GOOD
(is that right)	GOOD	GOOD
(what kind of a name is that)	CLOSE	GOOD
(yes i know)	GOOD	GOOD
(let+s see here)	GOOD	GOOD
(there will be a registration fee of forty dollars)	BAD	NIL
(what kind of payment do you have)	CLOSE	NIL
(can you give me the number)	GOOD	GOOD
(that is eight five seven three four two nine four)	GOOD	BAD
(you+re all right)	GOOD	NIL

	CR4	LR3
(you should get it in about a month)	GOOD	NIL
(which do you say would be the last)	BAD	NIL
(are you able to make a reservation there for me)	BAD	NIL
(no but i can give you the number)	GOOD	GOOD
(it+s eight four six eight two two four)	GOOD	GOOD
(the hilton is at eight three two four two nine eight)	GOOD	NIL
(is there anything else)	GOOD	GOOD
(no i don+t know)	GOOD	BAD
(are you a member of the ai society)	GOOD	NIL
(we+ll send you a registration form as well if you+re not)	BAD	NIL
(no thanks)	GOOD	GOOD
(i+m already a member)	GOOD	NIL
(send us a check for three hundred dollars in the next month)	BAD	NIL
(but i did not)	GOOD	GOOD
(okay)	GOOD	GOOD
(how should we send the information)	CLOSE	NIL
(we+ll get that right out to you)	BAD	NIL
(is there anything else you would like)	BAD	BAD
(can you tell me which hotel is closer to the conference site)	BAD	GOOD
(do you still have rooms)	GOOD	GOOD
(thanks i+ll tell them right now)	GOOD	NIL
(hello i would like to register to the ai conference)	BAD	NIL
(how do you spell mellon)	BAD	NIL
(have a good one)	GOOD	GOOD
(thanks)	GOOD	GOOD
(may i help you)	GOOD	NIL
(when is the deadline for a conference registration please)	CLOSE	NIL
(are you a member of the ai society mister)	GOOD	NIL
(i am a member of the information processing society of the bank)	GOOD	GOOD
(let+s see)	GOOD	GOOD
(the registration fee is three hundred dollars)	GOOD	NIL
(you get fifty dollars if you register immediately)	GOOD	GOOD
(i would like to register)	BAD	BAD
(please tell me how can i pay)	BAD	NIL
(will you be able to mail it if i send it by bus this afternoon)	GOOD	BAD
(could you tell me your name and address please)	GOOD	GOOD
(my name is john johnson)	GOOD	GOOD
(no)	GOOD	NIL
(my correct address is one two three four forbes avenue)	GOOD	GOOD

	CR4	LR3
(is this correct)	GOOD	GOOD
(no thank you very much)	GOOD	NIL
(i would like to register for ai ninety two)	GOOD	GOOD
(all right what is your name)	GOOD	GOOD
(okay your registration fee is five hundred twelve dollars)	GOOD	NIL
(okay what is your payment number)	GOOD	GOOD
(yes i would like to register for the conference)	GOOD	GOOD
(can you spell that)	GOOD	NIL
(ai conference office may i help you)	BAD	NIL
(all of the details will be in the forms we send you)	CLOSE	NIL
(hello i would like to get some information on the conference)	GOOD	EEC
(is this the right number for it)	BAD	NIL
(okay first i would like to know the deadline for the registration so that i can get the discount registration fee)	GOOD	NIL
(sure the deadline is may thirty first)	BAD	GOOD
(by the way what is the conference site)	GOOD	NIL
(the conference site is the heinz center)	GOOD	NIL
(oh is it closer to the museum)	BAD	NIL
(is there any interpreting in the conference)	BAD	NIL
(oh really that+s too good)	GOOD	GOOD
(is there an office of the conference)	GOOD	GOOD
(i would like to attend for the conference)	BAD	GOOD
(are you a member)	GOOD	GOOD
(what is your name)	BAD	NIL
(i+m not sure)	GOOD	NIL
(i said it+s one five two zero six)	GOOD	NIL
(that+s the conference fee)	GOOD	NIL
(you are welcome)	BAD	NIL
(how much will be in your party)	BAD	NIL
(we have rooms at the crystal and the hilton)	BAD	NIL
(which would you like)	GOOD	NIL
(no thank you)	GOOD	GOOD
(would you like to pay now or be called)	BAD	NIL
(please tell me)	GOOD	GOOD
(you will be called with information about the conference)	CLOSE	NIL
(that is a fee of four hundred fifty dollars)	GOOD	GOOD
(that+s fine then)	BAD	NIL
(could you send me information on the hotels in the area)	GOOD	GOOD
(would you like to make accommodations there)	BAD	NIL
(yes that would be fine)	GOOD	GOOD

	CR4	LR3
(i would like to get five rooms with four people to a room)	BAD	NIL
(all we have now is your name address and phone number)	BAD	NIL
(that+s five four people rooms at the hilton that will be reserved with your name)	BAD	NIL
(the number is four one two two six eight three nine seven)	GOOD	BAD
(if you have any questions please feel free to ask)	GOOD	GOOD
(could you give me your fee)	GOOD	GOOD
(can i get there by taxi)	BAD	NIL
(how do you like to be paid)	BAD	NIL
(can i make a reservation for a room for september seventh)	GOOD	NIL
(yes can i have your name please)	CLOSE	GOOD
(thank you i will)	BAD	BAD
(hello hilton hotel)	GOOD	NIL
(how may i help you)	GOOD	GOOD
(i+m expecting a conference today)	GOOD	GOOD
(do you have anything that would be possible)	BAD	BAD
(is there a desk in the hotel)	GOOD	NIL
(your rooms are reserved)	GOOD	NIL
(could i have your phone number please)	GOOD	NIL
(no that+s fine)	GOOD	GOOD
(i would like some information from you)	BAD	NIL
(my number is seven one eight five five five one five four three)	GOOD	BAD
(today i can be called at seven one eight five five five seven six five five)	GOOD	NIL
(thanks for your help)	BAD	NIL

D ATIS Sentences: Novice User Test

This appendix contains the training and testing sentences that were used by the non-expert for the ATIS task.

Training Sentences

(what airline is c o)
(show me all the nonstop flights from denver to san francisco leaving about three o+clock in the afternoon)
(show me the distance from the denver airport to downtown)
(show me the nonstop flights from san francisco to d f w before noon)
(show me the airfares on flights from d f w to denver before nine a m)
(what is restriction a p slash eighty)
(what is restriction v u slash one)
(what is class y)
(what is the fare on american airlines fourteen forty three flight)
(what type of aircraft is flying united airlines flight nine fifty three)
(show me flight nine fifty three+s arrival time and what type of meal it has)
(show me which airline flight leaves from denver to san francisco at eighteen ten)
(show me the flight that leaves san francisco for d f w at nine a m)
(book reservations for five from dallas to baltimore on may twelfth at two hundred and eighty eight dollars one-way)
(book reservations for five from dallas to baltimore on flight three fourteen on may twelfth)
(purchase tickets for five from dallas to baltimore on flight three fourteen on may twelfth)
(make reservations)
(show me all the flights from baltimore to philadelphia on may nineteenth)

(show me the fares)
(show me all the flights from baltimore to philadelphia on may twenty sixth)
(what does f a mean)
(show me all the flights and their fares from san francisco to boston on june second)
(show me all the flights from boston to d f w on june ninth)
(show me their fares)
(show me all the flights from boston to s f o on june second)
(show me all the flights from s f o to d f w on june nine)
(what do the transport abbreviations mean)
(please give me a list of flights from dallas to boston leaving on saturday mornings before noon)
(i need flight times from boston to dallas leaving on sunday afternoon after three o+clock)
(what is a y class and what does the d l under f a column mean)
(what does the d l under the column f a mean)
(are there any advance purchase fares from dallas to boston for round-trip tickets)
(give me a list of all flights from dallas to boston that only have one stop between dallas and boston)
(give me all flights from dallas to boston)
(give me a list of all airfares for round-trip tickets from dallas to boston flying on american airlines)
(show me a list of all flights from dallas to philadelphia)
(give me a list for all round-trip flights flying from dallas to philadelphia on american airlines)
(give me a list for all round-trip flights flying from dallas to san francisco on american airlines)
(give me a list of all flights from dallas to san francisco)
(let me see all the information from dallas fort worth to atlanta)
(what does a l mean)
(show me flights from dallas to atlanta)
(what does l h mean)
(show me the different flights)
(list the days and its meanings)
(show me a list of codes for the meals)
(what does e q p stand for)
(does american have any specials)
(what does restriction v u slash one mean)
(how much does the flight from dallas fort worth to atlanta round-trip cost)
(what does restriction a p slash eighty mean)
(what does class y n mean)
(do you have to take a y n flight only at night)
(list all information about flights from dallas fort worth to atlanta)
(how much does flight number eighty three cost one-way)
(show me the meanings of the classes again)
(show me the fares for each type of ground transportation in atlanta)
(show me all flight information from atlanta to san francisco)
(what+s the price of a one-way ticket on flight number one thirty seven)
(what does restriction a p slash sixty eight mean)
(show me the restrictions on flight number one thirty seven)
(show me ground transportation in san francisco)

(what type of aircraft is used on flight number one thirty seven)
(show me all information about aircraft type lockheed l one zero one one)
(show me all information about aircraft type boeing seven six seven)
(show me all the nonstop flights from boston to atlanta)
(give me a general description of flight number five four seven)
(what type of aircraft is flight number five four seven)
(what is the coach fare for one-way flight on number five four seven)
(what is restriction v u on flight number five four seven)
(what types of ground transportations services are available from the airport in atlanta to downtown atlanta)
(what is transport l)
(what is transport r)

Testing Sentences

(show me all the nonstop flights from atlanta to dallas)
(what type of aircraft is flight four four seven)
(what is flight code one zero two one four seven)
(what is the flight day of flight number five four seven)
(what type of ground transportation is available from d f w airport to downtown dallas)
(what is the one-way coach fare on flight number four four seven)
(what is the one-way first class fare on flight number four four seven)
(show me all the nonstop flights from dallas to san francisco)
(what type of aircraft is flight number four five nine)
(what ground transportation is available from the san francisco airport to downtown san francisco)
(what is the name of the airport in san francisco)
(what is the name of the airport in atlanta)
(show me all the nonstop flights from dallas to denver early in the morning)
(show me all the flights from denver to san francisco between two p m and seven p m)
(show me the distance from san francisco airport to downtown)
(what is the fare on flight eleven forty nine from continental airlines)
(what is the fare on united airlines nine fifty three)
(show me the airfare on flight eight eight zero for united airlines)
(show me the airfares on u a+s flight five one one)
(show me all the flights from dallas to baltimore on may twelfth)
(show me the prices for flights from dallas to baltimore on may twelfth)
(show me the price for flight three fourteen on may twelfth from dallas to baltimore)
(show me all the flights from philadelphia to boston on may twenty sixth)
(what airline is m l)
(show me the transportation from s f o to downtown san francisco)
(i need information on airlines servicing boston flying from dallas)
(is there more than one airport in the boston area that american and delta service)
(are there any excursion fares for round-trip tickets from dallas to boston)
(give me all nonstop flights from dallas to boston)
(describe each of the different classes for airfares)
(show me information that will take me from dallas to atlanta)
(what does d l mean)
(on the days is a one equal to sunday)

(does american airlines have any special fares)
(show me time tables of fare code seven one zero zero two six two)
(show me flight numbers of american from dallas fort worth to atlanta)
(show me ground transportation types in atlanta)
(show me classes for flight number one thirty seven and restrictions and what they mean)
(how much is flight number one thirty seven with a class y)
(is five hundred and fifty two dollars the cheapest fare from san francisco to dallas fort worth)
(show me capacity seatings for the boeing seven sixty seven)
(what is the capacity of flight number five four seven)
(flight five four seven is part of what airline)
(what is restriction a p fifty seven on flight number five four seven)
(under the category ground transportation what is transport a)
(what is the capacity of flight four four seven)
(what types of meals are available on flight number four four seven)
(what types of meals are available on flight number five four seven)
(what is the capacity of flight number four five nine)
(what is the one-way coach fare for flight number four five nine)
(what ground transportation is available from the airport in boston to downtown boston)
(what is the name of the airport in boston)

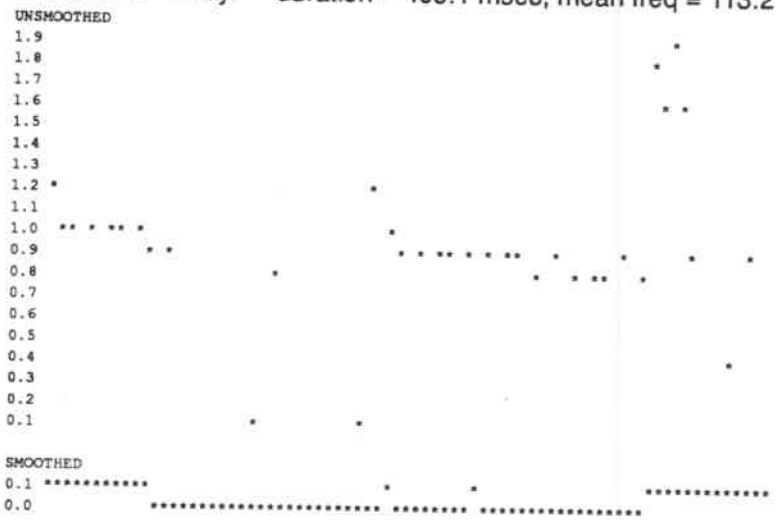
E Pitch Experiment Data

This appendix contains some of the data that was used for the experiments discussed in Chapter 8. Note that the plots tend to accentuate small differences in pitch values since they are quite coarse. A single example of 6 utterances, pronounced as statements and as questions, for two speakers (one male and one female) are included.

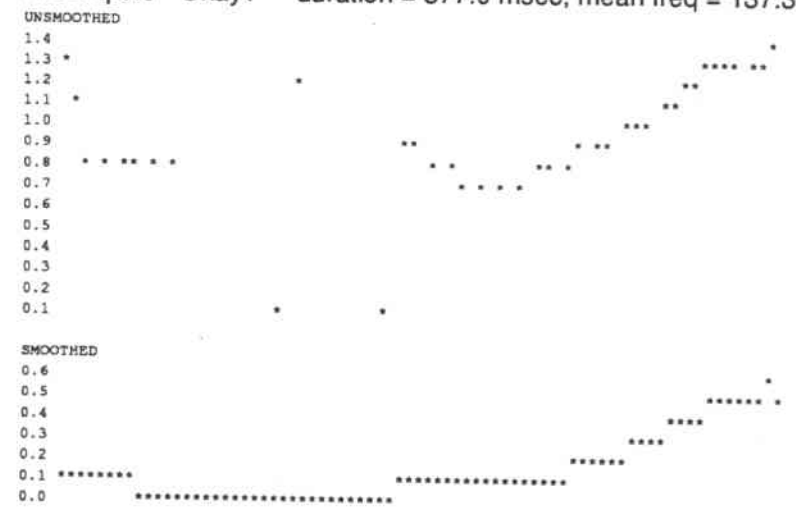
Speaker 1

These contours are from speaker ANJ (male).

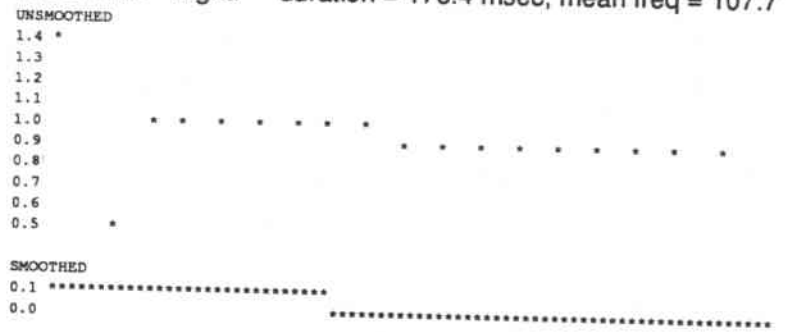
FILE: s.0.0 "Okay." duration = 409.1 msec, mean freq = 113.2



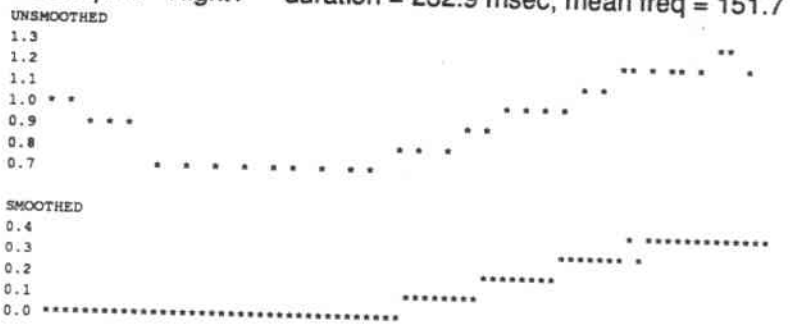
FILE: q.0.0 "Okay?" duration = 377.0 msec, mean freq = 137.3



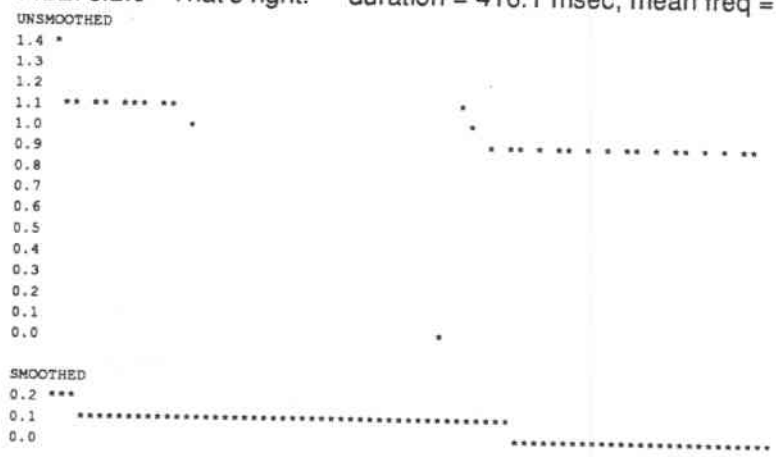
FILE: s.1.0 "Right." duration = 176.4 msec, mean freq = 107.7



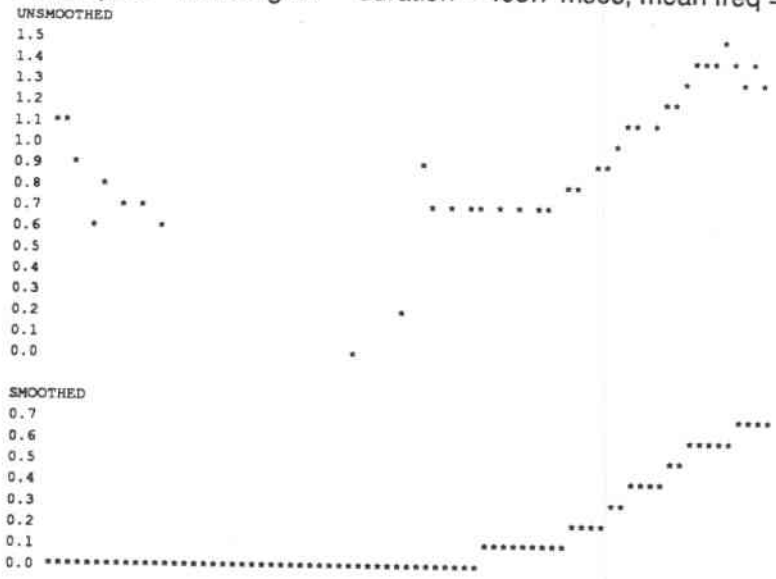
FILE: q.1.0 "Right?" duration = 232.9 msec, mean freq = 151.7



FILE: s.2.0 "That's right." duration = 416.1 msec, mean freq = 111.0

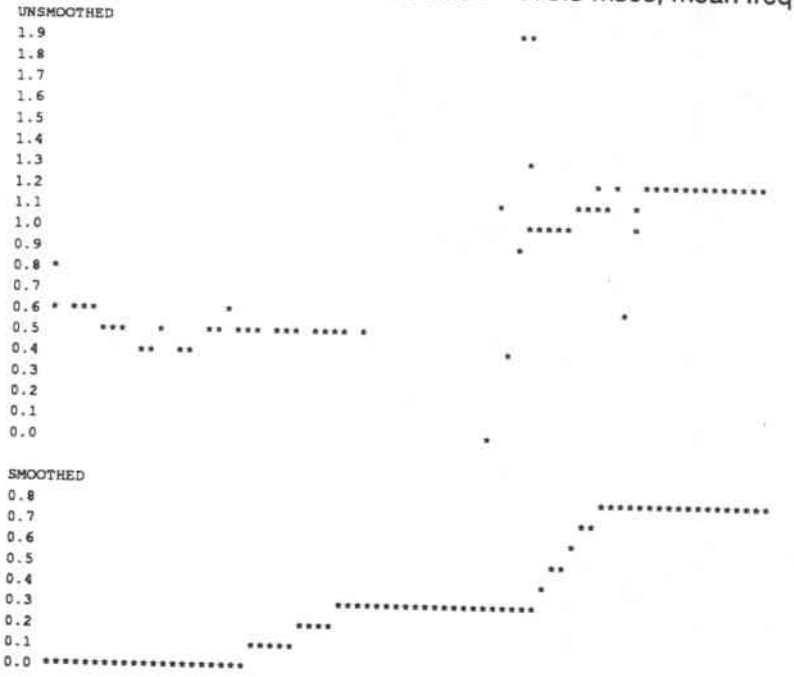


FILE: q.2.0 "That's right?" duration = 405.7 msec, mean freq = 142.2

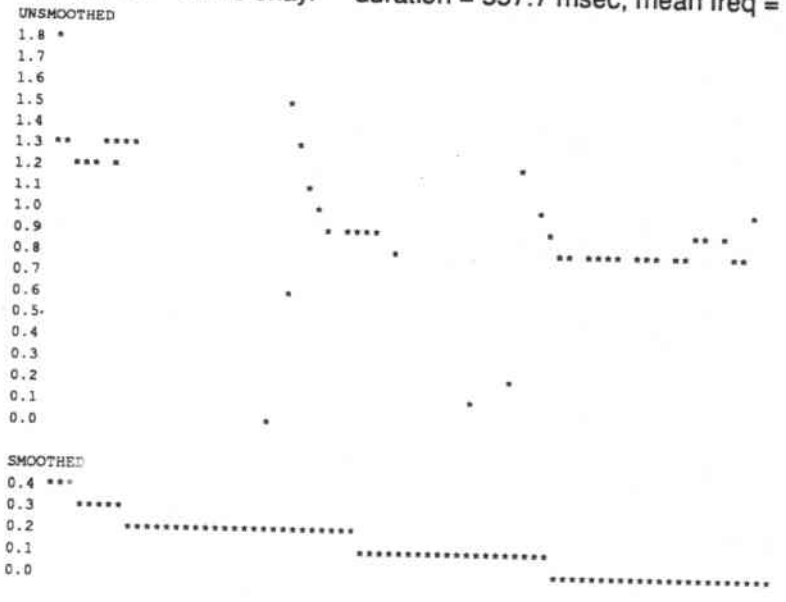


FILE: s.3.0 "You have one." duration = 653.6 msec, mean freq = 123.3

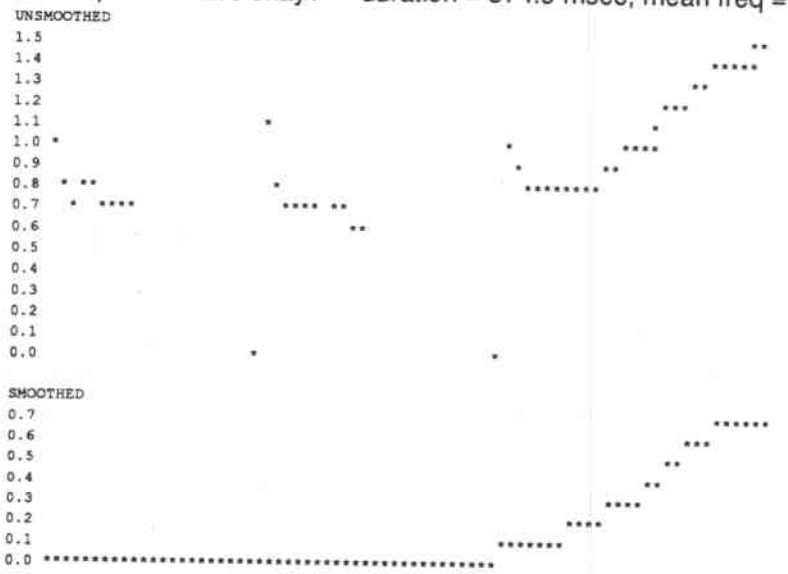
FILE: q.3.0 "You have one?" duration = 575.9 msec, mean freq = 196.0



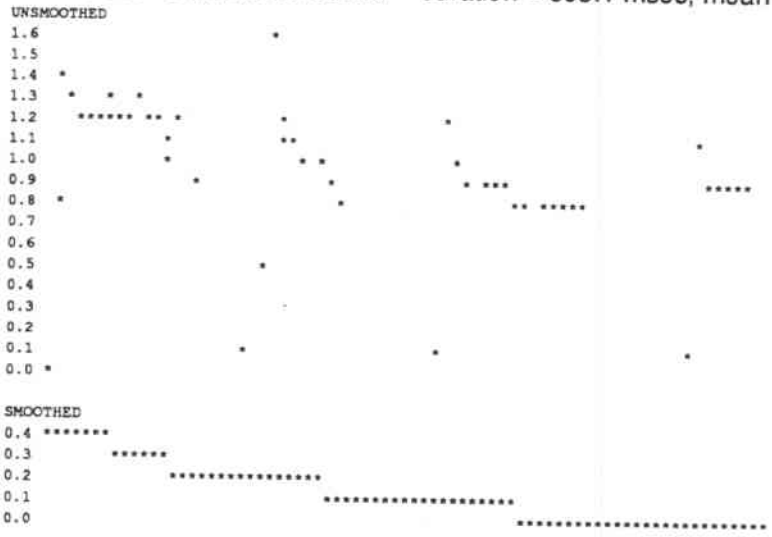
FILE: s.4.0 "That's okay." duration = 557.7 msec, mean freq = 116.9



FILE: q.4.0 "That's okay?" duration = 574.9 msec, mean freq = 149.6



FILE: s.5.0 "Conference office." duration = 603.1 msec, mean freq = 114.5

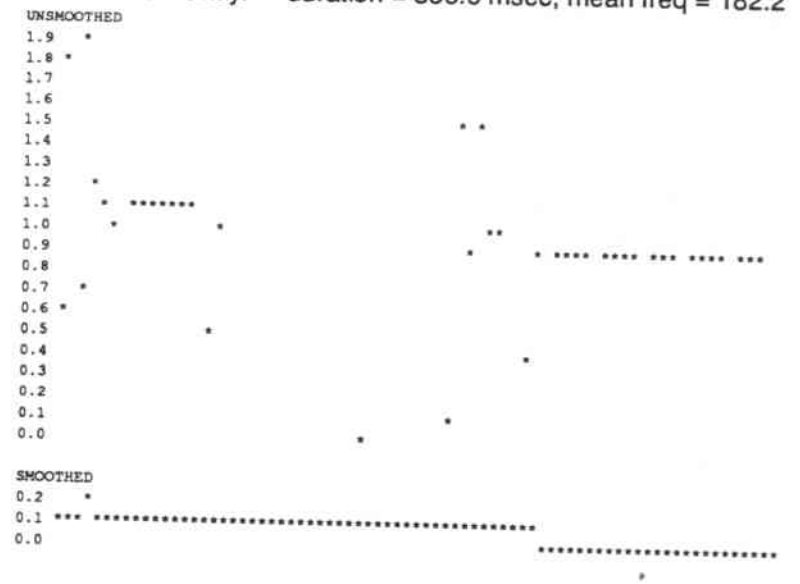


FILE: q.5.0 "Conference office?" duration = 634.6 msec, mean freq = 183.0

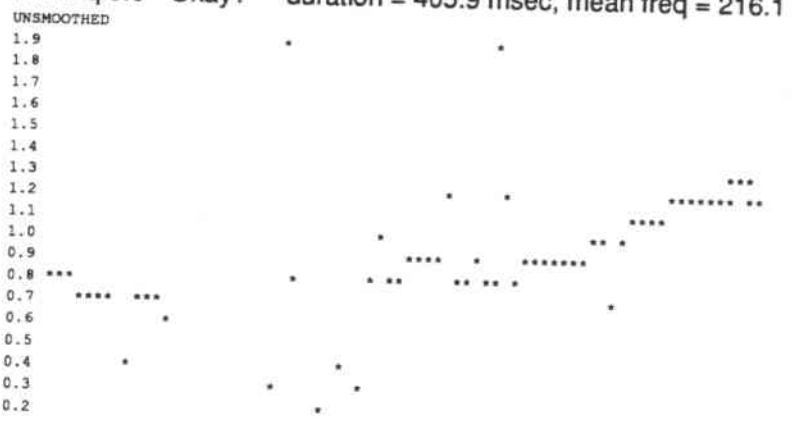
Speaker 2

These contours are from speaker TMJ (female). Note the higher mean frequencies for each corresponding contour. However, the normalized contours are quite similar.

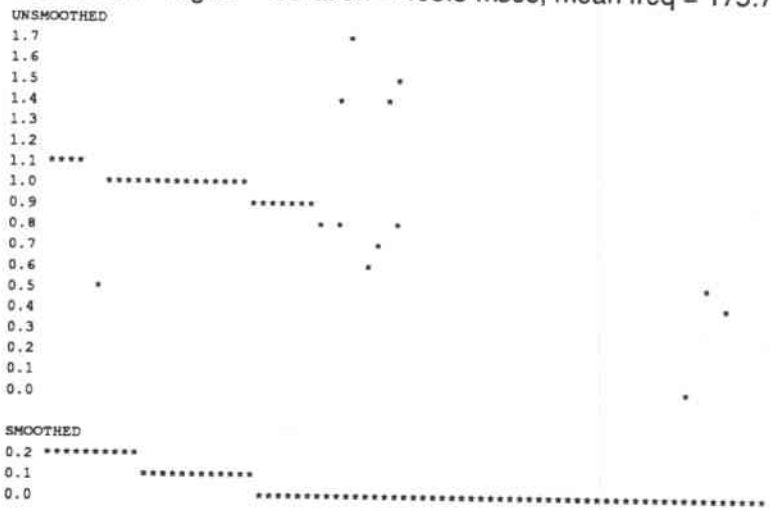
FILE: s.0.0 "Okay." duration = 336.6 msec, mean freq = 182.2



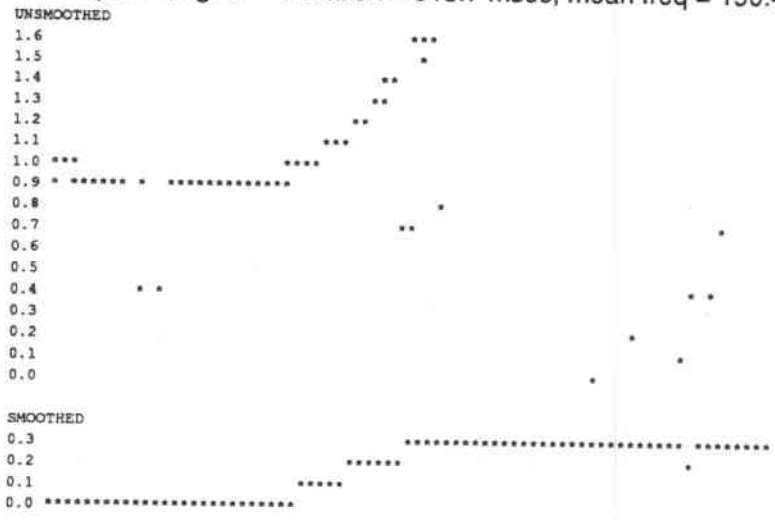
FILE: q.0.0 "Okay?" duration = 405.9 msec, mean freq = 216.1



FILE: s.1.0 "Right." duration = 408.8 msec, mean freq = 175.7

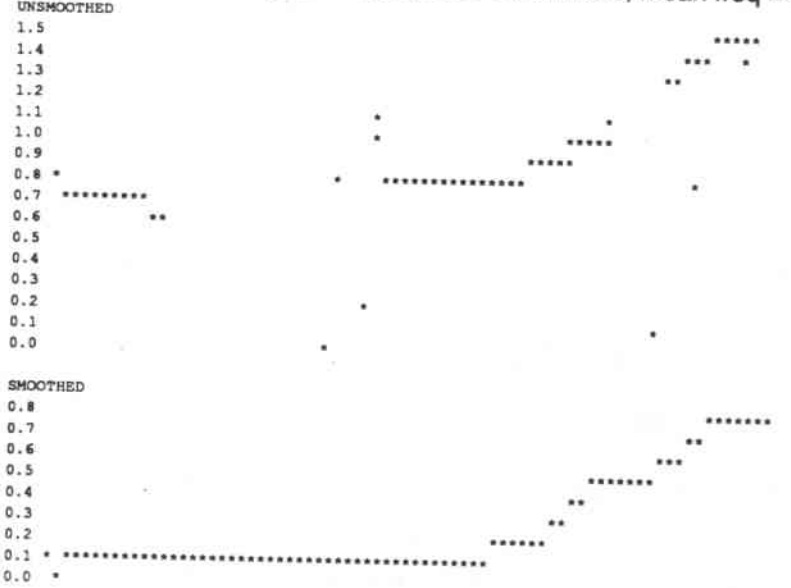


FILE: q.1.0 "Right?" duration = 513.7 msec, mean freq = 190.4

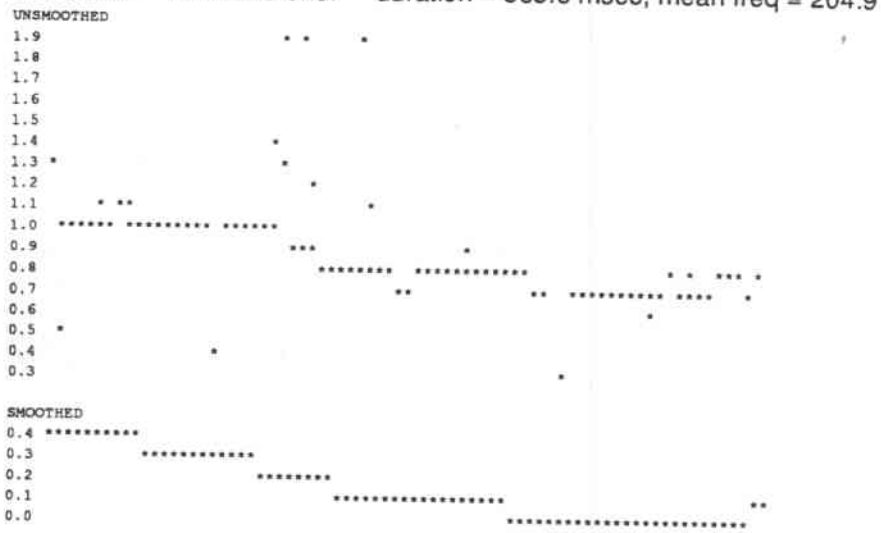


FILE: s.2.0 "That's right." duration = 489.7 msec, mean freq = 183.1

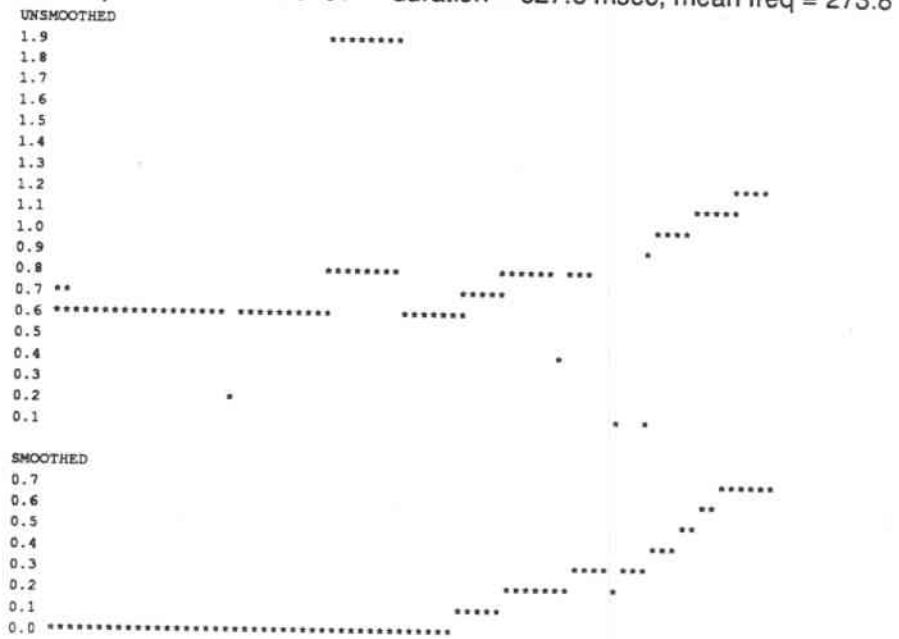
FILE: q.2.0 "That's right?" duration = 488.1 msec, mean freq = 224.7



FILE: s.3.0 "You have one." duration = 565.6 msec, mean freq = 204.9

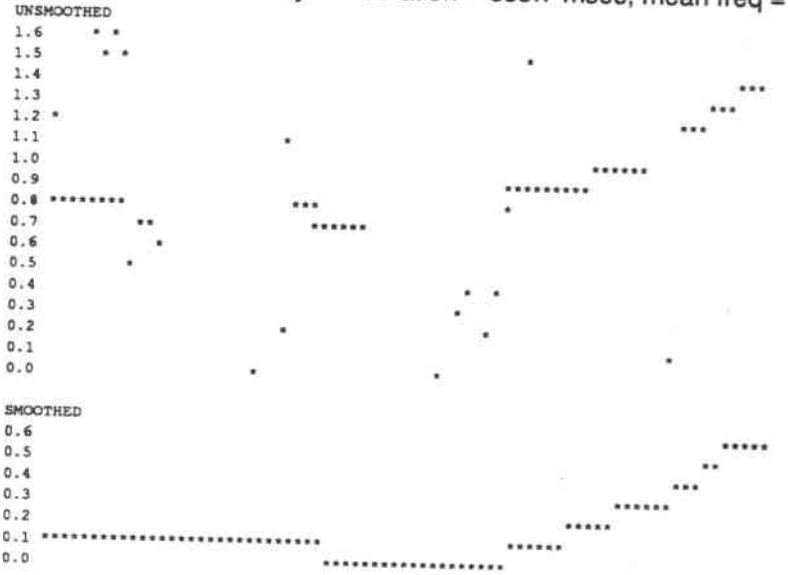


FILE: q.3.0 "You have one?" duration = 627.6 msec, mean freq = 273.8

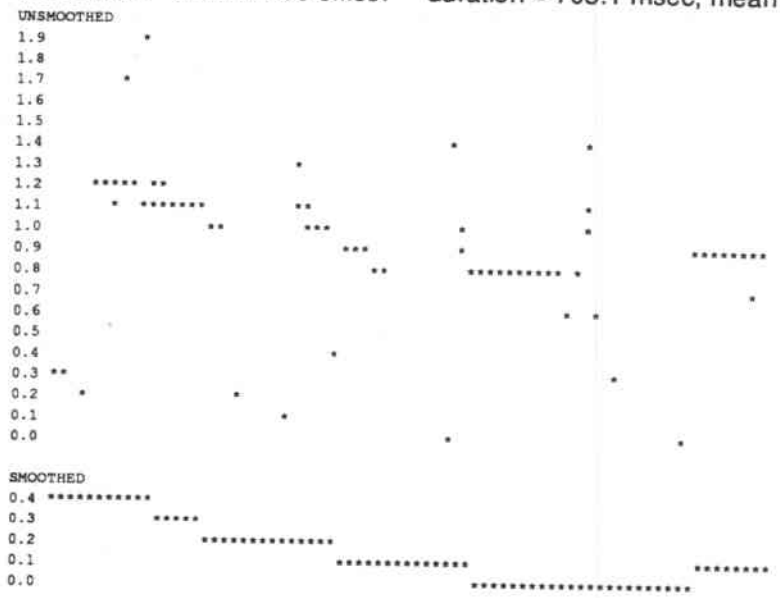


FILE: s.4.0 "That's okay." duration = 594.4 msec, mean freq = 183.3

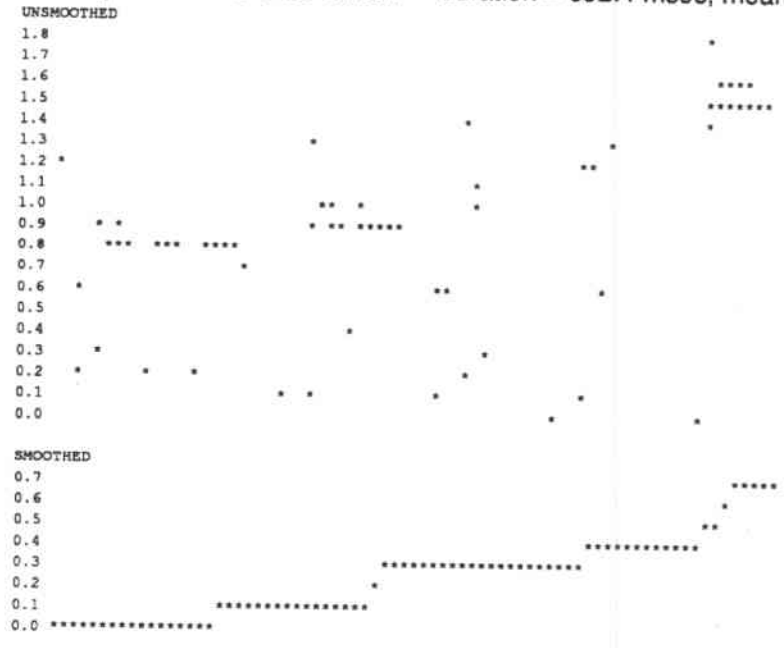
FILE: q.4.0 "That's okay?" duration = 639.7 msec, mean freq = 221.9



FILE: s.5.0 "Conference office." duration = 703.1 msec, mean freq = 194.3



FILE: q.5.0 "Conference office?" duration = 692.4 msec, mean freq = 225.8



Bibliography

Abney, S. P. 1991. Syntactic affixation and performance structures. In *Views on Phrase Structure*, ed. D. Bouchard and K. Leffel. Kluwer Academic Publishers.

Abney, S. P. 1991. Parsing by chunks. In *Principle-Based Parsing*, ed. R. Berwick, S. P. Abney, and C. Tenny. Kluwer Academic Publishers.

Allen, R. B. 1991. Knowledge representation and information retrieval with simple recurrent networks. In *AAAI Spring Symposium on Connectionist Natural Language Processing (Working Notes)*.

Allen, R. B. 1990. Connectionist language users. *Connection Science* 2: 279–311.

Barnard, E., R. A. Cole, M. P. Vea, F. A. Alleva. 1991. Pitch Detection with a Neural-Net Classifier. *IEEE Transactions on Signal Processing* 39(2): 298–307.

Berg, G. 1991. *Learning Recursive Phrase Structure: Combining the Strengths of PDP and X-Bar Syntax*. Technical Report 91-5, Department of Computer Science, Albany State University of New York.

Bruce, B. 1975. Case systems for natural language. *Artificial Intelligence* 6(4): 327–60.

Cleeremans, A., D. Servan-Schreiber, and J. L. McClelland. 1989. Finite state automata and simple recurrent networks. *Neural Computation* 1: 372–81.

Charniak, E. and E. Santos. 1987. A connectionist context-free parser which is not context-free but then it is not really connectionist either. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 70–7.

Church, K., W. Gale, P. Hanks, and D. Hindle. 1991. Parsing, word associations, and typical predicate-argument relations. In *Current Issues in Parsing Technologies*, ed. M. Tomita, 103–12. Norwell, MA: Kluwer Academic Publishers.

Cottrell, G.W. 1984. A model of lexical access of ambiguous words. In *Proceedings of the National Conference on Artificial Intelligence AAAI-84*.

Cottrell, G. W. 1989. *A Connectionist Approach to Word Sense Disambiguation*. San Mateo, CA: Morgan Kaufmann.

Cottrell, G. W. and S. Small. 1983. A connectionist scheme for modeling word sense disambiguation. *Cognition and Brain Theory* 6(1).

Dolan, C. P. 1989. *Tensor Manipulation Networks: Connectionist and Symbolic Approaches to Comprehension, Learning, and Planning*. PhD Thesis, Computer Science Department, University of California, Los Angeles. Available as Technical Report CSD–890030.

Elman, J. L. 1988. *Finding Structure in Time*. Tech. Rep. 8801, Center for Research in Language, University of California, San Diego.

Elman, J. L. 1989. *Representation and Structure in Connectionist Models*. Tech. Rep. 8903, Center for Research in Language, University of California, San Diego.

Elman, J. L. 1990. Finding structure in time. *Cognitive Science* 14(2): 179–212.

Fahlman, S. E. 1988. Faster-learning variations on back-propagation: An empirical study. In *Proceedings of the 1988 Connectionist Models Summer School*, ed. D. S. Touretzky, G. Hinton, and T. Sejnowski, 38–51. San Mateo, CA: Morgan Kaufmann.

Fahlman, S. E. and C. Lebiere. 1990. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, ed. D. S. Touretzky, 524–32. San Mateo, CA: Morgan Kaufmann.

Fanty, M. 1985. *Context Free Parsing in Connectionist Networks*. Tech. Rep. TR174, Computer Science Department, University of Rochester.

Fanty, M. 1986. Context-free parsing with connectionist networks. In *AIP Conference Proceedings number 151*, ed. J. S. Denker. New York: American Institute of Physics.

Fillmore, C. 1968. The case for case. In *Universals in Linguistic Theory*, ed. E. Bach and R. T. Harms. New York: Holt.

Fujisaki, T., F. Jelinek, J. Cocke, E. Black, and T. Nishino. A probabilistic parsing method for sentence disambiguation. In *Current Issues in Parsing Technologies*, ed. M. Tomita, 139–52. Norwell, MA: Kluwer Academic Publishers.

Gallant, S. 1990. *A Practical Approach for Representing Context and for Performing Word Sense Disambiguation Using Neural Networks*. Tech. Rep. NU–CCS–90–5, College of Computer Science, Northeastern University.

Gorin, A. L., S. E. Levinson, A. Ljolje, A. N. Gertner, E. R. Goldman, and L. Miller. 1990. On adaptive acquisition of language. In *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Hanson, S. J. and J. Kegl. 1987. PARSNIP: A connectionist network that learns natural language grammar from exposure to natural language sentences. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 106–119.

Hausser, R. 1989. *Computation of Language: An Essay on Syntax, Semantics, and Pragmatics in Natural Man-Machine Communication*. Berlin: Springer-Verlag.

Howells, T. 1988. VITAL: A connectionist parser. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, 18–25.

Huber, D. 1989. Parsing speech for structure and prominence. In *Proceedings of the International Workshop on Parsing Technologies*, 115–25. Available through the School of Computer Science, Carnegie Mellon University.

Jain, A. N. 1989. *A Connectionist Architecture for Sequential Symbolic Domains*. Tech. Rep. CMU–CS–89–187, School of Computer Science, Carnegie Mellon University.

Jain, A. N. and A. Waibel. 1989. A connectionist parser aimed at spoken language. In *Proceedings of the International Workshop on Parsing Technologies*, 221–9. Available through the School of Computer Science, Carnegie Mellon University.

Jain, A. N. and A. Waibel. 1990. Incremental parsing by modular recurrent connectionist networks. In *Advances in Neural Information Processing Systems 2*, ed. D. S. Touretzky. San Mateo, CA: Morgan Kaufmann.

Jain, A. N. and A. Waibel. 1991. Parsing in connectionist networks. In *Current Issues in Parsing Technologies*, ed. M. Tomita. Kluwer Academic Publishers.

Jain, A. N. and A. Waibel. 1990. Robust connectionist parsing of spoken language. In *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing*

Jain, A. N. 1991. Parsing complex sentences with structured connectionist networks. *Neural Computation* 3(1): 110–20.

Jain, A. N., A. E. McNair, A. Waibel, H. Saito, A. G. Hauptmann, and J. Tebelskis. 1991. Connectionist and symbolic processing in speech-to-speech translation: The JANUS system. In *Proceedings of the MT Summit III Machine Translation Conference.*

Jordan, M. I. 1986. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 531–46.

Kitano, H., H. Iida, T. Mitamura, and H. Tomabechi. 1989. An integrated discourse understanding model for an interpreting telephony under the direct memory access paradigm. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Kwasny, S. C. and Faisal, K. A. 1990. Connectionism and determinism in a syntactic parser. *Connection Science* 2(1): 63–82.

Marcus, M. 1980. *A Theory of Syntactic Recognition for Natural Language*. Cambridge, MA: The MIT Press.

McClelland, J. L. and A. H. Kawamoto. 1986. Mechanisms of sentence processing: Assigning roles to constituents. In *Parallel Distributed Processing*, vol. 2, ed. J. L. McClelland and D. E. Rumelhart. The MIT Press.

Miikkulainen, R. and M. G. Dyer. 1989. Encoding input/output representations in connectionist cognitive systems. In *Proceedings of the 1988 Connectionist Models Summer School*, ed. D. Touretzky, G. Hinton, and T. Sejnowski, 347–56. San Mateo, CA: Morgan Kaufmann.

Miikkulainen, R. 1990. *DISCERN: A Distributed Artificial Neural Network Model of Script Processing and Memory*. PhD thesis, Computer Science Department, University of California, Los Angeles.

Miikkulainen, R. 1990. A PDP architecture for processing sentences with relative clauses. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*.

Nakagawa, H. and T. Mori. 1988. A parser based on a connectionist model. In *COLING '88*, 454–8. Budapest.

Nijholt, A. 1990. Meta-parsing in neural networks. In *Cybernetics and Systems '90*, 969–76. Singapore: World Scientific Publishing.

Pierrehumbert, J. B. 1975. *The Phonology and Phonetics of English Intonation*. PhD Thesis, Massachusetts Institute of Technology.

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

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing*, vol. 1, ed. D. E. Rumelhart and J. L. McClelland, 318–62. The MIT Press.

Saito, H., and M. Tomita. 1988. Parsing noisy sentences. In *Proceedings of INFO JAPAN '88: International Conference of the Information Processing Society of Japan*, 553–59.

St. John, M. F. 1990. *The Story Gestalt—Text Comprehension by Cue-Based Constraint Satisfaction*. PhD Thesis. Department of Psychology, Carnegie Mellon University.

St. John, M. F., and J. L. McClelland. 1990. Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence* 46: 217–57.

Santos, S. 1989. A massively parallel self-tuning context-free parser. In *Advances in Neural Information Processing Systems 1*, ed. D. S. Touretzky. San Mateo, CA: Morgan Kaufmann.

Selman, B. 1985. *Rule-Based Processing in a Connectionist System for Natural Language Understanding*. Ph.D. Thesis, University of Toronto. Available as Tech. Rep. CSRI-168.

Selman, B. and G. Hirst. 1985. A rule-based connectionist parsing system. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, 212–21.

Seneff, S. 1989. TINA: A probabilistic syntactic parser for speech understanding systems. In *Proceedings of the 1989 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 711–4.

Steedman, M. 1991. Parsing spoken language using combinatory grammars. In *Current Issues in Parsing Technologies*, ed. M. Tomita, 113–26. Norwell, MA: Kluwer Academic Publishers.

Tebelskis, J., A. Waibel, B. Petek, and O. Schmidbauer. 1991. Continuous speech recognition using linked predictive neural networks. In *Proceedings of the 1991 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Tomita, M. 1985. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Norwell, MA: Kluwer Academic Publishers.

Tomita, M. (ed.). 1991. *Generalized LR Parsing*. Norwell, MA: Kluwer Academic Publishers.

Tomita, M. and J. G. Carbonell. 1987. *The Universal Parser Architecture for Knowledge-Based Machine Translation*. Technical Report CMU–CMT–87–01, Center for Machine Translation, Carnegie Mellon University.

Tomita, M. and E. Nyberg. 1988. *Generation Kit and Transformation Kit*. Technical Report CMU–CMT–88–MEMO, Center for Machine Translation, Carnegie Mellon University.

Touretzky, D. S. 1991. Connectionism and Compositional Semantics. In *Advances in Connectionist and Neurally Oriented Computation, Volume 1: High-Level Connectionist Models*, ed. J. A. Barnden and J. B. Pollack. Norwood, NJ: Ablex. Also available as Technical Report CMU–CS–89–147, School of Computer Science, Carnegie Mellon University.

Waibel, A. 1988. *Prosody and Speech Recognition*. Pitman/Morgan Kaufmann.

Waibel, A., T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37(3):328–39.

Waibel, A., A. N. Jain, A. E. McNair, H. Saito, A. G. Hauptmann, and J. Tebelskis. 1991. JANUS: A speech-to-speech translation system using connectionist and symbolic processing strategies. In *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 793–6.

Waltz, D. and J. Pollack. 1985. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science* 9:51–74.

Wang, Y. Y. and A. Waibel. 1991. A connectionist model for dialog processing. In *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 785–8.

Young, S. R., A. G. Hauptmann, W. H. Ward, E. T. Smith, and P. Werner. 1989. High level knowledge sources in usable speech recognition systems. *Communications of the ACM* 32(2):183–93.

Zhou, L. 1991. *Speaker-Independent Neural Network Pitch Tracker with Telephone Bandwidth Speech for Computer Speech Recognition*. MS Thesis, Oregon Graduate Institute of Science and Technology.

Zue, V., J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. 1990. The VOYAGER speech understanding system: Preliminary development and evaluation. In *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Zue, V., J. Glass, D. Goodine, M. Phillips, and S. Seneff. 1990. The SUMMIT speech recognition system: Phonological modeling and lexical access. In *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Author Index

Subject Index